The above discussion is not rigorous but leads to the correct expressions for **E** and **B**. We suggest that you accept (10.40) and (10.42) in the same spirit as you accepted Coulomb's law and the Biot–Savart law. All of classical electrodynamics can be reduced to (10.40) and (10.42) if we assume that the sources of all fields are charges, and all electric currents are due to the motion of charged particles. Note that (10.40) and (10.42) are consistent with the special theory of relativity and reduce to known results in the limit of stationary charges and steady currents.

Although (10.40) and (10.42) are deceptively simple (we do not even have to solve any differential equations), it is difficult to calculate the fields analytically even if the position of a charged particle is an analytic function of time. The difficulty is that we must find the retarded time t_{ret} from (10.32) for each observation position **R** and time t. For example, consider a charged particle whose motion is sinusoidal, that is, $x(t_{\text{ret}}) = A\cos\omega t_{\text{ret}}$. To calculate the fields at the position $\mathbf{R} = (X, Y, Z)$ at time t, we need to solve the following transcendental equation for t_{ret}:

$$t_{\text{ret}} = t - \frac{r_{\text{ret}}}{c} = t - \frac{1}{c}\sqrt{(X - A\cos^2\omega t_{\text{ret}})^2 + Y^2 + Z^2}. \tag{10.43}$$

The solution of (10.43) can be expressed as a root finding problem for which we need to find the zero of the function $f(t_{\text{ret}})$:

$$f(t_{\text{ret}}) = t - t_{\text{ret}} - \frac{r_{\text{ret}}}{c}. \tag{10.44}$$

There are various ways of finding the solution for the retarded time. For example, if the motion of the charges is given by an analytic expression, we can employ *Newton's method* or the *bisection method*. Because we will store the path of the charged particle, we use a simple method that looks for a change in the sign of the function $f(t_{\text{ret}})$ along the path. First find a value t_a such that $f(t_a) > 0$ and another value t_b such that $f(t_b) < 0$. Because $f(t_{\text{ret}})$ is continuous, there is a value of t_{ret} in the interval $t_a < t_{\text{ret}} < t_b$ such that $f(t_{\text{ret}}) = 0$. This technique is used in the `RadiatingCharge` class shown in Listing 10.6. Note that the particle's path is a sinusoidal oscillation specified in method `evaluate`. The name `evaluate` is used because `RadiatingCharge` implements the `Function` interface, which requires an `evaluate` method. What is the maximum velocity for a particle that moves according to this function?

Listing 10.6 The `RadiatingCharge` class computes the radiating electric and magnetic fields using Liénard–Wiechert potentials.

```
package org.opensourcephysics.sip.ch10;
import java.awt.Graphics;
import org.opensourcephysics.display.*;
import org.opensourcephysics.numerics.*;

public class RadiatingCharge implements Drawable, Function {
   Circle circle = new Circle(0, 0, 5);
   double t = 0;                                  // time
   double dt = 0.5;                               // time step
   // current number of points in storage
   int numPts = 0;
   double[][] path = new double[3][1024]; // storage for t,x,y
   double[] r = new double[2];
```



```
   double[] v = new double[2];
   double[] u = new double[2];
   double[] a = new double[2];
   // maximum velocity for charge in units where c = 1
   double vmax;

   public RadiatingCharge() {
      resetPath();
   }

   private void resizePath() {
      int length = path[0].length;
      if(length>32768) { // drop half the points
         System.arraycopy(path[0], length/2, path[0], 0, length/2);
         System.arraycopy(path[1], length/2, path[1], 0, length/2);
         System.arraycopy(path[2], length/2, path[2], 0, length/2);
         numPts = length/2;
         return;
      }
      double[][] newPath = new double[3][2*length]; // new path
      System.arraycopy(path[0], 0, newPath[0], 0, length);
      System.arraycopy(path[1], 0, newPath[1], 0, length);
      System.arraycopy(path[2], 0, newPath[2], 0, length);
      path = newPath;
   }

   void step() {
      t += dt;
      if(numPts>=path[0].length) {
         resizePath();
      }
      path[0][numPts] = t;
      path[1][numPts] = evaluate(t); // x position of charge
      path[2][numPts] = 0;
      numPts++;
   }

   void resetPath() {
      numPts = 0;
      t = 0;
      path = new double[3][1024];     // storage for t,x,y
      path[0][numPts] = t;
      path[1][numPts] = evaluate(t); // x position of charge
      path[2][numPts] = 0;
      numPts++; // initial position has been added
   }

   void electrostaticField(double x, double y, double[] field) {
      double dx = x-path[1][0];
      double dy = y-path[2][0];
      double r2 = dx*dx+dy*dy;
      double r3 = r2*Math.sqrt(r2);
      double ex = dx/r3;
      double ey = dy/r3;
      field[0] = ex;
      field[1] = ey;
```