Journal of Intelligent and Robotic Systems **9**: 5–23, 1994.

On Rotation Representations in Computational Robot Kinematics*

MURAT TANDIRCI, JORGE ANGELES and JOHN DARCOVICH
McGill Centre for Intelligent Machines and Department of Mechanical Engineering, McGill University, 817 Sherbrooke Street West, Montreal, Canada H3A 2K6

(Received: December 1992)

Abstract. Various methods of implementing forward and inverse kinematics of six-axes industrial robots are analyzed in this paper from the viewpoint of numerical conditioning and convergence speed both close to a solution and away from it. Computational complexities are derived in terms of the number of arithmetic operations and comparisons are made by observing the actual CPU time consumption. The formulations presented make use of different sets of invariants describing the orientation of the gripper. It is shown that, in inverse kinematics, there is a tradeoff between numerical stability and computational speed.

Key words. Computational kinematics, forward and inverse kinematics, robotic manipulators, rotation invariants.

1. Introduction

This paper is confined to the kinematics of open-chain manipulators consisting of six revolute joints, a short account of which is given in [1]. Moreover, we discuss inverse kinematics only in the most general case, when no closed-form solution is possible. As shown in [2], only a special class of manipulator architectures lends itself to closed-form solutions as a set of cascaded quadratic equations or, as shown in [3], in the form of a quaratic equation cascaded with a quadratic equation. Moreover, Mavroidis and Roth [4] have produced a comprehensive list of solvable manipulators that they call *manipulators with simple inverse kinematics*. Since quartic equations admit closed-form solutions, we will call manipulators leading to such a type of equations *solvable*. As Lee and Liang [5] showed, the most general case of 6-axes manipulator leads to a 16th-degree polynomial equation, whose roots can be computed only numerically. Raghavan and Roth [6] proposed a procedure to derive the underlying 16th-order polynomial. Most industrial manipulators are both *orthogonal* and *decoupled*. Here, orthogonal means that their consecutive axes make angles that are multiples of 90°; decoupled means that their last three axes are concurrent. These features allow a decoupling of the positioning and orientation tasks, which leads to either quadratic or quartic equations, at most. While many industrial manipulators are designed with an orthogonal and decoupled architecture, in some

* An abridged version of this paper was presented in the 1992 IEEE International Conference on Robotics and Automation, 1992 [1].

cases, when high-accuracy is required, a calibration is warranted. Moreover, after calibration, the underlying robotic architecture is no longer orthogonal and, certainly, neither decoupled. Hence, the inverse kinematics of calibrated industrial robots, even if nominally orthogonal and decoupled, calls for a numerical solution, which is the motivation behind this work. Furthermore, when solving the inverse kinematics of a robot iteratively, as proposed in [3, 7–9], a fast convergence to an accurate solution allows for robot simulation and control in real time. For example, in an interactive computer animation program, the end-effector (EE) of the robot can be made to follow a specified trajectory using a graphical input, and the motion resulting from the change of joint variables can be observed immediately. Moreover, quick inverse kinematics results are needed in telerobotics applications, where the operator may describe ongoing tasks based on the actual surroundings. Similarly, robots equipped with vision systems determine their tasks based on the information they gather about their environment; in such cases, quick and accurate inverse kinematics results are needed to perform the upcoming tasks.

In this paper, methods based on numerical procedures are discussed. Below, a brief comparison among the numerical and closed-form solutions, when the latter are possible at all, is given.

In on-line applications and path tracking, iterative procedures are attractive because the next solution is close to the current one. Thus, the current solution can be used as an initial guess, thereby allowing a quick convergence in a few iterations. Furthermore, in path tracking, only one solution is needed, and the iterative procedures do not spend extra time calculating the remaining solutions.

The disadvantage of the numerical procedures is that they do not provide information about the remaining solutions, if they are needed. Moreover, for the first point on a given path, an initial guess has to be supplied, which might lead unpredictably to divergence, although the occurrence of the latter can be avoided with the use of continuation [7, 9].

For special manipulator architectures that lend themselves to a closed-form solution, it is preferable to use the algebraic approach, since solutions can be obtained faster [10]. Moreover, unlike numerical procedures, direct methods allow the computation of solutions in a predetermined amount of time.

2. Forward Kinematics

Forward Kinematics refers to the calculation of the orientation and position of the EE given the joint angles. The well-known Hartenberg and Denavit notation [11] is used throughout.

For a general six-axis manipulator, the orientation of the EE in the base frame is expressed as the product of six rotation matrices, namely,

$$\mathbf{Q} = \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_4\mathbf{Q}_5\mathbf{Q}_6, \tag{1}$$

where $\mathbf{Q}_i \equiv [\mathbf{Q}_i]_i$ denotes a rotation matrix expressing the orientation of the $(i+1)$st

frame with respect to the ith frame, in ith-frame coordinates. Because the product matrix is orthogonal, only three out of its nine entries are independent. Different methods can be employed to represent a rotation, as discussed in [12]. Here, we use three different sets of *invariants* of the rotation matrix to express the said orientation equations [13]. These quantities are preferred over Euler angles because they are invariant under a change of coordinate frame.

Furthermore, three sets of invariants are analyzed, the first being linearly related to the rotation matrix, the second being linearly related to the square root of the said matrix. The first set is thus known as the *linear invariants*, whereas the second is known as the *Euler parameters* or, as suggested by Cheng and Gupta [14], the *Euler–Rodrigues parameters*. The third set is called in [13] the *natural invariants*, while, in the same reference, the Euler–Rodrigues parameters are also called the *quadratic invariants*. The linear invariants are defined from the *vector* and the *trace* of the rotation matrix as [13]

$$q_0 \equiv \frac{\mathrm{tr}(\mathbf{Q}) - 1}{2} = \cos\phi, \qquad \mathbf{q} \equiv \mathrm{vect}(\mathbf{Q}) = \sin\phi\,\mathbf{u}, \tag{2}$$

where $\mathbf{u}$ is the unit vector parallel to the axis of rotation, and ϕ is the angle of rotation about this axis. Both $\mathbf{u}$ and ϕ are the natural invariants of $\mathbf{Q}$. The quadratic invariants are defined in turn as [15]

$$\hat{q}_0 \equiv \frac{\mathrm{tr}(\sqrt{\mathbf{Q}}) - 1}{2} = \cos\frac{\phi}{2}, \qquad \hat{\mathbf{q}} \equiv \mathrm{vect}(\sqrt{\mathbf{Q}}) = \sin\frac{\phi}{2}\,\mathbf{u}, \tag{3}$$

where $\sqrt{\mathbf{Q}}$ denotes the *proper orthogonal* square root of $\mathbf{Q}$. Furthermore, the first two sets of invariants are related as follows:

$$\hat{q}_0 = \sqrt{\frac{1+q_0}{2}}, \qquad \hat{\mathbf{q}} = \mathbf{q}\,\frac{\sqrt{2(1+q_0)}}{2(1+q_0)} = \frac{\hat{q}_0}{(1+q_0)}\,\mathbf{q}. \tag{4}$$

From the above relations, we have:

- a square root operation is always necessary to compute the quadratic invariants;
- the vector of linear invariants vanishes when the rotation angle is π, an undesirable result;
- the quadratic invariants are well defined for all rotation angles;
- physically, the proper orthogonal $\sqrt{\mathbf{Q}}$ represents a rotation about the axis of rotation of $\mathbf{Q}$ through an angle of half that of $\mathbf{Q}$.

The EE position is readily derived, as indicated below, when the rotation matrices expressing the orientation of the EE in all frames are available.

$\mathbf{r}_6 \leftarrow \mathbf{a}_6$

```
For i=5 to 1 do
```

$\quad \mathbf{r}_i \leftarrow \mathbf{a}_i + \mathbf{Q}_i\mathbf{r}_{i+1}$

```
enddo
```

where $\mathbf{a}_i$ is the vector directed from O_i to O_{i+1}, expressed in the ith frame, whereas $\mathbf{r}_i$ is the vector directed from O_i to point P of the EE, expressed in the ith frame, $\mathbf{r}_1$ thus describing the desired EE position in the base frame (Figure 1).

2.1. DERIVATION OF THE INVARIANTS OF A PRODUCT OF ROTATIONS

Rodrigues [16] obtained the normalized quadratic invariants, also known as the Rodrigues parameters, of two concatenated rotations as functions of individual quadratic invariants of the said rotations. The three Rodrigues parameters are normalized Euler parameters in that the former are obtained by dividing the vector quadratic invariant $\hat{\mathbf{q}}$ by the scalar quadratic invariant $\hat{q}_0$. Moreover, as shown in Section A.2 of the Appendix, it is also possible to derive an expression for the linear invariants of the concatenated rotations from the individual linear invariants. An alternative approach is to actually multiply rotation matrices and derive the invariants of the product. Below, different methods are presented and their computational complexities are analyzed. A summary of computational costs is shown in Table I. Henceforth, A, M, D and S denote additions, multiplications, divisions and square roots, respectively.

Linear Invariants

Method 1: Matrix Multiplications. As shown in Section A.1, the cost of deriving the product of six rotation matrices is $117M$ and $60A$. The vector product requires $6M$ and $3A$, rotation while the scalar product $3M$ and $2A$.

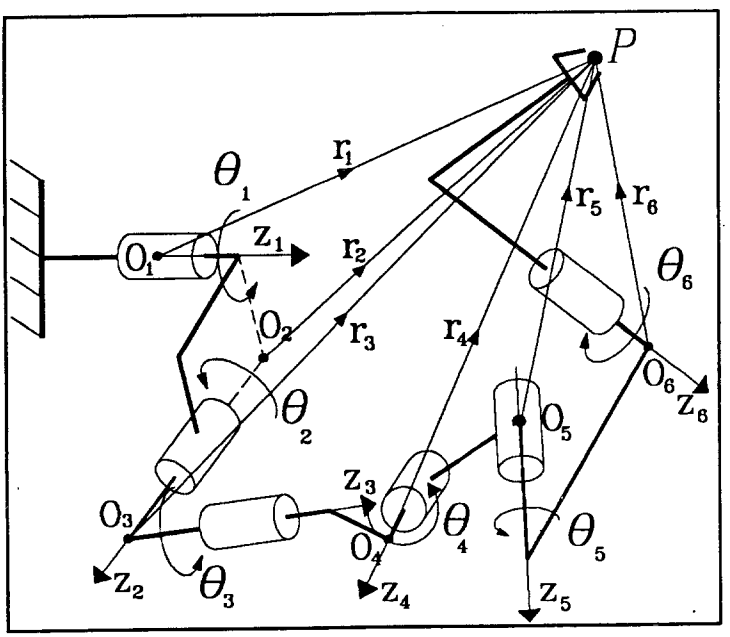

Fig. 1. General 6-axis manipulator.

Table I. Operations needed to calculate the invariants.

	Vector method			Matrix method	
	Linear inv.	Quadratic inv. (2)	Quadratic inv. (3)	Linear inv.	Quadratic inv. (1)
1×	$1D+33M$ $+29A$	$1S+3D+35M$ $+29A$	$2S+2D+29M$ $+24A$	$25M+18A$	$1S+1D+25M$ $+18A$
5×	$5D+149M$ $+125A$	$1S+7D+151M$ $+125A$	$6S+6D+125M$ $+96A$	$121M+66A$	$1S+1D+122M$ $+66A$
CPU (μsec)	137.5	227.5	197.5	77.5	85.0

Method 2: Vector Compositions. Let the linear invariants of a rotation matrix **A** be denoted by $\mathbf{q}^A$ and q_0^A, those of a second rotation matrix **B** by $\mathbf{q}^B$ and q_0^B. Moreover, the linear invariants of the product **AB** are denoted by $\mathbf{q}$ and q_0. Below we include expressions for $\mathbf{q}$ and q_0 in terms of $\mathbf{q}^A$, q_0^A, $\mathbf{q}^B$ and q_0^B, namely,

$$\mathbf{q} = \frac{\mathbf{n}}{2D}, \qquad q_0 = \frac{N-D}{2D}, \tag{5}$$

where

$$D \equiv (1+q_0^A)(1+q_0^B), \tag{6}$$

$$N \equiv (1+q_0^A)(1+q_0^B)(q_0^A+q_0^B+q_0^Aq_0^B)+(\mathbf{q}^A\cdot\mathbf{q}^B)(\mathbf{q}^A\cdot\mathbf{q}^B-2D), \tag{7}$$

$$\mathbf{n} \equiv (D-\mathbf{q}^A\cdot\mathbf{q}^B)[(1+q_0^B)\mathbf{q}^A+(1+q_0^A)\mathbf{q}^B+\mathbf{q}^A\times\mathbf{q}^B]. \tag{8}$$

The reader is referred to Section A.2 for the derivation of computational costs with this method.

Quadratic Invariants

Method 1: Matrix Multiplications. As discussed above, the cost of computing five matrix products is $117M$ and $60A$. Using the expressions below, which are derived using Eqs. (2) and (4), the calculation of the quadratic invariants from the product matrix requires $1S+1D+5M+6A$

$$\hat{q}_0 \equiv \frac{\sqrt{q_0+1}}{2}, \qquad \hat{\mathbf{q}} \equiv \frac{1}{2\hat{q}_0}\,\mathrm{vect}(\mathbf{Q}).$$

Method 2: Vector Composition of Linear Invariants. The method of calculation of the linear invariants using eqs. (5) can be extended to quadratic invariants. When eqs. (5) are substituted into eqs. (4), the relations shown below are found [17]:

$$\hat{q}_0 = \frac{1}{2}\sqrt{1+\frac{N}{D}}, \qquad \hat{\mathbf{q}} = \frac{1}{2}\frac{\sqrt{D(N+D)}}{D(N+D)}\,\mathbf{n} = \frac{\hat{q}_0}{N+D}\,\mathbf{n} \tag{9}$$

where D, N and $\mathbf{n}$ were defined in eqs. (6–8). The computational costs involved are included in Section A.3.

Method 3: Vector Composition of Quadratic Invariants. The underlying relations were found by Rodrigues [16]. They can be verified from the relations obtained in eqs. (4) and eqs. (5), namely,

$$\hat{q}_0 = \hat{q}_0^A \hat{q}_0^B - \hat{\mathbf{q}}^{\mathbf{A}} \cdot \hat{\mathbf{q}}^{\mathbf{B}}$$

$$\hat{\mathbf{q}} = \hat{q}_0(\hat{q}_0^B \hat{\mathbf{q}}^{\mathbf{A}} + \hat{q}_0^A \hat{\mathbf{q}}^{\mathbf{B}} + \hat{\mathbf{q}}^A \times \hat{\mathbf{q}}^{\mathbf{B}})$$

The computational costs involved are included in Section A.4.

The operation counts of the above methods for one and five products are reported in Table 1. Moreover, the CPU times to calculate the linear and quadratic invariants for five products are observed on an *IRIS 4D/210VGX* workstation. The results are summarized in the same table.

3. Inverse Kinematics

Inverse Kinematics refers to the calculation of joint angles, given the position and orientation of the gripper. Three numerical procedures will be analyzed. All procedures use vector invariants, namely linear, quadratic and natural invariants. The first two employ the Newton–Gauss method, while the third method employs the iterative scheme first proposed by Pieper [2].

3.1. LINEAR INVARIANTS

Here, the vector function is seven-dimensional, consisting of the differences between the current and the prescribed position and orientation values. The first four entries of this vector are the linear invariants, whereas the remaining three represent the position vector. The prescribed data set consists of a 3D position vector $\mathbf{p}_g$, and a rotation matrix $\mathbf{Q}_g$. Furthermore, to start the Newton–Gauss scheme, an initial guess of joint variables must be chosen.

The problem then is formulated as

$$\min_{\theta} \|\mathbf{f}(\theta)\|^2 \tag{10}$$

without constraints, where

$$\mathbf{f}(\theta) \equiv \begin{bmatrix} 2[\mathrm{vect}(\mathbf{Q}) - \mathrm{vect}(\mathbf{Q}_g)] \\ \mathrm{tr}(\mathbf{Q}) - \mathrm{tr}(\mathbf{Q}_g) \\ \mathbf{p} - \mathbf{p}_g \end{bmatrix}.$$

The first four components of vector $\mathbf{f}$ are nonlinearly dependent; in the absence of singularities, we have six independent equations. The factor 2 multiplying the first three entries is used to eliminate divisions by 2, thus saving time in computations.

In order to solve problem (10), the gradient of $\mathbf{f}(\theta)$ with respect to the joint variables has to be calculated. As derived in [6] for a six-axis manipulator, this

gradient can be expressed as

$$\mathbf{J}(\theta) \equiv \frac{\partial \mathbf{f}(\theta)}{\partial \theta} = \mathbf{HK},$$

where the 7×6 matrix $\mathbf{H}$ arises from the formulation of the orientation equations, while $\mathbf{K}$, a 6×6 matrix, is that commonly known as the *Jacobian matrix* and first derived by Whitney [18]. Thus, $\mathbf{K}$ maps the joint rates into angular and translational velocity and takes on the form

$$\mathbf{K} \equiv \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_6 \\ \mathbf{e}_1 \times \mathbf{r}_1 & \mathbf{e}_2 \times \mathbf{r}_2 & \dots & \mathbf{e}_6 \times \mathbf{r}_6 \end{bmatrix}. \tag{11}$$

As derived in [6], $\mathbf{H}$ is given in turn as

$$\mathbf{H} \equiv \begin{bmatrix} 1\,\mathrm{tr}(\mathbf{Q}) - \mathbf{Q} & \mathbf{O} \\ -2\,\mathrm{vect}(\mathbf{Q})^T & \mathbf{0}^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix},$$

where $\{\mathbf{e}_i\}_1^6$ are the unit vectors parallel to the axes of the joints, and $\{\mathbf{r}_i\}_1^6$ are the vectors directed from point O_i to point P of the EE as shown in Figure 1. Moreover, $\mathbf{0}$ and $\mathbf{O}$ denote the 3-dimensional zero vector and the 3×3 zero matrix, while $\mathbf{1}$ denotes the 3×3 identity matrix.

From its series expansion, a first-order approximation of function $\mathbf{f}(\theta)$, evaluated at the current value of θ allows the computation of $\Delta\theta$ from $\mathbf{J}\Delta\theta = -\mathbf{f}$.

Depending on the chosen initial guess, the above method yields different solutions upon convergence. As proven by Lee and Liang [5], up to 16 solutions are to be expected. The solution obtained will be a local minimum of the problem in eq. (10) that verifies the normality condition $\mathbf{J}^T\mathbf{f} = 0$.

3.2. QUADRATIC INVARIANTS (EULER–RODRIGUES PARAMETERS)

With this rotation representation, the formulation of the problem is similar to the one with the linear invariants. Now the quadratic invariants are required to match their prescribed counterparts. Thus, the problem is

$$\min_{\theta} \|\mathbf{f}(\theta)\|^2$$

without constraints, where

$$\mathbf{f}(\theta) \equiv \begin{bmatrix} 2[\mathrm{vect}(\sqrt{\mathbf{Q}}) - \mathrm{vect}(\sqrt{\mathbf{Q}_g})] \\ \mathrm{tr}(\sqrt{\mathbf{Q}}) - \mathrm{tr}(\sqrt{\mathbf{Q}_g}) \\ \mathbf{p} - \mathbf{p}_g \end{bmatrix}.$$

As shown in [15], the gradient of $\mathbf{f}(\theta)$ is now calculated by making use of the relation below:

$$\frac{\partial \sqrt{\mathbf{Q}}}{\partial \theta_i} = \mathbf{E}_i \sqrt{\mathbf{Q}}$$

where $\mathbf{E}_i$ is the cross-product matrix of vector $\mathbf{e}_i$, i.e.,

$$\mathbf{E}_i \equiv \frac{\partial(\mathbf{e}_i \times \mathbf{v})}{\partial \mathbf{v}}$$

for every $\mathbf{v}$. Hence,

$$\frac{\partial \operatorname{vect}(\sqrt{\mathbf{Q}})}{\partial \theta_i} = \operatorname{vect}(\mathbf{E}_i \sqrt{\mathbf{Q}}) = \tfrac{1}{2}[\operatorname{tr}(\sqrt{\mathbf{Q}})\mathbf{1} - \sqrt{\mathbf{Q}}]\mathbf{e}_i,$$

$$\frac{\partial \operatorname{tr}(\sqrt{\mathbf{Q}})}{\partial \theta_i} = \operatorname{tr}(\mathbf{E}_i \sqrt{\mathbf{Q}}) = -2[\operatorname{vect}(\sqrt{\mathbf{Q}})]^T \mathbf{e}_i,$$

where $\mathbf{1}$ denotes the 3×3 identity matrix. Therefore, the Jacobian matrix can now be factored as,

$$\mathbf{J}' \equiv \mathbf{H}'\mathbf{K},$$

where $\mathbf{H}'$ is the 7×6 matrix given below

$$\mathbf{H}' \equiv \begin{bmatrix} \mathbf{1}\operatorname{tr}(\sqrt{\mathbf{Q}}) - \sqrt{\mathbf{Q}} & \mathbf{O} \\ -2\operatorname{vect}(\sqrt{\mathbf{Q}})^T & \mathbf{0}^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix}$$

with $\mathbf{0}$,$\mathbf{O}$ and $\mathbf{1}$ already defined in Subsection 3.1 while $\mathbf{K}$ is the Jacobian as defined by Whitney [18]. The solution is obtained using the Newton–Gauss procedure.

3.3. NATURAL INVARIANTS

This method is different from the above two methods because it does not need any auxiliary matrix, such as $\mathbf{H}$ or $\mathbf{H}'$, in its Jacobian. Instead, the Jacobian used is simply $\mathbf{K}$, defined in Eq. (11). The objective here is twofold: First, we wish to minimize the difference between the current and the prescribed rotation matrices, $\mathbf{Q}$ and $\mathbf{Q}_g$. Secondly, we wish to minimize the difference between the current and prescribed position vectors of the EE, which are denoted below by $\Delta\mathbf{p}$ and $\Delta\mathbf{p}_g$. The velocity Jacobian relates the incremental joint angles to the vector of the difference in poses, namely [2],

$$\mathbf{K}(\theta)\Delta\theta = \begin{bmatrix} \sin\Delta\phi\,\mathbf{u} \\ \Delta\mathbf{p} \end{bmatrix} \tag{12}$$

where the unit vector $\mathbf{u}$ and the scalar $\Delta\phi$ express the difference between the current and prescribed orientations, $\mathbf{Q}$ and $\mathbf{Q}_g$. Thus, frame $\mathscr{C}$ is carried into $\mathscr{G}$ by a rotation about an axis parallel to the unit vector $\mathbf{u}$ through an angle $\Delta\phi$, which were termed the *natural invariants* [13] of the rotation involved. The prescribed pose is given as the $\mathscr{G}$-frame, the actual or current configuration as the $\mathscr{C}$-frame, and the base frame as the $\mathscr{F}$-frame, as shown in Figure 2. The rotation carrying $\mathscr{C}$ into $\mathscr{G}$, here denoted as $\Delta\mathbf{R}$, is first calculated in the $\mathscr{C}$ frame and then transformed into base coordinates using an orthogonal transformation carrying $\mathscr{F}$ into $\mathscr{C}$, and denoted as $\mathbf{Q}$,

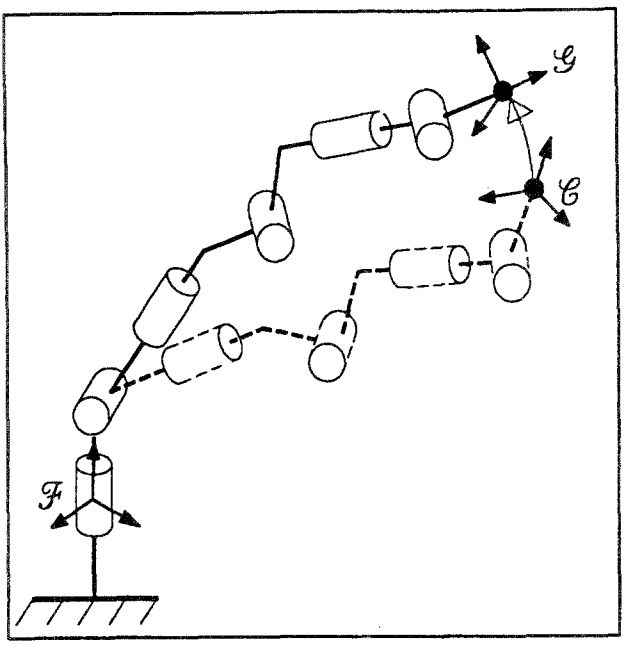

Fig. 2. General 6-axis manipulator at the current $\mathcal{C}$ and prescribed $\mathcal{G}$ configurations.

namely, $\mathbf{Q}\Delta\mathbf{R} = \mathbf{Q}_g$. Hence, $[\Delta\mathbf{R}]_{\mathcal{C}} = \mathbf{Q}^T\mathbf{Q}_g$, i.e., $\Delta\mathbf{R}$ in the $\mathcal{F}$ frame is given as

$$[\Delta\mathbf{R}]_{\mathcal{F}} = \mathbf{Q}[\Delta\mathbf{R}]_{\mathcal{C}}\mathbf{Q}^T = \mathbf{Q}_g\mathbf{Q}^T.$$

The product $\sin\Delta\phi\mathbf{u}$ is then readily calculated from $\Delta\mathbf{R}$ as

$$\sin\Delta\phi\mathbf{u} = \text{vect}(\Delta\mathbf{R})$$

Once the right-hand side of the algebraic system of Eq. (12) is determined, the solution $\Delta\theta$ can be obtained using the LU-decomposition [19]. The new vector of joint variables is then obtained as $\theta^1 = \theta^0 + \Delta\theta$ and, at the next iteration, θ^1 is used to compute the said right-hand side vector, as well as the Jacobian $\mathbf{K}$. The procedure continues until $\Delta\theta$ becomes smaller than a specified tolerance.

3.4. COMPARISONS AMONG INVERSE KINEMATICS METHODS

When computing inverse kinematics solutions, it is necessary to converge to a solution quickly. Thus, we base our comparisons between three methods on three items, namely,

- speed in calculation of the function $\mathbf{f}(\theta)$ and of its Jacobian matrix;
- condition number of the Jacobian, indicating the numerical conditioning of the problem formulation;
- overall performance of the numerical procedure based on the number of iterations needed to converge both close to a solution and away from it.

3.4.1. *Time Complexity in the Formulation of the Kinematic Model*

The formulation of the positioning equations is identical in all methods compared. The position vector $\mathbf{r}_1$ is available upon calculation of the Jacobian $\mathbf{K}$. Thus, only 3 subtractions are needed for the difference vector between the prescribed and current positions.

From the final rotation matrix product, which is calculated in Section A.1 with $117M$ and $60A$, the vector $2\mathbf{q}$ and the trace can be extracted in $5A$. Furthermore, $7A$ are needed to find the difference with the prescribed pose expressions. Thus, the computation of $\mathbf{f}(\theta)$ requires $117M$ and $72A$ in the case of linear invariants.

As shown in Section 2.1, the quadratic invariants $2\hat{\mathbf{q}}$ and $\hat{q}_0$ can be computed with $1S + 1D + 122M + 66A$, while the trace can be computed from $\hat{q}_0$ in $1M + 1A$. In the case of quadratic invariants, the computation of $\mathbf{f}(\theta)$ requires $1S + 1D + 123M + 74A$.

In the case of the method based on the natural invariants, we have first the relation $[\Delta\mathbf{R}]_{\mathscr{F}} = \mathbf{Q}_g\mathbf{Q}^T$, which requires $24M$ and $12A$, using a matrix-product scheme similar to the one outlined in Section A.1. Furthermore, $\text{vect}(\Delta\mathbf{R})$ requires $3M$ and $3A$, and thus, the derivation of the left-hand-side vector requires $144M$ and $78A$. Table II shows the operations needed to calculate the orientation equations with all three methods.

3.4.2. *Time Complexity in the Formulation of the Linear Algebraic System*

The Jacobian $\mathbf{K}$ is needed in all three formulations. The computational cost of the velocity Jacobian is calculated in Section A.5 as $12T$, $81M$ and $49A$, where T denotes trigonometric operations such as sine and cosine. The current position vector $\mathbf{r}_1$ is readily available from the computation of $\mathbf{K}$. The computation of the Jacobians of the methods based on the invariant vectors requires deviations of the auxiliary matrices and their products with $\mathbf{K}$. With linear invariants, the first three rows of $\mathbf{K}$ are multiplied by a 4×3 matrix $\mathbf{L}$, defined as

$$\mathbf{L} = \begin{bmatrix} \mathbf{1}\,\text{tr}(\mathbf{Q}) - \mathbf{Q} \\ -2\,\text{vect}(\mathbf{Q})^T \end{bmatrix}.$$

Moreover, the Jacobian expression can be written as,

$$\mathbf{J} = \begin{bmatrix} \mathbf{LA} \\ \mathbf{B} \end{bmatrix} = \begin{bmatrix} [\mathbf{1}\,\text{tr}(\mathbf{Q}) - \mathbf{Q}]\mathbf{A} \\ -2[\text{vect}(\mathbf{Q})]^T\mathbf{A} \\ \mathbf{B} \end{bmatrix},$$

Table II. Computational cost of the vector function.

	Linear inv.	Quadratic inv.	Natural inv.
Operations	$117M + 72A$	$1S + 1D + 123M + 74A$	$144M + 78A$

Table III. Computation of Jacobian **J** and vector function **f**.

Method	Operation count	CPU times (μsec)
Linear Invariants	12*T* 270*M* 172*A*	325.0
Quadratic Invariants	1*S* 12*T* 2*D* 291*M* 184*A*	345.0
Natural Invariants	12*T* 225*M* 127*A*	310.0

where **A** and **B** denote, respectively, the upper and the lower 3×6 parts of **K**, as given by Eq. (11).

It is noted that **Q**, tr(**Q**) and $2\mathbf{q}$ are available from the derivation of the function $\mathbf{f}(\theta)$. Hence, the construction of **L** takes $3A$ for the upper three rows and no operations for the fourth row. The product of **L** with **A** requires in turn $72M$ and $48A$. Thus, an additional $72M$ and $51A$ are needed for **J**, once **K** is derived.

In the case of the quadratic invariants, $\text{tr}(\sqrt{\mathbf{Q}})$, $2\hat{\mathbf{q}}$ and $\hat{q}_0$ are available from the derivation of $\mathbf{f}(\theta)$, and $\sqrt{\mathbf{Q}}$ can be computed from the quadratic invariants $\hat{\mathbf{q}}$ and $\hat{q}_0$ in $1D + 12M + 10A$, as shown in Section A.6, whereas $\hat{\mathbf{q}}$ is derived from $2\hat{\mathbf{q}}$ in $3M$. Furthermore, $\mathbf{J}'$ is calculated similar to the above case with $72M$ and $51A$.

On the other hand, using the natural invariants, the Jacobian is **K** itself, and no additional operations are needed. The number of operations and the observed CPU times needed to compute **J** and **f** are reported in Table III.

From Table III it is apparent that all three methods are comparable for the evaluation of $\mathbf{J}(\theta)$ and $\mathbf{f}(\theta)$. However, there is one more consideration: In the case of natural invariants, the Jacobian is of 6×6, and the LU-decomposition is used to solve the system of equations. The other two methods involve a 7×6 Jacobian matrix, and use Householder Reflections [20]. Because the array size is smaller and because the LU-decomposition is computationally less expensive than Householder reflections, in comparison with the other two, the third method is expected to be even less time consuming. In Table IV, the total CPU times observed for the derivation of the same solution are reported.

3.4.3. *Comparison of the Condition Numbers of the Jacobians*

The major disadvantage of the linear invariants is that the underlying Jacobian becomes singular when the angle of rotation ϕ is π. This type of singularity is known as *formulation* or *algebraic singularity*, since it arises only because of the way the rotation equations are formulated. The first three rows of matrix **L** defined above will be linearly dependent for $\phi = \pi$, and $\phi = \pm\pi/2$. However, the overall matrix

Table IV. Overall CPU times per iteration.

	Linear inv.	Quadratic inv.	Natural inv.
CPU times (μsec)	835.0	850.0	747.5

$\mathbf{L}$ is of full rank for $\phi = \pm\pi/2$, as shown in [15]. If $\mathbf{L}$ is rank-deficient, then the matrix factor $\mathbf{H}$ will also be rank-deficient. The product of a rank-deficient matrix with any other matrix being also rank-deficient, $\mathbf{J}$ will be in turn rank-deficient, and a solution cannot be obtained with the underlying numerical procedure.

At $\phi = \pi$, the matrix $\mathbf{M}$, defined as

$$\mathbf{M} \equiv \mathbf{1}\,\mathrm{tr}(\mathbf{Q}) - \mathbf{Q}$$

becomes

$$\mathbf{M}(\pi) = -2\mathbf{u}\mathbf{u}^T$$

which is a rank-one matrix, and $\mathrm{vect}(\mathbf{Q})$ vanishes by virtue of its symmetry. As shown in [15], the condition number of $\mathbf{L}$ takes on the form,

$$\kappa(\mathbf{L}) = \frac{2}{1 + \cos\phi}$$

from which the singularity at $\phi = \pi$ can be verified.

Unlike the linear invariants, in the case of quadratic invariants, the matrix $\mathbf{M}'$ defined below

$$\mathbf{M}' \equiv \mathbf{1}\,\mathrm{tr}(\sqrt{\mathbf{Q}}) - \sqrt{\mathbf{Q}}$$

remains of full rank for all possible values of the angle of rotation ϕ [15]. Therefore, the matrix $\mathbf{L}'$, defined as

$$\mathbf{L}' = \begin{bmatrix} \mathbf{1}\,\mathrm{tr}(\sqrt{\mathbf{Q}}) - \sqrt{\mathbf{Q}} \\ -2\,\mathrm{vect}(\sqrt{\mathbf{Q}})^T \end{bmatrix}$$

is always of full rank and the quadratic invariants do not lead to formulation singularities.

In the case of the natural invariants, formulation singularities do not exist, since the Jacobian $\mathbf{K}$ does not appear multiplied by any other matrix.

In order to assess the conditioning of the numerical schemes for inverse kinematics associated with each of the three rotation representations, experiments were done using closed path tracking applications with 100 data points. The said points are obtained on the intersection of two cylinders. Furthermore, Frenet–Serret frames are used [21] to specify the orientation of the EE. The maximum condition numbers encountered along the above paths are shown in Table V.

Table V. Condition number frequencies.

Range of $\kappa_{\max}$	Linear inv.	Quadratic inv.	Natural inv.
$\kappa_{\max} \leqslant 10$	0	18	0
$10 < \kappa_{\max} \leqslant 20$	50	73	29
$20 < \kappa_{\max} \leqslant 100$	33	8	71
$100 < \kappa_{\max} \leqslant 1000$	16	1	0
$\kappa_{\max} > 1000$	1	0	0

Table VI. Convergence speed in the vicinity of the solution.

	Linear inv.	Quadratic inv.	Natural inv.
Total CPU (msec)	340.0	2500.0	290.0

From Table V it is observed that the quadratic invariants allow more occurrences of lower condition numbers in the 100 points traced, whereas the linear invariants lead to high condition numbers in a large number of data points. Moreover, in this example, the natural invariants never lead to condition numbers higher than 100.

3.4.4. *Comparison on the Basis of Convergence Speed*

Since all three methods are based on approximations, convergence properties cannot be predicted theoretically. The first two methods rely on the series expansion of the function $\mathbf{f}(\theta)$ and employ the Newton–Gauss method. The third formulation is derived from the relations between joint rates and Cartesian velocities.

Convergence in the Neighbourhood of a Solution. Experiments were made in order to investigate the convergence properties of the algorithms studied. The foregoing set of data points are approached in the neighborhood of the solution, such that $\|\mathbf{f}\| < 1.0$ at the initial guess, the convergence speeds thus obtained being summarized in Table VII. Furthermore, total times spent to traverse the above three paths are measured on an *IRIS 4D/210VGX* workstation. These times are reported in Table VI. From Table VII it is observed that the natural invariants always converge in less than five iterations. Similarly, the linear invariants also converge very quickly, whereas the quadratic invariants converge much more slowly in the vicinity of the solution.

Convergence Away From the Solution. The above observations are valid if the initial guess is in the vicinity of the solution. Now observations are made when the initial guess lies far away from the prescribed set of data points. Five arbitrary solution points are selected, and two quantities are observed: (i) n, the number of iterations till convergence is reached, (ii) $\kappa_{\max}$, the maximum condition number encountered. The results of this observation are displayed in Table VIII, which shows a clear correlation between $\kappa_{\max}$ and n.

Table VII. Number of iterations to convergence.

Iterations: n	Linear inv.	Quadratic inv.	Natural inv.
$n \leqslant 5$	93	0	100
$5 < n \leqslant 10$	7	0	0
$10 < n \leqslant 20$	0	32	0
$20 < n \leqslant 40$	0	65	0
$n > 100$	0	3	0

Table VIII. Convergence speed and conditioning away from the solution.

	Linear inv.		Quadratic inv.		Natural inv.	
Test	n	$\kappa_{\max}$	n	$\kappa_{\max}$	n	$\kappa_{\max}$
1	7	26.0	21	14.0	7	39.0
2	5	9.0	19	8.0	5	14.0
3	19	60316.0	28	3407.0	26	607.0
4	5	10.0	21	10.0	5	15.0
5	4	9.0	20	8.0	4	14.0

4. Concluding Remarks

The forward kinematics problem has been solved using two different rotation representations in terms of invariants, which allow the implementation of rotation calculations with four scalar quantities and vector operations only. On the other hand, rotation calculations based on matrices require nine entries and involve matrix products. Although invariants are more elegant, they lead to more time-consuming algorithms, as shown in the computational complexity analysis. In on-line robot kinematics, it is always preferable to minimize the overhead in orientation calculations. In fact, at each iteration of the numerical procedure, the invariants have to be recalculated. Because of the significant time advantage, matrix calculations are recommended over invariant calculations for calculations pertaining to forward kinematics.

The inverse kinematics problem was solved here using three different sets of invariants. Although linear invariants are computationally less time consuming, they have the disadvantage of causing numerical instabilities in the Jacobian when the angle of rotation is close to π. On the other hand, quadratic invariants are well defined for all rotation angles and never admit formulation singularities, but their convergence rate is much slower than that of linear invariants. The third set of invariants studied here, which we call the natural invariants, consists of the unit vector parallel to the axis of rotation and the angle of rotation. Using the natural invariants, the associated Jacobian takes on a much simpler form than in the previous cases, thus reducing the overhead in the set-up time of the linear algebraic system. Secondly, because this Jacobian is square, in fact of 6×6, a faster solution technique can be employed, such as LU-decomposition. Moreover, the Jacobian used relates the joint rates and joint accelerations to the twist vector and to its time derivative, which allows not only for displacement, but also for velocity and acceleration inverse kinematics using the same Jacobian. One more advantage of the natural invariants is that they do not entail formulation singularities, the stability of the numerical procedure thus being dependent only on the configuration of the manipulator. Because of both time and stability advantages, the natural invariants are preferred in the implementation of inverse kinematics.

Further improvements in this research area should address the handling of

kinematic singularities during the numerical iterations of the displacement inverse kinematics. Although any of the above three inverse kinematics methods would come out of a singularity due to the robustness of the Newton-type methods used, a branch switching is very likely to occur. Definitely, a branch switching has to be avoided in on-line applications, for they lead to jump discontinuities in velocities and infinite discontinuities in accelerations.

Appendix

A.1. PRODUCT OF THE ROTATION MATRICES

Given the rotation matrices $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_6$ for a six-axes manipulator, the computational cost of the first product of those matrices is calculated below, an asterisk indicating a nonzero entry, while M and A denote multiplications and additions, respectively,

$$\begin{bmatrix} * & * & * \\ * & * & * \\ 0 & * & * \end{bmatrix} \begin{bmatrix} * & * & * \\ * & * & * \\ 0 & * & * \end{bmatrix} \Rightarrow \begin{bmatrix} 2M+1A & 3M+2A & 3M+2A \\ 2M+1A & 3M+2A & 3M+2A \\ 1M & 2M+1A & 2M+1A \end{bmatrix} \Rightarrow 21M+12A.$$

Therefore, the above product requires $21M$ and $12A$, each of the remaining four products requiring three additional multiplications which thus leads to $96M$ and $48A$. Hence, the computation of $\mathbf{Q}$ requires $117M$ and $60A$.

A.2. VECTOR COMPOSITION OF LINEAR INVARIANTS

Using vector calculations, the computational cost in the derivation of the linear invariants of the first product $\mathbf{Q}_1\mathbf{Q}_2$ is determined below in terms of the number of multiplications and additions required.

$$\begin{aligned}
q_0^{(1)} &= \frac{\mathrm{tr}(\mathbf{Q}_1)-1}{2} && 1M+3A,\\
q_0^{(2)} &= \frac{\mathrm{tr}(\mathbf{Q}_2)-1}{2} && 1M+3A,\\
\mathbf{q}^{(1)} &= \mathrm{vect}(\mathbf{Q}_1) && 3M+2A,\\
\mathbf{q}^{(2)} &= \mathrm{vect}(\mathbf{Q}_2) && 3M+2A,\\
(1+q_0^{(2)})\mathbf{q}^{(1)} && && 3M+1A,\\
(1+q_0^{(1)})\mathbf{q}^{(2)} && && 3M+1A,\\
\mathbf{q}^{(1)}\times\mathbf{q}^{(2)} && && 6M+3A,\\
D \equiv (1+q_0^{(1)})(1+q_0^{(2)}) && && 1M,\\
D-\mathbf{q}^{(1)}\cdot\mathbf{q}^{(2)} && && 3M+3A,\\
\frac{1}{2D} \equiv \frac{1}{D+D} && && 1D+1M,\\
\mathbf{n} \equiv (D-\mathbf{q}^{(1)}\cdot\mathbf{q}^{(2)})[(1+q_0^{(2)})\mathbf{q}^{(1)} + (1+q_0^{(1)})\mathbf{q}^{(2)} + \mathbf{q}^{(1)}\times\mathbf{q}^{(2)}] && && 3M+6A,
\end{aligned}$$

$$\mathbf{q} \equiv \frac{\mathbf{n}}{2D}$$

Thus, calculating $\mathbf{q}$ takes 1 more multiplication for a total of 1 division, 29 multiplications and 24 additions. To find the trace, one further needs to compute N:

$$\begin{aligned} \mathbf{N}' &\equiv (q_0^{(1)} + q_0^{(2)} + q_0^{(1)} q_0^{(2)}) && 1M + 2A, \\ \mathbf{q}_{12} &\equiv (\mathbf{q}^{(1)} \cdot \mathbf{q}^{(2)})(\mathbf{q}^{(1)} \cdot \mathbf{q}^{(2)} - 2D) && 1M + 1A, \\ & \quad D\mathbf{N}' && 1M. \end{aligned}$$

$$N = D\mathbf{N}' + \mathbf{q}_{12}.$$

Thus, N takes 3 more multiplications and 4 more additions after the derivation of $\mathbf{q}$. Finally,

$$q_0 = \frac{N - D}{2D},$$

Thus, q_0 will take 1 more multiplication and 1 more addition for a total of $1D + 33M + 29A$ for the computation of the linear invariants of the first product with the proposed method.

To compute the linear invariants of the matrix representing the orientation of the EE, four more products are required. For subsequent products $\mathbf{P}_i\mathbf{Q}_{i+2}$, for $i = 1, \ldots, 4$, the linear invariants of the first rotation matrix in the product is known from the previous step, and this will save $4M + 5A$. Therefore, the total cost with the proposed algorithm will be:

$$(1D + 33M + 29A) + 4[(1D + 33M + 29A) - (4M + 5A)] = 5D + 149M + 125A$$

A.3. QUADRATIC INVARIANTS FROM THE COMPOSITION OF LINEAR INVARIANTS

The proposed method of vector calculations can be extended to derive the quadratic invariants, namely,

$$\hat{q}_0 = \frac{1}{2}\sqrt{\frac{D+N}{D}}, \qquad \hat{\mathbf{q}} = \frac{\hat{q}_0}{N+D}\,\mathbf{n}.$$

The computation of $\hat{q}_0$ requires $1S + 1D + 1M + 1A$, and that of $\hat{\mathbf{q}}$ requires an additional $1D + 3M$. It is also recalled that the derivation of $\mathbf{n}$, N and D requires $1D + 31M + 28A$, which is $2M$ and $1A$ less than the derivation of $\mathbf{q}$ and q_0. Thus, the total cost for deriving the quadratic invariants for the first product is

$$1S + 3D + 35M + 29A.$$

Furthermore, the derivation of $\mathbf{n}$, N and D for the aforementioned orientation matrix requires $5D + 147M + 124A$, which is again $2M$ and $1A$ less than the number of operations required by the linear invariants of the final product. Thus, the total cost for deriving the quadratic invariants of the final product is

$$1S + 7D + 151M + 125A.$$

A.4. VECTOR COMPOSITIONS OF QUADRATIC INVARIANTS

The quadratic invariants of the end product of two rotation matrices can be derived from the quadratic invariants of the individual matrices, namely,

$$\hat{q}_0 = \hat{q}_0^{(1)}\hat{q}_0^{(2)} - \hat{\mathbf{q}}^{(1)} \cdot \hat{\mathbf{q}}^{(2)},$$

$$\hat{\mathbf{q}} = \hat{q}_0(\hat{q}_0^{(2)}\hat{\mathbf{q}}^{(1)} + \hat{q}_0^{(1)}\hat{\mathbf{q}}^{(2)} + \hat{\mathbf{q}}^{(1)} \times \hat{\mathbf{q}}^{(2)}).$$

The derivation of $\hat{q}_0$ requires $4M$ and $3A$, while $\hat{\mathbf{q}}$ requires $15M$ and $9A$. Furthermore, the derivation of each set of quadratic invariants requires $1S + 1D + 5M + 6A$. Thus, the total cost for the first product is $2S + 2D + 29M + 24A$.

For the five products we then have

$$(2S + 2D + 29M + 24A) + 4[(1S + 1D + 5M + 6A) + (19M + 12A)],$$

which gives a total of $6S + 6D + 125M + 96A$.

A.5. COMPUTATIONAL COST OF THE JACOBIAN K

Assuming that the product matrices expressing the orientation of the EE in each frame are known, we proceed to calculate the Jacobian $\mathbf{K}$ defined in Eq. (11).

Since the product of a rotation matrix with the vector $[0, 0, 1]^T$ amounts to the third column of that matrix, this product does not require any operation, and the unit vectors $\mathbf{e}_i$, for $i = 1, \ldots, 6$, are calculated at no cost. The computation of $\mathbf{r}_i$, for $i = 1, \ldots, 6$, is discussed next. Since we have

$\mathbf{r}_6 \leftarrow \mathbf{a}_6$

```
For i=5 to 1 do
```

$\quad \mathbf{r}_i \leftarrow \mathbf{a}_i + \mathbf{Q}_i\mathbf{r}_{i+1}$

```
enddo
```

we first need to calculate $\mathbf{a}_i$ for $i = 1, \ldots, 6$, defined as,

$$\mathbf{a}_i \equiv \begin{bmatrix} \cos\theta_i a_i \\ \sin\theta_i a_i \\ b_i \end{bmatrix}$$

each of which requires $2T + 2M$, for a total of $12T$ and $12M$. Next, $\mathbf{r}_6$ does not require any operation but $\mathbf{r}_i \leftarrow \mathbf{a}_i + \mathbf{Q}_i\mathbf{r}_{i+1}$ requires $8M + 7A$ each, for a total of $40M$ and $35A$ for the remaining five vectors.

Thus, the calculation of $\mathbf{r}_i$, for $i = 1, \ldots, 6$, requires $12T$, $52M$ and $35A$.

To calculate $[\mathbf{e}_i \times \mathbf{r}_i]_1$, for $i = 1, \ldots, 6$, we proceed as follows. First we have

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} -r_y \\ r_x \\ 0 \end{bmatrix}$$

and, hence, the cross products do not require any operation. However, to compute $[\mathbf{e}_2 \times \mathbf{r}_2]_1$ we have,

$$[\mathbf{e}_2 \times \mathbf{r}_2]_1 = \mathbf{Q}_1[\mathbf{e}_2 \times \mathbf{r}_2]_2 \rightarrow \begin{bmatrix} * & * & * \\ * & * & * \\ 0 & * & * \end{bmatrix} \begin{bmatrix} -r_y \\ r_x \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2M + 1A \\ 2M + 1A \\ 1M \end{bmatrix}$$

and for $[\mathbf{e}_i \times \mathbf{r}_i]_1$, for $i = 3, \ldots, 6$, we have

$$\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} -r_y \\ r_x \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2M + 1A \\ 2M + 1A \\ 2M + 1A \end{bmatrix}$$

thereby totalling $5M + 2A + 4[(6M + 3A)] = 29M + 14A$. Therefore, for the computation of the velocity Jacobian we add the cost of $\mathbf{r}_i$, for $i - 1, \ldots, 6$, and that of $[\mathbf{e}_i \times \mathbf{r}_i]_1$, for $i = 1, \ldots, 6$, which yields a total of $12T + 81M + 49A$.

A.6. ROTATION MATRIX FROM THE QUADRATIC INVARIANTS

The rotation matrix $\sqrt{\mathbf{Q}}$ can be computed from the quadratic invariants as

$$\sqrt{\mathbf{Q}} \equiv \hat{q}_0 \mathbf{1} + \frac{1}{1 + \hat{q}_0} \hat{\mathbf{q}}\hat{\mathbf{q}}^T + \hat{\mathbf{R}}$$

where $\hat{\mathbf{R}}$ is the cross-product matrix of $\hat{\mathbf{q}}$. The individual entries of the above matrix can be expressed with the aid of the auxiliary variables c, u, v, w, x, y, z as

$$\begin{aligned}
c &= \frac{1}{1 + \hat{q}_0} && 1D + 1A, \\
u &= c\hat{q}_1\hat{q}_2 && 2M, \\
v &= c\hat{q}_1\hat{q}_3 && 2M, \\
w &= c\hat{q}_2\hat{q}_3 && 2M, \\
x &= c\hat{q}_1\hat{q}_1 && 2M, \\
y &= c\hat{q}_2\hat{q}_2 && 2M, \\
z &= c\hat{q}_3\hat{q}_3 && 2M,
\end{aligned}$$

so that

$$\begin{aligned}
\hat{q}_{11} &= \hat{q}_0 + x && 1A, \\
\hat{q}_{12} &= u - \hat{q}_3 && 1A, \\
\hat{q}_{13} &= v + \hat{q}_2 && 1A, \\
\hat{q}_{21} &= u + \hat{q}_3 && 1A, \\
\hat{q}_{22} &= \hat{q}_0 + y && 1A, \\
\hat{q}_{23} &= w - \hat{q}_1 && 1A, \\
\hat{q}_{31} &= u - \hat{q}_2 && 1A, \\
\hat{q}_{32} &= w + \hat{q}_1 && 1A, \\
\hat{q}_{33} &= \hat{q}_0 + z && 1A.
\end{aligned}$$

The total cost for the above calculations thus being $1D + 12M + 10A$.

Acknowledgements

The research work reported here was made possible under NSERC (Natural Sciences and Engineering Research Council of Canada) Research Grants A4532, STR0100971 and EQP00-92729. The partial support of IRIS (Institute for Robotics and Intelligent Systems) is highly acknowledged.

References

1. Tandirci, M., Angeles, J. and Darcovich, J., The role of rotation representations in computational robot kinematics, *Proc. 1992 IEEE Int. Conf. Robotics and Automation*, Nice, May 12–14, 1992, pp. 344–349.
2. Pieper, D. L., *The kinematics of Manipulators Under Computer Control*, PhD Thesis, Stanford University, 1968.
3. Takano, M., A new effective solution to inverse kinematics problem of a robot with any type of configuration, *J. Faculty of Engineering, The University of Tokyo, Vol. B*(2), 107–135 (1985).
4. Mavroidis, C. and Roth, B., Manipulators with simple inverse kinematics, *Proc. Ninth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators – Romansy 9*, Udine, 1–4 September 1992.
5. Lee, H.-Y. and Liang, C.-G., Displacement analysis of the general spatial 7-link 7R mechanism, *Mechanism and Machine Theory* **23**(3), 219–226 (1988).
6. Raghavan, M. and Roth, B., Kinematic analysis of the 6R manipulator of general geometry, in H. Miura and S. Arimoto (eds), *Proc. 5th Int. Sympos. Rob. Res.*, MIT Press Cambridge, Mass, 1990, pp. 263–269.
7. Angeles, J., On the numerical solution of the inverse kinematics problem, *Int. J. Robotics Res.* **2**, 21–36 (1985).
8. Goldenberg, A. A., Benhabib, B. and Fenton, R. G., A complete generalized solutions to the inverse kinematics of robots, *IEEE J. Robotics Automat.* **RA-1**, 14–20 (1985).
9. 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. Mech. Transmissions Automat. Design* **107**(2), 189–200 (1985).
10. Eppinger, M. and Kreuzer, E., Evaluation of methods for solving the inverse kinematics of manipulators, Meerestechnik II – Strukturmechanik Technical report, Technische Universität Hamburg, Hamburg (1990).
11. Hartenberg, R. S. and Denavit, J., *Kinematic Synthesis of Linkages*, McGraw-Hill, New York (1964).
12. Funda, J. and Paul, R. P., A computational analysis of screw transformations in robotics, *IEEE Trans. Robot. Automat.* **6**(3), 348–356 (1989).
13. Angeles, J., *Rational Kinematics*, Springer-Verlag, New York (1988).
14. Cheng, H. and Gupta, K. C., An historical note on finite rotations, *ASME J. Appl. Mech.* **56**, 139–142 (1989).
15. Angeles, J., Die theoretischen Grundlagen zur Behandlung algebraischer Singularitäten der kinematischen Koordinatetenumkehr in der Robotertechnik, *Mech. Mach. Theory* **26**(3), 315–322 (1991).
16. Rodrigues, O., Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. *J. Math. Pures Appl.* **5**, 380–440 (1840).
17. Tandirci, M., Contributions to on-line robot kinematics, MEng Thesis, Dept of Mechanical Engineering, McGill University, Montreal (1991).
18. Whitney, D. E., The mathematics of coordinated control of prosthetic arms and manipulators, *ASME J. Dyn. Sys. Meas. Contr.* **94**(14), 303–309 (1972).
19. Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T., *Numerical Recipes in C*, Cambridge University Press, Cambridge (1988).