Unit Quaternions: A Mathematical Tool for Modeling, Path Planning and Control of Robot Manipulators

Ricardo Campa and Karla Camarillo
Instituto Tecnológico de la Laguna
Mexico

1. Introduction

Robot manipulators are thought of as a set of one or more kinematic chains, composed by rigid bodies (links) and articulated joints, required to provide a desired motion to the manipulator's end–effector. But even though this motion is driven by control signals applied directly to the joint actuators, the desired task is usually specified in terms of the pose (i.e., the position and orientation) of the end–effector. This leads to consider two ways of describing the configuration of the manipulator, at any time (Rooney & Tanev, 2003): via a set of joint variables or pose variables. We call these configuration spaces the *joint space* and the *pose space*, respectively.
But independently of the configuration space employed, the following three aspects are of interest when designing and working with robot manipulators:

- Modeling: The knowledge of all the physical parameters of the robot, and the relations among them. Mathematical (kinematic and dynamic) models should be extracted from the physical laws ruling the robot's motion. Kinematics is important, since it relates joint and pose coordinates, or their time derivatives. Dynamics, on the other hand, takes into account the masses and forces that produce a given motion.
- Task planning: The process of specifying the different tasks for the robot, either in pose or joint coordinates. This may involve from the design and application of simple time trajectories along precomputed paths (this is called *trajectory planning*), to complex computational algorithms taking real–time decisions during the execution of a task.
- Control: The elements that allow to ensure the accomplishment of the specified tasks in spite of perturbances or unmodeled dynamics. According to the type of variables used in the control loop, we can have joint space controllers or pose space controllers. Robot control systems can be implemented either at a low level (e.g. electronic controllers in the servo–motor drives) or via sophisticated high–level programs in a computer.

Fig. 1 shows how these aspects are related to conform a robot motion control system. By *motion control* we refer to the control of a robotic mechanism which is intended to track a desired time–varying trajectory, without taking into account the constraints given by the environment, i.e., as if moving in free space. In such a case the desired task (a time function along a desired path) is generated by a trajectory planner, either in joint or pose variables. The motion controller can thus be designed either in joint or pose space, respectively.

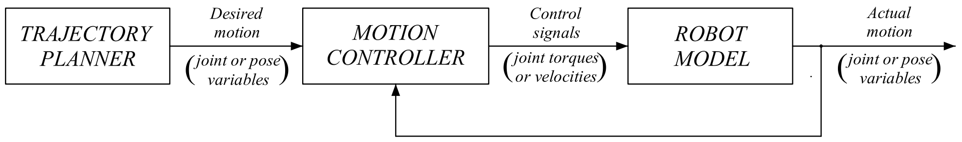

Figure 1. General scheme of a robot motion control system

Most of industrial robot manipulators are driven by *brushless DC* (*BLDC*) servoactuators. A BLDC servo system is composed by a permanent–magnet synchronous motor, and an electronic drive which produces the required power signals (Krause et al., 2002). An important feature of BLDC drives is that they contain several nested control loops, and they can be configured so as to accept as input for their inner controllers either the motor's desired position, velocity or torque. According to this, a typical robot motion controller (such as the one shown in Fig. 1) which is expected to include a position tracking control loop, should provide either velocity or torque reference signals to each of the joint actuators' inner loops. Thus we have kinematic and dynamic motion controllers, respectively.
On the other hand, it is a well–known fact that the number of degrees of freedom required to completely define the pose of a rigid body in free space is six, being three for position and three for orientation. In order to do so, we need to attach a coordinate frame to the body and then set the relations between this moving frame and a fixed one (see Fig. 2). Position is well described by a position vector $p \in \mathbb{R}^3$, but in the case of orientation there is not a generalized method to describe it, and this is mainly due to the fact that orientation constitutes a three–dimensional manifold, M^3, which is not a vector space, but a *Lie group*.

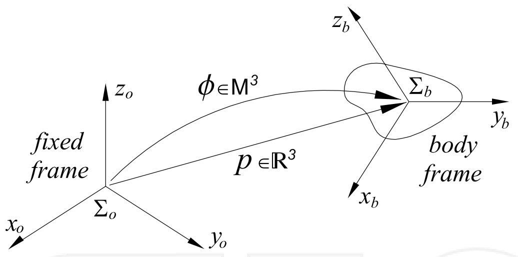

Figure 2. Position and orientation of a rigid body

Minimal representations of orientation are defined by three parameters, e.g., Euler angles. But in spite of their popularity, Euler angles suffer the drawbacks of representation singularities and inconsistency with the task geometry (Natale, 2003). There are other nonminimal parameterizations of orientation which use a set of $3+k$ parameters, related by k holonomic constraints, in order to keep the required three degrees of freedom. Common examples are the rotation matrices, the angle–axis pair and the Euler parameters. For a list of these and other nonminimal parameterizations of orientation see, e.g., (Spring, 1986).
This work focuses on Euler parameters for describing orientation. They are a set of four parameters with a unit norm constraint, so that they can be considered as points lying on the surface of the unit hypersphere $\mathrm{S}^3 \subset \mathbb{R}^4$. Therefore, Euler parameters are *unit quaternions*, a subset of the quaternion numbers, first described by Sir W. R. Hamilton in 1843, as an extension to complex numbers (Hamilton, 1844).

The use of Euler parameters in robotics has increased in the latter years. They are an alternative to rotation matrices for defining the kinematic relations among the robot's joint variables and the end–effector's pose. The advantage of using unit quaternions over rotation matrices, however, appears not only in the computational aspect (e.g., reduction of floating–point operations, and thus, of processing time) but also in the definition of a proper orientation error for control purposes.
The term *task space control* (Natale, 2003) has been coined for referring to the case of pose control when the orientation is described by unit quaternions, in opposition to the traditional *operational space control*, which employs Euler angles (Khatib, 1987). Several works have been done using this approach (see, e.g., Lin, 1995; Caccavale et al., 1999; Natale, 2003; Xian et al., 2004; Campa et al., 2006).
The aim of this work is to show how unit quaternions can be applied for modeling, path planning and control of robot manipulators. After recalling the main properties of quaternion algebra in Section 2, we review some quaternion–based algorithms that allow to obtain the kinematics and dynamics of serial manipulators (Section 3). Section 4 deals with path planning, and describes two algorithms for generating curves in the orientation manifold. Section 5 is devoted to task–space control. It starts with the specification of the pose error and the control objective, when orientation is given by unit quaternions. Then, two common pose controllers (the *resolved motion rate controller* and the *resolved acceleration controller*) are analyzed. Even though these controllers have been amply studied in literature, the Lyapunov stability analysis presented here is novel since it uses a quaternion–based space–state approach. Concluding remarks are given in Section 6.

Throughout this paper we use the notation $\lambda_m\{A\}$ and $\lambda_M\{A\}$ to indicate the smallest and largest eigenvalues, respectively, of a symmetric positive definite matrix $A(x)$ for all $x \in \mathbb{R}^n$. The Euclidean norm of vector x is defined as $\|x\| = \sqrt{x^T x}$. Also, for a given matrix A, the spectral norm $\| A \|$ is defined as $\| A \| = \sqrt{\lambda_M\{A^T A\}}$.

2. Euler Parameters

For the purpose of this work, an n–dimensional manifold $\mathrm{M}^n \subseteq \mathbb{R}^m$, with $n \le m$, can be easily understood as a subset of the Euclidean space $\mathbb{R}^m$ containing all the points whose coordinates $x_1, x_2, \ldots, x_m$ satisfy $r = m - n$ holonomic constraint equations of the form $\gamma_i(x_1, x_2, \ldots, x_m) = 0$ ($i = 1, 2, \ldots, r$). As a typical example of an n–dimensional manifold we have the generalized unit (hyper)sphere $\mathrm{S}^n \subset \mathbb{R}^{n+1}$, defined as

$$\mathrm{S}^n = \{x \in \mathbb{R}^{n+1} : \| x \| = 1\}.$$

It is well–known that the configuration space of a rigid–body's orientation is a tree–dimensional manifold, $\mathrm{M}^3 \subset \mathbb{R}^m$, where m is the number of parameters employed to describe the orientation. Thus we have minimal ($m = 3$) and non–minimal ($m > 3$) parameterizations of the orientation, being the most common the following:

1. Euler angles, $\mathrm{M}^3 \equiv \mathbb{R}^3$.
2. Rotation matrices, $\mathrm{M}^3 \equiv \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3} (\equiv \mathbb{R}^9)$.

3. Angle/axis pair, $M^3 \equiv \mathbb{R} \times S^2 \subset \mathbb{R}^4$.
4. Euler parameters, $M^3 \equiv S^3 \subset \mathbb{R}^4$.

Given the fixed and body coordinate frames (Fig. 2), Euler angles indicate the sequence of rotations around one of the frame's axes required to make them coincide with those of the other frame. There are several conventions of Euler angles (depending of the sequence of axes to rotate), but all of them have inherently the problem of singular points, i.e., orientations for which the set of Euler angles is not uniquely defined (Kuipers, 1999).

Euler also showed that the relative orientation between two frames with a common origin is equivalent to a single rotation of a minimal angle around an axis passing through the origin. This is known as the Euler's rotation theorem, and implies that other way of describing orientation is by specifying the minimal angle $\theta \in \mathbb{R}$ and a unit vector $u \in S^2 \subset \mathbb{R}^3$ indicating the axis of rotation. This parameterization, known as the *angle/axis pair*, is non–minimal since we require four parameters and one holonomic constraint, i.e., the unit norm condition of u.

The angle/axis parameterization, however, has some drawbacks that make difficult its application. First, it is easy to see that the pairs $\begin{bmatrix} \theta & u^T \end{bmatrix}^T$ and $\begin{bmatrix} -\theta & -u^T \end{bmatrix}^T \in \mathbb{R} \times S^2$ both give the same orientation (in other words, $\mathbb{R} \times S^2$ is a *double cover* of the orientation manifold). But the main problem is that the angle/axis pair has a singularity when $\theta = 0$, since in such case u is not uniquely defined.

A most proper way of describing the orientation manifold is by means of an orthogonal rotation matrix $R \in \mathrm{SO}(3)$, where the *special orthogonal group*, defined as

$$\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} : R^T R = I, \det(R) = 1\}$$

is a 3–dimensional manifold requiring nine parameters, that is, $M^3 \equiv \mathrm{SO}(3) \subset \mathbb{R}^{3\times 3} \equiv \mathbb{R}^9$ (it is easy to show that the matrix equation $R^T R = I$ is in fact a set of six holonomic constraints, and $\det(R) = 1$ is one of the two possible solutions). Furthermore, being SO(3) a multiplicative matrix group, it means that the composition of two rotation matrices is also a rotation matrix, or

$$R_1, R_2 \in \mathrm{SO}(3) \Rightarrow R_1 R_2 \in \mathrm{SO}(3).$$

A group which is also a manifold is called a *Lie group*.

Rotation matrices are perhaps the most extended method for describing orientation, mainly due to the convenience of matrix algebra operations, and because they do not have the problem of singular points, but rotation matrices are rarely used in control tasks due to the difficulty of extracting from them a suitable vector orientation error.

Quaternion numbers (whose set is named $\mathbb{H}$ after Hamilton) are considered the immediate extension to complex numbers. In the same way that complex numbers can be seen as arrays of two real numbers (i.e. $\mathbb{C} \equiv \mathbb{R}^2$) with all basic arithmetic operations defined, quaternions become arrays of four real numbers ($\mathbb{H} \equiv \mathbb{R}^4$). It is useful however to group the last three components of a quaternion as a vector in $\mathbb{R}^3$. Thus $\begin{bmatrix} a & b & c & d \end{bmatrix}^T \in \mathbb{H}$, and $\begin{bmatrix} a & v^T \end{bmatrix}^T \in \mathbb{H}$, with $v \in \mathbb{R}^3$, are both valid representations of quaternions (a is called the scalar part, and v

the vector part, of the quaternion). Given two quaternions $z_1, z_2 \in \mathbb{H}$, where $z_1 = \begin{bmatrix} a_1 & v_1^T \end{bmatrix}^T$ and $z_2 = \begin{bmatrix} a_2 & v_2^T \end{bmatrix}^T$, the following quaternion operations are defined:

- Conjugation:

$$z_1^* = \begin{bmatrix} a_1 \\ -v_1 \end{bmatrix} \in \mathbb{H}$$

- Inversion:

$$z_1^{-1} = \frac{z_1^*}{\| z_1 \|^2} \in \mathbb{H}$$

where the quaternion Euclidean norm is defined as usual $\| z_1 \| = \sqrt{z_1^T z_1} = \sqrt{a_1^2 + v_1^T v_1}$.

- Addition/subtraction:

$$z_1 \pm z_2 = \begin{bmatrix} a_1 \pm a_2 \\ v_1 \pm v_2 \end{bmatrix} \in \mathbb{H}$$

- Multiplication:

$$z_1 \otimes z_2 = \begin{bmatrix} a_1 a_2 - v_1^T v_2 \\ a_1 v_2 + a_2 v_1 + S(v_1) v_2 \end{bmatrix} \in \mathbb{H} \tag{1}$$

where $S(\cdot)$ is a matrix operator such that, for all $x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \in \mathbb{R}^3$ produces a skew–symmetric matrix, given by:

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}.$$

Skew–symmetric matrices have several useful properties, such as the following (for all $x, y, z \in \mathbb{R}^3$):

$$S(x)^T = S(-x) = -S(x), \tag{2}$$

$$S(x)(y+z) = S(x)y + S(x)z, \tag{3}$$

$$S(x)y = -S(y)x, \tag{4}$$

$$S(x)x = 0, \tag{5}$$

$$y^T S(x) y = 0, \tag{6}$$

$$x^T S(y) z = z^T S(x) y, \tag{7}$$

$$S(x)S(y) = yx^T - x^T y I. \tag{8}$$

Note that, by (4), quaternion multiplication is not commutative, that is, $z_1 \otimes z_2 \neq z_2 \otimes z_1$.

- Division: Due to the non–commutativity of the quaternion multiplication, dividing z_1 by z_2 can be either

$$z_1 \otimes z_2^{-1} = \frac{z_1 \otimes z_2^*}{\| z_2 \|^2}, \text{ or } z_2^{-1} \otimes z_1 = \frac{z_2^* \otimes z_1}{\| z_2 \|^2}.$$

Euler parameters are another way of describing the orientation of a rigid body. They are four real parameters, here named η, ε_1, ε_2, $\varepsilon_3 \in \mathbb{R}$, subject to a unit norm constraint, so that they are points lying on the surface of the unit hypersphere $\mathrm{S}^3 \subset \mathbb{R}^4$. Euler parameters are also *unit quaternions*. Let $\xi = \begin{bmatrix} \eta & \varepsilon^T \end{bmatrix}^T \in \mathrm{S}^3$, with $\varepsilon = \begin{bmatrix} \varepsilon_1 & \varepsilon_2 & \varepsilon_3 \end{bmatrix}^T \in \mathbb{R}^3$, be a unit quaternion. Then the unit norm constraint can be written as

$$\| \xi \| = \eta^2 + \varepsilon^T \varepsilon = 1. \tag{9}$$

An important property of unit quaternions is that they form a multiplicative group (in fact, a Lie group) using the quaternion multiplication defined in (1). More formally:

$$\begin{bmatrix} \eta_1 \\ \varepsilon_1 \end{bmatrix}, \begin{bmatrix} \eta_2 \\ \varepsilon_2 \end{bmatrix} \in \mathrm{S}^3 \Rightarrow \begin{bmatrix} \eta_1 \\ \varepsilon_1 \end{bmatrix} \otimes \begin{bmatrix} \eta_2 \\ \varepsilon_2 \end{bmatrix} = \begin{bmatrix} \eta_1\eta_2 - \varepsilon_1^T \varepsilon_2 \\ \eta_1 \varepsilon_2 + \eta_2 \varepsilon_1 + S(\varepsilon_1)\varepsilon_2 \end{bmatrix} \in \mathrm{S}^3. \tag{10}$$

The proof of (10) relies in showing that the multiplication result has unit norm, by using (9) and some properties of the skew–symmetric operator.
Given the angle/axis pair for a given orientation, the Euler parameters can be easily obtained, since (Sciavicco & Siciliano, 2000):

$$\eta = \cos\left(\frac{\theta}{2}\right), \quad \varepsilon = \sin\left(\frac{\theta}{2}\right) u. \tag{11}$$

And it is worth noticing that the Euler parameters solve the singularity of the angle/axis pair when $\theta = 0$. If a rotation matrix $R = \{r_{ij}\} \in \mathrm{SO}(3)$ is given, then the corresponding Euler parameters can be extracted using (Craig, 2005):

$$\begin{bmatrix} \eta \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \frac{1}{2\sqrt{r_{11} + r_{22} + r_{33} + 1}} \begin{bmatrix} r_{11} + r_{22} + r_{33} + 1 \\ r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \tag{12}$$

Conversely, given the Euler parameters $\begin{bmatrix} \eta & \varepsilon^T \end{bmatrix}^T \in \mathrm{S}^3$ for a given orientation, the corresponding rotation matrix $R \in \mathrm{SO}(3)$ can be found as (Sciavicco & Siciliano, 2000):

$$R(\eta, \varepsilon) = (\eta^2 - \varepsilon^T \varepsilon) I + 2\eta S(\varepsilon) + 2\varepsilon\varepsilon^T. \tag{13}$$

However, note in (13) that $R(\eta, \varepsilon) = R(-\eta, -\varepsilon)$, meaning that S^3 is a double cover of $\mathrm{SO}(3)$. That is to say that every given orientation $\phi \in \mathrm{M}^3$ corresponds to one and only one rotation matrix, but maps into two unit quaternions, representing antipodes in the unit hypersphere (Gallier, 2000):

$$R \in \mathrm{SO}(3) \quad \Leftrightarrow \quad \pm\begin{bmatrix} \eta \\ \varepsilon \end{bmatrix} \in \mathrm{S}^3. \tag{14}$$

Also from (13) we can verify that $R(\eta, -\varepsilon) = R(\eta, \varepsilon)^T$ and, as $\begin{bmatrix} \eta & -\varepsilon^T \end{bmatrix}^T \in \mathrm{S}^3$ is the conjugate (or inverse) of $\begin{bmatrix} \eta & \varepsilon^T \end{bmatrix}^T \in \mathrm{S}^3$, then we have that

$$R^T \in \mathrm{SO}(3) \quad \Leftrightarrow \quad \pm\begin{bmatrix} \eta \\ \varepsilon \end{bmatrix}^* = \pm\begin{bmatrix} \eta \\ -\varepsilon \end{bmatrix} \in \mathrm{S}^3.$$

Another important fact is that quaternion multiplication is equivalent to the rotation matrix multiplication. Thus, if $\pm\xi_1, \pm\xi_2 \in \mathrm{S}^3$ are the Euler parameters corresponding to the rotation matrices $R_1, R_2 \in \mathrm{SO}(3)$, respectively, then:

$$R_1 R_2 \quad \Leftrightarrow \quad \pm\xi_1 \otimes \xi_2. \tag{15}$$

Moreover, the premultiplication of a vector $v \in \mathbb{R}^3$ by a rotation matrix $R \in \mathrm{SO}(3)$ produces a transformed (rotated) vector $w = Rv \in \mathbb{R}^3$. The same transformation can be performed with quaternion multiplication using $\bar{w} = \xi \otimes \bar{v} \otimes \xi^*$, where $\xi \in \mathrm{S}^3$ is the unit quaternion corresponding to R and quaternions $\bar{v}$, $\bar{w} \in \mathbb{H}$ are formed adding a null scalar part to the corresponding vectors ($\bar{v} = \begin{bmatrix} 0 & v^T \end{bmatrix}^T$, $\bar{w} = \begin{bmatrix} 0 & w^T \end{bmatrix}^T$). In sum

$$Rv \in \mathbb{R}^3 \quad \Leftrightarrow \quad \xi_1 \otimes \bar{v} \otimes \xi_1^* \in \mathbb{R}^4. \tag{16}$$

In the case of a time–varying orientation, we need to establish a relation between the time derivatives of Euler parameters $\dot{\xi} = \begin{bmatrix} \dot{\eta} & \dot{\varepsilon}^T \end{bmatrix}^T \in \mathbb{R}^4$ and the angular velocity of the rigid body $\omega \in \mathbb{R}^3$. That relation is given by the so–called *quaternion propagation rule* (Sciavicco & Siciliano, 2000):

$$\dot{\xi} = \frac{1}{2} E(\xi)\omega. \tag{17}$$

where

$$E(\xi) = E(\eta, \varepsilon) = \begin{bmatrix} -\varepsilon^T \\ \eta I - S(\varepsilon) \end{bmatrix} \in \mathbb{R}^{4\times 3}. \tag{18}$$

By using properties (2), (8) and the unit norm constraint (9) it can be shown that $E(\xi)^T E(\xi) = I$, so that ω can be resolved from (17) as

$$\omega = 2E(\xi)^T \dot{\xi} \tag{19}$$

3. Robot Kinematics and Dynamics

Serial robot manipulators have an *open kinematic chain*, i.e., there is only one path from one end (the base) to the other (the end–effector) of the robotic chain (see Fig. 3). A serial manipulator with n joints, either translational (prismatic) or rotational, has n *degrees of freedom* (dof). This section deals only with the modeling of serial manipulators.

As explained in (Craig, 2005), any serial robot manipulator can be described by using four parameters for each joint, that is, a total of $4n$ parameters for a n –dof manipulator. Denavit and Hartenberg (Denavit & Hartenberg, 1955) proposed a general method to establish those parameters in a systematic way, by defining $n+1$ coordinate frames, one per each link, including the base. The four Denavit–Hartenberg (or simply D–H) parameters for the i -th joint can then be extracted from the relation between frame $i-1$ and frame i , as follows:

a_i : Distance from axis z_{i-1} to z_i , along axis x_i .

α_i : Angle between axes z_{i-1} and z_i , about axis x_i .

d_i : Distance from axis x_{i-1} to x_i , along axis z_{i-1} .

θ_i : Angle between axes x_{i-1} and x_i , about axis z_{i-1} .

Note that there are three constant D–H parameters for each joint; the other one (θ_i for rotational joints, d_i for translational joints) describes the motion produced by such i -th joint. To follow the common notation, let q_i be the variable D–H parameter corresponding to the i –th joint. Then the so–called joint configuration space is specified by the vector

$$q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T \in \mathbb{R}^n,$$

while the pose (position and orientation) of the end–effector frame with respect to the base frame (see Fig. 3), denoted here by x , is given by a point in the 6–dimensional pose configuration space $\mathbb{R}^3 \times \mathrm{M}^3$, using whichever of the representations for $\mathrm{M}^3 \subset \mathbb{R}^m$. That is

$$x = \begin{bmatrix} p \\ \phi \end{bmatrix} \in \mathbb{R}^3 \times \mathrm{M}^3 \subset \mathbb{R}^{3+m}.$$

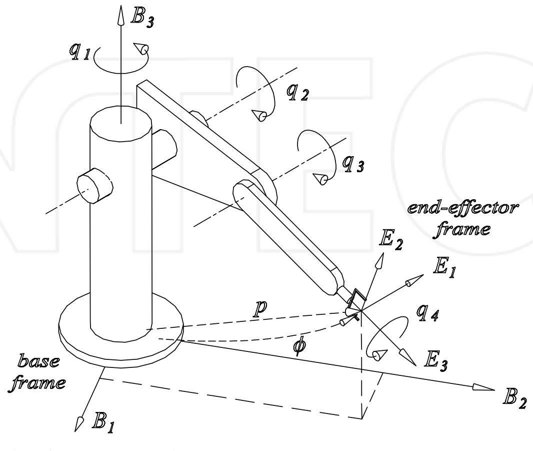

Figure 3. A 4-dof serial robot manipulator

3.1 Direct Kinematics

The direct kinematics problem consists on expressing the end–effector pose x in terms of q. That is, to find a function (called the *kinematics function*) $h : \mathbb{R}^n \to \mathbb{R}^3 \times \mathrm{M}^3$ such that

$$x = h(q) = \begin{bmatrix} p(q) \\ \phi(q) \end{bmatrix}. \tag{20}$$

Traditionally, direct kinematics is solved from the D–H parameters using *homogeneous matrices* (Denavit & Hartenberg, 1955). A homogeneous matrix combines a rotation matrix and a position vector in an extended 4×4 matrix. Thus, given $p \in \mathbb{R}^3$ and $R \in \mathrm{SO}(3)$, the corresponding homogeneous matrix T is

$$T = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} \in \mathrm{SE}(3) \subset \mathbb{R}^{4\times4}. \tag{21}$$

$\mathrm{SE}(3)$ is known as the *special Euclidean group* and contains all homogeneous matrices of the form (21). It is a group under the standard matrix multiplication, meaning that the product of two homogeneous matrices $T_1, T_2 \in \mathrm{SE}(3)$ is given by

$$T_2T_1 = \begin{bmatrix} R_2 & p_2 \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} R_1 & p_1 \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R_2R_1 & R_2p_1+p_2 \\ 0^T & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{22}$$

In terms of the D–H parameters, the relative position vector and rotation matrix of the i-th frame with respect to the previous one are (Sciavicco & Siciliano, 2000):

$$^{i-1}R_i = \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} \end{bmatrix} \in \mathrm{SO}(3), \qquad ^{i-1}p_i = \begin{bmatrix} a_iC_{\theta_i} \\ a_iS_{\theta_i} \\ d_i \end{bmatrix} \in \mathbb{R}^3. \tag{23}$$

where C_x, S_x stand for $\cos(x)$, $\sin(x)$, respectively.
The original method proposed by (Denavit & Hartenberg, 1955) uses the expressions in (23) to form relative homogeneous matrices $^{i-1}T_i$ (each depending on q_i), and then the composition rule (22) to compute the homogeneous matrix of the end–effector with respect to the base frame, $T(q) \in \mathrm{SE}(3)$:

$$T(q) = \begin{bmatrix} R(q) & p(q) \\ 0^T & 1 \end{bmatrix} = {}^0T_1\, {}^1T_2 \cdots {}^{n-1}T_n \in \mathrm{SE}(3).$$

This leads to the following general expressions for the pose of the end–effector, in terms of $p(q)$ and $R(q)$

$$R(q) = {}^0R_1\, {}^1R_2 \cdots {}^{n-1}R_n \tag{24}$$

$$p(q) = {}^0R_1\, {}^1R_2 \cdots {}^{n-2}R_{n-1}\, {}^{n-1}p_n + {}^0R_1\, {}^1R_2 \cdots {}^{n-3}R_{n-2}\, {}^{n-2}p_{n-1} + \ldots + {}^2R_1\, {}^1p_2 + {}^0p_1 \tag{25}$$

An alternative method for computing the direct kinematics of serial manipulators using quaternions, instead of homogeneous matrices, was proposed recently by (Campa et al., 2006). The method is inspired in one found in (Barrientos et al. 1997), but it is more general, and gives explicit expressions for the position $p(q) \in \mathbb{R}^3$ and orientation $\xi(q) \in \mathrm{S}^3 \subset \mathbb{R}^4$ in terms of the original D–H parameters.

Let us start from the expressions for ${}^{i-1}R_i$ and ${}^{i-1}p_i$ in (23). We can use (12) and some trigonometric identities to obtain the corresponding quaternion ${}^{i-1}\xi_i$ from ${}^{i-1}R_i$; also, we can extend ${}^{i-1}p_i$ to form the quaternion ${}^{i-1}\overline{p}_i$. The results are

$$ {}^{i-1}\xi_i = \begin{bmatrix} \cos\left(\frac{\theta_i}{2}\right)\cos\left(\frac{\alpha_i}{2}\right) \\ \cos\left(\frac{\theta_i}{2}\right)\sin\left(\frac{\alpha_i}{2}\right) \\ \sin\left(\frac{\theta_i}{2}\right)\sin\left(\frac{\alpha_i}{2}\right) \\ \sin\left(\frac{\theta_i}{2}\right)\cos\left(\frac{\alpha_i}{2}\right) \end{bmatrix} \in \mathrm{S}^3, \qquad {}^{i-1}\overline{p}_i = \begin{bmatrix} 0 \\ a_i\cos(\theta_i) \\ a_i\sin(\theta_i) \\ d_i \end{bmatrix} \in \mathbb{H} $$

Then, we can apply properties (15) and (16) iteratively to obtain expressions equivalent to (24) and (25):

$$ \begin{aligned} \xi(q) &= {}^0\xi_1 \otimes {}^1\xi_2 \otimes \cdots \otimes {}^{n-1}\xi_n, \\ \overline{p}(q) &= ({}^0\xi_1 \otimes {}^1\xi_2 \otimes \cdots \otimes {}^{n-2}\xi_{n-1}) \otimes {}^{n-1}\overline{p}_n \otimes ({}^0\xi_1 \otimes {}^1\xi_2 \otimes \cdots \otimes {}^{n-2}\xi_{n-1})^* + \\ &\quad ({}^0\xi_1 \otimes {}^1\xi_2 \otimes \cdots \otimes {}^{n-3}\xi_{n-2}) \otimes {}^{n-2}\overline{p}_{n-1} \otimes ({}^0\xi_1 \otimes {}^1\xi_2 \otimes \cdots \otimes {}^{n-3}\xi_{n-2})^* + \ldots + {}^0\xi_1 \otimes {}^1\overline{p}_2\, {}^0\xi_1^* + {}^0\overline{p}_1. \end{aligned} $$

$\xi(q)$ is the unit quaternion (Euler parameters) expressing the orientation of the end–effector with respect to the base frame. The position vector $p(q)$ is the vector part of the quaternion $\overline{p}(q)$.

An advantage of the proposed method is that the position and orientation parts of the pose are processed separately, using only quaternion multiplications, which are computationally more efficient than homogeneous matrix multiplications. Besides, the result for the orientation part is given directly as a unit quaternion, which is a requirement in task space controllers (see Section 5).

The *inverse kinematics problem*, on the other hand, consists on finding explicit expressions for computing the joint coordinates, given the pose coordinates, that is equivalent to find the inverse funcion of h in (20), mapping $\mathbb{R}^3\times\mathrm{M}^3$ to $\mathbb{R}^n$:

$$ q = h^{-1}(x). \tag{26} $$

The inverse kinematics, however, is much more complex than direct kinematics, and is not derived in this work.

3.2 Differential Kinematics

Differential kinematics gives the relationship between the joint velocities and the corresponding end–effector's linear and angular velocities (Sciavicco & Siciliano, 2000).

Thus, if $\dot{q} = \frac{d}{dt} q \in \mathbb{R}^n$ is the vector of joint velocities, $v \in \mathbb{R}^3$ is the linear velocity, and $\omega \in \mathbb{R}^3$ is the angular velocity of the end–effector, then the *differential kinematics equation* is given by

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_o(q) \end{bmatrix} \dot{q} = J(q)\dot{q} \tag{27}$$

where matrix $J(q) \in \mathbb{R}^{6 \times n}$ is the *geometric Jacobian* of the manipulator, which depends on its configuration. $J_p(q)$, $J_o(q) \in \mathbb{R}^{3 \times n}$ are the components of the geometric Jacobian producing the translational and rotational parts of the motion.
Now consider the time derivative of the generic *kinematics function* (20):

$$\dot{x} = \frac{\partial h(q)}{\partial q} \dot{q} = \begin{bmatrix} \dfrac{\partial p(q)}{\partial q} \\ \dfrac{\partial \phi(q)}{\partial q} \end{bmatrix} \dot{q} = J_A(q)\dot{q}. \tag{28}$$

Matrix $J_A(q) \in \mathbb{R}^{(3+m) \times n}$ is known as the *analytic Jacobian* of the robot, and it relates the joint velocities with the time derivatives of the pose coordinates (or *pose velocities*).
In the particular case of using Euler parameters for the orientation, $\phi(q) = \xi(q) \in S^3 \subset \mathbb{R}^4$, and $J_A(q) = J_{A_\xi}(q) \in \mathbb{R}^{7 \times n}$, i.e.

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\xi} \end{bmatrix} = J_{A_\xi}(q)\dot{q}. \tag{29}$$

The linear velocity v of the end–effector is simply the time derivative of the position vector p (i.e. $v = \dot{p}$); instead, the angular velocity ω is related with $\dot{\xi}$ by (19). So, we have

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J_{R_\xi}(q) \begin{bmatrix} \dot{p} \\ \dot{\xi} \end{bmatrix} \tag{30}$$

where the *representation Jacobian*, for the case of using Euler parameters, is defined as

$$J_{R_\xi}(q) = \begin{bmatrix} I & 0 \\ 0 & 2E(\xi(q))^T \end{bmatrix} \in \mathbb{R}^{6 \times 7} \tag{31}$$

with matrix $E(\xi)$ as in (18). Also notice, by combining (27), (29), and (30), that

$$J(q) = J_{R_\xi}(q) J_{A_\xi}(q)$$

Whichever the Jacobian matrix be (geometric, analytic, representation), it is often required to find its inverse mapping, although this is possible only if such a matrix has full rank. In the case of a matrix $J \in \mathbb{R}^{n \times m}$, with $n > m$, then there exists a *left pseudoinverse* matrix of J, $J^\dagger \in \mathbb{R}^{m \times n}$, defined as

$$J^\dagger = \left(J^T J\right)^{-1} J^T$$

such that $J^{\dagger}J = I \in \mathbb{R}^{m\times m}$. On the contrary, if $n < m$, then there exist a *right pseudoinverse* matrix of J, ${}^{\dagger}J \in \mathbb{R}^{m\times n}$, defined as

$${}^{\dagger}J = J^T\left(JJ^T\right)^{-1}$$

such that $J\,{}^{\dagger}J = I \in \mathbb{R}^{n\times n}$. It is easy to show that in the case $n = m$ (square matrix) then $J^{\dagger} = {}^{\dagger}J = J^{-1}$.

3.3 Lagrangian Dynamics

Robot kinematics studies the relations between joint and pose variables (and their time derivatives) during the motion of the robot's end–effector. These relations, however, are established using only a geometric point of view. The effect of mechanical forces (either external or produced during the motion itself) acting on the manipulator is studied by *robot dynamics*.

As pointed out by (Sciavicco & Siciliano, 2000), the derivation of the dynamic model of a manipulator plays an important role for simulation, analysis of manipulator structures, and design of control algorithms. There are two general methods for derivation of the dynamic equations of motion of a manipulator: one method is based on the *Lagrange formulation* and is conceptually simple and systematic; the other method is based on the *Newton–Euler formulation* and allows obtaining the model in a recursive form, so that it is computationally more efficient.

In this section we consider the application of the Lagrangian formulation for obtaining the robot dynamics in the case of using Euler parameters for describing the orientation. This approach is not new, since it has been employed for modeling the dynamics of rigid bodies (see, e.g., Shivarama & Fahrenthold, 2004; Lu & Ren, 2006, and references therein).

An important fact is that Lagrange formulation allows to obtain the dynamic model independently of the coordinate system employed to describe the motion of the robot. Let us consider a serial robot manipulator with n degrees of freedom. Then we can choose any set of $m = n + k$ coordinates, say $\rho \in \mathbb{R}^m$, with $k \geq 0$ holonomic constraints of the form $\gamma_i(\rho) = 0$ ($i = 1, 2, \ldots, k$), to obtain the robot's dynamic model. If $k = 0$ we talk of independent or *generalized coordinates*, if $k \geq 1$ we have dependent or *constrained coordinates*.

The next step is to express the total (kinetic and potential) energy of the mechanical system in terms of the chosen coordinates. Let $\mathcal{K}(\rho, \dot{\rho})$ be the kinetic energy, and $\mathcal{U}(\rho)$ the potential energy of the system; then the *Lagrangian function* $\mathcal{L}(\rho, \dot{\rho})$ is defined as

$$\mathcal{L}(\rho, \dot{\rho}) = \mathcal{K}(\rho, \dot{\rho}) - \mathcal{U}(\rho)$$

The Lagrange's dynamic equations of motion, for the general case (with constraints), are expressed by

$$\frac{d}{dt}\left\{\frac{\partial \mathcal{L}(\rho, \dot{\rho})}{\partial \dot{\rho}}\right\} - \frac{\partial \mathcal{L}(\rho, \dot{\rho})}{\partial \rho} = \tau_{\rho} + \left(\frac{\partial \gamma(\rho)}{\partial \rho}\right)^T \lambda \tag{32}$$

where $\tau_{\rho} \in \mathbb{R}^m$ is the vector of (generalized or constrained) forces associated with each of the coordinates ρ, vector $\gamma(\rho)$ is defined as

$$\gamma(\rho) = \begin{bmatrix} \gamma_1(\rho) \\ \gamma_2(\rho) \\ \vdots \\ \gamma_k(\rho) \end{bmatrix} \in \mathbb{R}^k, \quad \text{so that} \quad \frac{\partial \gamma(\rho)}{\partial \rho} \in \mathbb{R}^{k\times m}$$

and $\lambda \in \mathbb{R}^k$ is the vector of k *Lagrange multipliers*, which are employed to reduce the dimension of the system, producing only n independent equations in terms of a new set $r \in \mathbb{R}^n$ of generalized coordinates. In sum, a general constrained system such as (32), with $\rho \in \mathbb{R}^{n+k}$, can be transformed in an unconstrained system of the form

$$\frac{d}{dt}\left\{\frac{\partial \mathcal{L}(r,\dot{r})}{\partial \dot{r}}\right\} - \frac{\partial \mathcal{L}(r,\dot{r})}{\partial r} = \tau_r \tag{33}$$

Moreover, as explained by (Spong et al., 2006), equation (33) can be rewritten as:

$$M_r(r)\ddot{r} + C_r(r,\dot{r})\dot{r} + g_r(r) = \tau_r$$

where $M_r(r) \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix known as the inertia matrix, $C_r(r,\dot{r}) \in \mathbb{R}^{n\times n}$ is the matrix of centripetal and Coriolis forces, and $g_r(r) \in \mathbb{R}^n$ is the vector of forces due to gravity. These matrices satisfy the following useful properties (Kelly et al., 2005):

- Inertia matrix is directly related to kinetic energy by

$$\mathcal{K}(r,\dot{r}) = \frac{1}{2}\dot{r}^T M_r(r)\dot{r} \tag{34}$$

- For a given robot, matrix $C_r(r,\dot{r})$ is not unique, but it can always be chosen so that $\dot{M}_r(r) - 2C_r(r,\dot{r})$ be skew–symmetric, that is

$$x^T\left[\dot{M}_r(r) - 2C_r(r,\dot{r})\right]x = 0, \quad \forall x, r, \dot{r} \in \mathbb{R}^n \tag{35}$$

- The vector of gravity forces is the gradient of the potential energy, that is

$$g_r(r) = \frac{\partial \mathcal{U}(r)}{\partial r} \tag{36}$$

Now consider the total mechanical energy of the robot $\mathcal{E}(r,\dot{r}) = \mathcal{K}(r,\dot{r}) + \mathcal{U}(r)$. Using properties (34)–(36), it can be shown that the following relation stands

$$\dot{\mathcal{E}}(r,\dot{r}) = \dot{r}^T \tau_r. \tag{37}$$

And, as the rate of change of the total energy in a system is independent of the generalized coordinates, we can use (37) as a way of convert generalized forces between two different generalized coordinate systems.

For an n –dof robot manipulator, it is common to express the dynamic model in terms of the joint coordinates, i.e. $r \equiv q \in \mathbb{R}^n$, this leads us to the well–known equation (Kelly et al., 2005):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau, \quad (q,\dot{q},\ddot{q},\tau \in \mathbb{R}^n). \tag{38}$$

Subindexes have been omitted in this case for simplicity; τ is the vector of joint forces (and torques) applied to each of the robot joints.
Now consider the problem of modeling the robot dynamics of a non–redundant ($n \le 6$) manipulator, in terms of the pose coordinates of the end–effector, given by $x = \begin{bmatrix} p^T & \xi^T \end{bmatrix}^T \in \mathbb{R}^3 \times \mathrm{S}^3 \subset \mathbb{R}^7$. Notice that in this case we have holonomic constraints. If $n = 6$ then the only constraint is given by (9), so that:

$$\gamma(x) = \eta^2 + \varepsilon^T\varepsilon - 1;$$

If $n < 6$ then the pose coordinates require $6 - n$ additional constraints to form the vector $\gamma(x) \in \mathbb{R}^{7-n}$ The equations of motion would be

$$M_x(x)\ddot{x} + C_x(x,\dot{x})\dot{x} + g_x(x) = \tau_x + \frac{\partial \gamma(x)}{\partial x}^T \lambda, \quad (x,\dot{x},\ddot{x},\tau_x \in \mathbb{R}^7), \tag{39}$$

where matrices $M_x(x)$, $C_x(x,x)$, $g_x(x)$ can be computed from $M(q)$, $C(q,q)$, $g(q)$ in (38) using a procedure similar to the one presented in (Khatib, 1987) for the case of operational space. We get

$$M_x(x) = {}^{\dagger}(J_{A_\xi}(q)^T)M(q)J_{A_\xi}(q)^{\dagger}\Big|_{q=h^{-1}(x)}$$
$$C_x(x,\dot{x}) = {}^{\dagger}(J_{A_\xi}(q)^T)C(q,J_{A_\xi}(q)^{\dagger}\dot{x})J_{A_\xi}(q)^{\dagger} + {}^{\dagger}(J_{A_\xi}(q)^T)M(q)\dot{J}_{A_\xi}(q)^{\dagger}\Big|_{q=h^{-1}(x)}$$
$$g_x(x) = {}^{\dagger}(J_{A_\xi}(q)^T)g(q)\Big|_{q=h^{-1}(x)}$$

where $J_{A_\xi}(q)$ is the analytic Jacobian in (29). To obtain the previous equations we are assuming, according to (37), that

$$\dot{q}^T\tau = \dot{x}^T\tau_x = \dot{q}^T J_{A_\xi}(q)^T \tau_x$$

so that

$$\tau_x = {}^{\dagger}(J_{A_\xi}(q)^T)\tau\Big|_{q=h^{-1}(x)}.$$

Equation (39) expresses the dynamics of a robot manipulator in terms of the pose coordinates (p , ξ) of the end–effector. In fact, these are a set of seven equations which can be reduced to n , by using the $7 - n$ Lagrange multipliers and choosing a subset of n

generalized coordinates taken from x. This procedure, however, can be far complex, and sometimes unsolvable in analytic form.
In theory, we can use (32) to find a minimal system of equations in terms of any set of generalized coordinates. For example, we could choose the scalar part of the Euler parameters in each joint, η_i, ($i = 1, 2, \ldots, n$) as a possible set of coordinates. To this end, it is useful to recall that the kinetic energy of a free–moving rigid body is a function of the linear and angular velocities of its center of mass, v and ω, respectively, and is given by

$$\mathcal{K}(v,\omega) = \frac{1}{2} m v^T v + \frac{1}{2}\omega^T H \omega, \tag{40}$$

where m is the mass, and H is the matrix of moments of inertia (with respect to the center of mass) of the rigid body. The potential energy depends only on the coordinates of the center of mass, p, and is given by

$$\mathcal{U}(p) = m g^T p,$$

where g is the vector of gravity acceleration. Now, using (30), (31), we can rewrite (40) as

$$\mathcal{K}(\dot{p},\xi,\dot{\xi}) = \frac{1}{2} m \dot{p}^T \dot{p} + 2\dot{\xi}^T E(\xi) H E(\xi)^T \dot{\xi} \tag{41}$$

Considering this, we propose the following procedure to compute the dynamic model of robot manipulators in terms of any combination of pose variables in task space:

1. Using (41), find the kinetic and potential energy of each of the manipulator's links, in terms of the pose coordinates, as if they were independent rigid bodies. That is, for link i, compute $\mathcal{K}_i(\dot{p}_i, \xi_i, \dot{\xi}_i)$ and $\mathcal{U}_i(p_i)$.
2. Still considering each link as an independent rigid body, compute the total Lagrangian function of the n –dof manipulator as

$$\mathcal{L}(p,\xi,\dot{p},\dot{\xi}) = \sum_{i=1}^{n}\left[\mathcal{K}_i(\dot{p}_i,\xi_i,\dot{\xi}_i) - \mathcal{U}_i(p_i)\right]$$

   where $\rho \in \mathbb{R}^{7n}$ is a vector containing the pose coordinates of each link, i.e.

$$\rho = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \text{ with } x_i = \begin{bmatrix} p_i \\ \xi_i \end{bmatrix}$$

3. Now use (9), for each joint, and the knowledge of the geometric relations between links to find $6n$ holonomic constraints, in order to form vector $\gamma(\rho) \in \mathbb{R}^{6n}$.
4. Write the constrained equations of motion for the robot, i. e. (32).
5. Finally, solve for the $6n$ Lagrange multipliers, and get de reduced (generalized) system in terms of the chosen n generalized coordinates.

This procedure, even if not simple (specially for robots with $n > 3$) can be useful when requiring explicit equations for the robot dynamics in other than joint coordinates.

4. Path Planning

The minimal requirement for a manipulator is the capability to move from an initial posture to a final assigned posture. The transition should be characterized by motion laws requiring the actuators to exert joint forces which do not violate the saturation limits and do not excite typically unmodeled resonant modes of the structure. It is then necessary to devise planning algorithms that generate suitably *smooth* trajectories (Sciavicco & Siciliano, 2000).
To avoid confusion, a *path* denotes simply a locus of points either in joint or pose space; it is a pure geometric description of motion. A *trajectory*, on the other hand, is a path on which a time law is specified (e.g., in terms of velocities and accelerations at each point of the path). This section shows a couple of methods for designing paths in the task space, i.e., when using Euler parameters for describing orientation.
Generally, the specification of a path for a given motion task is done in pose space. It is much easier for the robot's operator to define a desired position and orientation of the end–effector than to compute the required joint angles for such configuration. A time–varying position $p(t)$ is also easy to define, but the same is not valid for orientation, due to the difficulty of visualizing the orientation coordinates.
Of the four orientation parameterizations mentioned in Section 2, perhaps the more intuitive is the angle/axis pair. To reach a new orientation from the actual one, employing a minimal displacement, we only need to find the required axis of rotation $\tilde{u} \in \mathrm{S}^2$ and the corresponding angle to rotate $\tilde{\theta} \in \mathbb{R}$. Given the Euler parameters for the initial orientation, $\left[\eta_o \quad \varepsilon_o{}^T\right]^T \in \mathrm{S}^3$, and the desired orientation, $\left[\eta_d \quad \varepsilon_d{}^T\right]^T \in \mathrm{S}^3$, it can be shown, using (11) and quaternion operations, that (Campa & de la Torre, 2006):

$$\tilde{\theta} = 2\arccos\left(\eta_o\eta_d + \varepsilon_o^T\varepsilon_d\right), \quad \tilde{u} = \frac{\eta_o\varepsilon_d - \eta_d\varepsilon_o + S(\varepsilon_o)\varepsilon_d}{\| \eta_o\varepsilon_d - \eta_d\varepsilon_o + S(\varepsilon_o)\varepsilon_d \|} \tag{42}$$

We can use expressions in (42) to define the minimal path between the initial and final orientations. A velocity profile can be employed for $\tilde{\theta}$ in order to generate a suitable $\theta(t)$. However, during a motion along such a path, it is important that $\tilde{u}$ remains fixed. Also notice that when the desired orientation coincides with the initial (i.e., $\eta_o = \eta_d$, $\varepsilon_o = \varepsilon_d$), then $\tilde{\theta} = 0$ and $\tilde{u}$ is not well defined.
Now consider the possibility of finding a path between a given set of points in the task space using interpolation curves. The simplest way of doing so is by means of linear interpolation. Let the initial pose be given by $\left[p_o^T \quad \xi_o^T\right]^T$ and the final pose by $\left[p_d^T \quad \xi_d^T\right]^T$, and assume, for simplicity, that the time required for the motion is normalized, that is, we need to find a trajectory $\left[p(t)^T \quad \xi(t)^T\right]^T$, such that

$$\begin{bmatrix} p(0) \\ \xi(0) \end{bmatrix} = \begin{bmatrix} p_o \\ \xi_o \end{bmatrix} \in \mathbb{R}^3 \times \mathrm{S}^3, \quad \text{and} \quad \begin{bmatrix} p(1) \\ \xi(1) \end{bmatrix} = \begin{bmatrix} p_d \\ \xi_d \end{bmatrix} \in \mathbb{R}^3 \times \mathrm{S}^3$$

For the position part of the motion, the linear interpolation is simply given by

$$p(t) = (1-t)\,p_o + t\,p_d$$

but for the orientation part we can not apply the same formula, because, as unit quaternions do not form a vector space, then such $\xi(t)$ would not be a unit quaternion anymore. Shoemake (1985) solved this by defining a *spherical linear interpolation*, or simply *slerp*, which is given by the following expression

$$\xi(t) = \frac{\sin([1-t]\Omega)\xi_o + \sin(t\Omega)\xi_d}{\sin(\Omega)} \tag{43}$$

where $\Omega = \cos^{-1}(\xi_o^T \xi_d)$.

It is possible to show (Dam et al., 1998) that such $\xi(t)$ in (43) satisfies that $\xi(t) \in \mathrm{S}^3$, for all $0 \le t \le 1$. (Dam et al., 1998) is also a good reference for understanding other types of interpolation using quaternions, including the so called *spherical spline quaternion interpolation* or *squad*, which produces smooth curves joining a series of points in the orientation manifold.

Even though these algorithms have been used in the fields of computer graphics and animation for years, as far as the authors' knowledge, there are few works using them for motion planning in robotics.

5. Task-space Control

As pointed out by (Sciavicco & Siciliano, 2000), the joint space control is sufficient in those cases where it is required motion control in free space; however, for the most common applications of interaction control (robot interacting with the environment) it is better to use the so–called *task space* control (Natale,2003). Thus, in the later years, more research has been done in this kind of control schemes.

Euler parameters have been employed in the later years for the control of generic rigid bodies, including spaceships and underwater vehicles. The use of Euler parameters in robot control begins with Yuan (1988) who applied them to a particular version of the *resolved–acceleration control* (Luh et al., 1980). More recently, several works have been published on this topic (see, e.g., Lin, 1995; Caccavale et al., 1999; Natale, 2003; Xian et al., 2004; Campa et al., 2006, and references therein).

The so–called *hierarchical control* of manipulators is a technique that solves the problem of motion control in two steps (Kelly & Moreno-Valenzuela, 2005): first, a task space kinematic control is used to compute the desired joint velocities from the desired pose trajectories; then, those desired velocities become the inputs for an inner joint velocity control loop (see Fig. 4). This two–loop control scheme was first proposed by Aicardi et al. (1995) in order to solve the pose control problem using a particular set of variables to describe the end–effector orientation.

The term *kinematic controller* (Caccavale et al., 1999) refers to that kind of controller whose output signal is a joint velocity and is the first stage in a hierarchical scheme. On the other hand *dynamic controller* is employed here for those controllers also including a position controller but producing joint torque signals.

In this section we analyze two common task–space control schemes: the *resolved–motion rate control* (*RMRC*), which is a simple kinematic controller, and the *resolved–acceleration control* (*RAC*), a dynamic controller. These controllers are well–known in literature, however, the stability analysis presented here is done employing Euler parameters as space–state variables for describing the closed–loop control system.

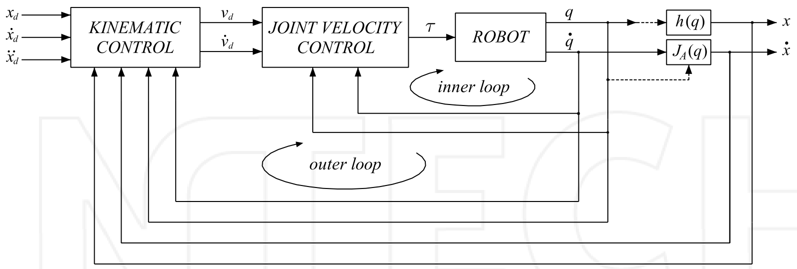

Figure 4. Pose–space hierarchical control

5.1 Pose Errror and Control Objective

For the analysis forthcoming let us consider that the motion of the manipulator's end–effector is described through its pose, given by $x(q) = \left[p(q)^T \quad \xi(q)^T \right]^T \in \mathbb{R}^3 \times \mathrm{S}^3$, and its linear and angular velocities referred to the base frame, $\dot{p}(q) \in \mathbb{R}^3$ and $\omega(q) \in \mathbb{R}^3$, respectively. These terms can be computed from the measured joint variables ($q, \dot{q} \in \mathbb{R}^n$) using the direct kinematics function (20) and the differential kinematics equation (27).

Also, let the desired end–effector trajectory be specified by the desired pose $x_d(t) = \left[p_d(t)^T \quad \xi_d(t)^T \right]^T \in \mathbb{R}^3 \times \mathrm{S}^3$, where $\xi_d(t) = \left[\eta_d(t) \quad \varepsilon_d(t)^T \right]^T \in \mathrm{S}^3$, and the desired linear and angular velocities referred to the base frame, $\dot{p}_d(t), \omega_d(t) \in \mathbb{R}^3$. The time derivative of the desired Euler parameters is related to the desired angular velocity through an expression similar to (17), i.e.

$$\dot{\xi}_d = \frac{1}{2} E(\xi_d)\omega_d. \tag{44}$$

with matrix operator $E(\cdot)$ defined in (18). The desired pose x_d can be considered as describing the position and orientation of a desired frame with respect to the base frame, as shown in Fig. 5.

The position error vector is simply the difference between the desired and actual positions the robot's end–effector, that is $\tilde{p} = p_d - p$; the linear and angular velocity errors are computed in a similar way

$$\dot{\tilde{p}} = \dot{p}_d - \dot{p}, \qquad \tilde{\omega} = \omega_d - \omega. \tag{45}$$

However, as unit quaternions do not form a vector space, they cannot be subtracted to form the orientation error; instead, we should use the properties of the quaternion group algebra. Let $\tilde{\xi} = \left[\tilde{\eta} \quad \tilde{\varepsilon}^T \right]^T$ be the Euler parameters of the orientation error, then (Lin, 1995):

$$\tilde{\xi} = \xi_d \otimes \xi^* = \begin{bmatrix} \eta_d \\ \varepsilon_d \end{bmatrix} \otimes \begin{bmatrix} \eta \\ -\varepsilon \end{bmatrix}$$

and applying (1) we get (Sciavicco & Siciliano, 2000):

$$\tilde{\xi} = \begin{bmatrix} \tilde{\eta} \\ \tilde{\varepsilon} \end{bmatrix} = \begin{bmatrix} \eta\eta_d + \varepsilon^T \varepsilon_d \\ \eta\varepsilon_d - \eta_d \varepsilon + S(\varepsilon)\varepsilon_d \end{bmatrix} \in \mathrm{S}^3. \tag{46}$$

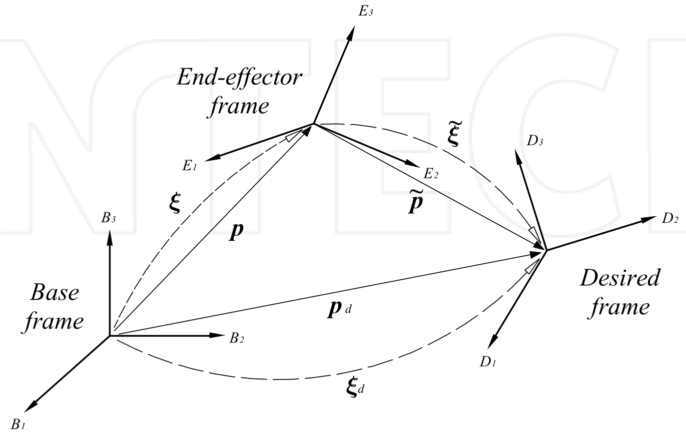

Figure 5. Actual and desired coordinate frames for defining the pose error

Taking the time derivative of (46), considering (17), (44) and properties (3), (4), and (8), it can be shown that (Fjellstad, 1994):

$$\begin{bmatrix} \dot{\tilde{\eta}} \\ \dot{\tilde{\varepsilon}} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\tilde{\varepsilon}^T \\ \tilde{\eta} I + S(\tilde{\varepsilon}) \end{bmatrix} \tilde{\omega} - \begin{bmatrix} 0 \\ S(\tilde{\varepsilon}) \end{bmatrix} \omega_d, \tag{47}$$

where $\tilde{\omega}$ is defined as in (45).

Notice that, considering the properties of unit quaternions, when the end-effector orientation equals the desired orientation, i.e. $\xi = \xi_d$, the orientation error becomes $\tilde{\xi} = \begin{bmatrix} 1 & 0^T \end{bmatrix}^T \in \mathrm{S}^3$. If $\xi = -\xi_d$, then $\tilde{\xi} = \begin{bmatrix} -1 & 0^T \end{bmatrix}^T \in \mathrm{S}^3$, however, by (14) this represents the *same orientation*.

From the previous analysis we can establish the *pose control objective* as

$$\lim_{t \to \infty} \begin{bmatrix} \tilde{p}(t) \\ |\tilde{\eta}(t)| \\ \tilde{\varepsilon}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}; \quad \tilde{p} \in \mathbb{R}^3, \begin{bmatrix} \tilde{\eta} \\ \tilde{\varepsilon} \end{bmatrix} \in \mathrm{S}^3, \tag{48}$$

or, in words, that the end–effector position and orientation converge to the desired position and orientation.

5.2 A Hierarchical Controller

Let us consider a control scheme such as the one in Fig. 4. As the robot model we take the joint–space dynamics given in (38).

The RMRC is a kinematic controller first proposed by Whitney (1969) to handle the problem of motion control in operational space via joint velocities (rates). Let $x, x_d \in \mathbb{R}^6$ be the actual and desired operational coordinates, respectively, so that $\tilde{x} = x_d - x$ is the pose error in operational space. The RMRC controller is given by

$$v_d = J_A(q)^\dagger \left[\dot{x}_d + K_x \tilde{x}\right] \tag{49}$$

where $J_A(q)^\dagger \in \mathbb{R}^{n\times 6}$ is the left pseudoinverse of the analytic Jacobian in (28), $K_x \in \mathbb{R}^{n\times n}$ is a positive definite matrix, and $v_d \in \mathbb{R}^n$ is the vector of (resolved) desired joint velocities. In fact, in the ideal case where $x = x_d$, then $\tilde{x} = 0$ and we have the inverse analytic Jacobian mapping. In the case of using Euler parameters, then $x = \begin{bmatrix} p^T & \xi^T \end{bmatrix}^T$, $x_d = \begin{bmatrix} p_d^T & \xi_d^T \end{bmatrix}^T$, and we can use (Campa et al., 2006):

$$v_d = J(q)^\dagger \begin{bmatrix} \dot{p}_d + K_p \tilde{p} \\ \omega_d + K_o \tilde{\varepsilon} \end{bmatrix} \tag{50}$$

which has a similar structure than (49) but now we employ the geometric Jacobian instead of the analytic one, and the orientation error is taken as $\tilde{\varepsilon}$ (the vector part of the unit quaternion $\tilde{\xi}$), which becomes zero when the actual orientation coincides with the desired one. $K_p, K_o \in \mathbb{R}^{3\times 3}$ are positive definite gain matrices for the position and orientation parts, respectively.
In order to apply the proposed control law it is necessary to assume that the manipulator does not pass through singular configurations —where $J(q)$ losses rank— during the execution of the task. Also it is convenient to assume that the submatrices of $J(q)$, $J_p(q)$ and $J_o(q)$, are bounded, i.e., there exist positive scalars k_{J_p} and k_{J_o} such that, for all $q \in \mathbb{R}^n$ we have

$$\| J_p(q) \| \le k_{J_p}, \quad \text{and} \quad \| J_o(q) \| \le k_{J_o}. \tag{51}$$

Finally, for the joint velocity controller in Fig. 4 we use the same structure proposed in (Aicardi et al., 1995):

$$\tau = M(q)[\dot{v}_d + K_v \tilde{v}] + C(q,\dot{q})\dot{q} + g(q) \tag{52}$$

where $K_v \in \mathbb{R}^{n\times n}$ is a positive definite gain matrix, $\tilde{v} \in \mathbb{R}^n$ is the joint velocity error, given by

$$\tilde{v} = v_d - \dot{q}, \tag{53}$$

and $\dot{v}_d$ can be precomputed from (50) as

$$\dot{v}_d = \dot{J}(q)^\dagger \begin{bmatrix} \dot{p}_d + K_p \tilde{p} \\ \omega_d + K_o \tilde{\varepsilon} \end{bmatrix} + J(q)^\dagger \begin{bmatrix} \ddot{p}_d + K_p \dot{\tilde{p}} \\ \dot{\omega}_d + K_o \left[\frac{1}{2} [\tilde{\eta} I + S(\tilde{\varepsilon})] \tilde{\omega} - S(\tilde{\varepsilon}) \omega_d \right] \end{bmatrix} \tag{54}$$

where we have employed (47). Fig. 6 shows in a detailed block diagram how this controller is implemented.
Substituting the control law (52) in the robot dynamics (38), and considering (53) we get $\dot{\tilde{v}} + K_v\tilde{v} = 0$. On the other hand, replacing (50) in (53), premultiplying by $J(q)$, and considering (27), and (45) we get two decoupled equations

$$\dot{\tilde{p}} + K_p\tilde{p} = J_p(q)\tilde{v} \tag{55}$$

$$\tilde{\omega} + K_o\tilde{\varepsilon} = J_o(q)\tilde{v} \tag{56}$$

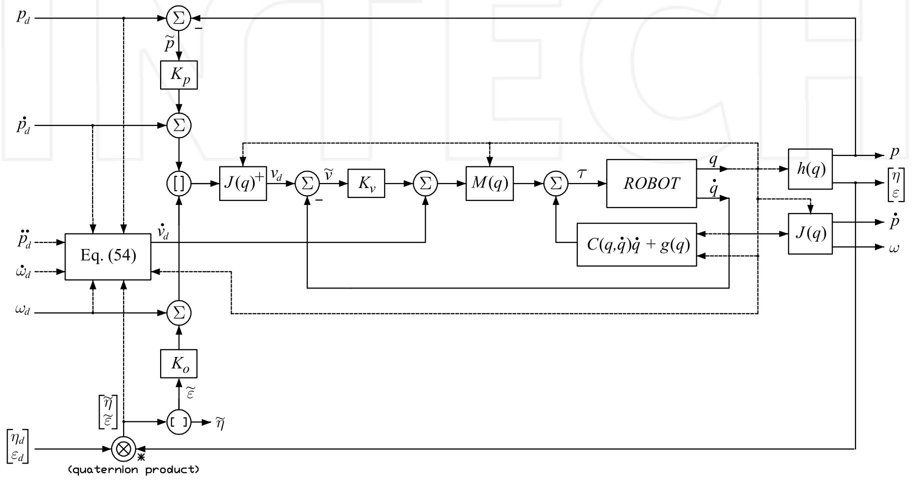

Figure 6. Hierarchical control: RMRC + inverse–dynamics velocity controller

The closed–loop equation of the whole controller can be obtained taking $\tilde{p}$, $\tilde{\eta}$ and $\tilde{\varepsilon}$ as states for the outer (task space) loop and v for the inner loop. The state equation becomes

$$\frac{d}{dt}\begin{bmatrix}\tilde{p}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{v}\end{bmatrix} = \begin{bmatrix} -K_p\tilde{p} + J_p(q)\tilde{v} \\ \frac{1}{2}\tilde{\varepsilon}^T[K_o\tilde{\varepsilon} - J_o(q)\tilde{v}] \\ -\frac{1}{2}[\tilde{\eta}I + S(\tilde{\varepsilon})][K_o\tilde{\varepsilon} - J_o(q)\tilde{v}] - S(\tilde{\varepsilon})\omega_d \\ -K_v\tilde{v}\end{bmatrix}, \tag{57}$$

where (56) has been used to substitute $\tilde{\omega}$ in (47). The domain of the state space, D_{rr}, is defined as follows

$$D_{rr} = \mathbb{R}^3 \times \mathrm{S}^3 \times \mathbb{R}^n = \left\{ \begin{bmatrix}\tilde{p}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{v}\end{bmatrix} \in \mathbb{R}^{7+n} : \tilde{\eta}^2 + \tilde{\varepsilon}^T\tilde{\varepsilon} - 1 = 0 \right\}.$$

System (57) is non–autonomous due to the term $\omega_d(t)$, and it is easy to see that it has two equilibria :

$$\begin{bmatrix}\tilde{p}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{v}\end{bmatrix}=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}=E_1\in D_{rr},\quad \text{and}\quad \begin{bmatrix}\tilde{p}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{v}\end{bmatrix}=\begin{bmatrix}0\\-1\\0\\0\end{bmatrix}=E_2\in D_{rr}.$$

Notice that both equilibria represent the case when the end–effector's pose has reached the desired pose, and the joint velocity error, $\tilde{v}$, is null. However, E_1 is an asymptotically stable equilibrium while E_2 is an unstable equilibrium.
The Lyapunov stability analysis presented below for equilibrium E_1 of system (57) was taken from (Campa et al., 2006). More details can be found in (Campa, 2005).
Let us consider the function:

$$V(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})=\alpha\left[\tilde{\varepsilon}^T\tilde{\varepsilon}+(\tilde{\eta}-1)^2+\frac{1}{2}\tilde{p}^T\tilde{p}\right]+\frac{1}{2}\tilde{v}^T\tilde{v} \tag{58}$$

with α such that

$$0<\alpha<\frac{4\lambda_m\{K_p\}\lambda_m\{K_o\}\lambda_m\{K_v\}}{2\lambda_m\{K_p\}k_{J_o}^2+\lambda_m\{K_o\}k_{J_p}^2}, \tag{59}$$

and k_{Jp}, k_{Jo} satisfying (51).
Notice that $\alpha>0$ is enough to ensure that $V(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})$ is positive definite in D_{rr}, around the equilibrium E_1, that is to say that $V(0,1,0,0)=0$ and there is a neighborhood $B\subset D_{rr}$ around E_1 such that $V(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})>0$, for all $\begin{bmatrix}\tilde{p}^T & \tilde{\eta} & \tilde{\varepsilon}^T & \tilde{v}^T\end{bmatrix}^T\neq\begin{bmatrix}0^T & 1 & 0^T & 0^T\end{bmatrix}^T\in B$.
Now, taking the time derivative of (58) along the trajectories of system (57) we get:

$$\dot{V}(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})=-\alpha\left[\tilde{p}^TK_p\tilde{p}+\varepsilon^TK_o\tilde{\varepsilon}-\tilde{p}^TJ_p(q)\tilde{v}-\tilde{\varepsilon}^TJ_o(q)\tilde{v}\right]-v^TK_v\tilde{v}$$

and it can be shown that it is bounded by

$$\dot{V}(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})\leq-\begin{bmatrix}\|\tilde{p}\|\\ \|\tilde{\varepsilon}\|\\ \|\tilde{v}\|\end{bmatrix}^T Q\begin{bmatrix}\|\tilde{p}\|\\ \|\tilde{\varepsilon}\|\\ \|\tilde{v}\|\end{bmatrix}-\frac{1}{2}\alpha\lambda_m\{K_o\}\left(1-\tilde{\eta}^2\right).$$

where

$$Q=\begin{bmatrix}\alpha\lambda_m\{K_p\} & 0 & -\frac{1}{2}\alpha k_{Jp}\\ 0 & \frac{1}{2}\alpha\lambda_m\{K_o\} & -\frac{1}{2}\alpha k_{Jo}\\ -\frac{1}{2}\alpha k_{Jp} & -\frac{1}{2}\alpha k_{Jo} & \lambda_m\{K_v\}\end{bmatrix}$$

If α satisfies condition (59), then Q is positive definite matrix and $\dot{V}(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})$ is negative definite function in D_{rr}, around E_1, that is, $\dot{V}(0,1,0,0)=0$ and there is a neighborhood $B \subset D_{rr}$ around E_1 such that $\dot{V}(\tilde{p},\tilde{\eta},\tilde{\varepsilon},\tilde{v})<0$, for all $\begin{bmatrix}\tilde{p}^T & \tilde{\eta} & \tilde{\varepsilon}^T & \tilde{v}^T\end{bmatrix} \neq \begin{bmatrix}0^T & 1 & 0^T & 0^T\end{bmatrix}^T \in B$. Thus, according to classical Lyapunov theory (see e.g. (Khalil, 2002)), we can conclude that the equilibrium E_1 is asymptotically stable in D_{rr}. This implies that, starting from an initial condition near E_1 in D_{rr}, the control objective (48) is fulfilled.

5.3 Resolved Acceleration Controller

The resolved acceleration control (RAC) is a well–known technique proposed by Luh et al. (1980) to solve the problem of pose space control of robot arms. It is based on the inverse dynamics methodology (Spong et al., 2006), and assumes that the Jacobian is full rank, in order to get a closed–loop system which is *almost* linear. The original RAC scheme used a parameterization of orientation known as the *Euler rotation*, but some other parameterizations have also been studied later (see Caccavale et al., 1998). The number of RAC closed–loop equilibria and their stability properties depend on the parameterization used for describing orientation.
The general expression for the RAC control law when using Euler parameters is given by

$$\tau = M(q)J^{\dagger}(q)\left[\begin{bmatrix}\ddot{p}_d + K_{V_p}\dot{\tilde{p}} + K_{P_p}\tilde{p} \\ \omega_d + K_{V_o}\tilde{\omega} + K_{P_o}\tilde{\varepsilon}\end{bmatrix} - \dot{J}(q)\dot{q}\right] + C(q,\dot{q})\dot{q} + g(q) \tag{60}$$

where $M(q)$, $C(q,\dot{q})$, $g(q)$ are elements of the joint dynamics (38); $J^{\dagger}(q)$ denotes the left pseudo–inverse of $J(q)$, and K_{P_p}, K_{V_p}, K_{P_o} are symmetric positive definite matrices, $K_{V_o} = k_v I$ is diagonal with $k_v > 0$. As in the RMRC controller in the previous subsection, we use $\tilde{\varepsilon}$ as an orientation error vector. Fig. 7 shows how to implement RAC's control law.
Substituting the control law (60) into the robot dynamics (38), considering that $J(q)$ is full rank, and the time derivative of (27) is given by

$$\begin{bmatrix}\dot{p} \\ \omega\end{bmatrix} = J(q)\ddot{q} + \dot{J}(q)\dot{q},$$

we get the (almost linear) system:

$$\begin{bmatrix}\ddot{\tilde{p}} + K_{V_p}\dot{\tilde{p}} + K_{P_p}\tilde{p} \\ \tilde{\omega} + K_{V_o}\tilde{\omega} + K_{P_o}\tilde{\varepsilon}\end{bmatrix} = 0 \in \mathbb{R}^6. \tag{61}$$

To obtain the full state–space representation of the RAC closed–loop system, we use (61) and include dynamics (47), to get

$$\frac{d}{dt}\begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{\omega}\end{bmatrix}=\begin{bmatrix}\dot{\tilde{p}}\\ -K_{V_p}\dot{\tilde{p}}-K_{P_p}\tilde{p}\\ -\frac{1}{2}\tilde{\varepsilon}^T\tilde{\omega}\\ \frac{1}{2}[\tilde{\eta}I+S(\tilde{\varepsilon})]\tilde{\omega}-S(\tilde{\varepsilon})\omega_d\\ -K_{V_o}\tilde{\omega}-K_{P_o}\tilde{\varepsilon}\end{bmatrix} \tag{62}$$

where now the domain of the state–space is

$$D_{ra}=\mathbb{R}^6\times\mathrm{S}^3\times\mathbb{R}^3=\left\{\begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{\eta}\\ \dot{\tilde{\varepsilon}}\\ \tilde{\omega}\end{bmatrix}\in\mathbb{R}^{13}:\tilde{\eta}^2+\tilde{\varepsilon}^T\tilde{\varepsilon}-1=0\right\}.$$

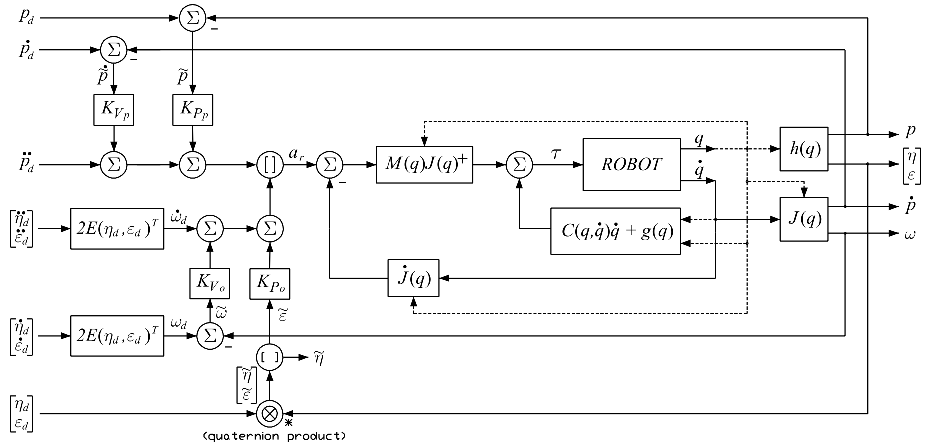
p_d
$\dot{p}_d$
$\ddot{p}_d$
$\dot{\tilde{p}}$
$\tilde{p}$
K_{V_p}
K_{P_p}
a_r
$M(q)J(q)^+$
τ
ROBOT
q
$\dot{q}$
$h(q)$
$J(q)$
p
$\begin{bmatrix}\eta\\ \varepsilon\end{bmatrix}$
$\dot{p}$
ω
$C(q,\dot{q})\dot{q}+g(q)$
$\dot{J}(q)$
$\begin{bmatrix}\ddot{\eta}_d\\ \ddot{\varepsilon}_d\end{bmatrix}$
$2E(\eta_d,\varepsilon_d)^T$
$\dot{\omega}_d$
K_{V_o}
K_{P_o}
$\tilde{\omega}$
$\tilde{\varepsilon}$
$\begin{bmatrix}\dot{\eta}_d\\ \dot{\varepsilon}_d\end{bmatrix}$
ω_d
$\tilde{\eta}$
$\begin{bmatrix}\tilde{\eta}\\ \tilde{\varepsilon}\end{bmatrix}$
$\begin{bmatrix}\eta_d\\ \varepsilon_d\end{bmatrix}$
(quaternion product)

Figure 7. Resolved acceleration controller with Euler parameters

The closed–loop system (62) is nonautonomous, because of the presence of $\omega_d(t)$, and it has two equilibria:

$$\begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{\omega}\end{bmatrix}=\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}=E_1\in D_{ra},\quad\text{and}\qquad\begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{\eta}\\ \tilde{\varepsilon}\\ \tilde{\omega}\end{bmatrix}=\begin{bmatrix}0\\0\\-1\\0\\0\end{bmatrix}=E_2\in D_{ra}.$$

As shown in (Campa, 2005) these two equilibria have different stability properties (E_1 is asymptotically stable, E_2 is unstable). Notwithstanding, they represent the same pose.

In order to prove asymptotic stability of the equilibrium E_1, let us consider the following Lyapunov function (Campa, 2005):

$$V(\tilde{p},\dot{\tilde{p}},\tilde{\eta},\tilde{\varepsilon},\tilde{\omega}) = V_p(\tilde{p},\dot{\tilde{p}}) + V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$$

where $V_p(\tilde{p},\dot{\tilde{p}})$ and $V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$ stand for the position and orientation parts, given by

$$V_p(\tilde{p},\dot{\tilde{p}}) = \frac{1}{2}\dot{\tilde{p}}^T\dot{\tilde{p}} + \frac{1}{2}\tilde{p}^T\left[K_{P_p} + \alpha K_{V_p}\right]\tilde{p} + \alpha\tilde{p}^T\dot{\tilde{p}},$$

$$V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega}) = (\tilde{\eta}-1)^2 + \tilde{\varepsilon}^T\tilde{\varepsilon} + \frac{1}{2}\tilde{\omega}^T K_{P_o}^{-1}\tilde{\omega} + \beta\tilde{\varepsilon}^T\tilde{\omega},$$

and α, β, satisfy

$$0 < \alpha < \lambda_m\{K_{V_p}\} \tag{63}$$

$$0 < \beta < \min\left\{\sqrt{2\lambda_m\left\{K_{P_o}^{-1}\right\}}, \frac{2k_v\lambda_m\{K_{P_o}\}\lambda_m\{K_{P_o}^{-1}\}}{\lambda_m\{K_{P_o}\} + \left[k_v + \|\omega_d\|_{\mathrm{M}}\right]^2}\right\}. \tag{64}$$

where $\|\omega_d\|_{\mathrm{M}}$ stands for the supremum of the norm $\|\omega_d\|$.

Let us notice that the position part of system is a second–order linear system, and it is easily proved that this part, considering (63), is asymptotically stable (see, e.g., the proof of the *computed–torque* controller in (Kelly et al., 2005)). Regarding the orientation part, it should be noticed that

$$V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega}) \geq (\tilde{\eta}-1)^2 + \frac{1}{2}\begin{bmatrix}\|\tilde{\varepsilon}\| \\ \|\tilde{\omega}\|\end{bmatrix}^T\begin{bmatrix}2 & -\beta \\ -\beta & \lambda_m\{K_{P_o}^{-1}\}\end{bmatrix}\begin{bmatrix}\|\tilde{\varepsilon}\| \\ \|\tilde{\omega}\|\end{bmatrix}.$$

Thus, given (64), we have that $V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$ is a positive definite function in $\mathrm{S}^3\times\mathbb{R}^3$, around E_1, that is to say that $V_o(1,0,0)=0$, and there is a neighborhood $B\subset\mathrm{S}^3\times\mathbb{R}^3\subset D_{ra}$, around E_1, such that $V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})>0$, for all $\begin{bmatrix}\tilde{\eta} & \tilde{\varepsilon}^T & \tilde{\omega}^T\end{bmatrix}^T \neq \begin{bmatrix}1 & 0^T & 0^T\end{bmatrix}^T \in B$.

The time derivative of $V_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$ along the trajectories of system (62) results in

$$\dot{V}_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega}) = -\beta\tilde{\varepsilon}^T K_{P_o}\tilde{\varepsilon} - \tilde{\omega}^T\left[K_{P_o}^{-1}K_{V_o} - \frac{1}{2}\beta\tilde{\eta}I\right]\tilde{\omega} - \beta\tilde{\varepsilon}^T\left[K_{V_o} + S(\omega_d)\right]\tilde{\omega},$$

where properties (4) to (7) on $S(\tilde{\varepsilon})$ have been evoked. Hence, we have

$$\dot{V}_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega}) \leq -\frac{1}{2}\begin{bmatrix}\|\tilde{\varepsilon}\| \\ \|\tilde{\omega}\|\end{bmatrix}^T Q\begin{bmatrix}\|\tilde{\varepsilon}\| \\ \|\tilde{\omega}\|\end{bmatrix} - \frac{1}{2}\beta\lambda_m\{K_{P_o}\}\left(1-\tilde{\eta}^2\right),$$

where

$$Q = \begin{bmatrix} \beta\lambda_m\{K_{P_o}\} & -\beta\left[k_v + \|\omega_d\|_{\mathrm{M}}\right] \\ -\beta\left[k_v + \|\omega_d\|_{\mathrm{M}}\right] & 2k_v\lambda_m\{K_{P_o}^{-1}\} - \beta \end{bmatrix},$$

the property $\|S(x)\| = \|x\|$ has ben considered, and the fact $\|\varepsilon\|^2 = \left(1-\eta^2\right)$ was employed to get the last term.
A condition for $\dot{V}_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$ to be negative definite in $\mathrm{S}^3\times\mathbb{R}^3$ is that $Q>0$, which is true if β satisfies (5). Thus, $\dot{V}_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})$ is negative definite in $\mathrm{S}^3\times\mathbb{R}^3$, around E_1, that is to say that $\dot{V}_o(1,0,0)=0$, and there is a neighborhood $B\subset \mathrm{S}^3\times\mathbb{R}^3 \subset D_{ra}$, around E_1, such that $\dot{V}_o(\tilde{\eta},\tilde{\varepsilon},\tilde{\omega})<0$, for all $\left[\tilde{\eta} \quad \tilde{\varepsilon}^T \quad \tilde{\omega}^T\right]^T \neq \left[1 \quad 0^T \quad 0^T\right]^T \in B$.
Thus, we can conclude that the equilibrium E_1 is asymptotically stable in D_{ra}. This implies that, starting from an initial condition near E_1 in D_{ra}, the control objective (48) is fulfilled.

6. Conclusion

Among the different parameterizations of the orientation manifold, Euler parameters are of interest because they are unit quaternions, and form a Lie group. This chapter was intended to recall some of the properties of the quaternion algebra, and to show the usage of Euler parameters for modeling, path planning and control of robot manipulators.
For many years, the Denavit–Hartenberg parameters have been employed for obtaining the direct kinematic model of robotic mechanisms. Instead of using homogeneous matrices for handling the D-H parameters, we proposed an algorithm that uses the quaternion multiplication as basis, thus being computationally more efficient than matrix multiplication. In addition, we sketched a general methodology for obtaining the dynamic model of serial manipulators when Euler parameters are employed. The so–called spherical linear interpolation (or slerp) is a method for interpolating curves in the orientation manifold. We reviewed such an algorithm, and proposed its application for motion planning in robotics. Finally, we studied the stability of two task space controllers that employ unit quaternions as state variables; such analysis allows us to conclude that both controllers are suitable for motion control tasks in pose space.
As a conclusion, even though Euler parameters are still not very known, it is our believe that they constitute a useful tool, not only in the robotics field, but wherever the description of objects in 3-D space is required. More research should be done in these topics.

7. Acknowledgement

This work was partially supported by DGEST and CONACYT (grant 60230), Mexico.

8. References

Aicardi, M.; Caiti, A.; Cannata, G. & Casalino, G. (1995). Stability and robustness analysis of a two layered hierarchical architecture for the closed loop control of robots in the operational space, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2771–2778, Nagoya, Japan, May 1995.

Barrientos, A.; Peñín, L. F.; Balaguer, C. & Aracil, R. (1997). *Fundamentos de Robótica*, McGraw-Hill, Madrid.

Caccavale, F.; Natale, C.; Siciliano, B. & Villani, L. (1998) Resolved–acceleration control of robot manipulators: A critical review with experiments. *Robotica*, Vol. 16, pp. 565–573.

Caccavale, F.; Siciliano, B. & Villani, L. (1999). The role of Euler parameters in robot control, *Asian Journal of Control*, Vol. 1, pp. 25–34.

Campa, R. (2005). *Control de Robots Manipuladores en Espacio de Tarea*. Doctoral thesis (in Spanish), CICESE Research Center, Ensenada, Mexico, March 2005.

Campa, R. & de la Torre, H. (2006). On the representations of orientation for pose control tasks in robotics. *Proceedings of the VIII Mexican Congress on Robotics* Mexico, D.F., October 2006.

Campa, R.; Camarillo, K. & Arias, L. (2006). Kinematic modeling and control of robot manipulators via unit quaternions: Application to a spherical wrist. *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 2006.

Craig, J. J. (2005) *Introduction to Robotics: Mechanics and Control*, Pearson Prentice–Hall, USA, 2005.

Dam, E. B.; Koch, M. & Lillholm, M. (1998) *Quaternions, Interpolation and Animation*. Technical report DIKU-TR-98/5, University of Copenhagen, Denmark.

Denavit, J. & Hartenberg, E. (1955) A kinematic notation for lower–pair mechanisms based on matrices. *Journal of Applied Mechanics*, Vol. 77, pp. 215–221.

Fjellstad, O. E. (1994). "Control of unmanned underwater vehicles in six degrees of freedom: a quaternion feedback approach", Dr.ing. thesis, Norwegian Institute of Technology, Norway.

Gallier, J. (2000). *Geometric Methods and Applications for Computer Science and Engineering*. Springer–Verlag, New York.

Hamilton, W. R. (1844). On a new species of imaginary quantities connected with a theory of quaternions. *Proceedings of the Royal Irish Academy*. Vol. 2, pp. 424-434. (http://www.maths.tcd.ie/pub/HistMath/People/Hamilton/Quatern1).

Kelly, R. & Moreno–Valenzuela, J. (2005). Manipulator Motion Control in Operational Space Using Joint Velocity Inner Loop, *Automatica*, Vol. 41, pp. 1423-1432.

Kelly, R.; Santibáñez, V. & Loría, A. (2005) *Control of Robot Manipulators in Joint Space*, Springer-Verlag, London.

Khalil, H. K. (2002) *Nonlinear Systems*, Prentice–Hall, New Jersey..

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*. Vol. 3, pp. 43–53.

Krause, P. C.; Wasynczuk, O. & Sudhoff S. D. (2002). *Analysis of Electric Machinery and Drive Systems*. Wiley-Interscience.

Kuipers, J. B. (1999). *Quaternions and rotation sequences*. Princeton University Press. Princeton, NJ.

Lin, S. K. (1995) Robot control in Cartesian space, In: *Progress in Robotics and Intelligent Systems*, Zobrit, G. W. & Ho, C. Y. (Ed.), pp. 85–124, Ablex, New Jersey.

Lu, Y. J. & Ren G. X. (2006) A symplectic algorithm for dynamics of rigid–body. *Applied Mathematics and Dynamics*, Vol. 27, No. 1, pp. 51-57.

Luh, J. Y. S., Walker M. W., Paul R. P. C. (1980). Resolved–acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*. Vol. 25, pp. 486–474.

Natale, C. (2003). *Interaction Control of Robot Manipulators: Six degrees–of–freedom Tasks*, Springer, Germany.

Rooney, J. & Tanev, T. K. (2003). Contortion and formation structures in the mappings between robotic jointspaces and workspaces. *Journal of Robotic Systems*, Vol. 20, No. 7, pp. 341–353.

Sciavicco, L. & Siciliano, B. (2000). *Modelling and Control of Robot Manipulators*, Springer–Verlag, London.

Shivarama, R. & Fahrenthold, E. P. (2004) Hamilton's equation with Euler parameters for rigid body dynamics modeling *Journal of Dynamics Systems, Measurement and Control* Vol. 126, pp. 124-130.

Shoemake, K. (1985) Animating rotation with quaternion curves, *ACM SIGGRAPH Computer Graphics* Vol. 19, No. 3, pp. 245-254.

Spong, M. W.; Hutchinson, S. & Vidyasagar, M. (2006) *Robot Modeling and Control*, John Wiley and Sons, USA, 2006.

Spring, K. W. (1986). Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: A review. *Mechanism and Machine Theory*, Vol. 21, No. 5, pp. 365-373.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, Vol. 10, no. 2, pp. 47-53.

Xian, B.; de Queiroz, M. S.; Dawson, D. & Walker, I. (2004). Task–space tracking control of robot manipulators via quaternion feedback. *IEEE Transactions on Robotics and Automation*. Vol. 20, pp. 160–167.

Yuan, J. S. C. (1988). Closed-loop manipulator control using quaternion feedback. *IEEE Journal of Robotics and Automation*. Vol. 4, pp. 434–440.