

Chapter 4

Quaternions and Gibbs Vectors



While most readers are familiar with the rotation matrices presented in the previous chapter, rotation matrices are not the most convenient or efficient way to represent rotations. Euler had already realized that expressing a rotation with a vector parallel to the axis of rotation was more elegant than using a rotation matrix. And the mathematical work by Hamilton and Gibbs on alternative representations of rotations, which is presented in this chapter, prepared the way not only for an efficient representation of rotations, but for the whole modern vector calculus.

4.1 Representing Rotations by Vectors

Rotation matrices are not the most efficient way to describe a rotation: they have nine elements, yet only three are actually needed to uniquely characterize a rotation. Another disadvantage of describing 3-D rotations with rotation matrices is that the three axes of rotation, as well as the sequence of the rotations about these axes, have to be defined arbitrarily, with different sequences leading to different rotation angles. Euler’s rotational theorem (Euler 1775) states that a more efficient way to characterize a rotation is to use a vector: the axis of rotation is defined by the direction of the vector \mathbf{q} , and the rotation magnitude θ is defined by the vector length (Fig. 4.1). The orientation is defined by the right-hand-rule (Fig. 4.2). Such a vector has only three parameters, and no sequence of multiple rotations is involved.

Different conventions can be used to define the vector:

- “Euler vectors” $|\mathbf{q}| = \theta$ Sect. 4.2
- “Quaternions” $|\mathbf{q}| = \sin(\theta/2)$ Sect. 4.3
- “Gibbs vectors” $|\mathbf{q}| = \tan(\theta/2)$ Sect. 4.4

Rotation matrices are often an easy way to establish a correspondence between measured values (e.g., induction coil voltages, or images) and the orientation of an object relative to a given reference orientation. But for working with 3-D orientations and for calculations, quaternions or Gibbs vectors have proven to be more

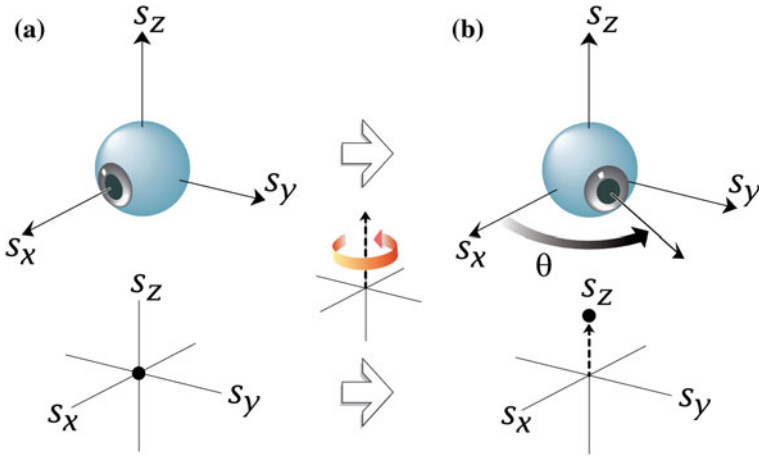


Fig. 4.1 Description of 3-D eye orientation by a vector: **a** The eye in the reference orientation (top) corresponds to the zero vector (bottom). **b** A different horizontal eye orientation (top) can be reached by rotating the eye from the reference orientation about the s_z -axis. This eye orientation is, thus, represented by a vector along the s_z -axis, with a length proportional to the angle of the rotation (bottom). Note that usually only the end-point of the vector describing the eye orientation is shown, not the whole vector

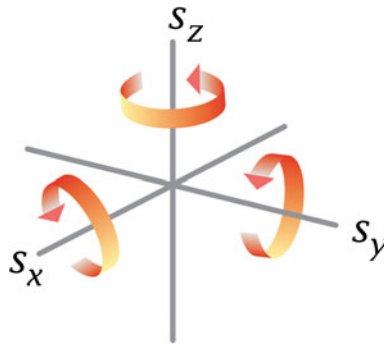


Fig. 4.2 According to the right-hand-rule *positive* rotations are yaw-rotations to the left (about s_z), pitch-rotations downward (about s_y), and roll-rotations clockwise as seen from the object (about s_x)

intuitive and efficient. They are nonredundant, using three parameters to describe the three degrees of freedom of rotations. And they do not require an arbitrarily chosen sequence of rotations, but describe orientation by a single rotation from the reference orientation to the current orientation. In addition, they form an intuitive way of parameterizing rotations by expressing them by their axis and size, allow for an easy combination of rotations, and are more accurate when used to integrate incremental changes in orientation over time.

4.2 Axis-Angle Euler Vectors

A vector \mathbf{x} can be rotated by an angle ρ about a vector \mathbf{n} through

$$\mathbf{R}(\mathbf{n}, \rho) \cdot \mathbf{x} = (\mathbf{n} \cdot \mathbf{x}) * \mathbf{n} + \mathbf{n} \times \mathbf{x} * \sin(\rho) - \mathbf{n} \times (\mathbf{n} \times \mathbf{x}) \cos(\rho) \quad (4.1)$$

or equivalently

$$\mathbf{R}(\mathbf{n}, \rho) \cdot \mathbf{x} = \mathbf{x} * \cos(\rho) + (1 - \cos(\rho)) * (\mathbf{n} \cdot \mathbf{x}) * \mathbf{n} + \sin(\rho) * \mathbf{n} \times \mathbf{x}. \quad (4.2)$$

The development of this parametrization of rotations can probably be attributed to (Rodrigues 1840), and Eq.(4.2) is, therefore, also called “Rodrigues’ rotation formula”. The representation of a rotation with an axis \mathbf{n} and an angle ρ is sometimes referred to as “axis-angle representation” of a rotation. And in honor of Euler’s rotation theorem (see p. 179), a vector with a direction \mathbf{n} and a length ρ is called “Euler vector”.

Note: While Euler vectors give a convenient representation of a rotation, no equation exists that allows to combine two Euler vectors. Therefore, practical implementations of rotations have to be based on rotation matrices, quaternions or Gibbs vectors (see below).

4.3 Quaternions

4.3.1 Background

The theory of quaternions was invented and developed by Hamilton in the mid-nineteenth century (Hamilton 1844). Hamilton realized that the complex numbers could be interpreted as points in a plane (see Fig. 3.3), and he was looking for a way to do the same for points in three-dimensional space. Points in space can be represented by their coordinates, which are triples of numbers. For many years he had known how to add and subtract triples of numbers. However, Hamilton had been stuck on the problem of multiplication and division for a long time. He could not figure out how to calculate the quotient of the coordinates of two points in space. Hamilton found that he could not accomplish this by using 3-component vectors, but had to use 4 components. He called these quadruples “quaternions”.

A detailed treatment of quaternions and their elegant mathematical properties can be found in mathematical texts (Brand 1948; Altmann 1986; Kuipers 1999), many papers on eye movements (Westheimer 1957; Tweed and Vilis 1987; Hepp et al. 1989; Tweed et al. 1990), and papers in more technical journals (Rooney 1977; Funda and Paul 1988). Recommendable introductions are also available on the Internet (see Appendix G).

A note for physicists, or for the mathematically more curious reader: quaternions are four-dimensional representations of *Clifford algebras* (see also Appendix A.4). The two-dimensional representations are the complex 2×2 -matrices, or “Pauli spin matrices” (SU2, or two-dimensional special unitary group). And the three-dimensional representations are the rotation matrices (SO3, or special three-dimensional orthogonal group). Especially in theoretical physics, the advantages of switching from 3-D representations over to 4-D quaternions can be massive (Girard 1984). For example, using Clifford algebra, the four Maxwell equations can be written in just one very compact, elegant equation (see also Appendix A.4):

$$\nabla F = \mu_0 J . \quad (4.3)$$

4.3.2 Quaternion Properties

The following description of quaternions will cover only their essential properties, and no mathematical proofs will be given.

A full quaternion $\tilde{\mathbf{q}}$ has four components, and is given by

$$\tilde{\mathbf{q}} = q_0 + (q_1 * \tilde{\mathbf{i}} + q_2 * \tilde{\mathbf{j}} + q_3 * \tilde{\mathbf{k}}) = q_0 + \mathbf{q} \cdot \mathbf{I}, \quad (4.4)$$

where $\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$, $\mathbf{I} = \begin{pmatrix} \tilde{\mathbf{i}} \\ \tilde{\mathbf{j}} \\ \tilde{\mathbf{k}} \end{pmatrix}$, and $(\tilde{\mathbf{i}}, \tilde{\mathbf{j}}, \tilde{\mathbf{k}})$ are defined by

$$\begin{array}{lll} \tilde{\mathbf{i}} \cdot \tilde{\mathbf{i}} = -1 & \tilde{\mathbf{j}} \cdot \tilde{\mathbf{j}} = -1 & \tilde{\mathbf{k}} \cdot \tilde{\mathbf{k}} = -1 \\ \tilde{\mathbf{i}} \cdot \tilde{\mathbf{j}} = \tilde{\mathbf{k}} & \tilde{\mathbf{j}} \cdot \tilde{\mathbf{k}} = \tilde{\mathbf{i}} & \tilde{\mathbf{k}} \cdot \tilde{\mathbf{i}} = \tilde{\mathbf{j}} \\ \tilde{\mathbf{j}} \cdot \tilde{\mathbf{i}} = -\tilde{\mathbf{k}} & \tilde{\mathbf{k}} \cdot \tilde{\mathbf{j}} = -\tilde{\mathbf{i}} & \tilde{\mathbf{i}} \cdot \tilde{\mathbf{k}} = -\tilde{\mathbf{j}} \end{array} \quad (4.5)$$

q_0 is often called the “scalar component” of the quaternion $\tilde{\mathbf{q}}$, and \mathbf{q} the “vector component” of $\tilde{\mathbf{q}}$. (Note that the quaternion is written as $\tilde{\mathbf{q}}$, and the quaternion *vector* as \mathbf{q} .)

With Eq. (4.5), one can show (see Appendix A.3) that the multiplication of two quaternions $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$, here denoted “ \circ ”, is given by

$$\tilde{\mathbf{q}} \circ \tilde{\mathbf{p}} = \sum_{i=0}^3 q_i I_i * \sum_{j=0}^3 p_j I_j = (q_0 p_0 - \mathbf{q} \cdot \mathbf{p}) + (q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p}) \cdot \mathbf{I}. \quad (4.6)$$

The right side of Eq. (4.6) is obtained by using the definitions in Eqs. (4.4) and (4.5). Similar to rotation matrices, the sequence of the quaternions in Eq. (4.6) is important, and the opposite sequence, $\tilde{\mathbf{p}} \circ \tilde{\mathbf{q}}$, would lead to a different quaternion.

The *inverse quaternion* is given by

$$\tilde{\mathbf{q}}^{-1} = \frac{q_0 - \mathbf{q} \cdot \mathbf{I}}{|\mathbf{q}|^2}. \quad (4.7)$$

The norm of a quaternion is given by the quadrature sum of all four components

$$|\tilde{\mathbf{q}}| = \sqrt{\sum_{i=0}^3 q_i^2}. \quad (4.8)$$

4.3.3 Interpretation of Quaternions

To interpret quaternions, it is helpful to group them into four classes:

- (1) Quaternions with the scalar component equal to 0 correspond to \mathbb{R}^3 , the space of three-dimensional vectors. (This group is sometimes also called “pure quaternions”.)
- (2) Quaternions with a zero vector component $\mathbf{0}$ correspond to the space of scalars, \mathbb{R} .
- (3) Unit quaternions, i.e., quaternions with $|\tilde{\mathbf{q}}| = 1$, correspond to $SO3$, the group of orthogonal matrices with a determinant of 1. Unit quaternions, sometimes also called “rotation quaternions”, can be used to describe rotations in space.
- (4) General quaternions with scalar and vector components unequal zero, with a norm unequal to 1. These quaternions describe a combination of a rotation and scaling of vectors (Rooney 1977). If the norm of the quaternion is > 1 , the objects are stretched; and if the norm is < 1 , objects are compressed.

4.3.4 Unit Quaternions

A quaternion describing a pure rotation in 3-D space is a “unit quaternion” and has a norm of $|\tilde{\mathbf{q}}| = 1$.

From Eq. (4.7), the inverse quaternion $\tilde{\mathbf{q}}^{-1}$ for a unit quaternion is given by

$$\tilde{\mathbf{q}}^{-1} = q_0 - \mathbf{q} \cdot \mathbf{I}, \quad (4.9)$$

Comparing Eqs. (4.5) and (3.9) to Fig. 3.3, which describes rotations in the complex plane, one can find a physical interpretation for $\tilde{\mathbf{i}}$, $\tilde{\mathbf{j}}$, and $\tilde{\mathbf{k}}$. A rotation of a complex number c by an angle ϕ is given by $c' = e^{j\phi} \cdot c$, where $j * j = -1$, and j can be interpreted as a vector pointing perpendicular out of the 2-D-plane. To

describe rotations in 3-D, we need three axes to rotate about: $\tilde{\mathbf{i}}$, $\tilde{\mathbf{j}}$, and $\tilde{\mathbf{k}}$. It can be shown that for a quaternion of the form

$$\tilde{\mathbf{q}} = \begin{pmatrix} 0 \\ \theta/2 * \mathbf{v} \end{pmatrix} \quad (4.10)$$

where $|\mathbf{v}| = 1$, the exponential of the quaternion is given by the unit quaternion

$$\exp(\tilde{\mathbf{q}}) = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2) * \mathbf{v} \end{pmatrix} \quad (4.11)$$

generalizing Eq. (3.5).

A unit quaternion describes a rotation by an angle θ around an axis described by the unit vector $\mathbf{n} = (n_i \tilde{\mathbf{i}}, n_j \tilde{\mathbf{j}}, n_k \tilde{\mathbf{k}})$

$$\tilde{\mathbf{q}} = \cos(\theta/2) + \sin(\theta/2)[n_i \tilde{\mathbf{i}} + n_j \tilde{\mathbf{j}} + n_k \tilde{\mathbf{k}}] = q_0 + \mathbf{q} \cdot \mathbf{I}, \quad (4.12)$$

where the orientation of \mathbf{n} describes the axis of rotation, as shown in Fig. 4.1b. The length of the vector component equals $\sin(\theta/2)$. The unit quaternion has the following properties:

$$|\tilde{\mathbf{q}}| = \sqrt{\cos^2(\theta/2) + \sin^2(\theta/2)} = 1 \quad (4.13a)$$

$$|\mathbf{q}| = \sqrt{q_1^2 + q_2^2 + q_3^2} = \sin(\theta/2) \quad (4.13b)$$

$$\mathbf{q} \parallel \mathbf{n} \quad (4.13c)$$

The $\theta/2$ property of rotation quaternions, i.e. the fact that the lengths of a unit quaternion vector is determined by *half* the rotation angle, $\theta/2$, can be explained by interpreting a rotation as a sum of two reflections, see Appendix Fig. A.4. Another way to explain it is by considering Eq. (4.14) discussed below. The rotation quaternion appears twice. This “double application” of $\theta/2$ leads to a final rotation by an angle θ .

Examples

40° yaw movement to the left: A yaw movement is a rotation about a vertical axis, so the quaternion vector has to be along the axis $\mathbf{n} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. The yaw rotation

to the left is positive (see Fig. 4.2). And since the magnitude of the rotation is 40°, the full quaternion is

$$\tilde{\mathbf{q}} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} \cos(40^\circ/2) \\ 0 \\ 0 \\ \sin(40^\circ/2) \end{pmatrix} = \begin{pmatrix} \cos(20^\circ) \\ 0 \\ 0 \\ \sin(20^\circ) \end{pmatrix}.$$

90° pitch rotation nose-up: A pitch movement is a rotation about the y -axis, so the quaternion vector has to be along the axis $\mathbf{n} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$. A pitch rotation nose-up is negative (Fig. 4.2). And since the magnitude of the rotation is 90°, the full quaternion is $\tilde{\mathbf{q}} = \begin{pmatrix} \cos(-45^\circ) \\ 0 \\ \sin(-45^\circ) \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(45^\circ) \\ 0 \\ -\sin(45^\circ) \\ 0 \end{pmatrix}$.

Relation to Rotation Matrix

The connection between a rotation quaternion $\tilde{\mathbf{q}}$ and a rotation matrix \mathbf{R} , both describing the rotation of a vector \mathbf{x} about an axis \mathbf{n} by an angle θ , can be derived from the definition of quaternions in Eqs. (4.4)–(4.13):

$$\begin{aligned} \tilde{\mathbf{x}}' &= \tilde{\mathbf{q}} \circ \tilde{\mathbf{x}} \circ \tilde{\mathbf{q}}^{-1} = \begin{pmatrix} 0 \\ \mathbf{x}' \end{pmatrix} \\ \mathbf{x}' &= \mathbf{R} \cdot \mathbf{x}. \end{aligned} \quad (4.14)$$

The proof of Eq. (4.14) is given in Appendix A.3.

$\tilde{\mathbf{x}}'$ in Eq. (4.14) is a full quaternion, but the scalar component evaluates to zero $q_0 = 0$. Hence, the rotation matrix \mathbf{R} corresponding to the quaternion $\tilde{\mathbf{q}}$ can be determined as

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad (4.15)$$

The inverse computation is given by

$$\mathbf{q} = 0.5 * \text{copysign} \left(\begin{array}{l} \sqrt{1 + R_{xx} - R_{yy} - R_{zz}}, R_{zy} - R_{yz} \\ \sqrt{1 - R_{xx} + R_{yy} - R_{zz}}, R_{xz} - R_{zx} \\ \sqrt{1 - R_{xx} - R_{yy} + R_{zz}}, R_{yx} - R_{xy} \end{array} \right), \quad (4.16)$$

where $\text{copysign}(x, y) = \text{sign}(y) * |x|$.

Sequential Rotations with Quaternions

Equation (4.14) is the quaternion equivalent of a matrix multiplication for rotation matrices \mathbf{R} . Therefore, a sequence of quaternion rotations is the same as the sequence of the corresponding rotation matrices. For combined rotations, care has to be taken with the sequence of quaternions: the same rules apply as for rotation matrices, which means that the first rotation acting on a vector is on the right-hand side of the quaternion multiplication in Eq. (4.6).

Relation to Sequential Rotations

The examples above show how quaternions are related to rotations about coordinate axes. Using the rules for quaternion multiplication Eq. (4.6), one can calculate the relationship between rotation angles for the nautical sequence and quaternions:

$$\tilde{q}_z(\theta_N) \circ \tilde{q}_y(\phi_N) \circ \tilde{q}_x(\psi_N) = \begin{pmatrix} \cos \frac{\theta_N}{2} * \cos \frac{\phi_N}{2} * \cos \frac{\psi_N}{2} + \sin \frac{\theta_N}{2} * \sin \frac{\phi_N}{2} * \sin \frac{\psi_N}{2} \\ \cos \frac{\theta_N}{2} * \cos \frac{\phi_N}{2} * \sin \frac{\psi_N}{2} - \sin \frac{\theta_N}{2} * \sin \frac{\phi_N}{2} * \cos \frac{\psi_N}{2} \\ \cos \frac{\theta_N}{2} * \sin \frac{\phi_N}{2} * \cos \frac{\psi_N}{2} + \sin \frac{\theta_N}{2} * \cos \frac{\phi_N}{2} * \sin \frac{\psi_N}{2} \\ \sin \frac{\theta_N}{2} * \cos \frac{\phi_N}{2} * \cos \frac{\psi_N}{2} - \cos \frac{\theta_N}{2} * \sin \frac{\phi_N}{2} * \sin \frac{\psi_N}{2} \end{pmatrix}. \quad (4.17)$$

For the Helmholtz sequence, this leads to

$$\tilde{q}_y(\phi_H) \circ \tilde{q}_z(\theta_H) \circ \tilde{q}_x(\psi_H) = \begin{pmatrix} \cos \frac{\theta_H}{2} * \cos \frac{\phi_H}{2} * \cos \frac{\psi_H}{2} - \sin \frac{\theta_H}{2} * \sin \frac{\phi_H}{2} * \sin \frac{\psi_H}{2} \\ \cos \frac{\theta_H}{2} * \cos \frac{\phi_H}{2} * \sin \frac{\psi_H}{2} + \sin \frac{\theta_H}{2} * \sin \frac{\phi_H}{2} * \cos \frac{\psi_H}{2} \\ \cos \frac{\theta_H}{2} * \sin \frac{\phi_H}{2} * \cos \frac{\psi_H}{2} + \sin \frac{\theta_H}{2} * \cos \frac{\phi_H}{2} * \sin \frac{\psi_H}{2} \\ \sin \frac{\theta_H}{2} * \cos \frac{\phi_H}{2} * \cos \frac{\psi_H}{2} - \cos \frac{\theta_H}{2} * \sin \frac{\phi_H}{2} * \sin \frac{\psi_H}{2} \end{pmatrix}. \quad (4.18)$$

And for the Euler sequence we get

$$\tilde{q}_z(\alpha_E) \circ \tilde{q}_x(\beta_E) \circ \tilde{q}_y(\gamma_E) = \begin{pmatrix} \cos \frac{\alpha_E}{2} * \cos \frac{\beta_E}{2} * \cos \frac{\gamma_E}{2} - \sin \frac{\alpha_E}{2} * \cos \frac{\beta_E}{2} * \sin \frac{\gamma_E}{2} \\ \cos \frac{\alpha_E}{2} * \sin \frac{\beta_E}{2} * \cos \frac{\gamma_E}{2} + \sin \frac{\alpha_E}{2} * \sin \frac{\beta_E}{2} * \sin \frac{\gamma_E}{2} \\ \sin \frac{\alpha_E}{2} * \sin \frac{\beta_E}{2} * \cos \frac{\gamma_E}{2} - \cos \frac{\alpha_E}{2} * \sin \frac{\beta_E}{2} * \sin \frac{\gamma_E}{2} \\ \cos \frac{\alpha_E}{2} * \cos \frac{\beta_E}{2} * \sin \frac{\gamma_E}{2} + \sin \frac{\alpha_E}{2} * \cos \frac{\beta_E}{2} * \cos \frac{\gamma_E}{2} \end{pmatrix}. \quad (4.19)$$

The inverse relationships, i.e., calculating the angles for the different rotation sequences, can be obtained by inserting the corresponding matrix elements from Eq. (4.15) into Eq. (3.24) for the nautical sequence, Eq. (3.27) for the Helmholtz sequence, and Eq. (3.30) for the Euler sequence.

4.4 Gibbs Vectors

4.4.1 Properties of Gibbs Vectors

Gibbs vectors are named after Josiah Willard Gibbs (1839–1903), the inventor of—among many other things—modern vector calculus. A *Gibbs vector*¹ \mathbf{r} corresponding to the rotation matrix \mathbf{R} is given by

$$\mathbf{r} = \frac{1}{1 + (R_{11} + R_{22} + R_{33})} * \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}. \quad (4.20)$$

From this one can show that

$$|\mathbf{r}| = \tan(\rho/2). \quad (4.21)$$

¹Some authors call Gibbs vectors “rotation vectors”.

The coefficients of the Gibbs vectors are sometimes referred to as “Rodrigues parameters” (Altmann 1986; Dai 2015).

Finding the relationship between Gibbs vectors and other descriptions of rotations, such as nautical angles, requires an equation for combined rotations with Gibbs vectors. Using Eqs. (4.6) and (4.21) gives

$$\mathbf{r}_q \circ \mathbf{r}_p = \frac{\mathbf{r}_q + \mathbf{r}_p + \mathbf{r}_q \times \mathbf{r}_p}{1 - \mathbf{r}_q \cdot \mathbf{r}_p}, \quad (4.22)$$

where \mathbf{r}_p is the first rotation (about a space-fixed axis parallel to \mathbf{r}_p), and \mathbf{r}_q the second rotation (about a space-fixed axis parallel to \mathbf{r}_q).

The Gibbs vector corresponding to the nautical angles in Eq. (3.23) can be obtained by combining three Gibbs vectors with Eq. (4.22). Denoting a Gibbs vector which describes a rotation about an axis \mathbf{n} by an angle θ with $\mathbf{r}(\mathbf{n}, \theta)$, this leads to

$$\begin{aligned} \mathbf{r} &= \mathbf{r}(\mathbf{e}_3, \theta_N) \circ \mathbf{r}(\mathbf{e}_y, \phi_N) \circ \mathbf{r}(\mathbf{e}_1, \psi_N) = \\ &= \frac{1}{1 - \tan(\frac{\theta_N}{2}) * \tan(\frac{\phi_N}{2}) * \tan(\frac{\psi_N}{2})} \begin{pmatrix} \tan \frac{\psi_N}{2} - \tan \frac{\theta_N}{2} * \tan \frac{\phi_N}{2} \\ \tan \frac{\phi_N}{2} + \tan \frac{\theta_N}{2} * \tan \frac{\psi_N}{2} \\ \tan \frac{\theta_N}{2} - \tan \frac{\phi_N}{2} * \tan \frac{\psi_N}{2} \end{pmatrix}, \quad (4.23) \end{aligned}$$

where $(\theta_A, \phi_A, \psi_A)$ are the nautical angles. For Helmholtz angles, the corresponding equation reads

$$\begin{aligned} \mathbf{r} &= \mathbf{r}(\mathbf{e}_y, \phi_H) \circ \mathbf{r}(\mathbf{e}_3, \theta_H) \circ \mathbf{r}(\mathbf{e}_1, \psi_H) = \\ &= \frac{1}{1 - \tan(\frac{\theta_H}{2}) * \tan(\frac{\phi_H}{2}) * \tan(\frac{\psi_H}{2})} \begin{pmatrix} \tan \frac{\psi_H}{2} + \tan \frac{\theta_H}{2} * \tan \frac{\phi_H}{2} \\ \tan \frac{\phi_H}{2} + \tan \frac{\theta_H}{2} * \tan \frac{\psi_H}{2} \\ \tan \frac{\theta_H}{2} - \tan \frac{\phi_H}{2} * \tan \frac{\psi_H}{2} \end{pmatrix}. \quad (4.24) \end{aligned}$$

Close to the reference position, the relations between nautical angles, Helmholtz angles, Gibbs vectors, and quaternions can be approximated by the simple formula

$$\begin{pmatrix} \psi \\ \phi \\ \theta \end{pmatrix}_{nautical} \approx \begin{pmatrix} \psi \\ \phi \\ \theta \end{pmatrix}_{Helmholtz} \approx 100 * \begin{pmatrix} r_1 \\ r_y \\ r_3 \end{pmatrix} \approx 100 * \begin{pmatrix} q_1 \\ q_y \\ q_3 \end{pmatrix} \quad (4.25)$$

where θ, ϕ, ψ are given in degrees.

Example

For example, with $\mathbf{r}_p = \begin{pmatrix} 0 \\ 0.176 \\ 0 \end{pmatrix}$ and $\mathbf{r}_q = \begin{pmatrix} 0 \\ 0 \\ 0.087 \end{pmatrix}$, Eq. (4.22) ($\mathbf{r}_q \circ \mathbf{r}_p$) would

describe a rotation of 20° about the horizontal axis \mathbf{s}_y , followed by a rotation of 10° about the space-fixed vertical axis \mathbf{s}_z . According to our discussion above of rotations of objects and coordinate systems, the same formula can also be interpreted as a first

rotation of 10° about the vertical axis \mathbf{b}_z , followed by a second rotation of 20° about the rotated, object-fixed axis \mathbf{b}_y - which corresponds to the horizontal and vertical rotation in a nautical gimbal.

4.4.2 Cascaded Rotations with Gibbs Vectors

For combined rotations, Gibbs vectors show the same sequences as the corresponding rotation matrices or quaternions. Using Gibbs vectors, Eq. (3.39) for combined eye-head movements can be expressed as

$$\mathbf{r}_{\text{gaze}} = \mathbf{r}_{\text{head}} \circ \mathbf{r}_{\text{eye}} . \quad (4.26)$$

This can be rearranged to yield the Gibbs vector describing the orientation of our object with respect to the reference frame (e.g. eye in head), \mathbf{r}_{eye} , as

$$\mathbf{r}_{\text{eye}} = \mathbf{r}_{\text{head}}^{-1} \circ \mathbf{r}_{\text{gaze}} . \quad (4.27)$$

The formula for the multiplication of two Gibbs vectors is given by Eq. (4.22), and the inverse of a Gibbs vector can be determined easily by $\mathbf{r}^{-1} = -\mathbf{r}$.

4.4.3 Gibbs Vectors and Their Relation to Quaternions

The Gibbs vector \mathbf{r} which corresponds to the quaternion $\tilde{\mathbf{q}}$ describing a rotation of θ about the axis \mathbf{n} is given by

$$\mathbf{r} = \frac{\mathbf{q}}{q_0} = \tan\left(\frac{\theta}{2}\right) \frac{\mathbf{q}}{|\mathbf{q}|} = \tan\left(\frac{\theta}{2}\right) \mathbf{n} , \quad (4.28)$$

with $|\mathbf{q}|$ the length of \mathbf{q} as defined in Eq. (4.13).

4.5 Applications

4.5.1 Targeting an Object in 3-D: Quaternion Approach

Let us revisit the *aerial gun* application in Sect. 3.6.1, but now assume that we have a targeting device that can be controlled with a quaternion. In other words, the zero quaternion describes the orientation where the targeting device is pointing straight ahead ($[1, 0, 0]$).

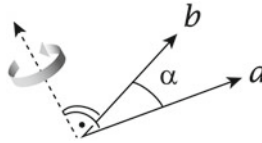


Fig. 4.3 The shortest rotation (α) that brings \mathbf{a} parallel to \mathbf{b} is about an axis perpendicular to \mathbf{a} and \mathbf{b}

Task: What quaternion would be needed to describe the target orientation, if the target is in an arbitrary location (x, y, z) ?

Solution: To answer that question, one can make use of the fact that the shortest rotation that brings a vector \mathbf{a} into alignment with a vector \mathbf{b} is a rotation about the direction perpendicular to \mathbf{a} and \mathbf{b} (see Fig. 4.3).

$$\mathbf{n} = \frac{\mathbf{a} \times \mathbf{b}}{|\mathbf{a} \times \mathbf{b}|} \quad (4.29)$$


by an angle equal to the angle α between the two vectors

$$\alpha = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right). \quad (4.30)$$

Given the rotation axis and angle, the most convenient way to represent that rotation is the quaternion vector

$$\mathbf{q}_{\text{adjust}} = \mathbf{n} * \sin(\alpha/2). \quad (4.31)$$

The corresponding algorithm is implemented in `skin.vector.q_shortest_rotation`. For example, if the target moved along an ∞ loop on a screen in 10m distance, the orientation of the following targeting device could be calculated with the following code:

 **python**™ **Code:** [C4_targeting.py](#): projecting an ∞ loop on a screen (p. 143).

Listing 4.1: `C4_targeting.py`

```

"""Orientation of targeting device.

"""
# author: Thomas Haslwanter, date: Nov-2017

# Import the required packages

```

```

import numpy as np
import matplotlib.pyplot as plt
import skinematics as skin

# Generate an "infinity"-loop, in 10m distance
t = np.arange(0,20,0.1) # 20 sec, at a rate of 0.1 Hz
y = np.cos(t)
z = np.sin(2*t)
x = 10 * np.ones_like(y)
data = np.column_stack( (x,y,z) )

# Calculate the target-orientation, i.e. the quaternion that
# rotates the vector [1,0,0] such that it points towards
# the target
q_target = skin.vector.q_shortest_rotation([1,0,0], data)

# Plot the results
fig, axs = plt.subplots(2,1)
axs[0].plot(-y,z)
axs[0].set_title('Target on screen, distance=10')
axs[1].plot(q_target)
axs[1].set_xlabel('Time')
axs[1].set_ylabel('Quaternion')
axs[1].legend(['x', 'y', 'z'])
plt.show()

```

4.5.2 Orientation of 3-D Acceleration Sensor

Task: Given an IMU with an accelerometer and a gyroscope only, what is the orientation of the IMU at the beginning of an experiment, based on the direction of gravity indicated by the accelerometer? Specifically, what would be the best guess of the orientation of the sensor in orientation 3 in the example in Fig. 4.4?

Solution: Many inertial sensors are shaped like a match box, and define their long side as the x -axis (\mathbf{b}_x), their shorter side as the y -axis (\mathbf{b}_y), and the “thick” side as the z -axis (\mathbf{b}_z). As explained in more detail in Sect. 2.2.2, a sensor lying flat and stationary on the ground (Fig. 4.4, orientation 1.) indicates an acceleration of

$$\mathbf{acc}_{\text{flat}} = \begin{pmatrix} 0 \\ 0 \\ +9.81 \end{pmatrix} \text{m/s}^2. \quad (4.32)$$

If this sensor is rotated “upright” by exactly 90° (Fig. 4.4, orientation 2.), the readout would indicate $(9.81/0/0) \text{m/s}^2$.

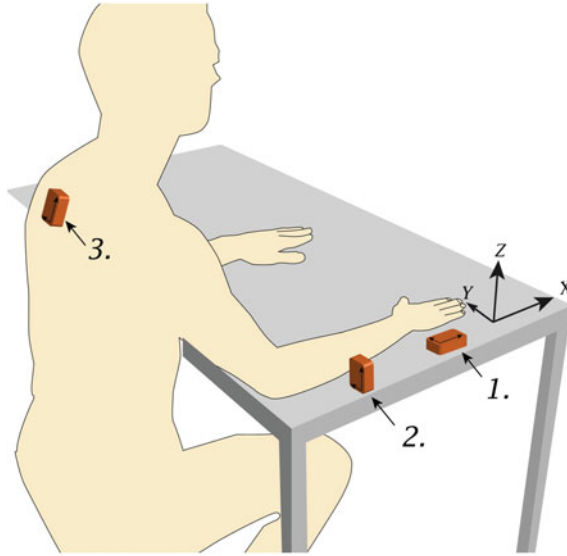


Fig. 4.4 *Orientation 1* Sensor aligned with space-fixed coordinate system. To find the orientation of the sensor on the back, based on the measured accelerations, we first specify the approximate sensor orientation (here *orientation 2*, sensor rotated by 90°). From the measured accelerations in *orientation 3* (sensor on back of subject), the tilt relative to orientation 2 can be determined as described in the text

A rotation “upright” can be indicated by a quaternion vector

$$\mathbf{q}_{\text{upright}} = \begin{pmatrix} 0 \\ -\sin(90^\circ/2) \\ 0 \end{pmatrix}. \tag{4.33}$$

In Fig. 4.4, orientation 3., this sensor is attached in approximately that orientation to the back of an upright standing or sitting person. Since the back of a person is not exactly vertical, the sensor is slightly rotated. What is the best estimate of the orientation of the sensor, when the readout, with the person stationary, indicates e.g. (9.75/0.98/ - 0.39)?

To answer this question, we need the shortest rotation $\tilde{\mathbf{q}}_{\text{adjust}}$ that brings the sensor from the “upright” orientation where the accelerometer indicates (9.81/0/0) to the current orientation, where it indicates (9.75/0.98/ - 0.39). Again, this is the rotation that brings two vectors into alignment, which can be found as in the example above.

Since a rotation about \mathbf{g} does not change the output of the accelerometer, the best estimate of the orientation of the accelerometer is

$$\tilde{\mathbf{q}}_{\text{total}} = \tilde{\mathbf{q}}_{\text{upright}} \circ \tilde{\mathbf{q}}_{\text{adjust}} , \tag{4.34}$$

where \circ indicates the quaternion multiplication. Using *scikit-kinematics*, this can be implemented as

```
# Import the required packages
import skinematics as skin

# Enter the measurements
g = [9.81, 0, 0]
g_upright = [9.75, 0.98, -0.39]

# Calculate the sensor orientation
q_adjust = skin.vector.qrotate(g, g_rotated)
q_upright = [0, np.sin(np.pi/4), 0]
q_total = skin.quat.quatmult(q_upright, q_adjust)
```

For some experiments, it may be impossible to mount the inertial sensors in an orientation approximating a space-fixed coordinate plane. For IMUs in arbitrary mounting orientation and position Seel et al. have proposed a set of methods that allow the determination of the local joint axis and position coordinates from arbitrary motions by exploitation of the kinematic constraints of the joint (Seel et al. 2014).


4.5.3 Calculating Orientation of a Camera on a Moving Object

Consider the problem where a camera in a missile must be pointed to look at a specific target. The missile attitude/orientation has three rotational degrees of freedom relative to the world. The camera attitude/orientation also has three rotational degrees of freedom but relative to the missile body. So the missile and camera gimbal forms a set of cascaded three-axis transformations. The camera gaze direction is $(1, 0, 0)$, i.e., the optical axis is along the x direction. When the camera is looking at the object in the world, the target's location in camera coordinates must, therefore, be $(x_{\text{obj}}^{\text{camera}}, 0, 0)$.

There are three coordinate systems in this scenario: (1) fixed to the world, (2) fixed to the missile body, and (3) fixed to the camera on the gimbal. The target direction in world coordinates is known from the location of the missile and the target. The target vector in the camera coordinates is $\mathbf{t}^c = [|\mathbf{t}|, 0, 0]$, i.e., the optical axis or gaze direction. The target direction in the missile body coordinates can be calculated.

In the example below, a missile is located at position $(10, 1700, -2200)$ m with an attitude (pitch = -1.2 , yaw = -0.2 , roll = -1.1) rad in Helmholtz sequence (roll–yaw–pitch from outside to inside). The target is located at position $(23, -560, -1800)$ m. How can one calculate the Helmholtz-sequence camera gimbal attitude, relative to the missile body, such that the camera optical axis points at the target? (To be on the optical axis in camera coordinates, the target vector in camera coordinates must be $(2295, 0, 0)$.)

4.5.3.1 Calculating Look-at Angles

 **python** Code: `C4_look_at.py`: How to calculate the orientation of a camera on a missile, in order to look in the direction of a given target.

Listing 4.2: `C4_look_at.py`

```

""" Given the positions of a missile and a target, and the
missile orientation, calculate the gimbal orientation of a
camera mounted on the missile, such that the camera
points at the target.
The optical axis of the camera is the x-axis.
"""

# author: ThH, date: July, 2018, ver: 1.1

# Import the required packages
import numpy as np
import skinematics as skin

def camera_orientation(missile_pos, missile_orient,
                      target_pos):
    '''Find camera orientation re missile, to focus on target.

    Inputs
    -----
    missile_pos : ndarray (3,) or (N,3)
        Position of missile in space
    missile_orient : ndarray (3,) or (N,3)
        Orientation of missile, in Helmholtz angles [rad]
    target_pos : ndarray (3,) or (N,3)
        Position of target in space

    Returns
    -----
    camera_orientation : ndarray (3,) or (N,3)
        Camera orientation, in Helmholtz angles [deg]
    '''

    # Required camera direction in space is a vector from
    # missile to target
    v_missile_target = target_pos - missile_pos

    # Camera direction re missile
    q = skin.rotmat.seq2quat(np.rad2deg(missile_orient),
                             seq='Helmholtz')
    tm_in_missile_CS = skin.vector.rotate_vector
    (v_missile_target, -q)

```

```

# Required camera orientation on missile, to focus on the
# target
camera_orientation = skin.vector.target2orient
(tm_in_missile_CS, orient_type='Helmholtz')

return camera_orientation

if __name__=='__main__':

# Set up the system
helm = [-1.2, -0.2, -1.1] # Missile orientation, in
# Helmholtz angles [rad]
target = np.r_[10, 1700, -2200]
missile = np.r_[23, -560, -1800]

# Find the camera orientation
camera = camera_orientation(missile, helm, target)

# Show the results
print('Camera orientation on missile, in Helmholtz
# angles:\n' +
# 'pitch={0:4.2f}, yaw={1:4.2f} [deg]'.
# format(*camera))

```

4.5.4 Object-Oriented Implementation of Quaternions

The Python module `scikit.quat` also contains a class `Quaternion` with multiplication, division, and inversion. A `Quaternion` can be created from vectors, rotation matrices, or from nautical angles, Helmholtz angles, or Euler angles. It provides operator overloading for `mult`, `div`, and `inv`, indexing, and access to the data in the attribute values.


```

import numpy as np
from skinematics.quat import Quaternion

data = np.array([[0,0,0.1], [0, 0.2, 0]])
data2 = np.array([[0,0,0.1], [0,0,0.1]])

eye = Quaternion(data)
head = Quaternion(data2)
gaze = head * eye
print(gaze)
#>> Quaternion [[ 0.98          0.          0.          0.19899749]
#>>           [ 0.97488461 -0.02          0.19899749 0.09797959]]

```


 **python**™ **Code:** [C4_examples_quat.py](#): Examples of working with quaternions: quaternion multiplication, conjugation, inversion, etc. (p. [144](#))