# Quaternion Math

## Application Note

## Abstract

This application note provides an overview of the quaternion attitude representation used by VectorNav products and how to convert it into other common attitude representations.

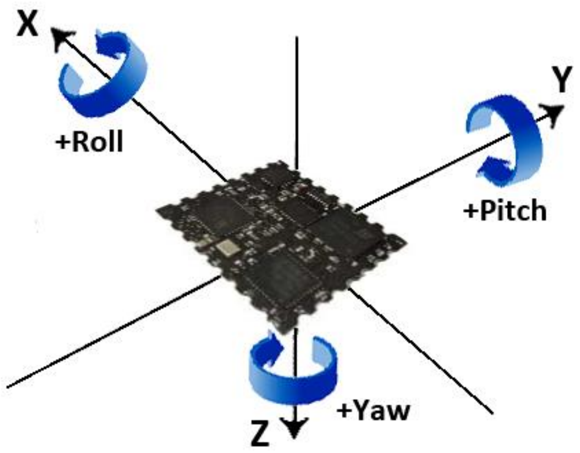
## Document Information

| | |
|---|---|
| Title | Quaternion Math |
| Subtitle | Quaternion based attitude representation |
| Document Type | Application Note |
| Document Number | AN002 |
| Document Status | Released |

## 1 What is a quaternion?

The quaternion is an abstract means of representing attitude. It is a four-dimensional vector used to describe a three-dimensional attitude representation. In order to understand what a quaternion is and why it is useful you first need to be aware of the alternative means of attitude representation. Hopefully you are familiar with the Euler angle representations of attitude, of which one is the 3-2-1 rotation sequence most commonly known as yaw (or heading), pitch, and roll. These three angles work great for most applications. Since the Euler angles are a three-dimensional vector that represents a three-dimensional attitude it is easy to just look at the numbers and get a feel for what they intuitively mean.

Figure 1 - Example of a 3-2-1 Euler angle set



There is one problem with the Euler angle attitude representation; there exists two attitudes where you have a singularity in the solution. Take a look at Figure 1 and imagine that the pitch angle is equal to 90 degrees. In this case the yaw and roll perform the same operation. This may not be a problem in computer graphics applications such as in the rendering of the orientation of an object since every attitude does still have a representation, however for applications that require the controls this can be detrimental because you can run into severe math problems when dealing with angles that are close to these singularity points. One way to get around this problem is to add an additional degree so that we have an over defined attitude representation. This is what the quaternion does in a very simplistic sense. The quaternion is based upon the principal of Euler's principal rotation.

**Euler's Principal Rotation:**

*A rigid body or coordinate reference frame can be brought from an arbitrary initial orientation to an arbitrary final orientation by a single rigid body rotation through a principal angle Φ about the principal axis **ê**; the principal axis is a judicious axis fixed in both initial and final orientation.*

What this means is that you can represent any arbitrary orientation with just a unit vector and an angle. The unit vector defines the direction of rotation, and the angle is the amount that you rotate about this axis to go from your initial to your final attitude. This works for any rotation. The quaternion is based upon this principal and can be derived from the principal axis and principal angle.

## 2 Quaternion used by VectorNav

While different organizations use different ordering of the terms, all quaternions fundamentally represent the same thing. Here are the equations for each of the terms used in our quaternion.

$$q_0 = e_x \sin\left(\frac{\vartheta}{2}\right)$$

$$q_1 = e_y \sin\left(\frac{\vartheta}{2}\right)$$

$$q_2 = e_z \sin\left(\frac{\vartheta}{2}\right)$$

$$q_3 = \cos\left(\frac{\vartheta}{2}\right)$$

Where $e = \begin{Bmatrix} e_x \\ e_y \\ e_z \end{Bmatrix}$ is the principal axis and $\vartheta$ is the principal angle.

The fourth term $q_3$ is known as the scalar term. The attitude (Yaw = 0, Pitch = 0, Roll = 0) is represented in quaternion form as q = [0, 0, 0, 1]. Here is how you would go about converting the quaternion into an Euler 321 sequence (yaw, pitch, roll).

$$yaw = \tan^{-1}\left(\frac{2(q_0q_1 + q_3q_2)}{{q_3}^2 - {q_2}^2 - {q_1}^2 + {q_0}^2}\right)$$

$$pitch = \sin^{-1}\big(-2(q_0q_2 - q_1q_3)\big)$$

$$roll = \tan^{-1}\left(\frac{2(q_1q_2 + q_0q_3)}{{q_3}^2 + {q_2}^2 - {q_1}^2 - {q_0}^2}\right)$$

## 3 Quaternion to Direction Cosine Matrix

If you want to convert the quaternion to a directional cosine matrix use the following formula:

$$[C] = \begin{bmatrix} {q_3}^2 + {q_0}^2 - {q_1}^2 - {q_2}^2 & 2(q_0q_1 + q_3q_2) & 2(q_0q_2 - q_3q_1) \\ 2(q_0q_1 - q_3q_2) & {q_3}^2 - {q_0}^2 + {q_1}^2 - {q_2}^2 & 2(q_1q_2 + q_3q_0) \\ 2(q_0q_2 + q_3q_1) & 2(q_1q_2 - q_3q_0) & {q_3}^2 - {q_0}^2 - {q_1}^2 + {q_2}^2 \end{bmatrix}$$

## 4 Directional Cosine Matrix to Quaternion

To convert from a directional cosine matrix to a quaternion use the following formulas:

$$q_3 = 0.5 * \sqrt{C_{00} + C_{11} + C_{22} + 1}$$

$$q_0 = \frac{C_{12} - C_{21}}{4q_3}$$

$$q_1 = \frac{C_{20} - C_{02}}{4q_3}$$

$$q_2 = \frac{C_{01} - C_{10}}{4q_3}$$

## 5 Kinematic differential equations

If you need the time rate of change of the quaternion then just set the VN-100 to output quaternion and angular rates and use the following formulas:

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2}\begin{bmatrix} q_0 & q_3 & -q_2 & q_1 \\ q_1 & q_2 & q_3 & -q_0 \\ q_2 & -q_1 & q_0 & q_3 \\ q_3 & -q_0 & -q_1 & -q_2 \end{bmatrix}\begin{pmatrix} 0 \\ \omega_0 \\ \omega_1 \\ \omega_2 \end{pmatrix}$$