17.5 ■ QUATERNION ARITHMETIC

The rotation of an arbitrary two-dimensional vector $\mathbf{A} = a_x\hat{x} + a_y\hat{y}$ by an angle θ can be reformulated using complex numbers as

$$a' = ae^{i\theta} = e^{i\theta/2}ae^{i\theta/2}, \tag{17.25}$$

where the vector $\mathbf{A}$ is expressed as a complex number $a = a_x + ia_y$. The real and imaginary components correspond to the vector components. This idea can be extended to three dimensions using quaternions. A quaternion can be represented in terms of real and *hypercomplex* numbers i, j, and k as

$$\hat{q} = q_0 + iq_1 + jq_2 + kq_3 = (q_0, q_1, q_2, q_3), \tag{17.26}$$

where the hypercomplex numbers obey Hamilton's rules

$$i^2 = j^2 = k^2 = ijk = -1. \tag{17.27}$$

Similar to imaginary numbers, the quaternion conjugate is defined as

$$\hat{q}^* = q_0 - iq_1 - jq_2 - kq_3 = (q_0, -q_1, -q_2, -q_3). \tag{17.28}$$

Unlike imaginary numbers, quaternion multiplication does not commute and obeys the rules

$$ij = k, \quad jk = i, \quad ki = j, \quad ji = -k, \quad kj = -1, \quad ik = -j. \tag{17.29}$$

Although it would be a mistake to identify hypercomplex numbers with unit vectors in three-dimensional space (just as it would be a mistake to identify the imaginary number i with the y direction in a two-dimensional space), it is convenient to think of a quaternion as the sum of a scalar q_0 and a vector $\mathbf{q}$:

$$\hat{q} = q_0 + \mathbf{q} = (q_0, \mathbf{q}). \tag{17.30}$$

The quaternion is said to be *pure* if the scalar part is zero.

By using the above definitions, the product of two quaternions $\hat{p}$ and $\hat{q}$ can be shown to be

$$\hat{p}\hat{q} = (q_0, \mathbf{q})(p_0, \mathbf{p}) = q_0p_0 - \mathbf{q}\cdot\mathbf{p} + q_0\mathbf{p} + p_0\mathbf{q} + \mathbf{p}\times\mathbf{q}. \tag{17.31}$$

Note that except for the additional cross product term, quaternion multiplication is similar to complex multiplication, $(a_0, a_1)(b_0, b_1) = (a_0b_0 - a_1b_1, a_1b_0 + b_1a_0)$. The norm (length) of a quaternion is defined to be $|\hat{q}\hat{q}^*| = q_0q_0 + q_1q_1 + q_2q_2 + q_3q_3$.

Exercise 17.10 Quaternion multiplication
Show that Hamilton's rules for hypercomplex numbers in (17.27) lead to (17.31). ■

Euler has shown that the most general change of a rigid body with one point fixed is a rotation about a fixed axis as shown in Figure 17.5. Quaternions provide an elegant representation of both the rotation angle and the axis orientation. It can be shown (see

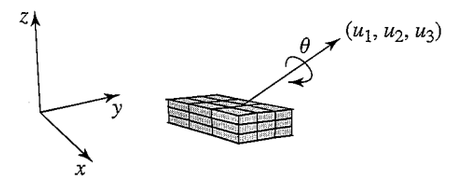

Figure 17.5 The most general change of a rigid body with one point fixed is a rotation about a fixed axis with direction cosines (u_1, u_2, u_3) by the angle θ.

Shoemake) that a rotation through an angle θ about an axis with direction cosines (u_1, u_2, u_3) can be represented as a unit quaternion with components

$$\hat{q} = \cos\frac{\theta}{2} + (iu_1 + ju_2 + ku_3)\sin\frac{\theta}{2}. \tag{17.32}$$

To stress the analogy with the complex exponential $e^{i\theta} = \cos\theta + i\sin\theta$, some textbooks represent this rotation through θ about the axes $\mathbf{u}$ using exponential notation $\hat{q} = e^{\mathbf{u}\theta/2}$.

Given a unit quaternion $\hat{q}$ that represents a rotation, how do we apply this rotation to an arbitrary vector $\mathbf{A}$? If we define a pure quaternion $\hat{a} = (0, a_x, a_y, a_z)$, it can be shown that

$$\hat{a}' = \hat{q}\hat{a}\hat{q}^*, \tag{17.33}$$

where the resulting quaternion $\hat{a}' = (0, a'_x, a'_y, a'_z)$ contains the components of the rotated vector $\mathbf{A}'$ (see Rapaport). Note the similarity to (17.25) if the quaternion is represented using exponential notation.

Exercise 17.11 Quaternion rotation

(a) Use the properties of the direction cosines to show that (17.32) defines a quaternion of unit length.
(b) Show that the length of the vector $\mathbf{A}$ does not change when using (17.33). ■

Bodies can be oriented using any representation of a rotation including quaternions, Euler angles, and rotation matrices. Because the quaternion representation does not involve trigonometric functions, it is very efficient for computing rigid body dynamics. To make it easy to create other representations of transformations, we define a `Transformation` interface in the `numerics` package. A concrete implementation of the interfaces based on rotation matrices is described in Appendix 17A. The quaternion implementation is similar and is available in the `numerics` package. Listing 17.9 shows how the quaternion implementation is used to rotate a `BoxWithArrows`. Because `BoxWithArrows` is a subclass of `Group` containing a box and three arrows, it is not listed.

Listing 17.9 A class that tests the quaternion representation of rotations.

```
package org.opensourcephysics.sip.ch17;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.numerics.Quaternion;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.display3d.simple3d.VisualizationHints;
```