

# Spacecraft Math

Stephen Leake

27 September 2008



# Chapter 1

## Introduction

This document presents a thorough summary of vector, quaternion, and matrix math used in spacecraft applications, in both flight and simulation. It presents a standard notation, shows recommended usage, and provides references to other treatments of these topics as appropriate.

We start with a discussion of basic concepts; rotations, translations, poses, rates, wrenches, mass and inertia. Then we discuss various algorithms used in spacecraft software.

Each section is composed of four subsections; Reference, Coding, Examples, and Derivations. Reference gives the definitions and equations for the section. Coding points to the software source code implementing the section, and discusses any software design issues, in particular variable naming conventions. We present two naming conventions; the SAL naming convention, and the GN&C naming convention adopted by the Goddard Guidance, Navigation and Control branch. Examples gives numerical examples to illustrate the equations; accompanying Ada code implements the examples. Derivations motivates the definitions in the Reference subsection, discusses how they relate to similar equations in other works, and derives the equations from first principles. In many cases, Derivation refers to Macsyma code that provides the detailed algebraic manipulations.

The SAL software library does not assume the SAL naming convention. This is because the library supports left and right multiply rotations, and active and passive rotations. It also allows using the library with other naming conventions (such as GN&C). The Coding sections of this document define the SAL naming convention for each choice of left/right, active/passive, and show how it relates to the SAL software library.

All equations are presented in vector notation, with the coordinate frames

clearly indicated.

## Chapter 2

# 3 Degrees of Freedom

The main math of interest in 3 degrees of freedom is rotating vectors.

There are many representations of rotations. The two most important for computation are unit quaternions and orthonormal 3-by-3 matrices; these can be multiplied to combine rotations, and used to rotate vectors. On the other hand, the most intuitive representation is angle axis; it is usually easy for people to visualize a rotation as an angle about an axis.

In many situations, the actual representation of a rotation is not important; we can think of an abstract rotation  $\mathbf{R}$ . So we first present notation for abstract rotations, then discuss angle axis, quaternions, rotation matrices, and other representations that have special purposes.

In general, a position vector represents a displacement from a base position (usually the origin of a frame) to an object position, expressed in a coordinate frame. We use the notation  ${}_{base}^{ref}\mathbf{r}_{obj}$ , where *ref* is the frame the vector is expressed in, *base* is the origin of the vector, and *obj* is the position.

Note that negating a vector reverses the direction:

$${}_{obj}^{ref}\mathbf{r}_{base} = - {}_{base}^{ref}\mathbf{r}_{obj} \quad (2.0.0-1)$$

Typically, *base* is the origin of *ref*, and we drop *ref*.

## 2.1 Abstract rotations

### 2.1.1 Reference

An abstract rotation is denoted by  $\mathbf{R}$ . The inverse rotation (same axis and magnitude, opposite direction) is denoted  $\mathbf{R}^{-1}$ .

There are two different but similar rotation operations that change the frame of a vector:

$$\begin{aligned} {}^b\mathbf{r}_{obj} &= {}^b\mathbf{R}_a {}^a\mathbf{r}_{obj} \\ {}^b\mathbf{r}_{obj} &= {}^a\mathbf{r}_{obj} {}^a\mathbf{R}^b \end{aligned}$$

We call  ${}^b\mathbf{R}_a$  a *left-multiply* operator, and  ${}^a\mathbf{R}^b$  a *right-multiply* operator. [4] uses left-multiply rotation matrices and right-multiply quaternions. [6] uses right-multiply quaternions and left-multiply rotation matrices.

In this document, we present algorithms for both left and right multiply operators. The derivations are usually done in full for left-multiply operators; the right-multiply can be obtained simply by substitution.

The distinction between left- and right-multiply only matters for quaternion and matrix representations, which have multiplication operators. We will see below that  ${}^b\mathbf{R}^a = {}^a\mathbf{R}_b^{-1}$ .

#### Left-multiply

Rotations can either change the reference frame of a vector, or rotate the vector into a new vector:

$${}^b_{base}\mathbf{r}_1 = {}^b\mathbf{R}_a {}^a_{base}\mathbf{r}_1 \quad (2.1.1-1)$$

$${}^a_{base}\mathbf{r}_2 = {}^a\mathbf{R}_{1-2} {}^a_{base}\mathbf{r}_1 \quad (2.1.1-2)$$

${}^b_{base}\mathbf{r}_1$  is the same physical vector as  ${}^a_{base}\mathbf{r}_1$ , but expressed in a different reference frame. This is called a *passive* rotation. The two labels on the rotation are two frames:  ${}^a\mathbf{R}_b$  changes the frame of a vector from  $a$  to  $b$ ; the rotation is also expressed in frame  $a$ .

On the other hand,  ${}^a_{base}\mathbf{r}_2$  is a different vector. It is not parallel to  ${}^a_{base}\mathbf{r}_1$ , but it is expressed in the same frame; this is called an *active* rotation. On an active rotation, the pre-superscript labels the frame the rotation is expressed in, the post-subscript 1 – 2 identifies the start and end orientations; it rotates from orientation 1 to orientation 2. Often the orientations are also frames.

We could use a different notation for the active rotation. For example,  ${}^a\mathbf{R}_{2-1} {}^a_{base}\mathbf{r}_1$  or  ${}^a_2\mathbf{R}_1 {}^a_{base}\mathbf{r}_1$  would make it slightly easier to match the start orientation label

to the vector position label. However, the first option is confusing, since we normally read left-to-right. The second option is hard to correlate with code notations. The chosen notation works slightly better for right-multiply rotations (see below).

Since a rotation does not move the base position of a vector, we can usually drop the base position from the notation.

Note that the notation for a passive rotation gives two frames, while the notation for an active rotation gives one frame and two orientations, which may also be three frames.

The notation helps get the frames and usage correct; when changing reference frames of vectors (passive rotation), the post-subscript must match the pre-superscript for the equation to make sense. On the other hand, when physically rotating vectors (active rotation), all the pre-superscripts must be the same. This convention is especially useful when more than one rotation is present:

$${}^c\mathbf{R}_a = {}^c\mathbf{R}_b {}^b\mathbf{R}_a \quad (2.1.1-3)$$

$${}^c\mathbf{r}_1 = {}^c\mathbf{R}_b {}^b\mathbf{R}_a {}^a\mathbf{r}_1 \quad (2.1.1-4)$$

$${}^a\mathbf{R}_{1-3} = {}^a\mathbf{R}_{1-2} {}^a\mathbf{R}_{2-3} \quad (2.1.1-5)$$

$${}^a\mathbf{r}_3 = {}^a\mathbf{R}_{2-3} {}^a\mathbf{R}_{1-2} {}^a\mathbf{r}_1 \quad (2.1.1-6)$$

Note that the inverse of a rotation is obtained by swapping labels:

$${}^a\mathbf{R}_b^{-1} = {}^b\mathbf{R}_a \quad (2.1.1-7)$$

$${}^a\mathbf{R}_{2-1}^{-1} = {}^a\mathbf{R}_{1-2} \quad (2.1.1-8)$$

Note that this holds for combined rotations:

$$\begin{aligned} {}^c\mathbf{R}_a &= {}^c\mathbf{R}_b {}^b\mathbf{R}_a \\ {}^c\mathbf{R}_a^{-1} &= {}^b\mathbf{R}_a^{-1} {}^c\mathbf{R}_b^{-1} \\ &= {}^a\mathbf{R}_b {}^b\mathbf{R}_c \\ &= {}^a\mathbf{R}_c \\ {}^a\mathbf{R}_{1-3} &= {}^a\mathbf{R}_{2-3} {}^a\mathbf{R}_{1-2} \\ {}^a\mathbf{R}_{1-3}^{-1} &= {}^a\mathbf{R}_{1-2}^{-1} {}^a\mathbf{R}_{2-3}^{-1} \\ &= {}^a\mathbf{R}_{2-1} {}^a\mathbf{R}_{3-2} \\ &= {}^a\mathbf{R}_{3-2} \end{aligned}$$

There are two passive rotations related to an active rotation:

$$\begin{aligned} {}^a\mathbf{R}_b &= {}^a\mathbf{R}_{a-b} \\ {}^b\mathbf{R}_a &= {}^a\mathbf{R}_{a-b}^{-1} \end{aligned} \quad (2.1.1-9)$$

Note that for these equations to make sense, the reference frame of the active rotation must be either  $a$  or  $b$ . We will see in 2.3 how to change the reference frame of an active rotation quaternion.

Rotations are not commutative;  $\mathbf{R}_b\mathbf{R}_a$  is not the same as  $\mathbf{R}_a\mathbf{R}_b$ . However, rotations are associative;  $(\mathbf{R}_c\mathbf{R}_b)\mathbf{R}_a = \mathbf{R}_c(\mathbf{R}_b\mathbf{R}_a)$ .

### Right-multiply

The notation for right-multiply abstract rotations is:

$$\begin{aligned} {}^b\mathbf{r}_1 &= {}^a\mathbf{r}_1 {}^a\mathbf{R}^b \\ {}^a\mathbf{r}_2 &= {}^a\mathbf{r}_1 {}_{1-2}\mathbf{R}^a \\ {}^a\mathbf{R}^c &= {}^a\mathbf{R}^b {}^b\mathbf{R}^c \\ {}^c\mathbf{r}_1 &= {}^a\mathbf{r}_1 {}^a\mathbf{R}^b {}^b\mathbf{R}^c \\ {}_{1-3}\mathbf{R}^a &= {}_{1-2}\mathbf{R}^a {}_{2-3}\mathbf{R}^a \\ {}^a\mathbf{r}_3 &= {}^a\mathbf{r}_1 {}_{1-2}\mathbf{R}^a {}_{2-3}\mathbf{R}^a & (2.1.1-10) \\ {}^b\mathbf{R}^a &= {}_{a-b}\mathbf{R}^a \\ {}^a\mathbf{R}^b &= {}_{a-b}\mathbf{R}^{a-1} & (2.1.1-11) \end{aligned}$$

Note that the order of the two orientations (in the subscript) in active rotations is *not* changed between left and right multiply notations.

Note that we do *not* have a different notation for vectors in equations when using right-multiply rotations, even though we do in code.

### 2.1.2 Coding

We summarize the coding conventions for translation vectors, quaternions, and matrices here, to have them all in one place.

In the SAL coding naming convention, a variable representing the abstract left-multiply passive rotation  ${}^a\mathbf{R}_b$  is written `A_Rot_B`. The equivalent right-multiply passive rotation  ${}^b\mathbf{R}^a$  is written `B_Rot_A`; the order of the labels reflects the order in the notation.

The type `SAL.Gen_Math.Gen_DOF_3.Cart_Vector_Type` implements Cartesian vectors.

When using left-multiply rotations, a vector  ${}^a\mathbf{r}_1$  is written `A_Tran_1`; when using right multiply rotations, the vector is written `1_Tran_A`. Thus (2.1.1-1) is written:

$$\text{B\_Tran\_1} := \text{B\_Rot\_A} * \text{A\_Tran\_1}$$

and the right rotation equivalent is written:

$$1\_Tran\_B := 1\_Tran\_A * A\_Rot\_B$$

However, Ada names cannot start with numbers, so care must be taken to use position and orientation labels that start with letters.

Since a single program should only use either the right-multiply or left-multiply convention, the notation should not become confusing.

The left-multiply active rotation  ${}^a\mathbf{R}_{1-2}$  is written `A_Rot_1_To_2`. Thus (2.1.1-2) is written:

$$A\_Tran\_2 := A\_Rot\_1\_To\_2 * A\_Tran\_1$$

and the right multiply equivalent:

$$2\_Tran\_A := 1\_Tran\_A * 1\_To\_2\_Rot\_A$$

In the SAL naming convention, the notation for quaternions is the same as for abstract rotations, with `Quat` instead of `Rot`. Similarly, for matrices the SAL naming convention uses `Rot_Mat` instead of `Rot`.

In the GN&C convention (which dictates more than just naming), rotation matrices are always passive left-multiply, and rotation quaternions are always passive right-multiply. The left multiply matrix  ${}^b\mathbf{M}_a$  is written `Dcm_AToB`; the right-multiply quaternion  ${}_a\mathbf{Q}^b$  is written `Quat_AToB`. A position vector for *obj* expressed in frame *a* is written `Pos_A_Obj`. Thus (2.1.1-1) is written:

$$Pos\_B\_Obj := Dcm\_AToB * Pos\_A\_Obj$$

and the quaternion right multiply equivalent is written:

$$Pos\_B\_Obj := Pos\_A\_Obj * Quat\_AToB$$

Thus for rotation matrices, the frame order in the names is swapped between SAL and GN&C naming conventions - the left-multiply passive rotation matrix  ${}^a\mathbf{M}_b$  is :

$${}^a\mathbf{M}_b = A\_Rot\_Mat\_B = Dcm\_BToA \quad (2.1.2-1)$$

However, for rotation quaternions, the frame order in the names is the same between SAL and GN&C naming conventions - the right-multiply passive rotation quaternion  ${}_b\mathbf{Q}^a$  is :

$${}_b\mathbf{Q}^a = B\_Quat\_A = Quat\_BToA \quad (2.1.2-2)$$

When using the GN&C convention, left and right multiplies are mixed in the same program, which will be confusing. To reduce the chances for error, left multiply matrices should only be used as user inputs; they should be immediately converted to right multiply quaternions before being used. In addition, the matrices should be named using the GN&C naming convention, while the

quaternions should use the SAL naming convention. The conversion is given by:

$${}_b\mathbf{Q}^a = (\text{to\_unit\_quaternion}({}^a\mathbf{M}_b))^{-1} \quad (2.1.2-3)$$

`B_Quat_A := SAL.Math_Double.Dof_3.Wertz.To_Unit_Quaternion (Dcm_BToA)`

where `to_unit_quaternion` is given by (2.4.1-5).

Note that

`SAL.Math_Double.Dof_3.Wertz.To_Unit_Quaternion` combines the inverse with the conversion to quaternion, preserving the frame order in this pair of variable names. This is because of the implicit conversion from left to right multiply, and it helps avoid mistakes in user code.

There is no Ada type corresponding to an abstract rotation. However, similar operations for the concrete rotation types are coded in similar ways, to make it easy to change the rotation type in applications.

It is tempting to define two different Ada types for quaternions (or matrices); left-multiply and right-multiply. Then the compiler would guarantee they never got mixed up. However, that would mean any higher-level types that had quaternion components would also need left and right versions. In addition, code for operations that do not depend on the choice of left or right-multiply must be duplicated. This is too heavy a burden. Instead, the SAL math library provides one package defining rotation quaternion and rotation matrix types, and all the operations on them that do not depend on the choice of left or right multiply. Then there are two child packages providing the operations that do depend on the choice. One provides left-multiply quaternion and matrix operations. The other, following [6] and [3], supports right-multiply quaternion and left-multiply matrix operations. For higher level packages that use rotations, a similar package structure is used.

In any case, careful attention must be paid to whether rotations are active vs passive, to the signs of angles, and to the order of operands in multiplication.

## 2.2 Angle Axis rotation representation

### 2.2.1 Reference

Every active rotation can be expressed as an angle about an axis. If  $\theta_{1-2}$  is the angle from vector  ${}^a\mathbf{r}_1$  to vector  ${}^a\mathbf{r}_2$  about axis  ${}^a\hat{\mathbf{n}}$  (using the right-hand rule for the sign of  $\theta_{1-2}$ ), then

$${}^a\mathbf{R}_{1-2} = (\theta_{1-2}, {}^a\hat{\mathbf{n}}) \quad (2.2.1-1)$$

Note that  $\hat{\mathbf{n}}$  has magnitude 1:

$$n_x^2 + n_y^2 + n_z^2 = 1 \quad (2.2.1-2)$$

The “right-hand rule” is: hold the right hand with the thumb extended and the fingers loosely curled. Then the right thumb is aligned with the axis of rotation, and the fingers curl in the positive direction of the rotation.

Note that the same rotation can be represented by negating both  $\theta_{1-2}$  and  ${}^a\hat{\mathbf{n}}$ :

$${}^a\mathbf{R}_{1-2} = (-\theta_{1-2}, -{}^a\hat{\mathbf{n}}) \quad (2.2.1-3)$$

So generally we can choose  $0 \leq \theta_{1-2} \leq \pi$ . The Ada code does not enforce this in general, but it does take advantage of it when converting from other rotation representations to angle axis.

The inverse rotation is:

$$\begin{aligned} {}^a\mathbf{R}_{2-1} &= (-\theta_{1-2}, {}^a\hat{\mathbf{n}}) \\ &= (\theta_{2-1}, {}^a\hat{\mathbf{n}}) \end{aligned} \quad (2.2.1-4)$$

The active rotation is computed by:

$${}^a\mathbf{r}_2 = {}^a\mathbf{r}_1 \cos\theta_{1-2} - {}^a\mathbf{r}_1 \times {}^a\hat{\mathbf{n}} \sin\theta_{1-2} + {}^a\hat{\mathbf{n}} ({}^a\mathbf{r}_1 \cdot {}^a\hat{\mathbf{n}}) (1 - \cos\theta_{1-2}) \quad (2.2.1-5)$$

Note that if we have two rotations  $(\theta_{1-2}, \hat{\mathbf{n}}_1)$  and  $(\theta_{2-3}, \hat{\mathbf{n}}_2)$ , there is no simple way to find the equivalent total rotation. Rotation quaternions or matrices provide that operation.

We now consider the passive rotations that are related to this active rotation. Since this involves changing frames, we use letters to label the two orientations involved in the active rotation. Then the passive rotations that change the frame of vectors from  $b$  to  $a$  and vice versa are:

$$\begin{aligned} {}^a\mathbf{R}_b &= (\theta_{a-b}, {}^a\hat{\mathbf{n}}) \\ {}^b\mathbf{R}_a &= (-\theta_{a-b}, {}^a\hat{\mathbf{n}}) \end{aligned} \quad (2.2.1-6)$$

An important case is obtaining the minimal rotation that takes one vector into another. Given two unit vectors  ${}^a\hat{\mathbf{n}}_1$  and  ${}^a\hat{\mathbf{n}}_2$  expressed in a common frame  $f$ , the active rotation that moves  ${}^a\hat{\mathbf{n}}_1$  to be parallel to  ${}^a\hat{\mathbf{n}}_2$  is:

$$\begin{aligned} {}^a\mathbf{v} &= {}^a\hat{\mathbf{n}}_1 \times {}^a\hat{\mathbf{n}}_2 \\ s &= |{}^a\mathbf{v}| \\ c &= {}^a\hat{\mathbf{n}}_1 \cdot {}^a\hat{\mathbf{n}}_2 \\ \theta_{1-2} &= \text{atan2}(s, c) \end{aligned}$$

if  $s = 0$  then

$${}^a\hat{\mathbf{n}} = (1.0, 0.0, 0.0)$$

else

$${}^a\hat{\mathbf{n}} = {}^a\mathbf{v}/s \quad (2.2.1-7)$$

end if

### 2.2.2 Coding

The type `SAL.Gen_Math.Gen_DOF_3.Mag_Axis_Type` implements the angle axis rotation representation. Operations are provided for scaling and conversion to quaternions and rotation matrices. `SAL.Gen_Math.Gen_DOF_3.Rotate` implements (2.2.1-5).

In the SAL naming convention, angle axis rotations are always active. See 2.3.2 and 2.4.2 for functions related to converting angle axis to quaternions and matrices.

### 2.2.3 Examples

Some simple numerical examples. See `spacecraft_math_examples.adb`, `Angle_Axis` block for Ada code implementing these.

First, an active rotation of a vector in the x-z plane in a positive direction about the y axis.

$$\begin{aligned} {}^{base}\mathbf{r}_1 &= (2.0, 0.0, 1.0) \\ {}^{base}\mathbf{R}_{1-2} &= (\theta_{1-2}, {}^{base}\hat{\mathbf{n}}) \\ &= (0.1, (0.0, 1.0, 0.0)) \\ {}^{base}\mathbf{r}_2 &= (2.08984, 0.00000, 0.79534) \end{aligned}$$

Note that the x component increases, and the z component decreases; the end point of the vector is moved closer to the x axis.

Now a spacecraft example; the direction of the z axis of the spacecraft. Assume the Sun is at the origin of the base frame, and the spacecraft is positioned on the negative z axis of the base frame. Initially, the z axis of the spacecraft frame is pointing at the sun. Then we rotate the spacecraft in a positive direction about the y axis. Frame `sc0` gives the initial orientation of the spacecraft; it is parallel to frame `base`. Frame `sc1` gives the final orientation of the spacecraft.

Unit vector  ${}^{base}\hat{\mathbf{n}}_{scz0}$  gives the direction of the z axis of the spacecraft in the base frame in the initial orientation;  ${}^{base}\hat{\mathbf{n}}_{scz1}$  in the final.

$$\begin{aligned} {}^{base}\hat{\mathbf{n}}_{scz0} &= (0.0, 0.0, 1.0) \\ {}^{base}\mathbf{R}_{sc0-sc1} &= (0.1, (0.0, 1.0, 0.0)) \\ {}^{base}\hat{\mathbf{n}}_{scz1} &= {}^{base}\mathbf{R}_{sc0-sc1} {}^{base}\hat{\mathbf{n}}_{scz0} \\ &= (0.09983, 0.00000, 0.99500) \end{aligned}$$

In the base frame, the z axis of the spacecraft has moved towards the positive x axis.

Now consider the direction from the spacecraft to the Sun; we know what it is in the base frame, and we need to find it in the spacecraft frame. The unit vector  $\hat{\mathbf{n}}_{sc-sun}$  gives the direction from the spacecraft to the sun, in a given frame.

$$\begin{aligned} {}^{base}\hat{\mathbf{n}}_{sc-sun} &= (0.0, 0.0, 1.0) \\ {}^{sc0}\hat{\mathbf{n}}_{sc-sun} &= (0.0, 0.0, 1.0) \\ {}^{sc1}\mathbf{R}_{sc0} &= {}^{base}\mathbf{R}_{sc0-sc1}^{-1} \\ &= (-0.1, (0.0, 1.0, 0.0)) \\ {}^{sc1}\hat{\mathbf{n}}_{sc-sun} &= {}^{sc1}\mathbf{R}_{sc0} {}^{sc0}\hat{\mathbf{n}}_{sc-sun} \\ &= (-0.09983, 0.00000, 0.99500) \end{aligned}$$

In the spacecraft frame, the direction to the sun has moved towards the negative x axis.

## 2.2.4 Derivations

First we show that (2.2.1-5) is correct. Let frame  $a$  be fixed to the non-rotated vector  ${}^a\mathbf{r}_1$ ; frame  $b$  is fixed to the rotated vector  ${}^a\mathbf{r}_2$ . We denote the axes of frame  $a$  by  $(\hat{\mathbf{x}}_a, \hat{\mathbf{y}}_a, \hat{\mathbf{z}}_a)$ . For simplicity, we use  $r_x$  to denote the x component of vector  $\mathbf{r}$  in either frame; similarly for y and z. By definition:

$${}^a\mathbf{r}_1 = r_x {}^a\hat{\mathbf{x}}_a + r_y {}^a\hat{\mathbf{y}}_a + r_z {}^a\hat{\mathbf{z}}_a \quad (2.2.4-1)$$

Because  ${}^a\mathbf{r}_2$  is a rotation of  ${}^a\mathbf{r}_1$ , it has the same values for its components in  $b$  as  ${}^a\mathbf{r}_1$  has in  $a$ , so we can write:

$${}^a\mathbf{r}_2 = r_x {}^a\hat{\mathbf{x}}_b + r_y {}^a\hat{\mathbf{y}}_b + r_z {}^a\hat{\mathbf{z}}_b \quad (2.2.4-2)$$

Without loss of generality, we can let the axis of rotation  ${}^a\hat{\mathbf{n}} = {}^a\hat{\mathbf{z}} = {}^b\hat{\mathbf{z}}$  (this just defines the two frames uniquely). Then the axes of the frames are related by:

$${}^a\hat{\mathbf{x}}_b = \cos\theta_{1-2} {}^a\hat{\mathbf{x}}_a + \sin\theta_{1-2} {}^a\hat{\mathbf{y}}_a \quad (2.2.4-3)$$

$${}^a\hat{\mathbf{y}}_b = -\sin\theta_{1-2} {}^a\hat{\mathbf{x}}_a + \cos\theta_{1-2} {}^a\hat{\mathbf{y}}_a \quad (2.2.4-4)$$

$${}^a\hat{\mathbf{z}}_b = {}^a\hat{\mathbf{z}}_a \quad (2.2.4-5)$$

Note that when  $\theta_{1-2} = 0$ ,  $b = a$ .

Substituting (2.2.4-3), (2.2.4-4) , (2.2.4-5) into (2.2.4-2):

$${}^a\mathbf{r}_2 = (r_x \cos\theta_{1-2} - r_y \sin\theta_{1-2}) {}^a\hat{\mathbf{x}}_a + (r_x \sin\theta_{1-2} + r_y \cos\theta_{1-2}) {}^a\hat{\mathbf{y}}_a + r_z {}^a\hat{\mathbf{z}}_a \quad (2.2.4-6)$$

Expanding (2.2.1-5), we have

$${}^a\mathbf{r}_{rot} = r_x \cos\theta_{1-2} {}^a\hat{\mathbf{x}}_a + r_y \cos\theta_{1-2} {}^a\hat{\mathbf{y}}_a + r_z \cos\theta_{1-2} {}^a\hat{\mathbf{z}}_a - \quad (2.2.4-7)$$

$$(-r_x {}^a\hat{\mathbf{y}}_a + r_y {}^a\hat{\mathbf{x}}_a) \sin\theta_{1-2} +$$

$$r_z (1 - \cos\theta_{1-2}) {}^a\hat{\mathbf{z}}_a$$

$$= (r_x \cos\theta_{1-2} - r_y \sin\theta_{1-2}) {}^a\hat{\mathbf{x}}_a + (r_x \sin\theta_{1-2} + r_y \cos\theta_{1-2}) {}^a\hat{\mathbf{y}}_a + r_z {}^a\hat{\mathbf{z}}_a \quad (2.2.4-8)$$

which is the same as (2.2.4-6). Repeating this analysis, picking  ${}^a\hat{\mathbf{n}} = {}^a\hat{\mathbf{x}}, {}^a\hat{\mathbf{y}}$ , shows that (2.2.1-5) is correct.

## 2.2.5 Comparison to Kane

(2.2.1-5) corresponds to [4, eqn 1.1(1)]. The derivation here follows Kane's.

## 2.3 Quaternion rotation representation

### 2.3.1 Reference

A rotation quaternion consists of four numbers, where the sum of the squares is 1:

$$\mathbf{Q} = (Q_x, Q_y, Q_z, Q_s) \quad (2.3.1-1)$$

$$1 = Q_x^2 + Q_y^2 + Q_z^2 + Q_s^2 \quad (2.3.1-2)$$

This is a unit quaternion; in general, non-unit quaternions (magnitude not restricted to 1) are also useful, but we do not consider them here.

Given an active angle axis rotation, the active left-multiply unit quaternion

equivalent is given by:

$$\begin{aligned}
{}^a\mathbf{R}_{a-b} &= (\theta_{a-b}, {}^a\hat{\mathbf{n}}) \\
{}^a\mathbf{R}_{a-b} &= {}^a\mathbf{Q}_{a-b} \\
{}^a\mathbf{Q}_{a-b} &= (\sin(\theta_{a-b}/2){}^a\hat{\mathbf{n}}, \cos(\theta_{a-b}/2)) \\
Q_x &= \sin(\theta_{a-b}/2)n_x \\
Q_y &= \sin(\theta_{a-b}/2)n_y \\
Q_z &= \sin(\theta_{a-b}/2)n_z \\
Q_s &= \cos(\theta_{a-b}/2)
\end{aligned} \tag{2.3.1-3}$$

$Q_x, Q_y, Q_z$  are called the *vector* components;  $Q_s$  is called the *scalar* component.

The related passive rotations are:

$$\begin{aligned}
{}^a\mathbf{Q}_b &= (\sin(\theta_{a-b}/2){}^a\hat{\mathbf{n}}, \cos(\theta_{a-b}/2)) \\
{}^b\mathbf{Q}_a &= (-\sin(\theta_{a-b}/2){}^a\hat{\mathbf{n}}, \cos(\theta_{a-b}/2))
\end{aligned} \tag{2.3.1-4}$$

Negating all elements of the quaternion represents the same rotation:

$${}^a\mathbf{R}_{a-b} = (-Q_x, -Q_y, -Q_z, -Q_s) \tag{2.3.1-5}$$

The inverse of a quaternion is given by:

$${}^b\mathbf{Q}_a = {}^a\mathbf{Q}_b^{-1} = (-Q_x, -Q_y, -Q_z, Q_s) \tag{2.3.1-6}$$

To obtain the active angle and axis  $(\theta_{a-b}, {}^a\hat{\mathbf{n}})$  from a left-multiply active quaternion  ${}^a\mathbf{Q}_{a-b}$ :

if  $Q_s < 0$  then

$$\mathbf{Q} = (-Q_x, -Q_y, -Q_z, -Q_s)$$

end if

$$s = \sqrt{Q_x^2 + Q_y^2 + Q_z^2}$$

if  $s > 0$  then

$$\begin{aligned}
\theta_{a-b} &= 2 \operatorname{atan2}(s, Q_s) \\
{}^a\hat{\mathbf{n}} &= (Q_x/s, Q_y/s, Q_z/s)
\end{aligned} \tag{2.3.1-7}$$

else

$$\begin{aligned}
\theta_{a-b} &= 0 \\
{}^a\hat{\mathbf{n}} &= (1, 0, 0)
\end{aligned}$$

end if

Note that the resulting  $\theta$  is in the range  $[0, \pi]$ . Given an angle axis with  $\theta$  in  $[-\pi, 0.0]$ , conversion to quaternion and back will negate the unit vector.

To multiply a vector by a quaternion :

$$\begin{aligned}
 \mathbf{r}' &= \mathbf{Q}\mathbf{r} \\
 a &= (Q_y r_z - Q_z r_y, Q_z r_x - Q_x r_z, Q_x r_y - Q_y r_x) \\
 r'_x &= 2(Q_s a_x + Q_y a_z - Q_z a_y) + r_x \\
 r'_y &= 2(Q_s a_y + Q_z a_x - Q_x a_z) + r_y \\
 r'_z &= 2(Q_s a_z + Q_x a_y - Q_y a_x) + r_z
 \end{aligned} \tag{2.3.1-8}$$

If  $\mathbf{Q}$  is an active quaternion (ie, it represents an active rotation),  $\mathbf{r}'$  is a rotated vector:

$$\mathbf{r}' = {}^a\mathbf{r}_2 = {}^a\mathbf{Q}_{1-2} {}^a\mathbf{r}_1 \tag{2.3.1-9}$$

If  $\mathbf{Q}$  is a passive rotation,  $\mathbf{r}'$  is  $\mathbf{r}$  expressed in a different frame:

$$\mathbf{r}' = {}^b\mathbf{r}_1 = {}^b\mathbf{Q}_a {}^a\mathbf{r}_1 \tag{2.3.1-10}$$

To multiply quaternions:

$$\begin{aligned}
 \mathbf{Q}_3 &= \mathbf{Q}_1 \mathbf{Q}_2 \\
 Q_{3x} &= Q_{1y} Q_{2z} - Q_{1z} Q_{2y} + Q_{1s} Q_{2x} + Q_{1x} Q_{2s} \\
 Q_{3y} &= -Q_{1x} Q_{2z} + Q_{1s} Q_{2y} + Q_{1z} Q_{2x} + Q_{1y} Q_{2s} \\
 Q_{3z} &= Q_{1s} Q_{2z} + Q_{1x} Q_{2y} - Q_{1y} Q_{2x} + Q_{1z} Q_{2s} \\
 Q_{3s} &= -Q_{1z} Q_{2z} - Q_{1y} Q_{2y} - Q_{1x} Q_{2x} + Q_{1s} Q_{2s}
 \end{aligned} \tag{2.3.1-11}$$

If the quaternions are active, this represents two successive physical rotations:

$$\begin{aligned}
 {}^a\mathbf{Q}_{1-3} &= {}^a\mathbf{Q}_{2-3} {}^a\mathbf{Q}_{1-2} \\
 {}^a\mathbf{r}_3 &= {}^a\mathbf{Q}_{2-3} {}^a\mathbf{Q}_{1-2} {}^a\mathbf{r}_1
 \end{aligned} \tag{2.3.1-12}$$

If the quaternions are passive, this represents two successive coordinate transforms:

$$\begin{aligned}
 {}^c\mathbf{Q}_a &= {}^c\mathbf{Q}_b {}^b\mathbf{Q}_a \\
 {}^c\mathbf{r}_1 &= {}^c\mathbf{Q}_b {}^b\mathbf{Q}_a {}^a\mathbf{r}_1
 \end{aligned} \tag{2.3.1-13}$$

The passive quaternions related to an active quaternion, when the reference frame of the active quaternion is either the start or end frame:

$$\begin{aligned}
 {}^a\mathbf{Q}_b &= {}^a\mathbf{Q}_{a-b} \\
 {}^b\mathbf{Q}_a &= {}^a\mathbf{Q}_{a-b}^{-1}
 \end{aligned} \tag{2.3.1-14}$$

To change the reference frame of an active quaternion:

$${}^b\mathbf{Q}_{b-c} = {}^b\mathbf{Q}_a {}^a\mathbf{Q}_{b-c} {}^b\mathbf{Q}_a^{-1} \quad (2.3.1-15)$$

To convert an active quaternion to a passive quaternion when the reference frame of the active quaternion is arbitrary:

$${}^c\mathbf{Q}_b = {}^b\mathbf{Q}_a {}^a\mathbf{Q}_{b-c}^{-1} {}^b\mathbf{Q}_a^{-1} \quad (2.3.1-16)$$

Successive passive quaternions related to successive active quaternions:

$$\begin{aligned} {}^a\mathbf{Q}_{a-c} &= {}^a\mathbf{Q}_{b-c} {}^a\mathbf{Q}_{a-b} \\ {}^a\mathbf{Q}_c &= {}^a\mathbf{Q}_b {}^b\mathbf{Q}_c \\ {}^c\mathbf{Q}_a &= {}^c\mathbf{Q}_b {}^b\mathbf{Q}_a \end{aligned} \quad (2.3.1-17)$$

Given two unit vectors, find the minimal rotation that takes one into the other:

$$\begin{aligned} {}^a\mathbf{v} &= {}^a\hat{\mathbf{n}}_1 \times {}^a\hat{\mathbf{n}}_2 \\ s &= |{}^a\mathbf{v}| \\ c &= {}^a\hat{\mathbf{n}}_1 \cdot {}^a\hat{\mathbf{n}}_2 \end{aligned}$$

if  $s = 0$  then

$${}^a\mathbf{Q}_{1-2} = (0.0, 0.0, 0.0, 1.0)$$

else

$$\begin{aligned} s2 &= \sqrt{\frac{1-c}{2}} \\ c2 &= \sqrt{\frac{1+c}{2}} \\ {}^a\mathbf{Q}_{1-2} &= \left( \frac{s2}{s} {}^a\mathbf{v}, c2 \right) \end{aligned} \quad (2.3.1-18)$$

end if

(2.3.1-18) defines the function `units_to_quat` for left-multiply.

### Right-multiply

For right-multiply quaternions, we have:

$$\begin{aligned} {}^b\mathbf{R}^a &= {}^b\mathbf{Q}^a \\ {}^b\mathbf{Q}^a &= {}^a\mathbf{Q}_b^{-1} \end{aligned} \quad (2.3.1-19)$$

Active quaternion and active angle axis:

$${}_{1-2}\mathbf{Q}^a = (-\sin(\theta_{1-2}/2))^a \hat{\mathbf{n}}, \cos(\theta_{1-2}/2) \quad (2.3.1-20)$$

$$\mathbf{r}' = \mathbf{r} \mathbf{Q}$$

$$a = (-Q'_y r_z + Q'_z r_y, -Q'_z r_x + Q'_x r_z, -Q'_x r_y + Q'_y r_x)$$

$$r'_x = 2(Q'_s a_x - Q'_y a_z + Q'_z a_y) + r_x$$

$$r'_y = 2(Q'_s a_y - Q'_z a_x + Q'_x a_z) + r_y$$

$$r'_z = 2(Q'_s a_z - Q'_x a_y + Q'_y a_x) + r_z \quad (2.3.1-21)$$

$${}_{1-3}\mathbf{Q}^a = {}_{1-2}\mathbf{Q}^a {}_{2-3}\mathbf{Q}^a \quad (2.3.1-22)$$

$${}^a\mathbf{r}_3 = {}^a\mathbf{r}_1 {}_{1-2}\mathbf{Q}^a {}_{2-3}\mathbf{Q}^a \quad (2.3.1-23)$$

Passive quaternions:

$$\begin{aligned} {}_b\mathbf{Q}^a &= {}_{a-b}\mathbf{Q}^a \\ {}_a\mathbf{Q}^b &= {}_{a-b}\mathbf{Q}^{a-1} \\ {}_a\mathbf{Q}^c &= {}_a\mathbf{Q}^b {}_b\mathbf{Q}^c \\ {}^c\mathbf{r}_1 &= {}^a\mathbf{r}_1 {}_a\mathbf{Q}^b {}_b\mathbf{Q}^c \end{aligned} \quad (2.3.1-24)$$

To obtain the active angle and axis  $(\theta_{1-2}, {}^a\hat{\mathbf{n}})$  from a right-multiply active quaternion  ${}_{1-2}\mathbf{Q}^a$ :

if  $Q_s < 0$  then

$$\mathbf{Q} = (-Q_x, -Q_y, -Q_z, -Q_s)$$

end if

$$s = \sqrt{Q_x^2 + Q_y^2 + Q_z^2}$$

if  $s > 0$  then

$$\begin{aligned} \theta_{1-2} &= 2 \operatorname{atan2}(s, Q_s) \\ {}^a\hat{\mathbf{n}}_b &= -(Q_x/s, Q_y/s, Q_z/s) \end{aligned} \quad (2.3.1-25)$$

else

$$\begin{aligned} \theta_{1-2} &= 0 \\ {}^a\hat{\mathbf{n}}_b &= (1, 0, 0) \end{aligned}$$

end if

Changing the frame of an active quaternion, and successive rotations:

$${}^a\mathbf{Q}^b = {}_{a-b}\mathbf{Q}^{a-1} \quad (2.3.1-26)$$

$${}_{b-c}\mathbf{Q}^b = {}^a\mathbf{Q}^{b-1} {}_{b-c}\mathbf{Q}^a {}^a\mathbf{Q}^b \quad (2.3.1-27)$$

$${}^b\mathbf{Q}^c = {}^a\mathbf{Q}^{b-1} {}_{b-c}\mathbf{Q}^{a-1} {}^a\mathbf{Q}^b \quad (2.3.1-28)$$

$${}_{a-c}\mathbf{Q}^a = {}_{a-b}\mathbf{Q}^a {}_{b-c}\mathbf{Q}^a$$

$${}^c\mathbf{Q}^a = {}^c\mathbf{Q}^b {}^b\mathbf{Q}^a$$

$${}^a\mathbf{Q}^c = {}^a\mathbf{Q}^b {}^b\mathbf{Q}^c \quad (2.3.1-29)$$

Function `units_to_quat`:

$$\begin{aligned} {}^a\mathbf{v} &= {}^a\hat{\mathbf{n}}_1 \times {}^a\hat{\mathbf{n}}_2 \\ s &= |{}^a\mathbf{v}| \\ c &= {}^a\hat{\mathbf{n}}_1 \cdot {}^a\hat{\mathbf{n}}_2 \end{aligned}$$

if  $s = 0$  then

$${}^a\mathbf{Q}_{1-2} = (0.0, 0.0, 0.0, 1.0)$$

else

$$\begin{aligned} s2 &= \sqrt{\frac{1-c}{2}} \\ c2 &= \sqrt{\frac{1+c}{2}} \\ {}^a\mathbf{Q}_{1-2} &= \left(-\frac{s2}{s} {}^a\mathbf{v}, c2\right) \end{aligned} \quad (2.3.1-30)$$

end if

### 2.3.2 Coding

The type `SAL.Gen_Math.Gen_DOF_3.Unit_Quaternion_Type` implements the quaternion rotation representation. The child package `SAL.Gen_Math.Gen_DOF_3.Gen_Left` provides left-multiply operations, the child package `SAL.Gen_Math.Gen_DOF_3.Gen_Wertz` provides right-multiply quaternion and left-multiply matrix operations.

It is up to the user to decide whether the quaternions are passive or active, when using these packages. In particular, in the conversions between quaternions and angle axis, the quaternion and angle axis are either both active or both passive.

In the SAL convention, a variable representing the passive left-multiply quaternion  ${}^a\mathbf{Q}_b$  is written `A_Quat_B`; a right-multiply quaternion  ${}_b\mathbf{Q}^a$  is written

`B_Quat_A`. An active left-multiply quaternion  ${}^a\mathbf{Q}_{1-2}$  is written `A_Quat_1_2`; a right-multiply quaternion  ${}_{1-2}\mathbf{Q}^a$  is written `1_2_Quat_A`. Any single program should use only left or right multiply, so this convention should not be confusing.

In the GN&C software naming convention a variable representing the passive right-multiply quaternion  ${}_b\mathbf{Q}^a$  is written `Quat_BTtoA`. The GN&C software naming convention has no notation for left-multiply or active quaternions.

### 2.3.3 Examples

See `spacecraft_math_examples.adb`, `Quaternions_Left` block for Ada code implementing these. Some of these examples are the same as in 2.2.3, but using left-multiply quaternions. We duplicate the descriptions here for convenience. The Ada code also shows the right-multiply equivalents.

First, an active rotation of a vector in the x-z plane in a positive direction about the y axis.

$$\begin{aligned} {}^{base}\mathbf{Q}_{1-2} &= \text{to\_unit\_quaternion}(\theta_{1-2}, {}^{base}\hat{\mathbf{n}}) \\ &= \text{to\_unit\_quaternion}(0.1, (0.0, 1.0, 0.0)) \\ &= (0.00000, 0.04998, 0.00000, 0.99875) \\ {}^{base}\mathbf{r}_1 &= (2.0, 0.0, 1.0) \\ {}^{base}\mathbf{r}_2 &= (2.08984, 0.00000, 0.79534) \end{aligned}$$

Note that the y component of the quaternion has the same sign as the angle. Note that the x component of  ${}^{base}\mathbf{r}_2$  increases, and the z component decreases; the end point of the vector is moved closer to the x axis.

Now we apply a second active rotation, and show that quaternion multiplication is associative:

$$\begin{aligned} {}^{base}\mathbf{Q}_{2-3} &= \text{to\_unit\_quaternion}(\pi/2, (1.0, 0.0, 0.0)) \\ &= (0.70711, 0.00000, 0.00000, 0.70711) \\ {}^{base}\mathbf{Q}_{1-3} &= {}^{base}\mathbf{Q}_{2-3} {}^{base}\mathbf{Q}_{1-2} \\ &= (0.70622, 0.03534, 0.03534, 0.70622) \\ {}^{base}\mathbf{r}_3 &= {}^{base}\mathbf{Q}_{2-3} {}^{base}\mathbf{r}_2 \\ &= (2.08984, -0.79534, 0.00000) \\ {}^{base}\mathbf{r}_3 &= {}^{base}\mathbf{Q}_{1-3} {}^{base}\mathbf{r}_1 \\ &= (2.08984, -0.79534, 0.00000) \end{aligned}$$

Now a spacecraft example; the direction of the z axis of the spacecraft. Assume the Sun is at the origin of the base frame, and the spacecraft is positioned on the

negative z axis of the base frame. Initially, the z axis of the spacecraft frame is pointing at the sun. First we rotate the spacecraft in a positive direction about the base y axis, then about the base x axis. Frame *sc0* gives the initial orientation of the spacecraft; it is parallel to frame *base*. Frame *sc1* gives the orientation of the spacecraft after the first rotation, *sc2* after the second. Unit vector  ${}^{base}\hat{\mathbf{n}}_{scz0}$  gives the direction of the z axis of the spacecraft in the base frame in the initial orientation;  ${}^{base}\hat{\mathbf{n}}_{scz1}$  after the first rotation.

$$\begin{aligned} {}^{base}\hat{\mathbf{n}}_{scz0} &= (0.0, 0.0, 1.0) \\ {}^{base}\mathbf{Q}_{sc0-sc1} &= \text{to\_unit\_quaternion}(0.1, (0.0, 1.0, 0.0)) \\ &= (0.00000, 0.04998, 0.00000, 0.99875) \\ {}^{base}\hat{\mathbf{n}}_{scz1} &= (0.09983, 0.00000, 0.99500) \end{aligned}$$

In the base frame, the z axis of the spacecraft has moved towards the positive x axis.

Now we apply a second rotation, about the base x axis.

$$\begin{aligned} {}^{base}\mathbf{R}_{sc1-sc2} &= \text{to\_unit\_quaternion}(\pi/2, (0.0, 1.0, 0.0)) \\ &= (0.70711, 0.00000, 0.00000, 0.70711) \\ {}^{base}\hat{\mathbf{n}}_{scz2} &= (0.09983, -0.99500, 0.00000) \end{aligned}$$

Now consider the direction from the spacecraft to the Sun; we know what it is in the base frame, and we need to find it in the spacecraft frame. The unit vector  $\hat{\mathbf{n}}_{sc-sun}$  gives the direction from the spacecraft to the sun, in a given frame. We use the same two rotations as above; we need passive quaternions for this, so the result of converting from active angle axis is inverted.

$$\begin{aligned} {}^{base}\hat{\mathbf{n}}_{sc-sun} &= (0.0, 0.0, 1.0) \\ {}^{sc0}\hat{\mathbf{n}}_{sc-sun} &= (0.0, 0.0, 1.0) \\ {}^{sc1}\mathbf{Q}_{base} &= (\text{to\_unit\_quaternion}(0.1, (0.0, 1.0, 0.0)))^{-1} \\ &= (0.00000, -0.04998, 0.00000, 0.99875) \\ {}^{sc1}\hat{\mathbf{n}}_{sc-sun} &= (-0.09983, 0.00000, 0.99500) \end{aligned}$$

In the spacecraft frame, the direction to the sun has moved towards the negative x axis.

Now we apply a second rotation, about the base x axis. We show the angle axis equivalent to each quaternion, since that may be easier to visualize. The Ada

code also shows that quaternion multiply is associative.

$$\begin{aligned}
{}^{sc1}\mathbf{Q}_{sc1-sc2} &= {}^{sc1}\mathbf{Q}_{base} {}^{base}\mathbf{Q}_{sc1-sc2} {}^{sc1}\mathbf{Q}_{base}^{-1} \\
&= (0.70357, -0.00000, 0.07059, 0.70711) \\
{}^{sc1}\mathbf{R}_{sc1-sc2} &= (1.57080, (0.99500, -0.00000, 0.09983)) \\
{}^{sc2}\mathbf{Q}_{sc1} &= {}^{sc1}\mathbf{Q}_{sc1-sc2}^{-1} \\
&= (-0.70357, 0.00000, -0.07059, 0.70711) \\
{}^{sc2}\mathbf{R}_{sc1} &= (1.57080, (-0.99500, 0.00000, -0.09983)) \\
{}^{sc2}\hat{\mathbf{n}}_{sc-sun} &= (0.00000, 1.00000, 0.00000)
\end{aligned}$$

### 2.3.4 Vector Scalar formulation

The equations for quaternion times quaternion and quaternion times vector can also be expressed in pure vector notation, without reference to any Cartesian frame.

First, we denote the vector and scalar parts of a quaternion by:

$$\mathbf{Q}_1 = (\mathbf{q}_{1v}, Q_{1s}) \quad (2.3.4-1)$$

Then (2.3.1-11) becomes

$$\begin{aligned}
\mathbf{Q}_3 &= \mathbf{Q}_1 \mathbf{Q}_2 \\
&= (\mathbf{q}_{2v} Q_{1s} + \mathbf{q}_{1v} Q_{2s} + \mathbf{q}_{1v} \times \mathbf{q}_{2v}, \\
&\quad Q_{1s} Q_{2s} - \mathbf{q}_{1v} \cdot \mathbf{q}_{2v})
\end{aligned} \quad (2.3.4-2)$$

This illustrates that there is a frame involved in a quaternion that is not shown in the quaternion notation; the reference frame of the vector part. However, the vector part is an eigenvector of the rotation (it is parallel to the axis of rotation), so it does not matter whether it is expressed in  $a$  or  $b$ .

To actively rotate a vector in this formulation, form a quaternion (not a unit quaternion) from the vector by setting the scalar part to zero. Then (2.3.1-9) becomes

$${}^a\mathbf{r}_2 = {}^a\mathbf{Q}_{1-2} {}^a\mathbf{r}_1 \quad (2.3.4-3)$$

$$= {}^a\mathbf{Q}_{1-2} ({}^a\mathbf{r}_1, 0) {}^a\mathbf{Q}_{1-2}^* \quad (2.3.4-4)$$

where  ${}^a\mathbf{Q}_{1-2}^*$  is the *conjugate quaternion*, and is the same as  ${}^a\mathbf{Q}_{1-2}^{-1}$ . The multiplication operator in (2.3.4-3) is given by (2.3.1-8); in (2.3.4-4), it is given by (2.3.4-2).

Expanding and applying vector and trig identities, this reduces to (2.2.1-5). See `derive_vect_scalar_quat.maxima` for details.

Similarly, for a passive rotation, (2.3.1-10) becomes

$$\begin{aligned} {}^b\mathbf{r}_1 &= {}^b\mathbf{Q}_a {}^a\mathbf{r}_1 \\ &= {}^b\mathbf{Q}_a ({}^a\mathbf{r}_1, 0) {}^b\mathbf{Q}_a^* \end{aligned} \quad (2.3.4-5)$$

When combining rotations, as in (2.1.1-4), this is:

$$\begin{aligned} {}^c\mathbf{Q}_a &= {}^c\mathbf{Q}_b {}^b\mathbf{Q}_a \\ {}^c\mathbf{r}_1 &= {}^c\mathbf{Q}_a ({}^a\mathbf{r}_1, 0) {}^c\mathbf{Q}_a^* \\ {}^c\mathbf{r}_1 &= {}^c\mathbf{Q}_b {}^b\mathbf{Q}_a ({}^a\mathbf{r}_1, 0) {}^b\mathbf{Q}_a^* {}^c\mathbf{Q}_b^* \end{aligned} \quad (2.3.4-6)$$

### Right-multiply

For right-multiply quaternions, the non-conjugate quaternion is on the right:

$$\begin{aligned} {}^a\mathbf{r}_2 &= {}^a\mathbf{r}_1 {}_{1-2}\mathbf{Q}^a \\ &= {}_{1-2}\mathbf{Q}^{a*} ({}^a\mathbf{r}_1, 0) {}_{1-2}\mathbf{Q}^a \end{aligned} \quad (2.3.4-7)$$

$$\begin{aligned} {}^b\mathbf{r}_1 &= {}^a\mathbf{r}_1 {}_a\mathbf{Q}^b \\ &= {}_a\mathbf{Q}^{b*} ({}^a\mathbf{r}_1, 0) {}_a\mathbf{Q}^b \end{aligned} \quad (2.3.4-8)$$

$$\begin{aligned} {}_a\mathbf{Q}^c &= {}_a\mathbf{Q}^b {}_b\mathbf{Q}^c \\ {}^c\mathbf{r}_1 &= {}_a\mathbf{Q}^{c*} ({}^a\mathbf{r}_1, 0) {}_a\mathbf{Q}^c \\ {}^c\mathbf{r}_1 &= {}_b\mathbf{Q}^{c*} {}_a\mathbf{Q}^{b*} ({}^a\mathbf{r}_1, 0) {}_a\mathbf{Q}^b {}_b\mathbf{Q}^c \end{aligned} \quad (2.3.4-9)$$

This is the motivation for the names *left-multiply*, *right-multiply* for quaternions.

### 2.3.5 Derivations

(2.3.1-3) is simply the definition of an active left-multiply rotation quaternion.

(2.3.1-4) is obtained from (2.3.1-3) by substituting  $-\theta_{1-2}$  for  $\theta_{1-2}$ .

To see that (2.3.1-5) gives the same rotation, note that  $Q_i$  always occurs in pairs in (2.3.1-8). Thus changing the sign of all  $Q_i$  has no effect on the rotated vector. Also compare this to (2.2.1-3).

Note that we could restrict  $Q_s \geq 0$ . However, enforcing that condition in every quaternion operation is inefficient; instead we only enforce it where it is useful, such as in conversion to angle axis.

(2.3.1-6) is derived from (2.2.1-4) and (2.3.1-5). We negate the vector components of the quaternion rather than the scalar component, because then the

inverse operation is the same as the *conjugate* operation, used in the vector-scalar formulation of quaternion times vector (see 2.3.4).

(2.3.1-7) is derived from (2.3.1-3) using the trigonometric identity  $\theta = \text{atan2}(\sin(\theta), \cos(\theta))$ .  $Q_s$  is forced positive to ensure that  $\theta_b$  is in the range  $[0, 2\pi)$ ; otherwise it would be in the range  $[0, 4\pi)$ . Care is taken to ensure that conversion from angle to axis to quaternion and back is identity for the angle in  $[0.0, \pi)$ .

(2.3.1-7) is singular when  $s$  is zero. To see how to handle the singularity, consider the quaternion  $(\epsilon, 0, 0, 1)$ , where  $\epsilon$  is the largest number such that  $1 + \epsilon = 1$  to machine precision. Note that this satisfies (2.3.1-2), since  $1 + \epsilon^2 = 1$ . Note that  $s = \epsilon$ , and (2.3.1-7) is well-behaved. Thus there is only a problem when  $s$  is identically zero. In that case, the magnitude of the rotation is zero, and the axis is undetermined, so we simply pick the x axis.

To prove that (2.3.1-8) is the equation for rotating a vector, substitute (2.3.1-3) into (2.3.1-8), and compare to (2.2.1-5). See `derive_quat_times_vect.maxima` for the details. Note that the temporary variable  $a$  provides a significant optimization; this was first published in [5].

To prove that (2.3.1-11) is the equation for combining rotations, we evaluate  $(Q_a Q_b)\mathbf{r}$  and  $Q_a(Q_b\mathbf{r})$  using (2.3.1-11) and (2.3.1-8), and show they are the same. See `derive_quat_times_quat.maxima` for the details.

To prove (2.3.1-15), start with a vector rotation, and change the frame of the vectors:

$$\begin{aligned} {}^a\mathbf{r}_2 &= {}^aQ_{b-c} {}^a\mathbf{r}_1 \\ {}^a\mathbf{r}_1 &= {}^bQ_a^{-1} {}^b\mathbf{r}_1 \\ {}^b\mathbf{r}_2 &= {}^bQ_a {}^a\mathbf{r}_2 \\ &= {}^bQ_a {}^aQ_{b-c} {}^a\mathbf{r}_1 \\ &= {}^bQ_a {}^aQ_{b-c} {}^bQ_a^{-1} {}^b\mathbf{r}_1 \\ {}^bQ_{b-c} &= {}^bQ_a {}^aQ_{b-c} {}^bQ_a^{-1} \end{aligned}$$

(2.3.1-16) is derived from (2.3.1-15) and (2.3.1-14), simplifying the inverses:

$$\begin{aligned} {}^cQ_b &= {}^bQ_{b-c}^{-1} \\ &= ({}^bQ_a {}^aQ_{b-c} {}^bQ_a^{-1})^{-1} \\ &= {}^bQ_a {}^aQ_{b-c}^{-1} {}^bQ_a^{-1} \end{aligned}$$

To prove (2.3.1-17), start by simply inverting the active quaternions:

$$\begin{aligned} {}^aQ_{a-c} &= {}^aQ_{b-c} {}^aQ_{a-b} \\ {}^aQ_{a-c}^{-1} &= {}^aQ_{a-b}^{-1} {}^aQ_{b-c}^{-1} \\ {}^cQ_a &= {}^bQ_a {}^aQ_{b-c}^{-1} \end{aligned}$$

To proceed from here, we need to get the second term into the form of (2.3.1-16). So we insert  ${}^b\mathbf{Q}_a^{-1}{}^b\mathbf{Q}_a$ , which is identity, and regroup:

$$\begin{aligned} {}^c\mathbf{Q}_a &= {}^b\mathbf{Q}_a({}^b\mathbf{Q}_a^{-1}{}^b\mathbf{Q}_a)^a\mathbf{Q}_{b-c}^{-1}({}^b\mathbf{Q}_a^{-1}{}^b\mathbf{Q}_a) \\ &= {}^b\mathbf{Q}_a{}^b\mathbf{Q}_a^{-1}({}^b\mathbf{Q}_a^a\mathbf{Q}_{b-c}^{-1}{}^b\mathbf{Q}_a^{-1}){}^b\mathbf{Q}_a \\ &= {}^c\mathbf{Q}_b{}^b\mathbf{Q}_a \end{aligned}$$

### Right-multiply

The right-multiply equations (2.3.1-20) thru (2.3.1-29) are obtained from the corresponding left multiply equations by consistently negating the vector components of the quaternions, and reversing the order of multiplied quaternions.

### 2.3.6 Comparison to Kane

[4] calls quaternion components ‘Euler parameters’, with symbol  $\epsilon$ . Set  $Q_x = \epsilon_1, Q_y = \epsilon_2, Q_z = \epsilon_3, Q_s = \epsilon_4$  to obtain our notation.

(2.3.1-3) is equivalent to [4, eqn 1.2(1)-(2)]; both are active rotations.

(2.3.1-9) is equivalent to [4, eqn 1.3(21)]. See `derive_quat_times_vect_kane.maxima` for details.

(2.3.1-12) is the same as [4, eqn 1.6(12)].

### 2.3.7 Comparison to Wertz

For historical reasons, this section compares left-multiply quaternions to Wertz. It was in the process of doing this comparison that I recognized the existence of right-multiply quaternions, and finally understood the origin of the sign difference between Kane and Wertz quaternions.

(2.3.4-2) is the same as [6, eqn (D-8)], with

$$\begin{aligned} q'' &= \mathbf{Q}_3 \\ q &= \mathbf{Q}_1 \\ q' &= \mathbf{Q}_2 \end{aligned}$$

(2.3.4-5) is equivalent to [6, eqn (D-11)], with  $\mathbf{U}' = {}^b\mathbf{r}_1, \mathbf{U} = {}^a\mathbf{r}_1, q = {}^b\mathbf{Q}_a^{-1}$ .

Thus (D-11) is equivalent to (2.3.1-10), if we set:

$$\begin{aligned} Q_x &= -q_1 \\ Q_y &= -q_2 \\ Q_z &= -q_3 \\ Q_s &= q_4 \end{aligned} \tag{2.3.7-1}$$

See `derive_quat_times_vect_wertz.maxima` for detailed proof.

Now consider combining passive rotations, as in (2.3.4-6). In Wertz's notation, this is:

$$\begin{aligned} q'' &= qq' \\ \mathbf{U}'' &= q''^* \mathbf{U} q'' \\ \mathbf{U}'' &= q'^* q^* \mathbf{U} q q' \end{aligned}$$

Thus, in our notation,  ${}^c\mathbf{Q}_b$  is the second passive rotation, and is on the left, while in Wertz,  $q'$  is the second passive rotation, and is on the right. This corresponds to whether the non-conjugate is on the left or the right in (2.3.4-5); Wertz uses the right-multiply convention.

(2.3.1-3) is equivalent to [6, eqns (12-11a) - (12-11d)], with (2.3.7-1) and  $\phi = \theta_b$ . Note that Wertz does not carefully define the direction of  $\phi$ .

## 2.4 Matrix rotation representation

### 2.4.1 Reference

A rotation matrix is a 3 by 3 orthonormal matrix, with determinant +1.

The inverse of a rotation matrix is given by the transpose:

$$\mathbf{M}^{-1} = \mathbf{M}^T \tag{2.4.1-1}$$

The active rotation matrix  ${}^a\mathbf{M}_{1-2}$  equivalent to the active angle axis rotation defined in (2.2.1-1) is given by:

$${}^a\mathbf{M}_{1-2} = \tag{2.4.1-2}$$

$$\begin{bmatrix} \cos\theta_{1-2} + n_x^2 \text{vers}\theta_{1-2} & -n_z \sin\theta_{1-2} + n_x n_y \text{vers}\theta_{1-2} & n_y \sin\theta_{1-2} + n_x n_z \text{vers}\theta_{1-2} \\ n_z \sin\theta_{1-2} + n_x n_y \text{vers}\theta_{1-2} & \cos\theta_{1-2} + n_y^2 \text{vers}\theta_{1-2} & -n_x \sin\theta_{1-2} + n_y n_z \text{vers}\theta_{1-2} \\ -n_y \sin\theta_{1-2} + n_x n_z \text{vers}\theta_{1-2} & n_x \sin\theta_{1-2} + n_y n_z \text{vers}\theta_{1-2} & \cos\theta_{1-2} + n_z^2 \text{vers}\theta_{1-2} \end{bmatrix}$$

where  $\text{vers}\theta_{1-2} = 1 - \cos\theta_{1-2}$ .

To obtain the angle axis from an active rotation matrix:

$$\begin{aligned} a &= \frac{\sqrt{(M_{xy} - M_{yx})^2 + (M_{zx} - M_{xz})^2 + (M_{yz} - M_{zy})^2}}{2} \\ b &= \frac{M_{xx} + M_{yy} + M_{zz} - 1}{2} \\ \theta_{1-2} &= \text{atan2}(a, b) \end{aligned}$$

if  $a > 0$  then

$${}^a \hat{\mathbf{n}} = \frac{(M_{yz} - M_{zy}, M_{zx} - M_{xz}, M_{xy} - M_{yx})}{2a} \quad (2.4.1-3)$$

else

$${}^a \hat{\mathbf{n}} = (1, 0, 0)$$

end if

To obtain the active rotation matrix  ${}^a \mathbf{M}_{1-2}$  equivalent to the active unit quaternion  ${}^a \mathbf{Q}_{1-2}$ :

$${}^a \mathbf{M}_{1-2} = \begin{bmatrix} 1 - 2Q_z^2 - 2Q_y^2 & 2Q_x Q_y - 2Q_s Q_z & 2Q_x Q_z + 2Q_s Q_y \\ 2Q_s Q_z + 2Q_x Q_y & 1 - 2Q_z^2 - 2Q_x^2 & 2Q_y Q_z - 2Q_s Q_x \\ 2Q_x Q_z - 2Q_s Q_y & 2Q_y Q_z + 2Q_s Q_x & 1 - 2Q_y^2 - 2Q_x^2 \end{bmatrix} \quad (2.4.1-4)$$

To obtain the active quaternion  ${}^a \mathbf{Q}_{1-2}$  from the active rotation matrix  ${}^a \mathbf{M}_{1-2}$ :

$$\begin{aligned} a &= M_{xx} + M_{yy} + M_{zz} \\ b &= M_{xx} - M_{yy} - M_{zz} \\ c &= -M_{xx} + M_{yy} - M_{zz} \\ d &= -M_{xx} - M_{yy} + M_{zz} \end{aligned} \quad (2.4.1-5)$$

if  $a > \max(b, \max(c, d))$  then

$$\begin{aligned} e &= 2\sqrt{1+a} \\ Q_x &= \frac{M_{zy} - M_{yz}}{e} \\ Q_y &= \frac{M_{xz} - M_{zx}}{e} \\ Q_z &= \frac{M_{yx} - M_{xy}}{e} \\ Q_s &= \frac{e}{4} \end{aligned}$$

elseif  $b > \max(a, \max(c, d))$  then

$$\begin{aligned} e &= 2\sqrt{1+b} \\ Q_x &= \frac{e}{4} \\ Q_y &= \frac{M_{xy} + M_{yx}}{e} \\ Q_z &= \frac{M_{xz} + M_{zx}}{e} \\ Q_s &= \frac{M_{zy} - M_{yz}}{e} \end{aligned}$$

elseif  $c > \max(a, \max(b, d))$  then

$$\begin{aligned} e &= 2\sqrt{1+c} \\ Q_x &= \frac{M_{xy} + M_{yx}}{e} \\ Q_y &= \frac{e}{4} \\ Q_z &= \frac{M_{yz} + M_{zy}}{e} \\ Q_s &= \frac{M_{xz} - M_{zx}}{e} \end{aligned}$$

else

$$\begin{aligned} e &= 2\sqrt{1+d} \\ Q_x &= \frac{M_{xz} + M_{zx}}{e} \\ Q_y &= \frac{M_{yz} + M_{zy}}{e} \\ Q_z &= \frac{e}{4} \\ Q_s &= \frac{M_{yx} - M_{xy}}{e} \end{aligned}$$

end if

It may be desirable to normalize the quaternion after applying (2.4.1-5), to eliminate roundoff errors, or small orthogonality errors in the original matrix.

To rotate a vector using a rotation matrix, use normal matrix multiply:

$$\mathbf{r}' = \mathbf{M}\mathbf{r} \quad (2.4.1-6)$$

If  $\mathbf{M}$  is an active matrix (ie, it represents an active rotation),  $\mathbf{r}'$  is a rotated vector:

$$\mathbf{r}' = {}^a\mathbf{r}_2 = {}^a\mathbf{M}_{1-2} {}^a\mathbf{r}_1 \quad (2.4.1-7)$$

If  $\mathbf{M}$  is a passive rotation,  $\mathbf{r}'$  is  $\mathbf{r}$  expressed in a different frame:

$$\mathbf{r}' = {}^b\mathbf{r}_1 = {}^b\mathbf{M}_a {}^a\mathbf{r}_1 \quad (2.4.1-8)$$

To multiply rotation matrices, use normal matrix multiply:

$$\mathbf{M}_c = \mathbf{M}_a \mathbf{M}_b \quad (2.4.1-9)$$

If the matrices are active, this represents two successive physical rotations:

$${}^a\mathbf{r}_3 = {}^a\mathbf{M}_{2-3} {}^a\mathbf{M}_{1-2} {}^a\mathbf{r}_1 \quad (2.4.1-10)$$

If the matrices are passive, this represents two successive coordinate transforms:

$${}^c\mathbf{r}_1 = {}^c\mathbf{M}_b {}^b\mathbf{M}_a {}^a\mathbf{r}_1 \quad (2.4.1-11)$$

### Right-multiply

Since the SAL math library does not provide right-multiply matrices, we do not present them here.

## 2.4.2 Coding

The type `SAL.Gen_Math.Gen_DOF_3.Rot_Matrix_Type` implements the orthonormal matrix rotation representation. The child packages `SAL.Gen_Math.Gen_DOF_3.Gen_Left` and `SAL.Gen_Math.Gen_DOF_3.Gen_Wertz` both provide left-multiply rotation matrix operations. Right-multiply rotation matrix operations are not provided.

A variable representing the passive left-multiply rotation matrix  ${}^a\mathbf{M}_b$  is written `A_Rot_Mat_B`.

In the GN&C convention the passive left multiply matrix  ${}^a\mathbf{M}_b$  is written `Dcm_BToA`.

`SAL.Math_Double.Dof_3.Left.To_Unit_Quaternion` implements (2.4.1-5); `SAL.Math_Double.Dof_3.Wertz.To_Unit_Quaternion` implements (2.4.1-5) with an additional inverse, so that the result represents the same rotation (the left multiply quaternion is the numerical inverse of the right multiply quaternion).

## 2.4.3 Examples

See `spacecraft_math_examples.adb`, `Rot_Matrix` block for Ada code implementing these. This example is the same as the first one in 2.3.3, but using left-multiply rotation matrices. Since the primary use of rotation matrices is to convert them to quaternions, the other examples are not relevant.

An active rotation of a vector in the x-z plane in a positive direction about the y axis.

$$\begin{aligned}
{}^{base}\mathbf{M}_{1-2} &= \text{to\_rot\_matrix}(\theta_{1-2}, {}^{base}\hat{\mathbf{n}}) \\
&= \text{to\_rot\_matrix}(0.1, (0.0, 1.0, 0.0)) \\
&= \begin{bmatrix} 0.99500 & 0.00000 & 0.09983 \\ 0.00000 & 1.00000 & 0.00000 \\ -0.09983 & 0.00000 & 0.99500 \end{bmatrix} \\
{}^{base}\mathbf{Q}_{1-2} &= \text{to\_unit\_quaternion}({}^{base}\mathbf{M}_{1-2}) \\
&= (0.00000, 0.04998, 0.00000, 0.99875) \\
{}^{base}\mathbf{r}_1 &= (2.0, 0.0, 1.0) \\
{}^{base}\mathbf{r}_2 &= {}^{base}\mathbf{M}_{1-2} {}^{base}\mathbf{r}_1 \\
&= (2.08984, 0.00000, 0.79534)
\end{aligned}$$

#### 2.4.4 Derivations

(2.4.1-6) is simply the definition of matrix multiplication.

(2.4.1-9) is the correct way to combine rotations, because of the associativity of matrix multiply.

(2.4.1-3) is derived from (2.2.1-5) by finding the coefficient matrix for  ${}^a\mathbf{r}_1$ ; see `derive_angle_axis_to_rot_matrix.maxima` for details.

To derive (2.4.1-3), start with (2.4.1-3), take various combinations of  $M_{ij}$ , and use  $n_x^2 + n_y^2 + n_z^2 = 1$ .

$$M_{xy} - M_{yx} = 2n_z \sin(\theta) \quad (2.4.4-1)$$

$$M_{zx} - M_{xz} = 2n_y \sin(\theta) \quad (2.4.4-2)$$

$$M_{yz} - M_{zy} = 2n_x \sin(\theta) \quad (2.4.4-3)$$

$$\begin{aligned}
M_{xx} + M_{yy} + M_{zz} &= 3\cos(\theta) + (n_x^2 + n_y^2 + n_z^2)\text{vers}(\theta) \\
&= 1 + 2\cos(\theta)
\end{aligned} \quad (2.4.4-4)$$

Now we can find  $\theta$  from (2.4.4-1), (2.4.4-2), (2.4.4-3), and (2.4.4-4):

$$\begin{aligned}
\sin(\theta) &= \frac{\sqrt{(M_{xy} - M_{yx})^2 + (M_{zx} - M_{xz})^2 + (M_{yz} - M_{zy})^2}}{2} \\
\cos(\theta) &= \frac{M_{xx} + M_{yy} + M_{zz} - 1}{2} \\
\theta &= \text{atan2}(\sin(\theta), \cos(\theta))
\end{aligned} \quad (2.4.4-5)$$

Note we have chosen  $\sin(\theta) \geq 0$ , so we have  $0 \leq \theta < \pi$ . The axis  $\mathbf{n}$  is found by:

$$\hat{\mathbf{n}} = \frac{(M_{yz} - M_{zy}, M_{zx} - M_{xz}, M_{xy} - M_{yx})}{2\sin(\theta)} \quad (2.4.4-6)$$

This is singular when  $\sin(\theta) = 0$ . Consider  $\theta = \epsilon$ ; then  $\cos(\theta) = 1$ ,  $\sin(\theta) = \epsilon$ , and (2.4.4-6) becomes:

$$\hat{\mathbf{n}} = \frac{(2n_x\epsilon, 2n_y\epsilon, 2n_z\epsilon)}{2\epsilon} \quad (2.4.4-7)$$

This is well behaved unless  $\sin(\theta)$  is exactly zero. In that case, the axis is indeterminate, so we simply choose the x axis. See `derive_rot_matrix_to_angle_axis.maxima` for details of the above derivation.

To derive (2.4.1-4), start with (2.3.1-8), and find the coefficients of each vector component; see `derive_quat_to_rot_matrix.maxima`.

To derive (2.4.1-5), start with the diagonal elements of (2.4.1-4), and (2.3.1-2); this gives us four equations in the four unknowns  $Q_i^2$ . We solve for one of them (call it  $Q_j$ ); then we can take various combinations of  $M_{ij}$  and divide by  $Q_j$  to find the remaining  $Q_i$ . We want the largest magnitude divisor, to avoid singularities. So first we need to decide which divisor is the largest.

$$\begin{aligned} 1 &= Q_s^2 + Q_x^2 + Q_y^2 + Q_z^2 \\ M_{xx} &= 1 - 2Q_z^2 - 2Q_y^2 \\ M_{yy} &= 1 - 2Q_z^2 - 2Q_x^2 \\ M_{zz} &= 1 - 2Q_y^2 - 2Q_x^2 \end{aligned} \quad (2.4.4-8)$$

To solve for  $Q_s$ :

$$\begin{aligned} M_{xx} + M_{yy} + M_{zz} &= 1 - 2Q_z^2 - 2Q_y^2 + \\ &\quad 1 - 2Q_z^2 - 2Q_x^2 + \\ &\quad 1 - 2Q_y^2 - 2Q_x^2 \\ &= 3 - 4(Q_x^2 + Q_y^2 + Q_z^2) \\ &= 3 - 4(1 - Q_s^2) \\ &= -1 + 4Q_s^2 \\ Q_s &= \sqrt{\frac{1 + M_{xx} + M_{yy} + M_{zz}}{4}} \end{aligned} \quad (2.4.4-9)$$

To solve for  $Q_x$ :

$$\begin{aligned} M_{xx} - M_{yy} - M_{zz} &= 1 - 2Q_z^2 - 2Q_y^2 - \\ &\quad 1 + 2Q_z^2 + 2Q_x^2 - \\ &\quad 1 + 2Q_y^2 + 2Q_x^2 \\ &= -1 + 4Q_x^2 \\ Q_x &= \sqrt{\frac{1 + M_{xx} - M_{yy} - M_{zz}}{4}} \end{aligned} \quad (2.4.4-10)$$

$Q_y$  and  $Q_z$  are similar. Thus to pick the largest magnitude divisor, we find the maximum of the four combinations of  $M_{ii}$ :

$$\begin{aligned} a &= M_{xx} + M_{yy} + M_{zz} \\ b &= M_{xx} - M_{yy} - M_{zz} \\ c &= -M_{xx} + M_{yy} - M_{zz} \\ d &= -M_{xx} - M_{yy} + M_{zz} \end{aligned} \quad (2.4.4-11)$$

Now we can take various combinations of  $M_{ij}$  and divide by  $Q_j$ . For example, to find  $Q_x$  when we have  $Q_s$ :

$$\begin{aligned} M_{zy} - M_{yz} &= 2Q_y Q_z + 2Q_s Q_x - (2Q_y Q_z - 2Q_s Q_x) \\ &= 4Q_s Q_x \\ Q_x &= \frac{M_{zy} - M_{yz}}{4Q_s} \end{aligned} \quad (2.4.4-12)$$

Similarly for the other  $Q_i$ . Note we can save some computations by first computing  $4Q_j$  instead of  $Q_j$ . See `derive_rot_matrix_to_quat.maxima` for full details.

## 2.4.5 Comparison to Kane

(2.4.1-8) corresponds to [4, eqn 1.2(9)]:

$${}^B v = {}^A v {}^A C^B$$

However, Kane uses the right-multiply convention for matrices. Thus  ${}^A C^B$  is related to our notation by  ${}^b \mathbf{M}_a = {}^A C^B$ .

(2.4.1-3) corresponds to [4, eqn 1.2(23)-(31)]. However, Kane's  $C$  is a passive matrix corresponding to an active angle axis rotation, using the right-multiply convention, while  ${}^a \mathbf{M}_b$  is an active matrix using the left-multiply convention. These two differences cancel, and we have  ${}^a \mathbf{M}_b = C$ .

(2.4.1-4) is similar to [4, eqn 1.3(6)-(14)]. However, Kane's  $C$  is a right-multiplying passive matrix, while our  ${}^a \mathbf{M}_b$  is a left-multiply active matrix. The quaternions are the same. Thus we have  $C = {}^a \mathbf{M}_b$ .

(2.4.1-5) is similar to [4, 1.3(6)-(14)]. In addition to the differences in the matrix convention, Kane does not consider the singularity.

Similarly, (2.4.1-11) is similar to [4, eqn 1.6(4)].

### 2.4.6 Comparison to Wertz

(2.4.1-8) is the same as [6, eqn (12-4)]. Note that Wertz's notation does not include frames.

(2.4.1-3) corresponds to [6, eqn (12-7a)], with  $\phi = \theta_{1-2}$ , except that Wertz's  $A$  is an active left-multiply matrix (see figure 12-2), so the matrix is inverted (transposed).

(2.4.1-3) is similar to [6, eqn (12-9) - (12-10c)], but Wertz does not consider the singularity.

(2.4.1-4) is equivalent to [6, eqn (12-13a)], with (2.3.7-1). Note that Wertz's  $q$  is right-multiply, while our  ${}^a\mathbf{Q}_b$  is left-multiply; the matrices are both left-multiply.

(2.4.1-5) is equivalent to [6, eqns (12-14a) - 12-14d)], with (2.3.7-1), and we have slightly optimized the handling of the singularity.

## 2.5 Rotation vector representation

### 2.5.1 Reference

A rotation vector combines the magnitude and axis information into one vector:

$$\boldsymbol{\theta} = \theta \hat{\mathbf{n}} \quad (2.5.1-1)$$

where  $\theta$  is the magnitude of the rotation, and  $\hat{\mathbf{n}}$  is the rotation axis. This is useful mainly for differential rotations, since then the rotation vectors can simply be added.

To find the rotation vector from a quaternion or rotation matrix:

$$\begin{aligned} (\theta, \hat{\mathbf{n}}) &= \text{to\_angle\_axis}(\mathbf{R}) \\ \boldsymbol{\theta} &= \theta \hat{\mathbf{n}} \end{aligned} \quad (2.5.1-2)$$

where `to_angle_axis` is either (2.3.1-7) or (2.4.1-3).

To find a quaternion or rotation matrix from a rotation vector:

$$\begin{aligned} \theta &= |\boldsymbol{\theta}| \\ \hat{\mathbf{n}} &= \boldsymbol{\theta} / \theta \\ \mathbf{Q} &= \text{to\_unit\_quaternion}(\theta, \hat{\mathbf{n}}) & (2.5.1-3) \\ \mathbf{M} &= \text{to\_rot\_matrix}(\theta, \hat{\mathbf{n}}) & (2.5.1-4) \end{aligned}$$

where `to_unit_quaternion` is given by (2.3.1-3), and `to_rot_matrix` is given by (2.4.1-3). (2.5.1-3) defines the function `rotvect_to_quat`.

An important case is obtaining the rotation due to a rotational velocity acting over a finite time; see section 2.6.1.

## 2.5.2 Coding

The type `SAL.Gen_Math.DOF_3.Cart_Vector_Type` is used for the rotation vector representation; there are functions to convert between rotation vectors and quaternions.

A variable representing the active rotation vector  ${}^a\theta_{1-2}$  is written `A_Rot_Vect_1_2` when using the left-multiply convention, `1_2_Rot_Vect_A` when using the right-multiply convention.

## 2.6 Rate

### 2.6.1 Reference

The time rate of change of the position of an object is called the “translation rate” or “velocity” of an object. It is defined by the time rate of change of the position of a point fixed in the object (normally the origin of the object’s coordinate frame). The translation rate is measured relative to a base frame, and is expressed in a coordinate frame. To indicate all these frames, we use the notation  ${}_{base}^{coord}\mathbf{v}_{obj}$ .

Often we need to deal with velocities measured in frames that are moving with respect to each other. Assume frame **b** is moving with velocity  ${}^a\mathbf{v}_b$  with respect to frame **a**; frames **a** and **b** are parallel. The velocity  ${}^a\mathbf{v}_{obj}$  with respect to frame **a** given the velocity with respect to frame **b** is computed by:

$${}^a\mathbf{v}_{obj} = {}^b\mathbf{v}_{obj} + {}^a\mathbf{v}_b \quad (2.6.1-1)$$

However, if the velocities involved are a significant fraction of the speed of light,

this must be replaced by the Lorentz transform:

$$\begin{aligned}
{}^a_b \mathbf{v}_{obj||} &= \frac{{}^a_b \mathbf{v}_{obj} \cdot {}^a_b \mathbf{v}_b}{{}^a_b \mathbf{v}_b \cdot {}^a_b \mathbf{v}_b} {}^a_b \mathbf{v}_b \\
{}^a_b \mathbf{v}_{obj\perp} &= {}^a_b \mathbf{v}_{obj} - {}^a_b \mathbf{v}_{obj||} \\
\gamma &= \sqrt{1 - \frac{{}^a_b \mathbf{v}_b \cdot {}^a_b \mathbf{v}_b}{c^2}} \\
{}^a_b \mathbf{v}_{obj} &= \frac{{}^a_b \mathbf{v}_b + {}^a_b \mathbf{v}_{obj||} + \gamma {}^a_b \mathbf{v}_{obj\perp}}{1 + \frac{{}^a_b \mathbf{v}_{obj} \cdot {}^a_b \mathbf{v}_b}{c^2}}
\end{aligned} \tag{2.6.1-2}$$

see section 4.1.1 for one use of this.

The time rate of change of orientation of an object is called the “rotation rate” of an object. It is defined by the time rate of change of an object frame (rigidly attached to the object). An object rate is measured relative to a base frame, and is expressed in a coordinate frame. To indicate all these frames, we use the notation  ${}^{ref}_{base} \boldsymbol{\omega}_{obj}$ .  $\boldsymbol{\omega}$  is called a “rotation velocity vector”. Usually, the base frame is an inertial frame, and is not specified.

In order to integrate the rotation position of a body over time, or to compute the rotation rate from a sequence of rotation positions, we need to show how to relate  $\boldsymbol{\omega}$  to the time derivative of the orientation of an object.

In the SAL convention the orientation of an object is represented by a passive quaternion that transforms vectors in the object frame to the base frame:

$${}^{base} \mathbf{r}_a = {}^{base} \mathbf{Q}_{obj} {}^{obj} \mathbf{r}_a \tag{2.6.1-3}$$

In the GNC (right-multiply) convention, the orientation of an object is represented by  ${}^{base} \mathbf{Q}^{obj}$ .

If the rotation rate of the object  ${}^{obj} \boldsymbol{\omega}_{obj}$  is constant over a finite time, the resultant active rotation of the object from time  $t_1$  to time  $t_2$  is:

$${}^{obj} \boldsymbol{\theta}_{t_1-t_2} = {}^{obj} \boldsymbol{\omega}_{obj} (t_2 - t_1) \tag{2.6.1-4}$$

### Left-multiply

A passive left-multiply quaternion equivalent to the active rotation  ${}^{obj} \boldsymbol{\theta}_{t_1-t_2}$  is (see (2.5.1-3)):

$${}^{t_1} \mathbf{Q}_{t_2} = \text{rotvect\_to\_quat}({}^{obj} \boldsymbol{\theta}_{t_1-t_2}) \tag{2.6.1-5}$$

Note that the reference frame of the rotation vector must be the object frame at time  $t_1$  or  $t_2$  for this operation to be valid; see (2.3.1-15).

A series of finite passive rotations can be combined to give a sequence of object orientations:

$${}^{base}\mathbf{Q}_{t_2} = {}^{base}\mathbf{Q}_{t_1} {}^{t_1}\mathbf{Q}_{t_2} \quad (2.6.1-6)$$

$${}^{base}\mathbf{Q}_{t_3} = {}^{base}\mathbf{Q}_{t_2} {}^{t_2}\mathbf{Q}_{t_3} \quad (2.6.1-7)$$

Then we can compute the rate  ${}^{obj}\boldsymbol{\omega}_{obj}$  from the sequence:

$${}^{obj}\boldsymbol{\omega}_{obj} = \text{quat\_to\_rotvect}({}^{base}\mathbf{Q}_{t_1}^{-1} {}^{base}\mathbf{Q}_{t_2}) / (t_2 - t_1) \quad (2.6.1-8)$$

$\boldsymbol{\omega}$  behaves like a vector under coordinate transforms:

$${}^b\boldsymbol{\omega}_b = {}^a\mathbf{Q}_b^{-1} {}^a\boldsymbol{\omega}_a \quad (2.6.1-9)$$

Note that since frames  $a$  and  $obj$  are both rigidly attached to the same object at the same point, we can simply change the label on  ${}^a\boldsymbol{\omega}_{obj}$ .

The translational velocity of a point on the rotating body is given by:

$${}^{obj}\mathbf{v}_1 = {}^{obj}\boldsymbol{\omega}_{obj} \times {}^{obj}\mathbf{r}_1 \quad (2.6.1-10)$$

This can be expressed as a left-multiply matrix operator:

$$\begin{aligned} {}^a\mathbf{v}_1 &= {}^a\mathbf{r}_{1 \times} {}^a\boldsymbol{\omega}_a \\ {}^a\mathbf{r}_{1 \times} &= \begin{bmatrix} 0 & r_z & -r_y \\ -r_z & 0 & r_x \\ r_y & -r_x & 0 \end{bmatrix} \end{aligned} \quad (2.6.1-11)$$

### Right-multiply

SAL convention:

$${}_{t_2}\mathbf{Q}^{t_1} = \text{rotvect\_to\_quat}({}^{obj}\boldsymbol{\theta}_{t_1-t_2}) \quad (2.6.1-12)$$

$${}_{t_2}\mathbf{Q}^{base} = {}_{t_2}\mathbf{Q}^{t_1} {}_{t_1}\mathbf{Q}^{base} \quad (2.6.1-13)$$

$${}^{obj}\boldsymbol{\omega}_{obj} = \text{quat\_to\_rotvect}({}_{t_2}\mathbf{Q}^{base} {}_{t_1}\mathbf{Q}^{base^{-1}}) / (t_2 - t_1) \quad (2.6.1-14)$$

GNC convention:

$${}_{t_1}\mathbf{Q}^{t_2} = \text{rotvect\_to\_quat}(-{}^{obj}\boldsymbol{\theta}_{t_1-t_2}) \quad (2.6.1-15)$$

$${}^{base}\mathbf{Q}^{t_2} = {}^{base}\mathbf{Q}^{t_1} {}_{t_1}\mathbf{Q}^{t_2} \quad (2.6.1-16)$$

$${}^{obj}\boldsymbol{\omega}_{obj} = -\text{quat\_to\_rotvect}({}^{base}\mathbf{Q}^{t_1^{-1}} {}^{base}\mathbf{Q}^{t_2}) / (t_2 - t_1) \quad (2.6.1-17)$$

Since  ${}^a\mathbf{r}_{1 \times}$  is a left-multiply matrix, and the SAL library does not provide right-multiply matrices, there is no right-multiply version of  ${}^a\mathbf{r}_{1 \times}$ .

### 2.6.2 Coding

A variable representing  ${}^{frame}\boldsymbol{\omega}_{obj}$  is written `Frame_Rot_Rate_Obj` in the left-multiply convention, `Obj_Rot_Rate_Frame` in right-multiply.

A variable representing  ${}^{coord}_{base}\mathbf{v}_{obj}$  is written `Coord_Vel_Obj_Wrt_Base` in the left multiply convention, `Obj_Wrt_Base_Vel_Coord` in right-multiply. “wrt” means “with respect to”.

The function `SAL.Gen_Math.Gen_DOF_3.Lorentz_Transform` implements (2.6.1-2).

### 2.6.3 Examples

See `spacecraft_math_examples.adb`, `Rotation_Rate_Left` block for Ada code implementing these.

To clarify the difference between active and passive rotations, we relate all rotations to the boresight of a star tracker mounted on a spacecraft. First we present the example using SAL convention left-multiply quaternions, then SAL and GNC orientation conventions for right-multiply quaternions. Note that the boresight vector, and the spacecraft rate vector, are the same for all three representations, but the quaternions are different.

We use the International Celestial Reference Frame (ICRF) as the base frame for this example. In the star tracker frame  $st$ , the boresight  ${}^{st}\hat{\mathbf{n}}_{bore}$  is a constant. Assume the initial star tracker orientation is  ${}^{ICRF}\mathbf{Q}_{st,t1}$ ; a passive quaternion. Then rotate it by 0.1 radians about the X axis in 0.5 seconds (an active rotation), to orientation  ${}^{ICRF}\mathbf{Q}_{st,t2}$ , and compute the boresight in ICRF in both

orientations:

$$\begin{aligned}
{}^{st}\hat{\mathbf{n}}_{bore} &= (0.0, 0.0, 1.0) \\
ICRF \mathbf{Q}_{st,t_1} &= (0.04998, 0.00000, 0.00000, 0.99875) \\
ICRF \hat{\mathbf{n}}_{bore,t_1} &= ICRF \mathbf{Q}_{st,t_1} {}^{st}\hat{\mathbf{n}}_{bore} \\
&= (0.00000, -0.09983, 0.99500) \\
\Delta\theta &= 0.1 \\
(t_2 - t_1) &= 0.5 \\
{}^{st}\boldsymbol{\omega}_{st} &= (\Delta\theta/(t_2 - t_1), 0.0, 0.0) \\
&= (0.20000, 0.00000, 0.00000) \\
ICRF \mathbf{Q}_{st,t_1-t_2} &= \text{rotvect\_to\_quat}({}^{st}\boldsymbol{\omega}_{st}(t_2 - t_1)) \\
&= (0.04998, 0.00000, 0.00000, 0.99875) \\
&= {}^{st,t_1}\mathbf{Q}_{st,t_2} \\
ICRF \mathbf{Q}_{st,t_2} &= ICRF \mathbf{Q}_{st,t_1} {}^{st,t_1}\mathbf{Q}_{st,t_2} \\
&= (0.09983, 0.00000, 0.00000, 0.99500) \\
ICRF \hat{\mathbf{n}}_{bore,t_2} &= ICRF \mathbf{Q}_{st,t_2} {}^{st}\hat{\mathbf{n}}_{bore} \\
&= (0.00000, -0.19867, 0.98007)
\end{aligned}$$

Now we have two ways to compute the rate; from the two boresight unit vectors, and from the two orientation quaternions:

$$\begin{aligned}
{}^{st,t_1}\mathbf{Q}_{st,t_2,a} &= ICRF \mathbf{Q}_{st,t_1-t_2} \\
&= \text{units\_to\_quat}({}^{ICRF}\hat{\mathbf{n}}_{bore,t_1}, {}^{ICRF}\hat{\mathbf{n}}_{bore,t_2}) \\
&= (0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st,t_1}\mathbf{Q}_{st,t_2,b} &= ICRF \mathbf{Q}_{st,t_1}^{-1} ICRF \mathbf{Q}_{st,t_2} \\
&= (0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st}\boldsymbol{\omega}_{st} &= \text{quat\_to\_rotvect}({}^{st,t_1}\mathbf{Q}_{st,t_2})/(t_2 - t_1) \\
&= (0.20000, 0.00000, 0.00000)
\end{aligned}$$

SAL right-multiply:

$$\begin{aligned}
{}^{st}\hat{\mathbf{n}}_{bore} &= (0.0, 0.0, 1.0) \\
{}^{st,t_1}\mathbf{Q}^{ICRF} &= (-0.04998, 0.00000, 0.00000, 0.99875) \\
ICRF\hat{\mathbf{n}}_{bore,t_1} &= {}^{st}\hat{\mathbf{n}}_{bore} {}^{st,t_1}\mathbf{Q}^{ICRF} \\
&= (0.00000, -0.09983, 0.99500) \\
{}^{st}\boldsymbol{\omega}_{st} &= (0.20000, 0.00000, 0.00000) \\
{}^{st,t_1-t_2}\mathbf{Q}^{ICRF} &= \text{rotvect\_to\_quat}({}^{obj}\boldsymbol{\omega}_{obj}(t_2 - t_1)) \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
&= {}^{st,t_2}\mathbf{Q}^{st,t_1} \\
{}^{st,t_2}\mathbf{Q}^{ICRF} &= {}^{st,t_2}\mathbf{Q}^{st,t_1} {}^{st,t_1}\mathbf{Q}^{ICRF} \\
&= (-0.09983, 0.00000, 0.00000, 0.99500) \\
ICRF\hat{\mathbf{n}}_{bore,t_2} &= {}^{st}\hat{\mathbf{n}}_{bore} {}^{st,t_2}\mathbf{Q}^{ICRF} \\
&= (0.00000, -0.19867, 0.98007) \\
{}^{st,t_2,a}\mathbf{Q}^{st,t_1} &= {}^{st,t_1-t_2}\mathbf{Q}^{ICRF} \\
&= \text{units\_to\_quat}(ICRF\hat{\mathbf{n}}_{bore,t_1}, ICRF\hat{\mathbf{n}}_{bore,t_2}) \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st,t_2,b}\mathbf{Q}^{st,t_1} &= {}^{st,t_2}\mathbf{Q}^{ICRF} {}^{st,t_1}\mathbf{Q}^{ICRF^{-1}} \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st}\boldsymbol{\omega}_{st} &= \text{quat\_to\_rotvect}({}^{st,t_2}\mathbf{Q}^{st,t_1})/(t_2 - t_1) \\
&= (0.20000, 0.00000, 0.00000)
\end{aligned}$$

GNC right-multiply:

$$\begin{aligned}
{}^{st}\hat{\mathbf{n}}_{bore} &= (0.0, 0.0, 1.0) \\
ICRF\mathbf{Q}^{st,t_1} &= (0.04998, 0.00000, 0.00000, 0.99875) \\
ICRF\hat{\mathbf{n}}_{bore,t_1} &= {}^{st}\hat{\mathbf{n}}_{bore} ICRF\mathbf{Q}^{st,t_1^{-1}} \\
&= (0.00000, -0.09983, 0.99500) \\
{}^{st}\boldsymbol{\omega}_{st} &= (0.20000, 0.00000, 0.00000)
\end{aligned}$$

$$\begin{aligned}
{}^{st,t_1-t_2}\mathbf{Q}^{ICRF} &= \text{rotvect\_to\_quat}(\text{obj}\boldsymbol{\omega}_{\text{obj}}(t_2 - t_1)) \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
&= {}^{st,t_2}\mathbf{Q}^{st,t_1} \\
ICRF\mathbf{Q}^{st,t_2} &= ICRF\mathbf{Q}^{st,t_1} {}^{st,t_1}\mathbf{Q}^{st,t_2} \\
&= (-0.09983, 0.00000, 0.00000, 0.99500) \\
ICRF\hat{\mathbf{n}}_{\text{bore},t_2} &= {}^{st}\hat{\mathbf{n}}_{\text{bore}} ICRF\mathbf{Q}^{st,t_2}{}^{-1} \\
&= (0.00000, -0.19867, 0.98007) \\
{}^{st,t_2,a}\mathbf{Q}^{st,t_1} &= {}^{st,t_1-t_2}\mathbf{Q}^{ICRF} \\
&= \text{units\_to\_quat}(ICRF\hat{\mathbf{n}}_{\text{bore},t_1}, ICRF\hat{\mathbf{n}}_{\text{bore},t_2}) \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st,t_2,b}\mathbf{Q}^{st,t_1} &= ICRF\mathbf{Q}^{st,t_2}{}^{-1} ICRF\mathbf{Q}^{st,t_1} \\
&= (-0.04998, 0.00000, 0.00000, 0.99875) \\
{}^{st}\boldsymbol{\omega}_{st} &= \text{quat\_to\_rotvect}({}^{st,t_2}\mathbf{Q}^{st,t_1})/(t_2 - t_1) \\
&= (0.20000, 0.00000, 0.00000)
\end{aligned}$$

Now we demonstrate using the rotation rate to find the translation rate of the lens of the star tracker, and show how both change under coordinate rotation. First we transform the rotation velocity to the ICRF frame. This is a special case, since the axis of rotation of the body is also the axis of rotation in this transform. There are two ways to compute the translation velocity in the second frame; rotate the velocity from the first frame, and use (2.6.1-10). We show that they give the same results.

$$\begin{aligned}
{}^{st}\mathbf{r}_{st-lens} &= (0.00000, 0.00000, 0.20000) \\
{}^{st}\mathbf{v}_{lens} &= (0.00000, -0.04000, 0.00000) \\
ICRF\boldsymbol{\omega}_{st,t_1} &= (0.20000, 0.00000, 0.00000) \\
ICRF\mathbf{v}_{lens,t_1,a} &= ICRF\mathbf{Q}_{st,t_1} {}^{st}\mathbf{v}_{lens} \\
&= (0.00000, -0.03980, -0.00399) \\
ICRF\mathbf{r}_{st-lens} &= (0.00000, -0.01997, 0.19900) \\
ICRF\mathbf{v}_{lens,t_1,b} &= ICRF\boldsymbol{\omega}_{st,t_1} \times ICRF\mathbf{r}_{st-lens} \\
&= (0.00000, -0.03980, -0.00399)
\end{aligned}$$

For a more general case, we introduce the spacecraft frame  $sc$ . For this example, the star tracker is rigidly attached to the spacecraft at the spacecraft frame

origin; see 3.2.3 for the case where translation is included.

$$\begin{aligned}
 {}^{sc}\mathbf{Q}_{st} &= \text{To\_Unit\_Quaternion}(\pi/2, Z) \\
 &= (0.00000, 0.00000, 0.70711, 0.70711) \\
 {}^{sc}\boldsymbol{\omega}_{sc} &= (0.00000, 0.20000, 0.00000) \\
 {}^{sc}\mathbf{v}_{lens,a} &= {}^{sc}\mathbf{Q}_{st} {}^{st}\mathbf{v}_{lens} \\
 &= (0.04000, 0.00000, 0.00000) \\
 {}^{sc}\mathbf{r}_{st-lens} &= (0.00000, 0.00000, 0.20000) \\
 {}^{sc}\mathbf{v}_{lens,b} &= {}^{sc}\boldsymbol{\omega}_{sc} \times {}^{sc}\mathbf{r}_{st-lens} \\
 &= (0.04000, 0.00000, 0.00000)
 \end{aligned}$$

### 2.6.4 Derivation

(2.6.1-1) is just classical velocity transformation; see any basic physics textbook.

(2.6.1-2) is the special relativity velocity transform; see an advanced physics textbook. For an accessible reference, see

[http://en.wikipedia.org/wiki/Velocity-addition\\_formula](http://en.wikipedia.org/wiki/Velocity-addition_formula).

To prove (2.6.1-8), start with (2.5.1-3) and invert rotvect\_to\_quat:

$$\begin{aligned}
 {}^{obj}\boldsymbol{\omega}_{obj} &= -\text{quat\_to\_rotvect}({}^{t_2}\mathbf{Q}_{t_1})/(t_2 - t_1) \\
 {}^{t_2}\mathbf{Q}_{t_1} &= {}^{base}\mathbf{Q}_{t_2}^{-1} {}^{base}\mathbf{Q}_{t_1} \\
 {}^{obj}\boldsymbol{\omega}_{obj} &= -\text{quat\_to\_rotvect}({}^{base}\mathbf{Q}_{t_2}^{-1} {}^{base}\mathbf{Q}_{t_1})/(t_2 - t_1)
 \end{aligned}$$

For another derivation relating quaternion difference to rotation rate, see Wertz, section 16.1.1.

See `derive_rate_transform.maxima` for detailed proof of (2.6.1-11).



## Chapter 3

# 6 Degrees of Freedom

The 6 degrees of freedom of a body (position and orientation) can be represented in one object called a “pose”.

Similarly, the force and torque are represented by an object called a “wrench”.

### 3.1 Poses

#### 3.1.1 Reference

Just as rotations can be either active or passive, translations can be as well. An active translation moves an object to a new position, relative to the same base position; a passive translation changes the base position. All translations are expressed in the same frame  $f$ :

$${}^f_b\mathbf{r}_1 = {}^f_a\mathbf{r}_1 - {}^f_a\mathbf{r}_b \quad (3.1.1-1)$$

$${}^f_a\mathbf{r}_2 = {}^f_a\mathbf{r}_1 + {}^f_a\mathbf{r}_{1-2} \quad (3.1.1-2)$$

Note that negation reverses the direction:

$${}^f_b\mathbf{r}_a = - {}^f_a\mathbf{r}_b \quad (3.1.1-3)$$

#### Left-multiply

In the SAL convention, the pose of an object relative to a base frame is represented by the passive transform  ${}^{base}\mathbf{T}_{obj}$  that transforms poses in the object

frame to the base frame:

$${}^{base}\mathbf{T}_a = {}^{base}\mathbf{T}_{obj} {}^{obj}\mathbf{T}_a \quad (3.1.1-4)$$

The base frame is also the reference frame for expressing the components of the transform. When used in this way, the transform is also called a pose.

The GNC convention does not define poses.

A pose consists of a translation part and a rotation part:

$${}^a\mathbf{T}_b = ({}^a\mathbf{r}_b; {}^a\mathbf{R}_b) \quad (3.1.1-5)$$

Transform and pose multiplication is defined by:

$$\begin{aligned} {}^a\mathbf{T}_c &= ({}^a\mathbf{r}_c; {}^a\mathbf{R}_c) \\ &= {}^a\mathbf{T}_b {}^b\mathbf{T}_c \\ &= ({}^a\mathbf{r}_b + {}^a\mathbf{R}_b {}^b\mathbf{r}_c; {}^a\mathbf{R}_b {}^b\mathbf{R}_c) \end{aligned} \quad (3.1.1-6)$$

The inverse transform is given by:

$$\begin{aligned} {}^a\mathbf{T}_b^{-1} &= {}^b\mathbf{T}_a \\ &= ({}^b\mathbf{r}_a; {}^b\mathbf{R}_a) \\ &= (-{}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_b; {}^a\mathbf{R}_b^{-1}) \end{aligned} \quad (3.1.1-7)$$

An important optimization is multiplying a pose by the inverse of a second pose:

$$\begin{aligned} {}^a\mathbf{T}_c &= {}^b\mathbf{T}_a^{-1} {}^b\mathbf{T}_c \\ &= ({}^b\mathbf{R}_a^{-1} ({}^b\mathbf{r}_c - {}^b\mathbf{r}_a); {}^b\mathbf{R}_a^{-1} {}^b\mathbf{R}_c) \end{aligned} \quad (3.1.1-8)$$

A differential pose can be represented by a dual Cartesian vector; the differential unit quaternion is converted to a rotation vector. The conversion of a pose to a dual Cartesian vector is represented by the function `pose_to_DCV`, and its inverse by `DVC_to_pose`:

$$\begin{aligned} {}^{base}\mathbf{p}_{obj} &= \text{pose\_to\_DCV}({}^{base}\mathbf{T}_{obj}) \\ &= ({}^{base}\mathbf{r}_{obj}; \text{quat\_to\_rotvect}({}^{base}\mathbf{R}_{obj})) \end{aligned} \quad (3.1.1-9)$$

$$\begin{aligned} {}^{base}\mathbf{T}_{obj} &= \text{DVC\_to\_pose}({}^{base}\mathbf{p}_{obj}) \\ &= ({}^{base}\mathbf{r}_{obj}; \text{rotvect\_to\_quat}({}^{base}\boldsymbol{\theta}_{obj})) \end{aligned} \quad (3.1.1-10)$$

### Right-multiply

The notation for poses when the rotation part is a right-multiply operator is:

$${}_{obj}\mathbf{T}^{ref} = ({}^{base}\mathbf{r}_{obj}; {}_{obj}\mathbf{R}^{base}) \quad (3.1.1-11)$$

Transform and pose multiplication is defined by:

$$\begin{aligned}
{}_c\mathbf{T}^a &= ({}^a\mathbf{r}_c; {}_c\mathbf{R}^a) \\
&= {}_c\mathbf{T}^b {}_b\mathbf{T}^a \\
&= ({}^b\mathbf{r}_c {}_b\mathbf{R}^a + {}^a\mathbf{r}_b; {}_c\mathbf{R}^b {}_b\mathbf{R}^a)
\end{aligned} \tag{3.1.1-12}$$

(3.1.1-7) and (3.1.1-8) become:

$$\begin{aligned}
{}_b\mathbf{T}^{a-1} &= {}_a\mathbf{T}^b \\
&= (-{}^a\mathbf{r}_b {}_b\mathbf{R}^{a-1}; {}_b\mathbf{R}^{a-1})
\end{aligned} \tag{3.1.1-13}$$

$$\begin{aligned}
{}_c\mathbf{T}^a &= {}_c\mathbf{T}^b {}_b\mathbf{T}^a \\
&= (({}^b\mathbf{r}_c - {}^b\mathbf{r}_a) {}_a\mathbf{R}^{b-1}; {}_c\mathbf{R}^b {}_a\mathbf{R}^{b-1})
\end{aligned} \tag{3.1.1-14}$$

### 3.1.2 Coding

The type `SAL.Gen_Math.DOF_6.Pose_Type` implements the pose representation.

The child package `SAL.Gen_Math.Gen_DOF_6.Gen_Left` provides SAL convention pose operations using left-multiply rotations, the child package `SAL.Gen_Math.Gen_DOF_6.Gen_Wertz` provides SAL convention pose operations using right-multiply rotations.

For left-multiply, a variable representing  ${}^a\mathbf{T}_b$  is written `A_Pose_B`.

For right-multiply, a variable representing  ${}_b\mathbf{T}^a$  is written `B_Pose_A`.

### 3.1.3 Examples

A typical use of poses is with articulated mechanisms (see 5). Suppose we have an antenna mounted on a single gimbal joint. We know the pose of the joint relative to the spacecraft frame  ${}^{BcsF}\mathbf{T}_{gimbal_0}$ , and the pose of the antenna relative to the joint is  ${}^{gimbal_0}\mathbf{T}_{gimbal_1}$ ; this includes the joint rotation about the X axis of the joint frame, and the translation from the joint to the antenna tip. Then the pose of the antenna tip relative to the spacecraft frame is:

$$\begin{aligned}
{}^{BcsF}\mathbf{T}_{gimbal_0} &= ((0.0, 2.0, 0.0), \text{To_Unit_Quaternion}(\pi/2.0, Y)) \\
&= ((0.0, 2.0, 0.0), (0.00000, 0.70711, 0.00000, 0.70711)) \\
{}^{gimbal_0}\mathbf{T}_{gimbal_1} &= ((1.0, 0.0, 0.0), \text{To_Unit_Quaternion}(\pi/2.0, X)) \\
&= ((1.0, 0.0, 0.0), (0.70711, 0.00000, 0.00000, 0.70711)) \\
{}^{BcsF}\mathbf{T}_{gimbal_1} &= {}^{BcsF}\mathbf{T}_{gimbal_0} {}^{gimbal_0}\mathbf{T}_{gimbal_1} \\
&= ((0.00000, 2.00000, -1.00000), (0.50000, 0.50000, -0.50000, 0.50000))
\end{aligned}$$

Right multiply:

$$\begin{aligned}
gimbal_0 \mathbf{T}^{BcsF} &= ((0.0, 2.0, 0.0), \text{To\_Unit\_Quaternion}(\pi/2.0, Y)) \\
&= ((0.0, 2.0, 0.0), (0.00000, -0.70711, 0.00000, 0.70711)) \\
gimbal_1 \mathbf{T}^{gimbal_0} &= ((1.0, 0.0, 0.0), \text{To\_Unit\_Quaternion}(\pi/2.0, X)) \\
&= ((1.0, 0.0, 0.0), (-0.70711, 0.00000, 0.00000, 0.70711)) \\
gimbal_1 \mathbf{T}^{BcsF} &= gimbal_0 \mathbf{T}^{BcsF} gimbal_1 \mathbf{T}^{gimbal_0} \\
&= ((0.00000, 2.00000, -1.00000), (-0.50000, -0.50000, 0.50000, 0.50000))
\end{aligned}$$

### 3.1.4 Derivations

**Left-multiply**

To prove (3.1.1-8), substitute (3.1.1-7) into (3.1.1-6):

$$\begin{aligned}
{}^a \mathbf{T}_c &= {}^b \mathbf{T}_a^{-1} {}^b \mathbf{T}_c \\
&= (-{}^b \mathbf{R}_a^{-1} {}^b \mathbf{r}_a + {}^b \mathbf{R}_a^{-1} {}^b \mathbf{r}_c; {}^b \mathbf{R}_a^{-1} {}^b \mathbf{R}_c) \\
&= ({}^b \mathbf{R}_a^{-1} ({}^b \mathbf{r}_c - {}^b \mathbf{r}_a), {}^b \mathbf{R}_a^{-1} {}^b \mathbf{R}_c) \\
&= ({}^a \mathbf{r}_c; {}^a \mathbf{R}_c)
\end{aligned}$$

**Right-multiply**

Similar to left-multiply.

## 3.2 Rate

### 3.2.1 Reference

The time rate of change of pose of an object is simply called the “rate” of an object. It is defined by the rate of an object frame (rigidly attached to the object), measured relative to a base frame, and expressed in a coordinate frame. To indicate all these frames, we use the notation  ${}_{base}^{coord} \dot{\mathbf{p}}_{obj} = ({}_{base}^{coord} \mathbf{v}_{obj}; {}_{base}^{coord} \boldsymbol{\omega}_{obj})$ . Usually, the base frame is not significant or can be inferred from context, and is not specified. Note that the coordinate frame of the rotation part is not required to be the object frame.

We need to be able to transform rates in one frame to rates in another frame.

**Left-multiply**

To find the rate  ${}^b\dot{\mathbf{p}}_b$  of frame  $b$  attached to rigid body, also expressed in frame  $b$ , given the rate  ${}^a\dot{\mathbf{p}}_a = ({}^a\mathbf{v}_a; {}^a\boldsymbol{\omega}_a)$  of frame  $a$  on the same body, and  ${}^a\mathbf{T}_b = ({}^a\mathbf{r}_b; {}^a\mathbf{R}_b)$ :

$${}^b\dot{\mathbf{p}}_b = {}^a\mathbf{R}_b^{-1}({}^a\mathbf{v}_a + {}^a\boldsymbol{\omega}_a \times {}^a\mathbf{r}_b; {}^a\boldsymbol{\omega}_a) \quad (3.2.1-1)$$

This operation is called a *velocity transform*, and represented by  $\mathbf{T}_{\mathbf{v}:}$ :

$${}^b\dot{\mathbf{p}}_b = {}^b\mathbf{T}_{\mathbf{v}:a} {}^a\dot{\mathbf{p}}_a \quad (3.2.1-2)$$

This can also be expressed as a left-multiply matrix:

$$\begin{aligned} {}^b\mathbf{T}_{\mathbf{v}:a} &= \text{to\_rate\_transform}({}^a\mathbf{T}_b) \\ &= \begin{bmatrix} {}^a\mathbf{R}_b^{-1} & {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_b \times \\ \mathbf{0} & {}^a\mathbf{R}_b^{-1} \end{bmatrix} \end{aligned} \quad (3.2.1-3)$$

Note that the coordinate frame of the rate must be the object frame for the velocity transform operation to be defined; the two vectors involved in the cross product must be expressed in the same coordinate frame. This is why  ${}^b\mathbf{T}_{\mathbf{v}:a}$  is defined in terms of  ${}^a\mathbf{T}_b$  instead of  ${}^b\mathbf{T}_a$ . The notation helps get the frames correct; the post-subscript of the velocity transform must match the pre-superscript of the rate for the equation to make sense.

In matrix form, velocity transforms can be combined, using left matrix multiply:

$$\begin{aligned} {}^a\mathbf{T}_c &= {}^a\mathbf{T}_b {}^b\mathbf{T}_c \\ {}^c\mathbf{T}_{\mathbf{v}:a} &= {}^c\mathbf{T}_{\mathbf{v}:b} {}^b\mathbf{T}_{\mathbf{v}:a} \\ &= \begin{bmatrix} {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_b \times + {}^b\mathbf{R}_c^{-1} {}^b\mathbf{r}_{b-c} \times {}^a\mathbf{R}_b^{-1} \\ \mathbf{0} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} \end{bmatrix} \end{aligned} \quad (3.2.1-4)$$

**Right-multiply**

Since the velocity transform operator is defined in terms of left-multiply matrices, we only define a left-multiply velocity transform. Given a right-multiply pose, we convert it to a left multiply pose, then convert that to a velocity transform.

$$\begin{aligned} {}^a\mathbf{T}_b &= \text{to\_left}({}^b\mathbf{T}^a) \\ ({}^a\mathbf{r}_b; {}^a\mathbf{R}_b) &= ({}^a\mathbf{r}_b; {}^b\mathbf{R}^{a^{-1}}) \end{aligned} \quad (3.2.1-5)$$

### 3.2.2 Coding

The type `SAL.Gen_Math.DOF_6.Rate_Transform_Type` implements rate transforms; it stores  ${}^a\mathbf{R}_b^{-1}$  and  ${}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{b\times}$  as left-multiply matrices.

The child package `SAL.Gen_Math.Gen_DOF_6.Gen_Left` provides rate operations using left-multiply rotations and velocity transforms, the child package `SAL.Gen_Math.Gen_DOF_6.Gen_Wertz` provides rate operations using right-multiply quaternions and left-multiply velocity transforms. Only one of these two child packages should be used in any single program.

A variable representing the translation velocity  ${}_{base}^{coord}\dot{\mathbf{p}}_{obj}$  is written `Coord_Base-Tran_Vel_Obj` in the left-multiply convention, `Obj-Tran_Vel_Base_Coord` in the right-multiply convention.

Similarly, a variable representing the rotation velocity  ${}_{base}^{coord}\dot{\mathbf{w}}_{obj}$  is written `coord_base_Rot_Vel_obj` in the left-multiply convention, `obj_Rot_Vel_base_coord` in the right-multiply convention.

A variable representing  ${}^{coord}\mathbf{T}_{v:obj}$  is written `Coord_Rate_Trnsf_Obj` in the left-multiply convention, `Obj_Rate_Trnsf_Coord` in the right-multiply convention.

### 3.2.3 Examples

### 3.2.4 Derivations

(3.2.1-1) is a combination of (2.6.1-10) and (2.6.1-9).

(3.2.1-3) follows from (2.6.1-9) and (2.6.1-11).

We need the following identity:

$${}^b\mathbf{r}_{1\times} = {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{1\times} {}^a\mathbf{R}_b \quad (3.2.4-1)$$

This follows from the definition of  $\mathbf{r}_{\times}$ :

$$\begin{aligned} {}^b\mathbf{r}_1 &= {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_1 \\ {}^a\mathbf{v}_1 &= {}^a\boldsymbol{\omega}_a \times {}^a\mathbf{r}_1 \\ &= {}^a\mathbf{r}_{1\times} {}^a\boldsymbol{\omega}_a \\ {}^b\mathbf{v}_1 &= {}^b\mathbf{r}_{1\times} {}^b\boldsymbol{\omega}_b \\ &= ({}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{1\times} {}^a\mathbf{R}_b) {}^a\mathbf{R}_b^{-1} {}^a\boldsymbol{\omega}_a \\ &= {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{1\times} {}^a\boldsymbol{\omega}_a \\ &= {}^a\mathbf{R}_b^{-1} {}^a\mathbf{v}_1 \end{aligned}$$

(3.2.1-4) follows from (3.2.1-3) by matrix multiply. We can show it is equivalent to `to_rate_transform` of the pose product by doing the matrix multiply and applying (3.2.4-1):

$$\begin{aligned}
{}^c\mathbf{T}_{v:a} &= {}^c\mathbf{T}_{v:b} {}^b\mathbf{T}_{v:a} \\
&= \begin{bmatrix} {}^b\mathbf{R}_c^{-1} & {}^b\mathbf{R}_c^{-1} {}^b\mathbf{r}_{b-c\times} \\ \mathbf{0} & {}^b\mathbf{R}_c^{-1} \end{bmatrix} \begin{bmatrix} {}^a\mathbf{R}_b^{-1} & {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{a-b\times} \\ \mathbf{0} & {}^a\mathbf{R}_b^{-1} \end{bmatrix} \\
&= \begin{bmatrix} {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{a-b\times} + {}^b\mathbf{R}_c^{-1} {}^b\mathbf{r}_{b-c\times} {}^a\mathbf{R}_b^{-1} \\ \mathbf{0} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} \end{bmatrix} \\
&= \begin{bmatrix} {}^a\mathbf{R}_c^{-1} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{a-b\times} + {}^b\mathbf{R}_c^{-1} ({}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{b-c\times} {}^a\mathbf{R}_b) {}^a\mathbf{R}_b^{-1} \\ \mathbf{0} & {}^a\mathbf{R}_c^{-1} \end{bmatrix} \\
&= \begin{bmatrix} {}^a\mathbf{R}_c^{-1} & {}^b\mathbf{R}_c^{-1} {}^a\mathbf{R}_b^{-1} ({}^a\mathbf{r}_{a-b\times} + {}^a\mathbf{r}_{b-c\times}) \\ \mathbf{0} & {}^a\mathbf{R}_c^{-1} \end{bmatrix} \\
&= \begin{bmatrix} {}^a\mathbf{R}_c^{-1} & {}^a\mathbf{R}_c^{-1} {}^a\mathbf{r}_{a-c\times} \\ \mathbf{0} & {}^a\mathbf{R}_c^{-1} \end{bmatrix} \\
&= \text{to\_rate\_transform}({}^a\mathbf{T}_c)
\end{aligned}$$

### 3.3 Wrench

#### 3.3.1 Reference

##### Left-multiply

The wrench on an object is measured at an object frame, and expressed in a coordinate frame. Wrenches are denoted by  ${}^{coord}\mathbf{w}_{wrench;label}$ .

To find the wrench at frame  $b$ , expressed in frame  $b$  on a rigid body when given the wrench in a frame  $a$  at another point on the same body, and the transform  ${}^a\mathbf{T}_b$  from frame  $a$  to frame  $b$ :

$${}^b\mathbf{w}_b = {}^a\mathbf{R}_b^{-1} ({}^a\mathbf{f}_a; {}^a\boldsymbol{\tau}_a + {}^a\mathbf{f}_a \times {}^a\mathbf{r}_b) \quad (3.3.1-1)$$

This operation is called a *wrench transform*, and represented by  $\mathbf{T}_w$ :

$$\begin{aligned}
{}^b\mathbf{w}_b &= {}^b\mathbf{T}_{w:a} {}^a\mathbf{w}_a \\
&= \begin{bmatrix} {}^a\mathbf{R}_b^{-1} & \mathbf{0} \\ {}^a\mathbf{R}_b^{-1} {}^a\mathbf{r}_{b\times} & {}^a\mathbf{R}_b^{-1} \end{bmatrix} \quad (3.3.1-2)
\end{aligned}$$

Note that the coordinate frame must be the same as the wrench frame for the wrench transform operation to be defined.

Wrench transforms can be combined:

$$\begin{aligned} {}^a\mathbf{T}_c &= {}^a\mathbf{T}_b {}^b\mathbf{T}_c \\ {}^c\mathbf{T}_{\mathbf{w}:a} &= {}^c\mathbf{T}_{\mathbf{w}:b} {}^b\mathbf{T}_{\mathbf{w}:a} \end{aligned} \quad (3.3.1-3)$$

### Right-multiply

As for the velocity transform, since the wrench transform operator is defined in terms of left-multiply matrices, we only define a left-multiply wrench transform. Given a right-multiply pose, we convert it to a left multiply pose, then convert that to a wrench transform.

### 3.3.2 Coding

The type `SAL.Gen_Math.DOF_6.Wrench_Transform_Type` implements wrench transforms.

The child package `SAL.Gen_Math.Gen_DOF_6.Gen_Left` provides wrench operations using left-multiply rotations and wrench transforms, the child package `SAL.Gen_Math.Gen_DOF_6.Gen_Wertz` provides rate operations using right-multiply quaternions and left-multiply wrench transforms. Only one of these two child packages should be used in any single program.

A variable representing  ${}^{coord}\mathbf{w}_{obj}$  is written `Coord_Wrench_Obj` in the left-multiply convention, `Obj_Wrench_Coord` in the right-multiply convention.

A variable representing  ${}^{coord}\mathbf{T}_{\mathbf{w}:obj}$  is written `Coord_Wrench_Trnsf_Obj` in the left-multiply convention, `Obj_Wrench_Trnsf_Coord` in the right-multiply convention.

### 3.3.3 Derivations

Similar to the velocity transform; see 3.2.4.

## 3.4 Mass and Inertia

### 3.4.1 Reference

#### Left-multiply

The zeroth, first and second mass moments of an object about an inertia frame, expressed in a coordinate frame, give the total mass  $m_{obj}$ , the center of mass  ${}^{coord}\mathbf{r}_{inertia-cm}$ , and the inertia  ${}^{coord}_{inertia}\mathbf{I}_{obj}$ .  $\mathbf{I}$  is symmetric, so we define the six elements:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \quad (3.4.1-1)$$

There is no general formula for how the inertia changes when the mass frame is changed by a transform. However, if we are given the inertia  ${}^{cm}\mathbf{I}$  about the center of mass frame  $cm$ , we can find it about another frame  $a$  (not necessarily parallel to  $cm$ ) by first rotating the inertia, and then using the parallel axis theorem:

$${}^a_{cm}\mathbf{I} = {}^a\mathbf{R}_{cm} {}^{cm}\mathbf{I} {}^a\mathbf{R}_{cm}^{-1} \quad (3.4.1-2)$$

$$\begin{aligned} {}^a I_{xx} &= {}^a_{cm} I_{xx} + m({}^a r_{cm,y}^2 + {}^a r_{cm,z}^2) \\ {}^a I_{yy} &= {}^a_{cm} I_{yy} + m({}^a r_{cm,z}^2 + {}^a r_{cm,x}^2) \\ {}^a I_{zz} &= {}^a_{cm} I_{zz} + m({}^a r_{cm,x}^2 + {}^a r_{cm,y}^2) \\ {}^a I_{xy} &= {}^a_{cm} I_{xy} - m {}^a r_{cm,x} {}^a r_{cm,y} \\ {}^a I_{xz} &= {}^a_{cm} I_{xz} - m {}^a r_{cm,x} {}^a r_{cm,z} \\ {}^a I_{yz} &= {}^a_{cm} I_{yz} - m {}^a r_{cm,y} {}^a r_{cm,z} \end{aligned} \quad (3.4.1-3)$$

(3.4.1-3) is implemented by the function  ${}^a\mathbf{I} = \text{par\_axis}(m, {}^a\mathbf{r}_{cm}, {}^a_{cm}\mathbf{I})$ .

We define the mass 3-tuple  ${}^a\mathbf{M} \equiv (m, {}^a\mathbf{r}_{cm}, {}^a_{cm}\mathbf{I})$  to contain the total mass, the center of mass relative to the object frame, and the inertia about a frame located at the center of mass, parallel to the object frame.

Changing the object frame of a mass is given by:

$$\begin{aligned} {}^a\mathbf{M} &= {}^a\mathbf{T}_b {}^b\mathbf{M} \\ &= (m, \\ &\quad {}^a\mathbf{T}_b {}^b\mathbf{r}_{cm}, \\ &\quad {}^a\mathbf{R}_b {}^b_{cm}\mathbf{I} {}^a\mathbf{R}_b^{-1}) \end{aligned} \quad (3.4.1-4)$$

To add two masses  ${}^a\mathbf{M}_1$ ,  ${}^b\mathbf{M}_2$  together, we need the frame  ${}^a\mathbf{T}_b$  giving the location of  $b$  relative to  $a$ . Then the mass of the combined object  ${}^a\mathbf{M}_3$  is given

by the function *add*:

$${}^a\mathbf{M}_3 = \text{add}({}^a\mathbf{M}_1, {}^b\mathbf{M}_2, {}^a\mathbf{T}_b) \quad (3.4.1-5)$$

$$\begin{aligned} &= (m_1 + m_2, \\ &\quad (m_1 {}^a\mathbf{r}_{cm_1} + m_2 {}^a\mathbf{T}_b {}^b\mathbf{r}_{cm_2})/m_3, \\ &\quad \text{par\_axis}(m_1, {}^a\mathbf{r}_{cm_3} - {}^a\mathbf{r}_{cm_1}, {}^{a, cm_1}\mathbf{I}_1) + \\ &\quad \text{par\_axis}(m_2, {}^a\mathbf{r}_{cm_3} - {}^a\mathbf{T}_b {}^b\mathbf{r}_{cm_2}, {}^a\mathbf{R}_b {}^b\mathbf{I}_2 {}^a\mathbf{R}_b^{-1})) \end{aligned} \quad (3.4.1-6)$$

### Right-multiply

$${}_{cm}^a\mathbf{I} = {}_{cm}\mathbf{R}^{a-1} {}_{cm}^a\mathbf{I} {}_{cm}\mathbf{R}^a \quad (3.4.1-7)$$

$$\begin{aligned} {}^a\mathbf{M} &= {}^b\mathbf{M}_b \mathbf{T}^a \\ &= (m, \\ &\quad {}^b\mathbf{r}_{cmb} \mathbf{T}^a, \\ &\quad {}^b\mathbf{R}^{a-1} {}^b\mathbf{I} {}^b\mathbf{R}^a) \end{aligned} \quad (3.4.1-8)$$

## 3.4.2 Coding

The type `SAL.Gen_Math.DOF_6.Mass_Type` implements the mass tuple. The tuple is extended with the inertia about the object frame, since that optimizes some operations.

The type `SAL.Gen_Math.DOF_6.CM_Mass_Type` implements a simplified mass tuple, where the coordinate frame is always the center of mass (so  ${}^a\mathbf{r}_{cm} = 0$ ). Various operations are defined that allow extracting the total mass, center of mass, and inertia.

The child package `SAL.Gen_Math.Gen_DOF_6.Gen_Left` provides mass operations using left-multiply rotations, the child package `SAL.Gen_Math.Gen_DOF_6.Gen_Wertz` provides rate operations using right-multiply quaternions and left-multiply matrices.

A variable representing  ${}^{coord}\mathbf{M}_{label}$  is written `Coord_Mass_Label` in the left-multiply convention, and `Label_Mass_Coord` in the right-multiply convention.

A variable representing the scalar mass  $m_{obj}$  is written `Mass_Obj` in either convention; the meaning will be clear from the type of the variable.

A variable representing  ${}^{coord}_{center}\mathbf{I}_{obj}$  is written `Coord_Center_MOI_Obj` in the left-multiply convention, `Obj_MOI_Center_Coord` in the right-multiply convention. The center frame is left out if it is the center of mass, as for an object of type

`SAL.Gen_Math.DOF_6.CM_Mass_Type`. It is also left out if it is the same as the coordinate frame.

### 3.4.3 Derivations

See any basic physics book for the parallel axis theorem.



# Chapter 4

## Miscellaneous

### 4.1 Velocity aberration

#### 4.1.1 Reference

Star sensors that determine a direction to a star (for example, star trackers, guide telescopes, or sun sensors) depend on the velocity of photons from the stars, which is then subject to the velocity transform (2.6.1-1). When the velocity of the spacecraft is not parallel to the velocity of the photons, the spacecraft perceives the star direction as having shifted relative to the true direction. This effect is called “velocity aberration”.

For star sensors that sense only a single star (guide telescopes and sun sensors), flight software needs to compute the true unit vector to the star, and a simulator needs to compute the aberrated unit vector.

Assume the star is fixed in some inertial frame  $i$ . Define the moving frame  $sci$

to translate with the spacecraft, but stay parallel to the fixed frame. Then:

$$\begin{aligned}
{}^i\hat{\mathbf{n}}_{sc-star} &= \text{To\_Unit\_Vector}({}_{sc}^i\hat{\mathbf{n}}_{sc-star} + \\
&\quad {}_{sc}^i\hat{\mathbf{n}}_{sc-star} \times ({}_{sc}^i\hat{\mathbf{n}}_{sc-star} \times \frac{{}_i\mathbf{v}_{sci}}{c})) \\
&= \text{To\_Unit\_Vector}({}_{sc}^i\hat{\mathbf{n}}_{sc-star} + \\
&\quad {}_{sc}^i\hat{\mathbf{n}}_{sc-star} ({}_{sc}^i\hat{\mathbf{n}}_{sc-star} \cdot \frac{{}_i\mathbf{v}_{sci}}{c}) - \frac{{}_i\mathbf{v}_{sci}}{c}) \quad (4.1.1-1)
\end{aligned}$$

$$\begin{aligned}
{}_{sc}^i\hat{\mathbf{n}}_{sc-star} &= \text{To\_Unit\_Vector}({}_i\hat{\mathbf{n}}_{sc-star} - \\
&\quad {}_i\hat{\mathbf{n}}_{sc-star} \times ({}_i\hat{\mathbf{n}}_{sc-star} \times \frac{{}_i\mathbf{v}_{sci}}{c})) \\
&= \text{To\_Unit\_Vector}({}_i\hat{\mathbf{n}}_{sc-star} - \\
&\quad {}_i\hat{\mathbf{n}}_{sc-star} ({}_i\hat{\mathbf{n}}_{sc-star} \cdot \frac{{}_i\mathbf{v}_{sci}}{c}) + \frac{{}_i\mathbf{v}_{sci}}{c}) \quad (4.1.1-2)
\end{aligned}$$

The notation  ${}_{sc}^i\hat{\mathbf{n}}_{sc-star}$  is a bit confusing; it is a unit vector, expressed in the stationary inertial frame, giving the direction from the spacecraft to the star as measured in the moving inertial frame. The two forms for each equation are equivalent, due to the vector identity  $a \times (b \times c) = b(a \cdot c) - c(a \cdot b)$ .

Star trackers use multiple stars to determine an orientation frame. If the star tracker knows the spacecraft velocity, it will use to deaberrate each star vector before combining them to compute an orientation.

However, if the star tracker does not know the spacecraft velocity, then it assumes  ${}_{sc}^i\hat{\mathbf{n}}_{sc-star} = {}_i\hat{\mathbf{n}}_{sc-star}$ , and computes a frame that is slightly rotated from the true spacecraft orientation. We define the frame  $ab$  to be the frame measured by the star tracker, using the apparent direction to the stars.

If the star tracker field of view is narrow enough, all of the stars are aberrated by approximately the same amount, and we can approximate either  ${}_{sc}^i\hat{\mathbf{n}}_{sc-star}$  or  ${}_i\hat{\mathbf{n}}_{sc-star}$  by  ${}_i\hat{\mathbf{n}}_{bore}$ , the boresight of the star tracker. This allows us to correct quaternions output by star trackers that don't know the spacecraft velocity, without knowing individual star vectors.

In flight software, we have  ${}_i\hat{\mathbf{n}}_{bore}$ , and we need to find the active quaternion  ${}^i\mathbf{Q}_{ab-i}$  that rotates the frame measured by the star tracker to the true inertial frame:

$${}^i\mathbf{Q}_{ab-i} = \text{rotvect\_to\_quat}(-\frac{{}_i\hat{\mathbf{n}}_{bore} \times {}_i\mathbf{v}_{sci}}{c}) \quad (4.1.1-3)$$

In a simulator, we also have  ${}_i\hat{\mathbf{n}}_{bore}$ , and we need to find the opposite quaternion  ${}^i\mathbf{Q}_{i-ab}$ :

$${}^i\mathbf{Q}_{i-ab} = \text{rotvect\_to\_quat}(\frac{{}_i\hat{\mathbf{n}}_{bore} \times {}_i\mathbf{v}_{sci}}{c}) \quad (4.1.1-4)$$

Note that in applying (4.1.1-3) or (4.1.1-4), we must use the active quaternion; this is *not* a simple rotation of frame coordinates. See the example section (4.1.3) for details.

For an example magnitude, the velocity of the Earth in orbit around the Sun is approximately 30\_000.0 meters per second. Then  $\frac{v}{c} = 0.000_1$ , and the maximum aberration is 0.000\_1 radians.

### 4.1.2 Coding

The function `SAL.Gen_Math.Gen_DOF_3.Light_Vector_Transform` implements (4.1.1-2).

The function `SAL.Gen_Math.Gen_DOF_3.Gen_Wertz.Light_Vector_Rotation` implements (4.1.1-4).

### 4.1.3 Example

Consider a spacecraft and star in the geometry shown in figure 4.1.3.1. We assume the boresight of the star tracker is pointed towards the star;  ${}^i\hat{\mathbf{n}}_{bore} = (0.0, 0.0, 1.0)$ .

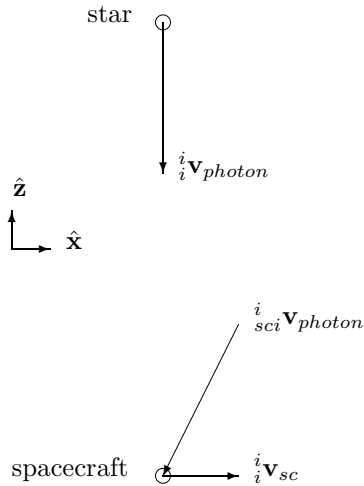


Figure 4.1.3.1: Star tracker aberration velocities

The magnitudes of the velocity vectors are not to scale, and the angle of aberration is grossly exaggerated, for clarity.

${}^i{}_{sc}{}^i\mathbf{v}_{photon}$  is the apparent velocity of the photon in the moving spacecraft frame.

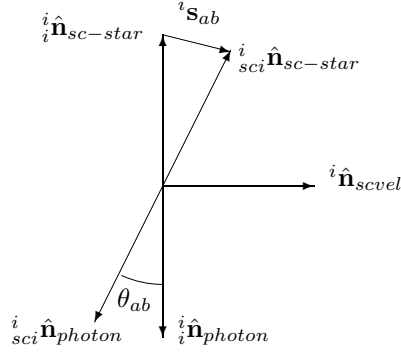


Figure 4.1.3.2: Star tracker aberration unit vectors

$\theta_{ab}$  is the effective rotation angle due to aberration;  ${}^i\mathbf{s}_{ab}$  is the aberration vector.

Now we put in some numbers, using right multiply quaternions, and compute  ${}^i_{sci}\hat{\mathbf{n}}_{sc-star}$  four different ways. See `spacecraft_math_examples.adb`, section Aberration, for the corresponding Ada code.

$$\begin{aligned}
 {}^i\mathbf{v}_{photon} &= (0.0, 0.0, -299792456.199999988) \\
 {}^i\hat{\mathbf{n}}_{sc-star} &= \text{To\_Unit\_Vector}(-{}^i\mathbf{v}_{photon}) \\
 &= (0.0, 0.0, -1.0) \\
 {}^i\mathbf{v}_{sc} &= (30000.000000000, 0.0, 0.0)
 \end{aligned}$$

First we use the classical velocity transform (2.6.1-1):

$$\begin{aligned}
 {}^i_{sci}\mathbf{v}_{photon,1} &= {}^i\mathbf{v}_{photon} - {}^i\mathbf{v}_{sc} \\
 &= (-30000.000000000, 0.0, -299792456.199999988) \\
 {}^i_{sci}\hat{\mathbf{n}}_{sc-star,1} &= \text{To\_Unit\_Vector}(-{}^i_{sci}\mathbf{v}_{photon}) \\
 &= (0.000100069, 0.0, 0.999999995)
 \end{aligned}$$

Now we use the full Lorentz transform (2.6.1-2):

$$\begin{aligned}
 {}^i_{sci}\mathbf{v}_{photon,2} &= \text{Lorentz\_Transform}({}^i\mathbf{v}_{photon}, -{}^i\mathbf{v}_{sc}) \\
 &= (-30000.000000000, 0.0, -299792454.698961556) \\
 {}^i_{sci}\hat{\mathbf{n}}_{sc-star,2} &= \text{To\_Unit\_Vector}(-{}^i_{sci}\mathbf{v}_{photon}) \\
 &= (0.000100069, 0.0, 0.999999995)
 \end{aligned}$$

Now we use (4.1.1-2):

$$\begin{aligned} {}^i_{sci}\hat{\mathbf{n}}_{sc-star,3} &= \text{Light\_Vector\_Transform}({}^i\hat{\mathbf{n}}_{sc-star}, -{}^i_{sci}\mathbf{v}_{photon}) \\ &= (0.000100069, 0.0, 0.999999995) \end{aligned}$$

Now we use (4.1.1-4):

$$\begin{aligned} {}^i\hat{\mathbf{n}}_{bore} &= (0.0, 0.0, 1.0) \\ {}_{i-ab}\mathbf{Q}^i &= \text{Light\_Vector\_Rotation}({}^i\hat{\mathbf{n}}_{bore}, {}^i\mathbf{v}_{sc}); \text{ active} \\ &= (0.0, -0.000050035, 0.0, 0.999999999) \\ {}^i_{sci}\hat{\mathbf{n}}_{sc-star,4} &= {}^i\hat{\mathbf{n}}_{sc-star} {}_{i-ab}\mathbf{Q}^i & (4.1.3-1) \\ &= (0.000100069, 0.0, 0.999999995) \end{aligned}$$

Note the frame labels in  ${}^i_{sci}\hat{\mathbf{n}}_{sc-star}$  is *not*  ${}^{ab}\hat{\mathbf{n}}_{sc-star}$ ; by definition,  ${}^{ab}\hat{\mathbf{n}}_{sc-star} = {}^i\hat{\mathbf{n}}_{sc-star}$ , which is why the *ab* frame is wrong in the first place.

All four computations give the same result for  ${}^i_{sci}\hat{\mathbf{n}}_{sc-star}$ .

Finally, we compute the aberration vector  ${}^i\mathbf{s}_{ab}$ :

$$\begin{aligned} {}^i\mathbf{s}_{ab} &= {}^i_{sci}\hat{\mathbf{n}}_{sc-star} - {}^i\hat{\mathbf{n}}_{sc-star} \\ &= (0.000100069, -0.000000000, -0.000000005) \end{aligned}$$

Now we show that photons at other angles to  ${}^i\mathbf{v}_{sc}$  are also bent towards the velocity direction ( ${}^i\hat{\mathbf{n}}_{bore}$  is adjusted to point towards the star):

$$\begin{aligned} {}^i\hat{\mathbf{n}}_{sc-star} &= (0.707106781, 0.0, 0.707106781) \\ {}^i_{sci}\hat{\mathbf{n}}_{sc-star} &= (0.707156810, 0.0, 0.707056748) \\ {}^i\mathbf{s}_{ab} &= (0.000050031, 0.0, -0.000050035) \end{aligned}$$

$$\begin{aligned} {}^i\hat{\mathbf{n}}_{sc-star} &= (-0.707106781, 0.0, 0.707106781) \\ {}^i_{sci}\hat{\mathbf{n}}_{sc-star} &= (-0.707056741, 0.0, 0.707156818) \\ {}^i\mathbf{s}_{ab} &= (0.000050038, 0.0, 0.000050035) \end{aligned}$$

#### 4.1.4 Derivation

To derive (4.1.1), we express the photon velocity with respect to the two frames  $i, sci$  as  $c$  times a unit vector pointing from the spacecraft to the star. We also represent the spacecraft velocity similarly:

$$\begin{aligned} {}^i\mathbf{v}_{photon} &= -c {}^i\hat{\mathbf{n}}_{sc-star} \\ {}^{sci}\mathbf{v}_{photon} &= -c {}^{sci}\hat{\mathbf{n}}_{sc-star} \\ \beta &= \frac{|{}^i\mathbf{v}_{sci}|}{c} \\ {}^i\mathbf{v}_{sci} &= c\beta {}^i\hat{\mathbf{n}}_{vsc} \end{aligned}$$

These velocities are related by the Lorentz transform (2.6.1-2). Keeping only terms to first order in  $\beta$ , and solving for  ${}^i\hat{\mathbf{n}}_{sc-star}$ , this reduces to (4.1.1). We renormalize the unit vector, because some precision is lost in keeping only terms to first order in *beta*. See `derive_velocity_aberration.maxima` for details.

(4.1.1-2) is found from (4.1.1) by symmetry; the Lorentz transform works in both directions. Swap  $i$  and  $sci$  post and pre subscripts, and negate  ${}^i\mathbf{v}_{sci}$ .

(4.1.1-3) is found from letting  ${}^{sci}\hat{\mathbf{n}}_{sc-star} = {}^i\hat{\mathbf{n}}_{bore}$  in (4.1.1), applying the vector identity  $a \times (b \times c) = b(a \cdot c) - c(a \cdot b)$  and finally applying `units_to_quat`:

$$\begin{aligned} {}^i\hat{\mathbf{n}}_{sc-star} &= {}^i\hat{\mathbf{n}}_{bore} + {}^i\hat{\mathbf{n}}_{bore} \times \left( {}^i\hat{\mathbf{n}}_{bore} \times \frac{{}^i\mathbf{v}_{sci}}{c} \right) \\ {}^i\hat{\mathbf{n}}_{sc-star} &= {}^i\hat{\mathbf{n}}_{bore} + {}^i\hat{\mathbf{n}}_{bore} \left( {}^i\hat{\mathbf{n}}_{bore} \cdot \frac{{}^i\mathbf{v}_{sci}}{c} \right) - \frac{{}^i\mathbf{v}_{sci}}{c} \\ {}^i\mathbf{Q}_{ab-i} &= \text{units\_to\_quat}({}^i\hat{\mathbf{n}}_{bore}, {}^i\hat{\mathbf{n}}_{sc-star}) \end{aligned} \quad (4.1.4-1)$$

where `units_to_quat` is given by (2.3.1-18) (for left-multiply) or 2.3.1-30 (for right-multiply). Expanding and once again keeping only first order terms in  $\frac{v}{c}$ , this reduces to (4.1.1-3). See `derive_velocity_aberration.maxima` for details.

(4.1.1-4) is again found by symmetry.

(4.1.1-3) agrees with Wertz eqn 5-52, which is:

$$\Delta\theta = \frac{v}{c} \sin(\theta)$$

where  $\Delta\theta$  is the aberration magnitude, and  $\theta$  is the angle between  ${}^i\mathbf{v}_{sc}$  and  ${}^{sci}\hat{\mathbf{n}}_{sc-star,ab}$ . Note that Wertz does not carefully define the sign of the aberration angle.

More importantly, (4.1.1) and (4.1.1-3) agree with the Galileo star tracker, which performs velocity deabberation in the star tracker, and has been tested in space.

## 4.2 Propellant tank inertia

The fuel or oxidizer (collectively called “propellant”) in a tank contribute significantly to the mass moments of a spacecraft. In addition, since the propellant is expelled as thrusters are fired, the mass moments change over time.

Modeling the exact mass distribution of propellants in free-fall is very difficult; they slosh around in the tank as different thrusters are fired, and are influenced strongly by surface tension when thrusters are not firing. Generally there are anti-slosh structures in the tank that also influence the shape.

The important aspects for real-time simulation are that the total mass be correct (since that affects the acceleration due to thrusters), the center of mass be nearly correct (since that affects the moment arm for torque from thrusters), and the moment of inertia be reasonable (it is usually small compared to the total spacecraft moment of inertia). Thus it is usually sufficient to approximate the propellant shape in the tank by some simple shape.

Typically we know the total mass of the propellant by integrating mass flow thru the thrusters, and need to compute the shape from that and the tank geometry. We also need to compute the center of mass and moments of inertia, about the center of mass of the propellant. See 3.4.1 for more on the math of masses.

A simple shape to compute is a sphere. One advantage of this approach is it requires no knowledge of the actual shape of the tank.

A more accurate shape that can be computed is a spherical section. When a single thruster is fired for a long time, with sufficient force to overcome surface tension, the propellant in a spherical tank will settle into a spherical section (if the anti-slosh device allows this). Computing the height of the spherical section requires complex arithmetic; the center of mass and moment of inertia are more straight-forward.

### 4.2.1 Reference

For a sphere, given a total mass  $m$  and propellant density  $\rho$ , the radius  $r$  and moment of inertia  $I_{cm}$  about the center of mass of the sphere containing  $m$  are:

$$r = \left(\frac{3m}{4\pi\rho}\right)^{\frac{1}{3}} \quad (4.2.1-1)$$

$$I_{cm} = \frac{2}{5}mr^2 \quad (4.2.1-2)$$

The center of mass is at the center of the sphere. Note that this makes no assumption about the shape of the tank. The center of mass and moment of inertia about some point on the tank can be found via the parallel axis theorem (3.4.1-3).

For a spherical section, we assume the tank is oriented with  $\hat{z}$  the axis of symmetry (4.2.1.1).

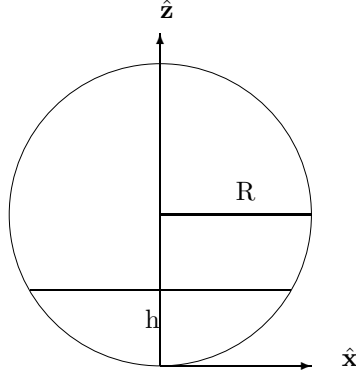


Figure 4.2.1.1: propellant tank spherical section geometry

Given tank radius  $R$ , the height of the propellant  $h$  above the bottom of the tank is given by:

$$\begin{aligned}
 \zeta &= \frac{\sqrt{3}i}{2} - \frac{1}{2} \\
 A &= \frac{\sqrt{3}i\sqrt{m}\sqrt{|4\pi R^3\rho - 3m|}}{2\pi\rho} \\
 B &= \frac{2\pi R^3\rho - 3m}{2\pi\rho} \\
 U &= (B + A)^{\frac{1}{3}} \\
 h &= \frac{R^2\zeta}{U} + \frac{U}{\zeta} + R
 \end{aligned} \tag{4.2.1-3}$$

where  $i$  is  $\sqrt{-1}$ , and  $\zeta$  is a complex cube root of  $-1$ ; evaluating this equation requires complex arithmetic.  $h$  will always be real, of course.

$U$  will never be zero for a real mass. However, the argument of the square root in  $A$  may be slightly negative due to round off error when the tank is full ( $h = 2R$ ); thus we take the absolute value to ensure it is positive.

Given  $h$ , the height of the center of mass above the bottom of the tank is:

$$z_{cm} = \frac{h(8R - 3h)}{4(3R - h)} \tag{4.2.1-4}$$

The moments of inertia about the center of mass are:

$${}^{cm}I_{zz} = \frac{h^3 \pi (20 R^2 - 15 h R + 3 h^2) \rho}{30} \quad (4.2.1-5)$$

$${}^{bot}I_{xx} = \frac{\pi h^3 (20 R^2 + 15 h R - 9 h^2) \rho}{60} \quad (4.2.1-6)$$

$${}^{cm}I_{xx} = {}^{cm}I_{yy} \quad (4.2.1-7)$$

$$= {}^{bot}I_{xx} - m z_{cm}^2 \quad (4.2.1-8)$$

### 4.2.2 Coding

The package `SAL.Gen_Math.Gen_Tank` implements these equations. For the spherical section complex arithmetic, `Ada.Numerics.Generic_Complex_Types` and `Ada.Numerics.Generic_Complex_Elementary_Functions` are used.

### 4.2.3 Derivation

The mass of a sphere of radius  $r$  is:

$$m = \frac{4}{3} \pi \rho r^3$$

(4.2.1-2) is obtained by solving for  $r$ . The moment of inertia of a sphere can be found in any basic physics textbook.

We present an outline of the derivation for the spherical section here; for details, see `derive_spherical_section_moi.maxima`.

The mass  $m(h)$  of a spherical section of height  $h$  is found by integrating thin disks with mass  $m_d(z)$  along the  $z$  axis. First we find the radius  $r(z)$  of a disk at height  $z$  above the bottom of the sphere:

$$\begin{aligned} m(h) &= \int_0^h m_d(z) dz \\ m_d(z) &= \pi r(z)^2 \rho \\ r(z) &= \sqrt{2 R z - z^2} \\ m(h) &= \frac{\pi (3 h^2 R - h^3) \rho}{3} \end{aligned}$$

$h$  is found by solving this cubic equation using Cardano's method (wikipedia). See `derive_spherical_section_moi.maxima` for details; Maxima's `solve` implements Cardano's method. Simple experimentation shows that Maxima's first root gives  $h$  in the range 0 to  $2R$ .

To show that  $U$  is never zero, consider  $B + A$ :

$$t = \frac{\sqrt{3}i\sqrt{m}\sqrt{4\pi R^3\rho - 3m}}{2\pi\rho} + \frac{2\pi R^3\rho - 3m}{2\pi\rho}$$

Since the first term is imaginary, and the second real, this can only be zero if both are zero simultaneously. That would require:

$$\begin{aligned} m &= \frac{4\pi R^3\rho}{3} \\ m &= \frac{2\pi R^3\rho}{3} \end{aligned}$$

which is clearly impossible.

The center of mass  $z_{cm}(h)$  is the integral of  $z m_d(z)$ , divided by the total mass:

$$z_{cm}(h) = \frac{\int_0^h z m_d(z) dz}{m(h)}$$

The moment of inertia about the  $z$  axis is the integral of the moment of inertia of disks about the  $z$  axis:

$$\begin{aligned} {}^{cm}I_{zz-disk}(z) &= \frac{m_d(z) * r(z)^2}{2} \\ {}^{cm}I_{zz} &= \int_0^h ({}^{cm}I_{zz-disk}(z)) dz \end{aligned}$$

The moment of inertia about the  $x$  axis at the bottom of the tank  ${}^{bot}I_{xx}$  (not the center of mass) is the integral of the moment of inertia of disks about the  $x$  axis  ${}^{bot}I_{xx-disk}(z)$ , using the parallel axis theorem (3.4.1-3). Then one more application of (3.4.1-3) gives the moment of inertia about the  $x$  axis at the center of mass  ${}^{cm}I_{xx}$ :

$$\begin{aligned} {}^{bot}I_{xx-disk}(z) &= \frac{m_d(z)r(z)^2}{4} + m_d(z)z^2 \\ {}^{bot}I_{xx} &= \int_0^h ({}^{bot}I_{xx-disk}(z)) dz \\ {}^{cm}I_{xx} &= {}^{bot}I_{xx} - mz_{cm}^2 \end{aligned}$$

The last computation is not worth expanding.

### 4.3 Firing thrusters

When integrating equations of motion for the spacecraft, the spacecraft momentum is one of the state variables being integrated. Therefore we need to

compute the change in spacecraft momentum due to firing thrusters. In this case “spacecraft” includes all the propellant still on board the spacecraft, but not the propellant that has been expelled; the mass of the spacecraft changes.

When thrusters fire, there are two effects on the momentum of the spacecraft. One is due to the reaction force of the escaping gases (the nominal thruster force), the other is due to the portion of total momentum carried by those gases.

### 4.3.1 Reference

The rate of change of spacecraft linear momentum and angular momentum due to firing a thruster is given by:

$${}^{sys}\dot{\mathbf{P}}_{sccm} = -\dot{m} {}^{sys}\mathbf{v}_{sccm} + {}^{sys}\mathbf{f}_{thr} \quad (4.3.1-1)$$

$${}^{sccm}\dot{\mathbf{L}}_{sc} = {}^{sys}\mathbf{r}_{sccm-thr} \times {}^{sys}\mathbf{f}_{thr} \quad (4.3.1-2)$$

where  $\mathbf{f}_{thr}$  is the force exerted by the thruster on a test stand,  $\dot{m}$  is the rate at which propellant mass is expelled, and  $\mathbf{v}_{sccm}$  is the velocity of the spacecraft center of mass in an inertial frame.

Note that we define  $\dot{m}$  to be positive for a thruster firing.

### 4.3.2 Example

To see the two effects, consider a simplified model consisting of two solid masses;  $m_p$  representing the propellant that will be expelled,  $m_s$  the remaining spacecraft mass. Initially, both are moving with some velocity  $\mathbf{v}_{sccm0}$  (figure 4.3.2.1); this will typically be the orbital velocity, which is large compared to the change in velocity due to thruster firings. Then we fire an imaginary thruster for a short time, which separates the two masses; they are now moving with velocities  $\mathbf{v}_{p1}$ ,  $\mathbf{v}_{sccm1}$  (figure 4.3.2.2).

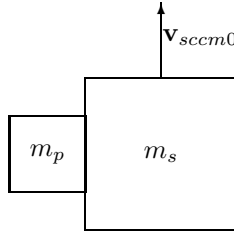


Figure 4.3.2.1: simplified spacecraft and propellant

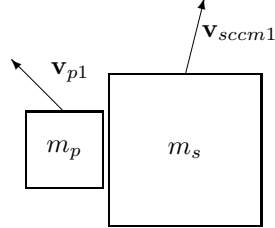


Figure 4.3.2.2: simplified spacecraft and propellant after firing

Now put in some numbers. Assume  $\mathbf{v}_{sccm}$  is initially in the x direction, and the thruster is fired in the y direction.

$$\begin{aligned}
 m_{sccm0} &= 500 \text{ kg} \\
 \mathbf{v}_{sccm0} &= (1000.0, 0.0, 0.0) \text{ m/s} \\
 \mathbf{f}_{thr} &= (0.0, 500.0, 0.0) \text{ N} \\
 \dot{m} &= 0.5 \text{ kg/s} \\
 \Delta t &= 10 \text{ seconds}
 \end{aligned}$$

We integrate (4.3.1-1) over the thruster firing, assuming the thruster force is constant over that time, and assuming the change in spacecraft velocity is small compared to the initial velocity. Doing the integration numerically without these assumptions gives very similar results.

$$\Delta \mathbf{P}_{sccm} = (-\dot{m} \mathbf{v}_{sccm} + \mathbf{f}_{thr}) \Delta t$$

Computing  $\Delta P$  and  $\mathbf{v}_{sccm1}$ :

$$\begin{aligned}
 \Delta \mathbf{P}_{sc} &= (-0.5 * (1000.0, 0.0, 0.0) + (0.0, 500.0, 0.0)) * 10.0 \\
 &= (-5000.0, 5000.0, 0.0) \\
 \mathbf{P}_{sc0} &= (500000.0, 0.0, 0.0) \\
 \mathbf{P}_{sc1} &= (495000.0, 5000.0, 0.0) \\
 m_{sc1} &= 495 \\
 \mathbf{v}_{sccm1} &= (1000.0, 10.10, 0.0)
 \end{aligned}$$

Note that the spacecraft velocity in the x direction does not change, as expected. The momentum in the x direction does change, because the mass of the spacecraft changed; the “missing” momentum is carried away by the expelled propellant.

If we had left out the  $\dot{m}\mathbf{v}_{scm}$  term, we would have had:

$$\begin{aligned}\Delta\mathbf{P}_{sc} &= (0.0, 500.0, 0.0) * 10.0 \\ &= (0.0, 5000.0, 0.0) \\ \mathbf{P}_{sc0} &= (500000.0, 0.0, 0.0) \\ \mathbf{P}_{sc1} &= (500000.0, 5000.0, 0.0) \\ m_{sc1} &= 495 \\ \mathbf{v}_{scm1} &= (1010.10, 10.10, 0.0)\end{aligned}$$

which is clearly wrong; the velocity in the x direction has changed with no force applied in the x direction.

### 4.3.3 Derivation

The integration of linear momentum is performed in an inertial frame not located at the system center of mass; the initial velocity of the spacecraft cannot be taken to be zero.

If we take the system to include both masses, there are no external forces acting on the system, so momentum is conserved, and we can solve for  $\mathbf{v}_{scm1}$ :

$$\begin{aligned}\mathbf{P}_{total-0} &= (m_p + m_s)\mathbf{v}_{scm0} \\ \mathbf{P}_{total-1} &= \mathbf{P}_{total-0} \\ &= m_p\mathbf{v}_{p1} + m_s\mathbf{v}_{scm1} \\ \mathbf{v}_{scm1} &= \frac{m_s\mathbf{v}_{scm0} + m_p(\mathbf{v}_{scm0} - \mathbf{v}_{p1})}{m_s}\end{aligned}$$

Now compute the change of momentum of the spacecraft:

$$\begin{aligned}\mathbf{P}_{sc0} &= (m_p + m_s)\mathbf{v}_{scm0} \\ \mathbf{P}_{sc1} &= m_s\mathbf{v}_{scm1} \\ \Delta\mathbf{P}_{sc} &= \mathbf{P}_{sc1} - \mathbf{P}_{sc0} \\ &= m_s\mathbf{v}_{scm1} - (m_p + m_s)\mathbf{v}_{scm0} \\ &= m_s\mathbf{v}_{scm0} + m_p(\mathbf{v}_{scm0} - \mathbf{v}_{p1}) - (m_p + m_s)\mathbf{v}_{scm0} \\ &= -m_p\mathbf{v}_{p1}\end{aligned}$$

We can write  $\mathbf{v}_{p1}$  as the sum of two components; the initial velocity, and the

change due to firing the thruster:

$$\Delta \mathbf{P}_{sc} = -m_p(\mathbf{v}_{sccm0} + \mathbf{v}_{thr})$$

The first term is the portion of initial spacecraft momentum carried away by the expelled propellant; the second term is the nominal force due to firing the thruster.

To convert this to a differential equation, divide by  $\Delta t$ , and let  $\Delta t \Rightarrow 0$ :

$$\begin{aligned} \dot{\mathbf{P}}_{sc} &= -\dot{m}(\mathbf{v}_{sccm0} + \mathbf{v}_{thr}) \\ &= -\dot{m}\mathbf{v}_{sccm0} + {}^{sys}\mathbf{f}_{thr} \end{aligned}$$

A note on terminology. Since  $\dot{m}\mathbf{v}_{sccm0}$  has the dimensions of a force, we could call it a “force”. However, that goes against the common notion of forces. For one thing, the magnitude of  $\dot{m}\mathbf{v}_{sccm0}$  depends on the choice of inertial reference frame; if we switch to a second frame that is moving with respect to the first,  $\mathbf{v}_{sccm0}$  changes.

So we avoid calling this a force, and simply make sure it is included in the integration code.

Since we cannot find a reference that describes the  $\dot{m}\mathbf{v}_{sccm0}$ , we need to do a similar analysis for angular momentum, to see if there are any similar terms there.

The integrator state variable is the angular momentum of the spacecraft about its center of mass. The integration is performed in the spacecraft center of mass frame. Figures 4.3.3.1 and 4.3.3.2 show the relevant variables.

Because the integration is performed in the spacecraft center of mass frame, we can use an inertial reference frame located at the system center of mass for this analysis. The spacecraft is initially at rest in this frame, since the spacecraft is the only mass present.

There are three frames of interest:

- the system frame *sys*, which is inertial, located at the system center of mass, and moving with respect to *sys*
- the spacecraft frame *sc*, which is fixed in the body of the spacecraft, and moving with velocity  ${}^{sys}\mathbf{v}_{sc}$
- the spacecraft center of mass frame *sccm*, which is moving with velocity  ${}^{sys}\mathbf{v}_{sccm}$ , rotates with *sc*, but translates relative to *sc* as the thruster is fired

Note that  ${}^{sys}\mathbf{v}_{sccm}$  is not equal to  ${}^{sys}\mathbf{v}_{sc}$ , since the spacecraft center of mass is moving in the *sc* frame as the propellant is expelled.

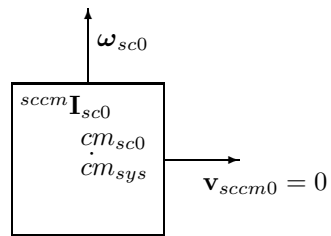


Figure 4.3.3.1: spacecraft angular momentum before firing

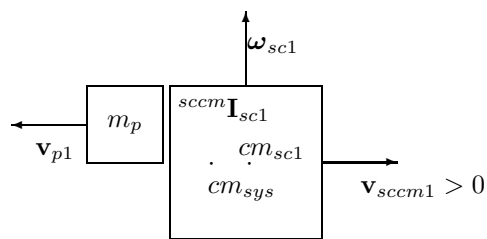


Figure 4.3.3.2: spacecraft angular momentum after firing

The total system angular momentum, taken about the system center of mass, before and after firing is:

$$\begin{aligned} {}^{sys}\mathbf{L}_{total-0} &= {}^{sccm}\mathbf{I}_{sc0} {}^{sccm}\boldsymbol{\omega}_{sc0} \\ {}^{sys}\mathbf{L}_{total-1} &= {}^{sccm}\mathbf{I}_{sc1} {}^{sccm}\boldsymbol{\omega}_{sc1} + \\ &\quad m_p {}^{sys}\mathbf{v}_{p1} \times {}^{sys}\mathbf{r}_{p1} + \\ &\quad m_{sc1} {}^{sys}\mathbf{v}_{sccm1} \times {}^{sys}\mathbf{r}_{sccm1} \end{aligned}$$

The total system linear momentum is:

$$\begin{aligned} {}^{sys}\mathbf{P}_{total-0} &= m_{sc0} {}^{sys}\mathbf{v}_{sccm0} \\ {}^{sys}\mathbf{P}_{total-1} &= m_p {}^{sys}\mathbf{v}_{p1} + m_s {}^{sys}\mathbf{v}_{sccm1} \end{aligned}$$

Since  ${}^{sys}\mathbf{v}_{sccm0} = 0$ , setting these equal gives:

$$\begin{aligned} m_p {}^{sys}\mathbf{v}_{p1} &= -m_s {}^{sys}\mathbf{v}_{sccm1} \\ &= {}^{sys}\mathbf{f}_{thr}\Delta t \end{aligned}$$

The location of the thruster on the spacecraft  ${}^{sys}\mathbf{r}_{sccm-thr}$  is:

$${}^{sys}\mathbf{r}_{sccm1-thr} = {}^{sys}\mathbf{r}_{p1} - {}^{sys}\mathbf{r}_{sccm1}$$

Then we can write  ${}^{sys}\mathbf{L}_{total-1}$  as:

$$\begin{aligned} {}^{sys}\mathbf{L}_{total-1} &= {}^{sccm}\mathbf{I}_{sc1} {}^{sccm}\boldsymbol{\omega}_{sc1} + \\ &\quad \Delta t {}^{sys}\mathbf{f}_{thr} \times {}^{sys}\mathbf{r}_{sccm1-thr} \end{aligned}$$

Setting  ${}^{sys}\mathbf{L}_{total-1} = {}^{sys}\mathbf{L}_{total-0}$ , we have:

$$\begin{aligned} {}^{sccm}\mathbf{I}_{sc0} {}^{sccm}\boldsymbol{\omega}_{sc0} &= {}^{sccm}\mathbf{I}_{sc1} {}^{sccm}\boldsymbol{\omega}_{sc1} + \\ &\quad \Delta t {}^{sys}\mathbf{f}_{thr} \times {}^{sys}\mathbf{r}_{sccm1-thr} \end{aligned}$$

${}^{sccm}\mathbf{I}_{sc} {}^{sccm}\boldsymbol{\omega}_{sc}$  is the angular momentum of the spacecraft about the spacecraft center of mass  ${}^{sccm}\mathbf{L}_{sc}$ . The change in this momentum is:

$$\begin{aligned} \Delta {}^{sccm}\mathbf{L}_{sc} &= {}^{sccm}\mathbf{L}_{sc1} - {}^{sccm}\mathbf{L}_{sc0} \\ &= \Delta t {}^{sys}\mathbf{r}_{sccm-thr} \times {}^{sys}\mathbf{f}_{thr} \\ {}^{sccm}\dot{\mathbf{L}}_{sc} &= {}^{sys}\mathbf{r}_{sccm-thr} \times {}^{sys}\mathbf{f}_{thr} \end{aligned}$$

Which agrees with standard physics texts; no surprise terms.

## Chapter 5

# Articulated Mechanisms

An articulated mechanism is a single open chain of joints connected by rigid links. The joints can be rotational or translational.

Given the joint positions, we are interested in computing the pose of the last link and the mass moments of the mechanism; this is the forward kinematics problem.

### 5.1 Specifying Geometry

We need to be able to compute the transform from each joint to the next, and ultimately from the base of the mechanism to the tip. This requires specifying the joint geometry (how it moves), and the link geometry (the rigid connection between joints).

Joints may be revolute (rotating) or prismatic (sliding linearly).

In the robotics literature, joint and link geometry is usually specified by Denavit-Hartenberg (D-H) parameters ([2], [1, section 3.3]). They allow a single parameterization format to specify the geometry of arbitrary joints. The SAL software uses D-H parameters for articulated mechanisms in general.

However, the spacecraft community does not typically use D-H parameters. Therefore we define a different parameterization, corresponding to common usage at NASA Goddard Spaceflight Center.

### 5.1.1 Reference

#### Denavit-Hartenberg Parameters

There is a coordinate frame  $i$  attached to each link. Unfortunately, there is disagreement about where to locate this frame; [2] places it at the end of the link farthest from the base, [1] places it at the end closest to the base. We follow [1]. The link frame origin is on the joint axis, at the point where that axis is closest to the next joint axis. Note that this may be outside the physical link. The joint motion is along or about the Z axis of the link frame. The X axis points to the next joint. The transform from one link frame to the next is given by 5 parameters for joint  $i$ :

Name	Description
$a_{i-1}$	translation along $\hat{\mathbf{x}}_{i-1}$ from $\hat{\mathbf{z}}_{i-1}$ to $\hat{\mathbf{z}}_i$
$\alpha_{i-1}$	rotation about $\hat{\mathbf{x}}_{i-1}$ from $\hat{\mathbf{z}}_{i-1}$ to $\hat{\mathbf{z}}_i$
$d_i$	translation along $\hat{\mathbf{z}}_i$ from $\hat{\mathbf{x}}_{i-1}$ to $\hat{\mathbf{x}}_i$
$\theta_i$	rotation about $\hat{\mathbf{z}}_i$ from $\hat{\mathbf{x}}_{i-1}$ to $\hat{\mathbf{x}}_i$

For revolute joints,  $\theta_i$  is the joint position. For prismatic joints,  $d_i$  is the joint position.

Joints are typically numbered starting at 1, but some situations may require other numberings.

Denavit-Hartenberg places restrictions on the orientation of the base frame relative to the first link frame; it can only be a rotation about  $\hat{\mathbf{x}}$ . So we define an additional parameter  $^{base}\mathbf{T}_{dh}$  where *base* is the desired arbitrary mechanism base frame, and *dh* is the base frame required by Denavit-Hartenberg.

We also define a final arbitrary frame *tool* specified by the parameter  $^{last}\mathbf{T}_{tool}$ . In spacecraft, the “tool” will typically be an antenna or solar panel.

The first and last links in the chain need special attention.

The first link (closest to the base) has no previous link, so we define the previous link frame to be the *dh* frame. Typically, the *dh* frame is located at the first joint, and parallel to the first link frame when the joint is at 0.0, so  $a_{dh}$  and  $\alpha_{dh}$  are 0.0.

The last link (farthest from the base) has no second joint, so  $d_{last}$  and  $\theta_{last}$  are set to 0.0.

The mass of each link is specified in the link frame;  ${}^i\mathbf{M}_i$ . We allow for a separate tool mass; in robotics applications that can change often.

**Left multiply** To compute  ${}^{i-1}\mathbf{T}_i$ :

$$\begin{aligned}
 r_x &= A \\
 r_y &= -\sin(\alpha)D \\
 r_z &= \cos(\alpha)D \\
 Q_x &= \sin(\alpha/2)\cos(\theta/2) \\
 Q_y &= -\sin(\alpha/2)\sin(\theta/2) \\
 Q_z &= \cos(\alpha/2)\sin(\theta/2) \\
 Q_s &= \cos(\alpha/2)\cos(\theta/2)
 \end{aligned} \tag{5.1.1-1}$$

**Right multiply** To compute  ${}_i\mathbf{T}^{i-1}$ :

$$\begin{aligned}
 r_x &= A \\
 r_y &= -\sin(\alpha)D \\
 r_z &= \cos(\alpha)D \\
 Q_x &= -\sin(\alpha/2)\cos(\theta/2) \\
 Q_y &= \sin(\alpha/2)\sin(\theta/2) \\
 Q_z &= -\cos(\alpha/2)\sin(\theta/2) \\
 Q_s &= \cos(\alpha/2)\cos(\theta/2)
 \end{aligned} \tag{5.1.1-2}$$

### Base at Nominal Parameters

In spacecraft usage, joint and link geometry is typically specified by giving the axis of motion and the offset to the axis of motion of each joint relative to the base frame of the mechanism, when the mechanism joints are at some nominal position (typically, but not always, 0.0). Only revolute joints are currently supported. In that nominal position, the link frames are all parallel to the base frame.

We call this the “base-at-nominal” parameterization. The parameters are:

Name	Description
${}^{base}\mathbf{r}_i$	translation from the mechanism base to joint i, with all joints at $\theta_{nom}$
<i>Axis</i>	axis of rotation for joint i; +-X, +-Y, or +-Z
$\theta_{nom}$	nominal position of joint i

In practice, because the axes are all at right angles to each other, and typically some intersect, it is easy to convert this parameterization to Denavit-Hartenberg manually. So we do not present an algorithm for that process.

The mass moments of each link are specified in the base frame. They must be transformed to the Denavit-Hartenberg link frames. This is a different transform for each link, but it is usually simple to derive.

### 5.1.2 Coding

The package `SAL.Gen_Math.Gen_Den_Hart` implements the Denavit-Hartenberg parameters. The child package `SAL.Gen_Math.Gen_Den_Hart.Gen_Wertz` provides `To_Pose` for right-multiply quaternions; `SAL.Gen_Math.Gen_Den_Hart.Gen_Left` for Wertz.

## 5.2 Mechanism pose, mass

### 5.2.1 Reference

Given the Denavit-Hartenberg parameters, the pose of the tool frame relative to the base frame  ${}^{base}\mathbf{T}_{tool}$  is given by multiplying all the link transforms.

Similarly, the total mass moments of the mechanism  ${}^{base}\mathbf{M}_{total}$  are found by adding the mass moments of each link. Since this requires  ${}^{base}\mathbf{T}_i$ , we can compute the pose and the mass in the same loop.

**Left multiply**

$$\begin{aligned} {}^{base}\mathbf{T}_1 &= {}^{base}\mathbf{T}_{dh} {}^{dh}\mathbf{T}_1 \\ {}^{base}\mathbf{M}_{total} &= {}^{base}\mathbf{T}_1^{-1} \mathbf{M}_1 \end{aligned}$$

for i in 2 .. last loop

$$\begin{aligned} {}^{base}\mathbf{T}_i &= {}^{base}\mathbf{T}_{i-1} {}^{i-1}\mathbf{T}_i \\ {}^{base}\mathbf{M}_{total} &= add({}^i\mathbf{M}_i, {}^{base}\mathbf{M}_{total}, {}^{base}\mathbf{T}_i) \end{aligned}$$

end loop

$$\begin{aligned} {}^{base}\mathbf{T}_{tool} &= {}^{base}\mathbf{T}_i {}^{last}\mathbf{T}_{tool} \\ {}^{base}\mathbf{M}_{total} &= add({}^{tool}\mathbf{M}_{tool}, {}^{base}\mathbf{M}_{total}, {}^{base}\mathbf{T}_{tool}) \end{aligned} \quad (5.2.1-1)$$

**Right multiply**

$$\begin{aligned} {}_1\mathbf{T}^{base} &= {}_1\mathbf{T}^{dh} {}_{dh}\mathbf{T}^{base} \\ {}^{base}\mathbf{M}_{total} &= {}^1\mathbf{M}_1 {}_1\mathbf{T}^{base} \end{aligned}$$

for i in 2 .. last loop

$$\begin{aligned} {}_i\mathbf{T}^{base} &= {}_i\mathbf{T}^{i-1} {}_{i-1}\mathbf{T}^{base} \\ {}^{base}\mathbf{M}_{total} &= add({}_i\mathbf{M}_i, {}^{base}\mathbf{M}_{total}, {}_i\mathbf{T}^{base}) \end{aligned}$$

end loop

$$\begin{aligned} {}^{tool}\mathbf{T}^{base} &= {}^{tool}\mathbf{T}^{last} {}_i\mathbf{T}^{base} \\ {}^{base}\mathbf{M}_{total} &= add({}^{tool}\mathbf{M}_{tool}, {}^{base}\mathbf{M}_{total}, {}^{tool}\mathbf{T}^{base}) \quad (5.2.1-2) \end{aligned}$$

### 5.2.2 Coding

The packages `SAL.Gen_Math.Gen_Manipulator.Gen_Left`, `SAL.Gen_Math.Gen_Manipulator.Gen_Wertz` implement these equations.



# Bibliography

- [1] "Introduction to Robotics Mechanics & Control" John J. Craig, Addison-Wesley, 1986
- [2] Denavit-Hartenberg parameters [http://uwf.edu/ria/robotics/robotdraw/DH\\_parm.htm](http://uwf.edu/ria/robotics/robotdraw/DH_parm.htm)
- [3] Solar Dynamics Observatory (SDO) ACS Flight Software Variable Naming Standard 464-ACS-HDBK-0013
- [4] Thomas R. Kane, Peter W. Likins, David A. Levinson, Spacecraft Dynamics, 1983
- [5] Ken Shoemake, "Animating Rotation with Quaternion Curves", Computer Graphics, volume 19, number 3, 1985. pp 245-254
- [6] James R. Wertz, ed., Spacecraft Attitude Determination and Control, 1978.