AAS 99-139

MATLAB TOOLBOX FOR RIGID BODY KINEMATICS

Hanspeter Schaub* **and John L. Junkins†**

A MATLAB toolbox is presented which provides many commonly used functions in rigid body kinematics. Transformations between the direction cosine matrix, Euler parameters, Gibbs vector, modified Rodrigues parameters, principal rotation vector and the 12 sets of Euler angles are provided. Whenever possible, direct analytical transformations between these sets are used to provide computationally efficient code. Furthermore, the ability to perform two successive rotations and compute the resulting overall attitude vector is provided for each attitude coordinate choice. Similarly, the toolbox allows for the relative attitude vector between two given attitude vectors to be computed. Finally, subroutines are provided that compute the matrix relating the attitude parameter rates to the body angular velocity vector, along with the analytic inverse of this matrix. With these routines the attitude vector time derivative is computed for a given body angular velocity vector.

INTRODUCTION

MATLAB is a very popular tool among engineers to quickly develop numerical simulations of dynamical systems and analyze the resulting motion. The capabilities of MATLAB are expanded by installing toolboxes which provide additional functionality. In attitude control problems, commonly used toolboxes include the *Control Toolbox* and the *Robust Control Toolbox*.

When developing numerical simulations involving rigid body rotations, numerous possibilities exist to describe the orientation or attitude of this body. While each attitude description has its strengths and weaknesses, some are clearly superior to others. Often the programmer may only be familiar with a few attitude coordinate choices and/or may not be aware of several direct transformations of these coordinates. For example, using any one of the 12 Euler angle sets it is difficult to describe arbitrary rotations, since Euler angles are never more than a 90 degree rotation away from a singularity. Despite this short-coming, Euler angles are a very popular set of attitude coordinates since they are easy to visualize for small rotations. Therefore, many times a job description asks for the attitude time history to be shown in terms of a particular set of Euler angles. However, if the final output is required to be in these Euler angles, it may be more convenient to use other, less singular attitude coordinates within the simulation and translate the output to the desired Euler angles. The presented rigid body kinematics MATLAB toolbox aims to provide a library of the commonly used rigid body kinematics functions which will facilitate this task of using various attitude coordinates. In particular, functions providing kinematic relationships between the direction cosine matrix (rotation

*Post-Doctoral Research Associate, Aerospace Engineering Department, Texas A&M University, College Station TX 77843.

†George Eppright Chair Professor of Aerospace Engineering, Aerospace Engineering Department, Texas A&M University, College Station TX 77843.

matrix), the Euler parameters (quaternions), the Gibbs vector (classical Rodrigues parameters), the Modified Rodrigues Parameters (MRPs), the Principal Rotation Vector (PRV) and the 12 sets of Euler angles are examined. Providing direct transformations between various sets of attitude coordinates makes it possible to use whatever coordinates make numerically the most sense for the application and then translate the resulting attitude description to the desired attitude coordinates which are easier to visualize.

Another common requirement in attitude kinematics is the addition of two successive rotations. If the body orientation is known relative to some reference frame, and the reference frame orientation is known relative to an inertial frame, it may be required to compute the direct orientation description of the body attitude relative to the inertial reference frame. Unfortunately, adding two attitude vectors is not a simple task as it is in translational kinematics. One method to add or construct the composition of two successive rotations is to first translate the respective attitude vectors into direction cosine matrices. After evaluating the overall direction cosine matrix, the desired attitude vector describing the body frame to the inertial frame is extracted from the corresponding direction cosine matrix. While this method is relatively straight forward, it isn't very efficient computationally. The attitude vector addition subroutines in this toolbox make use of compact attitude additions algorithms when they exist for the selected attitude coordinates. This speeds up the task of computing an overall attitude vector.

Similarly, it may be required to compute the relative attitude vector between two given attitudes. To reuse the previous example, consider the case where both the body attitude and reference frame attitude are known relative to the inertial frame. In this case it is important to compute the *relative* attitude vector from the reference frame to the body frame. In translational kinematics this would be equivalent to subtracting two position vectors to compute the relative position error. In rotational kinematics, this task could be done by mapping all attitude descriptions first into the direction cosine matrix. However, to improve computational efficiency, the corresponding MATLAB routines use the compact, direct algorithms unique for each attitude description.

Finally, the rigid body kinematics MATLAB toolbox provides the necessary functions to compute the time derivative of the various attitude coordinates. An intermediate result of this is finding the matrix $[B]$ which relates the orthogonal components of the body angular velocity vector $\boldsymbol{\omega}$ to the various attitude coordinate rates. For all presented attitude coordinates compact closed form solutions exist for computing the inverse of $[B]$, which enhances the numerical efficiency of this common operator.

All MATLAB m-files have comment lines which explain the purpose of each operator. The user can invoke help for a particular `command` by typing `help command`. Further, all MATLAB m-files in this rigid body kinematics toolbox are listed alphabetically in an appendix and the end of this paper along with a brief description of each operator.

ATTITUDE COORDINATE TRANSFORMATIONS

This paper assumes that the reader is already familiar with the various attitude coordinate choices discussed here. Each type of attitude parameters are briefly discussed below to illustrate the notation used. Euler's principal rotation theorem states that any instantaneous orientation can be described through a corresponding principal rotation angle Φ and a unit principal rotation axis $\hat{\boldsymbol{e}}$. The principal rotation vector $\boldsymbol{\gamma}$ is then defined as

$$\boldsymbol{\gamma} = \Phi \hat{\boldsymbol{e}} \tag{1}$$

The non-singular, redundant Euler parameter (quaternion) vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)^T$ is defined through the principal rotation elements as

$$\beta_0 = \cos\frac{\Phi}{2} \qquad \beta_i = \hat{e}_i \sin\frac{\Phi}{2} \qquad \text{for } i = 1, 2, 3 \tag{2}$$

By increasing the dimension of the attitude vector by one, the Euler parameters are able to describe any orientation without encountering singularities. Note that $\boldsymbol{\beta}$ is not unique. Both $\boldsymbol{\beta}$ and $-\boldsymbol{\beta}$

describe the same orientation and obviously $\boldsymbol{\beta}^T\boldsymbol{\beta} = 1$. The Gibbs (classical Rodrigues parameter) vector $\boldsymbol{q}$ is defined as[1,2]

$$\boldsymbol{q} = \tan\frac{\Phi}{2}\hat{\boldsymbol{e}} \tag{3}$$

Clearly $\boldsymbol{q}$ is singular as $\Phi \rightarrow \pm 180$ degrees. The MRP vector $\boldsymbol{\sigma}$ is defined as[1,3–5]

$$\boldsymbol{\sigma} = \tan\frac{\Phi}{4}\hat{\boldsymbol{e}} \tag{4}$$

It is singular whenever $\Phi \rightarrow \pm 360$ degrees. However, as is the case with the Euler parameters, the MRPs are not unique. They have a corresponding "shadow" set $\boldsymbol{\sigma}^S$ which has a different singular behavior. The $\boldsymbol{\sigma}^S$ vector is singular as $\Phi \rightarrow 0$ degrees. By switching between the two feasible MRP sets it is possible to obtain a minimal attitude description which is non-singular, but discontinuous during the switching. The mapping between $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}^S$ is[4,6]

$$\boldsymbol{\sigma}^S = -\frac{1}{|\boldsymbol{\sigma}|^2}\boldsymbol{\sigma} \tag{5}$$

By switching whenever $|\boldsymbol{\sigma}| > 1$, the current MRP vector with $|\boldsymbol{\sigma}| < 1$ always indicates the shortest rotational distance back to the origin. The corresponding MATLAB routine to perform this MRP switch is `MRPswitch(q,S)`, where `q` is the 3×1 MRP vector and `S` is a positive scalar number, typically set to 1. If the magnitude of `q` exceeds `S`, then the MRP vector is mapped to the corresponding alternate shadow set.

A very common set of attitude coordinates are the Euler angles. There exist 12 distinct sets of Euler angles which differ in the sequence of successive rotations made to describe the current orientation. Each Euler angle vector is labeled as $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^T$ in this development. Symmetric sets of Euler angles have rotation sequences such as (i, j, i), where the first and last rotation are about the same instantaneous body axis. Asymmetric sets have rotation sequences such as (i, j, k) where a single-axis rotation is performed about each of the body axes. The integers i, j and k are either 1, 2 or 3 and none are repeated for the asymmetric case.

Direction Cosine Matrix

The function `...2C(q)` returns the 3×3 direction cosine matrix corresponding to the particular choice in attitude coordinates. Instead of ..., the user adds what type of attitude vector `q` is. For Euler parameters, an `EP` is added. If `q` is a Gibbs vector, then `Gibbs` is added. The MRP vector simply has `MRP` added, while the principal rotation vector has `PRV` added. If `q` is an Euler angle vector, then `Eulerijk` is added where i, j and k are replaced with the appropriate rotation sequence. Therefore, if `q` is a (3-2-1) Euler angle vector, then the corresponding direction cosine matrix is found through the command `Euler3212C(q)`. The attitude coordinate appreviations introduced here are used throughout the MATLAB subroutines.

The following equations were used to compute the corresponding direction cosine matrix $[C]$. To speed up numerical efficiency, each matrix element is defined directly. In terms of the Euler parameters, $[C]$ is defined as

$$[C(\boldsymbol{\beta})] = \begin{bmatrix} \beta_0^2+\beta_1^2-\beta_2^2-\beta_3^2 & 2\left(\beta_1\beta_2+\beta_0\beta_3\right) & 2\left(\beta_1\beta_3-\beta_0\beta_2\right) \\ 2\left(\beta_1\beta_2-\beta_0\beta_3\right) & \beta_0^2-\beta_1^2+\beta_2^2-\beta_3^2 & 2\left(\beta_2\beta_3+\beta_0\beta_1\right) \\ 2\left(\beta_1\beta_3+\beta_0\beta_2\right) & 2\left(\beta_2\beta_3-\beta_0\beta_1\right) & \beta_0^2-\beta_1^2-\beta_2^2+\beta_3^2 \end{bmatrix} \tag{6}$$

In terms of the Gibbs vector $\boldsymbol{q}$, $[C]$ is defined as

$$[C(\boldsymbol{q})] = \frac{1}{1+\boldsymbol{q}^T\boldsymbol{q}} \begin{bmatrix} 1+q_1^2-q_2^2-q_3^2 & 2\left(q_1q_2+q_3\right) & 2\left(q_1q_3-q_2\right) \\ 2\left(q_2q_1-q_3\right) & 1-q_1^2+q_2^2-q_3^2 & 2\left(q_2q_3+q_1\right) \\ 2\left(q_3q_1+q_2\right) & 2\left(q_3q_2-q_1\right) & 1-q_1^2-q_2^2+q_3^2 \end{bmatrix} \tag{7}$$

In terms of the MRP, $[C]$ is defined as

$$[C(\boldsymbol{\sigma})] = \frac{1}{(1+\sigma^2)^2}\begin{bmatrix} 4\left(\sigma_1^2-\sigma_2^2-\sigma_3^2\right)+\Sigma^2 & 8\sigma_1\sigma_2+4\sigma_3\Sigma & 8\sigma_1\sigma_3-4\sigma_2\Sigma \\ 8\sigma_2\sigma_1-4\sigma_3\Sigma & 4\left(-\sigma_1^2+\sigma_2^2-\sigma_3^2\right)+\Sigma^2 & 8\sigma_2\sigma_3+4\sigma_1\Sigma \\ 8\sigma_3\sigma_1+4\sigma_2\Sigma & 8\sigma_3\sigma_2-4\sigma_1\Sigma & 4\left(-\sigma_1^2-\sigma_2^2+\sigma_3^2\right)+\Sigma^2 \end{bmatrix} \tag{8}$$

where $\Sigma = 1-\boldsymbol{\sigma}^T\boldsymbol{\sigma}$. Using $\Phi = |\boldsymbol{\gamma}|$ and $\hat{e} = \boldsymbol{\gamma}/\Phi$, the direction cosine matrix $[C]$ is written in terms of the principal rotation vector $\boldsymbol{\gamma}$ as

$$[C] = \begin{bmatrix} e_1^2\Sigma + c\Phi & e_1e_2\Sigma + e_3s\Phi & e_1e_3\Sigma - e_2s\Phi \\ e_2e_1\Sigma - e_3s\Phi & e_2^2\Sigma + c\Phi & e_2e_3\Sigma + e_1s\Phi \\ e_3e_1\Sigma + e_2s\Phi & e_3e_2\Sigma - e_1s\Phi & e_3^2\Sigma + c\Phi \end{bmatrix} \tag{9}$$

The direction cosine matrix $[C]$ in terms of the (3-2-1) Euler angles $\boldsymbol{\theta}_{321}$ is

$$[C(\boldsymbol{\theta}_{321})] = \begin{bmatrix} c\theta_2\,c\theta_1 & c\theta_2 s\theta_1 & -s\theta_2 \\ s\theta_3 s\theta_2 c\theta_1 - c\theta_3 s\theta_1 & s\theta_3 s\theta_2 s\theta_1 + c\theta_3 c\theta_1 & s\theta_3 c\theta_2 \\ c\theta_3 s\theta_2 c\theta_1 + s\theta_3 s\theta_1 & c\theta_3 s\theta_2 s\theta_1 - s\theta_3 c\theta_1 & c\theta_3 c\theta_2 \end{bmatrix} \tag{10}$$

where $c\theta_i$ and $s\theta_i$ stand for $\cos\theta_i$ and $\sin\theta_i$ respectively. Similar equations are used to compute the remaining 11 Euler angle transformations.

The direction cosine matrix $[C]$ is translated back to various attitude coordinate vectors using the command `C2...(C)` where ... is replaced with the previously discussed attitude coordinate appreviations. To translate the direction cosine matrix $[C]$ back to the corresponding Euler parameters, the robust algorithm by Stanley in Ref. 7 is used. Both the Gibbs and the MRP vectors are found by first translating $[C]$ into Euler parameters and then to the desired coordinates using[1,4]

$$q_i = \frac{\beta_i}{\beta_0} \qquad \text{for } i = 1,2,3 \tag{11}$$

$$\sigma_i = \frac{\beta_i}{1+\beta_0} \qquad \text{for } i = 1,2,3 \tag{12}$$

The principal rotation vector $\boldsymbol{\gamma}$ is obtained through

$$\Phi = \cos^{-1}\left(\frac{\text{Trace}([C])-1}{2}\right) \tag{13}$$

$$\boldsymbol{\gamma} = \frac{\Phi}{2\sin\Phi}\begin{pmatrix}(C_{23}-C_{32}) \\ (C_{31}-C_{13}) \\ (C_{12}-C_{21})\end{pmatrix} \tag{14}$$

The various transformations to any of the 12 Euler angle sets have been well documented in the literature. A convenient table listing these mappings is found in Reference 2.

Euler Parameter Vector

To translate an Euler parameter vector `q` to another attitude parameter vector, the command `EP2...(q)` is used. The Euler parameters are transformed into classical Rodrigues parameters (Gibbs vector) using Eq. (11) or into MRPs using Eq. (12). To map to the principal rotation vector, the inverse of Eq. (2) is used.

All these mappings are rather simple since all attitude coordinates involved are essentially derived from the principal rotation elements Φ and $\hat{e}$. However, how to translate Euler parameters into corresponding Euler angles is less obvious. Junkins and Turner in Reference 2 outline an elegant method to obtain compact, analytical expressions which transform the Euler angle sets into corresponding Euler parameters. For the case where the desired Euler angles are symmetric sets, compact

inverse transformation are possible mapping Euler parameters back to Euler angles. For example, for the (3-1-3) case, the transformation is

$$\theta_1 = \tan^{-1}\left(\frac{\beta_3}{\beta_0}\right) + \tan^{-1}\left(\frac{\beta_2}{\beta_1}\right) \tag{15}$$

$$\theta_2 = 2\cos^{-1}\left(\sqrt{\beta_0^2 + \beta_3^2}\right) \tag{16}$$

$$\theta_3 = \tan^{-1}\left(\frac{\beta_3}{\beta_0}\right) - \tan^{-1}\left(\frac{\beta_2}{\beta_1}\right) \tag{17}$$

Note the range of the symmetric Euler angles is chosen such that $-\pi \leq \theta_1, \theta_3 \leq \pi$ and $0 < \theta_2 < \pi$. However, for the asymmetric Euler angle case, it is not attractive to invert the Euler angle to Euler parameter relationship in Ref. 2. Instead, the Euler parameters are used to compute the necessary direction cosine matrix elements and then use standard techniques to map $[C]$ to the required Euler angles. For example, to obtain the $(3-2-1)$ Euler angles from Euler parameters, we would then compute

$$\theta_1 = \tan^{-1}\left(\frac{C_{12}}{C_{11}}\right) = \tan^{-1}\left(\frac{2\left(\beta_1\beta_2 + \beta_0\beta_3\right)}{\beta_0^2+\beta_1^2-\beta_2^2-\beta_3^2}\right) \tag{18}$$

$$\theta_2 = \sin^{-1}\left(-C_{13}\right) = \sin^{-1}\left(-2\left(\beta_1\beta_3 - \beta_0\beta_2\right)\right) \tag{19}$$

$$\theta_3 = \tan^{-1}\left(\frac{C_{23}}{C_{33}}\right) = \tan^{-1}\left(\frac{2\left(\beta_2\beta_3 + \beta_0\beta_1\right)}{\beta_0^2-\beta_1^2-\beta_2^2+\beta_3^2}\right) \tag{20}$$

The asymmetric Euler angle range is chosen such that $-\pi \leq \theta_1, \theta_3 \leq \pi$ and $-\pi/2 < \theta_2 < \pi/2$. Analogous transformations are used to map the Euler parameters into the remaining 10 Euler angle sets.

Gibbs Vector

To translate a Gibbs vector `q` (classical Rodrigues parameters) into other attitude coordinates, the command `Gibbs2...(q)` is used. All of these transformations will inherently be singular as $\Phi \to$ 180 degrees due to the singular nature of the Gibbs vector itself. The mapping from a Gibbs vector to an Euler parameter vector is accomplished using

$$\beta_0 = \frac{1}{\sqrt{1+\boldsymbol{q}^T\boldsymbol{q}}} \qquad \beta_i = \frac{q_i}{\sqrt{1+\boldsymbol{q}^T\boldsymbol{q}}} \qquad \text{for } i = 1,2,3 \tag{21}$$

A MRP vector is obtained through[3]

$$\boldsymbol{\sigma} = \frac{1}{1+\sqrt{1+\boldsymbol{q}^T\boldsymbol{q}}}\boldsymbol{q} \tag{22}$$

while the principal rotation vector is obtained by solving Eq. (3). To obtain any one of the 12 Euler angle sets, the Gibbs vector is first mapped into an Euler parameter vector and then into the corresponding Euler angles. All the attitude coordinates which are derived from the principal rotation elements can trivially be obtained from the Euler parameters. Therefore, since an efficient algorithm exists to map between Euler angles and Euler parameters, translating to or from the Euler angles is typically done via the Euler parameters.

Modified Rodrigues Parameter Vector

To translate a MRP vector `q` to another attitude coordinate set, the command `MRP2...(q)` is used. The Euler parameters are obtained through

$$\beta_0 = \frac{1-\sigma^2}{1+\sigma^2} \qquad \beta_i = \frac{2\sigma_i}{1+\sigma^2} \qquad \text{for } i = 1,2,3 \tag{23}$$

where the notation $\sigma^2 = \boldsymbol{\sigma}^T\boldsymbol{\sigma}$ is used. The Gibbs vector is computed through[3]

$$\boldsymbol{q} = \frac{2}{1-\sigma^2}\boldsymbol{\sigma} \tag{24}$$

The principal rotation vector $\boldsymbol{\gamma}$ is obtained by solving Eq. (4), while the various Euler angle sets are computed by first mapping the MRP vector into Euler parameters and then into respective Euler angles.

Principal Rotation Vector

To translate the principal rotation vector `q` into another attitude coordinate set, the command `PRV2...(q)` is used. To obtain Euler parameters, classical or modified Rodrigues parameters, the 3×1 principal rotation vector is first decomposed into the principal rotation elements Φ and $\hat{\boldsymbol{e}}$ using the command `PRV2elem(q)`. Then Eqs. (2) through (4) are used to compute the appropriate attitude coordinates. The Euler angles are once again obtained by first mapping the principal rotation vector into Euler parameters and then into the Euler angles.

Euler Angle Set Vectors

To translate an (i,j,k) Euler angle set `q`=$(\theta_1, \theta_2, \theta_3)$ into another attitude coordinate set, the command `Eulerijk2...(q)` is used. Excluded here are transformations between various Euler angle sets. These could be computed by first mapping the given set into another set of attitude coordinates such as the direction cosine matrix or the Euler parameters. To translate Euler angles into Euler parameters, the compact direct transformations in Ref. 2 are used. Transformations for two commonly used Euler angle sets are shown below. For the (3-1-3) set, the transformation is

$$\beta_0 = \cos\frac{\theta_2}{2}\cos\left(\frac{\theta_1+\theta_3}{2}\right) \tag{25a}$$

$$\beta_1 = \sin\frac{\theta_2}{2}\cos\left(\frac{\theta_1-\theta_3}{2}\right) \tag{25b}$$

$$\beta_2 = \sin\frac{\theta_2}{2}\sin\left(\frac{\theta_1-\theta_3}{2}\right) \tag{25c}$$

$$\beta_3 = \cos\frac{\theta_2}{2}\sin\left(\frac{\theta_1+\theta_3}{2}\right) \tag{25d}$$

For the (3-2-1) set, the transformation is

$$\beta_0 = \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2} \tag{26a}$$

$$\beta_1 = \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2} - \sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2} \tag{26b}$$

$$\beta_2 = \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\cos\frac{\theta_3}{2} + \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\sin\frac{\theta_3}{2} \tag{26c}$$

$$\beta_3 = \sin\frac{\theta_1}{2}\cos\frac{\theta_2}{2}\cos\frac{\theta_3}{2} - \cos\frac{\theta_1}{2}\sin\frac{\theta_2}{2}\sin\frac{\theta_3}{2} \tag{26d}$$

The remaining 10 transformations are listed in Ref. 2. The transformations from Euler angles to the classical or modified Rodrigues parameters or the principal rotation vector are all performed via the Euler parameters.

SUCCESSIVE AND RELATIVE ATTITUDE VECTORS

When working with multiple reference frames, it is often required to form the composition of two successive rotations. For example, if the attitude of a free-floating robot is known relative to the space station, and the space station attitude is known relative to some inertial reference frame, then, in order to find the robot attitude relative to the inertial frame, the two relative attitudes have to be

summed. Let the generic attitude vector $\boldsymbol{q}'$ describe the orientation of the $\mathcal{B}$ frame relative to the $\mathcal{N}$ frame, while $\boldsymbol{q}''$ describes the orientation of the $\mathcal{F}$ frame relative to the $\mathcal{B}$ frame. The attitude of $\mathcal{F}$ relative to $\mathcal{N}$ is given by $\boldsymbol{q}$. If this were a translational problem, then $\boldsymbol{q} = \boldsymbol{q}'' + \boldsymbol{q}'$. However, adding rotations is not this simple. A very fundamental method to add rotations is to use compositions of successive rotations in terms of the direction cosine matrices:

$$[FN(\boldsymbol{q})] = [FB(\boldsymbol{q}'')][BN(\boldsymbol{q}')] \tag{27}$$

In fact, all of the compact successive rotations algorithms listed below are derived from this equation. While using Eq. (27) directly to sum two attitude vectors is relatively straight forward to program, it isn't very efficient numerically. For most attitude coordinate cases more compact transformations are possible that will directly compute $\boldsymbol{q}$ from the two relative vector $\boldsymbol{q}'$ and $\boldsymbol{q}''$. The command to add two successive rotations is `add...(q1,q2)` where `q1` is the first rotation vector and `q2` is the second. Note that these commands assume that *both* relative attitude vectors are of the same type (i.e. Euler parameters, MRPs, ...).

In other applications it is often required that the *relative* attitude vector $\boldsymbol{q}''$ is found given $\boldsymbol{q}$ and $\boldsymbol{q}'$. For example, in a attitude tracking problem, assume there is a commanded reference frame and the body reference frame. These two frames are typically given at any point of time, but the feedback control law requires the relative attitude error vector from the commanded frame to the actual body frame. If this were a translational problem, this would simply involve subtracting one position vector from another through $\boldsymbol{q}'' = \boldsymbol{q} - \boldsymbol{q}'$. For rotations, the relative vector must be solved differently. Using Eq. (27) we find

$$[FB(\boldsymbol{q}'')] = [FN(\boldsymbol{q})][BN(\boldsymbol{q}')]^T \tag{28}$$

While this matrix equation can be used to compute $\boldsymbol{q}''$, for many attitude coordinates the "rotational subtraction" algorithm can be reduced to a more compact form. Since this process in essence subtracts one rotational vector from another to find the relative attitude vector, the corresponding MATLAB command command is called `sub...(q1,q2)`. Here the relative attitude vector from `q2` to `q1` is computed.

A very attractive property of the Euler parameters is that solving Eq. (27) for the overall Euler parameters leads to a very elegant, bi-linear relationship. Performing the successive rotations $\boldsymbol{\beta}'$ and $\boldsymbol{\beta}''$ leads to[1,2]

$$\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{bmatrix} \beta'_0 & -\beta'_1 & -\beta'_2 & -\beta'_3 \\ \beta'_1 & \beta'_0 & -\beta'_3 & \beta'_2 \\ \beta'_2 & \beta'_3 & \beta'_0 & -\beta'_1 \\ \beta'_3 & -\beta'_2 & \beta'_1 & \beta'_0 \end{bmatrix} \begin{pmatrix} \beta''_0 \\ \beta''_1 \\ \beta''_2 \\ \beta''_3 \end{pmatrix} \tag{29}$$

Since the matrix relating $\boldsymbol{\beta}$ to $\boldsymbol{\beta}''$ is orthogonal, "subtracting" $\boldsymbol{\beta}'$ from $\boldsymbol{\beta}$ is simply

$$\begin{pmatrix} \beta''_0 \\ \beta''_1 \\ \beta''_2 \\ \beta''_3 \end{pmatrix} = \begin{bmatrix} \beta'_0 & \beta'_1 & \beta'_2 & \beta'_3 \\ -\beta'_1 & \beta'_0 & \beta'_3 & -\beta'_2 \\ -\beta'_2 & -\beta'_3 & \beta'_0 & \beta'_1 \\ -\beta'_3 & \beta'_2 & -\beta'_1 & \beta'_0 \end{bmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} \tag{30}$$

These simple bi-linear formulas form the basis for developing corresponding formulas for the classical or modified Rodrigues parameters and the principal rotation vector.

Given two Gibbs vectors $\boldsymbol{q}'$ and $\boldsymbol{q}''$, their rotational sum $\boldsymbol{q}$ is computed through[1,8]

$$\boldsymbol{q} = \frac{\boldsymbol{q}'' + \boldsymbol{q}' - \boldsymbol{q}'' \times \boldsymbol{q}'}{1 - \boldsymbol{q}'' \cdot \boldsymbol{q}'} \tag{31}$$

while the relative vector $\boldsymbol{q}''$ is found through

$$\boldsymbol{q}'' = \frac{\boldsymbol{q} - \boldsymbol{q}' + \boldsymbol{q} \times \boldsymbol{q}'}{1 + \boldsymbol{q} \cdot \boldsymbol{q}'} \tag{32}$$

Summing two MRP vectors $\boldsymbol{\sigma}'$ and $\boldsymbol{\sigma}''$ takes on a slightly more complex form than the Gibbs vector algorithm.[1]

$$\boldsymbol{\sigma} = \frac{(1-|\boldsymbol{\sigma}'|^2)\boldsymbol{\sigma}'' + (1-|\boldsymbol{\sigma}''|^2)\boldsymbol{\sigma}' - 2\boldsymbol{\sigma}'' \times \boldsymbol{\sigma}'}{1+|\boldsymbol{\sigma}'|^2|\boldsymbol{\sigma}''|^2 - 2\boldsymbol{\sigma}' \cdot \boldsymbol{\sigma}''} \tag{33}$$

The relative MRP vector $\boldsymbol{\sigma}''$ is computed through

$$\boldsymbol{\sigma}'' = \frac{(1-|\boldsymbol{\sigma}'|^2)\boldsymbol{\sigma} - (1-|\boldsymbol{\sigma}|^2)\boldsymbol{\sigma}' + 2\boldsymbol{\sigma} \times \boldsymbol{\sigma}'}{1+|\boldsymbol{\sigma}'|^2|\boldsymbol{\sigma}|^2 + 2\boldsymbol{\sigma}' \cdot \boldsymbol{\sigma}} \tag{34}$$

To sum two principal rotation vectors $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$, they are first translated into their corresponding principal rotation elements $(\Phi_1, \hat{\boldsymbol{e}}_1)$ and $(\Phi_2, \hat{\boldsymbol{e}}_2)$. The sum $\boldsymbol{\gamma} = \Phi\hat{\boldsymbol{e}}$ of $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$ is then computed through[1]

$$\Phi = 2\cos^{-1}\left(\cos\frac{\Phi_1}{2}\cos\frac{\Phi_2}{2} - \sin\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2}\hat{\boldsymbol{e}}_1 \cdot \hat{\boldsymbol{e}}_2\right) \tag{35}$$

$$\hat{\boldsymbol{e}} = \frac{\cos\frac{\Phi_2}{2}\sin\frac{\Phi_1}{2}\hat{\boldsymbol{e}}_1 + \cos\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2}\hat{\boldsymbol{e}}_2 + \sin\frac{\Phi_1}{2}\sin\frac{\Phi_2}{2}\hat{\boldsymbol{e}}_1 \times \hat{\boldsymbol{e}}_2}{\sin\frac{\Phi}{2}} \tag{36}$$

The relative principal rotation vector $\boldsymbol{\gamma}_2 = \Phi_2\hat{\boldsymbol{e}}_2$ is found through

$$\Phi_2 = 2\cos^{-1}\left(\cos\frac{\Phi}{2}\cos\frac{\Phi_1}{2} + \sin\frac{\Phi}{2}\sin\frac{\Phi_1}{2}\hat{\boldsymbol{e}} \cdot \hat{\boldsymbol{e}}_1\right) \tag{37}$$

$$\hat{\boldsymbol{e}}_2 = \frac{\cos\frac{\Phi_1}{2}\sin\frac{\Phi}{2}\hat{\boldsymbol{e}} - \cos\frac{\Phi}{2}\sin\frac{\Phi_1}{2}\hat{\boldsymbol{e}}_1 + \sin\frac{\Phi}{2}\sin\frac{\Phi_1}{2}\hat{\boldsymbol{e}} \times \hat{\boldsymbol{e}}_1}{\sin\frac{\Phi_2}{2}} \tag{38}$$

To sum two Euler angle sets, the symmetric and asymmetric sets must be considered separately. Let the first rotation be expressed by the Euler angle vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ and the second rotation by the vector $\boldsymbol{\phi} = (\phi_1, \phi_2, \phi_3)$. The rotational sum of these two attitude coordinates is given by the Euler angle vector $\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \varphi_3)$. In Reference 9 Junkins and Shuster show very elegantly using spherical triangles that *symmetric Euler angle sets* all have the identical rotational summation formula. Therefore, if $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are symmetric Euler angle vectors, then the summation of these two rotations is given by[1,9]

$$\varphi_1 = \theta_1 + \tan^{-1}\left(\frac{\sin\theta_2\sin\phi_2\sin(\theta_3+\phi_1)}{\cos\phi_2 - \cos\theta_2\cos\varphi_2}\right) \tag{39}$$

$$\varphi_2 = \cos^{-1}\left(\cos\theta_2\cos\phi_2 - \sin\theta_2\sin\phi_2\cos(\theta_3+\phi_1)\right) \tag{40}$$

$$\varphi_3 = \phi_3 + \tan^{-1}\left(\frac{\sin\theta_2\sin\phi_2\sin(\theta_3+\phi_1)}{\cos\theta_2 - \cos\phi_2\cos\varphi_2}\right) \tag{41}$$

The relative orientation vector $\boldsymbol{\phi}$ is found through

$$\phi_1 = -\theta_3 + \tan^{-1}\left(\frac{\sin\theta_2\sin\varphi_2\sin(\varphi_1-\theta_1)}{\cos\theta_2\cos\phi_2 - \cos\varphi_2}\right) \tag{42}$$

$$\phi_2 = \cos^{-1}\left(\cos\theta_2\cos\varphi_2 + \sin\theta_2\sin\varphi_2\cos(\varphi_1-\theta_1)\right) \tag{43}$$

$$\phi_3 = \varphi_3 - \tan^{-1}\left(\frac{\sin\theta_2\sin\varphi_2\sin(\varphi_1-\theta_1)}{\cos\theta_2 - \cos\phi_2\cos\varphi_2}\right) \tag{44}$$

Unfortunately, no such simple transformations are known to exist for the asymmetric Euler angles. In these cases, finding the sum or the relative attitude vector is performed indirectly using the direction cosine matrix formulations in Eqs. (27) and (28).

DIFFERENTIAL KINEMATIC EQUATIONS

The body angular velocity time history $\boldsymbol{\omega}(t)$ is typically solved using some kinetic differential equation such as Euler's equation of motion. Given some initial attitude vector $\boldsymbol{q}(t_0)$ and this $\boldsymbol{\omega}(t)$ vector, the time evolution of the attitude vector $\boldsymbol{q}$ is found by solving the corresponding kinematic differential equation. Note that while the kinetic differential equation is often an approximation, the kinematic differential equation is exact. The attitude coordinate rate vector $\dot{\boldsymbol{q}}$ is related to $\boldsymbol{\omega}$ through a matrix $[B(\boldsymbol{q})]$.

$$\dot{\boldsymbol{q}} = [B(\boldsymbol{q})]\boldsymbol{\omega} \tag{45}$$

The MATLAB command `d...(q,w)` computes the time derivative of the attitude vector `q` for a given body angular velocity vector `w`. For example, if $\boldsymbol{q}$ is a (1-2-3) Euler angle vector, then the command `dEuler123(q)` would be invoked. Subroutines are also provided that compute just the $[B(\boldsymbol{q})]$ matrix and it's inverse. All attitude coordinates discussed have compact analytical inverse formulas for $[B(\boldsymbol{q})]$. The $[B(\boldsymbol{q})]$ matrix is computed with the command `Bmat...(q)` and its inverse with `Binv...(q)`.

For the Euler parameters, the differential kinematic equation is written as

$$\dot{\boldsymbol{\beta}} = \frac{1}{2}[B(\boldsymbol{\beta})]\boldsymbol{\omega} \tag{46}$$

with the 4×3 matrix $[B(\boldsymbol{\beta})]$ defined as

$$[B(\boldsymbol{\beta})] = \begin{bmatrix} -\beta_1 & -\beta_2 & -\beta_3 \\ \beta_0 & -\beta_3 & \beta_2 \\ \beta_3 & \beta_0 & -\beta_1 \\ -\beta_2 & \beta_1 & \beta_0 \end{bmatrix} \tag{47}$$

Since $[B(\boldsymbol{\beta})]$ is a 4×3 matrix, talking about its inverse is not completely proper. In this context `BinvEP(q)` returns a 3×4 matrix $[Bi(\boldsymbol{\beta})]$ such that

$$\boldsymbol{\omega} = 2[Bi(\boldsymbol{\beta})]\dot{\boldsymbol{\beta}} \tag{48}$$

with $[Bi(\boldsymbol{\beta})]$ defined as

$$[Bi(\boldsymbol{\beta})] = \begin{bmatrix} -\beta_1 & \beta_0 & \beta_3 & -\beta_2 \\ -\beta_2 & -\beta_3 & \beta_0 & \beta_1 \\ -\beta_3 & \beta_2 & -\beta_1 & \beta_0 \end{bmatrix} \tag{49}$$

Note that $[Bi(\boldsymbol{\beta})] = [B(\boldsymbol{\beta})]^T$. The differential kinematic equation of the Gibbs vector is written as

$$\dot{\boldsymbol{q}} = \frac{1}{2}[B(\boldsymbol{q})]\boldsymbol{\omega} \tag{50}$$

with the 3×3 matrix $[B(\boldsymbol{q})]$ defined as

$$[B(\boldsymbol{q})] = \begin{bmatrix} 1 + q_1^2 & q_1 q_2 - q_3 & q_1 q_3 + q_2 \\ q_2 q_1 + q_3 & 1 + q_2^2 & q_2 q_3 - q_1 \\ q_3 q_1 - q_2 & q_3 q_2 + q_1 & 1 + q_3^2 \end{bmatrix} \tag{51}$$

The factor $\frac{1}{2}$ is extracted from $[B(\boldsymbol{q})]$ such that linearizing $[B(\boldsymbol{q})]$ about $\boldsymbol{q} = 0$ provides $[B(\boldsymbol{q})] = I_{3\times 3}$. The inverse of $[B(\boldsymbol{q})]$ is given by[1]

$$[B(\boldsymbol{q})]^{-1} = \frac{1}{1 + \boldsymbol{q}^T\boldsymbol{q}} \begin{pmatrix} 1 & q_3 & -q_2 \\ -q_3 & 1 & q_1 \\ q_2 & -q_1 & 1 \end{pmatrix} \tag{52}$$

The kinematic differential equation of the MRPs is given by

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}[B(\boldsymbol{\sigma})]\boldsymbol{\omega} \tag{53}$$

with the 3×3 matrix $[B(\boldsymbol{\sigma})]$ defined as

$$[B(\boldsymbol{\sigma})] = \begin{bmatrix} 1-\sigma^2+2\sigma_1^2 & 2(\sigma_1\sigma_2-\sigma_3) & 2(\sigma_1\sigma_3+\sigma_2) \\ 2(\sigma_2\sigma_1+\sigma_3) & 1-\sigma^2+2\sigma_2^2 & 2(\sigma_2\sigma_3-\sigma_1) \\ 2(\sigma_3\sigma_1-\sigma_2) & 2(\sigma_3\sigma_2+\sigma_1) & 1-\sigma^2+2\sigma_3^2 \end{bmatrix} \tag{54}$$

The $[B(\boldsymbol{\sigma})]$ matrix has the remarkable property that it is near-orthogonal. To within a scaling factor, it's inverse is equal to its transpose.[3,10]

$$[B(\boldsymbol{\sigma})]^{-1} = \frac{1}{(1+\sigma^2)^2}[B(\boldsymbol{\sigma})]^T \tag{55}$$

The principal rotation vector differential kinematic equation is written as

$$\dot{\gamma} = [B(\gamma)]\boldsymbol{\omega} \tag{56}$$

with the 3×3 matrix $[B(\gamma)]$ given by[1]

$$[B(\gamma)] = \left[I + \frac{1}{2}[\tilde{\gamma}] + \frac{1}{\Phi^2}\left(1 - \frac{\Phi}{2}\cot\left(\frac{\Phi}{2}\right)\right)[\tilde{\gamma}]^2\right] \tag{57}$$

where $\Phi = ||\gamma||$. The inverse of $[B(\gamma)]$ is[1]

$$[B(\gamma)]^{-1} = \left[I - \left(\frac{1-\cos\Phi}{\Phi^2}\right)[\tilde{\gamma}] + \left(\frac{\Phi-\sin\Phi}{\Phi^3}\right)[\tilde{\gamma}]^2\right] \tag{58}$$

The kinematic relationships between the Euler angle rates and the body angular velocity vector have been well studied in the literature. Ref. 2 provides a complete list of all 12 $[B(\boldsymbol{\theta})]$ matrices and their inverses. The general Euler angle differential kinematic equation is written as

$$\dot{\boldsymbol{\theta}} = [B(\boldsymbol{\theta})]\boldsymbol{\omega} \tag{59}$$

For the (3-1-3) Euler angle case, the $[B(\boldsymbol{\theta})]$ matrix is written as[1,2]

$$[B(\boldsymbol{\theta}_{313})] = \frac{1}{\sin\theta_2}\begin{bmatrix} \sin\theta_3 & \cos\theta_3 & 0 \\ \cos\theta_3\sin\theta_2 & -\sin\theta_3\sin\theta_2 & 0 \\ -\sin\theta_3\cos\theta_2 & -\cos\theta_3\cos\theta_2 & \sin\theta_2 \end{bmatrix} \tag{60}$$

The inverse of $[B(\boldsymbol{\theta}_{313})]$ is given by

$$[B(\boldsymbol{\theta}_{313})]^{-1} = \begin{bmatrix} \sin\theta_3\sin\theta_2 & \cos\theta_3 & 0 \\ \cos\theta_3\sin\theta_2 & -\sin\theta_3 & 0 \\ \cos\theta_2 & 0 & 1 \end{bmatrix} \tag{61}$$

The $[B(\theta)]$ and $[B(\boldsymbol{\theta})]^{-1}$ matrices for the remaining 11 Euler angle sets are computed in an analogous manner.

CONCLUSION

A rigid body kinematics toolbox is presented. Transformations between the direction cosine matrix, the Euler parameters, the Gibbs vector, the modified Rodrigues parameters, the principal rotation vector and the various Euler angle sets are provided. These commands facilitate switching between various types of attitude coordinates and more easily allow the programmer to use whatever attitude coordinates are proper numerically. Also, subroutines are provided to composite two successive rotations and to compute a relative attitude vector. Care is taken to provide numerically efficient routines by avoiding whenever possible having to intermittently translate attitude coordinates to the direction cosine matrices. Finally, subroutines are provided that compute the various time derivatives of the attitude coordinate vectors, along with their mapping matrix and its inverse.

REFERENCES

1. Shuster, Malcom D., "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 439–517.
2. Junkins, John L. and Turner, James D., *Optimal Spacecraft Rotational Maneuvers*. Amsterdam, Netherlands: Elsevier Science Publishers, 1986.
3. Schaub, Hanspeter, *Novel Coordinates for Nonlinear Multibody Motion with Applications to Spacecraft Dynamics and Control*. PhD thesis, Texas A&M University, College Station, TX, May 1998.
4. Schaub, Hanspeter and Junkins, John L., "Stereographic Orientation Parameters for Attitude Dynamics: A Generalization of the Rodrigues Parameters," *Journal of the Astronautical Sciences*, Vol. 44, No. 1, Jan.–Mar. 1996, pp. 1–19.
5. Tsiotras, Panagiotis and Longuski, Jams M., "A New Parameterization of the Attititude Kinematics," *Journal of the Astronautical Sciences*, Vol. 43, No. 3, 1996, pp. 342–262.
6. Marandi, S. R. and Modi, V. J., "A Preferred Coordinate System and the Associated Orientation Representation in Attitude Dynamics," *Acta Astronautica*, Vol. 15, No. 11, 1987, pp. 833–843.
7. Stanley, W. S., "Quaternion from Rotation Matrix," *AIAA Journal of Guidance and Control*, Vol. I, No. 3, May 1978, pp. 223–224.
8. Federov, F., *The Lorentz Group*. Moscow: Nauka, 1979.
9. Junkins, John L. and Shuster, Malcolm D., "The Geometry of Euler Angles," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 531–543.
10. Tsiotras, Panagiotis, Junkins, John L., and Schaub, Hanspeter, "Higher Order Cayley Transforms with Applications to Attitude Representations," *Journal of Guidance, Control and Dynamics*, Vol. 20, No. 3, May–June 1997, pp. 528–534.

APPENDIX: LIST OF MATLAB COMMANDS

`addEP(q1,q2)`	Sum the two Euler parameter vectors.
`addEulerijk(q1,q2)`	Sum the two (i-j-k) Euler angle vectors.
`addGibbs(q1,q2)`	Sum the two Gibbs vectors.
`addMRP(q1,q2)`	Sum the two MRP vectors.
`addPRV(q1,q2)`	Sum the two principal rotation vectors.
`BinvEP(q)`	Compute the inverse of $[B(\boldsymbol{\beta})]$.
`BinvEulerijk(q)`	Compute the inverse of $[B(\boldsymbol{\theta}_{ijk})]$.
`BinvGibbs(q)`	Compute the inverse of $[B(\boldsymbol{q})]$.
`BinvMRP(q)`	Compute the inverse of $[B(\boldsymbol{\sigma})]$.
`BinvPRV(q)`	Compute the inverse of $[B(\boldsymbol{\gamma})]$.
`BmatEP(q)`	Compute the matrix $[B(\boldsymbol{\beta})]$
`BmatEulerijk(q)`	Compute the matrix $[B(\boldsymbol{\theta}_{ijk})]$
`BmatGibbs(q)`	Compute the matrix $[B(\boldsymbol{q})]$
`BmatMRP(q)`	Compute the matrix $[B(\boldsymbol{\sigma})]$
`BmatPRV(q)`	Compute the matrix $[B(\boldsymbol{\gamma})]$
`C2EP(C)`	Extract the Euler parameters from $[C]$.
`C2Eulerijk(C)`	Extract the (i-j-k) Euler angles from $[C]$.
`C2Gibbs(C)`	Extract the Gibbs vector from $[C]$.
`C2MRP(C)`	Extract the MRP vector from $[C]$.
`C2PRV(C)`	Extract the principal rotation vector from $[C]$.
`dEP(q,w)`	Compute the Euler parameter time derivative.
`dEulerijk(q,w)`	Compute the (i-j-k) Euler angles time derivative.
`dGibbs(q,w)`	Compute the Gibbs vector time derivative.
`EP2C(q)`	Translate the Euler parameters into $[C]$.

`Eulerijk2C(q)`	Translate the (i-j-k) Euler angles into $[C]$.
`Gibbs2C(q)`	Translate the Gibbs vector into $[C]$.
`MRP2C(q)`	Translate the MRP vector into $[C]$.
`PRV2C(q)`	Translate the principal rotation vector into $[C]$.
`dMRP(q,w)`	Compute the MRP vector time derivative.
`dPRV(q,w)`	Compute the principal rotation vector time derivative.
`elem2PRV(q)`	Translates the $(\Phi, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ into the principal rotation vector.
`EP2Eulerijk`	Translate Euler parameters into (i-j-k) Euler angles.
`EP2Gibbs`	Translate Euler parameters into a Gibbs vector.
`EP2MRP`	Translate Euler parameters into a MRP vector.
`EP2PRV`	Translate Euler parameters into a PRV vector.
`Euler1(theta)`	Returns the elementary rotation matrix about the first body axis.
`Euler2(theta)`	Returns the elementary rotation matrix about the second body axis.
`Euler3(theta)`	Returns the elementary rotation matrix about the third body axis.
`Eulerijk2EP(q)`	Translate the (i-j-k) Euler angles into Euler parameters.
`Eulerijk2Gibbs(q)`	Translate the (i-j-k) Euler angles into the Gibbs vector.
`Eulerijk2MRP(q)`	Translate the (i-j-k) Euler angles into MRPs.
`Eulerijk2PRV(q)`	Translate the (i-j-k) Euler angles into the principal rotation vector.
`Gibbs2EP(q)`	Translate the Gibbs vector into Euler parameters.
`Gibbs2Eulerijk(q)`	Translate the Gibbs vector into (i-j-k) Euler angles.
`Gibbs2MRP(q)`	Translate the Gibbs vector into MRPs.
`Gibbs2PRV(q)`	Translate the Gibbs vector into the principal rotation vector.
`MRP2EP(q)`	Translate the MRPs into Euler parameters.
`MRP2Eulerijk(q)`	Translate the MRPs into (i-j-k) Euler angles.
`MRP2Gibbs(q)`	Translate the MRPs into the Gibbs vector.
`MRP2PRV(q)`	Translate the MRPs into the principal rotation vector.
`MRPswitch(q,S)`	Switch the MRP vector such that $\|\boldsymbol{\sigma}\|^2 < S$.
`PRV2elem(q)`	Translates the principal rotation vector to $(\Phi, \hat{e}_1, \hat{e}_2, \hat{e}_3)$.
`PRV2EP(q)`	Translates the principal rotation vector to Euler parameters.
`PRV2Eulerijk(q)`	Translates the principal rotation vector to (i-j-k) Euler angles.
`PRV2Gibbs(q)`	Translates the principal rotation vector to the Gibbs vector.
`PRV2MRP(q)`	Translates the principal rotation vector to MRPs.
`subEP(q,q1)`	Compute the relative Euler parameter vector from `q1` to `q`.
`subEulerijk(q,q1)`	Compute the relative (i-j-k) Euler angles vector from `q1` to `q`.
`subGibbs(q,q1)`	Compute the relative Gibbs vector from `q1` to `q`.
`subMRP(q,q1)`	Compute the relative MRP vector from `q1` to `q`.
`subPRV(q,q1)`	Compute the relative PRV vector from `q1` to `q`.