```
      I3 = Iz;
      // orient top along y-axis
      setOrientation(new double[] {1/Math.sqrt(2),
           1/Math.sqrt(2), 0, 0});
      spaceView.initialize();
   }

   public void advanceTime() {
      super.advanceTime();
      spaceView.update();
   }

   void setBodyFrameOmega(double[] omega) {
      super.setBodyFrameOmega(omega);
      spaceView.initialize();
   }

   void computeBodyFrameTorque(double[] state) {
      // external force in space frame
      double[] vec = new double[] {0, 0, 1};
      RigidBodyUtil.spaceToBody(state, vec);
      t1 = -vec[1]; // torque components declared in RigidBody
      t2 = vec[0];
      t3 = 0;
   }
}
```

The visualization of a spinning top can take many forms. Listing 17.15 presets the spinning top model as a table-top gyroscope using cylinders and arrows. The shaft of the gyroscope is the body's axis of symmetry (the body frame's $\hat{3}$-axis) and draws a trace showing the gyroscope's precession. The spinning about the $\hat{3}$-axis might appear to be incorrect if the animation step is too large. This aliasing effect is often seen in movies showing carriage wheel spokes rotating backward to the direction of travel.

Listing 17.15 The `SpinningTopSpaceView` class models a symmetric body rotating about the axis of symmetry.

```
package org.opensourcephysics.sip.ch17;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.display3d.simple3d.*;
import org.opensourcephysics.numerics.Transformation;

public class SpinningTopSpaceView {
   Group topGroup = new Group();
   Element shaft = new ElementCylinder();
   Element disk = new ElementCylinder();
   Element base = new ElementCylinder();
   Element post = new ElementCylinder();
   Element orientation = new ElementArrow();
   ElementTrail orientationTrace = new ElementTrail();
   Display3DFrame frame = new Display3DFrame("Space View");
   SpinningTop rigidBody;

   public SpinningTopSpaceView(SpinningTop rigidBody) {
      this.rigidBody = rigidBody;
      frame.setSize(600, 600);
```



```
      double d = 4;
      frame.setPreferredMinMax(-d, d, -d, d, -d, d);
      frame.setDecorationType(VisualizationHints.DECORATION_AXES);
      orientation.getStyle().setFillColor(java.awt.Color.RED);
      orientationTrace.getStyle().setLineColor(java.awt.Color.BLACK);
      base.setSizeXYZ(2, 2, 0.15);
      base.getStyle().setResolution(new Resolution(4, 12, 1));
      base.getStyle().setFillColor(java.awt.Color.RED);
      base.setZ(-3);
      post.setSizeXYZ(0.2, 0.2, 3);
      post.getStyle().setResolution(new Resolution(2, 10, 15));
      post.setZ(-1.5); // shift by half the length
      post.getStyle().setFillColor(java.awt.Color.RED);
      shaft.setSizeXYZ(0.2, 0.2, 3);
      shaft.setXYZ(0, 0, 1.5);
      shaft.getStyle().setResolution(new Resolution(1, 10, 15));
      disk.setSizeXYZ(1.75, 1.75, 0.25);
      disk.setXYZ(0, 0, 2.0);
      disk.getStyle().setResolution(new Resolution(4, 12, 1));
      topGroup.addElement(shaft);
      topGroup.addElement(disk);
      topGroup.setTransformation(rigidBody.getTransformation());
      frame.addElement(base);
      frame.addElement(post);
      frame.addElement(orientation);
      frame.addElement(orientationTrace);
      frame.addElement(topGroup);
   }

   void initialize() {
      // dimension of ellipsoid is inverse to inertia
      double dx = 1/Math.sqrt(rigidBody.I1);
      double dy = 1/Math.sqrt(rigidBody.I2);
      double dz = 1/Math.sqrt(rigidBody.I3);
      // bounding dimension
      double scale = Math.max(Math.max(4*dx, 4*dy), 4*dz);
      frame.setPreferredMinMax(-scale, scale, -scale, scale,
                               -scale, scale);
      orientationTrace.clear();
      update();
   }

   void update() {
      Transformation transformation = rigidBody.getTransformation();
      topGroup.setTransformation(transformation);
      double s = 1.5*shaft.getSizeZ();
      double[] vec = topGroup.toSpaceFrame(new double[] {0, 0, 1});
      orientation.setSizeXYZ(s*vec[0], s*vec[1], s*vec[2]);
      orientationTrace.addPoint(s*vec[0], s*vec[1], s*vec[2]);
      frame.render();
   }
}
```

Problem 17.18 Uniform precession

If the angular velocity about the axis of symmetry is large, there are two possible rates of steady precession. These precession rates $\dot{\phi}$ are approximately