An Efficient Method for Extracting Euler Angles from Direction Cosine Matrices

Dmitry Savransky[1]
Lawrence Livermore National Labs, Livermore, CA

N. Jeremy Kasdin[2]
Princeton University, Princeton, NJ

We derive an algorithm to calculate any of the 24 standard Euler angle sets from a direction cosine matrix, using the set of rotation axes as an indexing set and thereby removing the requirement of tabulating separate equations for the various possible rotation sequences. We describe the specific structure of the direction cosine matrix that makes this possible and develop the basic equations describing ordered simple rotations. We also present a MATLAB implementation based on this algorithm.

I. Introduction

Many methods exist for describing the orientation and rotation of a rigid body in 3 dimensional space. Euler angles—a set of 3 ordered simple rotations—are one of the simplest representations, but suffer from potential ambiguities caused by the loss of one degree of freedom in certain orientations, known as gimbal lock. These ambiguities can be solved in physical systems by adding another gimbal (creating an overdetermined system), or by using two or more orderings and switching between them when one is near gimbal lock. Alternate representations such as quaternions or Rodrigues parameters solve this problem by introducing a fourth coordinate, at the expense of a non-linear constraint on the parameter set. All of these can be related to a direction cosine matrix (DCM), which encodes rotations using nine values, and is thus significantly overdetermined, but represents a

---

[1] Postdoctoral Fellow, LLNL, Livermore, CA
[2] Professor of Mechanical and Aerospace Engineering, Princeton University

unique encoding of the rotation. The DCM is also the linear operator used to transform coordinate representations of vectors between different reference frames.

Any DCM can be related to one of 24 standard Euler angle sets, as shown in §II. Typically, this is done by solving solving for each specific set algebraically, and tabulating the results, as in Kane et al. [1]. When implementing code to solve for Euler angles, this translates to writing programs with up to 24 logical branches. This note provides a single algorithm for calculating any of the Euler angle sets, making it much simpler to work with multiple sets in situations where frequent gimbal locks may occur.

II. The Direction Cosine Matrix

We will use the conventions and symbols used in Kane et al. [1] and Kasdin and Paley [2], so that vectors will be denoted by lowercase, bold symbols (i.e., $\mathbf{v}$) and second-rank tensors will be denoted by uppercase, bold symbols (i.e., $\mathbf{D}$). When expressing these in terms of the coordinates of a specific reference frame (e.g., $\mathcal{I}$), we will write $[\mathbf{v}]_{\mathcal{I}}$, where the brackets represent a 3×1 column vector whose entries are the Cartesian coordinates of $\mathbf{v}$ in frame $\mathcal{I}$ (or a 3×3 matrix, in the case of second-rank tensors). Matrices will be written as uppercase symbols (i.e., A) and an element (i, j) of matrix A will be denoted by A_{ij}.

Given a reference frame $\mathcal{A} = (O, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$, where O is the frame origin and $\{\mathbf{a}_i\}_{i=1}^3$ a set of dextral unit vectors, and a frame $\mathcal{B} = (O, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$, we define a simple rotation as one where $\mathcal{A}$ and $\mathcal{B}$ share a common axis $\boldsymbol{\lambda}$ such that:

$$[\boldsymbol{\lambda}]_{\mathcal{A}} = [\boldsymbol{\lambda}]_{\mathcal{B}} \, . \tag{1}$$

The magnitude of the rotation is denoted by an angle θ, whose value is given by:

$$\cos\theta = \boldsymbol{\alpha} \cdot \boldsymbol{\beta} \, , \tag{2}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are unit vectors in $\mathcal{A}$ and $\mathcal{B}$, respectively, that are equal when $\mathcal{A}$ and $\mathcal{B}$ are aligned

(i.e., when $\mathbf{a}_i = \mathbf{b}_i \, \forall i$). We can convert the coordinates of a vector $\mathbf{v}$ via the DCM, ${}^{\mathcal{A}}C^{\mathcal{B}}$, as:

$$[\mathbf{v}]_{\mathcal{A}} = {}^{\mathcal{A}}C^{\mathcal{B}} \, [\mathbf{v}]_{\mathcal{B}} \, . \tag{3}$$

The DCM ${}^{\mathcal{A}}C^{\mathcal{B}}$ is thus the matrix representation of the tensor ${}^{\mathcal{A}}\mathbf{C}^{\mathcal{B}}$, where $\boldsymbol{\beta} = {}^{\mathcal{A}}\mathbf{C}^{\mathcal{B}} \cdot \boldsymbol{\alpha}$, expressed with respect to frame $\mathcal{A}$:

$${}^{\mathcal{A}}C^{\mathcal{B}} = \left[{}^{\mathcal{A}}\mathbf{C}^{\mathcal{B}} \right]_{\mathcal{A}} \, . \tag{4}$$

An element (i, j) of the DCM ${}^{\mathcal{A}}C^{\mathcal{B}}$ is given by $\mathbf{a}_i \cdot \mathbf{b}_j$ so that:

$${}^{\mathcal{A}}C^{\mathcal{B}}_{ij} = [\mathbf{a}_i]^T_{\mathcal{A}} \, [\mathbf{b}_j]_{\mathcal{A}} = \mathbf{a}_i \cdot {}^{\mathcal{A}}\mathbf{C}^{\mathcal{B}} \cdot \mathbf{a}_j \, . \tag{5}$$

We see, therefore, that the DCM is a 3×3, orthogonal matrix, meaning that its inverse is equal to its transpose:

$$\left({}^{\mathcal{A}}C^{\mathcal{B}} \right)^{-1} \triangleq {}^{\mathcal{B}}C^{\mathcal{A}} = \left({}^{\mathcal{A}}C^{\mathcal{B}} \right)^T \, . \tag{6}$$

When $\mathcal{B}$ is a simple rotation away from $\mathcal{A}$, we can thus relate the unit vectors defining the frames as:

$$[\mathbf{a}_i]_{\mathcal{A}} = {}^{\mathcal{B}}C^{\mathcal{A}} \, [\mathbf{b}_i]_{\mathcal{A}} = [\mathbf{b}_i]_{\mathcal{B}} \, , \tag{7}$$

and

$$[\mathbf{b}_i]_{\mathcal{A}} = {}^{\mathcal{A}}C^{\mathcal{B}} \, [\mathbf{b}_i]_{\mathcal{B}} \, . \tag{8}$$

Let the set of matrices $\{C_l(\theta)\}_{l=1}^3$ represent the DCMs associated with a simple rotation about an axis parallel to $\mathbf{a}_l$ (i.e., $\boldsymbol{\lambda} = \mathbf{a}_l$), so that $\mathbf{a}_i = \mathbf{b}_i \, \forall i$ prior to the rotation, but only $\mathbf{a}_l = \mathbf{b}_l$ after

the rotation:

$$C_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad C_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$
$$C_3(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{9}$$

Now let us consider a set of three successive rotations between $\mathcal{A}$ and $\mathcal{B}$, such that a rigid body begins with its body-fixed reference frame aligned with $\mathcal{A}$, is rotated once into an intermediate frame $\mathcal{D} = (O, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$, rotated again into another intermediate frame $\mathcal{F} = (O, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$, and rotated a third time into the final orientation of the body frame $\mathcal{B}$. If all rotations are made about a body-fixed frame axis (i.e., first about one of $\mathbf{a}_i$, then about one of $\mathbf{d}_i$, and then about one of $\mathbf{f}_i$) this is called a Body rotation. If all three rotations are made about an axis in $\mathcal{A}$, this is known as a space rotation. If three distinct axes are used it is a 3-axis rotation, and if only two are used, it is a 2-axis rotation (in this case, the repeating axis cannot be used in two subsequent rotations, or a degree of freedom in the final orientation is lost).

Let the set $S = \{i, j, k\}$ represent this ordered series of rotations for the set of rotation angles $\{\theta_l\}_{l=1}^{3}$, where $i \neq j$ and $j \neq k$. Three axis rotations (Body-3/Space-3) indicate that $i \neq j \neq k$, whereas 2-axis rotations (Body-2/Space-2) indicate that $i = k$. Thus, a Body-2 rotation with $S = \{1, 3, 1\}$ (usually written as Body-2 1-3-1) indicates a rotation of θ_1 about $\mathbf{a}_1$, a rotation of θ_2 about $\mathbf{d}_3$ and a rotation θ_3 about $\mathbf{f}_1$.

The net direction cosine matrix relating $\mathcal{A}$ and $\mathcal{B}$ is therefore:

$$^{\mathcal{A}}C^{\mathcal{B}} = {}^{\mathcal{A}}C^{\mathcal{D}}{}^{\mathcal{D}}C^{\mathcal{F}}{}^{\mathcal{F}}C^{\mathcal{B}}. \tag{10}$$

For Body rotations, each of these intermediate DCMs corresponds directly to one of C_l, so that:

$$ {}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Body}} = C_i(\theta_1)C_j(\theta_2)C_k(\theta_3)\,. \tag{11} $$

For a Space rotation, on the other hand, the intermediate DCMs are about the $\mathcal{A}$ frame axes, and so they can be related to the C_l matrices only by first rotating each one into the proper frame. As demonstrated in Kane et al. [1] (see Eqs. 1.2.12 and 1.2.46), the matrix representation of any tensor with respect to a frame $\mathcal{A}$ can be converted to the equivalent coordinates with respect to frame $\mathcal{B}$ as:

$$ [\mathbf{D}]_{\mathcal{B}} = {}^{\mathcal{B}}C^{\mathcal{A}}\,[\mathbf{D}]_{\mathcal{A}}\,{}^{\mathcal{A}}C^{\mathcal{B}}\,. \tag{12} $$

The first of the intermediate space rotations starts with the body frame aligned with $\mathcal{A}$, so we still have

$$ {}^{\mathcal{A}}C^{\mathcal{D}}_{\text{Space}} = C_i(\theta_1)\,. \tag{13} $$

The second and third rotations, however, must be expressed with respect to frame $\mathcal{A}$ and, by Eq. (12), are thus:

$$ {}^{\mathcal{D}}C^{\mathcal{F}}_{\text{Space}} = C_i(\theta_1)^T C_j(\theta_2)C_i(\theta_1) \tag{14} $$

$$ {}^{\mathcal{F}}C^{\mathcal{B}}_{\text{Space}} = (C_j(\theta_2)C_i(\theta_1))^T C_k(\theta_3)C_j(\theta_2)C_i(\theta_1)\,. \tag{15} $$

Therefore, the Space DCM between $\mathcal{A}$ and $\mathcal{B}$ is

$$ {}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Space}} = C_k(\theta_3)C_j(\theta_2)C_i(\theta_1)\,. \tag{16} $$

As previously mentioned, DCMs encode the set of three rotation angles in a significantly overdetermined form (9 elements). Regardless of the rotation type and specific set of rotation axes, the three angles can be extracted from four elements of the DCM (closely related to the quaternion

encoding of the same three Euler angles). A fifth element is also of interest as it is always in the form of a sine or cosine of θ_2. The useful four elements are sines and cosines of θ_1 or θ_3, multiplied by this fifth term. The remaining four entries in the matrix are sums of products of cosines and sines, and thus more difficult to use in extracting the Euler angles. For example, the DCM for a Body-3 1-2-3 rotation is:

$$ {}^{\mathcal{A}}C^{\mathcal{B}} = \begin{bmatrix} \cos\theta_2\cos\theta_3 & -\cos\theta_2\sin\theta_3 & \sin\theta_2 \\ \cos\theta_3\sin\theta_1\sin\theta_2 + \cos\theta_1\sin\theta_3 & \cos\theta_1\cos\theta_3 - \sin\theta_1\sin\theta_2\sin\theta_3 & -\cos\theta_2\sin\theta_1 \\ -\cos\theta_1\cos\theta_3\sin\theta_2 + \sin\theta_1\sin\theta_3 & \cos\theta_3\sin\theta_1 + \cos\theta_1\sin\theta_2\sin\theta_3 & \cos\theta_1\cos\theta_2 \end{bmatrix} . \quad (17) $$

The five elements of interest are located on the first row and third column, with element (1,3) equal to $\sin\theta_2$ and the other four 'simple' elements all products of $\sin\theta_2$ with $\sin\theta_1$, $\cos\theta_1$, $\sin\theta_3$, and $\cos\theta_3$. Typically, all 24 Space and Body DCMs are written out and separate equations for extracting the Euler angles are written for each one [1]. Here, we describe a simple algorithm for identifying the locations and signs of these five terms for any rotation set, and calculating the corresponding set of Euler angles.

III. Derivation

From the form of the C matrices in Eqs. (11) and (16), we immediately see that the first and third matrix (C_i and C_k) determine the elements of the DCM that will be simple products or linear combinations. For Body rotations, all of the 'simple' terms will be in the ith row and kth column and for Space rotations they will all be in the kth row and ith column. Element (i, k) will always be the only non-product term. For Body-3/Space-3 DCMs this term will have magnitude $\sin\theta_2$, for Body-2/Space-2 it will have magnitude $\cos\theta_2$. The remaining four terms on these rows and columns will all be products of sines and cosines of the other two angles with $\cos\theta_2$ for Body-3/Space-3 rotations and $\sin\theta_2$ for Body-2/Space-2 DCMs.

Since the other angle argument in these four terms corresponds to the order of rotations, we know that for Body rotations the ith row will contain the terms with θ_3 and the kth column will contain the terms with θ_1. For Space rotations, this is transposed so that the ith column contains

the terms with θ_3 and the kth row contains the terms with θ_1.

Finally, the signs of the various terms and specific location of cosines and sines will depend on the particular rotation set. For Body-3/Space-3 rotations, the signs of the DCM terms will always be distributed as in a skew-symmetric matrix (where zeros are treated as positive):

$$J = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}. \tag{18}$$

This is equivalent to saying that for Body-3 rotations, the single $\sin\theta_2$ term will be positive for even (circular) permutation rotation sets and negative for odd permutation rotation sets (this is reversed for Space-3 rotations). The terms containing cosines of θ_1 and θ_3 will always be diagonal elements and the terms containing the sines of θ_1 and θ_3 will always be off-diagonal elements. We can thus tabulate the locations of the elements of interest for Body-3 rotations as follows:

$$\begin{array}{cc} \sin\theta_2 & (i,k) \\ \cos\theta_2\cos\theta_3 & (i,i) \\ \cos\theta_2\sin\theta_3 & (i,j) \\ \cos\theta_2\cos\theta_1 & (k,k) \\ \cos\theta_2\sin\theta_1 & (j,k) \end{array} \tag{19}$$

with the order of indices reversed for the Space-3 case (this is equivalent to using the same elements from the transpose of the DCM).

For Body-2/Space-2 rotations, the single $\cos\theta_2$ term will always be positive, and of the remaining four terms the only negative will be element (2,3) when $j = 1$, element (3,1) when $j = 2$, and element (1,2) when $j = 3$. The ordering of the cosines and sines of θ_1 and θ_3 will circularly permute such

that for Body-2 rotations:

$$\begin{aligned} \cos\theta_2 \quad & (i,i) \\ \sin\theta_2\cos\theta_3 \quad & (i,p) \\ \sin\theta_2\sin\theta_3 \quad & (i,j) \\ \sin\theta_2\cos\theta_1 \quad & (p,i) \\ \sin\theta_2\sin\theta_1 \quad & (j,i) \end{aligned} \tag{20}$$

where p is the element missing from the rotation set (i.e., $p = 3$ for $S = \{1, 2, 1\}$) and can be found by $p = 6 - (i + j)$. Again, the indices are reversed for Space-2 rotations.

For 3-axis rotations this leads to the algorithm:

$$\sin\theta_2 = A_{ik} \tag{21}$$

$$\cos\theta_2 = \sqrt{A_{ii}^2 + A_{ij}^2} \tag{22}$$

$$\theta_1 = \text{atan2}\left(A_{jk}/\cos\theta_2, A_{kk}/\cos\theta_2\right) \tag{23}$$

$$\theta_3 = \text{atan2}\left(A_{ij}/\cos\theta_2, A_{ii}/\cos\theta_2\right) \tag{24}$$

where atan2 is the two argument arctangent function and the matrix A is the DCM multiplied element by element by J (transposed for Space rotations):

$$A_{ij} = \begin{cases} J_{ij} \times \left({}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Body}}\right)_{ij} & \text{Body} \\ J_{ji} \times \left({}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Space}}\right)_{ji} & \text{Space}\,. \end{cases} \tag{25}$$

For 2 axis rotations the algorithm is:

$$\cos\theta_2 = A_{ii} \tag{26}$$

$$\sin\theta_2 = \sqrt{A_{ip}^2 + A_{ij}^2} \tag{27}$$

$$\theta_1 = \text{atan2}\left(A_{ji}/\sin\theta_2, A_{pi}/\sin\theta_2\right) \tag{28}$$

$$\theta_3 = \text{atan2}\left(A_{ij}/\sin\theta_2, A_{ip}/\sin\theta_2\right) \tag{29}$$

where A is equal to ${}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Body}}$ for Body rotations and $\left({}^{\mathcal{A}}C^{\mathcal{B}}_{\text{Space}}\right)^T$ for space rotations, and the sign of one element is flipped, depending on the value j in S:

$$\begin{aligned} A_{23} &= -A_{23} \qquad j = 1 \\ A_{31} &= -A_{31} \qquad j = 2 \\ A_{12} &= -A_{12} \qquad j = 3 \end{aligned} \tag{30}$$

for Body rotations, and the equivalent transposed elements for Space rotations.

IV. Implementation and Conclusions

Listing 1 shows a MATLAB implementation of the unified algorithm. Rather than tabulating 24 separate sets of equations (which can often allow mistakes to creep in), we now have a single piece of code, with only two branching pathways to account for the differences between space and body rotations. Note also that the default matrix data type used in MATLAB allows us to further streamline the code via a mixture of matrix and array operations. This functionality makes it possible to extend this code to calculate all of the Euler angle sets simultaneously, without employing any explicit FOR loops, allowing the user to consistently pick the best set for a given purpose.

Acknowledgments

Portions of this work were performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] T. R. Kane, P. W. Likins, and D. A. Levnison. *Spacecraft Dynamics.* McGraw-Hill Book Co., New York, 1983.

[2] N. J. Kasdin and D. A. Paley. *Engineering Dynamics: A Comprehensive Introduction.* Princeton Univ Press, 2011.

Listing 1 MATLAB implementation of the algorithm.

```
function [t1,t2,t3] = calcEulerAngs(A,rotSet,space)
%Calculate the Euler Angles associated with a specific ordered rotation
%producing the direction cosine matrix A.
%
%INPUT
% A         3x3 Direction Cosine Matrix
% rotSet    1x3 (or 3x1) array of rotation axes to use
% space     logical, true for Space rotation, false for Body
%
%OUTPUT
% t1,t2,t3  Euler angles (in radians)
%
%EXAMPLE
% %body-3 1-2-3 Euler angles of 3x3 DCM matrix dcm:
% [th1,th2,th3] = calcEulerAngs(dcm,[1,2,3],false);
%
% Written 01.13.2012 Dmitry Savransky

%figure out if this is 2 or 3 axis
n = length(unique(rotSet));

%assume Body rotations unless told otherwise
if ~exist('space','var'),space = false;end

ax2neginds = [8,3,4]; %negative elements for 2 axis rotations
i = rotSet(1); j = rotSet(2);

switch n
    case 3
        A = A.*[1 -1 1; 1 1 -1; -1 1 1];
        if space, A = A.';end
        k = rotSet(3);

        c2 = sqrt(A(i,i)^2 + A(i,j)^2);
        t1 = atan2(A(j,k)/c2,A(k,k)/c2);
        t2 = atan2(A(i,k),c2);
        t3 = atan2(A(i,j)/c2,A(i,i)/c2);

    case 2
        A(ax2neginds(j)) = -A(ax2neginds(j));
        if space, A = A.';end
        p = 6 - (i+j); %element missing from rotSet

        s2 = sqrt(A(i,p)^2 + A(i,j)^2);
        t1 = atan2(A(j,i)/s2,A(p,i)/s2);
        t2 = atan2(s2,A(i,i));
        t3 = atan2(A(i,j)/s2,A(i,p)/s2);

    otherwise
        error('calcEulerAngs:inputError','Invalid rotSet.')
end
```