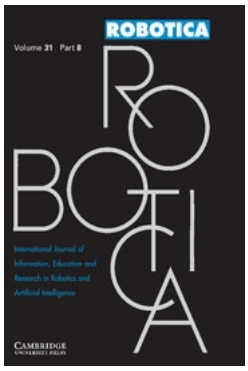

Local SVD inverse of robot Jacobians

Jing Yuan

Robotica / Volume 19 / Issue 01 / January 2001, pp 79 - 86
DOI: 10.1017/S0263574700002769, Published online: 17 January 2001

Robotica (2001) volume 19, pp. 79–86. Printed in the United Kingdom © 2001 Cambridge University Press

Local SVD inverse of robot Jacobians

Jing Yuan

Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hunghom, Kwoloon (Hong Kong) (P.R. of China)
email: mmjyan@polyu.edu.hk

(Received in Final Form: April 22, 2000)

SUMMARY
This study presents a fast inverse kinematics algorithm for a class of robots, including PUMA and SCARA. It decomposes a robot Jacobian into a product of sub-matrices to locate singularities. Singular value decomposition (SVD) is applied to each singular sub-matrix to find a local least-squares inverse. Perfect inverses are derived for all non-singular sub-matrices. The proposed algorithm is extremely fast. A total inverse requires 54 flops for PUMA and 43 for SCARA. Simulation and experiment are conducted to test the accuracy and real-time speed of the algorithm.

KEYWORDS: Robot Jacobian; Inverse kinematics; Singular value decomposition; Least-squares inverse; Cartesian space control.

1. INTRODUCTION

Most industrial robots have articulated kinematic structures similar to the arms of human beings. These kinematic structures enable robots to handle work-pieces with certain flexibility, but introduce nonlinear mappings between joint coordinates of robots and Cartesian coordinates of end-effectors. The Jacobian matrix $\mathbf{J}(\mathbf{q})$ plays an important rule in robot kinematics. It relates Cartesian velocities $\dot{\mathbf{x}} \in R^6$ of the end-effector to joint velocity $\dot{\mathbf{q}} \in R^6$ of the robot by

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

where $\mathbf{v} \in R^3$ *is the linear velocity and* $\boldsymbol{\omega} \in R^3$ angular velocity of the end-effector. For a six-joint robot, $\mathbf{J}(\mathbf{q})$ is a square matrix and a function of joint coordinate vector $\mathbf{q} \in R^6$. In many industrial applications, it is often desired to operate a robot such that the end-effector moves along specific curves or lines in Cartesian space. A computer must calculate the joint velocity by $\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}}$, so that the controller can regulate the joint velocity properly.

Unfortunately, $\mathbf{J}(\mathbf{q})$ may become singular in certain configurations known as robot singularities. The neighborhood of a robot singularity is called a robot singular region where a singular value of the Jacobian becomes smaller than a prescribed constant ε. An ideal inverse kinematics algorithm should be able to inverse $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\mathbf{q}$ regardless of robot singularities. It is expected to obtain a perfect inverse when $\mathbf{J}(\mathbf{q})$ is non-singular or a least-squares solution when the robot is inside a singular region.

Numerical singular value decomposition (SVD) meets the expectations at the expense of roughly $12n^3$ flops per inversion for an n-joint robot. The computational cost of numerical SVD prompted a study by Kirćanski and Borić[1] that resulted in a symbolic SVD algorithm for robots with the PUMA kinematic structure (PKS). The symbolic SVD is faster than numerical SVD by approximately 10 times. An experimental study by Kirćanski et al. reported a real-time implementation of the symbolic SVD in a PC for a two-joint robot.[2] For a six-joint robot, however, real-time implementation of the symbolic SVD was still not manageable with an available PC.

An alternative to SVD is the damped least squares inverse (DLSI)[3,4] that solves a continuous joint velocity $\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{q})[\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q}) + \lambda\mathbf{I}]^{-1}\dot{\mathbf{x}}$ regardless of robot singularities. Both symbolic SVD and DLSI are well accepted by researchers. The accuracy of DLSI can be improved by estimating the singular values of the Jacobian matrix.[5–7] Chiaverini et al.[8] applied DLSI to a six-joint robot and proposed the weighted DLSI to distribute the total error in user-defined directions. The closed-loop inverse kinematics (CLIK) algorithm[9,10] also uses DLSI to resolve the joint rates or joint accelerations.

This study presents a fast SVD algorithm for robots with PKS. It only requires 54 flops per inversion – the most efficient SVD algorithm to the author's best knowledge. Some robots, such s the ABB IRb-200, eliminate the elbow singularity by restricting the range of a joint. The corresponding inverse kinematics problem requires 48 flops by the proposed method. The algorithm has been tested in a real-time experiment where a single 33MHz-486 controls a six-joint Zebra-0 robot (IMI, Berkeley, CA, USA). The robot manufacturer allows 1.5 milliseconds per sampling interval for a user program to compute the next target point. The proposed algorithm managed to resolve joint rate within the limit for a trajectory that starts in the intersection of the shoulder and wrist singularities and repetitively crosses the shoulder singularity. The algorithm can be extended to other robots with spherical wrists. When applied to the SCARA robot, for example, only 43 flops are required to solve the inverse kinematics problem.

The paper is organized in the following way: Section 2 presents the algorithm with a detailed analysis of computational counts. Simulation and experimental results are presented in Section 3. Section 4 explains a possible

application of the proposed algorithm in conjunction with the CLIK method. A brief conclusions is given in Section 5.

2. THE FAST SVD INVERSE ALGORITHM

The subject of this section is the Jacobian of PKS. The last three joints of these robots are rotational with their axes intersecting at a common point to form a spherical wrist. Since many industrial robots belong to this class, it is chosen to be the subject of this study. The result, however, can be extended to any six-joint robots with spherical wrists.

Figure 1 shows a PKS skeleton and its home configuration. The *i*th link is associated with a coordinate frame $\{\mathbf{o}_i, \mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$ were $\mathbf{o}_i$ is the origin while $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ are the axes. These symbols are not explicitly labeled in Figure 1. Instead, each frame is represented by three axes embedded in a solid cylinder representing the pivot of a joint. The centre of the *i*th cylinder is the origin of the *i*th frame. The $\mathbf{z}_i$ axis is the pivot axis, with $\mathbf{x}_i$ and $\mathbf{y}_i$ determined by the right hand rule. The base frame $\{\mathbf{o}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0\}$ is attached to the base cylinder of the skeleton. Other frames are counted upwards. The sixth frame is not shown in the picture. It is to be defined by the end user according to the geometric shape of the end effector. This fact, however, does not affect the analysis.

The Jacobian can be obtained by a simple method explained in reference [11]. Let $\mathbf{J}_i$ denote the *i*th column of $\mathbf{J}(\mathbf{q})$, then it is given by

$$\mathbf{J}_i = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{o}_6 - \mathbf{o}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \times \mathbf{d}_{i-1} \\ \mathbf{z}_{i-1} \end{bmatrix} + \begin{bmatrix} \mathbf{z}_{i-1} \times \mathbf{h} \\ \mathbf{0} \end{bmatrix} \tag{2}$$

where $\mathbf{d}_{i-1} = \mathbf{o}_4 - \mathbf{o}_{i-1}$, $\mathbf{h} = \mathbf{o}_6 - \mathbf{o}_4$ or $\mathbf{d}_{i-1} + \mathbf{h} = \mathbf{o}_6 - \mathbf{o}_{i-1}$. The wrist of PKS, shown in Figure 1, coincides with $\mathbf{o}_4$ where the last three axes intersect. It is related to the origin of the sixth frame by $\mathbf{h} = \mathbf{o}_6 - \mathbf{o}_4$. Such a notation allows the user to specify $\mathbf{o}_6$ at any point on the end-effector regardless of Jacobian inverse.

It is a popular approach to express the end-effector velocity down to the robot wrist by $\mathbf{v}_\omega = \mathbf{v} - \boldsymbol{\omega} \times \mathbf{h}$.[12,13] Such a treatment reduces the inverse kinematic problem to

$$\dot{\mathbf{q}} = \mathbf{J}_w^{-1}(\mathbf{q}) \begin{bmatrix} \mathbf{v}_w \\ \boldsymbol{\omega} \end{bmatrix} \tag{3}$$

where a block-triangle matrix

$$\mathbf{J}_w(\mathbf{q}) = \begin{bmatrix} \mathbf{z}_0 \times \mathbf{d}_0 & \mathbf{z}_1 \times \mathbf{d}_1 & \mathbf{z}_2 \times \mathbf{d}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{z}_0 & \mathbf{z}_1 & \mathbf{z}_2 & \mathbf{z}_3 & \mathbf{z}_4 & \mathbf{z}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \tag{4}$$

is the Jacobian of the robot wrist, hence the subscript "w". One may express $\mathbf{J}_w(\mathbf{q})$ as

$$\mathbf{J}_w(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ \mathbf{J}_{21} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{J}_{22} \end{bmatrix} \tag{5}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix. It is not difficult to see

$$\begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ \mathbf{J}_{21} & \mathbf{I}_3 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ -\mathbf{J}_{21} & \mathbf{I}_3 \end{bmatrix} \tag{6}$$

Substituting (6) into (5), one obtains

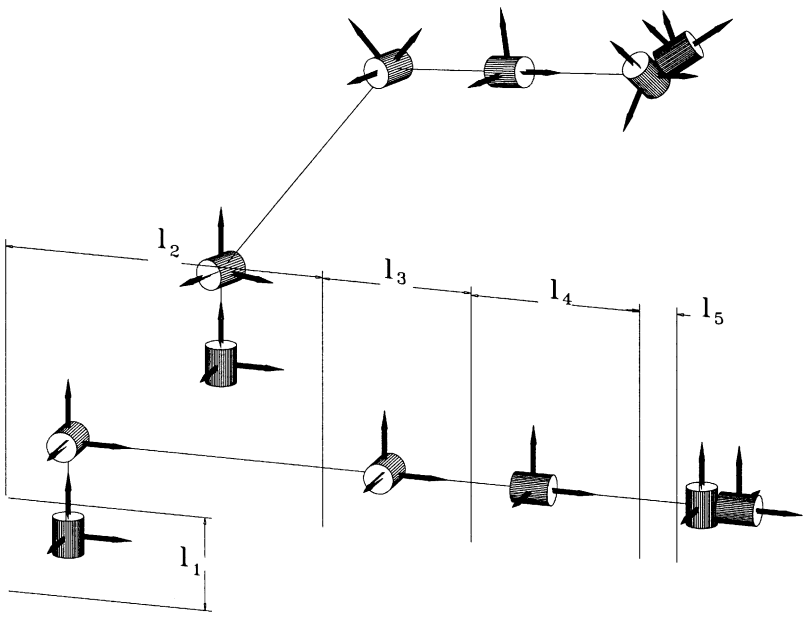

Fig. 1. A PKS skeleton and its home configuration.

Table I. Flop count of $\mathbf{f}_1 = \mathbf{J}_{11}^{\dagger}\mathbf{v}_w$.

Matrix ×	$\mathbf{\Omega}_1$	$\mathbf{\Omega}_2$	$\mathbf{\Omega}_3$	$\mathbf{\Omega}_4$	$\mathbf{C}_1$
Flops	6	6	10	3	25

$$\mathbf{J}_w^{-1}(\mathbf{q}) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ \mathbf{O} & \mathbf{J}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I}_3 & \mathbf{O} \\ -\mathbf{J}_{21} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{J}_{11}^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_3 \end{bmatrix}. \quad (7)$$

Only two 3×3 inverse matrices, $\mathbf{J}_{11}^{-1}$ and $\mathbf{J}_{22}^{-1}$, are needed to construct a 6×6 inverse $\mathbf{J}_w^{-1}(\mathbf{q})$. Let $\mathbf{J}_{11}^{\dagger}$ and $\mathbf{J}_{22}^{\dagger}$ denote, respectively, the SVD least-square inverses of $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$. Substituting into (7), one may solve (3) in 3 steps:

$$\mathbf{f}_1 = \mathbf{J}_{11}^{\dagger}\,\mathbf{v}_w, \quad (8)$$

$$\mathbf{f}_2 = \omega - \mathbf{J}_{21}\mathbf{f}_1, \quad (9)$$

$$\mathbf{f}_3 = \mathbf{J}_{22}^{\dagger}\mathbf{f}_2. \quad (10)$$

The inverse kinematic solution is given by $\dot{\mathbf{q}}^T = [\mathbf{f}_1^T\ \mathbf{f}_3^T]$.

Since the computation of $\mathbf{v}_w$ is a common exercise required by all inverse kinematics algorithms, most researchers, such as Kirćanski and Borić,[1] excluded these computations when comparing efficiency of inverse kinematics algorithms. By the same token, total flop count of the present algorithm is based on (8)–(10). Let $\mathbf{C}_i$, $\mathbf{C}_2$ and $\mathbf{C}_3$ denote, respectively, flop counts of these steps, then the present algorithm has a total flop count of $\mathbf{C}_1+\mathbf{C}_2+\mathbf{C}_3$.

2.1 Flop count of $f_1 = J_{11}^{\dagger}v_w$

Table I lists a sub-total of flops needed to obtain $\mathbf{f}_1 = \mathbf{J}_{11}^{\dagger}\mathbf{v}_w$. Since $\mathbf{J}_{11}$ only involves the first three joint angles, its analytical version is derivable from Figure 1 as

$$\mathbf{J}_{11} = \begin{bmatrix} -\alpha c_1 & (l_2 s_2 + \bar{l}_3 s_{23}) s_1 & \bar{l}_3 s_{21} s_{23} \\ -\alpha s_1 & -(l_2 s_2 + \bar{l}_3 s_{23}) c_1 & -\bar{l}_3 c_1 s_{23} \\ 0 & \alpha & \bar{l}_3 c_{23} \end{bmatrix}$$

$$\bar{l}_3 = l_3 + l_4 \quad (11)$$

where $\alpha = l_2 c_2 = \bar{l}_3 c_{23}$ is the (2,3)th element of $\mathbf{J}_{11}$ without any on-line computation when $\mathbf{J}_w(\mathbf{q})$ is available.

Evidently, (11) depends on the definition of home position and selections of frames attached to the first three links. While the expression of (11) may change for different home configurations or frame selections, the principle developed here remains valid. One can use (11) without losing generality. A closer examination of (11) reveals

$$\mathbf{J}_{11} = \begin{bmatrix} -\alpha c_1 & -s_1 & 0 \\ -\alpha s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -s_2 & -s_{23} \\ 0 & c_2 & c_{23} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & l_2 & 0 \\ 0 & \bar{l}_3 & \bar{l}_3 \end{bmatrix} \quad (12)$$

This structure implies an analytical inverse of $\mathbf{J}_{11}$, in the form of

$$\mathbf{J}_{11}^{\dagger} = \mathbf{\Omega}_4\mathbf{\Omega}_3\mathbf{\Omega}_2\mathbf{\Omega}_1 \quad (13)$$

where

$$\mathbf{\Omega}_1 = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{\Omega}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -s_2 & c_2 \\ 0 & -s_{23} & c_{23} \end{bmatrix},$$

$$\mathbf{\Omega}_3 = \begin{bmatrix} -\frac{1}{\alpha} & 0 & 0 \\ 0 & \left(\frac{0.5}{1+c_3}+\frac{0.5}{1-c_3}\right) & \left(\frac{0.5}{1+c_3}-\frac{0.5}{1-c_3}\right) \\ 0 & \left(\frac{0.5}{1+c_3}-\frac{0.5}{1-c_3}\right) & \left(\frac{0.5}{1+c_3}+\frac{0.5}{1-c_3}\right) \end{bmatrix} \text{ and}$$

$$\mathbf{\Omega}_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & l_2^{-1} & 0 \\ 0 & -l_2^{-1} & \bar{l}_3^{-1} \end{bmatrix}.$$

Since $\mathbf{\Omega}_3$ contains the shoulder singular value $|\alpha|$ and elbow singular value $1\pm c_3$, $\frac{1}{\alpha}$ should be replaced by $\frac{sgn(\alpha)}{\varepsilon}$, or $\frac{0.5}{1\pm c_3}$ by $\frac{0.5}{\varepsilon}$, whenever $|\alpha|$ or $1\pm c_3$ is smaller than ε. Some industrial robots, such as the ABB IRb-2000, restricts the range of q_3 to eliminate the elbow singularity. Then (13) can be simplified to

$$\mathbf{J}_{11}^{\dagger} = \mathbf{\Omega}_4 \begin{bmatrix} -\frac{1}{\alpha} & 0 & 0 \\ 0 & \frac{c_{23}}{s_3} & \frac{s_{23}}{s_3} \\ 0 & -\frac{c_2}{s_3} & -\frac{s_2}{s_3} \end{bmatrix} \mathbf{\Omega}_1. \quad (14)$$

It is not difficult to verify $\mathbf{J}_{11}^{\dagger}\mathbf{J}_{11} = \mathbf{I}$ for either (13) or (14) by routine multiplications of the sub-matrices.

Table I is based on (13) that corresponds to the worst case with both shoulder and elbow singularities. It takes 6 flops, another 6 flops and then 3 flops, respectively, to multiply $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$ and $\mathbf{\Omega}_4$ since elements of these sub-matrices are either pre-computed or available as intermediate variables from the construction of $\mathbf{J}_{11}$. To count the computations required for multiplying $\mathbf{\Omega}_3$, one needs a further decomposition

$$\mathbf{\Omega}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{\alpha} & 0 & 0 \\ 0 & \frac{0.5}{1+c_3} & 0 \\ 0 & 0 & \frac{0.5}{\epsilon} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix}$$

where $\frac{0.5}{\epsilon}$ replaces $\frac{0.5}{1-c_3}$ assuming $-\delta < q_3 < \delta$ and $\delta = \cos^{-1}\epsilon$. If $-\delta < q_3 - \pi < \delta$, then $\frac{0.5}{1+c_3}$ should be replaced by $\frac{0.5}{\epsilon}$ instead. The algorithm detects the elbow singularity by a negligible integer operation in the encoder level using δ. The flop count, shown in Table I, is $2+6+2=10$ for multiplying $\mathbf{\Omega}3$. It includes one flop for detecting the shoulder singularity, but excludes the computation of $\frac{0.5}{\epsilon}$ – a pre-computed constant. If the algorithm detects that the robot is outside the elbow singular region, it automatically switches to (14) in the first place. That reduces $\mathbf{C}_1$ to $6+10+3=19$ flops (including one flop for detecting the shoulder singularity) instead of 25.

2.2 Flop count of $f_2 = \omega - J_{21}f_1$

This is the easiest step where $\mathbf{J}_{21} = [\mathbf{z}_0\ \mathbf{z}_1\ \mathbf{z}_2]$. Since $\mathbf{z}_0 = [0\ 0\ 1]^T$ and $\mathbf{z}_1 = \mathbf{z}_2 = [c_1\ s_1\ 0]^T$, one can write

Table II. Flop count of $\mathbf{f}_3 = \mathbf{J}_{22}^{\dagger}\mathbf{f}_2$.

Matrix ×	$\mathbf{J}_{22}^T$	$(\mathbf{J}_{22}^T\mathbf{J}_{22})^{\dagger}$	$\mathbf{C}_3$
Flops	15	8	23

$$\mathbf{f}_2 = \boldsymbol{\omega} - \mathbf{J}_{21}\mathbf{f}_1 = \begin{bmatrix} \omega_1 - f_\sigma c_1 \\ \omega_2 - f_\sigma s_1 \\ \omega_3 - f_{11} \end{bmatrix},$$

where $f_\sigma = f_{12} + f_{13}$, $\mathbf{f}_1 = [f_{11}\ f_{12}\ f_{13}]^T$ and $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3]^T$. This step requires $\mathbf{C}_2 = 6$ flops.

2.3 Flop count of $F_3 = J_{22}^{\dagger} f_2$

The least squares inverse of $\mathbf{J}_{22}$ is expressed as $\mathbf{J}_{22}^{\dagger} = (\mathbf{J}_{22}^T\mathbf{J}_{22})^{\dagger}\mathbf{J}_{22}^T$ (Table II) This step involves two matrix multiplications. The first one is a multiplication by $\mathbf{J}_{22}^T$ that requires $3 \times 5 = 15$ flops. The second one is a multiplication by $(\mathbf{J}_{22}^T\ \mathbf{J}_{22})^{\dagger}$. The three columns of $\mathbf{J}_{22}$ represent the axes of the spherical wrist. These are unit-length vectors. Particularly, $\mathbf{z}_4$ is always perpendicular to both $\mathbf{z}_3$ and $\mathbf{z}_5$. This fact implies

$$\mathbf{J}_{22}^T\mathbf{J}_{22} = \begin{bmatrix} 1 & 0 & c_5 \\ 0 & 1 & 0 \\ c_5 & 0 & 1 \end{bmatrix}.$$

Therefore $(\mathbf{J}_{22}^T\mathbf{J}_{22})^{\dagger}$ has two versions. Inside the wrist singular region $-\delta < q_5 < \delta$, it has a SVD expression

$$(\mathbf{J}_{22}^T\mathbf{J}_{22})^{\dagger} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{0.5}{1+c_5} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{0.5}{\epsilon} \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

where $\frac{0.5}{\epsilon}$ substitutes $\frac{0.5}{1-c_5}$. If $-\delta < q_5 - \pi < \delta$, then $\frac{0.5}{1+c_5}$ should be replaced by $\frac{0.5}{\epsilon}$ instead. This multiplication requires $2+4+2=8$ flops. Outside the wrist singular region, $(\mathbf{J}_{22}^T\mathbf{J}_{22})^{\dagger}$ is a perfect inverse given by

$$(\mathbf{J}_{22}^T\mathbf{J}_{22})^{-1} = \begin{bmatrix} \frac{1}{s_5^2} & 0 & \frac{-c_5}{s_5^2} \\ 0 & 1 & 0 \\ \frac{c_5}{s_5^2} & 0 & \frac{1}{s_5^2} \end{bmatrix}. \quad (15)$$

It requires 7 flops. Therefore $\mathbf{C}_3 = 23$ flops and $\mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3 = 54$ flops. For some industrial robots, such as ABB IRb-2000, the elbow singularity is eliminated by limiting the range of q_3. The computation is reduced to 48 flops.

2.4 Computation savings from Jacobian construction

The present algorithm does not necessarily require a complete reconstruction of $\mathbf{J}_w(\mathbf{q})$. Step 1 only requires $\cos q_1$, $\sin q_1$, $\cos q_2$, $\sin q_2$, $\cos q_3$, $\sin q_3$, $\cos(q_2+q_3)$ and $\sin(q_2+q_3)$ plus $\alpha = l_2\cos q_2 + l_3\cos(q_2+q_3)$. This is significantly more efficient than constructing a complete $\mathbf{J}_{11}$. Step 2 uses $\cos q_1$ and $\sin q_1$ available from step 1. Only step 3 requires $\mathbf{J}_{22}$ plus $\cos q_5$ or $\sin q_5$. As a whole, the algorithm saves many computations required to construct $\mathbf{J}_w(\mathbf{q})$ in addition to its fast speed in the inverse process.

2.5 Extension to robots with other kinematic structures

The present algorithm also applies to robot with other kinematic structures, as stated in the introduction. The six-axis gantry robot is a trivial example. It has a spherical wrist and three translational axes, rendering $\mathbf{J}_w(\mathbf{q})$ an identity matrix. The SCARA kinematic structure is another popular robot kinematic structure, which is suitable for the proposed algorithm. With a translational joint, SCARA does not have the shoulder singularity. Its $\mathbf{J}_w(\mathbf{q})$ differs from that of PKS in

$$\mathbf{J}_{11} = \begin{bmatrix} -s_1 & -s_{12} & 0 \\ c_1 & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_1 & 0 & 0 \\ l_2 & l_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and}$$

$$\mathbf{J}_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The SVD inverse of $\mathbf{J}_{11}$ has a cascade form:

$$\mathbf{J}_{11}^{\dagger} = \begin{bmatrix} l_1^{-1} & 0 & 0 \\ -l_1^{-1} & l_2^{-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{\Lambda}_{11}^{\dagger} \begin{bmatrix} -s_1 & c_1 & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The worst case scenario happens when the robot enters its elbow singular region. In that case, multiplication by $\boldsymbol{\Lambda}_{11}^{\dagger}$ can be carried out in the following sub-steps

$$\boldsymbol{\Lambda}_{11}^{\dagger} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{0.5}{1+c_2} & 0 & 0 \\ 0 & \frac{0.5}{\varepsilon} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\frac{0.5}{\varepsilon}$ substitutes $\frac{0.5}{1-c_2}$ assuming $-\delta < q_2 < \delta$. If $-\delta < q_2 - \pi < \delta$, then $\frac{0.5}{1+c_2}$ should be replaced by $\frac{0.5}{\varepsilon}$ instead. Similar to the inverse kinematics of PKS, less computations are needed when SCARA is outside its elbow singular region.

With a simpler Jacobian than that of PKS, the inverse kinematics of SCARA requires less computations. Step 1 requires $\mathbf{C}_1 = 6+8+3=17$ flops. Steps 2 and 3 require $\mathbf{C}_2 = 3$ flops and $\mathbf{C}_3 = 23$ flops respectively. The total flop count is therefore $\mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3 = 43$.

3. SIMULATION AND EXPERIMENT

The simulation needs a DLSI algorithm as the standard to compare the accuracy of the proposed algorithm. There are many different varieties of DLSI. The one chosen here is given by

$$\dot{\mathbf{q}} = \mathbf{J}^T[\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}]^{-1}\dot{\mathbf{x}} \quad \lambda^2 = \begin{cases} 0 & \text{if } \sigma_{min}^2 \geq \epsilon^2; \\ \epsilon^2 - \sigma_{min}^2 & \text{otherwise} \end{cases} \quad (16)$$

where σ_{min}^2 is the smallest eigenvalue of $\mathbf{J}^T\mathbf{J}$ and $\epsilon > 0$ is a prescribed constant that determines the singular regions of a robot. Let $\mathbf{J} = \sum_{i=1}^{6} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ denote the SVD of $\mathbf{J}$, then (16) is equivalent to

$$\dot{\mathbf{q}} = \sum_{i=1}^{6} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{u}_i^T \dot{\mathbf{x}}. \quad (17)$$

The varying λ is intended to render $\frac{\sigma_i}{\sigma_i^2+\lambda^2} = \frac{1}{\sigma_i}$ when the robot is outside its singular regions and reduce the approximation error when the robot is inside singular regions. This algorithm is computational expensive because of σ_{min}^2. Several algorithms were proposed [5, 6] to use an estimate $\hat{\sigma}_{min}^2$ instead of computing σ_{min}^2. It has been discovered that the accuracy of $\hat{\sigma}_{min}^2$ deteriorates when the two smallest eigenvalues of $\mathbf{J}^T\mathbf{J}$ cross over. A better solution is to estimate the two smallest eigenvalues of $\mathbf{J}^T\mathbf{J}$ simultaneously. That requires more computations.[7,8] Since the accuracy of DLSI depends on the accuracy of $\hat{\sigma}_{min}^2$, (16) should be the most accurate DLSI algorithm as it uses the exact value of σ_{min}^2.

Evidently, the proposed method is equivalent to DLSI when the Jacobian is non-singular. Both produce exact inverses of $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$. The two methods take different approaches in singular regions, and produce different solutions. The damping of DLSI affects all the singular values as shown in (17). The proposed method, however, handles individual singularities separately. In case of the shoulder singularity, for example, the proposed method only replaces $|\alpha|$ by ϵ in (14). It does not make any other changes unless the robot enters the intersection of the shoulder singular region and other singular regions. Such a treatment avoids errors in the non-singular directions of the Cartesian velocity vector. An exact solution is still possible in a singular region if the Cartesian velocity is orthogonal to the singular direction.

Both methods are applied to the robot skeleton shown in Figure 2 with $l_2 = l_3 = l_3 + l_4 = 0.85$ meters and $l_5 = 0.1$ meters, which is similar to the kinematic structure of an ABB IRb 2000. The initial configuration of the skeleton is plotted in Figure 3, which is the intersection of the shoulder and wrist singularities.

The inverse kinematics algorithms are supposed to compute $\dot{\mathbf{q}}$ such that the robot end effector translates along a linear path with a desired velocity of

$$\dot{\mathbf{p}}^T = [0.05, 0.2, 0.2] \cos(\frac{\pi}{2} t) = \dot{\mathbf{x}}_p^T \cos(\frac{\pi}{2} t) \text{ and } \boldsymbol{\omega} = \mathbf{0}.$$

Since a robot loses one or more degrees of freedom at its singularities, its actual movement will inevitably lag behind the desired trajectory no matter what algorithms are used to compute $\dot{\mathbf{q}}$, as observed in references [15, 16]. In this study, both algorithms apply the feedback error correction by replacing $\dot{\mathbf{x}}$ with $\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e}$ where $\dot{\mathbf{x}}_d$ is the desired Cartesian velocity, $\mathbf{K} = 20\mathbf{I}$ and $\mathbf{e}$ defined in reference 8.

Following the choice of reference 8, $\epsilon = 0.04$ is selected to prevent numerical overflow. Yet it is still possible that some of the components of $\dot{\mathbf{q}}$ have excessively large magnitudes, since the robot joint velocity range is usually much smaller than the floating point numerical range. For this reason, the components of $\dot{\mathbf{q}}$ are hard limited according to Table III for both algorithms. The values of this table are chosen exactly the same as the joint speed ranges of an ABB IRb 2000.[8]

Table III. The joint speed ranges of the robot.

Joint	q_1	q_2	q_3	q_4	q_5	q_6
Max speed (rad/s)	2.01	2.01	2.01	4.89	5.24	5.24

There are infinitely many inverse kinematics solutions for the position and orientation shown in Figure 3. One alternative solution is shown in Figure 4. The two configurations share the same wrist center $\mathbf{o}_4$. With a proper q_6, the alternative configuration achieves the same end-effector position and orientation but avoids the wrist

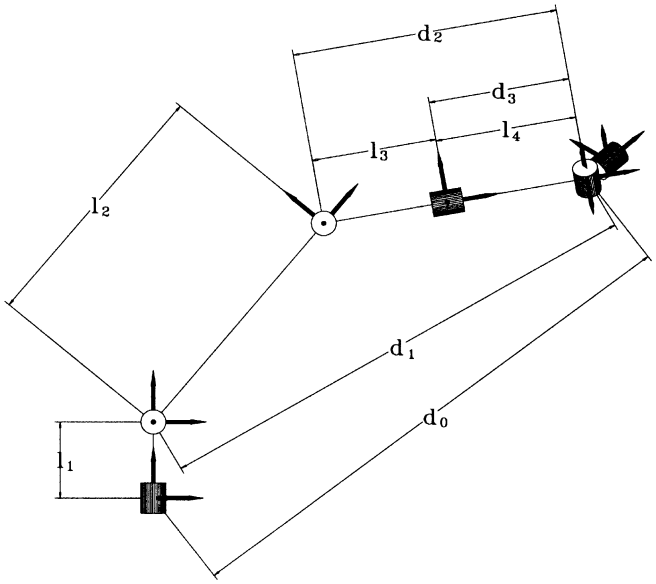

Fig. 2. A different view of the robot skeleton.

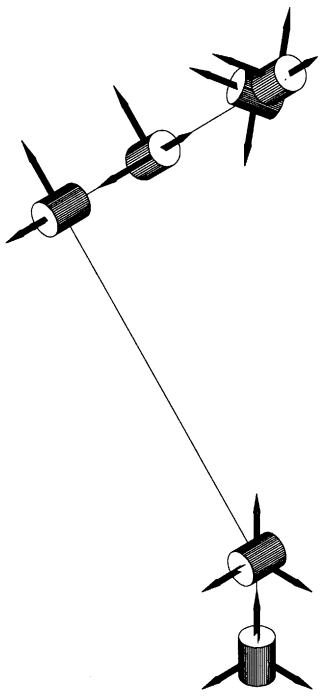

Fig. 3. Initial configuration.

singularity. A more important difference between these two configurations is related to the prescribed velocity vector $\dot{\mathbf{x}}_p$ For Figure 3, the shoulder singular direction is $\mathbf{z}_1 = [1\ 0\ 0]^T$. It can be shown that $\mathbf{x}_1^T \dot{\mathbf{x}}_p = 0.05$. The robot is not able to move in this direction when it is in the shoulder singularity. A large tracking error is inevitable in such a situation. This is, however, not the case for Figure 4 where the $\mathbf{z}_1$ axis has been rotated to $\mathbf{z}_1 = [\frac{0.2}{\sqrt{0.05^2+0.2^2}} - \frac{0.05}{\sqrt{0.05^2+0.2^2}}\ 0]^T$. This differences causes $\mathbf{z}_1^T \dot{\mathbf{x}}_p = 0$. The corresponding shoulder singular direction is orthogonal to the prescribed velocity vector. If a robot is at the configuration shown in Figure 4, it has no difficulty moving in the direction of $\dot{\mathbf{x}}_p$.

It is interesting that both methods adjust the joint positions and velocities in such a way that the robot manages to avoid the wrist singularity when the end effector returns to its starting position after one period of the motion. Although the shoulder singularity can not be avoided, the robot automatically positions itself to the configuration shown in Figure 4. As the result, the least-squares inverses of both methods are able to direct the normal robot motion along the prescribed path when crossing the shoulder singularity again.

The simulation results are plotted in Figure 5, which uses solid lines to plot the results of the proposed method, and dashed lines to plot the results of DLSI. The tracking error norms exhibit a large peak at the initial transient period and then quickly converge to small values. Both methods demonstrate good tracking performance, with similar joint velocity profiles as shown in Figure 5.

The proposed method is also tested in a real-time experiment, using a six-joint Zebra-0 made by Integrated Motions Inc. (Berkeley, CA, USA). The robot is controlled by a 33MHz 486, and its control software has a C function run_user_path() that can test the speed of the proposed algorithm. When called by a user program, run_user_path() samples joint angle and end-effector force 140 times per second. During each sampling interval, it calls a user-defined function

```
int next_path_point(vect6 current_angle, vect6 next_
angle, vect6 current_force)
{
    next_angle = current_angle + J†(q)(ẋ_d + Ke)dt;
    // user codes here.
}
```

to provide "current_angle" and "current_force". The user must code next_path_point() properly such that it finishes next_angle = current_angle + $\mathbf{J}^{\dagger}(\mathbf{q})(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e})dt$ within 1.5 ms. If next_path_point() takes more than 1.5 ms to execute, then run_user_path() will lose synchronization with internal control functions. It has to abort execution and leave the robot stopped wherever it is.

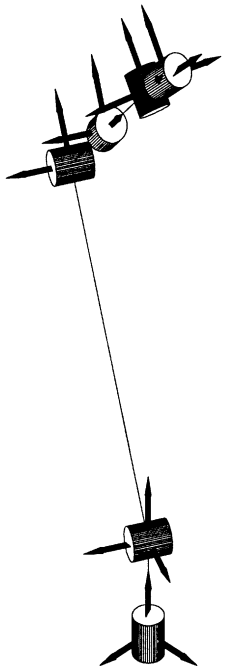

Fig. 4. An alternative configuration.

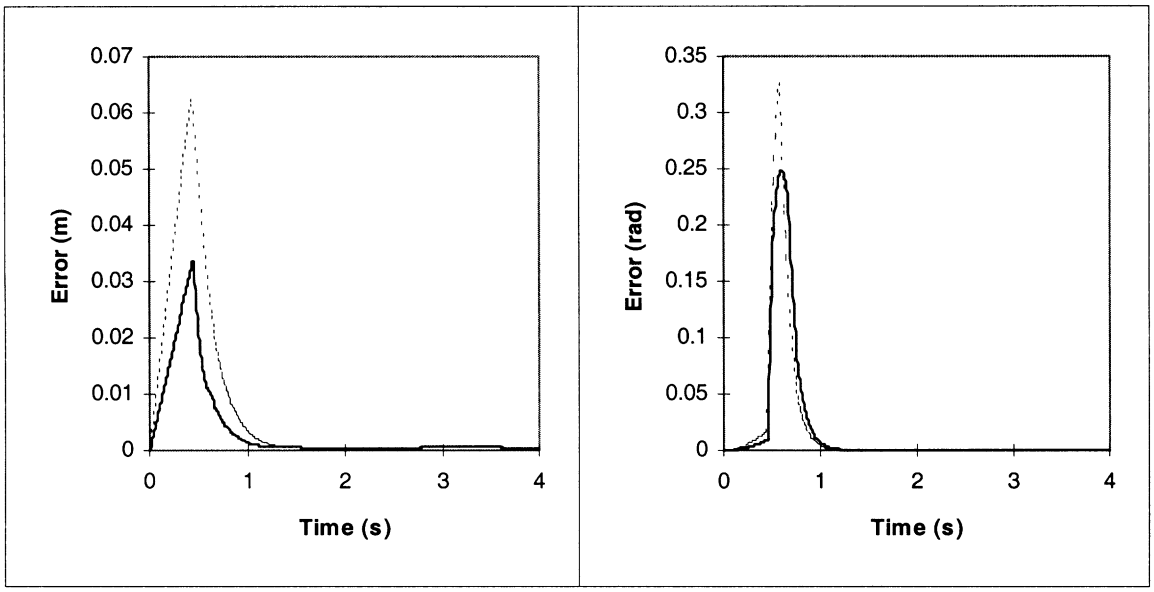

Fig. 5. Joint velocities profiles of two methods.

The proposed algorithm is written as user codes in next_path_point(), including all necessary computations required by the inverse kinematics. The Zebra-0 is approximately $\frac{1}{4}$ the size of the simulated robot. Therefore the simulated path must be scaled down by $\frac{1}{4}$ to fit the work space of Zebra-0. It is a constraint of Zebra-0 that each component of next_angle be within 1 degree of that of current_angle. The desired speed of the path has to be scaled down to

$$\dot{\mathbf{p}}^T = [0.01, 0.05, 0.05] \cos(\frac{\pi}{5} t) \quad \text{and} \quad \boldsymbol{\omega} = \mathbf{0}.$$

The initial configuration of Zebra-0 is chosen to be the same as that of simulation study – right in the shoulder and wrist singularities. The experiment runs smoothly with tracking results similar to that of the simulation. It proves that the inverse kinematics problem is solvable within 1.5 ms by a 33MHz 486 running debug mode of Microsoft Quick C. Since the sampling rate of run_user_path() is fixed by the manufacturer, no attempts are made to increase the sampling rate and test the maximum speed of the proposed algorithm.

4 APPLICATION OF THE ALGORITHM

The present algorithm obtains a perfect solution $\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}}$ in the normal region of a robot work space. Let $\mathbf{I}$ and $\mathbf{O}$ denote, respectively, an identity and an all-zero matrix. Then one can write

$$\mathbf{J}^{\dagger}(\mathbf{q}) = \mathbf{J}^{-1}(\mathbf{q}) \quad \text{or} \quad \boldsymbol{\Delta} = \mathbf{I} - \mathbf{J}(\mathbf{q})\mathbf{J}^{\dagger}(\mathbf{q}) = \mathbf{O} \qquad (18)$$

whenever the robot is in its normal region of work space. An application of the proposed algorithm is to the CLIK originated by Chiacchio et al.[9] A second-order CLIK was proposed Caccavale et al.[10] It can be expressed as

$$\ddot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})[\ddot{\mathbf{x}}_d - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}] \qquad (19)$$

where $\mathbf{x}_d$ denotes the desired Cartesian trajectory and $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ the tracking error in Cartesian space. $\mathbf{K}_D$ and $\mathbf{K}_P$ are positive definite gain matrices. Substituting (19) into robot kinematic equation

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}, \qquad (20)$$

one obtains

$$\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} = \mathbf{0}, \qquad (21)$$

whenever the robot is in the normal region of its work space and (18) becomes valid. It implies an asymptotically stable control system if the robot stays away from its singularities.

If a robot has to cross one of its singularities, Caccavale et al. showed that the control loop is still stable[10] and the tracking error is bounded. The error bound was shown proportional to $\boldsymbol{\Delta} = \mathbf{I} - \mathbf{J}(\mathbf{q})\mathbf{J}^{\dagger}(\mathbf{q})$ and several measures were proposed to reduce the tracking error. A most effective measure seems to be an advanced damping scheme similar to the one given by (16). It requires on-line computation of σ^2_{min} and the computation cost will be increased significantly. According to the results of Section 3, the proposed algorithm is able to achieve the same objective at a very low implementation cost. It is therefore an efficient tool for CLIK. Another potential application of the proposed algorithm is on-line hybrid force-trajectory tracking, which is a subject of further study.

5 CONCLUSION

This paper presents a fast inverse kinematics algorithm for a class of six-joint robots, including PUMA and SCARA. The algorithm decomposes the Jacobian into a product of sub-matrix and solves the SVD inverse in the local sense. The algorithm is extremely fast, with each inverse costing 54 flops for PUMA and 43 for SCARA respectively. Simulation and experiment are conducted to test the proposed algorithm with good tracking performance.

References

1. M.V. Kirćanski and M.D. Borc, "Symbolic singular value decomposition for a PUMA robot and its application to a robot operation near singularities," *Int. J. Robotics Research* **12**, No. 5, 460–472 (1993).
2. M.V. Kirćanski, N. Kirćanski, D. Leković and M. Vukobratović, "An experimental study of resolved acceleration control of robots at singularities: damped least-squares approach", *ASME J. Dynamic Systems, Measurement and Control* **119**, No. 1, 97–101 (1997).
3. C.W. Wampler II, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst., Man, Cybern.* **SMC-16**, 93–101 (1986).
4. Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *ASME J. Dyn. Syst., Meas., Contr.* 109, 163–171 (1986).
5. C.A. Klein and B.E. Blaho, "Dexterity measures for the dsign and control of kinematically redundant manipulators", *Int. J. Robot Res.* **6**, No. 2, 72–83 (1987).
6. I. Spangelo, J.R. Sagli and O. Egeland, "Bounds on the largest singular value of the manipulator Jacobian," *IEEE Trans. Robot. Auto.* **9**, 93–96 (1993).
7. S. Chiaverini, "Estimate of the two smallest singular values of the Jacobian matrix: Application to damped least-squares inverse kinematics," *J. Robotic Syst.* **10**, No. 8, 991–1008 (1993).
8. S. Chiaverini, B. Siciliano and O. Egeland, "Review of the damped least-squares invrse kinematics with experiments on an industrial robot manipulator," *IEEE Trans. Control Systems Technology* **2**, No. 2, 123–134 (June 1994).
9. P. Chiacchio, S. Chiaverini, L. Sciavicco and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *Int. J. Robotics Research* **10**, 410–425 (1991).
10. F. Caccavale, S. Chiaverini and B. Siciliano, "Second-order kinematic control of robot manipulators with Jacobian damped least squares inverse: theory and experiments," *IEE Trans. Mechatronics* **2**, No. 3, 188–194 (Sept 1997).
11. M. Spong and M. Vidyasagar, *Robot Dynamics and Control* (J. Wiley & Son, New York, 1989).
12. J.M. Hollerbach and G. Sahar, "Wrist-partitioned inverse kinematic acceleration and manipulator dynamics," *Int. J. Robotics Research* **2**, No. 4, 61–76 (1983).
13. R.P. Paul and H. Zhang, "Computational efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation," *Int. J. Robotics Research* **5**, No. 2, 32–44 (1986).
14. G.H. Golub and C.F. Van Loan, *Matrix Computations* 2nd Ed. (Baltimore, MD: Johns Hopkins University Press, 1989).

15. L. Sciavicco and B. Siciliano, “Coordinate transformation: a solution algorithm for one class of robots,” *IEEe Trans. Syst. Man, Cyber.* **16**, 550–559 (1986).

16. Y.T. Tsai and D.E. Orin, “A strictly convergent real-time solution for inverse kinematics of robot manipulators,” *J. of Robotics Syst.*, **4**, 447–501 (1987).