**Proceedings of**
**A Symposium Commemorating the**
**Legacy, Works, and Life of**
***Sir Robert Stawell Ball***
**Upon the 100th Anniversary of**
***A Treatise on the Theory of Screws***
**July 9-11, 2000**
**University of Cambridge, Trinity College**

# Orientation representations used within ABB robot controllers.

John-Erik Snell
ABB Robotics
SE-721 68
Västerås
Sweden

## Abstract.

Programming and storage of robot Quaternion representation is used for specifying orientation information with the ABB robot controllers. Arguments for making this selection are presented. It is also shown that it may be very inefficient to use the same representation through out the system. Efficiency makes it necessary to use different representations internally.

**Key words:** Robot, kinematics, quaternion, jacobian

## Introduction.

In 1975 ABB launched its first generation robot controller, S1. Programming and storage of robot positions was done in joint coordinates. Execution of a robot path was performed through joint interpolation between programmed joint positions.

It was very soon realized that the effectiveness of application programming would very much increase if robot motion could be defined along straight lines and circular arcs. How to describe the orientation of the robot and how to perform reorientation between robot positions in cartesian mode were some of the major decisions that had to be made for a new generation of controllers.

Orientation information for a robot application is used in different ways within the controller. The most obvious one is in the user interface where the user of the robot system has to specify the orientation of each programmed position and of all user defined frames. Internally all frame operations need to operate on it, cartesian interpolation requires techniques for reorientation along paths, direct and inverse kinematics must adapt their algorithms to the selected orientation representation.

## Choice of orientation representation.

Initially the main alternatives for orientation representation were Euler angles and rotation matrices, until a paper was found that gave a full answer to all the issues that had to be solved. It was a Research Report from IBM "Planning and Execution of Straight-line Manipulator Trajectories" by Russel H Taylor. [1]

The report presented the use of quaternion representation to specify the orientation of the robot tool. It also showed that quaternions provided a convenient and efficient way of orientation transition between programmed positions. The algorithm for single segment interpolation from orientation $R_0$ to orientation $R_1$ was defined in the following way

The orientation along a linear cartesian path is given by the relation
$R(L) = R_1 * (\cos(L \cdot \theta / 2) + \overline{n} \cdot \sin(L \cdot \theta / 2))$
L represents the fraction of the segment to traverse. (L=1 in the start point and 0 in the end point)
* represents quaternion multiplication

The two major factors for the benefit of the quaternion representation over the rotation matrix representation were the lower memory requirement for the storage of orientation and the efficiency in the path interpolation.

The next issue was what representation to use internally in the controller for the direct and inverse kinematics. Literature references and contacts with universities showed that traditional rotation matrix notation for orientation was the most convinient way. For the direct kinematics the internally computed rotation matrix was converted into quaternion representation at the output. For the inverse kinematics the reverse was done. Closed form solutions for the inverse kinematics using basic trigonometry could be found for ABB robots when the quaternion notation was introduced in the S2 controller in 1982.

Some time later it was realized that there was some problem with the conversion from rotation matrix notation to quaternion notation in some situations. This became especially clear when offline programming was used more frequently.

The algorithm used for the conversion from the rotation matrix T

$$T=\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

to a quaternion Q = $(q_{0,}q_1, q_2, q_3)$ was initially given by

$$q_0 = {}^{+}\!/_{-}\sqrt{(Tr(T)+1)}/2$$
$$q_1 = {}^{+}\!/_{-}\sqrt{(1+2\cdot t_{11} - Tr(T))}/2$$
$$q_2 = {}^{+}\!/_{-}\sqrt{(1+2\cdot t_{22} - Tr(T))}/2$$
$$q_3 = {}^{+}\!/_{-}\sqrt{(1+2\cdot t_{33} - Tr(T))}/2$$

where Tr represents the trace operator.

The signs of $q_1, q_2, q_3$ are given by the three equations

$$q_1 = sign(t_{32} - t_{23})$$
$$q_2 = sign(t_{13} - t_{31})$$
$$q_3 = sign(t_{21} - t_{12})$$

It was realized that when the rotation matrix became close to symmetric the sign information became uncertain. That occured when a rotation close to +/- $\pi$ was performed,. In these cases an alternative algorithm had to be supplied[2].
First from the equations

$$q_1^{\,2} = (t_{11} + 1)/2$$
$$q_2^{\,2} = (t_{22} + 1)/2$$
$$q_3^{\,2} = (t_{33} + 1)/2$$

the largest of the quantities $q_1, q_2, q_3$ is selected.
This is then used in the equations

$$t_{21} + t_{12} = 4 \cdot q_1 \cdot q_2$$
$$t_{31} + t_{13} = 4 \cdot q_1 \cdot q_3$$
$$t_{32} + t_{23} = 4 \cdot q_2 \cdot q_3$$

for determining the remaining two quantities.
This solved the uncertanty that had been observed.

The next opportunity to change orientation representation came in the end of the 80´s when the development of the S4 controller started.

The S4 generation became a completely new platform for future robot control. It was implemented in high level language using floating point hardware. Orientation representation became once again an issue. The same arguments for the quaternion representation were still valid. Now,we had also many years of experience from using this representation and this was mainly positive. However, there was one major disadvantage and this was the fact that it was very difficult for the average programmer to visualize the orientation from the 4 quaternion parameters $q_{0,}q_1, q_2, q_3$.

It was once again decided to keep to the quaternion representation in the external interface (teachpendant presentation and the RAPID programme) but also to supply the controller with tools that allowed the user to convert between quaternion and Euler angle notation

A second issue was what representation to use in the internal computations. The increasing number of robot variants that appeared and the need for a full kinematic error model description for robot absolute accuracy made it necessary to generalize the internal kinematic modelling. This required use of iterative methods for the inverse kinematics and made it necessary to more deeply analyse what orientation representation would be the best one to use. The orientation representation would effect the structure of the kinematic equations and might also influence the choice of the numerical methods that had to be used and the convergence properties of these.
Prof Jorge Angeles presented in [3] an analysis of two different representations (Linear Invariants and Quaternions). In a subsequent paper from 1992 [4] a further investigation was done where a third representation was introduced, natural invariants.

We implementated and tested the three methods before a final decision was made.

**Linear invariants**

The prescribed rotation matrix representation Q is converted into a linear set

$$q_0 = (tr(Q) - 1)/2 = \cos(\phi)$$

$\overline{q} = vect(Q) = \overline{n} \cdot \sin(\phi)$
where $\overline{n}$ = the rotation axis of the rotation defined by the rotation matrix Q(eigenvector of Q with eigenvalue 1)
$\phi$ = the rotation angle about the axis of rotation

The orientation of the kinematic equation of the manipiulator is formulated as a product of six rotation matrices one for each axis
$Q = Q_1 \cdot Q_2 \cdot Q_3 \cdot Q_4 \cdot Q_5 \cdot Q_6$
The kinematic system of equations can now be formulated as

$$f(\overline{\theta}) = \begin{bmatrix} 2 * \left[ vect(Q) - vect(Q_p) \right] \\ tr(Q) - tr(Q_p) \\ \overline{p} - \overline{p}_p \end{bmatrix}$$

where $p_p$ and $Q_p$ are the prescribed position and orientation resp.

$f(\overline{\theta})$ has seven  components but only six degrees of freedom for a six axes robot. The over-
determination is however only formal, because the first four components that describe the orientation
are dependent. For the solution of the system of equations numerical metods that can solve over- determined systems of equations must be used.

The Jacobian matrix $J(\overline{\theta})$of$f(\overline{\theta})$can be expressed as the product  of two parts H and K.

Where $K$ is the velocity Jacobian according to Whitney [5]

$$H = \begin{bmatrix} \overline{1} \cdot \mathrm{tr(Q)} - \mathrm{Q} & \overline{0}_3 \\ -2 \cdot vect(Q)^T & 0 \\ \overline{0}_3 & \overline{1}_3 \end{bmatrix}$$

The major disadvantage with the linear invariants is that the matrix H introduces an algebraic singularity in the representation. This occures when the angle of rotation $\phi$ is +/-$\pi$

**Quaternion representation**

Using the quaternion representation the kinematic equation gets the form

$$f(\overline{\theta}) = \begin{bmatrix} 2 * \left[ vect(\sqrt{Q}) - vect(\sqrt{Q_p}\right] \\ tr(\sqrt{Q}) - tr(\sqrt{Q_p}) \\ \overline{p} - \overline{p}_p \end{bmatrix}$$

The Jacobian gets the form

$J^{'} = H^{'} K$

where $K$ is the Jacobian according to Whitney

$$H' = \begin{bmatrix} \overline{1} \cdot \mathrm{tr}(\sqrt{\mathrm{Q}}) - \sqrt{\mathrm{Q}} & \overline{0}_3 \\ -2 \cdot vect(\sqrt{Q})^T & 0 \\ \overline{0}_3 & \overline{1}_3 \end{bmatrix}$$

This representation has the advantage that the algebraic singularities are removed, however at the expence of more complex computations.

**Natural invariants**

The kinematic equation gets the form

$$f(\overline{\theta}) = \begin{bmatrix} \sin(\Delta\phi) \cdot \overline{n} \\ \overline{p} - \overline{p}_p \end{bmatrix}$$

$\Delta\phi$ and $\overline{n}$ represent the rotation between current and prescribed orientation

The Jacobian matrix gets the form
$J^{"} = K$

The three techniques were studied through analysis and simulations. Two factors  are critical for a working implementation,
- operation count
- number of iterations needed for convergance.

For all two factors the natural invariant formulation gave the best performance. The reason for this  were
- the Jacobian matrix is square which lets you use less expensive numerical methods.
- no additional operations are needed to premultiply with the H matrix to determine the Jacobian matrix.

Although the quaternion representation gave a mathematically elegant way of handling all aspects from definition of orientation to formulating ,computing the Jacobian and solution of the inverse kinematic problem, it showed to be the least effcient method as to the convergence properties of the three methods investigated.

This led us to the descision that the natural invariant formulation in the inverse kinematics was the one to implement for robots that could not be solved with close form technique.

## **Summary.**

When selecting a representation for orientation it must be considered that orientation information is used in different situations within the controller. Using the same notation throughout all situations may result in inefficient overall use of computing resources. In the worst situation problems related only to selection of representation may be introduced. We think that by internally converting between different representations we have made use of the benifits of different representations.

**References:**

1. Russel H. Taylor, Planning and Execution of Straight-line Manipulator Trajectories, IBM J. Res Develop. Vol. 23 No 4 July 1979
2. O. Friberg, Computation of Euler Parameters from Multipoint Data, Trans of the ASME Vol 110 June 1988
3. Jorge Angeles, Die theoretischen Grundlagen zur Behandlung algebraicher Singularitäten der kinematischen Koordinatedumkehr in der Robotertechnik, Mechanisms & Machine Theory, March 1990
4. Murat Tandirch, Jorge Angeles, John Darovich, On Rotation Representations in Computational Robot Kinematics, Int. J Robotics & Intelligent Systems, Nov 1992
5. Dennis E Whitney, The mathematics of coordinated control of prosthetic arms and manipulators, ASME J. Dyn. Sys. Meas. Contr,Vol 94, No 14