JACOBIAN-BASED ALGORITHMS:
A BRIDGE BETWEEN KINEMATICS
AND CONTROL

The PRISMA Lab
www.prisma.unina.it

Bruno Siciliano
Dipartimento di Ingegneria dell’Informazione e Ingegneria Elettrica
Università di Salerno
Via Ponte Don Melillo, 84084 Fisciano (SA), Italy
bsiciliano@unisa.it

Lorenzo Sciavicco
Dipartimento di Informatica e Automazione
Università di Roma Tre
Via della Vasca Navale 79, 00146 Roma, Italy
sciavicco@dia.uniroma3.it

Stefano Chiaverini
Dipartimento di Automazione, Elettromagnetismo,
Ingegneria dell’Informazione e Matematica Industriale
Università di Cassino
Via G. Di Biasio 43, 03043 Cassino (FR), Italy
chiaverini@unicas.it

Pasquale Chiacchio, Luigi Villani
Dipartimento di Informatica e Sistemistica
Università di Napoli Federico II
Via Claudio 21, 80125 Napoli, Italy
pasquale.chiacchio|luigi.villani@unina.it

Fabrizio Caccavale
Dipartimento di Ingegneria e Fisica dell’Ambiente
Università della Basilicata
Contrada Macchia Romana, 85100 Potenza, Italy
caccavale@unibas.it
Abstract  Whenever it comes to the point of controlling complex robotic systems, all goes back to fundamental kinematic issues which should be thoroughly mastered to design effective control systems. This has been the experience of many scholars working in the field whose research has been influenced, directly or indirectly, by the seminal work of bright designers and kinematicians like Professor Bernard Roth. The present paper aims at surveying a body of literature on Jacobian-based algorithms developed by the PRISMA Lab, e.g. a group of researchers with control background who have constantly found inspiration in the fascinating world of kinematics.

Keywords: Robot manipulators, kinematics, Jacobian, kinematic control, serial manipulators, redundant manipulators, parallel manipulators, cooperative multiarm systems, underwater vehicle-manipulator systems, spacecraft-manipulator systems, flexible manipulators

1. Kinematic Control

No doubt, the Jacobian is the most important tool for characterizing articulated mechanical systems, such as robot manipulators. It is useful for finding singular configuration, analyzing redundancy, developing inverse kinematics algorithms, describing the statics, deriving dynamic equations of motion and, last but not least, designing control schemes.

Robot kinematic control consists of solving the motion control problem into two stages, i.e. the desired end-effector trajectory is transformed via inverse kinematics into the corresponding joint trajectories, which then constitute the reference inputs to some joint space control scheme [Siciliano, 1990a]. This approach differs from operational space control [Khatib, 1987] in the sense that manipulator kinematics is handled outside the control loop thus allowing the problem of kinematic singularities and/or redundancy to be solved separately from the motion control problem. The key point of kinematic control is the solution to the inverse kinematics problem.

In all general terms, the kinematics of a robotic systems is expressed in terms of a nonlinear relationship between the vector \( q \) of configuration variables and the vector \( x \) of task variables, i.e.

\[
x = k(q).
\]  

Well-assessed methods exist to calculate the function \( k \) in a closed form [Bottema and Roth, 1979]. On the other hand, finding solutions to (1) has attracted a wide number of kinematicians who have spent considerable efforts on investigating the existence of admissible solutions, e.g. [Pieper and Roth, 1969] and, in the affirmative case, computing all possible solutions [Raghavan and Roth, 1995].
An effective numerical approach to solve (1) can be pursued by resorting to the differential mapping

\[ \dot{x} = J_A(q) \dot{q}, \]  

(2)

where \( J_A(q) = \frac{\partial k}{\partial q} \) is the robot analytical Jacobian. The term “analytical” merely reflects the analytical derivation of the Jacobian, as opposed to the “geometric” Jacobian which would relate configuration velocities to task velocities [Sciavicco and Siciliano, 2000]; this issue is dealt with in further detail in the next section.

The main advantage of (2) over (1) is its linearity in the configuration velocities. This allows solving the differential kinematics by a suitable inversion of the Jacobian matrix to compute [Whitney, 1969]

\[ \dot{q} = J_A^{-1}(q) \dot{x}, \]  

(3)

which then can be integrated over time to give \( q \). Of course, if \( J_A \) is square matrix, a simple inverse suffices; yet, if \( J_A \) is not full-rank, numerical robustness could be achieved by resorting to a damped least-squares inverse [Chiaverini et al, 1994], where the damping factor can be suitably tuned as a function of proximity to singularities, e.g. on the basis of the estimates of the smallest singular values of the Jacobian matrix [Chiaverini, 1993].

On the other hand, if the Jacobian is a non-square matrix, then a suitable right pseudo-inverse can be adopted, i.e.

\[ \dot{q} = J_A^\dagger(q) \dot{x}. \]  

(4)

From an algorithmic viewpoint, it is advisable to introduce a feedback correction term aimed at eliminating any numerical drift of the solution, i.e.

\[ \dot{q} = J_A^\dagger(q)(\dot{x}_d + Ke), \]  

(5)

where \( e = x_d - x \) is the error between the desired and the actual set of task variables, and \( K \) is a suitable positive definite matrix gain. The resulting scheme of the Jacobian inverse kinematics algorithm is illustrated in Fig. 1. This scheme solves the so-called kinematic control problem, i.e. to find suitable joint trajectories \( q(t) \) corresponding to a desired task trajectory \( x_d(t) \).

An effective alternative to the above algorithm based on (5) can be adopted by resorting to a Jacobian transpose in lieu of its inverse, i.e. [Sciavicco and Siciliano, 1986]

\[ \dot{q} = J_A^T(q)Ke, \]  

(6)
The Jacobian Inverse Kinematics Algorithm can be proved to guarantee limited tracking errors and null steady-state errors [Chiacchio and Siciliano, 1989]. The resulting scheme is illustrated in Fig. 2.

The Jacobian Transpose Kinematics Algorithm

The Jacobian-based algorithms hereby formulated can be used to solve a number of kinematic control problems, as illustrated in the following sections; namely, for serial manipulators, redundant manipulators, parallel manipulators, cooperative multiarm systems, underwater vehicle-manipulator systems, spacecraft-manipulator systems, and flexible manipulators.

2. Serial Manipulators

The kinematics of a serial open-chain manipulator can be described in the form (1) where \( q \) is the set of joint variables, and \( x_e \) is the set of end-effector variables

\[
x_e = \begin{bmatrix} p_e \\ \varphi_e \end{bmatrix},
\]

where \( p_e \) denotes the end-effector position and \( \varphi_e \) denotes a minimal representation of orientation in terms of three Euler angles, with respect to some base frame (Fig. 3).
Notice that the vector $x_e$ is defined in the task space, i.e. in which the robot task is specified. The dimension of this space is at most $m = 6$, since 3 coordinates specify position and 3 coordinates specify orientation. Nevertheless, depending on the geometry of the task, a reduced number of task space variables may be specified; for instance, for a planar robot it is $m = 3$, since two coordinates specify position and one coordinate specifies orientation.

This representation of position and orientation allows the description of the end-effector task in terms of a number of inherently independent parameters. Nevertheless, the computation of Euler angles from the joint variables goes through the computation of the end-effector frame rotation matrix

$$R_e = \begin{bmatrix} n_e & s_e & a_e \end{bmatrix}$$  \hspace{1cm} (8)

and, as such, it suffers from the so-called representation singularities. These clearly appear in the mapping between the time derivatives of the Euler angles and the end-effector angular velocity, i.e.

$$\omega_e = T(\varphi_e)\dot{\varphi}_e,$$  \hspace{1cm} (9)

where $T(\varphi_e)$ depends on the particular choice of Euler angles.

In particular, the angular velocity $\omega_e$ is related to the time derivative of the rotation matrix as

$$\dot{R}_e = S(\omega_e)R_e,$$  \hspace{1cm} (10)
where $S(\cdot)$ is the matrix operator performing the cross product between two $(3 \times 1)$ vectors.

Therefore, the differential kinematics equation for a serial manipulator is established in the form

$$\mathbf{v} = \begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = J(q) \dot{q},$$

(11)

where $J(q)$ is the geometric Jacobian. The computation of this matrix usually follows a geometric procedure that is based on computing the contributions of each joint velocity to the linear and angular end-effector velocities [Whitney, 1972].

It can be easily recognized that the geometric Jacobian in general differs from the analytical Jacobian. Concerning their use, the geometric Jacobian is adopted when physical quantities are of interest while the analytical Jacobian is adopted when task space quantities are of interest. It is always possible to pass from one Jacobian to the other, except at a representation singularity.

The algorithm based on (4) for a serial manipulator constitutes an inverse kinematics algorithm which makes use of the analytical Jacobian. More insight about the implications of different end-effector orientation descriptions can be gained by separating the position from the orientation components. Using the geometric Jacobian in lieu of the analytical Jacobian, the solution can be rewritten as

$$\dot{q} = J^T(q) \mathbf{v}$$

(12)

with

$$\mathbf{v} = \begin{bmatrix} v_p \\ v_o \end{bmatrix},$$

(13)

where $v_p, v_o$ represent two resolved velocities that shall be chosen so as to ensure tracking of the desired end-effector motion.

For what concerns position, the choice is rather straightforward, i.e.

$$v_p = \dot{p}_d + K_p e_p,$$

(14)

where the position error

$$e_p = p_d - p_e(q)$$

(15)

between the desired and actual end-effector positions has been defined.

On the other hand, for what concerns the orientation error, some considerations are in order depending on the type of description adopted. If Euler angles are adopted, the resolved angular velocity in (13) is chosen as

$$v_o = T(\varphi_e)(\dot{\varphi}_d + K_o e_{o,Eul}),$$

(16)
where
\[ e_{o,Eul} = \varphi_d - \varphi_e(q) \]  
(17)
is the orientation error.

In order to overcome the drawback of representation singularities in (16), an algorithm based on an alternative Euler angles description can be conceived [Caccavale et al., 1998] which makes use of the rotation matrix describing the mutual orientation between the desired and the actual end-effector frame, i.e.
\[ eR_d = R_e^T(q)R_d. \]  
(18)
Differentiating (18) with respect to time and accounting for (10) gives
\[ e\dot{R}_d = S(e\omega_{de})eR_d, \]  
(19)
where \( \omega_{de} = \omega_d - \omega_e(q) \) is the end-effector angular velocity error.

Let \( \varphi_{de} \) denote the set of Euler angles that can be extracted from \( eR_d \). Then, in view of (9) and (10), the angular velocity \( e\omega_{de} \) in (19) is related to the time derivative of \( \varphi_{de} \) as
\[ e\omega_{de} = T(\varphi_{de})\dot{\varphi}_{de}. \]  
(20)
At this point, the resolved angular velocity in (13) can be chosen as
\[ v_o = \omega_d + R_eT(\varphi_{de})K_o e_{o,EulAlt} \]  
(21)
with
\[ e_{o,EulAlt} = \varphi_{de}. \]  
(22)

The clear advantage of the alternative over the classical Euler angles algorithm based on (16) is that, by adopting a representation \( \varphi_{de} \) for which \( T(0) \) is nonsingular, representation singularities occur only for large orientation errors. In other words, the ill-conditioning of matrix \( T \) is not influenced by the desired or actual end-effector orientation but only by the orientation error. In this respect, the choice of a particular Euler angles description among the twelve possible should be carefully made, i.e. in the sense of avoiding a representation singularity.

In order to overcome the problem of representation singularities, an inverse kinematics algorithm based on the angle/axis description of orientation can be devised. From (18), the rotation \( \theta_{de} \) and the unit vector \( r_{de} \) can be extracted using well-known formulae. Then, the orientation error can be defined as
\[ e_{o,AnAx} = \sin \theta_{de} r_{de}. \]  
(23)
Notice that (23) gives a unique solution for $-\pi/2 < \vartheta_{de} < \pi/2$, but this interval is not limiting for a convergent inverse kinematics algorithm. It can be shown that a computational expression of the orientation error in (23) is given by [Sciavicco and Siciliano, 2000]

$$e_{o,AnAx} = \frac{1}{2} \left( S(n_e(q))n_d + S(s_e(q))s_d + S(a_e(q))a_d \right), \quad (24)$$

where the triplet of unit vectors has been used for both the desired and the actual end-effector rotation matrix. Note that the above limitation on $\vartheta_{de}$ sets the conditions $n^T_e n_d \geq 0$, $s^T_e s_d \geq 0$, $a^T_e a_d \geq 0$.

Differentiation of (24) with respect to time gives

$$\dot{e}_{o,AnAx} = L^T \omega_d - L \omega_e \quad (25)$$

with

$$L = -\frac{1}{2} \left( S(n_d)S(n_e) + S(s_d)S(s_e) + S(a_d)S(a_e) \right). \quad (26)$$

At this point, the resolved angular velocity in (13) can be chosen as

$$v_o = L^{-1} (L^T \omega_d + K_o e_{o,AnAx}). \quad (27)$$

Another inverse kinematics algorithm is based on the unit quaternion description of orientation, so as to overcome the above limitation on the angle/axis description. Let $Q_d = \{\eta_d, \epsilon_d\}$ and $Q_e = \{\eta_e, \epsilon_e\}$ represent the unit quaternions associated with $R_d$ and $R_e$, respectively. The mutual orientation can be expressed in terms of the unit quaternion $Q_{de} = \{\eta_{de}, \epsilon_{de}\}$ where

$$\eta_{de} = \eta_e(q)\eta_d + \epsilon_e^T(q)\epsilon_d \quad (28)$$

$$\epsilon_{de} = \eta_e(q)\epsilon_d - \eta_d\epsilon_e(q) - S(\epsilon_d)\epsilon_e(q).$$

It can be recognized that $Q_{de} = \{1, 0\}$ if and only if $R_e$ and $R_d$ are aligned, and thus it is sufficient to consider $\epsilon_{de}$ to express an end-effector orientation error, i.e. [Chiaverini and Siciliano, 1999a]

$$e_{o,Quat} = \epsilon_{de}. \quad (29)$$

Notice that the explicit computation of $\eta_e(q)$ and $\epsilon_e(q)$ is not possible, but it requires the intermediate computation of the rotation matrix $R_e(q)$ that is available from the manipulator direct kinematics; then, the unit quaternion can be extracted using well-known formulae, e.g.
At this point, the resolved angular velocity in (13) can be chosen as

\[ v_o = \omega_d + K_o e_o \text{Quat}, \]  

which can be shown to ensure null tracking errors.

As a final remark, the above kinematic control schemes can be easily extended to the second order, i.e. solving the joint motion at the acceleration level instead of velocity level. This may be advantageous for model-based dynamic control in the joint space [Caccavale et al, 1997].

### 3. Redundant Manipulators

Whenever a manipulator possesses more joint variables than the number of task variables, it is said to be kinematically redundant.

If it is desired to exploit redundant degrees of freedom, solution (5) can be generalized to

\[ \dot{q} = J_A^T(q) (\dot{x}_d + K e) + \left( I - J_A^T(q) J_A(q) \right) \dot{q}_0, \]  

where the matrix \((I - J_A^T J)\) is a projector of the joint vector \(\dot{q}_0\) onto the null space of \(J\) [Chiacchio et al, 1991].

This result is of fundamental importance for redundancy resolution, since solution (31) evidences the possibility of choosing the vector \(\dot{q}_0\) so as to exploit the redundant degrees of freedom. In fact, the contribution of \(\dot{q}_0\) is to generate null space motions of the structure that do not contribute to the end-effector motion but allow the manipulator to reach configurations which are more dexterous for the execution of the given task [Siciliano, 1990b]. In case of numerical problems in the neighborhood of singularities, the pseudoinverse can be replaced with a suitable damped least-squares inverse [Chiaverini, 1997].

An effective way to choosing \(\dot{q}_0\) can be derived by combining the Jacobian pseudoinverse solution with the Jacobian transpose solution as illustrated below. This is carried out in the framework of the so-called augmented task space approach to exploit redundancy in robotic systems [Sciavicco and Siciliano, 1988]. The idea is to introduce an additional constraint task by specifying a vector \(x_C\) as a function of the joint variables, i.e.

\[ x_C = k_C(q), \]  

so as to constrain at most all the available redundant degrees of freedom. The constraint task vector \(x_C\) can be chosen by embedding scalar objective functions, e.g. to improve dexterity, avoid obstacles, etc.

Differentiating (32) with respect to time gives

\[ \dot{x}_C = J_C(q) \dot{q}, \]
where \( J_C(q) = \frac{\partial k_C}{\partial q} \) is the constraint Jacobian. The result is an augmented differential kinematics equation given by (2) and (33), based on a Jacobian matrix

\[
J' = \begin{bmatrix}
J_A \\
J_C
\end{bmatrix}.
\]  

(34)

When a constraint task is specified independently of the end-effector task, there is no guarantee that the matrix \( J' \) remains full-rank along the entire task path; singularities of \( J' \) are termed \textit{artificial singularities} and it can be shown that those are given by singularities of the matrix \( J_C(\mathbf{I} - J_A^\dagger J_A) \).

The above discussion suggests that, when solving for joint velocities, a \textit{task priority strategy} is advisable so as to avoid conflicting situations between the end-effector task and the constraint task [Nakamura et al., 1987]. Substituting (31) into (33) gives

\[
\dot{x}_C = J_C(q)J_A^\dagger(q)(\dot{x}_d + Ke) + J_C(q)\left(\mathbf{I} - J_A^\dagger(q)J_A(q)\right)\dot{q}_0
\]  

(35)

which could be solved for \( \dot{q}_0 \) provided that artificial singularities are avoided. Observing that equality (35) can be achieved only for the components of \( \dot{x}_C \) belonging to the range space of \( J_C \), it is sufficient to consider the equation

\[
J_C^\dagger(q)\dot{x}_C = J_A^\dagger(q)(\dot{x}_d + Ke) + \left(\mathbf{I} - J_A^\dagger(q)J_A(q)\right)\dot{q}_0
\]  

(36)

that can be solved for \( \dot{q}_0 \) giving

\[
\dot{q}_0 = \left(\mathbf{I} - J_A^\dagger(q)J_A(q)\right)^\dagger \left(J_C^\dagger(q)\dot{x}_C - J_A^\dagger(q)(\dot{x}_d + Ke)\right).
\]  

(37)

By recalling that \( (\mathbf{I} - J_A^\dagger J_A)^\dagger = (\mathbf{I} - J_A^\dagger J_A) \), solution (37) reduces to the simple form [Chiaverini, 1997]

\[
\dot{q}_0 = \left(\mathbf{I} - J_A^\dagger(q)J_A(q)\right)J_C^\dagger(q)\dot{x}_C.
\]  

(38)

Folding (38) back into (31) and exploiting the idempotence of \( (\mathbf{I} - J_A^\dagger J_A) \) gives

\[
\dot{q} = J_A^\dagger(q)(\dot{x}_d + Ke) + \left(\mathbf{I} - J_A^\dagger(q)J_A(q)\right)J_C^\dagger(q)(\dot{x}_{Cd} + K_C e_C)
\]  

(39)

with \( e_C = x_{Cd} - x_C \), being \( x_{Cd} \) the desired value of the constraint task, and \( K_C \) is a positive definite matrix. The operator \( (\mathbf{I} - J_A^\dagger J_A) \) projects the secondary velocity contribution \( \dot{q}_0 \) on the null space of \( J_A \), guaranteeing correct execution of the primary end-effector task while
the secondary constraint task is correctly executed as long as it does not interfere with the end-effector task. Obviously, if desired, the order of priority can be switched, e.g. in an obstacle avoidance task when an obstacle comes to be along the end-effector path.

In the case when $J_C$ becomes singular, a damped least-squares inverse of $J_C$ in lieu of the pseudoinverse in (38) can be used. Otherwise, by recalling the Jacobian transpose solution for the end-effector task (6), the null space joint velocity vector can be conveniently chosen as [Chiacchio et al., 1991]

$$\dot{q}_0 = J_C^T(q)K_C(x_{Cd} - x_C). \quad (40)$$

which allows the algorithm to work at a singularity of $J_C$ and even at an artificial singularity. A tracking error arises for the constraint task but, observing that the desired constraint task is often constant over time ($\dot{x}_{Cd} = 0$), it can be concluded that the solution based on (40) performs equally well at steady state.

4. Parallel Manipulators

In the latest years, a novel type of closed-chain manipulators have been receiving quite a deal of attention; namely, the parallel manipulators [Merlet, 2000] which are constituted by a fixed base and a mobile base, connected by a number of independent kinematic chains. This allows obtaining high structural stiffness and performing high-speed motions, e.g. the Delta robot and the Hexa robot. One drawback with respect to open-chain manipulators, though, is a typically reduced workspace.

A particular family of parallel manipulators is characterized by having a radial link connected to the end effector of variable length. To this family belongs the industrial robot Tricept in Fig. 4. It is a six-degree-of-freedom (dof) robot manipulator comprising a three-dof parallel structure and a spherical wrist.

Let $p$ denote the position of the origin of the frame attached to the mobile base. The inverse differential kinematics can be written as

$$\dot{q} = J^{-1}(p)v, \quad (41)$$

where $q$ is the vector of joint variables and $v$ is the linear velocity vector. The geometric Jacobian inverse $J^{-1}$ can be calculated from the inverse kinematics function, which is available via simple geometry.

On the other hand, the direct kinematics problem for a parallel robot consists of finding the vector of position coordinates $p$ as a function of the vector of joint variables $q$. Typically, such a problem does not admit closed-form solutions and numerical algorithms should be used.
Therefore, an effective solution to such kind of kinematic control problem for a parallel manipulator can be devised by transposing the above Jacobian-based algorithm, i.e. [Siciliano, 1999a]

\[ v = J(p)(\dot{q}_d + Ke), \]  

(42)

where \( e = q_d - q \) denotes the error between \( q_d \) and the computed joint variables \( q \). The resulting scheme is illustrated in Fig. 5.

An alternative and computationally more efficient solution can be devised which avoids the inversion of the inverse geometric Jacobian to compute \( J \) in (42). In fact, the choice

\[ v = J^{-T}(p)Ke \]  

(43)

is based on the transpose of the inverse geometric Jacobian, and can be proved to guarantee limited tracking errors and null steady-state errors. The resulting scheme is illustrated in Fig. 6.

5. **Cooperative Multiarm Systems**

Various robotic applications require the adoption of multiarm systems in lieu of single robot manipulators. These include, for instance, manipulation of heavy or non-rigid objects and mating of mechanical pieces.
In all such cases the multiple robots should operate in a cooperative fashion so as to synchronize the relative motions, avoid undesired collisions, maintain the grasp between the arms and the object, etc.

Without loss of generality, consider a system of two cooperative manipulators. For each manipulator \( i = 1, 2 \), let \( p_i \) and \( R_i \) respectively denote the end-effector position vector and rotation matrix which are referred to a common base frame.

The absolute position of the system can be defined as the origin of the absolute frame \( a \) which can be expressed as a function of the positions of the two end effectors. One simple choice is

\[
 p_a = \frac{1}{2}(p_1 + p_2). \tag{44}
\]

Then, the rotation matrix giving the absolute orientation can be defined as

\[
 R_a = R_1 R_{12} (\theta_{12}/2), \tag{45}
\]

where \( R_{12} \) and \( \theta_{12} \) are respectively the unit vector and the angle that realize the rotation described by \( R_2 \), i.e. the orientation of frame 2 with respect to frame 1. Therefore, the above choice corresponds to make a
rotation about axis $^1k_{12}$ by an angle which is half the angle needed to align $R_2$ with $R_1$.

The absolute position and orientation describe the task in terms of the composition of the position and orientation of the single manipulators. It is clear that there exist infinite end-effector configurations giving the same absolute position and orientation. Therefore, in order to fully describe a coordinated motion, the position and orientation of one manipulator relative to the other is also of concern.

The relative position between the two end effectors can be defined as

$$p_r = p_2 - p_1.$$  \hspace{1cm} (46)

The relative orientation between the two end effectors can be defined with reference to the end-effector frame of either manipulator —say the first one— in terms of the rotation matrix

$$^1R_r = ^1R_2.$$  \hspace{1cm} (47)

The position and orientation of the various frames is illustrated in Figure 7.

![Figure 7. Absolute and Relative Frames in a Dual-Arm System](image)

Notice that, in order too be independent of the absolute motion of the system, it is more convenient to specify the relative position with reference to the absolute frame, i.e. $^a p_r$. The relationship between $^a p_r$ and $p_r$ is given by

$$p_r = R_a {^a p_r}.$$  \hspace{1cm} (48)
A distinguished feature of the proposed formulation is that coordinated motion of the system is achieved without necessarily assuming that the two manipulators are kinematically constrained through the presence of an object between the two end effectors. Nevertheless, if the two end effectors hold a common object, general manipulation tasks can be described by the above formulation. For instance, if the task is to move a tightly grasped object without deforming it, a trajectory has to be assigned to \( p_a \) and \( R_a \) while \( a^p_r \) and \( ^1R_r \) have to be kept constant. Yet, if the task is to stretch, bend or shear the object, suitable trajectories have to be specified for the relative variables too. Cases of non-tight grasp can be handled as well [Chiacchio et al., 1996].

Having established a task formulation for the direct kinematics of the two-manipulator system, it is useful to derive also the differential kinematics relating the coordinated (absolute and relative) velocities to the corresponding velocities of the two manipulators. This is of interest not only for characterizing the velocity mappings, similar to task formulations using the grasp matrix, but also for solving the inverse kinematics of the two-manipulator system as well as for handling the presence of redundant degrees of freedom in the system.

The absolute linear velocity of the system is obtained as the time derivative of (44), i.e.
\[
\dot{p}_a = \frac{1}{2}(\dot{p}_1 + \dot{p}_2).
\] (49)

Differentiating (45) with respect to time and using (10) yields
\[
\omega_a = \frac{1}{2}(\omega_1 + \omega_2).
\] (50)

The relative linear velocity of the system is obtained as the time derivative of (46), which in view of (48) gives
\[
\dot{p}_r = R_a a^p_r + S(\omega_a)p_r
\] (51)
with \( \omega_a \) as in (50). Finally, differentiating (47) with respect to time and using (10) yields
\[
\omega_r = \omega_2 - \omega_1
\] (52)
which has been expressed in the base frame.

The differential kinematics of each manipulator is described by
\[
\begin{bmatrix}
\dot{p}_i \\
\omega_i
\end{bmatrix} = J_i(q_i)\dot{q}_i \quad i = 1, 2,
\] (53)
with obvious meaning of the quantities.
At this point, combining (49),(50) and taking into account (53) yields
\[
\begin{bmatrix}
\dot{p}_a \\
\omega_a
\end{bmatrix} = J_a(q_1, q_2) \begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2
\end{bmatrix}.
\] (54)

where the absolute Jacobian is defined as
\[ J_a = \begin{bmatrix}
\frac{1}{2} J_1 & \frac{1}{2} J_2
\end{bmatrix}. \] (55)

Further, combining (51),(52) and taking into account (53) yields
\[
\begin{bmatrix}
\dot{p}_r \\
\omega_r
\end{bmatrix} = J_r(q_1, q_2) \begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2
\end{bmatrix},
\] (56)

where the relative Jacobian is defined as
\[ J_r = \begin{bmatrix}
-J_1 & J_2
\end{bmatrix}. \] (57)

The algorithm based on (12) can be keenly applied to solve the inverse
kinematics for the two-manipulator system at issue. To this purpose, it
is sufficient to define the joint variable vector as
\[ q = \begin{bmatrix}
q_1 \\
q_2
\end{bmatrix}, \] (58)

and the Jacobian as
\[ J = \begin{bmatrix}
J_a \\
J_r
\end{bmatrix}, \] (59)

where \( J_a, J_r \) are given as in (55) and (57), respectively.

The resolved velocity \( v \) in (12) can be chosen as
\[
\begin{bmatrix}
v_{ad} + K_a e_a \\
v_{rd} + K_r e_r
\end{bmatrix}.
\] (60)

The absolute velocity term is given by
\[ v_{ad} = \begin{bmatrix}
\dot{p}_{ad} \\
\omega_{ad}
\end{bmatrix}, \] (61)

where \( \dot{p}_{ad} \) and \( \omega_{ad} \) are respectively the desired absolute linear and angular velocities specified by the user in the base frame. The relative velocity term is given by
\[
\begin{bmatrix}
R_a^a \dot{p}_{rd} + S(\omega_a) R_a^a p_{rd} \\
\omega_{rd}
\end{bmatrix},
\] (62)
where \( a^q p_{rd} \) is the desired relative linear velocity specified by the user in the object frame and \( 1^q \omega_{rd} \) is the desired relative angular velocity specified by the user in the end-effector frame of the first manipulator. Notice that the expression of the translational part of the relative velocity presents an additional term which is a consequence of having assigned the relative position with reference to the absolute frame.

Finally, the absolute error term in (60) has a position and an orientation component and is given by

\[
e_a = \left[ \frac{1}{2} \left( S(n_a)n_{ad} + S(s_a)s_{ad} + S(a_a)a_{ad} \right) \right],
\]

where \( p_{ad} \) is the desired absolute position specified by the user in the base frame, \( p_a \) is the actual absolute position that can be computed as in (44), \( n_{ad}, s_{ad}, a_{ad} \) are the column vectors of the rotation matrix \( R_{ad} \) giving the desired absolute orientation specified by the user in the base frame, and \( n_a, s_a, a_a \) are the column vectors of the rotation matrix \( R_a \) in (45).

The relative error is given by

\[
e_r = \left[ \frac{1}{2} R_{a} \left( S(1^n r_1)n_{rd} + S(1^s r_1)s_{rd} + S(1^a r_1)a_{rd} \right) \right].
\]

The rotation \( R_a \) is aimed at expressing the desired relative position \( a^q p_{rd} \), assigned by the user in the absolute frame, in the base frame; in this way, the specification of the desired relative position between the two end effectors is not affected by the absolute frame orientation. Further in (64) notice that: \( p_r \) can be computed as in (46); \( 1^n r_1, 1^s r_1, 1^a r_1 \) are the column vectors of the rotation matrix \( 1^q R_{rd} \) giving the desired relative orientation specified by the user in the end-effector frame of the first manipulator; \( 1^n r, 1^s r, 1^a r \) are the column vectors of the rotation matrix \( 1^q R_r \) in (47); and the rotation \( R_1 \) is aimed at expressing the orientation error in the base frame.

6. Underwater Vehicle–Manipulator Systems

Execution of underwater manipulation tasks requires the use of a robot manipulate mounted onboard a vehicle actuated by thrusters; for that, an Underwater Vehicle–Manipulator System (UVMS) is always kinematically redundant due to the dof’s provided by the vehicle itself besides those provided by the robot arm. This naturally poses a redundancy resolution problem for motion coordination between the vehicle and the manipulator, whose solution can significantly increase efficiency of the system.
As a matter of fact, while it is obvious that gross motion of the UVMS must be provided by vehicle movements and fine motion of the end effector is best accomplished by the sole manipulator motion, reconfiguration of the whole system is required when the manipulator would be asked to work at the boundaries of its workspace or close to a kinematic singularity. The reconfiguration of the UVMS must suitably trade off the need of saving energy and keeping the control bandwidth high (that would call for motion of the manipulator) against the need of ensuring high manipulability for the robot arm (that would require continuous adjustment of the vehicle position and orientation).

The above redundancy resolution problem can be dealt with in the framework of task-priority Jacobian-based inverse kinematics algorithms as follows [Antonelli and Chiaverini, 2003a].

The vehicle is completely described by its position and orientation with respect to a base frame (frame $b$) that is assumed to be earth-fixed and inertial. Define the vector $x_v = [p_v^T \varphi_v^T]^T$, where $p_v$ is the vector of vehicle position coordinates and $\varphi_v$ is the vector of vehicle Roll-Pitch-Yaw Euler-angle coordinates, both quantities defined with reference to the base frame.

It is useful to define the vehicle’s velocity in a frame attached to the vehicle (frame $v$); let $v_x = [v^p_v^T \, v^\omega_v^T]^T$, where $v^p_v$ is the linear velocity of the vehicle with respect to the base frame and $v^\omega_v$ is the angular velocity of the vehicle, both quantities defined with reference to frame $v$.

The above velocity vectors satisfy the following equations:

$$v^p_v = vR \, p_v$$
$$v^\omega_v = T(\varphi_v) \, \dot{\varphi}_v,$$

where $vR$ is the rotation matrix from frame $b$ to frame $v$, and $T$ is a suitable transformation matrix as in (9).

At this point, define $q = [q_1 \cdots q_n]^T$ as the vector of joint positions, $n$ being the number of joints. By introducing the vector $\zeta = [v^p_v \, v^\omega_v \, \dot{q}^T]^T$ it is possible to rewrite the relation between the above velocities in compact form, i.e.

$$\zeta = \begin{bmatrix} vR & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \dot{x}_v \\ \dot{\varphi}_v \end{bmatrix} = J_S \begin{bmatrix} \dot{x}_v \\ \dot{q} \end{bmatrix},$$

where $J_S$ is the system Jacobian, while $I$ and $O$ respectively denote identity and null matrices of appropriate dimensions.

Since the task to be executed requires position/orientation control of the manipulator’s end effector, it is necessary to consider the end-effector
posture \( x_e = [p_e^T \, \varphi_e^T]^T \), where \( p_e \) is the end-effector position and \( \varphi_e \) is the end-effector orientation (Euler angles), both quantities defined with reference to the base frame. Notice that the vector \( p_e \) is a function of the system configuration, i.e. \( p_e(x_v, q) \), while the vector \( \varphi_e \) does only depend on the vehicle orientation, i.e. \( \varphi_e(\varphi_v, q) \). The relationship between the end-effector posture and the system configuration can be expressed by the following nonlinear kinematic equation

\[
x_e = k(x_v, q).
\]  

The time derivatives of the end-effector position and orientation are related to the actual end-effector velocities via relations analogous to (65) and (66); namely,

\[
\begin{align*}
\dot{e}p_e &= eR \dot{p}_e, \\
\dot{e}\omega_e &= T(\varphi_e) \dot{\varphi}_e,
\end{align*}
\]  

where \( eR \) is the rotation matrix from the base frame to the end-effector frame (see Figure 8) and \( T \) is the matrix associated to the use of the Euler angles of the end-effector frame.

\[\text{Figure 8. Sketch of a UVMS with Relevant Frames}\]

The end-effector velocities in the base frame are related to the system velocities by a suitable Jacobian matrix, i.e.

\[
v_e = \begin{bmatrix} \dot{p}_e \\ R_e \dot{\omega}_e \end{bmatrix} = J(x_v, q) \zeta.
\]  

\]
Notice that the Jacobian has been derived with respect to the angular velocity of the end effector expressed in the base frame. This is done in order to allow implementation of an inverse kinematics algorithm in terms of quaternions as described in the following.

The objective of kinematic control for a UVMS is to find suitable vehicle/joint trajectories $x_v(t), q(t)$ that correspond to a desired end-effector trajectory $x_{ed}(t)$. The outputs of the inverse kinematics algorithm provide the reference values to the control law of the UVMS. This control law will be in charge of computing the driving forces aimed at tracking the reference trajectory for the system while counteracting dynamic effects, external disturbances, and modeling errors.

Equation (71) maps the $(6+n)$-dimensional vehicle/joint velocities into the $m$-dimensional end-effector task velocities, where the typical case $(6+n) \geq m$ will be considered. Notice that, in case of UVMS, the Jacobian has always full rank due to the mobility of the vehicle, i.e. a rigid body with 6 dof’s. However, movement of the vehicle has to be avoided when unnecessary.

Considering as primary task the end-effector position/orientation, the Jacobian-based inverse kinematics algorithm (39) becomes [Antonelli and Chiaverini, 1998]:

$$
\zeta_r = J^\dagger(x_v, q)(v_{ed} + Ke) + (I - J^\dagger(x_v, q)J(x_v, q))J_C^\dagger(x_v, q)(\dot{x}_{Cd} + K_C e_C),
$$

where the constraint task is to be defined later, while $e$ and $e_C$ are suitable expressions of the errors.

As for the vector components related to the position variables, the error is simply given by the difference between the desired and the actual values. By using the quaternion attitude representation for the error components related to the orientation variables, the vector $\epsilon$ turns out to be [Chiaverini and Siciliano, 1999a]:

$$
e = \begin{bmatrix} p_{ed} - p_e \\
\eta_e d - \eta_d e - S(\epsilon_d) \epsilon \end{bmatrix},
$$

where $Q_d = \{\eta_d, \epsilon_d\}$ and $Q = \{\eta, \epsilon\}$ are the desired and actual attitudes expressed by quaternions, respectively.

Because of the different inertia characteristics of the vehicle and of the manipulator, it would be preferable to perform fast motions of small amplitude by means of the manipulator while leaving the vehicle with the task of executing slow gross motions. This might be achieved by adopting a weighted pseudoinverse

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

(74)
with the \((6+n) \times (6+n)\) matrix

\[
W^{-1}(\beta) = \begin{bmatrix}
(1-\beta)I & O \\
O & \beta I
\end{bmatrix},
\]

(75)

where \(\beta\) is a weighting factor belonging to the interval \([0, 1]\) such that \(\beta = 0\) corresponds to sole vehicle motion and \(\beta = 1\) to sole manipulator motion.

During task execution, setting a constant value of \(\beta\) would mean to fix the motion distribution between the vehicle and the manipulator. Nevertheless, the use of a fixed weighting factor inside the interval \([0, 1]\) has a drawback: it causes motion of the manipulator also if the desired end-effector posture is out of reach; on the other hand, it causes motion of the vehicle also if the manipulator alone could perform the task.

Another problem is the necessity to handle a large number of variables; UVMS’s, in fact, are complex systems and several variables must be monitored during the motion, e.g. the manipulator manipulability, the joint range limits to avoid mechanical breaks, the vehicle roll and pitch angles for correct tuning of the proximity sensors, the yaw angle to exploit the vehicle shape in presence of ocean current, etc. As it can be easily understood, it is quite difficult to handle all these terms without a kinematic control approach. Nevertheless, the existing techniques do not allow finding a flexible and reliable solution.

To overcome this drawback a fuzzy theory approach can be considered at two different levels. First, it is desired to manage the distribution of motion between the vehicle and the manipulator; second, it is possible to consider multiple secondary tasks that are activated only when the corresponding variable is outside (inside) a desired range. This can be done using different weighting factors adjusted on-line by a fuzzy inference system [Antonelli and Chiaverini, 2003a; Antonelli and Chiaverini, 2003b]. In detail, the crisp outputs are the scalar \(\beta\) in (75) and a vector of coefficients \(\alpha_i\) that are used in the task priority equation as follows

\[
\zeta = J_W^\dagger (v_{ed} + Ke) + \left( I - J_W^\dagger J_W \right) \left( \sum_i \alpha_i J_{C_i}^\dagger w_{C_i} \right),
\]

(76)

where \(w_{C_i}\) are suitably defined constraint task variables and \(J_{C_i}\) are the associated Jacobians. Both \(\beta\) and \(\alpha_i\)’s are tuned according to the state of the system and to given behavioral rules. The inputs of the fuzzy inference system depend on the variables of interest in the specific mission. As an example, the end-effector error, the ocean current measure, the system’s dexterity, as well as the force sensor readings, can be easily taken into account by setting up a suitable set of fuzzy rules.
To avoid the exponential growth of the fuzzy rules to be implemented as the number of tasks is increased, the secondary tasks are suitably organized in a hierarchy. Also, the rules have to guarantee that only one $\alpha_i$ is high at a time so as to avoid conflict between the secondary tasks.

7. Spacecraft–Manipulator Systems

Entrusting extravehicular activities to a mechanical robot manipulator rather than to an astronaut is foreseen to reduce the danger of space servicing jobs.

Consider a system composed by an n-dof manipulator with rigid links mounted on a rigid body spacecraft, free-floating in a zero-gravity environment. In the following, $\mathbf{q}$ will denote the ($n \times 1$) vector of joint variables. The vector $\mathbf{p}_s$ represents the position of a spacecraft-fixed coordinate frame ($s$) with respect to an inertial reference frame; $\mathbf{R}_s$ is the rotation matrix expressing the spacecraft attitude, i.e. the orientation of the spacecraft frame with respect to the inertial frame. Let also $\mathbf{p}_e$ and $\mathbf{R}_e$ express, respectively, the position and orientation of the end-effector frame with respect to the same inertial reference frame.

The kinematic equation relating the joint and spacecraft variables to the end-effector position can be written as

$$\mathbf{p}_e = \mathbf{p}_s + s\mathbf{p}_{es}(\mathbf{q}), \quad (77)$$

where $s\mathbf{p}_{es}$ is the position of the end-effector frame relative to the spacecraft frame; the end-effector orientation can be described by the rotation matrix

$$\mathbf{R}_e = R_s^s \mathbf{R}_e(\mathbf{q}), \quad (78)$$

where $s\mathbf{R}_e = R^T_s \mathbf{R}_e$ is the rotation matrix expressing the relative orientation between the end-effector frame and the spacecraft frame. Notice that $s\mathbf{p}_{es}(\mathbf{q})$ and $s\mathbf{R}_e(\mathbf{q})$ represent the usual direct kinematics equations of a ground-fixed manipulator with respect to its base frame (see Figure 9).

In view of solving the inverse kinematics for such a system, it is convenient to consider differential kinematics in lieu of (77),(78). Let $\mathbf{v}_e = [\dot{\mathbf{p}}_e^T \omega_e^T]^T$ be the vector of generalized end-effector velocity, where $\dot{\mathbf{p}}_e$ and $\omega_e$ denote the linear and angular velocity, respectively. Let $s\mathbf{J}_{es}$ be the manipulator geometric Jacobian relating the joint velocities $\dot{\mathbf{q}}$ to the end-effector velocity relative to the spacecraft frame $s\mathbf{v}_{es} = [s\dot{\mathbf{p}}_{es}^T s\omega_{es}^T]^T$, where $s\omega_{es} = \mathbf{R}_e^s(\omega_e - \omega_s)$. Let also $\mathbf{v}_s = [\mathbf{p}_s^T \omega_s^T]^T$ be the vector of generalized spacecraft velocity. Then, by differentiating (77) and (78),
it follows
\[ v_c = J_s(q, R_s)v_s + R_s^sJ_{es}(q)\dot{q}, \quad (79) \]
with
\[ J_s = \begin{bmatrix} I & -S(R_s p_{es}) \\ O & I \end{bmatrix}. \quad (80) \]
Since the spacecraft position is of no concern, \( \dot{p}_s \) can be eliminated from (79). This can be achieved by exploiting the geometrical definition of the mass center of the system \( p_G \)
\[ p_G \sum_i m_i = \sum_i m_i r_i, \quad (81) \]
where \( m_i \) is the mass of the \( i \)th rigid body in the system and \( r_i \) represents the position of its center of mass with respect to the inertial reference frame. Hence, by following the guidelines in [Umetani and Yoshida, 1989] and assuming null initial velocity for the system’s center of mass, the following expression can be derived from (79) and (81)
\[ v_c = J_s(q, R_s)\omega_s + J_{c}(q, R_s)\dot{q}, \quad (82) \]
where the matrices \( \bar{J}_s \) and \( \bar{J}_e \) depend on the Jacobian matrices \( J_s \) and \( J_{es} \) in (79), the masses \( m_i \) and the position vectors \( r_i \).

It is assumed that no external forces or torques act on the center of mass of the system. Thus, there are no devices intended to change spacecraft attitude, e.g. reaction wheels or thrusters. In this case, momentum conservation dictates that

\[
\sum_i m_i \dot{r}_i = 0 \tag{83}
\]

for the translational momentum and

\[
\sum_i (M_i \omega_i + m_i r_i \times \dot{r}_i) = 0, \tag{84}
\]

where \( \omega_i \) is the angular velocity of the \( i \)th body in the system and \( M_i \) is the inertia matrix around its center of mass. It has been assumed that the initial momentum of the free-floating system is null, without loss of generality.

The quantities \( r_i, \dot{r}_i, \omega_i \) and \( M_i \) are referred to the inertial frame, and it is not difficult to compute their expressions as a function of \( R_s \) and \( q \) [Nakamura and Mukherjee, 1991]. Substituting such expressions into eqs. (83),(84), momentum conservation can be compactly expressed as

\[
M_s \omega_s + M_e \dot{q} = 0, \tag{85}
\]

where \( M_s \) is a \( (3 \times 3) \) matrix related to the spacecraft inertia and \( M_e \) is a \( (3 \times n) \) matrix related to the manipulator inertia.

Equations (82) and (85) are fundamental for analyzing the motion of the system composed by the robotic manipulator mounted on the free-floating spacecraft. Since \( M_s \omega_s \) represents the spacecraft rotational momentum, \( M_s \) is a non-singular matrix; then, solving (85) for \( \omega_s \) and substituting in (82) allows eliminating the dependence on the spacecraft attitude changes, i.e.

\[
v_e = J_G \dot{q} \tag{86}
\]

where the matrix

\[
J_G = \bar{J}_e - \bar{J}_s M_s^{-1} M_e \tag{87}
\]

is termed the generalized Jacobian for the spacecraft/manipulator system [Umetani and Yoshida, 1989].

The attractive feature of (86) is its formal analogy with the well-known differential kinematics equation for ground-fixed manipulators. The manipulator Jacobian \( J_e \) is modified by the presence of a term accounting for the relative inertial weight between the spacecraft and the manipulator. The larger the spacecraft inertia, the smaller the reaction caused...
by the manipulator motion; in the limit of a very massive spacecraft, the generalized Jacobian will tend to the manipulator Jacobian.

The algorithm based on (12) can be keenly applied to solve the inverse kinematics for the free-floating manipulator system at issue with $J_G$ in lieu of $J$.

With reference to the use of the unit quaternion for the orientation error, the resolved velocity can be chosen as

$$v = \begin{bmatrix} \dot{p}_d + K_p e_p \\ \omega_d + K_o e_{de} \end{bmatrix}. \tag{88}$$

In the framework of inverse kinematics algorithms, it is important to recognize the presence of redundant degrees of freedom in the system with respect to the required task. As pointed out in [Nenchev et al., 1992], three cases of redundancy can be distinguished in connection with the number of degrees of mobility (joint variables) $n$ versus the number of dof’s characterizing the assigned task (task space variables) $m_s + m_e$, where $m_s$ and $m_e$ refer to the spacecraft and manipulator task, respectively. If $n < m_s + m_e$, the manipulator can be redundant with respect either to the spacecraft task ($n > m_s$) or to the end-effector task ($n > m_e$), but redundancy will not allow specifying a coordinated task for the spacecraft and the end-effector. If $n = m_s + m_e$, the available dof’s can be exploited to coordinate the motion of the spacecraft with that of the end-effector. If $n > m_s + m_e$, it is possible to introduce additional constraints to be satisfied along with spacecraft/manipulator motion coordination. In the following it is assumed that $n \geq m_s + m_e$.

Of course, the number of dof’s $n$ becomes larger than $m_s + m_e$ either by increasing the number of joints or by relaxing some task variables. Hence, the Jacobian matrix to be considered in (86) may be obtained by eliminating some rows of $J_G$ corresponding to the relaxed task variables, i.e. its dimensions become $(m_e \times n)$ with $m_e \leq 6$ and $n \geq 6$. This implies that suitable strategies have to be pursued to manage both the presence of a non-square Jacobian matrix and the redundant dof’s in the inverse kinematics algorithms previously defined.

One possibility would be to apply the Jacobian transpose algorithm to the robotic system described by (86). With this solution, however, the resulting spacecraft attitude varies as the manipulator end-effector moves along the trajectory. It is then advisable to exploit the redundant dof’s $n - m_e \geq m_s$ to impose a desired time evolution of spacecraft attitude $R_{sd}$. This is a typical case of augmented task space for redundant manipulators.

Since a conflict may arise between the end-effector and the constraint (spacecraft) task, an order of priority should be assigned. By revisiting
the solution algorithm for redundant manipulators in (39), the joint velocity solution can be computed as [Caccavale and Siciliano, 2001]

\[
\dot{q} = J_G^T v + (I - J_G^T J_C) J_C^T K_C e_C,
\]

(89)

where \( e_C \) is the error for the constraint task and \( J_C \) is the associated Jacobian.

If \( n - m_e = m_s \), then the error relative to the constraint task can be computed in terms of the vector part of the unit quaternion \( \{\eta_{s,d}, \epsilon_{s,d} \} \), representing the orientation displacement between the desired and the actual spacecraft frame: \( e_C = \epsilon_{s,d} \); the corresponding Jacobian matrix can be derived from (85)

\[
J_C = -M_e^{-1} M_e.
\]

(90)

In this case, manipulator redundancy is exploited to reach the desired spacecraft attitude \( R_{s,d} \) while tracking the desired end-effector trajectory. On the other hand, if \( n - m_e > m_s \), then the constraint task can encompass also an additional constraint, such as mechanical joint range, obstacle avoidance etc.

With solution (89), the constraint task is not guaranteed to be satisfied along the whole motion execution. Therefore, for those applications where the constraint task is judged to be more important than the end-effector task, the order of priority can be switched with obvious transposition of subscripts in (89).

An interesting case is that when it is desired to keep the spacecraft attitude constant during manipulator motion (\( \omega_{s,d} = 0 \)), e.g. not disturbing the orientation of some antenna for communication between spacecraft and earth. According to the above technique, the joint velocity solution can be computed as

\[
\dot{q} = J_C^T K_C e_C + (I - J_C^T J_C) J_C^T K_P e_{de}
\]

(91)

with \( e_{de} = [e_p^T, e_{de}^T]^T \).

If no other constraint is imposed and \( n - m_e = m_s \), then \( J_C \) is given as in (90); it is assumed that \( J_C \) is non-singular, otherwise a transpose should be employed. By using the expression for \( J_G \) given in (87), one can write

\[
J_G = \bar{J}_e + \bar{J}_s J_C;
\]

(92)

then computing the matrix \( J_G(I - J_C^T J_C) \) leads to

\[
J_G(I - J_C^T J_C) = \bar{J}_e(I - M_e^T M_e),
\]

(93)

which coincides with the so-called fixed-attitude-restricted (FAR) Jacobian introduced in [Nenchev et al, 1990].
As a consequence, solution (91) provides end-effector trajectories not changing spacecraft attitude which are computed via the transpose of the fixed-attitude-restricted Jacobian and then can be simplified into
\[
\dot{q} = J_C^T K_C e_C + (I - M_e^T M_e)\bar{J}_e^T K_P e_{de}.
\]

(94)

8. Flexible manipulators

In order to improve the performance of typically bulky industrial robots, one of the current trends is to adopt lightweight materials in the construction of manipulators. These are believed to offer a number of advantages such as smaller energy consumption, higher payload-to-arm weight ratio and faster movements [Book, 1993].

From a modeling standpoint, the scenario is complicated by the presence of additional deflection variables, compared to the case of rigid manipulators where the joint variables suffice to describe the system configuration.

Without loss of generality, consider a planar \( n \)-link flexible manipulator with revolute joints which are subject only to bending deformations in the plane of motion, i.e. torsional effects are neglected. A sketch of a two-link arm is shown in Fig. 10 with coordinate frame assignment. The rigid motion is described by the joint angles \( \vartheta_i \), while \( w_i(x_i) \) denotes the transversal deflection of link \( i \) at \( x_i \), \( 0 \leq x_i \leq \ell_i \), being \( \ell_i \) the link length.

Let \( ^i p_i(x_i) = [x_i \ w_i(x_i)]^T \) be the position of a point along the deflected link \( i \) with respect to frame \( (X_i, Y_i) \) and \( p_i \) be the position of the same point in the base frame. Also let \( ^i r_{i+1} = ^i p_i(\ell_i) \) be the position of
the origin of frame \((X_{i+1}, Y_{i+1})\) with respect to frame \((X_i, Y_i)\), and \(r_{i+1}\) its position in the base frame.

The joint (rigid) rotation matrix \(R_i\) and the rotation matrix \(E_i\) of the (flexible) link at the end point are, respectively,

\[
R_i = \begin{bmatrix}
\cos \vartheta_i & -\sin \vartheta_i \\
\sin \vartheta_i & \cos \vartheta_i \\
\end{bmatrix}
\] (95)

and

\[
E_i = \begin{bmatrix}
1 & -w'_{ie} \\
w'_{ie} & 1 \\
\end{bmatrix}
\] (96)

where \(w'_{ie} = (\partial w_i/\partial x_i)|_{x_i = \ell_i}\) and the small deflection approximation \(\arctan w'_{ie} \simeq w'_{ie}\) has been made. Hence the above absolute position vectors can be expressed as

\[
p_i = r_i + W_i^i p_i
\] (97)

and

\[
r_{i+1} = r_i + W_i^i r_{i+1},
\] (98)

where \(W_i\) is the global transformation matrix from the base frame to \((X_i, Y_i)\) given by the recursive equation

\[
W_i = W_{i-1} E_{i-1} R_i = \hat{W}_{i-1} R_i,
\] (99)

with

\[
\hat{W}_0 = I.
\] (100)

On the basis of the above relations, the kinematics of any point along the manipulator is completely specified as a function of joint and link deflection.

A finite-dimensional model (of order \(m_i\)) of link flexibility can be obtained by the assumed modes technique. By exploiting separability in time and space of solutions to the Euler-Bernoulli equation for flexible beams

\[
(EI)_{i} \frac{\partial^4 w_i(x_i,t)}{\partial x_i^4} + \rho_i \frac{\partial^2 w_i(x_i,t)}{\partial t^2} = 0,
\] (101)

for \(i = 1, \ldots, n\) where \(\rho_i\) is the uniform density and \((EI)_i\) is the constant flexural rigidity of link \(i\), the link deflection can be expressed as

\[
w_i(x_i,t) = \sum_{j=1}^{m_i} \phi_{ij}(x_i) \delta_{ij}(t),
\] (102)

where \(\delta_{ij}(t)\) are the time-varying variables associated with the assumed spatial mode shapes \(\phi_{ij}(x_i)\) of link \(i\). The mode shapes have to satisfy
proper boundary conditions at the base (clamped) and at the end of each link (mass).

In view of (102), a direct kinematics equation can be derived expressing the position \( p \) of the manipulator tip point as a function of the joint variable vector \( \theta = [\vartheta_1 \ldots \vartheta_n]^T \) and the deflection variable vector \( \delta = [\delta_{11} \ldots \delta_{1m_1} \ldots \delta_{n1} \ldots \delta_{nm_n}]^T \), i.e.

\[
p = k(\theta, \delta). \tag{103}
\]

For later use in the inverse kinematics scheme, also the differential kinematics is needed. The absolute linear velocity of a manipulator point is

\[
\dot{p}_i = \dot{r}_i + W_i^i p_i + W_i^i \dot{p}_i, \tag{104}
\]

with \( \dot{r}_{i+1} = \dot{\theta}_i \). Since the links are assumed inextensible \( (\dot{x}_i = 0) \), then \( \dot{p}_i(x_i) = \begin{bmatrix} 0 & \dot{w}_i(x_i) \end{bmatrix}^T \). The computation of (104) takes advantage of the recursion

\[
\dot{W}_i = \dot{W}_{i-1} R_i + \dot{W}_{i-1} \dot{R}_i \quad \tag{105}
\]

with

\[
\dot{W}_i = \dot{W}_i E_i + \dot{W}_i E_i. \tag{106}
\]

Also, note that

\[
\dot{R}_i = S R_i \dot{\delta}_i, \quad \dot{E}_i = S \dot{\delta}_i, \tag{107}
\]

with

\[
S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{108}
\]

In view of (102), it is not difficult to show that the tip velocity can be expressed as

\[
\dot{p} = J_\delta(\theta, \delta) \dot{\theta} + J_\delta(\theta, \delta) \dot{\delta}. \tag{109}
\]

In a static situation the deflections are seen to satisfy the equation [De Luca and Siciliano, 1993]

\[
g_\delta(\theta) + K \delta = 0, \tag{110}
\]

where \( g_\delta \) is the gravity vector in the flexible dynamic equations that is only a function of \( \theta \) and \( K \) is the link stiffness matrix

\[
K = \text{diag}(k_{11}, \ldots, k_{1m_1}, \ldots, k_{n1}, \ldots, k_{nm_n}) \tag{111}
\]

with

\[
k_{ij} = \int_0^\ell_i (EI)_i \phi_{ij}^2(x_i) dx_i. \tag{112}
\]
From (110) the deflection variables can be computed as
\[ \delta = -K^{-1}g_\delta(\theta). \]

(113)

For later use in the inverse kinematics scheme, differentiating (113) with respect to time gives
\[ \dot{\delta} = -K^{-1}J_\delta(\theta)\dot{\theta}, \]

(114)

where \( J_\delta = dg/d\theta \). Plugging (113) into (109) yields
\[ \dot{p} = J_p(\theta, \delta)\dot{\theta}, \]

(115)

where
\[ J_p = J_\theta - J_\delta K^{-1}J_\delta \]

(116)
is the overall Jacobian matrix relating joint velocity to tip velocity. Notice that the Jacobian in (116) is obtained by modifying the rigid-body Jacobian \( J_\theta \) with a term that accounts for the deflections induced by gravity.

The kinematic control problem for a flexible manipulator can be formulated as follows: Given a desired constant tip position \( p_d \), find the corresponding joint variables and deflection variables that place the arm tip under gravity at the given position.

The attractive feature of the differential kinematics equation (115) is its formal analogy with the differential kinematics equation for a rigid manipulator. Therefore any Jacobian-based inverse kinematics scheme can be adopted in principle, e.g. applying the algorithm based on (6) gives [Siciliano, 1999b]
\[ \dot{\theta} = J_p^T(\theta, \delta)K_pe_p \]

(117)

with \( e_p = p_d - p \).

The approach can be easily extended to the case of a flexible manipulator in contact with a compliant surface [Siciliano and Villani, 2001].

Acknowledgements

The research work surveyed here wants to be a tribute to Bernard Roth. The authors have been greatly inspired not only by his publications in the field but more importantly by several discussions and private communications with him. Thanks a lot, Bernie, for showing us the way many times throughout the past twenty years!

References


