IV

Robotics

Miomir Vukobratović

19

Robot Kinematics

Bruno Siciliano
Università degli Studi di Napoli Federico II

19.1 Introduction

From a mechanical viewpoint, a robotic system generally consists of a locomotion apparatus (legs, wheels) to move in the environment and a manipulation apparatus to operate on the objects present. It is then important to distinguish between mobile robots and robot manipulators.

The mechanical structure of a robot manipulator consists of a sequence of links connected by means of joints. Links and joints are usually made as rigid as possible to achieve high precision in robot positioning. The presence of elasticity at the joint transmissions or the use of lightweight materials for the links poses a number of interesting issues that lead to separating the study of flexible robot manipulators from that of rigid robot manipulators. The latter are implicitly meant by the term "robots" throughout this chapter.

This chapter surveys the fundamentals of robot kinematics. Basic mathematical tools such as the rotation matrix, the unit quaternion, and the Euler angles are briefly recalled. They serve to describe the orientation of the robot's end effector that, together with the position can be expressed as a function of the joint variables. This is the direct kinematics equation that is derived through a systematic procedure based on the use of homogeneous transformations and the so-called Denavit-Hartenberg convention. The inverse kinematics problem is considered and closed-form solutions are found for simple geometries. Further, a treatment of differential kinematics based on the robot's Jacobian matrix, hereafter simply called the Jacobian (geometric or analytical) is provided. Specific attention is paid to the occurrence of singularities or redundancy in the context of the differential kinematics inversion. The material ends with the presentation of inverse kinematics algorithms with special emphasis on the definition of the end-effector orientation error; both a pseudoinverse and a transpose of the Jacobian are considered.

19.2 Description of Orientation

Robot manipulation tasks are typically specified in terms of the position and orientation of an end-effector frame with respect to a base frame. Position is uniquely described by the Cartesian coordinates of the origin of the end-effector frame, whereas various representations of orientation exist. Therefore, as a natural prelude to deriving the direct kinematics equation of a robot, some basic concepts about the orientation of a rigid body in space are briefly recalled in the following.

19.2.1 Rotation Matrix

The location of a rigid body in space is typically described in terms of the (3×1) *position vector* $\boldsymbol{p}$ and the (3×3) *rotation matrix* $\boldsymbol{R}$ describing the origin and the orientation of a frame attached to the body with respect to a fixed reference frame, i.e.,

$$\boldsymbol{R} = [\boldsymbol{x} \;\; \boldsymbol{y} \;\; \boldsymbol{z}] \tag{19.1}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ are the unit vectors expressing the direction cosines of the axes of the body frame with respect to the reference frame. It is straightforward to verify that the matrix $\boldsymbol{R}$ is orthogonal, meaning that

$$\boldsymbol{R}^{\mathrm{T}} \boldsymbol{R} = \boldsymbol{I} \tag{19.2}$$

thus implying the useful result that the transpose of a rotation matrix is equal to its inverse, i.e., $\boldsymbol{R}^{\mathrm{T}} = \boldsymbol{R}^{-1}$. Frame orientation is conventionally taken to be left-handed.

A rotation matrix possesses three equivalent geometrical meanings:

- It describes the mutual orientation between two coordinate frames (as above).
- It represents the coordinate transformation between the coordinates of a point expressed in two different frames (with common origin).
- It is the operator that allows rotating a vector in the same coordinate frame.

Elementary rotations are those made about one of the coordinate axes,

$$\boldsymbol{R}_X(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19.3}$$

$$\boldsymbol{R}_Y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{19.4}$$

$$\boldsymbol{R}_Z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{19.5}$$

which denote the *elementary rotation matrices* with respect to the *X, Y, Z* axes. These are useful to describe rotations about an arbitrary axis in space, as shown below.

Rotation matrices between multiple frames — say frames 0, 1, 2 — can be nicely composed according to the simple rule

$${}^0\boldsymbol{R}_2 = {}^0\boldsymbol{R}_1 {}^1\boldsymbol{R}_2 \tag{19.6}$$

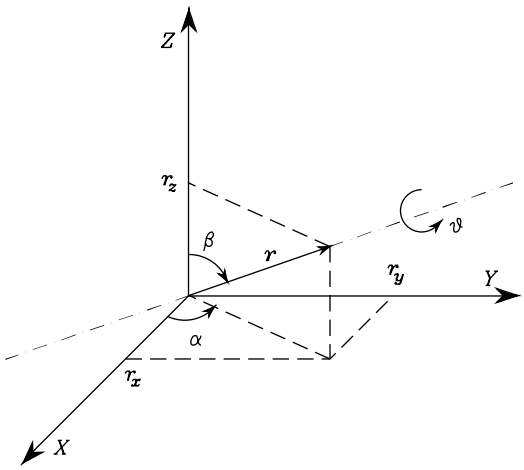

FIGURE 19.1 Rotation of a given angle about an arbitrary axis.

where the notation ${}^{j}\boldsymbol{R}_{i}$ denotes the rotation matrix of frame i with respect to frame j, and successive rotations are composed with respect to the axes of the current frame. Note also that ${}^{i}\boldsymbol{R}_{j} = ({}^{j}\boldsymbol{R}_{i})^{\mathrm{T}}$.

Expressing a rotation of a given angle about an arbitrary axis in space is often desired. Let $\boldsymbol{r} = [r_x \; r_y \; r_z]^{\mathrm{T}}$ be the unit vector of a rotation axis with respect to the reference frame. To derive the rotation matrix $\boldsymbol{R}(\vartheta, \boldsymbol{r})$ expressing the rotation of an angle ϑ about axis $\boldsymbol{r}$, it is convenient to compose elementary rotations about the coordinate axes of the reference frame. The angle is positive if the rotation is made counter-clockwise about axis $\boldsymbol{r}$.

As shown in Figure 19.1, a possible solution is obtained through the following sequence of rotations:

- Align Z with $\boldsymbol{r}$, which is obtained as the sequence of a rotation by α about Z and a rotation by β about Y;
- Rotate by ϑ about Z;
- Realign with the initial direction of Z, which is obtained as the sequence of a rotation by $-\beta$ about Y and a rotation by $-\alpha$ about Z.

The resulting rotation matrix is

$$\boldsymbol{R}(\vartheta, \boldsymbol{r}) = \boldsymbol{R}_Z(\alpha)\, \boldsymbol{R}_Y(\beta)\, \boldsymbol{R}_Z(\vartheta)\, \boldsymbol{R}_Y(-\beta)\, \boldsymbol{R}_Z(-\alpha). \tag{19.7}$$

By using the following relations:

$$\sin\alpha = \frac{r_y}{\sqrt{r_x^2 + r_y^2}} \qquad \cos\alpha = \frac{r_x}{\sqrt{r_x^2 + r_y^2}}$$
$$\sin\beta = \sqrt{r_x^2 + r_y^2} \qquad \cos\beta = r_z,$$

the rotation matrix of the *angle/axis description* in Equation (19.7) can be expressed as

$$\boldsymbol{R}(\vartheta, \boldsymbol{r}) = \begin{bmatrix} r_x^2(1-c_\vartheta) + c_\vartheta & r_x r_y(1-c_\vartheta) - r_z s_\vartheta & r_x r_z(1-c_\vartheta) + r_y s_\vartheta \\ r_x r_y(1-c_\vartheta) + r_z s_\vartheta & r_y^2(1-c_\vartheta) - c_\vartheta & r_y r_z(1-c_\vartheta) - r_x s_\vartheta \\ r_x r_z(1-c_\vartheta) - r_y s_\vartheta & r_y r_z(1-c_\vartheta) - r_x s_\vartheta & r_z^2(1-c_\vartheta) + c_\vartheta \end{bmatrix} \tag{19.8}$$

where the standard abbreviations for $\cos\vartheta$ and $\sin\vartheta$ have been used. Equation (19.8) can be cast in the more compact form

$$\boldsymbol{R}(\vartheta,\boldsymbol{r}) = c_\vartheta \boldsymbol{I} + (1 - c_\vartheta)\,\boldsymbol{r}\boldsymbol{r}^{\mathrm{T}} - s_\vartheta \boldsymbol{S}(\boldsymbol{r}) \tag{19.9}$$

where $\boldsymbol{I}$ is the (3×3) identity matrix and $\boldsymbol{S}(\cdot)$ is the matrix operator performing the cross product between two (3×1) vectors, i.e., $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$.

Although the axis can be arbitrary, the three components of $\boldsymbol{r}$ are constrained by the unit norm condition

$$\boldsymbol{r}^{\mathrm{T}}\boldsymbol{r} = 1. \tag{19.10}$$

Also, it is clear that $\boldsymbol{R}(-\vartheta, -\boldsymbol{r}) = \boldsymbol{R}(\vartheta, \boldsymbol{r})$, i.e., a rotation by $-\vartheta$ about $-\boldsymbol{r}$ cannot be distinguished from a rotation by ϑ about $\boldsymbol{r}$; hence, for $\vartheta = \pi$ the representation is not unique.

The angle and axis corresponding to a given rotation matrix

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{19.11}$$

are

$$\vartheta = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\boldsymbol{r} = \frac{1}{2\sin\vartheta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{19.12}$$

for $\sin\vartheta \neq 0$. Instead, if $\sin\vartheta = 0$, then it is necessary to refer directly to the particular expressions attained by $\boldsymbol{R}$ and find the solving formulæ in the two cases: if $\vartheta = 0$ the unit vector is arbitrary (no rotation has occurred), while if $\vartheta = \pi$, the above nonuniqueness problem is encountered. This drawback can be overcome by adopting a different four-parameter description, namely, the unit quaternion introduced next.

19.2.2 Unit Quaternion

With reference to the above angle/axis description of orientation, the *unit quaternion* (viz. Euler parameters) is defined as

$$\boldsymbol{Q} = \{\eta, \boldsymbol{\varepsilon}\} \tag{19.13}$$

where

$$\eta = \cos\frac{\vartheta}{2}$$

$$\boldsymbol{\varepsilon} = \sin\frac{\vartheta}{2}\boldsymbol{r}, \tag{19.14}$$

with $\eta \geq 0$ for $\vartheta \in [-\pi, \pi]$; η is called the scalar part of the quaternion while $\boldsymbol{\varepsilon}$ is called the vector part of the quaternion.

The constraint Equation (19.10) transforms into

$$\eta^2 + \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} = 1. \tag{19.15}$$

It is worth remarking that, different than the angle/axis description, a rotation by $-\vartheta$ about $-\boldsymbol{r}$ gives a vector part of the quaternion of the opposite sign from the one associated with a rotation by ϑ about $\boldsymbol{r}$, while the scalar part does not change. This solves the above nonuniqueness problem. The rotation matrix corresponding to a given quaternion is

$$\boldsymbol{R}(\eta, \boldsymbol{\varepsilon}) = (\eta^2 - \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon})\boldsymbol{I} + 2\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\mathrm{T}} - 2\eta\boldsymbol{S}(\boldsymbol{\varepsilon}). \tag{19.16}$$

On the other hand, the unit quaternion corresponding to a given rotation matrix Equation (19.11) is

$$\begin{aligned} \eta &= \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \\ \boldsymbol{\varepsilon} &= \begin{bmatrix} \frac{1}{2}\operatorname{sgn}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \frac{1}{2}\operatorname{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ \frac{1}{2}\operatorname{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix}. \end{aligned} \tag{19.17}$$

19.2.3 Euler Angles

Rotation matrices in general give a redundant description of frame orientation; in fact, they are characterized by nine elements that are not independent but are related by six constraints due to the orthogonality conditions in Equation (19.2). Even in the case of describing orientation in terms of rotation about an arbitrary axis or a unit quaternion, a representation in terms of four parameters is obtained. These components are not independent but are constrained by either condition (19.10) or condition (19.15). This implies that there are actually three free parameters to describe orientation.

A minimal representation of orientation can be obtained by using a set of three *Euler angles* $\boldsymbol{\varphi} = [\alpha \quad \beta \quad \gamma]^{\mathrm{T}}$. Among the 12 possible definitions of Euler angles, without loss of generality, the *XYZ* representation is considered to lead to the rotation matrix

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{\varphi}) &= \boldsymbol{R}_X(\alpha)\,\boldsymbol{R}_Y(\beta)\,\boldsymbol{R}_Z(\gamma) \\ &= \begin{bmatrix} c_\beta c_\gamma & -c_\beta s_\gamma & s_\beta \\ s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta \\ -c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha c_\beta \end{bmatrix} \end{aligned} \tag{19.18}$$

The set of the Euler angles corresponding to a given rotation matrix (19.11) is

$$\begin{aligned} \alpha &= \operatorname{Atan2}(-r_{23}, r_{33}) \\ \beta &= \operatorname{Atan2}\left(r_{13}, \sqrt{r_{11}^2 + r_{12}^2}\right) \\ \gamma &= \operatorname{Atan2}(-r_{12}, r_{11}) \end{aligned} \tag{19.19}$$

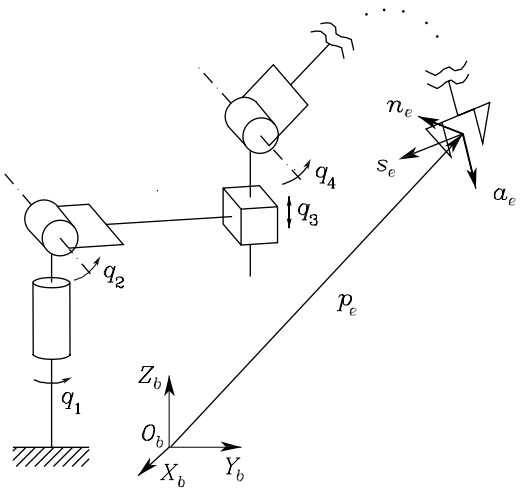

FIGURE 19.2 Schematic of an open-chain robot manipulator with a base frame and end-effector frame.

with $\beta \in (-\pi/2, \pi/2)$, whereas the solution is

$$\alpha = \text{Atan2}(r_{23}, -r_{33})$$
$$\beta = \text{Atan2}\left(r_{13}, -\sqrt{r_{11}^2 + r_{12}^2}\right) \tag{19.20}$$
$$\gamma = \text{Atan2}(r_{12}, -r_{11})$$

with $\beta \in (\pi/2, 3\pi/2)$; the function Atan2($y$, x) computes the arctangent of the ratio y/x but utilizes the sign of each argument to determine to which quadrant the resulting angle belongs.

Solutions (19.19) and (19.20) degenerate when $\beta = \pm\pi/2$; in this case, it is possible to determine only the sum or difference of α and γ, i.e.,

$$\alpha \pm \gamma = \text{Atan2}(r_{21}, r_{22}) \tag{19.21}$$

where the plus sign applies for $\beta = +\pi/2$ and the minus sign applies for $\beta = -\pi/2$.

19.3 Direct Kinematics

A robot manipulator consists of a kinematic chain of $n + 1$ links connected by means of n joints. Joints can essentially be of two types: *revolute* and *prismatic*; complex joints can be decomposed into these simple joints. Revolute joints are usually preferred because of their compactness and reliability. One end of the chain is connected to the base link to which a suitable base frame is attached, whereas an *end-effector* is connected to the other end and a suitable end-effector frame is attached. The basic structure of a robot is the open kinematic chain that occurs when only one sequence of links connects the two ends of the chain. Alternatively, a robot contains a closed kinematic chain when a sequence of links forms a loop. In Figure 19.2, an open-chain robot manipulator is illustrated with conventional representation of revolute and prismatic joints.

Direct kinematics of a robot consist of determining the mapping between the joint variables and the position and orientation of the end-effector frame with respect to the base frame.

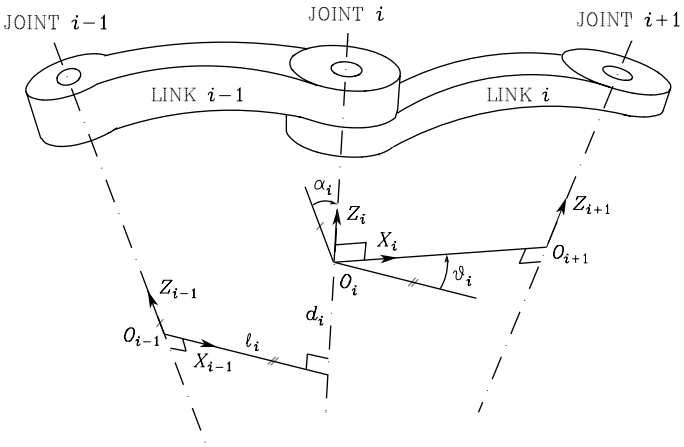

FIGURE 19.3 Kinematic parameters with modified Denavit-Hartenberg convention.

19.3.1 Homogeneous Transformation

As discussed above, the position of a rigid body in space is expressed in terms of the position of a suitable point on the body with respect to a reference frame (translation), while its orientation is expressed in terms of the components of the unit vectors of a frame attached to the body (with origin in the above point) with respect to the same reference frame (rotation).

The complete *coordinate transformation* between two frames (say frames 0, 1) is given by composing the translation ${}^0\boldsymbol{p}_1$ between the origins of the frames and the rotation ${}^0\boldsymbol{R}_1$ between the axes of the frames into a (4×4) homogeneous transformation matrix

$$ {}^0\boldsymbol{T}_1 = \begin{bmatrix} & {}^0\boldsymbol{R}_1 & & {}^0\boldsymbol{p}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{19.22} $$

Similar to the composition of rotations expressed by (19.6), a sequence of coordinate transformations from frame 0 to frame n can be composed as in the product

$$ {}^0\boldsymbol{T}_n = {}^0\boldsymbol{T}_1 \, {}^1\boldsymbol{T}_2 \ldots {}^{n-1}\boldsymbol{T}_n \tag{19.23} $$

where ${}^{i-1}\boldsymbol{T}_i$ denotes the homogeneous transformation expressing the position and orientation of frame i with respect to frame $i-1$. The relationship (19.23) is the basic tool for deriving the direct kinematics equation of a robot.

19.3.2 Denavit-Hartenberg Convention

An effective procedure for computing the direct kinematics function for a general robot is based on the so-called modified *Denavit-Hartenberg* convention. According to this convention, a coordinate frame is attached to each link of the chain and the overall transformation matrix from link 0 to link n is derived by composition of transformations between consecutive frames. With reference to Figure 19.3, let joint i connect link $i-1$ to link i, where the links are assumed to be rigid; frame i is attached to link i and can be defined as follows:

- Choose axis Z_i aligned with the axis of joint i.
- Choose axis X_i along the common normal to axes Z_i and Z_{i+1} with direction from joint i to joint $i+1$.
- Choose axis Y_i to complete a right-handed frame.

Once the link frames have been established, the position and orientation of frame i with respect to frame $i-1$ are completely specified by the following *kinematic parameters*.

α_i Angle between Z_{i-1} and Z_i about X_{i-1} measured counter-clockwise
ℓ_i Distance between Z_{i-1} and Z_i along X_{i-1}
ϑ_i Angle between X_{i-1} and X_i about Z_i measured counter-clockwise
d_i Distance between X_{i-1} and X_i along Z_i

Let $\mathbf{Rot}(K,\delta)$ $\left(\mathbf{Trans}(K,\delta)\right)$ denote the homogeneous transformation matrix expressing the rotation (translation) about (along) axis K by an angle (distance) δ. Then, the coordinate transformation of frame i with respect to frame $i-1$ can be expressed in terms of the above four parameters by the matrix

$$
\begin{aligned}
{}^{i-1}\boldsymbol{T}_i &= \mathbf{Rot}(X,\alpha_i)\mathbf{Trans}(X,\ell_i)\mathbf{Rot}(Z,\vartheta_i)\mathbf{Trans}(Z,d_i) \\
&= \begin{bmatrix}
\cos\vartheta_i & -\sin\vartheta_i & 0 & \ell_i \\
\cos\alpha_i\sin\vartheta_i & \cos\alpha_i\cos\vartheta_i & -\sin\alpha_i & -d_i\sin\alpha_i \\
\sin\alpha_i\sin\vartheta_i & \sin\alpha_i\cos\vartheta_i & \cos\alpha_i & d_i\cos\alpha_i \\
0 & 0 & 0 & 1
\end{bmatrix} \\
&= \begin{bmatrix}
 & {}^{i-1}\boldsymbol{R}_i & & {}^{i-1}\boldsymbol{p}_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{aligned}
\tag{19.24}
$$

where ${}^{i-1}\boldsymbol{R}_i$ is the (3×3) matrix defining the orientation of frame i with respect to frame $i-1$, and ${}^{i-1}\boldsymbol{p}_i$ is the (3×1) vector defining the origin of frame i with respect to frame $i-1$.

Dually, the transformation matrix defining frame $i-1$ with respect to frame i is given by

$$
\begin{aligned}
{}^{i}\boldsymbol{T}_{i-1} &= \mathbf{Trans}(Z,-d_i)\mathbf{Rot}(Z,-\vartheta_i)\mathbf{Trans}(X,-\ell_i)\mathbf{Rot}(X,-\alpha_i) \\
&= \begin{bmatrix}
 & & & -\ell_i\cos\vartheta_i \\
 & {}^{i-1}\boldsymbol{R}_i^{\mathrm{T}} & & \ell_i\sin\vartheta_i \\
 & & & -d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{aligned}
\tag{19.25}
$$

Two of the four parameters (ℓ_i and α_i) are always constant and depend only on the size and shape of link i. Of the remaining two parameters, only one is variable (degree of freedom) depending on the type of joint that connects link $i-1$ to link i. If q_i denotes the joint i variable, then it is

$$
q_i = \bar{\xi}_i\vartheta_i + \xi_i d_i \tag{19.26}
$$

where $\bar{\xi}_i = 1-\xi_i$, i.e.,

- $\xi_i = 0$ if joint i is revolute ($q_i = \vartheta_i$),
- $\xi_i = 1$ if joint i is prismatic ($q_i = d_i$).

In view of (19.26), the equation

$$
\bar{q}_i = \xi_i\vartheta_i + \bar{\xi}_i d_i \tag{19.27}
$$

gives the constant parameter at each joint to add to α_i and ℓ_i.

The above procedure does not yield a unique definition of frames 0 and n that can be chosen arbitrarily. Also, in all cases of nonuniqueness in the definition of the frames, it is convenient to make as many link parameters zero as possible, because this will simplify kinematics computation. A number of remarks are in order.

- A simple choice to define frame 0 is to take it coincident with frame 1 when $q_1 = 0$; this makes $\alpha_1 = 0$ and $\ell_1 = 0$, and $\bar{q}_1 = 0$.
- A similar choice for frame n is to take X_n along X_{n-1} when $q_n = 0$; this makes $\bar{q}_n = 0$.
- If joint i is prismatic, the direction of Z_i is fixed while its location is arbitrary; it is convenient to locate Z_i either at the origin of frame $i - 1$ ($\ell_i = 0$) or at the origin of frame $i + 1$ ($\ell_{i+1} = 0$).
- When the joint axes i and $i + 1$ are parallel, it is convenient to locate X_i to achieve either $d_i = 0$ or $d_{i+1} = 0$ if either joint is revolute.

In view of (19.23), through the composition of the individual link transformations, the coordinate transformation describing the position and orientation of frame n with respect to frame 0 is given by

$$ {}^0\boldsymbol{T}_n(\boldsymbol{q}) = {}^0\boldsymbol{T}_1(q_1)\,{}^1\boldsymbol{T}_2(q_2)\cdots{}^{n-1}\boldsymbol{T}_n(q_n), \tag{19.28} $$

where $\boldsymbol{q}$ denotes the $(n \times 1)$ vector of joint variables. To derive the direct kinematics, two further *constant* transformations have to be introduced; namely, the transformation from the base frame b to frame 0 ($^bT_0$) and the transformation from frame n to the end-effector frame e (^nT_e), i.e.,

$$ \begin{aligned} {}^b\boldsymbol{T}_e(\boldsymbol{q}) &= {}^b\boldsymbol{T}_0\,{}^0\boldsymbol{T}_n(\boldsymbol{q})\,{}^n\boldsymbol{T}_e \\ &= \begin{bmatrix} {}^b\boldsymbol{n}_e(\boldsymbol{q}) & {}^b\boldsymbol{s}_e(\boldsymbol{q}) & {}^b\boldsymbol{a}_e(\boldsymbol{q}) & {}^b\boldsymbol{p}_e(\boldsymbol{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{19.29} $$

where the normal, sliding, and approach unit vectors $\boldsymbol{n}$, $\boldsymbol{s}$, $\boldsymbol{a}$ have been formally introduced (Figure 19.2). Subscripts and superscripts can be omitted when the relevant frames are clear from the context.

The "modified" Denavit-Hartenberg convention stems from the fact that, in the "classical" convention, axis Z_i is aligned with the axis of joint $i + 1$ and the kinematic parameters differ accordingly.

An example of an open-chain robot is the anthropomorphic robot.

With reference to the frames illustrated in Figure 19.4, the Denavit-Hartenberg parameters are specified in Table 19.1.

Computing the transformation matrices in (19.24) and composing them as in (19.28) gives

$$ {}^0\boldsymbol{T}_6 = \begin{pmatrix} {}^0\boldsymbol{n}_6 & {}^0\boldsymbol{s}_6 & {}^0\boldsymbol{a}_6 & {}^0\boldsymbol{p}_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{19.30} $$

where

$$ {}^0\boldsymbol{p}_6 = \begin{bmatrix} c_1\left(c_2\,\ell_3 - s_{23}d_4\right) \\ s_1\left(c_2\,\ell_3 - s_{23}d_4\right) \\ s_2\,\ell_3 + c_{23}d_4 \end{bmatrix} \tag{19.31} $$

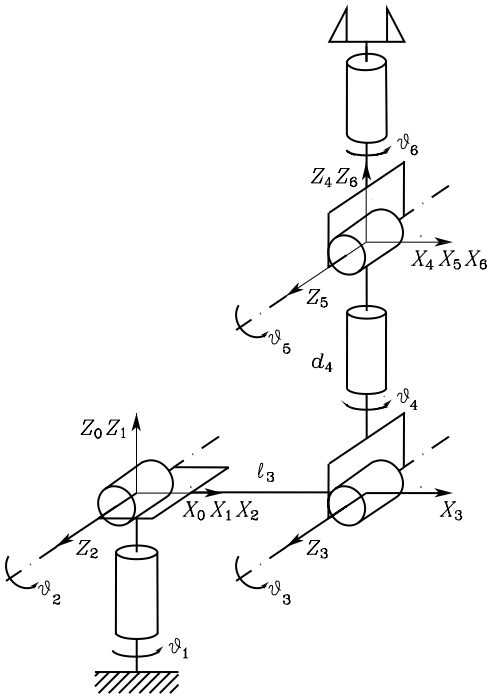

FIGURE 19.4 Anthropomorphic robot with frame assignment.

TABLE 19.1 Denavit-Hartenberg Parameters of the Anthropomorphic Robot

i	α_i	ℓ_i	ϑ_i	d_i
1	0	0	q_1	0
2	$\pi/2$	0	q_2	0
3	0	ℓ_3	q_3	0
4	$-\pi/2$	0	q_4	d_4
5	$\pi/2$	0	q_5	0
6	$-\pi/2$	0	q_6	0

for the position, and

$$
{}^0\boldsymbol{n}_6 = \begin{bmatrix} c_1\left(c_{23}\left(c_4c_5c_6 - s_4s_6\right) - s_{23}s_5c_6\right) - s_1\left(s_4c_5c_6 + c_4s_6\right) \\ s_1\left(c_{23}\left(c_4c_5c_6 - s_4s_6\right) - s_{23}s_5c_6\right) + c_1\left(s_4c_5c_6 + c_4s_6\right) \\ s_{23}\left(c_4c_5c_6 - s_4s_6\right) + c_{23}s_5c_6 \end{bmatrix} \tag{19.32}
$$

$$
{}^0\boldsymbol{s}_6 = \begin{bmatrix} c_1\left(-c_{23}\left(c_4c_5s_6 + s_4c_6\right) + s_{23}s_5s_6\right) + s_1\left(s_4c_5s_6 - c_4c_6\right) \\ s_1\left(-c_{23}\left(c_4c_5s_6 + s_4c_6\right) + s_{23}s_5s_6\right) - c_1\left(s_4c_5s_6 - c_4c_6\right) \\ -s_{23}\left(c_4c_5s_6 + s_4s_6\right) - c_{23}s_5s_6 \end{bmatrix} \tag{19.33}
$$

$$
{}^0\boldsymbol{a}_6 = \begin{bmatrix} -c_1\left(c_{23}c_4s_5 + s_{23}c_5\right) + s_1s_4s_5 \\ -s_1\left(c_{23}c_4s_5 + s_{23}c_5\right) - c_1s_4s_5 \\ -s_{23}c_4s_5 + c_{23}c_5 \end{bmatrix} \tag{19.34}
$$

for the orientation, where $c_i = \cos\vartheta_i$, $s_i = \sin\vartheta_i$, $c_{23} = \cos\left(\vartheta_2 + \vartheta_3\right)$, and $s_{23} = \sin\left(\vartheta_2 + \vartheta_3\right)$.

19.3.3 Joint Space and Task Space

If a task has to be assigned to the end-effector, it is necessary to specify both the end-effector's position and orientation. This is easy for the position $\boldsymbol{p}_e$. However, specifying the orientation through the unit vector triple ($\boldsymbol{n}_e$, $\boldsymbol{s}_e$, $\boldsymbol{a}_e$) is difficult, because their nine components must be guaranteed to satisfy the orthonormality constraints imposed by (19.2). Even with a four-parameter description of the orientation, one constraint in the form of either (19.10) or (19.15) should be satisfied.

On the other hand, if a minimal representation is adopted in terms of the Euler angles describing the orientation of the end-effector frame with respect to the base frame, a suitable ($m \times 1$) vector can be considered as

$$
\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_e \\ \boldsymbol{\varphi}_e \end{bmatrix}, \tag{19.35}
$$

where $\boldsymbol{p}_e$ describes the end-effector position and $\boldsymbol{\varphi}_e$ its orientation. This representation of position and orientation allows the description of the end-effector task in terms of a number of inherently independent parameters. The vector $\boldsymbol{x}$ is defined in the space in which the robot task is specified; hence, this space is typically called *task space* (operational space). The dimension of the task space is at most $m = 6$, because three coordinates specify position and three angles 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 $m = 3$, because two coordinates specify position and one angle specifies orientation.

On the other hand, the *joint space* (configuration space) denotes the space in which the ($n \times 1$) vector of joint variables $\boldsymbol{q}$ is defined. Taking into account the dependence of position and orientation from the joint variables, the direct kinematics equation can be written in a form other than (19.24), i.e.,

$$
\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}). \tag{19.36}
$$

It is worth noticing that the explicit dependence of the function $\boldsymbol{k}(\boldsymbol{q})$ from the joint variables for the orientation components is not available except for simple cases. In fact, on the most general assumption of a six-dimensional task space ($m = 6$), the computation of the three components of the function $\boldsymbol{\varphi}_e(q)$ cannot be performed in closed form but goes through the computation of the elements of the rotation matrix.

The notion of joint space and task space naturally allows introducing the concept of *kinematic redundancy.* This occurs when the dimension of the task space is smaller than the dimension of the joint space ($m < n$). Redundancy is a concept *relative* to the task assigned to the robot; a robot can be redundant with respect to a task and nonredundant with respect to another, depending on the number of task space variables of interest.

For instance, a three-degree-of-freedom planar robot becomes redundant if end-effector orientation is of no concern ($m = 2$, $n = 3$). Yet, the typical example of redundant robot is the human arm that has seven degrees of freedom: three in the shoulder, one in the elbow, and three in the wrist, without considering the degrees of freedom in the fingers ($m = 6$, $n = 7$).

19.4 Inverse Kinematics

The direct kinematics equation, either in the form (19.24) or in the form (19.36), establishes the functional relationship between the joint variables and the end-effector position and orientation. *Inverse kinematics* concerns the determination of the joint variables $\boldsymbol{q}$ corresponding to a given end-effector position $\boldsymbol{p}_e$ and orientation $\boldsymbol{R}_e$. The solution to this problem is of fundamental importance in order to translate the specified motion, naturally assigned in the task space, into the equivalent joint space motion that allows execution of the desired task.

With regard to the direct kinematics Equation (19.24), the end-effector position and rotation matrix are uniquely computed, once the joint variables are known. In general, this cannot be said for Equation (19.36), because the Euler angles are not uniquely defined. On the other hand, the inverse kinematics problem is much more complex for the following reasons.

- The equations to solve are general nonlinear equations for which it is not always possible to find closed-form solutions.
- Multiple solutions may exist.
- Infinite solutions may exist, e.g., in the case of a kinematically redundant robot.
- There might not be admissible solutions, in view of the robot kinematic structure.

Of course, the existence of solutions is guaranteed if the given end-effector position and orientation belong to the robot workspace.

On the other hand, the problem of multiple solutions depends not only on the number of degrees of freedom but also on the Denavit-Hartenberg parameters; in general, the greater the number of nonnull parameters, the greater the number of admissible solutions. For a six-degrees-of-freedom robot without mechanical joint limits, in general up to 16 admissible solutions exist. This occurrence demands some criteria to choose among admissible solutions.

The computation of closed-form solutions requires either algebraic intuition to find those significant equations containing the unknowns, or geometric intuition to discover those significant points on the structure for which it is convenient to express position and orientation. Or, in all those cases when there are no — or it is difficult to find — *closed-form solutions,* it might be appropriate to resort to numerical solution techniques. These clearly have the advantage of being applicable to any kinematic structure, but generally they do not allow computation of all admissible solutions.

19.4.1 Closed-Form Solutions

Most of the existing robots are kinematically simple, because they are typically formed by an arm (three or more degrees of freedom) which provides mobility and by a wrist which provides dexterity (three degrees of freedom). This choice is partially motivated by the difficulty of finding solutions to the inverse kinematics problem in the general case. In particular, a six-degrees-of-freedom robot has closed-form inverse kinematics solutions if three consecutive revolute joint axes intersect at a common point. This situation occurs when a robot has a so-called *spherical wrist* that is characterized by

$$\ell_5 = d_5 = \ell_6 = 0 \qquad \xi_4 = \xi_5 = \xi_6 = 0, \tag{19.37}$$

with $\sin\alpha_5 \neq 0$ and $\sin\alpha_6 \neq 0$ so as to avoid parallel axes (degenerate robot). In that case, it is possible to divide the inverse kinematics problem into two subproblems, because the solution for the *position* is *decoupled* from that for the *orientation*.

In the case of a three-degrees-of-freedom arm, for given end-effector position ${}^0\boldsymbol{p}_e$ and orientation ${}^0\boldsymbol{R}_e$, the inverse kinematics can be solved according to the following steps:

- Compute the wrist position ${}^0\boldsymbol{p}_4$ from ${}^0\boldsymbol{p}_e$;
- Solve inverse kinematics for (q_1, q_2, q_3);
- Compute ${}^0\boldsymbol{R}_3\ (q_1, q_2, q_3)$;
- Compute ${}^3\boldsymbol{R}_6(q_4, q_5, q_6) = {}^3\boldsymbol{R}_0\ {}^0\boldsymbol{R}_e\ {}^e\boldsymbol{R}_6$;
- Solve inverse kinematics for (q_4, q_5, q_6).

Therefore, on the basis of this kinematic decoupling, it is possible to solve the inverse kinematics for the arm separately from the inverse kinematics for the spherical wrist.

Consider the anthropomorphic robot in Figure 19.4, whose direct kinematics was given in (19.30). Finding the vector of joint variables $\boldsymbol{q}$ corresponding to given end-effector position ${}^0\boldsymbol{p}_e$ and orientation ${}^0\boldsymbol{R}_e$ is desired; without loss of generality, assume that ${}^0\boldsymbol{p}_e = {}^0\boldsymbol{p}_6$ and ${}^6\boldsymbol{R}_e = \boldsymbol{I}$.

Observing that ${}^0\boldsymbol{p}_6 = {}^0\boldsymbol{p}_4$, the first three joint variables can be solved from (19.31) which can be rewritten as

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} c_1\left(c_2\ell_3 - s_{23}d_4\right) \\ s_1\left(c_2\ell_3 - s_{23}d_4\right) \\ s_2\ell_3 + c_{23}d_4 \end{bmatrix}. \tag{19.38}$$

From the first two components of (19.38), it is

$$q_1 = \text{Atan}\,2\left(p_{y,}p_x\right). \tag{19.39}$$

Notice that another solution is

$$q_1 = \pi + \text{Atan}\,2\left(p_{y,}p_z\right). \tag{19.40}$$

The second joint variable can be found by squaring and summing the first two components of (19.38), i.e.,

$$p_x^2 + p_y^2 = \left(c_2\ell_3 - s_{23}d_4\right)^2; \tag{19.41}$$

then, squaring the third component and summing it to (19.41) lead to the solution

$$q_3 = \text{Atan}\,2\left(s_3, c_3\right) \tag{19.42}$$

where

$$s_3 = \frac{\ell_3^2 + d_4^2 - p_x^2 - p_y^2 - p_z^2}{2\ell_3 d_4} \qquad c_3 = \pm\sqrt{1 - s_3^2}.$$

Substituting q_3 in (19.41), taking the square root thereof and combining the result with the third component of (19.38) lead to a system of equations in the unknowns s_2 and c_2; its solution can be found as

$$s_2 = \frac{\left(\ell_3 - s_3 d_4\right) p_z - c_3 d_4 \sqrt{p_x^2 + p_y^2}}{p_x^2 + p_y^2 + p_z^2}$$

$$c_2 = \frac{\left(\ell_3 - s_3 d_4\right) \sqrt{p_x^2 + p_y^2} + c_3 d_4 p_z}{p_x^2 + p_y^2 + p_z^2},$$

and thus the second joint variable is

$$q_2 = \mathrm{Atan}\,2\left(s_2, c_2\right). \tag{19.43}$$

Notice that four admissible solutions are obtained according to the values of q_1, q_2, q_3, namely, shoulder-right/elbow-up, shoulder-left/elbow-up, shoulder-right/elbow-down, shoulder-left/elbow-down.

To solve for the three joint variables of the wrist, the following procedure can be used. Given the matrix

$${}^0\boldsymbol{R}_6 = \begin{pmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{pmatrix}, \tag{19.44}$$

the matrix ${}^0\boldsymbol{R}_3$ can be computed from the first three joint variables via (19.24), and thus the following equation is to be considered:

$$\begin{bmatrix} {}^3n_x & {}^3s_x & {}^3a_x \\ {}^3n_y & {}^3s_y & {}^3a_y \\ {}^3n_z & {}^3s_z & {}^3a_z \end{bmatrix} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 \\ s_5c_6 & -s_5s_6 & c_5 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 \end{bmatrix}. \tag{19.45}$$

The elements of the matrix on the right-hand side of (19.45) have been obtained by computing ${}^3\boldsymbol{R}_6$ via (19.24), whereas the elements of the matrix on the left-hand side of (19.45) can be computed as ${}^3\boldsymbol{R}_0\ {}^0\boldsymbol{R}_6$ with ${}^0\boldsymbol{R}_6$ as in (19.44), i.e.,

$$\begin{aligned} {}^3n_x &= c_{23}\left(c_1n_x + s_1n_y\right) + s_{23}n_z \\ {}^3n_y &= -s_{23}\left(c_1n_x + s_1n_y\right) + s_{23}n_z \\ {}^3n_z &= s_1n_x - c_1n_y; \end{aligned} \tag{19.46}$$

the other elements $\left({}^3s_x, {}^3s_y, {}^3s_z\right)$ and $\left({}^3a_x, {}^3a_y, {}^3a_z\right)$ can be computed from (19.46) by replacing $\left(n_x, n_y, n_z\right)$ with $\left(s_x, s_y, s_z\right)$ and $\left(a_x, a_y, a_z\right)$, respectively.

At this point, inspecting (19.45) reveals that from the elements [1, 3] and [3, 3], q_4 can be computed as

$$q_4 = \text{Atan2}\left({}^3a_z, -{}^3a_x\right). \tag{19.47}$$

Then, q_5 can be computed by squaring and summing the elements [1, 3] and [3, 3], and from the element [2, 3] as

$$q_5 = \text{Atan2}\left(\sqrt{\left({}^3a_x\right)^2 + \left({}^3a_z\right)^2}, {}^3a_y\right). \tag{19.48}$$

Finally, q_6 can be computed from the elements [2, 1] and [2, 2] as

$$q_6 = \text{Atan2}\left(-{}^3s_y, {}^3n_y\right). \tag{19.49}$$

It is worth noticing that another set of solutions is given by the triplet

$$q_4 = \text{Atan2}\left(-{}^3a_z, {}^3a_x\right) \tag{19.50}$$

$$q_5 = \text{Atan2}\left(-\sqrt{\left({}^3a_x\right)^2 + \left({}^3a_z\right)^2}, {}^3a_y\right) \tag{19.51}$$

$$q_6 = \text{Atan2}\left({}^3s_y, -{}^3n_y\right). \tag{19.52}$$

Notice that both sets of solutions degenerate when ${}^3a_x = {}^3a_z = 0$; in this case, q_4 is arbitrary and simpler expressions can be found for q_5 and q_6.

In conclusion, four admissible solutions have been found for the arm and two admissible solutions have been found for the wrist, resulting in a total of eight admissible inverse kinematics solutions for the anthropomorphic robot with a spherical wrist.

19.5 Differential Kinematics

The (3×1) vector $\dot{\boldsymbol{p}}$ of linear velocity of a rigid body in space is given by the time derivative of the position vector, while the (3×1) vector ω of angular velocity can be defined through the time derivative of the rotation matrix in the form

$$\dot{\boldsymbol{R}} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}. \tag{19.53}$$

With reference to the other descriptions of orientation, the relationship between the angular velocity and the time derivative of the unit quaternion is

$$\begin{bmatrix} \dot{\eta} \\ \dot{\boldsymbol{\varepsilon}} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^{\mathrm{T}} \\ \boldsymbol{\omega} & -\boldsymbol{S}(\boldsymbol{\omega}) \end{bmatrix}\begin{bmatrix} \eta \\ \boldsymbol{\varepsilon} \end{bmatrix} \tag{19.54}$$

which is known as the quaternion propagation rule, whereas that between the angular velocity and the time derivative of the Euler angles is

$$\boldsymbol{\omega} = \boldsymbol{T}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}} \tag{19.55}$$

where $\boldsymbol{T}(\boldsymbol{\varphi})$ depends on the particular choice of Euler angles.

The mapping between the $(n \times 1)$ vector of joint velocities $\dot{\boldsymbol{q}}$ and the (6×1) vector of end-effector (linear and angular) velocities $\boldsymbol{v}$ is established by the differential kinematics equation

$$\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{19.56}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the $(6 \times n)$ Jacobian matrix. 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. Hence, $\boldsymbol{J}(\boldsymbol{q})$ can be termed the geometric Jacobian of the robot.

19.5.1 Geometric Jacobian

In view of simple geometry, the velocity contributions of each joint to the linear and angular velocities of link n give the following relationship:

$$\begin{bmatrix} \dot{p}_n \\ \boldsymbol{\omega}_n \end{bmatrix} = \begin{bmatrix} \xi_1 z_1 + \bar{\xi}_1 (z_1 \times p_{1n}) & \cdots & \xi_n z_n + \bar{\xi}_n (z_n \times p_{nn}) \\ \bar{\xi}_1 z_1 & \cdots & \bar{\xi}_n z_n \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_n(q)\dot{q} \tag{19.57}$$

where z_k is the unit vector of axis Z_k and $\boldsymbol{p}_{kn}$ denotes the vector from the origin of frame k to the origin of frame n. Notice that $\boldsymbol{J}_n$ is a function of $\boldsymbol{q}$ through the vectors z_k and $\boldsymbol{p}_{kn}$ that can be computed on the basis of direct kinematics.

The geometric Jacobian can be computed with respect to any frame i; in that case, the k-th column of ${}^{i}\boldsymbol{J}_n$ is given by

$${}^{i}J_{nk} + \begin{bmatrix} \xi_k {}^{i}z_k + \bar{\xi}_k {}^{i}R_k S({}^{k}z_k) {}^{k}p_n \\ \bar{\xi}_k {}^{i}z_k \end{bmatrix} \tag{19.58}$$

where ${}^{k}\boldsymbol{p}_n = {}^{k}\boldsymbol{p}_{kn}$. In view of the expression of ${}^{k}z_k = [0\ 0\ 1]$, Equation (19.58) can be rewritten as

$${}^{i}\boldsymbol{J}_{nk} = \begin{bmatrix} \xi_k {}^{i}\boldsymbol{z}_k + \bar{\xi}_k (-{}^{k}p_{ny} {}^{i}\boldsymbol{x}_k + {}^{k}p_{nx} {}^{i}\boldsymbol{y}_k) \\ \bar{\xi}_k {}^{i}\boldsymbol{z}_k \end{bmatrix} \tag{19.59}$$

where ${}^{k}p_{nx}$ and ${}^{k}p_{ny}$ are the x and y components of ${}^{k}\boldsymbol{p}_n$. A number of remarks are in order.

- The transformation of the Jacobian from frame i to a different frame l can be obtained as

$${}^{l}\boldsymbol{J}_n = \begin{bmatrix} {}^{l}\boldsymbol{R}_i & 0 \\ 0 & {}^{l}\boldsymbol{R}_i \end{bmatrix} {}^{i}\boldsymbol{J}_n. \tag{19.60}$$

- The Jacobian relating the end-effector velocity to the joint velocities can be computed either by using (19.57) and replacing $\boldsymbol{p}_{kn}$ with $\boldsymbol{p}_{ke}$, or by using the relationship

$$ {}^{i}\boldsymbol{J}_{e} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{S}\left({}^{i}\boldsymbol{p}_{ne}\right) \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} {}^{i}\boldsymbol{J}_{n}. \tag{19.61} $$

A Jacobian ${}^{i}\boldsymbol{J}_{n}$ can be decomposed as the product of three matrices, where the first two are full-rank, while the third one has the same rank as ${}^{i}\boldsymbol{J}_{n}$ but contains simpler elements to compute. To achieve this, the Jacobian of link n can be expressed as a function of a generic Jacobian

$$ J_{n,h} = \begin{bmatrix} \xi_1 z_1 + \bar{\xi}_1 \left(z_1 \times p_{1h}\right) & \cdots & \xi_n z_n + \bar{\xi}_n \left(z_n \times p_{nh}\right) \\ \bar{\xi}_1 z_1 & \cdots & \bar{\xi}_n z_n \end{bmatrix} \tag{19.62} $$

giving the velocity of a frame fixed to link n attached instantaneously to frame h. Then $\boldsymbol{J}_{\mathrm{n}}$ can be computed via (19.61) as

$$ \boldsymbol{J}_{n} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{S}\left(\boldsymbol{p}_{hn}\right) \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \boldsymbol{J}_{n,h} \tag{19.63} $$

which can be expressed with respect to frame i, giving

$$ {}^{i}\boldsymbol{J}_{n} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{S}\left({}^{i}\boldsymbol{R}_{h}\,{}^{h}\boldsymbol{p}_{n}\right) \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} {}^{i}\boldsymbol{J}_{n,h}. \tag{19.64} $$

Combining (19.60) with (19.64) yields the result that the matrix ${}^{l}\boldsymbol{J}_{n}$ can be computed as the product of three matrices

$$ {}^{l}\boldsymbol{J}_{n} = \begin{bmatrix} {}^{l}\boldsymbol{R}_{i} & \boldsymbol{O} \\ \boldsymbol{O} & {}^{l}\boldsymbol{R}_{i} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{S}\left({}^{i}\boldsymbol{R}_{h}\,{}^{h}\boldsymbol{p}_{n}\right) \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} {}^{i}\boldsymbol{J}_{n,h}, \tag{19.65} $$

where remarkably the first two matrices are full-rank. In general, the values of h and i leading to the Jacobian ${}^{i}\boldsymbol{J}_{n,h}$ of simplest expression are given by

$$ i = \mathrm{int}(n/2) \qquad h = \mathrm{int}(n/2) + 1. $$

Hence, for a robot with six degrees of freedom, the matrix ${}^{3}\boldsymbol{J}_{6,4}$ is expected to have the simplest expression; if the wrist is spherical ($\boldsymbol{p}_{46} = 0$), then the second matrix in (19.65) is identity and ${}^{3}\boldsymbol{J}_{6,4} = {}^{3}\boldsymbol{J}_{6}$.

As an example, the geometric Jacobian for the anthropomorphic robot in Figure 19.4 can be computed on the basis of the matrix

$$ {}^{3}\boldsymbol{J}_{6} = \begin{bmatrix} 0 & \ell_3 s_3 - d_4 & -d_4 & 0 & 0 & 0 \\ 0 & \ell_3 c_3 & 0 & 0 & 0 & 0 \\ -\ell_3 c_2 + d_4 s_{23} & 0 & 0 & 0 & 0 & 0 \\ s_{23} & 0 & 0 & 0 & s_4 & -c_4 s_5 \\ c_{23} & 0 & 0 & 1 & 0 & c_5 \\ 0 & 1 & 1 & 0 & c_4 & s_4 s_5 \end{bmatrix}. \tag{19.66} $$

19.5.2 Analytical Jacobian

If the end-effector position and orientation are specified in terms of a minimum number of parameters in the task space as in (19.36), it is possible to also compute the Jacobian matrix by direct differentiation of the direct kinematics equation, i.e.,

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \dot{\boldsymbol{\varphi}}_e \end{bmatrix} = \boldsymbol{J}_a(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{19.67}$$

where the matrix $\boldsymbol{J}_a(\boldsymbol{q}) = \partial \boldsymbol{k}/\partial \boldsymbol{q}$ is termed *analytical Jacobian*.

The relationship between the analytical Jacobian and the geometric Jacobian is expressed as

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{T}(\boldsymbol{\varphi}_e) \end{bmatrix} = \boldsymbol{T}_a(\boldsymbol{\varphi}_e)\boldsymbol{J}_a, \tag{19.68}$$

where $\boldsymbol{T}(\boldsymbol{\varphi}_e)$ is the transformation matrix defined in (19.55) that depends on the particular set of Euler angles used to represent end-effector orientation.

It can be easily recognized that the two Jacobians are in general different; note, however, that the two coincide for the positioning part. 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 the focus. It is always possible to pass from one Jacobian to the other, except when the transformation matrix is singular. The orientations at which the determinant of $\boldsymbol{T}(\boldsymbol{\varphi}_e)$ vanishes are called *representation singularities* of $\boldsymbol{\varphi}_e$. For instance, with reference to the *XYZ* representation in (19.18), the transformation matrix is

$$\boldsymbol{T}(\boldsymbol{\varphi}_e) = \begin{bmatrix} 1 & 0 & s_\beta \\ 0 & c_\alpha & -s_\alpha c_\beta \\ 0 & s_\alpha & c_\alpha c_\beta \end{bmatrix}. \tag{19.69}$$

$\boldsymbol{T}$ becomes singular at the representation singularities $\beta = \pm \pi/2;$ notice that, in these configurations, it impossible to describe an arbitrary angular velocity with a set of Euler angle time derivatives. It should be remarked that each of the other Euler angle descriptions suffers from the occurrence of two representation singularities.

19.5.3 Singularities

The differential kinematics Equation (19.56) defines a linear mapping between the vector of joint velocities $\dot{\boldsymbol{q}}$ and the vector of end-effector velocities $\boldsymbol{v}$. The Jacobian is in general a function of the robot configuration $\boldsymbol{q}$. Those configurations at which $\boldsymbol{J}$ is rank-deficient are called *kinematic singularities*.

The simplest means to find singularities is to compute the determinant of the Jacobian matrix. For instance, for the above Jacobian in (19.66) it is

$$\det\left({}^3\boldsymbol{J}_6\right) = \ell_3 d_4 c_3 s_5 \left(d_4 s_{23} - \ell_3 c_2\right) \tag{19.70}$$

leading to three types of singularities $\left(\ell_3, d_4 \neq 0\right)$: *elbow singularity*

$$c_3 = 0$$

when links 2 and 3 are aligned; *shoulder singularity*

$$d_4 s_{23} - \ell_3 c_2 = 0$$

when the origin of frame 4 is along axis Z_0; and *wrist singularity*

$$s_5 = 0$$

when axes Z_4 and Z_6 are aligned. Notice that elbow singularity is not troublesome because it occurs at the boundary of the robot workspace $\left(q_3 = \pm\pi/2\right)$. Shoulder singularity is characterized in the task space and thus it can be avoided when planning an end-effector trajectory. Instead, wrist singularity is characterized in the joint space $\left(q_5 = 0,\ \pi\right)$, and thus it is difficult to predict when planning an end-effector trajectory.

An effective tool for analyzing the linear mapping from the joint velocity space into the task velocity space defined by (19.56) is offered by the singular value decomposition (SVD) of the Jacobian matrix is given by

$$\boldsymbol{J} = \boldsymbol{U}\,\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} = \sum_{i=1}^{m} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\boldsymbol{T}}, \tag{19.71}$$

where $\boldsymbol{U}$ is the $(m \times m)$ matrix of the output singular vectors $\boldsymbol{u}_i$, $\boldsymbol{V}$ is the $(n \times n)$ matrix of the input singular vectors $\boldsymbol{v}_i$, and $\Sigma = [\boldsymbol{S} \quad \boldsymbol{O}]$ is the $(m \times n)$ matrix whose $(m \times m)$ diagonal submatrix $\boldsymbol{S}$ contains the singular values σ_i of the matrix $\boldsymbol{J}$. If r denotes the rank of $\boldsymbol{J}$, the following properties hold:

- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_m = 0$,
- $\boldsymbol{R}(\boldsymbol{J}) = \text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$,
- $\boldsymbol{N}(\boldsymbol{J}) = span\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$.

The null space $\boldsymbol{N}(\boldsymbol{J})$ is the set of joint velocities that yield null task velocities at the current configuration; these joint velocities are termed *null space joint velocities*. A base of $\boldsymbol{N}(\boldsymbol{J})$ is given by the $(n - r)$ last input singular vectors, which represent independent linear combinations of the joint velocities. Hence, one effect of a singularity is to increase the dimension of $\boldsymbol{N}(\boldsymbol{J})$ by introducing a linear combination of joint velocities that produce a null task velocity.

The range space $\boldsymbol{R}(\boldsymbol{J})$ is the set of task velocities that can be obtained as a result of all possible joint velocities; these task velocities are termed *feasible space task velocities*. A base of $\boldsymbol{R}(\boldsymbol{J})$ is given by the first r output singular vectors that represent independent linear combinations of the single components of task velocities. Accordingly, another effect of a singularity is to decrease the dimension of $\boldsymbol{R}(\boldsymbol{J})$ by eliminating a linear combination of task velocities from the space of feasible velocities.

The singular value decomposition (19.71) shows that the i-th singular value of $\boldsymbol{J}$ can be viewed as a gain factor relating the joint velocity along the $\boldsymbol{v}_i$ direction to the task velocity along the $\boldsymbol{u}_i$ direction. When a singularity is approached, the r-th singular value tends to zero and the task velocity produced by a fixed joint velocity along $\boldsymbol{v}_r$ is decreased proportionally to s_r. At the singular configuration, the joint velocity along $\boldsymbol{v}_r$ is in the null space and the task velocity along $\boldsymbol{u}_r$ becomes infeasible.

In the general case, the joint velocity has components in any $\boldsymbol{v}_i$ direction, and the resulting task velocity can be obtained as a combination of the single components along each output singular vector direction.

19.6 Differential Kinematics Inversion

The differential kinematics equation, in terms of either the geometric or the analytical Jacobian, establishes a linear mapping between joint space velocities and task space velocities, even if the Jacobian is a function of joint configuration. This feature suggests the use of the differential kinematics Equation (19.56) to solve the inverse kinematics problem.

Assume that a task space trajectory is given ($\boldsymbol{x}(t)$, $\boldsymbol{v}(t)$). The goal is to find a feasible joint space trajectory $\left(\boldsymbol{q}(t),\dot{\boldsymbol{q}}(t)\right)$ that reproduces the given trajectory. Joint velocities can be obtained by solving the differential kinematics equation for $\dot{\boldsymbol{q}}$ at the current joint configuration; then, joint positions $\boldsymbol{q}(t)$ can be computed by integrating the velocity solution over time with known initial conditions. This approach is based on knowledge of the robot Jacobian and is applicable to any robot structure, on the condition that a suitable inverse for the matrix $\boldsymbol{J}$ can be found.

19.6.1 Pseudoinverse

With reference to the geometric Jacobian, the basic inverse solution to (19.56) is obtained by using the *pseudoinverse* $\boldsymbol{J}^{\dagger}$ of the matrix $\boldsymbol{J}$; this is a unique matrix satisfying the Moore-Penrose conditions

$$\boldsymbol{J}\boldsymbol{J}^{\dagger}\boldsymbol{J}=\boldsymbol{J} \qquad \boldsymbol{J}^{\dagger}\boldsymbol{J}\boldsymbol{J}^{\dagger}=\boldsymbol{J}^{\dagger}$$

$$\left(\boldsymbol{J}\boldsymbol{J}^{\dagger}\right)^{\mathbf{T}}=\boldsymbol{J}\boldsymbol{J}^{\dagger} \qquad \left(\boldsymbol{J}^{\dagger}\boldsymbol{J}\right)^{\mathbf{T}}=\boldsymbol{J}^{\dagger}\boldsymbol{J} \tag{19.72}$$

or, alternatively, the equivalent conditions

$$\begin{aligned} \boldsymbol{J}^{\dagger}\boldsymbol{a}&=\boldsymbol{a} && \forall \boldsymbol{a}\in\boldsymbol{N}^{\perp}(\boldsymbol{J}) \\ \boldsymbol{J}^{\dagger}\boldsymbol{b}&=0 && \forall \boldsymbol{b}\in\boldsymbol{R}^{\perp}(\boldsymbol{J}) \\ \boldsymbol{J}^{\dagger}(\boldsymbol{a}+\boldsymbol{b})&=\boldsymbol{J}^{\dagger}\boldsymbol{a}+\boldsymbol{J}^{\dagger}\boldsymbol{b} && \forall \boldsymbol{a}\in\boldsymbol{R}(\boldsymbol{J}),\forall \boldsymbol{b}\in\boldsymbol{R}^{\perp}(\boldsymbol{J}). \end{aligned} \tag{19.73}$$

The inverse solution can then be written as

$$\dot{\boldsymbol{q}}=\boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{v} \tag{19.74}$$

that provides a least-squares solution with minimum norm to Equation (19.56); in detail, solution (19.74) satisfies the condition

$$\min_{\dot{\boldsymbol{q}}}\|\dot{\boldsymbol{q}}\| \tag{19.75}$$

of all $\dot{\boldsymbol{q}}$ that fulfill

$$\min_{\dot{\boldsymbol{q}}}\|\boldsymbol{v}-\boldsymbol{J}\dot{\boldsymbol{q}}\|. \tag{19.76}$$

If the Jacobian matrix is full-rank, the right pseudoinverse of $\boldsymbol{J}$ can be computed as

$$\boldsymbol{J}^{\dagger}=\boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}\right)^{-1}, \tag{19.77}$$

and (19.74) provides an exact solution to (19.56). Further, if $\boldsymbol{J}$ square, the pseudoinverse (19.77) reduces to the standard inverse Jacobian matrix $\boldsymbol{J}^{-1}$.

To gain insight into the properties of the inverse mapping described by (19.74), it is useful to consider the singular value decomposition (19.71) of $\boldsymbol{J}$, and thus

$$\boldsymbol{J}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^{T} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^{T} \tag{19.78}$$

where r denotes the rank of $\boldsymbol{J}$. The following properties hold:

- $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_m = 0$,
- $\boldsymbol{R}(\boldsymbol{J}^{\dagger}) = \boldsymbol{N}^{\perp}(\boldsymbol{J}) = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$,
- $\boldsymbol{N}(\boldsymbol{J}^{\dagger}) = \boldsymbol{R}^{\perp}(\boldsymbol{J}) = \text{span}\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_n\}$.

The null space $\boldsymbol{N}(\boldsymbol{J}^{+})$ is the set of task velocities that yields null joint space velocities at the current configuration; these task velocities belong to the orthogonal complement of the feasible space task velocities. Hence, one effect of the pseudoinverse solution (19.74) is to filter the infeasible components of the given task velocities while allowing exact tracking of the feasible components; this is due to the minimum norm property (19.75).

The range space $\boldsymbol{R}(\boldsymbol{J}^{\dagger})$ is the set of joint velocities that can be obtained as a result of all possible task velocities. Because these joint velocities belong to the orthogonal complement of the null space joint velocities, the pseudoinverse solution (19.74) satisfies the least-squares condition (19.76).

If a task velocity is assigned along $\boldsymbol{u}_i$, the corresponding joint velocity computed via (19.74) lies along $\boldsymbol{v}_i$ and is magnified by the factor $1/\sigma_i$. When a singularity is approached, the r-th singular value tends to zero and a fixed task velocity along $\boldsymbol{u}_r$ requires large joint velocities. At a singular configuration, the $\boldsymbol{u}_r$ direction becomes infeasible and $\boldsymbol{v}_r$ adds to the set of null space velocities of the robot.

19.6.2 Redundancy

For a kinematically redundant robot a nonempty null space $\boldsymbol{N}(\boldsymbol{J})$ exists which is available to set up systematic procedures for an effective handling of redundant degrees of freedom. The general inverse solution can be written as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{v} + \left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_0 \tag{19.79}$$

which satisfies the least-squares condition (19.76) but loses the minimum norm property (19.75) by virtue of the addition of the homogeneous term $\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)\dot{\boldsymbol{q}}_0$. The matrix $\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)$ is a projector of the joint vector $\dot{\boldsymbol{q}}_0$ onto $\boldsymbol{N}(\boldsymbol{J})$.

In terms of the singular value decomposition, solution (19.79) can be written in the form

$$\dot{\boldsymbol{q}} = \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{v} + \sum_{i=r+1}^{m} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \dot{\boldsymbol{q}}_0 + \sum_{i=m+1}^{n} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \dot{\boldsymbol{q}}_0. \tag{19.80}$$

Three contributions can be recognized in (19.80), namely, the least-squares joint velocities, the null space joint velocities due to singularities (if $r < m$), and the null space joint velocities due to redundant degrees of freedom (if $m < n$).

This result is of fundamental importance for redundancy resolution, because solution (19.79) evidences the possibility of choosing the vector $\dot{\boldsymbol{q}}_0$ to exploit the redundant degrees of freedom.

In fact, the contribution of $\dot{\boldsymbol{q}}_0$ is to generate null space motions of the structure that do not alter the task space configuration but allow the robot to reach more dexterous postures for the execution of the given task.

A typical choice of the null space joint velocity vector is

$$\dot{\boldsymbol{q}}_0 = \alpha\left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^{\mathrm{T}} \tag{19.81}$$

with $\alpha > 0$; $w(\boldsymbol{q})$ is a scalar objective function of the joint variables, and $\left(\partial w(\boldsymbol{q}) / \partial \boldsymbol{q}\right)^{\mathrm{T}}$ is the vector function representing the gradient of w. In this way, locally optimizing w in accordance with the kinematic constraint expressed by (19.56) is sought. Usual objective functions are

- The *manipulation* measure defined as

$$w(\boldsymbol{q}) = \sqrt{\det\left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})\right)}, \tag{19.82}$$

  which vanishes at a singular configuration, and thus redundancy may be exploited to escape singularities.
- The *distance from mechanical joint limits* defined as

$$w(\boldsymbol{q}) = -\frac{1}{2n}\sum_{i=1}^{n}\left(\frac{q_i - \overline{q}_i}{q_{iM} + q_{im}}\right)^2, \tag{19.83}$$

  where q_{iM} (q_{im}) denotes the maximum (minimum) limit for q_i and $\overline{q}_i$ the middle of the joint range, and thus redundancy may be exploited to keep the robot from joint limits.
- The *distance from an obstacle* defined as

$$w(\boldsymbol{q}) = \min_{\boldsymbol{p},\boldsymbol{o}} \left\| \boldsymbol{p}(\boldsymbol{q}) - \boldsymbol{o} \right\|, \tag{19.84}$$

  where $\boldsymbol{o}$ is the position vector of an opportune point on the obstacle and $\boldsymbol{p}$ is the position vector of the closest robot point to the obstacle, and thus redundancy may be exploited to avoid collisions with obstacles.

19.6.3 Damped Least-Squares Inverse

In the neighborhood of singular configurations the use of a pseudoinverse is not adequate and a numerically robust solution is achieved by the *damped least-squares inverse* technique based on the solution to the modified differential kinematics equation

$$\boldsymbol{J}^{\mathrm{T}}\boldsymbol{v} = \left(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda^2\boldsymbol{I}\right)\dot{\boldsymbol{q}} \tag{19.85}$$

in place of Equation (19.56); in (19.85) the scalar λ is the so-called *damping factor*. Note that when $\lambda = 0$, Equation (19.85) reduces to (19.56).

The solution to (19.85) can be written in either of the equivalent forms

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\mathrm{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}} + \lambda^2\boldsymbol{I}\right)^{-1}\boldsymbol{v} \tag{19.86}$$

$$\dot{\boldsymbol{q}} = \left(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda^2\boldsymbol{I}\right)^{-1}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{\upsilon}. \tag{19.87}$$

The computational load of (19.86) is lower than that of (19.87), being usually $n \geq m$. Let

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\boldsymbol{\upsilon} \tag{19.88}$$

indicate the damped least-squares inverse solution computed with either of the above forms. Solution (19.88) satisfies the condition

$$\min_{\dot{\boldsymbol{q}}} \left\| \boldsymbol{\upsilon} - \boldsymbol{J}\dot{\boldsymbol{q}} \right\|^2 + \lambda^2 \left\| \dot{\boldsymbol{q}} \right\|^2 \tag{19.89}$$

that gives a trade-off between the least-squares condition (19.76) and the minimum norm condition (19.75). In fact, condition (19.89) accounts for both accuracy and feasibility in choosing the joint space velocity $\dot{\boldsymbol{q}}$ required to produce the given task space velocity $\boldsymbol{\upsilon}$. In this regard, it is essential to select a suitable value for the damping factor; small values of λ give accurate solutions but low robustness in the neighborhood of singular configurations, while large values of λ result in low tracking accuracy even if feasible and accurate solutions are possible.

Resorting to the singular value decomposition, the damped least-squares inverse solution (19.88) can be written as

$$\dot{\boldsymbol{q}} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}\boldsymbol{v}_i\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{\upsilon}. \tag{19.90}$$

Remarkably, it is

- $\boldsymbol{R}\left(\boldsymbol{J}^{\#}\right) = \boldsymbol{R}\left(\boldsymbol{J}^{\dagger}\right) = \boldsymbol{N}^{\perp}(\boldsymbol{J}) = \mathrm{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r\}$,
- $\boldsymbol{N}\left(\boldsymbol{J}^{\#}\right) = \boldsymbol{N}\left(\boldsymbol{J}^{\dagger}\right) = \boldsymbol{R}^{\perp}(\boldsymbol{J}) = \mathrm{span}\{\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_n\}$,

that is, the structural properties of the damped least-squares inverse solution are analogous to those of the pseudoinverse solution.

It is clear that with respect to the pure least-squares solution (19.74) the components for which $\sigma_i \gg \lambda$ are not influenced by the damping factor, because in this case it is

$$\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \approx \frac{1}{\sigma_i}. \tag{19.91}$$

On the other hand, when a singularity is approached, the smallest singular value tends to zero while the associated component of the solution is driven to zero by the factor σ_i/λ^2; this progressively reduces the joint velocity to achieve near-degenerate components of the commanded task velocity. At the singularity, solutions (19.88) and (19.74) behave identically as long as the remaining singular values are significantly larger than the damping factor. Note that an upper bound of $1/2\lambda$ is set on the magnification factor relating the task velocity component along $\boldsymbol{u}_i$ to the resulting joint velocity along $\boldsymbol{v}_i$; this bound is reached when $\sigma_i = \lambda$.

The damping factor λ determines the degree of approximation introduced with respect to the pure least-squares solution. Then, using a constant value for λ may turn out to be inadequate for obtaining good performance over the entire robot workspace. An effective choice is to adjust λ as a function of some measure of closeness to the singularity at the current configuration of the robot. To this purpose, manipulability measures or estimates of the smallest singular value can be adopted.

Remarkably, currently available microprocessors even allow real-time computation of full singular-value decomposition.

A singular region can be defined on the basis of the smallest singular value estimate of $\boldsymbol{J}$. Outside the region the exact solution is used, while inside the region a configuration-varying damping factor is introduced to obtain the desired approximate solution. The factor must be chosen so that continuity of joint velocity $\dot{\boldsymbol{q}}$ is ensured in the transition at the border of the singular region.

Without loss of generality, for a six-degree-of-freedom robot, the damping factor can be selected according to the following law:

$$\lambda^2 = \begin{cases} 0 & \hat{\sigma}_6 \geq \varepsilon \\ \left(1 - \left(\frac{\hat{\sigma}_6}{\varepsilon}\right)^2\right)\lambda_{\max}^2 & \hat{\sigma}_6 < \varepsilon, \end{cases} \tag{19.92}$$

where $\hat{\sigma}_6$ is the smallest singular value estimate, and ε defines the size of the singular region; the value of $\lambda_{\max}$ is at the user's disposal to suitably shape the solution in the neighborhood of a singularity.

Equation (19.92) requires computation of the smallest singular value. To avoid a full singular-value decomposition, we can resort to a recursive algorithm to find an estimate of the smallest singular value. Suppose that an estimate $\hat{\boldsymbol{v}}_6$ of the last input singular vector is available, so that $\hat{\boldsymbol{v}}_6 \approx \boldsymbol{v}_6$ and $\left\|\hat{\boldsymbol{v}}_6\right\|$. This estimate is used to compute the vector $\hat{\boldsymbol{v}}_6'$ from

$$\left(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda^2\boldsymbol{I}\right)\hat{\boldsymbol{v}}_6' = \hat{\boldsymbol{v}}_6. \tag{19.93}$$

Then the square of the estimate $\hat{\sigma}_6$ of the smallest singular value can be found as

$$\hat{\sigma}_6^2 = \frac{1}{\left\|\hat{\boldsymbol{v}}_6'\right\|} - \lambda^2, \tag{19.94}$$

while the estimate of $\boldsymbol{v}_6$ is updated using

$$\hat{\boldsymbol{v}}_6 = \frac{\hat{\boldsymbol{v}}_6'}{\left\|\hat{\boldsymbol{v}}_6'\right\|}. \tag{19.95}$$

The above estimation scheme is based on the assumption that $\boldsymbol{v}_6$ is slowly rotating, which is normally the case. However, if the robot is close to a double singularity (e.g., a shoulder and a wrist singularity for the anthropomorphic robot), the vector $\boldsymbol{v}_6$ will instantaneously rotate if the two smallest singular values cross. The estimate of the smallest singular value will then track σ_5 initially, before $\hat{\boldsymbol{v}}_6$ converges again to $\boldsymbol{v}_6$. Therefore, it is worth extending the scheme by estimating not only the smallest but also the second smallest singular value. Assume that the estimates $\hat{\boldsymbol{v}}_6$ and $\hat{\sigma}_6$ are available and define the matrix

$$\boldsymbol{M} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda^2\boldsymbol{I} - \left(\hat{\sigma}_6^2 + \lambda^2\right)\hat{\boldsymbol{v}}_6\hat{\boldsymbol{v}}_6^{\mathrm{T}}. \tag{19.96}$$

With this choice, the second smallest singular value of $\boldsymbol{J}$ plays in

$$\boldsymbol{M}\hat{\boldsymbol{v}}_5' = \hat{\boldsymbol{v}}_5 \tag{19.97}$$

the same role as σ_6 in (19.93) and then will provide a convergent estimate of $\hat{\boldsymbol{v}}_5$ to $\boldsymbol{v}_5$ and $\hat{\sigma}_5$ to σ_5.

At this point, suppose that $\hat{\boldsymbol{v}}_5$ is an estimate of $\boldsymbol{v}_5$ so that $\hat{\boldsymbol{v}}_5 \approx \boldsymbol{v}_5$ and $\|\hat{\boldsymbol{v}}_5\| = 1$. This estimate is used to compute $\hat{\boldsymbol{v}}_5'$ from (19.97). Then, an estimate of the square of the second smallest singular value of $\boldsymbol{J}$ is found from

$$\hat{\sigma}_5^2 = \frac{1}{\|\hat{\boldsymbol{v}}_5'\|} - \lambda^2, \tag{19.98}$$

and the estimate of $\boldsymbol{v}_5$ is updated using

$$\hat{\boldsymbol{v}}_5 = \frac{\hat{\boldsymbol{v}}_5'}{\|\hat{\boldsymbol{v}}_5'\|}. \tag{19.99}$$

On the basis of this modified estimation algorithm, crossing of singularities can be effectively detected; also, by switching the two singular values and the associated estimates $\hat{\boldsymbol{v}}_5$ and $\hat{\boldsymbol{v}}_6$, the estimation of the smallest singular value will be accurate even when the two smallest singular values cross.

19.6.4 User-Defined Accuracy

The above damped least-squares inverse method achieves a compromise between accuracy and robustness of the solution. This is performed without specific regard to the components of the particular task assigned to the robot's end-effector. The *user-defined accuracy* strategy based on the weighted, damped, least-squares inverse method allows discriminating between directions in the task space where higher accuracy is desired and directions where lower accuracy can be tolerated. This is the case, for instance, of spot welding or spray painting in which the tool angle about the approach direction is not essential to the fulfillment of the task.

Let a weighted end-effector velocity vector be defined as

$$\bar{\boldsymbol{v}} = \boldsymbol{W}\boldsymbol{v} \tag{19.100}$$

where $\boldsymbol{W}$ is the $(m \times m)$ task-dependent weighting matrix taking into account the anisotropy of the task requirements. Substituting (19.100) into (19.56) gives

$$\bar{\boldsymbol{v}} = \bar{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{19.101}$$

where $\bar{\boldsymbol{J}} = \boldsymbol{W}\boldsymbol{J}$. It is worth noticing that if $\boldsymbol{W}$ is full-rank, solving (19.56) is equivalent to solving (19.101), but with different conditioning of the system of equations to solve. This suggests selecting only the strictly necessary weighting action to avoid undesired ill-conditioning of $\bar{\boldsymbol{J}}$.

Equation (19.101) can be solved by using the weighted, damped, least-squares inverse technique, i.e.,

$$\bar{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{q})\bar{\boldsymbol{v}} = \left(\bar{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{q})\bar{\boldsymbol{J}}(\boldsymbol{q}) + \lambda^2\boldsymbol{I}\right)\dot{\boldsymbol{q}}. \tag{19.102}$$

Again, the singular value decomposition of the matrix $\bar{\boldsymbol{J}}$ is helpful, i.e.,

$$\bar{\boldsymbol{J}} = \sum_{i=1}^{r} \bar{\sigma}_i \bar{\boldsymbol{u}}_i \bar{\boldsymbol{v}}_i^{\mathrm{T}} \tag{19.103}$$

and the solution to (19.102) can be written as

$$\dot{\boldsymbol{q}} = \sum_{i=1}^{r} \frac{\bar{\sigma}_i}{\bar{\sigma}_i^2 + \lambda^2} \bar{\boldsymbol{v}}_i \bar{\boldsymbol{u}}_i^{\mathrm{T}} \boldsymbol{x}_d. \tag{19.104}$$

It is clear that the singular values $\bar{\sigma}_i$ and the singular vectors $\bar{\boldsymbol{u}}_i$ and $\bar{\boldsymbol{v}}_i$ depend on the choice of the weighting matrix $\boldsymbol{W}$. While this has no effect on the solution $\dot{\boldsymbol{q}}$ as long as $\bar{\sigma}_r \gg \lambda$, close to singularities where $\bar{\sigma}_r \ll \lambda$, for some $r < m$, the solution can be shaped by properly selecting the matrix $\boldsymbol{W}$.

For a six-degree-of-freedom robot with a spherical wrist, it is worthwhile to devise special handling of the wrist singularity, because such a singularity is difficult to predict at the planning level in the task space. It can be recognized that at the wrist singularity only two components of the angular velocity vector can be generated by the wrist itself. The remaining component might be generated by the inner joints, although at the expense of loss of accuracy along some other task space directions. For this reason, lower weight should be put on the angular velocity component that is infeasible to the wrist. For the anthropomorphic robot, this is easily expressed in the frame attached to link 4; let $\boldsymbol{R_4}$ denote the rotation matrix describing orientation of this frame with respect to the base frame so that the infeasible component is aligned with the X-axis. Then the weighting matrix can be chosen as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{4\mathrm{diag}}\{w,1,1\}\boldsymbol{R}_4^T \end{bmatrix}. \tag{19.105}$$

Similar, to the choice of the damping factor as in (19.92), the weighting factor w is selected according to the following expression:

$$(1-w)^2 = \begin{cases} 0 & \hat{\sigma}_6 \geq \varepsilon \\ \left(1 - \left(\dfrac{\hat{\sigma}_6}{\varepsilon}\right)^2\right)\left(1 - w_{\min}\right)^2 & \hat{\sigma}_6 < \varepsilon, \end{cases} \tag{19.106}$$

where $w_{\min} > 0$ is a design parameter.

19.7 Inverse Kinematics Algorithms

The differential kinematics equation has been utilized above to solve for joint velocities. Open-loop reconstruction of joint variables through numerical integration unavoidably leads to solution drift and then to task space errors. This drawback can be overcome by devising a closed-loop *inverse kinematics algorithm* based on the task space error between the desired and actual end-effector locations $\boldsymbol{x}_d$ and $\boldsymbol{x}$, i.e., $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}(\boldsymbol{q})$. It is also worth considering the differential kinematics equation in the form (19.67) where the definition of the task error has required consideration of the analytical Jacobian $\boldsymbol{J_a}$ in lieu of the geometric Jacobian.

19.7.1 Jacobian Pseudoinverse

The joint velocity vector should be chosen so that the task error tends to zero. The simplest algorithm is obtained by using the *Jacobian pseudoinverse*

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}\right), \tag{19.107}$$

which plugged into (19.67) gives

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{0}. \tag{19.108}$$

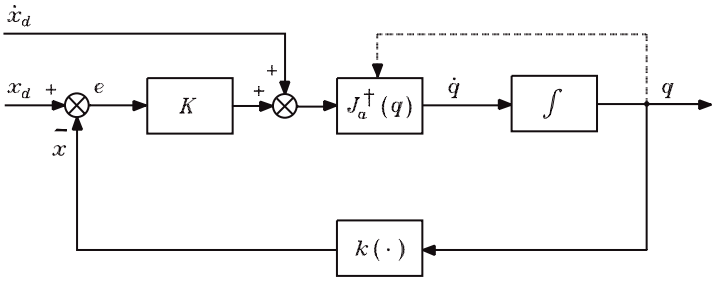

FIGURE 19.5 Block scheme of the inverse kinematics algorithm with the Jacobian pseudoinverse.

If $\boldsymbol{K}$ is a positive definite (diagonal) matrix, the linear system (19.108) is asymptotically stable; the tracking error along the given trajectory converges to zero with a rate depending on the eigenvalues of $\boldsymbol{K}$.

A block scheme of the inverse kinematics algorithm based on the Jacobian pseudoinverse is illustrated in Figure 19.5.

If it is desired to exploit redundant degrees of freedom, solution (19.107) can be generalized to

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_a^\dagger(\boldsymbol{q})\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}\right) + \left(\boldsymbol{I} - \boldsymbol{J}_a^\dagger(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_0 \tag{19.109}$$

that logically corresponds to (19.79). In the case of numerical problems in the neighborhood of singularities, the pseudoinverse can be replaced with a suitable damped least-squares inverse.

19.7.2 Jacobian Transpose

A computationally efficient inverse kinematics algorithm can be derived by considering the *Jacobian transpose* in lieu of the pseudoinverse.

Consider the joint velocity vector

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_a^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{K}\boldsymbol{e} \tag{19.110}$$

where $\boldsymbol{K}$ is a symmetric positive definite matrix. A simple Lyapunov argument can be used to analyze the convergence of the algorithm. Consider the positive definite function candidate

$$\boldsymbol{V} = \frac{1}{2}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{e}; \tag{19.111}$$

its time derivative along the trajectories of the system (19.67) and (19.110) is

$$\dot{\boldsymbol{V}} = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{K}\dot{\boldsymbol{x}}_d - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{J}_a(\boldsymbol{q})\boldsymbol{J}_a^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{K}\boldsymbol{e}. \tag{19.112}$$

If $\dot{\boldsymbol{x}}_d = \boldsymbol{0}$, it is easy to see that $\dot{\boldsymbol{V}}$ is negative definite as long as $\boldsymbol{J}_a$ is full-rank, and then it can be concluded that $\boldsymbol{e} = \boldsymbol{0}$ is an asymptotically stable equilibrium point for the system (19.67) and (19.110) as long as $\boldsymbol{J}_a$ is full-rank for all joint configurations $\boldsymbol{q}$. A number of remarks are in order.

- If $\dot{\boldsymbol{x}}_d \neq \boldsymbol{0}$, only boundedness of tracking errors can be established; an estimate of the bound is given by

$$\|\boldsymbol{e}\|_{\max} = \frac{\|\dot{\boldsymbol{x}}_d\|_{\max}}{k\sigma_r^2(\boldsymbol{J}_a)} \tag{19.113}$$

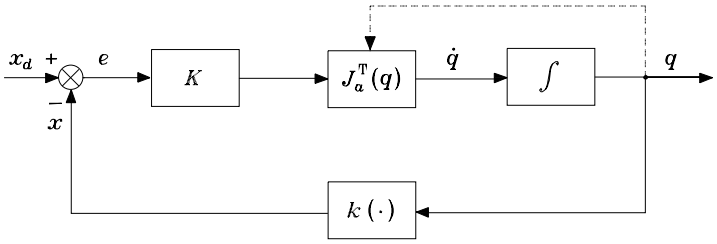

FIGURE 19.6 Block scheme of the inverse kinematics algorithm with Jacobian transpose.

where $\boldsymbol{K}$ has been conveniently chosen as a diagonal matrix $\boldsymbol{K} = k\boldsymbol{I}$. It is anticipated that k can be increased to diminish the errors, but, in practice, upper bounds exist due to discrete-time implementation of the algorithm.

- When a singularity is encountered, $N\left(\boldsymbol{J}_a^T\right)$ is nonempty and $\dot{V}$ is only semi-definite; $\dot{V} = 0$ for $\boldsymbol{e} \neq \boldsymbol{0}$ with $\boldsymbol{Ke} \in N\left(\boldsymbol{J}_a^T\right)$, and the algorithm may get stuck. It can be shown, however, that such an equilibrium point is unstable as long as $\dot{\boldsymbol{x}}_d$ drives $\boldsymbol{Ke}$ outside $N\left(\boldsymbol{J}_a^T\right)$. An enhancement of the algorithm can be achieved by rendering the matrix $\boldsymbol{J}_a^T\boldsymbol{K}$ less sensitive to variations of joint configurations along the task trajectory. This is accomplished by choosing a configuration-dependent $\boldsymbol{K}$ that compensates for variations of $\boldsymbol{J}_a$.

A block scheme of the inverse kinematics algorithm based on the Jacobian transpose is illustrated in Figure 19.6.

The most attractive feature of the Jacobian transpose algorithm is certainly the need of computing only direct kinematics functions $\boldsymbol{k}(\boldsymbol{q})$ and $\boldsymbol{J}_a(\boldsymbol{q})$. Further insight into the performance of solution (19.110) can be gained by considering the singular value decomposition of the Jacobian transpose, and thus

$$\boldsymbol{J}^T = \sum_{i=1}^{m} \sigma_i \boldsymbol{v}_i \boldsymbol{u}_i^T \tag{19.114}$$

which reveals continuous, smooth behavior of the solution close and through singular configurations. Note that in (19.114) the geometric Jacobian has been considered, and it has been assumed that no representation singularities are introduced.

19.7.3 Use of Redundancy

In case of redundant degrees of freedom, it is possible to combine 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. The idea is to introduce an additional constraint task by specifying a $(p \times 1)$ vector $\boldsymbol{x}_c$ as a function of the robot joint variables, i.e.,

$$\boldsymbol{x}_c = \boldsymbol{k}_c(\boldsymbol{q}), \tag{19.115}$$

with $\boldsymbol{p} \leq n - m$ to constrain at most all the available redundant degrees of freedom. The constraint task vector $\boldsymbol{x}_c$ can be chosen by embedding scalar objective functions of the kind introduced in (19.82)–(19.84).

Differentiating (19.115) with respect to time gives

$$\dot{\boldsymbol{x}}_c = \boldsymbol{J}_c(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{19.116}$$

where $\boldsymbol{J}_c(\boldsymbol{q}) = \partial \boldsymbol{k}_c / \partial \boldsymbol{q}$ is the constraint Jacobian. The result is an augmented differential kinematics equation given by (19.67) and (19.116), based on a Jacobian matrix

$$\boldsymbol{J}' = \begin{bmatrix} \boldsymbol{J}_a \\ \boldsymbol{J}_c \end{bmatrix}. \tag{19.117}$$

When a constraint task is specified independently of the end-effector task, there is no guarantee that the matrix $\boldsymbol{J}'$ remains full-rank along the entire task path. Even if rank $(\boldsymbol{J}_a) = m$ and rank $(\boldsymbol{J}_c) = p$, then rank $(\boldsymbol{J}') = m + p$ if and only if $\boldsymbol{R}(\boldsymbol{J}_a^{\mathrm{T}}) \cap R(\boldsymbol{J}_c^{\mathrm{T}}) = \{\varnothing\}$. Singularities of $\boldsymbol{J}'$ are termed *artificial singularities,* and it can be shown that those are given by singularities of the matrix $\boldsymbol{J}_c\left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a\right)$.

The above discussion suggests that, when solving for joint velocities, a *task priority strategy* is advisable to avoid conflicting situations between the end-effector task and the constraint task. Substituting (19.109) into (19.116) gives

$$\dot{\boldsymbol{x}}_c = \boldsymbol{J}_c(\boldsymbol{q})\boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\,(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) + \boldsymbol{J}_c(\boldsymbol{q})\left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_0 \tag{19.118}$$

which could be solved for $\dot{\boldsymbol{q}}_0$ provided that artificial singularities — those of the matrix $\boldsymbol{J}_c(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a)$ — are avoided. Observing that equality (19.118) can be achieved only for the components of $\dot{\boldsymbol{x}}_c$ belonging to $\boldsymbol{R}(\boldsymbol{J}_c)$, it is sufficient to consider the equation

$$\boldsymbol{J}_c^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{x}}_c = \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\,(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) + \left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)\dot{\boldsymbol{q}}_0 \tag{19.119}$$

that can be solved for $\dot{\boldsymbol{q}}_0$ giving

$$\dot{\boldsymbol{q}}_0 = \left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)^{\dagger}\left(\boldsymbol{J}_c^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{x}}_c - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e})\right). \tag{19.120}$$

By recalling that $(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a)^{\dagger} = (\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a)$, solution (19.120) reduced to the simple form

$$\dot{\boldsymbol{q}}_0 = \left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)\boldsymbol{J}_c^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{x}}_c. \tag{19.121}$$

Folding (19.121) back into (19.109) and exploiting the idempotence of $(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a)$ gives

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) + \left(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}(\boldsymbol{q})\boldsymbol{J}_a(\boldsymbol{q})\right)\boldsymbol{J}_c^{\dagger}(\boldsymbol{q})(\dot{\boldsymbol{x}}_{cd} + \boldsymbol{K}_c\boldsymbol{e}_c) \tag{19.122}$$

where $\boldsymbol{e_c} = \boldsymbol{x}_{cd} - \boldsymbol{x}_c$, $\boldsymbol{x}_{cd}$ being the desired value of the constraint task, and $\boldsymbol{K}_c$ is a positive definite matrix. The operator $(\boldsymbol{I} - \boldsymbol{J}_a^{\dagger}\boldsymbol{J}_a)$ projects the secondary velocity contribution $\dot{\boldsymbol{q}}_0$ on the null space $N(\boldsymbol{J}_a)$, 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 is along the end-effector path.

In the case when $\boldsymbol{J}_c$ becomes singular, a damped least-squares inverse of $\boldsymbol{J}_c$ in lieu of the pseudoinverse in (19.121) can be used. Otherwise, by recalling the Jacobian transpose solution for the end-effector task (19.110), the null space joint velocity vector can be conveniently chosen as

$$\dot{\boldsymbol{q}}_0 = \boldsymbol{J}_c^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{K}_c(\boldsymbol{x}_{cd} - \boldsymbol{x}_c), \tag{19.123}$$

which allows the algorithm to work at a singularity of $\boldsymbol{J}_c$ 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 ($\dot{\boldsymbol{x}}_{cd} = \boldsymbol{0}$), it can be concluded that the solution based on (19.123) performs equally well.

19.7.4 Orientation Errors

The above inverse kinematics algorithms make use of the analytical Jacobian since they operate on error variables (position and orientation) that are defined in the task space. More insight about the implications of different end-effector orientation descriptions can be gained by separating position from orientation components. With reference to the pseudoinverse algorithm based on (19.107), using the geometric Jacobian in lieu of the analytical Jacobian, the solution can be rewritten as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{\upsilon}_p \\ \boldsymbol{\upsilon}_o \end{bmatrix} \tag{19.124}$$

where $\boldsymbol{\upsilon}_p$, $\boldsymbol{\upsilon}_o$ represent two resolved velocities chosen to ensure tracking of the desired end-effector motion. Substituting (19.124) into (19.56) gives

$$\dot{\boldsymbol{p}}_e = \boldsymbol{\upsilon}_p \tag{19.125}$$

$$\boldsymbol{\omega}_e = \boldsymbol{\upsilon}_o \tag{19.126}$$

where the explicit end-effector linear and angular velocities have been separated.

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

$$\boldsymbol{\upsilon}_p = \dot{\boldsymbol{p}}_d + \boldsymbol{K}_p \boldsymbol{e}_p \tag{19.127}$$

where the position error

$$\boldsymbol{e}_p = \boldsymbol{p}_d - \boldsymbol{p}_e(\boldsymbol{q}) \tag{19.128}$$

between the desired and actual end-effector positions has been defined. Substituting (19.127) into (19.125) gives

$$\dot{\boldsymbol{e}}_p + \boldsymbol{K}_p \boldsymbol{e}_p = \boldsymbol{0} \tag{19.129}$$

and the choice of a positive definite matrix $\boldsymbol{K}_p$ guarantees asymptotic stability of the error system which in turn implies tracking of $\boldsymbol{p}_d$.

On the other hand, for 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 (19.124) is chosen as

$$\boldsymbol{\upsilon}_o = \boldsymbol{T}(\boldsymbol{\varphi}_e)(\dot{\boldsymbol{\varphi}}_d + \boldsymbol{K}_o \boldsymbol{e}_{o,\mathrm{Eul}}) \tag{19.130}$$

where

$$\boldsymbol{e}_{o,\mathrm{Eul}} = \boldsymbol{\varphi}_d - \varphi_e(\boldsymbol{q}) \tag{19.131}$$

is the orientation error. Substituting (19.130) into (19.126) gives

$$\dot{\boldsymbol{e}}_{o,\mathrm{Eul}} + \boldsymbol{K}_o \boldsymbol{e}_{o,\mathrm{Eul}} = \boldsymbol{0} \tag{19.132}$$

provided that the matrix $\boldsymbol{T}(\boldsymbol{\varphi}_e)$ is nonsingular. The system (19.132) is asymptotically stable for a positive definite $\boldsymbol{K}_o$, which in turn implies tracking of $\boldsymbol{\varphi}_d$.

To overcome the drawback of representation singularities in (19.130), an algorithm based on an alternative Euler angles description can be conceived that makes use of the rotation matrix describing the mutual orientation between the desired and the actual end-effector frame, i.e.,

$${}^e\boldsymbol{R}_d = \boldsymbol{R}_e^{\boldsymbol{T}}(\boldsymbol{q})\boldsymbol{R}_d. \tag{19.133}$$

Differentiating (19.133) with respect to time and accounting for (19.53) gives

$${}^e\dot{\boldsymbol{R}}_d = \boldsymbol{S}({}^e\boldsymbol{\omega}_{de})\,{}^e\boldsymbol{R}_d \tag{19.134}$$

where $\boldsymbol{\omega}_{de} = \boldsymbol{\omega}_d - \boldsymbol{\omega}_e(\boldsymbol{q})$ is the end-effector angular velocity error.

Let $\boldsymbol{\varphi}_{de}$ denote the set of Euler angles that can be extracted from ${}^e\boldsymbol{R}_d$. Then, in view of (19.55) and (19.53), the angular velocity ${}^e\boldsymbol{\omega}_{de}$ in (19.134) is related to the time derivative of $\boldsymbol{\varphi}_{de}$ as

$${}^e\boldsymbol{\omega}_{de} = \boldsymbol{T}(\boldsymbol{\varphi}_{de})\dot{\boldsymbol{\varphi}}_{de}. \tag{19.135}$$

At this point, the resolved angular velocity in (19.124) can be chosen as

$$\boldsymbol{v}_o = \boldsymbol{\omega}_d + \boldsymbol{R}_e\boldsymbol{T}(\boldsymbol{\varphi}_{de})\boldsymbol{K}_o\boldsymbol{e}_{o,\mathrm{EulAlt}} \tag{19.136}$$

where

$$\boldsymbol{e}_{o,\mathrm{EulAlt}} = \boldsymbol{\varphi}_{de}. \tag{19.137}$$

Substituting (19.136) into (19.126) gives

$$\dot{\boldsymbol{e}}_{o,\mathrm{EulAlt}} + \boldsymbol{K}_o\boldsymbol{e}_{o,\mathrm{EulAlt}} = \boldsymbol{0} \tag{19.138}$$

provided that the matrix $\boldsymbol{T}(\boldsymbol{\varphi}_{de})$ is nonsingular.

The clear advantage of the alternative over the classical Euler angles algorithm based on (19.130) is that by adopting a representation ϕ_{de} for which $\boldsymbol{T}(\boldsymbol{0})$ is nonsingular, representation singularities occur only for large orientation errors, e.g., when $\beta_{de} = \pm\pi/2$ for the *XYZ* representation. In other words, the ill-conditioning of matrix $\boldsymbol{T}$ is not influenced by the desired or actual end-effector orientation but only by the orientation error; hence, as long as the error parameter $|\beta_{de}| < \pi/2$, the behavior of system (19.138) is not affected by representation singularities. In this respect, the choice of a particular Euler angle description among the 12 possible should be carefully made, i.e., in the sense of avoiding a representation singularity for the second angle of the type $\beta = 0$.

To overcome the problem of representation singularities, an inverse kinematics algorithm based on the angle/axis description of orientation can be devised. From (19.133), the rotation ϑ_{de}, and the unit vector $\boldsymbol{r}_{de}$ can be extracted using the formulæ (19.12). Then, the orientation error can be defined as

$$\boldsymbol{e}_{o,\mathrm{AnAx}} = \sin\vartheta_{de}\boldsymbol{r}_{de}. \tag{19.139}$$

Notice that (19.139) gives a unique solution for $-\pi/2 < \vartheta < \pi/2$, 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 (19.139) is given by

$$\boldsymbol{e}_{o,\mathrm{AnAx}} = \frac{1}{2}\left(\boldsymbol{S}(\boldsymbol{n}_e(\boldsymbol{q}))\boldsymbol{n}_d + \boldsymbol{S}(\boldsymbol{s}_e(\boldsymbol{q}))\boldsymbol{s}_d + \boldsymbol{S}(\boldsymbol{a}_e(\boldsymbol{q}))\boldsymbol{a}_d\right), \tag{19.140}$$

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 ϑ sets the conditions $\boldsymbol{n}_e^{\mathrm{T}}\boldsymbol{n}_d \geq 0, \boldsymbol{s}_e^{\mathrm{T}}\boldsymbol{s}_d \geq 0, \boldsymbol{a}_e^{\mathrm{T}}\boldsymbol{a}_d \geq 0$.

Differentiation of (19.140) with respect to time gives

$$\dot{\boldsymbol{e}}_{o,\mathrm{AnAx}} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{\omega}_d - \boldsymbol{L}\boldsymbol{\omega} \tag{19.141}$$

where

$$\boldsymbol{L} = -\frac{1}{2}\left(\boldsymbol{S}(\boldsymbol{n}_d)\boldsymbol{S}(\boldsymbol{n}_e) + \boldsymbol{S}(\boldsymbol{s}_d)\boldsymbol{S}(\boldsymbol{s}_e) + \boldsymbol{S}(\boldsymbol{a}_d)\boldsymbol{S}(\boldsymbol{a}_e)\right). \tag{19.142}$$

At this point, the resolved angular velocity in (19.124) can be chosen as

$$\boldsymbol{\upsilon}_o = \boldsymbol{L}^{-1}(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{\omega}_d + \boldsymbol{K}_o\boldsymbol{e}_{o,\mathrm{AnAx}}). \tag{19.143}$$

Substituting (19.143) into (19.126) gives

$$\dot{\boldsymbol{e}}_{o,\mathrm{AnAx}} + \boldsymbol{K}_o\boldsymbol{e}_{o,\mathrm{AnAx}} = \boldsymbol{0} \tag{19.144}$$

provided that the matrix $\boldsymbol{L}$ is nonsingular. In this respect, if the angle ϑ_{de} is extended to the interval $(-\pi, \pi)$, then a singularity occurs at $\vartheta_{de} = \pm\pi/2$ for the matrix $\boldsymbol{L}$ which does not allow the computation of $\boldsymbol{\upsilon}_o$ as in (19.143).

The final inverse kinematics algorithm is based on the unit quaternion description of orientation. Let $\boldsymbol{Q}_d = \{\eta_d, \boldsymbol{\varepsilon}_d\}$ and $\boldsymbol{Q}_e = \{\eta_e, \boldsymbol{\varepsilon}_e\}$ represent the unit quaternions associated with $\boldsymbol{R}_d$ and $\boldsymbol{R}_e$, respectively. The mutual orientation can be expressed in terms of the unit quaternion $\boldsymbol{Q}_{de} = \{\eta_{de}, \boldsymbol{\varepsilon}_{de}\}$ where

$$\eta_{de} = \eta_e(\boldsymbol{q})\eta_d + \boldsymbol{\varepsilon}_e^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{\varepsilon}_d$$

$$\boldsymbol{\varepsilon}_{de} = \eta_e(\boldsymbol{q})\boldsymbol{\varepsilon}_d - \eta_d\boldsymbol{\varepsilon}_e(\boldsymbol{q}) - \boldsymbol{S}(\boldsymbol{\varepsilon}_d)\boldsymbol{\varepsilon}_e(\boldsymbol{q}). \tag{19.145}$$

It can be recognized that $\boldsymbol{Q}_{de} = \{1, \boldsymbol{0}\}$ if and only if $\boldsymbol{R}_e$ and $\boldsymbol{R}_d$ are aligned, and thus it is sufficient to consider $\boldsymbol{\varepsilon}_{de}$ to express an end-effector orientation error, i.e.,

$$\boldsymbol{e}_{o,\mathrm{Quat}} = \boldsymbol{\varepsilon}_{de}. \tag{19.146}$$

Note that the explicit computation of $\eta_e(\boldsymbol{q})$ and $\boldsymbol{\varepsilon}_e(\boldsymbol{q})$ is not possible, but it requires intermediate computation of the rotation matrix $\boldsymbol{R}_e(\boldsymbol{q})$ that is available from the robot direct kinematics; then, the unit quaternion can be extracted using the formulæ (19.17).

At this point, the resolved angular velocity in (19.124) can be chosen as

$$\boldsymbol{\upsilon}_o = \boldsymbol{\omega}_d + \boldsymbol{K}_o\boldsymbol{e}_{o,\mathrm{Quat}} \tag{19.147}$$

Substituting (19.147) into (19.126) gives

$$\boldsymbol{\omega}_{de} + \boldsymbol{K}_o \boldsymbol{e}_{o,\text{Quat}} = \mathbf{0}. \tag{19.148}$$

It should be observed that now the orientation error equation is not homogeneous in $\boldsymbol{e}_{o,\text{Quat}}$ since it contains the end-effector angular velocity error instead of the time derivative of the orientation error. To study the stability of system (19.148), consider the positive definite Lyapunov function

$$\boldsymbol{V} = (\eta_d - \eta_e)^2 + (\boldsymbol{\varepsilon}_d - \boldsymbol{\varepsilon}_e)^{\mathrm{T}}(\boldsymbol{\varepsilon}_d - \boldsymbol{\varepsilon}_e). \tag{19.149}$$

In view of the quaternion propagation (19.54), the time derivative of $\boldsymbol{V}$ along the trajectories of system (19.148) is given by

$$\dot{\boldsymbol{V}} = -\,\mathbf{e}_{o,\text{Quat}}^{\mathrm{T}} \boldsymbol{K}_o \boldsymbol{e}_{o,\text{Quat}} \tag{19.150}$$

which is negative definite, implying that $\boldsymbol{e}_{o,\text{Quat}}$ converges to zero.

19.8 Further Reading

Kinematic modelling of rigid robot manipulators can be found in any classical robotics textbook, e.g., Craig,[17] Dombre and Khalil,[19] Paul,[43] Sciavicco and Siciliano,[51] Spong and Vidyasagar,[54] Vukobratović.[56] Important reference sources on kinematics are also Angeles,[1] Bottema and Roth,[3] Hunt,[25] McCarthy,[38] Vukobratović and Kirćanski.[57] Symbolic software packages have been developed to derive robot kinematic models, Khalil.[26]

The Denavit-Hartenberg notation dates back to the original work of Denavit and Hartenberg,[18] which was recently modified in Craig[17] and Khalil and Kleinfinger.[28] One advantage of the so-called modified Denavit-Hartenberg notation over the classical one is that it can also be used for tree-structured and closed-chain robots.[28] The homogeneous transformation representation for direct kinematics of open-chain robots was first proposed in Pieper.[45]

Sufficient conditions for the inverse kinematics problem of closed-form solutions were given in Pieper.[45] These ensure the existence of solutions to six-degrees-of-freedom robots provided there are three revolute joints with intersecting axes or three prismatic joints; in the former case, at most eight admissible solutions exist, while the number reduces to two in the latter case. The kinematic decoupling resulting for spherical-wrist robots was developed in Featherstone,[22] Hollerbach,[24] Khalil and Bennis,[27] and Paul and Zhang.[44] An algebraic approach to the inverse kinematics problem for robots having closed-form solutions was presented in Paul,[43] and consists of successively post- (or pre-) multiplying both sides of the direct kinematics equation by partial transformation matrices to isolate the joint variables one after another; the types of equations obtained with this approach were formalized in Dombre and Khalil.[19] Recent methods[32,46] have found the inverse kinematics solution to general six-revolute-joint robots in the form of a polynomial equation of degree 16, i.e., the maximum number of admissible solutions is 16. On the other hand, numerical solution techniques based on iterative algorithms have been proposed, e.g., Goldenberg et al.[23] and Tsai and Morgan.[55]

The geometric Jacobian of the differential kinematics equation was originally proposed in Whitney.[59] The decomposition of the Jacobian into the product of three matrices is due to Renaud.[47] The problem of efficient Jacobian computation was addressed in Orin and Schrader.[42] The analytical Jacobian concept was introduced in Khatib[29] in connection with the operational space control problem. A treatment of differential kinematics mapping properties can be found in Sciavicco and Siciliano[51]; the reader is referred to Klema and Laub[31] for SVD decomposition.

The inversion of differential kinematics dates back to Whitney[59] under the name of resolved motion rate control. The adoption of the pseudoinverse of the Jacobian is due to Klein and Huang.[30] More on the properties of the pseudoinverse can be found in Boullion and Odell.[4] The use of null-space joint velocities for redundancy resolution was proposed in Liégeois,[33] and further refined in Maciejewski and Klein[36] and Yoshikawa[60] concerning the choice of objective functions. The reader is referred to Nakamura[39] for a complete treatment of redundant robots.

The adoption of the damped least-squares inverse was independently presented in Nakamura and Hanafusa[40] and Wampler.[58] More about kinematic control in the neighborhood of kinematic singularities can be found in Chiaverini.[9] The technique for estimating the smallest singular value of the Jacobian is due to Maciejewski and Klein,[37] and its modification to include the second smallest singular value was achieved by Chiaverini.[10] The use of the damped least-squares inverse for redundant robots was presented in Egeland et al.[21] The user-defined accuracy strategy was proposed in Chiaverini et al.[12] and further refined in Chiaverini et al.[13] A review of the damped least-squares inverse kinematics with experiments on an industrial robot was recently presented.[16]

Closed-loop inverse kinematics algorithms are discussed in Sciavicco and Siciliano.[51] The original Jacobian transpose inverse kinematics algorithm was proposed in Sciavicco and Siciliano;[49] the choice of suitable gains for achieving robustness to singularities was discussed in Chiacchio and Siciliano.[7] Singular value decomposition of the Jacobian transpose is due to Chiaverini et al.[14] Combining the Jacobian transpose solution with the pseudoinverse solution was proposed in Chiacchio and Siciliano.[8] References on the augmented task space approach are Egeland,[20] Samson et al.,[48] Sciavicco and Siciliano,[50] and Seraji.[52] The occurrence of artificial singularities was pointed out in Baillieul,[2] and their properties were studied in Chiacchio et al.[6] The task priority strategy was originally proposed in Nakamura et al.[41] and has recently been refined in Chiaverini[11] concerning robustness to artificial singularities. The use of the Jacobian transpose for the constraint task was presented in Chiaverini et al.[15] and Siciliano.[53] The expression of the end-effector orientation error based on an angle/axis description of orientation is due to Luh et al.[35] and its properties were studied in Lin.[34] The use of a quaternion-based orientation error is due to Yuan.[61] More about the possible definitions of the orientation error can be found in Caccavale et al.[5]

References

1. Angeles, J., *Spatial Kinematic Chains: Analysis, Synthesis, Optimization,* Springer-Verlag, Berlin, 1982.
2. Baillieul, J., Kinematic programming alternatives for redundant manipulators, in *Proc. 1985 IEEE Int. Conf. Robotics and Automation,* St. Louis, MO, 1985, 722.
3. Bottema, O. and Roth, B., *Theoretical Kinematics,* North Holland, Amsterdam, 1979.
4. Boullion, T. L. and Odell, P. L., *Generalized Inverse Matrices,* Wiley, New York, 1971.
5. Caccavale, F., Natale, C., Siciliano, B., and Villani, L., Resolved-acceleration control of robot manipulators: A critical review with experiments, *Robotica,* 16, 565, 1998.
6. Chiacchio, P., Chiaverini, S., Sciavicco, L., and Siciliano, B., Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy, *Int. J. Robotics Res.,* 10, 410, 1991.
7. Chiacchio, P., and Siciliano, B., Achieving singularity robustness: An inverse kinematic solution algorithm for robot control, in *Robot Control: Theory and Applications,* IEE Control Engineering Series 36, Warwick, K. and Pugh, A., Eds., Peter Peregrinus, Herts, U.K., 149, 1988.
8. Chiacchio, P. and Siciliano, B., A closed-loop Jacobian transpose scheme for solving the inverse kinematics of nonredundant and redundant robot wrists, *J. Robotic Systems,* 6, 601, 1989.
9. Chiaverini, S., Inverse differential kinematics of robotic manipulators at singular and near-singular configurations, in *Prepr. 1992 IEEE Int. Conf. Robotics Automation — Tutorial on Redundancy: Performance Indices, Singularities Avoidance, and Algorithmic Implementations,* Nice, 1992.
10. Chiaverini, S., Estimate of the two smallest singular values of the Jacobian matrix: Application to damped least-squares inverse kinematics, *J. Robotic Systems,* 10, 991, 1993.

11. Chiaverini, S., Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators, *IEEE Trans. Robotics Automation,* 13, 398, 1997.
12. Chiaverini, S., Egeland, O., and Kanestrøm, R. K., Achieving user-defined accuracy with damped least-squares inverse kinematics, in *Proc. 5th Int. Conf. Advanced Robotics,* Pisa, I, 672, 1991.
13. Chiaverini, S., Egeland, O., Sagli, J. R., and Siciliano, B., User-defined accuracy in the augmented task space approach for redundant manipulators, *Lab. Robotics Automation,* 4, 59, 1992.
14. Chiaverini, S., Sciavicco, L., and Siciliano, B., Control of robotic systems through singularities, in *Advanced Robot Control,* Lecture Notes in Control and Information Science 162, Canudas de Wit, C., Ed., Springer-Verlag, Berlin, 285, 1991.
15. Chiaverini, S., Siciliano, B., and Egeland, O., Redundancy resolution for the human-arm-like manipulator, *Robotics Autonomous Systems,* 8, 239, 1991.
16. Chiaverini, S., Siciliano, B., and Egeland, O., Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator, *IEEE Trans. Control Systems Technology,* 2, 123, 1994.
17. Craig, J. J., *Introduction to Robotics: Mechanics and Control,* 2nd ed., Addison-Wesley, Reading, MA, 1989.
18. Denavit, J. and Hartenberg, R. S., A kinematic notation for lower-pair mechanisms based on matrices, *ASME J. Appl. Mech.,* 22, 215, 1955.
19. Dombre, E. and Khalil, W., *Modélisation et Commande des Robots,* Hermès, Paris, 1988.
20. Egeland, O., Task-space tracking with redundant manipulators, *IEEE J. Robotics Automation,* 3, 471, 1987.
21. Egeland, O., Sagli, J. R., Spangelo, I., and Chiaverini, S., A damped least-squares solution to redundancy resolution, in *Proc. 1991 IEEE Int. Conf. Robotics Automation,* Sacramento, CA, 945, 1991.
22. Featherstone, R., Position and velocity transformations between robot end-effector coordinates and joint angles, *Int. J. Robotics Res.,* 2(2), 35, 1983.
23. Goldenberg, A. A., Benhabib, B., and Fenton, R. G., A complete generalized solution to the inverse kinematics of robots, *IEEE J. Robotics Automation,* 1, 14, 1985.
24. Hollerbach, J. M., Wrist-partitioned inverse kinematic accelerations and manipulator dynamics, *Int J. Robotics Res.,* 2(4), 61, 1983.
25. Hunt, K. H., *Kinematic Geometry of Mechanisms,* Clarendon, Oxford, U.K., 1978.
26. Khalil, W., A system for generating the symbolic models of robots, in *Postpr. 4th IFAC Symp. Robot Control,* Capri, 416, 1994.
27. Khalil, W. and Bennis, F., Automatic generation of the inverse geometric model of robots, *Robotics and Autonomous Systems,* 7, 1, 1991.
28. Khalil, W. and Kleinfinger, J. F., A new geometric notation for open and closed-loop robots, in *Proc. 1986 IEEE Int. Conf. Robotics Automation,* San Francisco, CA, 1174, 1986.
29. Khatib, O., A unified approach for motion and force control of robot manipulators: The operational space formulation, *IEEE J. Robotics Automation,* 3, 43, 1987.
30. Klein, C. A. and Huang, C. H., Review of pseudoinverse control for use with kinematically redundant manipulators, *IEEE Trans. Systems, Man, Cybernetics,* 13, 245, 1983.
31. Klema, V. C. and Laub, A. J., The singular value decomposition: Its computation and some applications, *IEEE Trans. Automatic Control,* 25, 164, 1980.
32. Lee, H. Y. and Liang, C. G., Displacement analysis of the general 7-link 7R mechanism, *Mechanism Machine Theory,* 23, 219, 1988.
33. Liégeois, A., Automatic supervisory control of the configuration and behavior of multibody mechanisms, *IEEE Trans. Systems, Man, Cybernetics,* 7, 868, 1977.
34. Lin, S. K., Singularity of a nonlinear feedback control scheme for robots, *IEEE Trans. Systems, Man, Cybernetics,* 19, 134, 1989.
35. Luh, J. Y. S., Walker, M. W., and Paul, R. P. C., Resolved-acceleration control of mechanical manipulators, *IEEE Trans. Automatic Control,* 25, 468, 1980.
36. Maciejewski, A. A. and Klein, C. A., Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, *Int. J. Robotics Res.,* 4(3), 109, 1985.

37. Maciejewski, A. A. and Klein, C. A., Numerical filtering for the operation of robotic manipulators through kinematically singular configurations, *J. Robotic Systems,* 5, 527, 1988.
38. McCarthy, J. M., *An Introduction to Theoretical Kinematics,* MIT Press, Cambridge, MA, 1990.
39. Nakamura, Y., *Advanced Robotics: Redundancy and Optimization,* Addison-Wesley, Reading, MA, 1991.
40. Nakamura, Y. and Hanafusa, H., Inverse kinematic solutions with singularity robustness for robot manipulator control, *ASME J. Dynamic Systems, Measurement, Control,* 108, 163, 1986.
41. Nakamura, Y., Hanafusa, H., and Yoshikawa, T., Task-priority based redundancy control of robot manipulators, *Int. J. Robotics Res.,* 6(2), 3, 1987.
42. Orin, D. E. and Schrader, W. W., Efficient computation of the Jacobian for robot manipulators, *Int. J. Robotics Res.,* 3(4), 66, 1984.
43. Paul, R. P., *Robot Manipulators: Mathematics, Programming, and Control,* MIT Press, Cambridge, MA, 1981.
44. Paul, R. P. and Zhang, H., Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation, *Int. J. Robotics Res.,* 5(2), 32, 1986.
45. Pieper, D. L., *The Kinematics of Manipulators under Computer Control,* memo. AIM 72, Stanford Artificial Intelligence Laboratory, 1968.
46. Raghavan, M. and Roth, B., Inverse kinematics of the general 6R manipulator and related linkages, *ASME J. Mechanical Design,* 115, 502, 1990.
47. Renaud, M., Calcul de la matrice jacobienne necessaire à la commande coordonnee d'un manipulateur, *Mechanism and Machine Theory,* 15, 81, 1980.
48. Samson, C., Le Borgne, M., and Espiau, B., *Robot Control: The Task Function Approach,* Oxford Engineering Science Series 22, Clarendon, Oxford, U.K., 1991.
49. Sciavicco, L. and Siciliano, B., Coordinate transformation: A solution algorithm for one class of robots, *IEEE Trans. Systems, Man, Cybernetics,* 16, 550, 1986.
50. Sciavicco, L. and Siciliano, B., A solution algorithm to the inverse kinematic problem for redundant manipulators, *IEEE J. Robotics Automation,* 4, 403, 1988.
51. Sciavicco, L. and Siciliano, B., *Modelling and Control of Robot Manipulators,* 2nd ed., Springer, London, 2000.
52. Seraji, H., Configuration control of redundant manipulators: Theory and implementation, *IEEE Trans. Robotics Automation,* 5, 472, 1989.
53. Siciliano, B., Solving manipulator redundancy with the augmented task space method using the constraint Jacobian transpose, in *Prepr. 1992 IEEE Int. Conf. Robotics and Automation — Tutorial on Redundancy: Performance Indices, Singularities Avoidance, and Algorithmic Implementations,* Nice, 1992.
54. Spong, M. W. and Vidyasagar, M., *Robot Dynamics and Control,* Wiley, New York, 1989.
55. Tsai, L. W. and Morgan, A. P., Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods, *ASME J. Mechanisms, Transmission, Automation Design,* 107, 189, 1985.
56. Vukobratović, M., *Introduction to Robotics,* Springer, Berlin, 1989.
57. Vukobratović, M. and Kirćanski, M., *Kinematics and Trajectory Synthesis of Manipulation Robots,* Scientific Fundamentals of Robotics 3, Springer, Berlin, 1986.
58. Wampler, C. W., Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods, *IEEE Trans. Systems, Man, Cybernetics,* 16, 93, 1986.
59. Whitney, D. E., Resolved motion rate control of manipulators and human prostheses, *IEEE Trans. Man–Machine Systems,* 10, 47, 1969.
60. Yoshikawa, T., Manipulability of robotic mechanisms, *Int. J. Robotics Res.,* 4(2), 3, 1985.
61. Yuan, J. S.-C., Closed-loop manipulator control using quaternion feedback, *IEEE J. Robotics Automation,* 4, 434, 1988.