Quaternion to DCM and Back Again

Kurt A. Motekew

April 23, 2014

Abstract

Given the definition of a quaternion reference frame transformation, the conversion from a unit quaternion to a direction cosine matrix and back is derived. The reader is expected to be familiar with basic quaternion properties and algebra. A method of selecting the initial quaternion element to solve for utilizing the quaternion norm condition to guarantee numerical stability is presented and compared to Shepperd's method of inspecting components of the transformation matrix.

1 Background Information

The conversion from a quaternion to a direction cosine matrix (DCM) is straightforward, easy to comprehend, and readily available in literature. The reverse process is, however, more complicated and typically given only a terse description. The derivation presented here is based on the singularity free algorithm originally developed by Shepperd [2] and reviewed by Shuster [3][4] along with hints presented by Stevens and Lewis [5, p. 42].

There are many standards to choose from when representing quaternions. In this text, the notation closely follows the excellent introduction to quaternions and their applications by Kuipers [1]. Quaternions are denoted as:

$$q = q_s + \mathbf{q} = q_s + q_i + q_j + q_k$$

where the scalar is q_s. Only the subset of quaternions associated with rotation sequences are considered, imposing the quaternion norm condition:

$$q_s^2 + q_i^2 + q_j^2 + q_k^2 = 1 \tag{1}$$

Vectors are in bold:

$$\mathbf{r} = \begin{bmatrix} r_i & r_j & r_k \end{bmatrix}^{\mathrm{T}}$$

Reference frame transformations (vs. vector rotations in a fixed coordinate system) are

$$q^* \mathbf{r} q \tag{2}$$

such that q^* is the complex conjugate of the quaternion. Quaternion rotation sequences are carried out from left to right. Other references, such as Shuster [3, pp. 464-467], employ the opposite convention as typically used for matrix algebra while reversing the order of Eqn. (2) .

2 Quaternion to DCM

Consider the reference frame transformation of a position vector $\mathbf{r}$ through the quaternion frame rotation [1, p. 126]:

$$q^* \mathbf{r} q = (2q_s^2 - 1)\mathbf{r} + 2(\mathbf{r} \cdot \mathbf{q})\mathbf{q} + 2q_s(\mathbf{r} \times \mathbf{q}) \tag{3}$$

To derive the transformation from a quaternion to a rotation matrix, begin by expanding Eqn. (3)

$$\begin{aligned} q^* \mathbf{r} q &= (2q_s^2 - 1) \begin{bmatrix} r_i \\ r_j \\ r_k \end{bmatrix} + 2(r_i q_i + r_j q_j + r_k q_k) \begin{bmatrix} q_i \\ q_j \\ q_k \end{bmatrix} + 2q_s \begin{bmatrix} r_j q_k - r_k q_j \\ r_k q_i - r_i q_k \\ r_i q_j - r_j q_i \end{bmatrix} \\ &= \begin{bmatrix} 2r_i q_s^2 - r_i + 2r_i q_i^2 + 2r_j q_i q_j + 2r_k q_i q_k + 2r_j q_s q_k - 2r_k q_s q_j \\ 2r_j q_s^2 - r_j + 2r_i q_i q_j + 2r_j q_j^2 + 2r_k q_j q_k + 2r_k q_s q_i - 2r_i q_s q_k \\ 2r_k q_s^2 - r_k + 2r_i q_i q_k + 2r_j q_j q_k + 2r_k q_k^2 + 2r_i q_s q_j - 2r_j q_s q_i \end{bmatrix} \end{aligned}$$

and factoring:

$$q^* \mathbf{r} q = \begin{bmatrix} r_i \left(2q_s^2 - 1 + 2q_i^2\right) + 2r_j \left(q_i q_j + q_s q_k\right) + 2r_k \left(q_i q_k - q_s q_j\right) \\ 2r_i \left(q_i q_j - q_s q_k\right) + r_j \left(2q_s^2 - 1 + 2q_j^2\right) + 2r_k \left(q_j q_k + q_s q_i\right) \\ 2r_i \left(q_i q_k + q_s q_j\right) + 2r_j \left(q_j q_k - q_s q_i\right) + r_k \left(2q_s^2 - 1 + 2q_k^2\right) \end{bmatrix}$$

The vector $\mathbf{r}$ can now be separated,

$$\begin{aligned} q^* \mathbf{r} q &= \begin{bmatrix} 2q_s^2 - 1 + 2q_i^2 & 2\left(q_i q_j + q_s q_k\right) & 2\left(q_i q_k - q_s q_j\right) \\ 2\left(q_i q_j - q_s q_k\right) & 2q_s^2 - 1 + 2q_j^2 & 2\left(q_j q_k + q_s q_i\right) \\ 2\left(q_i q_k + q_s q_j\right) & 2\left(q_j q_k - q_s q_i\right) & 2q_s^2 - 1 + 2q_k^2 \end{bmatrix} \begin{bmatrix} r_i \\ r_j \\ r_k \end{bmatrix} \\ &= \mathrm{C}\mathbf{r} \end{aligned}$$

Substituting the quaternion norm condition (1) into the diagonal components of C, the DCM derived from the elements of the quaternion q can be rewritten into a form that will be more useful in §3:

$$\mathrm{C} = \begin{bmatrix} q_s^2 + q_i^2 - q_j^2 - q_k^2 & 2\left(q_i q_j + q_s q_k\right) & 2\left(q_i q_k - q_s q_j\right) \\ 2\left(q_i q_j - q_s q_k\right) & q_s^2 - q_i^2 + q_j^2 - q_k^2 & 2\left(q_j q_k + q_s q_i\right) \\ 2\left(q_i q_k + q_s q_j\right) & 2\left(q_j q_k - q_s q_i\right) & q_s^2 - q_i^2 - q_j^2 + q_k^2 \end{bmatrix} \tag{4}$$

A reference frame transformation may now be computed via

$$\mathbf{r_2} = \mathrm{C}\mathbf{r_1}$$

instead of

$$\mathbf{r_2} = q^* \mathbf{r_1} q$$

3 DCM to Quaternion

Extracting the equivalent quaternion from a DCM is not as direct. Keeping in mind the components of Eqn. (4), c_{ij}, are all known values, quaternion elements can be solved for with some care.

$$\mathrm{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

When added together, the diagonal components of C can be used to determine the square of the scalar quaternion element,

$$\begin{aligned} c_{11} + c_{22} + c_{33} &= q_s^2 + q_i^2 - q_j^2 - q_k^2 + q_s^2 - q_i^2 + q_j^2 - q_k^2 + q_s^2 - q_i^2 - q_j^2 + q_k^2 \\ &= 3q_s^2 - q_i^2 - q_j^2 - q_k^2 \\ &= 3q_s^2 - (q_i^2 + q_j^2 + q_k^2) \\ &= 3q_s^2 - (1 - q_s^2) \\ &= 4q_s^2 - 1 \end{aligned}$$

or,

$$\begin{aligned} 4q_s^2 &= 1 + c_{11} + c_{22} + c_{33} \\ &= 1 + \mathrm{Tr}(\mathrm{C}) \end{aligned}$$

where $\mathrm{Tr}(\cdot)$ is the matrix trace operation. The diagonal components can be added and subtracted in eight different combinations. However, only four unique equations result after simplification:

$$4q_s^2 = 1 + c_{11} + c_{22} + c_{33} \tag{5}$$
$$4q_i^2 = 1 + c_{11} - c_{22} - c_{33} \tag{6}$$
$$4q_j^2 = 1 - c_{11} + c_{22} - c_{33} \tag{7}$$
$$4q_k^2 = 1 - c_{11} - c_{22} + c_{33} \tag{8}$$

Isolating quaternion elements with Eqns. (5) through (8) provides only value magnitudes, not signs (e.g., $\sqrt{a^2} = \pm a$). The solution to this problem begins by recalling q performs the same transformation as $-q$. Negating the scalar component changes the direction of the rotation while negating the complex components reverses the direction of the axis of rotation — two quaternions exist for every unique rotation matrix. Because of this, any of Eqns. (5) through (8) may be used to solve for the initial quaternion element with an arbitrarily chosen sign.

Once an initial quaternion element is determined, the remaining, along with appropriate signs, may be solved for using equations formed from the off-diagonal components of (4). Inspection (or trial by error) reveals terms c_{ij} may be combined with c_{ji} $(i \neq j)$ such that each resulting equation contains

only two quaternion elements. A total of six unique equations may be formed using this method:

$$
\begin{aligned}
4q_s q_i &= c_{23} - c_{32} && (9)\\
4q_s q_j &= c_{31} - c_{13} && (10)\\
4q_s q_k &= c_{12} - c_{21} && (11)\\
4q_i q_j &= c_{12} + c_{21} && (12)\\
4q_i q_k &= c_{31} + c_{13} && (13)\\
4q_j q_k &= c_{23} + c_{32} && (14)
\end{aligned}
$$

Given an initial quaternion element, three suitable equations from (9) through (14) are chosen to solve for the remaining unknowns.

Solving for the scalar element first using Eqn. (5) requires Eqns. (9), (10), and (11).

$$
\begin{aligned}
q_s &= +\sqrt{\frac{1 + c_{11} + c_{22} + c_{33}}{4}} && (15)\\
q_i &= \frac{c_{23} - c_{32}}{4q_s}\\
q_j &= \frac{c_{31} - c_{13}}{4q_s}\\
q_k &= \frac{c_{12} - c_{21}}{4q_s}
\end{aligned}
$$

The first complex quaternion element, via Eqn. (6), is used in conjunction with Eqns. (9), (12), and (13).

$$
\begin{aligned}
q_i &= +\sqrt{\frac{1 + c_{11} - c_{22} - c_{33}}{4}} && (16)\\
q_s &= \frac{c_{23} - c_{32}}{4q_i}\\
q_j &= \frac{c_{12} + c_{21}}{4q_i}\\
q_k &= \frac{c_{31} + c_{13}}{4q_i}
\end{aligned}
$$

The solutions beginning with the second

$$
\begin{aligned}
q_j &= +\sqrt{\frac{1 - c_{11} + c_{22} - c_{33}}{4}} && (17)\\
q_s &= \frac{c_{31} - c_{13}}{4q_j}\\
q_i &= \frac{c_{12} + c_{21}}{4q_j}\\
q_k &= \frac{c_{23} + c_{32}}{4q_j}
\end{aligned}
$$

and third

$$\begin{aligned} q_k &= +\sqrt{\frac{1 - c_{11} - c_{22} + c_{33}}{4}} \\ q_s &= \frac{c_{12} - c_{21}}{4q_k} \\ q_i &= \frac{c_{31} + c_{13}}{4q_k} \\ q_j &= \frac{c_{23} + c_{32}}{4q_k} \end{aligned} \tag{18}$$

complex components are similar. Each of these equation sets suffer numerical instability as the first element to be solved approaches zero in magnitude. For example, a rotation angle of π radians results in $q_s = 0$ using equation set (15).

Choosing the largest result from equations (5) through (8) is a direct method of determining which quaternion element is the furthest from zero in magnitude and therefore a good candidate as a divisor in subsequent operations. Shepperd [2] shows an equivalent (yet more efficient) method is to determine the largest of the set

$$\{\mathrm{Tr(C)}, c_{11}, c_{22}, c_{33}\}$$

and choose the quaternion element that would be maximized as the one to solve first (Algorithm 1). Smaller positive diagonal components of C will be

Algorithm 1 Shepperd's Selection Algorithm

if (Tr(C) $> c_{11}$) and (Tr(C) $> c_{22}$) and (Tr(C) $> c_{33}$) **then**
  Use Eqn. Set (15)
else if ($c_{11} > c_{22}$) and ($c_{11} > c_{33}$) **then**
  Use Eqn. Set (16)
else if ($c_{22} > c_{33}$) **then**
  Use Eqn. Set (17)
else
  Use Eqn. Set (18)
end if

subtracted and negative values will ultimately be added when forming the term under the radical with this method.

A technical report written by *id* Software, Inc. [6] simplifies the first set of comparisons by deciding to solve for the scalar term if the trace of the DCM is greater than zero. Inspection of Eqn. (5) reveals the scalar component will then be equal to at least 1/2:

$$4q_s^2 = 1 + 0 \;\rightarrow\; q_s = \frac{1}{2}$$

If the first test fails, Shepperd's selection method is resumed.

An alternative method introduced here involves recognizing the quaternion norm condition (1) requires at least one quaternion element to be 1/2 or greater in magnitude [4, p. 547]. Therefore, a quaternion element $q_\star$,

$$q_\star \geq \frac{1}{2}$$

results in,

$$4q_\star^2 \geq 1$$

This allows the full quaternion to be computed once the first (right hand side) of equations (5) through (8) evaluates to a value κ such that

$$\epsilon < \kappa \leq 1$$

where ϵ is sufficiently large to not cause numerical instability. Algorithm 2 demonstrates making use of a language such as C or Java where variable assignment while evaluating logical expressions is permitted (if $\kappa = 1$, then the $\geq$ comparisons should be used instead of $>$).

Algorithm 2 Norm Constraint Selection Algorithm

$\kappa = 0.25$
if $(tmp = 1 + c_{11} + c_{22} + c_{33}) > \kappa$ **then**
  $q_s = \sqrt{tmp/4}$
  Complete q with remainder of Eqn. Set (15)
else if $(tmp = 1 + c_{11} - c_{22} - c_{33}) > \kappa$ **then**
  $q_i = \sqrt{tmp/4}$
  Complete q with remainder of Eqn. Set (16)
else if $(tmp = 1 - c_{11} + c_{22} - c_{33}) > \kappa$ **then**
  $q_j = \sqrt{tmp/4}$
  Complete q with remainder of Eqn. Set (17)
else if $(tmp = 1 - c_{11} - c_{22} + c_{33}) > \kappa$ **then**
  $q_k = \sqrt{tmp/4}$
  Complete q with remainder of Eqn. Set (18)
else
  Throw an error!
end if

The first comparison in Algorithm 2 is on par with Algorithm 1 — although the decision process has one additional operation, there are fewer logical evaluations along with simplified code. Nothing is lost if the first comparison fails and the second passes, although the remaining comparisons are computationally more expensive than those of Algorithm 1. Table 1 compares the frequency with which Eqn. sets (15) through (18) are selected when using Algorithm 1 vs. Alg. 2 with two reasonable values of κ. Choosing $\kappa = 1$, a conservative value, results in approximately 84% (67% + 17%) of all conversions occurring within the first two comparisons. If κ is lowered to 1/4, requiring the divisor quaternion element to be at least 1/4 in magnitude, then 97% (84% + 13%) of the

Selected Divisor	Alg. 1	Alg. 2	Alg. 2
		$\kappa = 1$	$\kappa = 1/4$
Test 1 $\rightarrow q_s$	58%	67%	84%
Test 2 $\rightarrow q_i$	14%	17%	13%
Test 3 $\rightarrow q_j$	14%	11%	3%
Test 4 $\rightarrow q_k$	14%	5%	<1%

Table 1: *Execution trends for equation sets (15) through (18) are shown for selection Algorithms 1 and 2. Statistics were accumulated by creating two million randomly generated rotation matrices and tracking which of the quaternion elements would be selected as the divisor by both algorithms. The rotation matrices were produced by first creating a quaternion from an angle and rotation axis and then converting the quaternion to a DCM (§2). A uniform distribution,* $U(-1, 1)$*, was used to create each element of the rotation axis before normalization. Half of the rotation angles were created using* $U(-\pi, \pi)$ *while the other half used* $U(0, 2\pi)$*.*

conversions will pass either the first or second of Algorithm 2 tests. Note that for simulations not involving large rotations (such as attitude measured relative to some form of local level reference frame), the first test will always pass given the total angle of rotation must exceed 120° even when $\kappa = 1$.

Concluding Remarks

Both the derivation and implementation of the quaternion to DCM conversion are simple processes. The quaternion reference frame operation directly leads to a single formula for each DCM component given quaternion elements. The formulas relating DCM components to quaternion elements provide more equations than unknowns for the reverse procedure. Proper selection of these equations eliminates numerical instability when converting a DCM to a quaternion given a proper transformation matrix. Programming languages allowing variable assignment while evaluating logical expressions may use the quaternion norm constraint to simplify coding of this selection process without impacting numerical stability or performance.

References

[1] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, Princeton University Press, Princeton, NJ, 1999.

[2] S. W. Shepperd, *Quaternion from Rotation Matrix*, Journal of Guidance and Control, 1 (1978), pp. 223–224.

[3] M. D. Shuster, *A Survey of Attitude Representations*, The Journal of the Astronautical Sciences, 41 (1993), pp. 439–517.

[4] M. D. Shuster and G. A. Natanson, *Quaternion Computation from a Geometric Point of View*, The Journal of the Astronautical Sciences, 41 (1993), pp. 545–556.

[5] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, Inc., New York, 1992.

[6] J. M. P. van Waveren, From Quaternion to Matrix and Back, tech. rep., Id Software, Inc., 2005.