Problem 17.14 Torque-free rotation about a principal axis

(a) Add a second visualization showing the motion as viewed in the noninertial body frame.

(b) If the angular velocity vector coincides with a body's principal axis, the angular momentum and the angular velocity coincide. The body should then rotate steadily about the corresponding principal axis because the net torque is zero and the angular momentum is constant. Does the simulation show this result for all three axes if the moments of inertia are unequal? Perturb the angular velocity. Are rotations about the three principal axes stable or unstable? Check each axis. Repeat this simulation with a different set of moments of inertia. ■

Problem 17.15 Torque-free rotation of a symmetrical body

(a) Add a plot showing the time dependence of the angular velocity components $\omega_1(t)$, $\omega_2(t)$, and $\omega_3(t)$ to the FeynmanPlateApp class.

(b) Verify that $\omega_3(t)$ is constant if $I_1 = I_2 = I_s$.

(c) Verify that $\omega_1(t)$ and $\omega_2(t)$ exhibit an out-of-phase sinusoidal dependence if $I_1 = I_2$.

(d) If α denotes the angle between the body frame $\hat{3}$-axis and the axis of rotation, verify that

$$\Omega = \left(\frac{I_3}{I_s} - 1\right)\omega\cos\alpha, \tag{17.45}$$

where Ω is the angular velocity of the $\boldsymbol{\omega}$ vector's precession about the body frame's $\hat{3}$-axis. ■

Problem 17.16 Free rotation of a thin cylindrical rod

Model the free rotation of a thin cylindrical rod. Show that $\boldsymbol{\omega}$ precesses about the axis of the rod. Why does $\boldsymbol{\omega}$ precess? Why does **L** not precess? How is the frequency of precession related to the moment of inertia of the rod? ■

17.7 ■ RIGID BODY MODEL

As seen in Problem 17.13, the simple rigid body algorithm presented in Section 17.6 does a good job of conserving energy and angular momentum but should be improved if our goal is to compute accurate trajectories over long times. To use a higher-order differential equation solver, we need to obtain an expression for the second derivative of the quaternion by eliminating the angular velocities from Euler's equation of motion (17.23). The resulting quaternion acceleration is the rate for the quaternion velocity in a differential equation solver.

If we start with

$$\frac{d}{dt}\hat{q}^*\dot{\hat{q}} = \dot{\hat{q}}^*\dot{\hat{q}} + \hat{q}^*\ddot{\hat{q}}, \tag{17.46}$$

multiply by $\hat{q}$, and rearrange the terms, we obtain

$$\ddot{\hat{q}} = \hat{q}\left(\frac{d}{dt}\hat{q}^*\dot{\hat{q}} - \dot{\hat{q}}^*\dot{\hat{q}}\right). \tag{17.47}$$

Some messy algebra shows that (17.47) can be written in matrix form as

$$\begin{bmatrix} \ddot{q}_0 \\ \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = \frac{1}{2}W^T \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \\ -2\Sigma\dot{q}_m^2 \end{bmatrix}. \tag{17.48}$$

The dynamical behavior of a rigid body can now be expressed in terms of a quaternion state vector $(q_0, \dot{q}_0, q_1, \dot{q}_1, q_2, \dot{q}_2, q_3, \dot{q}_3, t)$. The following steps summarize the algorithm for computing the rate for this state.

1. Use the state vector to compute the angular velocity $(\omega_1, \omega_2, \omega_3)$ in the body frame using (17.40).
2. Project the external torques onto the body frame and compute $\dot{\boldsymbol{\omega}}$ using Euler's equation (17.23).
3. Compute the quaternion acceleration using (17.48).

These steps are implemented in the RigidBody class shown in Listing 17.12.

Listing 17.12 The RigidBody class solves Euler's equation of motion for a rotating rigid body using quaternions.

```
package org.opensourcephysics.sip.ch17;
import org.opensourcephysics.numerics.*;

public class RigidBody implements ODE {
   Quaternion rotation = new Quaternion(1, 0, 0, 0);
   double[] state = new double[9];
   ODESolver solver = new RK45MultiStep(this);
   double[] omegaBody = new double[3];          // body frame omega
   // body frame angular momentum
   double[] angularMomentumBody = new double[3];
   // principal moments of inertia
   double I1 = 1, I2 = 1, I3 = 1;
   // angular acceleration in the body frame
   double wxdot = 0, wydot = 0, wzdot = 0;
   // torque in the body frame
   double t1 = 0, t2 = 0, t3 = 0;

   void setOrientation(double[] q) {
      double norm = Math.sqrt(q[0]*q[0]+q[1]*q[1]+q[2]*q[2]+q[3]*q[3]);
      state[0] = q[0]/norm;
      state[2] = q[1]/norm;
      state[4] = q[2]/norm;
      state[6] = q[3]/norm;
      rotation.setCoordinates(state[0], state[2], state[4], state[6]);
   }
```