One consequence of (9.29) is that there are $\omega_Q/\omega_0 + 1$ independent coefficients for a_k (including a_0) and ω_Q/ω_0 independent coefficients for b_k, a total of $N + 1$ independent coefficients. (Recall that $\omega_Q/\omega_0 = N/2$, where $\omega_0 = 2\pi/T$ and $T = N\Delta$.) Because $\sin \omega_Q t = 0$ for all values of t that are multiples of Δ, we have $b_{N/2} = 0$ from (9.26b). Consequently, there are $N/2 - 1$ values for b_k, and hence a total of N Fourier coefficients that can be computed. This conclusion is reasonable because the number of meaningful Fourier coefficients should be the same as the number of data points.

The `Analyze` class computes the Fourier coefficients a_k and b_k of a function $f(t)$ defined between $t = 0$ and $t = T$ at intervals of Δ. To compute the coefficients, we do the integrals in (9.26) numerically using the simple rectangular approximation (see Section 11.1):

$$a_k \approx \frac{2\Delta}{T} \sum_{i=0}^{N-1} f(t_i) \cos \omega_k t_i \tag{9.30a}$$

$$b_k \approx \frac{2\Delta}{T} \sum_{i=0}^{N-1} f(t_i) \sin \omega_k t_i, \tag{9.30b}$$

where the ratio $2\Delta/T = 2/N$.

Listing 9.5 The `Analyze` class calculates the Fourier coefficients of a function.

```
package org.opensourcephysics.sip.ch09;
import org.opensourcephysics.numerics.Function;

public class Analyze {
   Function f;
   double period, delta;
   double omega0;
   int N;

   Analyze(Function f, int N, double delta) {
      this.f = f;
      this.delta = delta;
      this.N = N;
      period = N*delta;
      omega0 = 2*Math.PI/period;
   }

   double getSineCoefficient(int n) {
      double sum = 0;
      double t = 0;
      for(int i = 0;i<N;i++) {
         sum += f.evaluate(t)*Math.sin(n*omega0*t);
         t += delta;
      }
      return 2*delta*sum/period;
   }

   double getCosineCoefficient(int n) {
      double sum = 0;
      double t = 0;
      for(int i = 0;i<N;i++) {
```



```
         sum += f.evaluate(t)*Math.cos(n*omega0*t);
         t += delta;
      }
      return 2*delta*sum/period;
   }
}
```

The `AnalyzeApp` program reads a function and creates an `Analyze` object to plot the Fourier coefficients a_k and b_k versus k. Note how we use a *parser* to convert a string of characters into a function using a `ParsedFunction` object. Because typing mistakes are common when entering mathematical expressions, we return from the `calculate` method if a syntax error is detected.

Listing 9.6 A program that analyzes the Fourier components of a function.

```
package org.opensourcephysics.sip.ch09;
import java.awt.*;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.numerics.*;

public class AnalyzeApp extends AbstractCalculation {
   PlotFrame frame = new PlotFrame("frequency", "coefficients",
                     "Fourier analysis");

   public AnalyzeApp() {
      frame.setMarkerShape(0, Dataset.POST);
      // semitransparent red
      frame.setMarkerColor(0, new Color(255, 0, 0, 128));
      frame.setMarkerShape(1, Dataset.POST);
      // semitransparent blue
      frame.setMarkerColor(1, new Color(0, 0, 255, 128));
      frame.setXYColumnNames(0, "frequency", "cos");
      frame.setXYColumnNames(1, "frequency", "sin");
   }

   public void calculate() {
      double delta = control.getDouble("delta");
      int N = control.getInt("N");
      int numberOfCoefficients = control.getInt("# of coefficients");
      String fStr = control.getString("f(t)");
      Function f = null;
      try {
         f = new ParsedFunction(fStr, "t");
      } catch(ParserException ex) {
         control.println("Error parsing function string: "+fStr);
         return;
      }
      Analyze analyze = new Analyze(f, N, delta);
      double f0 = 1.0/(N*delta);
      for(int i = 0;i<=numberOfCoefficients;i++) {
         frame.append(0, i*f0, analyze.getCosineCoefficient(i));
         frame.append(1, i*f0, analyze.getSineCoefficient(i));
      }
      // Data tables can be displayed using
```