A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy

PYUNG H. CHANG, MEMBER, IEEE

Abstract—**A closed-form solution formula for inverse kinematics of manipulators with redundancy is derived using the Lagrangian multiplier method. The proposed method is proved to provide the exact equilibrium state for the resolved-motion method. The repeatability problem in the resolved-motion method does not exist in the proposed method. The method is demonstrated to give more accurate trajectories than the resolved-motion method.**

I. Introduction

A KINEMATICALLY redundant robot manipulator is a manipulator that has more degrees of freedom than necessary to place the end effector at a desired location.[1] For example, if we want to place some point of a two-dimensional end effector at a specified location, we need three degrees of freedom. Thus a robot manipulator with more than three degrees of freedom is kinematically redundant in the two-dimensional space.

The major advantages of adding redundant degrees of freedom to a robot manipulator are as follows: 1) One achieves greater dexterity in maneuvering in a workspace with obstacles. 2) One can avoid singular configurations of the manipulators.

A. The Resolved-Motion Method Versus the Inverse Kinematic Method

Because of these significant advantages, an increasing amount of research has focused on the kinematically redundant manipulator, and the progress in this field has been rapid [8]-[10], [16], [17]. Much of this research has involved the resolution of motion using the pseudoinverse of the Jacobian matrix—also known as the Moore-Penrose generalized inverse matrix—to resolve the redundancy. This resolved-motion technique first determines the joint velocity using the pseudoinverse matrix and then incrementally determines the joint displacement; it thus transforms from workspace to joint space *via* joint velocity. In contrast to this direction of research, relatively little research (e.g., [4], [8]) has involved the inverse kinematics—the direct mapping from the workspace to the joint space—for kinematically redundant manipulators.

The advantages and disadvantages of the inverse kinematic method over the resolved-motion (RM) method are well-known in the case of nonredundant robot manipulators. The RM method—now using the inverse of the Jacobian matrix instead of the pseudoinverse matrix—is well defined for general manipulator kinematics, except for numerical problems near kinematic singularities. Furthermore, the joint velocities can be efficiently computed from the workspace velocities using the Jacobian matrix without requiring an explicit matrix inverse. The RM method, however, has some weak points:

- the method has intrinsic inaccuracy because of linear approximation characteristics of the Jacobian matrix; thus it accumulates errors which become larger as the velocity increases;
- the method does not directly give the joint values for a given location of the end effector.

The inverse kinematic method, on the other hand, has symbolic solutions which provide the joint variables with explicit functions in terms of the end effector location, only for *some* types of manipulator kinematics [14]. For general manipulator kinematics, there are only iterative solutions based on numerical methods which can be computationally expensive unless we have initial conditions sufficiently near to the solution. However, this method, be it symbolic or numerical, is attractive because of the direct mapping from the workspace to joint space, fixing most of the aforementioned problems of the RM method.

B. Inverse Kinematic Method in the Redundant Manipulator

The comparison between the two methods remains essentially unchanged in the case of redundant robot manipulators. Thus we have good reason to choose the inverse kinematic method.

As in the nonredundant case, no symbolic solution has been developed yet for the general redundant manipulator, for we cannot obtain symbolic solutions unless certain conditions are met by the manipulator structure. For example, in [4] and [8] only some of the joint variables were obtained symbolically. To obtain these solutions, the manipulator structure and the number of degrees of freedom were fixed, explicitly in [8] and implicitly in [4].

An additional difficult task for a redundant manipulator, regardless of whether or not it has symbolic solutions, is to rationally (or optimally) use the extra degrees of freedom to achieve objectives such as singularity avoidance or obstacle avoidance. In other words, the task is to resolve the redundancy while achieving some additional objectives.

Manuscript received November 26, 1985; revised December 12, 1986. This work was supported in part by the Defense Advanced Research Projects Agency under Office of Naval Research Contract N00014-80-C-0505, and in part by the Systems Development Foundation under a grant awarded to the MIT Artificial Intelligence Laboratory. This is primarily an *analytical* paper.

The author is with the Massachusetts Institute of Technology Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139.

IEEE Log Number 8715797.

[1] Location means position and orientation.

To our knowledge, the general close-form method to resolve the redundancy at the inverse kinematic level has not yet appeared. In this paper we propose a method or a general formula—derived from the Lagrangian multiplier method—to resolve the redundancy, thus fully specifying the kinematic equations. The resulting system of equations will be qualitatively compared with the two existing methods: the RM method [9], [10] and the extended Jacobian method [2]. From this comparison the relationships between the proposed method and each of the two methods will be examined. To evaluate the proposed method numerically, the system of equations—which requires numerical iterative solutions—is solved for a simulated task, and its solutions are compared with those of the RM method. In Section II, the proposed method will be derived. In Section III, the comparison and the relationships will be covered. The proposed method will be evaluated by simulations in Section IV, and then results will be discussed in Section V.

II. Derivation of the Proposed Equation

In this section we will derive extra equations which, together with the kinematic equations of the manipulator, can fully specify the under-determined problem. Then the characteristics and the extent of applications of the resulting equation will be discussed. In addition, the computational effort required to solve the inverse kinematic problem with the proposed method will be evaluated.

A. Derivation

The kinematic equation for the redundant manipulator is given as the following vector equation:

$$\boldsymbol{x}=\boldsymbol{f}(\vec{\theta}) \tag{1}$$

where $\boldsymbol{x}$ is an m-dimensional vector representing the location of the end effector with respect to the base coordinate system in the workspace, $\vec{\theta}$ is an n-dimensional vector representing joint variables, and $\boldsymbol{f}$ is a vector function consisting of m scalar functions with $m < n$. Equation (1) may be rewritten as

$$\begin{aligned} \boldsymbol{F}(\vec{\theta}) &= \boldsymbol{f}(\vec{\theta})-\boldsymbol{x} \\ &= \boldsymbol{0}. \end{aligned} \tag{2}$$

Let $H(\vec{\theta})$ be some criteria function with continuous first-order partial derivatives with respect to joint variables which quantitatively represents the desired performance—for instance, singularity avoidance or obstacle avoidance.

Let us define the Lagrangian function $L(\vec{\theta})$ as the following [7]:

$$L(\vec{\theta})=\vec{\lambda}^T\boldsymbol{F}(\vec{\theta})+H(\vec{\theta}) \tag{3}$$

where $\vec{\lambda}$ is an m-dimensional Lagrangian multiplier vector. At the stationary points of L,

$$\begin{aligned} \frac{\partial L}{\partial \vec{\theta}} &= \vec{\lambda}^T \frac{\partial \boldsymbol{F}}{\partial \vec{\theta}}+\frac{\partial H}{\partial \vec{\theta}} \\ &= \boldsymbol{0} \end{aligned} \tag{4}$$

where the $m \times n$ matrix $\partial \boldsymbol{F}/\partial\vec{\theta}$ is the Jacobian matrix $\boldsymbol{J}$ [16]. In (4), as $\boldsymbol{x}$ is expressed in the base coordinate system, so are the Lagrangian function and the Jacobian matrix. The second term in the right side of (4) is the transpose of the gradient vector $\boldsymbol{h}$ such that

$$\boldsymbol{h}=(h_1, h_2, \cdots, h_n)^T$$

$$h_i=\frac{\partial H}{\partial \theta_i}, \qquad i=1, 2, \cdots, n. \tag{5}$$

Thus (4) becomes the following:

$$\vec{\lambda}^T\boldsymbol{J}=-\boldsymbol{h}^T.$$

Transposing, we get

$$\boldsymbol{J}^T\vec{\lambda}=-\boldsymbol{h}$$

or

$$\begin{bmatrix}(J^1)^T\\(J^2)^T\\ \vdots \\(J^n)^T\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_m\end{bmatrix}=-\begin{bmatrix}h_1\\h_2\\ \vdots\\h_n\end{bmatrix} \tag{6}$$

where $(J^i)^T$ denotes the transpose of ith column vector of the Jacobian matrix. In (6), we have n linear equations with m unknowns, $\lambda_1, \lambda_2, \cdots, \lambda_m$. Selecting m linearly independent equations from (6), which without loss of generality may be chosen to be the first m equations, we have

$$\begin{bmatrix}(J^1)^T\\(J^2)^T\\ \vdots \\(J^m)^T\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_m\end{bmatrix}=-\begin{bmatrix}h_1\\h_2\\ \vdots\\h_m\end{bmatrix}. \tag{7}$$

Inverting, we obtain $\vec{\lambda}$ as

$$\begin{bmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_m\end{bmatrix}=-\begin{bmatrix}(J^1)^T\\(J^2)^T\\ \vdots \\(J^m)^T\end{bmatrix}^{-1}\begin{bmatrix}h_1\\h_2\\ \vdots\\h_m\end{bmatrix}.$$

Substituting this into the remaining $n-m$ equations in (6), we have

$$-\begin{bmatrix}(J^{m+1})^T\\(J^{m+2})^T\\ \vdots \\(J^n)^T\end{bmatrix}\begin{bmatrix}(J^1)^T\\(J^2)^T\\ \vdots \\(J^m)^T\end{bmatrix}^{-1}\begin{bmatrix}h_1\\h_2\\ \vdots\\h_m\end{bmatrix}=-\begin{bmatrix}h_{m+1}\\h_{m+2}\\ \vdots\\h_n\end{bmatrix}. \tag{8}$$

For brevity, let us denote

$$\boldsymbol{J}_m=\begin{bmatrix}(J^1)^T\\(J^2)^T\\ \vdots \\(J^m)^T\end{bmatrix} \qquad \boldsymbol{J}_{n-m}=\begin{bmatrix}(J^{m+1})^T\\(J^{m+2})^T\\ \vdots \\(J^n)^T\end{bmatrix}$$

$$\boldsymbol{h}_m=\begin{bmatrix}h_1\\h_2\\ \vdots\\h_m\end{bmatrix} \qquad \boldsymbol{h}_{n-m}=\begin{bmatrix}h_{m+1}\\h_{m+2}\\ \vdots\\h_n\end{bmatrix}.$$

Adding $\boldsymbol{h}_{n-m}$ and multiplying both sides of (8) by -1, we have

$$\boldsymbol{J}_{n-m}\boldsymbol{J}_m^{-1}\boldsymbol{h}_m - \boldsymbol{h}_{n-m} = \boldsymbol{0}$$

which may be alternatively expressed as

$$[\boldsymbol{J}_{n-m}\boldsymbol{J}_m^{-1} : -\boldsymbol{I}_{n-m}] \begin{bmatrix} \boldsymbol{h}_m \\ \boldsymbol{h}_{n-m} \end{bmatrix} = \boldsymbol{0}$$

where $\boldsymbol{I}_{n-m}$ is an identity matrix of rank $(n - m)$. If we denote

$$\boldsymbol{Z} = [\boldsymbol{J}_{n-m}\boldsymbol{J}_m^{-1} : -\boldsymbol{I}_{n-m}], \tag{9}$$

then (8) becomes

$$\boldsymbol{Z}\boldsymbol{h} = \boldsymbol{0} \tag{10}$$

where $\boldsymbol{Z}$ and $\boldsymbol{h}$ are defined as before. If we combine the kinematic equation (1) with (10) as a system of equations, we get

$$\left\{ \begin{matrix} \boldsymbol{x} = \boldsymbol{f}(\vec{\theta}) \\ \boldsymbol{Z}\boldsymbol{h} = \boldsymbol{0} \end{matrix} \right\}. \tag{11}$$

Since $\boldsymbol{Z}$ is an $(n - m) \times n$ matrix and $\boldsymbol{h}$ is an n-dimensional vector, (10) consists of $(n - m)$ scalar equations with n unknowns $\vec{\theta}$. On the other hand, the kinematic equation (1) has m scalar equations. Therefore, (11) has n independent nonlinear equations which now fully specify the n unknowns. Note that (11) has to be solved numerically.

B. Characteristics

The additional set of equations (10) resolves the redundancy—at the inverse kinematic level—in such a way that the criteria function $H(\vec{\theta})$ may be minimized. Equation (10) may be viewed as the direct counterpart of the homogeneous solution term of the resolved motion method which uses the null space to resolve the redundancy, while (1) corresponds to the special solution term. In fact, it will be shown in Appendix II that $\boldsymbol{Z}$ is composed of $n - m$ linearly independent vectors which span the null space of $\boldsymbol{J}$.

In (10), note that the matrix $\boldsymbol{Z}$ depends on the intrinsic kinematic property of a manipulator, while $\boldsymbol{h}$ depends on an arbitrarily imposed property. Therefore, if a symbolic form of $\boldsymbol{Z}$ is available—which is not very difficult once the Jacobian matrix can be expressed in a symbolic form—we have only to replace $\boldsymbol{h}$, depending on the desired performance, without having to derive the equation all over again.

In (10), *any* criteria function may be used as long as the function can be reduced to an expression in terms of joint variables only. For instance, consider the following criteria function $H(\vec{\theta})$ for the obstacle avoidance problem [12][2]:

$$H = k_O \sum_{i=1}^{l} 1/\{C_O(\boldsymbol{x}_i) - 1\} + k_J \sum_{j=1}^{n} 1/(\theta_{j\,\max}^2 - \theta_j^2) \tag{12}$$

where k_O and k_J are scaling factors; $\boldsymbol{x}_i = (x_{1i}, x_{2i}, x_{3i})^T$ is the position of the ith point among l points on the manipulator; $\theta_{j\,\max}$ is the limit of jth joint; and the model of the obstacle in the workspace $C_O(\boldsymbol{x}_i)$ is defined as

$$C_O(\boldsymbol{x}_i) = \sum_{k=1}^{3} \left(\frac{x_{ki} - x_{kc}}{r_k} \right)^s \tag{13}$$

where x_{kc}, r_k, and s are the center coordinate, radii, and roundness exponent of the obstacle object, respectively. Although H obviously contains other variables $\boldsymbol{x}_i$ in addition to joint variables, we can reduce H to a function of $\vec{\theta}$ only by transforming $\boldsymbol{x}_i$ into $f_i(\vec{\theta})$ using (1) and thus can apply (10) to it.

C. Computational Consideration

Since the computational efficiency is not the primary concern of this paper, only a brief analysis will be made with regard to the computational effort required to solve (11). One can find more detailed discussions in the references cited in this subsection. For the sake of generality we consider the general manipulator without any special geometry, which would enable much more efficient computations.

The computational effort will be measured in terms of arithmetic operations such as addition, subtraction, multiplication, and division. Furthermore, we assume that the four arithmetic operations require approximately the same effort of computation time, and the effort required to evaluate trigonometric functions is negligible as compared to the total computational effort.

Since (11) is to be solved iteratively by a numerical method, the computational effort at each iteration step N_1 is evaluated as follows:

$$N_1 = N_{FK} + N_J + N_{Zh} + N_{\mathrm{nl}} \tag{14}$$

where N_{FK} denotes the computational effort required for the forward kinematics (1), N_J the effort to obtain the Jacobian matrix N_{Zh} for $\boldsymbol{Zh}$ in (10), and N_{nl} the effort needed by the numerical method for solving the system of nonlinear equations. For the general manipulator with revolute joints—no significant difference is expected when a few prismatic joints are included—N_{FK} is given as [1]

$$N_{FK} = N_R + N_T$$
$$N_R = 36(n-1) + 4 \qquad N_T = 16(n-1) + 2 \tag{15}$$

where N_R and N_T indicate the computational efforts required to compute orientation and position of the end effector.[3]

N_J for the general manipulator, with Waldron's scheme, is given as [13]

$$N_J = 45n - 93. \tag{16}$$

Since we have not at this point chosen a specific criteria function from a variety of choices, let us simply assume that we have derived $\boldsymbol{h}$ with relatively a small amount of

[2] The problem was originally treated in the dynamic context by using a potential function.

[3] We derived these general formulas on the basis of the computation algorithm in [1]. Slightly different formulas can result, depending on different computation details.

computation as compared to the other parts of computation. If we assume at the same time that a symbolic form of $\boldsymbol{Z}$ is not available, $\boldsymbol{Zh}$ in (10) is more efficiently computed by first solving for $\vec{\lambda}$ in (7) with Gaussian elimination and then substituting it as in (8). Thus N_{Zh} is given as follows:

$$N_{Zh} = N_G + N_s$$

$$N_G = \frac{2m^3}{3} + \frac{3m^2}{2} - \frac{7m}{6} \qquad N_s = (n-m)(2m-1) \quad (17)$$

where N_G is the effort for Gaussian elimination [12] and N_s is that for the substitution.

If we use, for example, MINPACK-1, one of the well-known software packages, to solve the system of nonlinear equations, the computational effort N_{nl}, becomes [11]

$$N_{\text{nl}} = 11.5n^2.$$

Consequently, the total effort of computation for the general manipulator with kinematic redundancy per iteration for the proposed method becomes

$$N_1 = \frac{2}{3}m^3 + 11.5n^2 + 2nm - \frac{m^2}{2} - \frac{m}{6} + 96n - 139. \quad (18)$$

As an example to show the total computational effort as well as the relative significance of each part in it in a realistic situation, let us evaluate the effort when $n = 7$ and $m = 6$. According to aforementioned estimations, we have $N_1 = 1306$ in which $N_{\text{FK}} = 318$, $N_J = 222$, $N_{Zh} = 202$, and $N_{\text{nl}} = 564$.

III. Comparison with Other Methods

In this section, the proposed method corresponding to (10) will be compared with two existing methods: the RM method and the extended Jacobian method. The relationships will be investigated on the basis of these comparisons.

A. Relationship with the RM Method

A general solution for the equation $\dot{x} = J\vec{\theta}$, when $n > m$, was given as [5]

$$\vec{\theta} = J^+\dot{x} + \alpha(I - J^+J)h$$
$$J^+ = J^T(JJ^T)^{-1} \quad (19)$$

where J^+ and $(I - J^+J)$ are the pseudoinverse matrix and the null space of J, with α a gain constant and h an arbitrary vector. Equation (19) gives a way to resolve the redundancy at the velocity level. We may call the first term in the right-hand side a special solution and the second term a homogeneous soultion.

Liégeois [10] developed a formulation of resolution of redundancy such that a scalar criteria function may be minimized by setting to the vector h the gradient vector of the criteria function h as in (5). He expresses (19) in terms of the infinitesimal displacement as

$$d\vec{\theta} = J^+dx + \bar{\alpha}(I - J^+J)h$$
$$\bar{\alpha} = \alpha dt. \quad (20)$$

In (20) (or (19)), at time t the special solution first moves the end effector to a location $x(t)$. Then the homogeneous solution, on the other hand, forces joints to have self-motion to achieve an equilibrium (or optimum) $\vec{\theta}^*$, where H has a local minimum for $x(t)$. In practice, however, it takes a certain amount of time δt for the joint variables to reach $\vec{\theta}^*$ when the end effector has already moved to a new location $x(t + \delta t)$, thus requiring a new $\vec{\theta}^*$. Therefore, the joint variables never reach the optimal configuration but continuously trail behind with a slight difference in the direction of the end-effector displacement dx. In other words, if the end effector would stay at a location sufficiently long, the joints could eventually achieve the optimal configuration corresponding to that location.

Because of these characteristics the RM method in (20) (or(19)) has an undesirable property: it does not preserve the *repeatability* of joint values for repeated end-effector motions. By the *repeatability* of joint values, we mean the ability to give the same joint values for a given end-effector location regardless of the path to the location. More specifically, the repeatability problem is caused by the following two factors.

1) Because of the *directionality* of dx in the special solution, the joint variables have different values depending on the *direction* of the repeated path—for instance, a cyclic path—in the workspace. Note, however, that the repeatability can be preserved when tracing only one direction of the cyclic path, even in the presence of this factor [2].

2) Because of the *irreversibility* of the homogeneous solution part, they never return to the initial configuration once the joint variables reach a steady-state trajectory near to the optimal trajectory $\vec{\theta}(t)^*$. This situation can happen at the initial transient period when initially guessed joint values are far from the optimum, for the optimal joint values cannot be known in advance.

In Appendix II we prove that (10) is the necessary and sufficient condition to be satisfied when (20) has converged to its equilibrium states, provided that $dx = \mathbf{0}$. In other words, (11) gives the exact equilibrium state, the optimal joint configuration, at which (20) will eventually arrive by self-motion for a fixed end-effector location. Thus we may regard (20) in the RM method as an approximated equation linearized at states that are exactly determined by (11). The speed of convergence is determined by the value of $\bar{\alpha}$: the larger the value, the faster the convergence. The value of $\bar{\alpha}$ has an upper limit above which the equation becomes numerically unstable, not to mention that the manipulator cannot respond because of the torque limitation.

Computational Consideration: In the light of the relationship between the two methods, we may regard the RM method (20) as a one-step search which, if the homogeneous part is iteratively applied with a numerical integration scheme, leads to the exact and repeatable inverse kinematic solution—thus an alternative numerical method to solve (11). Then, naturally, the following two questions would arise: *how many iterations* and *how many arithmetic operations per iteration* does each method require to achieve the same degree of accuracy (and repeatability) from the same initial condition? The former will be answered through a computation example for a special case in the following sections; the latter question will be briefly examined here by first evaluating the number of operations per iteration for the RM method and then comparing it with that for the proposed method already obtained.

As in Section II, it is assumed that the numerical value of $\boldsymbol{h}$ is provided at each iteration step with a relatively small amount of computation. The computational effort N_1, required for the proposed method at each iteration step, was obtained in (18) as

$$N_1=\frac{2}{3}m^3+11.5n^2+2nm-\frac{m^2}{2}-\frac{m}{6}+96n-133.$$

On the other hand, the total computational effort N_2 required for the RM method at each iteration step or integration step may be expressed as

$$N_2=N_{ev}(N_J+N_a)+N_b \tag{21}$$

where N_J is the effort required to compute the Jacobian matrix, N_a is the effort required to evaluate the right side of (20) once the Jacobian matrix is given, N_{ev} is the number of evaluations of (20), and N_b is the effort for numerical integrations. For the general manipulator, N_J is given in (16) as

$$N_J=45n-93 \tag{22}$$

and N_a is evaluated in [12] as

$$N_a=\frac{2}{3}m^3+nm^2+5nm+m^2-\frac{2}{3}m-n. \tag{23}$$

N_{ev}, together with N_b, depends on the specific numerical integration method to be used. For example, the fourth-order Runge–Kutta method requires that $N_{ev}=4$ and $N_b=13n$, while most predictor–corrector methods require that $N_{ev}=2$ and $N_b=23n$ [6]. Considering that N_2 heavily depends on the sum of N_a and N_J, the latter method is much more efficient.

Therefore, N_2 with the predictor–corrector method is given as

$$N_2=\frac{4}{3}m+2nm^2+10nm+2m^2+111n-\frac{4}{3}m-186. \tag{24}$$

On the other hand, the Runge–Kutta method requires

$$N_2=\frac{8}{3}m^3+4nm^2+20nm+4m^2+189n-\frac{8}{3}m-372. \tag{25}$$

When $n=7$ and $m=6$, $N_2=1867$ for the predictor–corrector method and $N_2=3503$ for Runge–Kutta method, while $N_1=1306$. In this comparison, we see that if the predictor–corrector methods are used for the numerical integration, the RM method requires about 43-percent more computational effort than the proposed method per iteration when $n=7$ and $m=6$.

B. Relationship with Extended Jacobian Method

An alternative method to resolve the redundancy, also at the inverse kinematic level, is presented in [2], [3]. This method derives the additional constraints by using the orthogonality characteristics between the gradient vector of the criteria function and the null-space matrix of $\boldsymbol{J}$ at the optimum as

$$(\boldsymbol{I}-\boldsymbol{J}^+\boldsymbol{J})\boldsymbol{h}=\boldsymbol{0} \tag{26}$$

where $\boldsymbol{J}^+$ is the pseudoinverse of $\boldsymbol{J}$ as defined before. From the resulting fully specified system of equations, one derives the new Jacobian matrix, called the extended Jacobian, by partially differentiating with respect to the joint variables in just the same way as in the nonredundant case.

Among the n scalar equations in (26), only $n-m$ equations are independent constraints to be determined, for the rank of the null-space matrix is $n-m$. The detailed procedures of determining the $n-m$ constraints and their concrete functional forms are not yet known, except for the case $n=m+1$. When $n=m+1$, that is, for the manipulator with just one redundant degree of freedom, a constraint is derived through a series of procedures as the following [2]:

$$\begin{aligned}G(\vec{\theta})&=\boldsymbol{n}_J\boldsymbol{h}\\&=0\end{aligned} \tag{27}$$

where

$$\begin{gathered}\boldsymbol{n}_J=(\Delta_1,\Delta_2,\cdots,\Delta_n)^T\\ \Delta_i=(-1)^{i+1}\det(J^1,J^2,\cdots,J^{i-1},J^{i+1},\cdots,J^n)\end{gathered} \tag{28}$$

where det $(\cdot)$ is the determinant with J^k the kth column vector of the Jacobian matrix. Considering that $\boldsymbol{Z}$ in (10) is a null-space matrix of rank $n-m$, as shown in Appendix II, and that (10) reduces to (27) in the case that $n=m+1$, as proved in Appendix I, we may regard (10) as a concrete expression of $n-m$ independent constraints.

IV. Simulation

In this section, we select a kinematically redundant manipulator and apply (10). The resulting system of equations is solved numerically for $\boldsymbol{x}$, the end effector location, which makes a cyclic path. In parallel to using (11), the RM method is applied to the same manipulator with the same tip motion. The points we try to examine or verify through the simulation are as follows:

1) whether the resulting system in (11) gives kinematically correct joint variables for a given $\boldsymbol{x}$, achieving the performance represented by the criteria function we select;
2) how the present method compares to the RM method in terms of accuracy, repeatability, and computational efficiency;
3) whether the present method gives, in fact, the same equilibrium states as the RM method will eventually reach.

We use the same manipulator presented in [17] for the sake of comparison with data obtained in that paper: a three-degrees-of-freedom manipulator with the end effector moving in the (x, y) plane, thus kinematically redundant. The schematic diagram with necessary parameters is in Fig. 1.

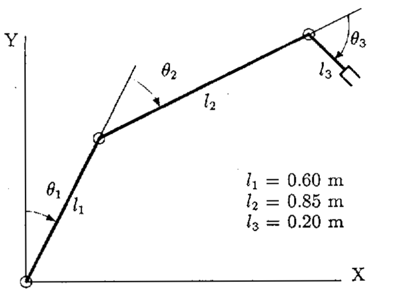


Fig. 1. Schematic diagram of redundant manipulator.

The task is to trace a square path—thus a cyclic path—while avoiding singularities. A good criteria function for this objective may be the manipulability [17], which is given as

$$H = \det(JJ^T). \tag{29}$$

The kinematic equation is given as

$$\begin{aligned} x &= l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ y &= l_1 c_1 + l_2 c_{12} + l_3 c_{123} \end{aligned} \tag{30}$$

where l_1, l_2, and l_3 represent the length of each link, while the variables with subscripts are defined as

$$\begin{aligned} s_i &= \sin(\theta_i), \qquad s_{ij\cdots k} = \sin(\theta_i + \theta_j + \cdots + \theta_k) \\ c_i &= \cos(\theta_i), \qquad c_{ij\cdots k} = \cos(\theta_i + \theta_j + \cdots + \theta_k), \\ & \qquad\qquad i, j, \cdots, k = 1, 2, 3. \end{aligned}$$

Then the Jacobian matrix is obtained as

$$J = \begin{pmatrix} vc_{123} + uc_{12} + c_1 & vc_{123} + uc_{12} & vc_{123} \\ -vs_{123} - us_{12} - s_1 & -vs_{123} - us_{12} & -vs_{123} \end{pmatrix} \tag{31}$$

where

$$u = \frac{l_2}{l_1} \qquad v = \frac{l_3}{l_1}.$$

Equation (29), after algebraic manipulation, reduces to

$$H = 2s_3^2 u^2 v^2 + 2s_{23} s_3 uv^2 + 2s_{23}^2 v^2 + 2s_2 s_{23} uv + s_2^2 u^2.$$

By partially differentiating, we obtain the gradient vector $\boldsymbol{h} = (h_1, h_2, h_3)^T$, with

$$\begin{aligned} h_1 &= 0 \\ h_2 &= 2(c_{23} s_3 u + s_{2233}) v^2 + 2s_{223} uv + s_{22} u^2 \\ h_3 &= 2(s_{33} u^2 + s_{233} u + s_{2233}) v^2 + 2c_{23} s_2 uv. \end{aligned}$$

Meanwhile, (9) is simplified to

$$Z = [s_3 uv \quad -s_3 uv - s_{23} v \quad s_{23} v + s_2 u].$$

By applying (10), we get

$$\begin{aligned} & 2u^2 v^3 (s_{23} s_{33} - c_{23} s_3^2) + uv^3 (2s_{23} s_{233} - 3s_{2233} s_3) \\ & + 2u^3 v^2 s_2 s_{33} + u^2 v^2 (c_{33} - c_{22}) + 2uv^2 (s_2 s_{2233} - c_2 s_{23}^2) \\ & - u^3 v s_{22} s_3 - 2u^2 v s_2 s_3 = 0 \end{aligned} \tag{32}$$

where the same definition as in (30) is used for the subscripts.

The system of equations (30) and (32) now fully specify the originally under-determined system of equations (30) while maximizing the criteria H in (29). They system of equations may be solved either purely numerically or by symbolically reducing variables—in this example θ_1—and then by using numerical methods. Incidentally, this example suggests that it is possible to reduce variables, thus reducing the order of the system of nonlinear equations, after resolving the redundancy first with all the joint variables. As shown in Fig. 3, the x–y coordinates of the four vertices of the command path in the workspace are successively given counterclockwise from the upper left vertex as follows:

$$(446.00, 91.514) \quad (446.00, -8.486)$$
$$(546.00, -8.486) \quad (546.00, 91.514)$$

where the units are mm.

The system of equations were numerically solved for joint values with regard to consecutive equidistant points on the command path—in this example, 100 segments each side—in the counterclockwise direction. As mentioned before, the MINPACK-1 subroutine was used to solve the system of nonlinear equations, which is primarily based on an improved version of Powell's hybrid method, a method that combines the Newton–Raphson method with the steepest descent method [11]. The subroutine allows only one set of local solutions among multiple sets of solutions.

Meanwhile, the RM method in (19) is also applied to this example for the same path, which the end effector is to track with a constant speed of 10 mm/s. Equation (19), a system of differential equations, was solved with the fourth-order Runge–Kutta integration method together with the LINPACK subroutines, with the integration time step size $\Delta t = 0.01$ s. We can also use the predictor–corrector method for better efficiency at the cost of a non-self-starting disadvantage and a more complex program.

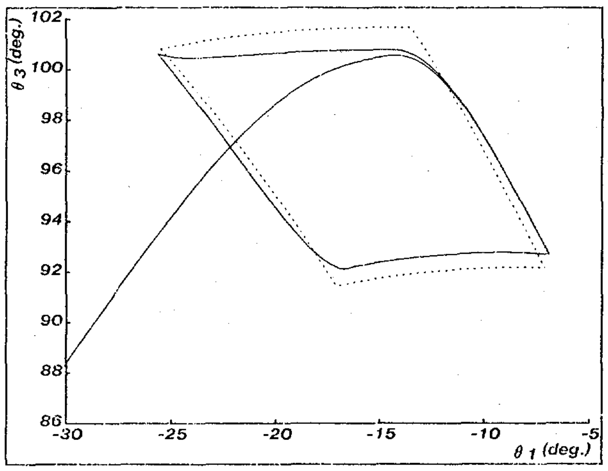
The simulation for the RM method was made with initial joint-angle values of (−40.5006, 141.6408, 78.4169) in degrees, which are far from equilibrium states, that were deliberately selected to examine repeatability. The same initial value was also used as the initial guess for the foregoing nonlinear equations for fair comparison of the two methods.

V. Results and Discussions

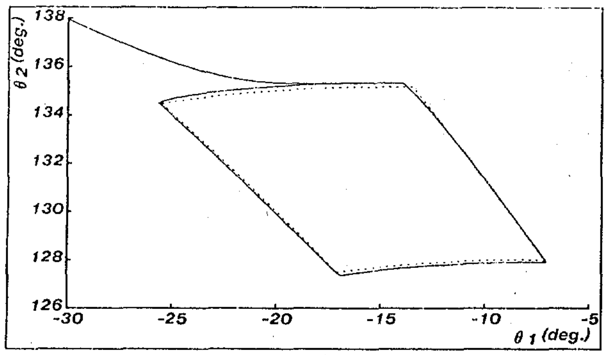
A. Results

The numerical results of simulations are plotted and listed in Figs. 2 and 3 and Tables I–III.

- Fig. 2 is the plot of joint variables solved with the two methods. Note that the three-dimensional trajectory of joint variables is represented with two two-dimensional plots: θ_1 versus θ_2 and θ_1 versus θ_3.
- On the other hand, Fig. 3 shows actual trajectories with the two methods of the end effector in the workspace, as compared to the command path. The actual trajectory was determined by forward kinematics with joint values obtained by each method.



(a)



(b)

Fig. 2. Joint trajectories obtained with two methods. —— Resolved motion method. ···· Proposed method. (a) θ_1 versus θ_3. (b) θ_1 versus θ_2.

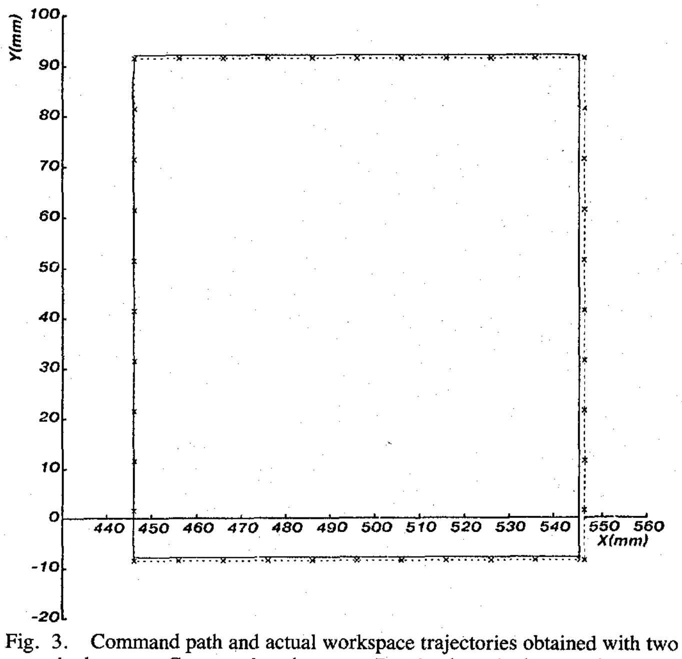


Fig. 3. Command path and actual workspace trajectories obtained with two methods. × × Command path. —— Resolved method. ···· Proposed method.

- Table I enables one to compare numerically the accuracy of the tip location and observe the irreversibility factor due to the homogeneous solution term of (20) at the transient period. For our purposes we have selected from the data in Figs. 2 and 3 five sets of data which correspond to the consecutive vertices of the square path—counterclockwise from the upper left one, when the tip begins to trace at the very first cycle.
- Table II shows the comparison of the accuracy and repeatability when tracing opposite directions in the steady state. To obtain the data the tip was made to reciprocate a straight line segment which was set to the left vertical side of the square path after the first two cycles of the square path when the transient effect appeared negligible.
- Table III verifies the relationship proved in Appendix II: the solution with the RM method becomes equal to that with the proposed method after sufficient time, provided that $dx = \mathbf{0}$. The four vertices were selected for the comparison as in Table I. To obtain the data with the RM method, the tip was commanded to stop at each vertex, where it made the self-motion through iterations of the homogeneous term of (20) until the joints converged to an equilibrium configuration. At each vertex about 25 iterations were needed to converge to the joint values obtained with the proposed method, within the accuracy of 10^{-4} in degrees with $\bar{\alpha} = 10$, which provides about the upper limit speed of convergence without causing numerical instability.

B. Discussion

From the results of the simulations we may evaluate the proposed method in terms of accuracy and repeatability by comparing it with the RM method. We can also derive some useful ideas from the relationship between the two methods.

1) Accuracy: As shown in Table I and Fig. 3, the proposed method gives joint variables which exactly correspond to the commanded x and y while maximizing the criteria function to avoid singularities. Clearly, we see that the accuracy in the workspace achieved with the proposed method is better than that with the RM method. Therefore, the proposed method provides useful means for accurate position control of the end effector when the manipulator is kinematically redundant.

2) Repeatability: Table II and Fig. 2 show that the repeatability is *not* preserved with the RM method because of the two factors mentioned in Section III: the initial joint variables, which are far from optimal joint values (Fig. 2), and the dependence on the direction, clockwise or counterclockwise, of the path to be traced (Table II). The lack of repeatability can be a considerable drawback in robot manipulators which perform cyclic tasks because as the end effector traces the cyclic path, joint variables evolve into states which cannot be predicted in advance.

On the other hand, it is obvious that the proposed method preserves the repeatability regardless of direction. In other words, the method provides a *fixed transformation* from workspace to joint space. The property of fixed transformation is useful not only for the prediction of joint variables, but also

TABLE I
THE COMPARISON OF ACCURACY AND REPEATABILITY OBTAINED WITH THE TWO METHODS AT THE FIRST CYCLE, WHEN TRANSIENT EFFECT IS VISIBLE

	X (mm)	Y (mm)	θ_1 (°)	θ_2 (°)	θ_3 (°)
Command path	446.00	91.514			
Proposed method	446.00	91.514	−25.5116	134.4894	100.8165
RM method	446.00	91.514	−40.5006	141.6408	78.4169
Command path	446.00	−8.4866			
Proposed method	446.00	−8.4866	−13.4927	135.1801	101.6627
RM method	445.74	−6.5824	−14.0445	135.3160	101.2448
Command path	546.00	−8.4866			
Proposed method	546.00	−8.4863	−7.1232	128.0020	92.1837
RM method	545.52	−6.8216	−7.1924	127.9635	92.4919
Command path	546.00	91.514			
Proposed method	546.00	91.514	−17.0753	127.4846	91.4484
RM method	545.73	92.947	−17.0519	127.3890	91.7938
Command path	446.00	91.514			
Proposed method	446.00	91.514	−25.5116	134.4894	100.8165
RM method	445.97	93.167	−25.7427	134.4867	100.7127

TABLE II
THE COMPARISON OF ACCURACY AND REPEATABILITY OBTAINED WITH THE TWO METHODS AT THE STEADY STATE, WHEN THE TIP RECIPROCATES A VERTICAL LINE SEGMENT

	X (mm)	Y (mm)	θ_1 (°)	θ_2 (°)	θ_3 (°)
Command path	446.00	91.514			
Proposed method	446.00	91.514	−25.5116	134.4894	100.8165
RM method	445.89	93.140	−25.7472	134.4929	100.7206
Command path	446.00	−8.4866			
Proposed method	446.00	−8.4866	−13.4927	135.1801	101.6627
RM method	445.63	−6.6244	−14.0476	135.3249	101.2557
Command path	446.00	91.514			
Proposed method	446.00	91.514	−25.5116	134.4894	100.8165
RM method	445.60	92.912	−25.4178	134.3821	101.2799
Command path	446.00	−8.4866			
Proposed method	446.00	−8.4866	−13.4927	135.1801	101.6627
RM method	445.57	−6.6184	−14.0526	135.3293	101.2610

TABLE III
THE COMPARISON OF SOLUTIONS: PROPOSED METHOD VERSUS RM METHOD; FOR RM METHOD, SELF-MOTION WAS ITERATIVELY MADE BY SETTING $dx = 0$

	X (mm)	Y (mm)	θ_1 (°)	θ_2 (°)	θ_3 (°)
Command path	446.00	91.514			
Proposed method	446.00	91.514	−25.5116	134.4894	100.8165
RM method	446.00	91.514	−25.5115	134.4894	100.8164
Command path	446.00	−8.4866			
Proposed method	446.00	−8.4866	−13.4927	135.1801	101.6627
RM method	446.00	−8.4868	−13.4927	135.1801	101.6626
Command path	546.00	−8.4866			
Proposed method	546.00	−8.4863	−7.1232	128.0020	92.1837
RM method	546.00	−8.4863	−7.1232	128.0020	92.1837
Command path	546.00	91.514			
Proposed method	546.00	91.514	−17.0753	127.4846	91.4484
RM method	546.00	91.514	−17.0752	127.4846	91.4484

for the precomputation of position-dependent terms, such as the Jacobian matrix and the inertia matrix [15].

3) Computational Effort: In the simulation example, since we have the Jacobian matrix in a symbolic form, N_J is only 12 instead of 42 from (16). In addition, since rotational displacement does not exist, the forward kinematics (30) requires that $N_{\mathrm{FK}} = 8$. Thus N_1 and N_2, when $n = 3$ and $m = 2$, become $N_1 = 136$ and $N_2 = 239$, respectively. If the predictor–corrector method were used instead of the Runge–Kutta method, N_2 would be 175.

The number of iterations to achieve the accuracy of 10^{-4} in degrees by the proposed method with MINPACK-1 is from 10 to 20, depending on the length of the line segment $\Delta\boldsymbol{x}$ on the command path up to the length of one side (100 mm). On the other hand, the RM method requires about 25 iterations[4] or integration steps for the self-motion alone with a fixed tip location to achieve that accuracy.

In addition, the number of integration steps N_{tip} needed for the displacement of the tip $\Delta\boldsymbol{x}$ is estimated as

$$N_{\mathrm{tip}} = \frac{\|\Delta\boldsymbol{x}\|}{\|\dot{\boldsymbol{x}}\|\Delta t}$$

where $\dot{\boldsymbol{x}}$ is the speed of the tip motion, Δt is the integration time step, and $\|\cdot\|$ represents the Euclidean norm of vectors. Thus N_{tip} is directly proportional to the distance of tip motion and the inverses of speed and time step. In the simulation, where $\|\dot{\boldsymbol{x}}\| = 100$ mm/s and $\Delta t = 0.01$ s, N_{tip} is 100 for a 100-mm displacement. We can reduce N_{tip} by increasing the speed of the tip motion and time step at the cost of accuracy and numerical stability.

Regardless of the value of N_{tip}, however, the number of iterations for self-motion alone in the RM method already exceeds that of the proposed method. Furthermore, if we consider that the proposed method requires about ten iterations under similar conditions, the RM method requires about 2.5 times more iterations and over three times the total computational effort, even if the predictor–corrector method is used, than the proposed method in this special example. In other words, to obtain the same data as in Table III, the RM method requires over three times more computational effort than that of the proposed method.

Although it is not immediately clear how the two methods will compare for manipulators with more degrees of freedom, we expect the RM method would require a larger computational effort than the proposed method, considering the following factors:

- the former requires about 43-percent more computation per iteration than the latter;
- owing to an elaborate step-size control scheme available with the latter [11], the number of iterations required with latter is expected to be smaller—at most, not 40 percent larger—than that with the former.

4) Relationship Between the Two Methods: The result in Table III shows a nearly perfect agreement of both solutions, verifying that (11) in the proposed method is the equilibrium equation at which (19) will finally arrive. Because of the relationship between the two methods, we may use them *interchangeably* as follows.

- The exact equilibrium state can be determined either with the proposed method or with the RM method by setting $d\boldsymbol{x} = \boldsymbol{0}$. The latter, however, would require more computations than the former, as discussed before.
- The incremental displacement $d\vec{\theta}$, which (19) in the RM method easily provides, can also be obtained by first differentiating (11) to get the extended Jacobian matrix [2] and then by inverting it.

We can also make use of the relationship *complementarily*:

- the proposed method could be more effective if the initial guess for the method is provided by the RM method;
- the proposed method can supply a set of joint values for a given tip location from which the RM method, a rate equation, successively obtains the joint rates.

APPENDIX I

THE DERIVATION OF THE EXTENDED JACOBIAN METHOD

In this Appendix we will prove that (10) reduces to the extended Jacobian method when $n = m + 1$. The additional equation to resolve the redundancy in the extended Jacobian method is given [2] in (27) and (28) as

$$\begin{aligned} G(\vec{\theta}) &= \boldsymbol{n}_J \boldsymbol{h} \\ &= 0 \end{aligned} \tag{33}$$

where

$$\boldsymbol{n}_J = (\Delta_1, \Delta_2, \cdots, \Delta_n)$$
$$\Delta_i = (-1)^{i+1} \det(J^1, J^2, \cdots, J^{i-1}, J^{i+1}, \cdots, J^n)$$

and where J^k is the kth column vector of the Jacobian matrix $\boldsymbol{J}$ derived from (1). When $n = m + 1$, our result (10) is specified as

$$\begin{aligned} G_2(\vec{\theta}) &= [(J^n)^T \boldsymbol{J}_m^{-1}, -1]\boldsymbol{h} \\ &= 0 \end{aligned} \tag{34}$$

where (J^n) is the transpose of the nth column vector of the Jacobian matrix $\boldsymbol{J}$. $\boldsymbol{J}_m^{-1}$ can be derived as

$$\boldsymbol{J}_m^{-1} = \frac{1}{D_m}\boldsymbol{A}_m \tag{35}$$

where $\boldsymbol{A}_m$ is the adjoint matrix of $\boldsymbol{J}_m$ and D_m is the determinant of $\boldsymbol{J}_m$. Thus $\boldsymbol{A}_m$ is expressed as

$$\boldsymbol{A}_m = \begin{pmatrix} Cof_{11} & Cof_{21} & \cdots & Cof_{m1} \\ Cof_{12} & Cof_{22} & \cdots & Cof_{m2} \\ \vdots & \vdots & \cdots & \vdots \\ Cof_{1m} & Cof_{2m} & \cdots & Cof_{mm} \end{pmatrix} \tag{36}$$

where Cof_{ij} is the cofactor of j_{ij} of the Jacobian matrix $\boldsymbol{J}$. From (34)–(36) we get

$$(J^n)^T \boldsymbol{J}_m^{-1} = \frac{1}{D_m}[(J^n)^T A_m^1, (J^n)^T A_m^2, \cdots, (J^n)^T A_m^m] \tag{37}$$

[4] This number is based on the Runge–Kutta method. An efficient step-size control with the predictor–corrector method may reduce the number of iterations.

where A_m^i is the ith column vector of A_m. Since

$$(J^n)^T A_m^1 = j_{1n} Cof_{11} + j_{2n} Cof_{12} + \cdots + j_{mn} Cof_{1m} = -\Delta_1,$$

likewise, we get

$$(J^n)^T A_m^i = -\Delta_i. \tag{38}$$

From the definition of Δ_n, we get

$$\Delta_n = \det (J_m^T) = \det (J_m) = D_m. \tag{39}$$

Therefore,

$$((J^n)^T J_m^{-1}, -1) = (-\frac{\Delta_1}{\Delta_n}, -\frac{\Delta_2}{\Delta_n}, \cdots, -1). \tag{40}$$

Since J_m is nonsingular—and thus Δ_n is nonzero—we can multiply by it on both sides of (34), resulting in (27). Thus we have proved that (10) is a general expression which yields the additional equation in the extended Jacobian method.

Appendix II

The Relationship Between the Proposed Method and the RM Method

The matrix in (9) is, again,

$$Z = [J_{n-m} J_m^{-1} : -I_{n-m}].$$

Since the rank of any matrix is the dimension of its largest nonsingular submatrix, which is I_{n-m} for Z, the rank of Z (and Z^T, too) is $n - m$. In addition, since

$$JZ^T = [J_m^T : J_{n-m}^T] \begin{bmatrix} (J_m^{-1})^T (J_{n-m}^{-1})^T \\ -I_{n-m} \end{bmatrix} = \mathbf{0}, \tag{41}$$

column vectors of Z^T are a set of basis vectors which are orthogonal to J. Thus row vectors of J, together with a column vectors of Z^T, constitute the basis of the n-dimensional vector space.

Accordingly, any n-dimensional vector h can be represented as

$$h = J^T h_1 + Z^T h_2 \tag{42}$$

where h_1 and h_2 are arbitrary vectors of m and $n - m$ dimensions, respectively. Premultiplying (42) by J, we have

$$Jh = JJ^T h_1$$

Thus,

$$h_1 = (JJ^T)^{-1} Jh. \tag{43}$$

Similarly, if we multiply (42) with Z and solve for h_2, we get

$$h_2 = (ZZ^T)^{-1} Zh. \tag{44}$$

From (42) to (44), we obtain the following relationship:

$$(I - J^T (JJ^T)^{-1} J) h = Z^T (ZZ^T)^{-1} Zh$$

or from (19),

$$(I - J^+ J) h = Z^T (ZZ^T)^{-1} Zh. \tag{45}$$

For a constant location of the end effector (no tip motion), when $dx = \mathbf{0}$, (20) with (45) becomes

$$d\vec{\theta} = Z^T (ZZ^T)^{-1} Zh. \tag{46}$$

If $Zh = \mathbf{0}$ as in (10), then from (46), $d\vec{\theta} = \mathbf{0}$, which means that joint variables reach an equilibrium state—or a stationary state—which is mostly the optimal configuration for a given tip location. Conversely, if $d\vec{\theta} = \mathbf{0}$, we have $Zh = \mathbf{0}$ since the rank of $Z^T (ZZ^T)^{-1}$ in (46) is $n - m$ and Zh is an $(n - m)$-dimensional vector. Therefore, $Zh = \mathbf{0}$ or (10) of the proposed method is the necessary and sufficient condition to be satisfied when (20) of the RM method reaches its equilibrium state.

Acknowledgment

In this research, the idea of the fixed transformation from the workspace to the joint space was first suggested by Berthold Horn, who also made comments on the final draft, which the author gratefully acknowledges. He is also thankful to Thomás Lozano-Pérez and Ken Salisbury for their detailed comments on drafts.

References

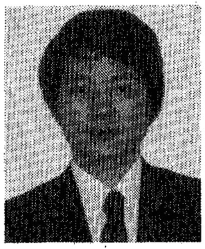
[1] J. Angeles, "On the numerical solution of the inverse kinematic problem," *Int. J. Robotics Res.*, vol. 4, no. 2, pp. 21-37, 1985.
[2] J. Baillieul, "Kinematic programming alternatives for redundant manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, MO, Mar. 25-28, 1985, pp. 722-728.
[3] ——, "Avoiding obstacles and resolving kinematic redundancy," in *Proc. IEEE Int. Conf. Robotics and Automation,* San Francisco, CA, Apr. 1985, pp. 1698-1704.
[4] M. Benati, P. Morasso, and V. Tagliasco, "The inverse kinematic problem for anthropomorphic manipulator arms," *ASME J. Dynamic Syst., Meas., Contr.*, vol. 104, pp. 110-113, 1982.
[5] A. Ben-Israel and T. Greville, *Generalized Inverses: Theory and Applications.* New York: Krieger, 1980.
[6] R. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed. New York: McGraw-Hill, 1973.
[7] F. Hildebrand, *Methods of Applied Mathematics*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1965.
[8] J. Hollerbach, "Optimum kinematic design for a seven degree of freedom manipulator," in *Proc. Robotics Research: 2nd Int. Symp.*, H. Hanafusa and H. Inoue, Eds. Cambridge, MA: MIT Press, 1985, pp. 215-222.
[9] C. Klein and C. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 2, pp. 245-250, 1983.
[10] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 868-871, Dec. 1977.
[11] J. Moré, B. Garbow, and K. Hillstrom, "User guide for MINPACK-1," Argonne National Laboratory, 1980.
[12] Y. Nakamura, "Kinematical studies on the trajectory control of robot manipulators," Ph.D. dissertation, Automation Research Lab., Kyoto Univ., Kyoto, Japan, 1985.
[13] D. Orin and W. Schrader, "Efficient Jacobian determination for robot manipulators," in *Proc. Robotics Research: 1st Int. Symp.*, M. Brady and R. Paul, Eds. Cambridge, MA: MIT Press, 1984, pp. 727-734.

[14] D. Pieper, "The kinematics of manipulators under computer control," Ph.D. dissertation, Dept. Computer Science, Stanford Univ., Stanford, CA, 1968.

[15] M. Raibert and B. Horn, "Manipulator control using the configuration space method," *Ind. Robot,* vol. 5, no. 2, pp. 69–73, June 1978.

[16] D. Whitney, "The mathematical coordinated control of manipulators and human prostheses," *ASME J. Dynamic Syst., Meas., Contr.*, vol. 94, no. 4, pp. 303–309, 1972.

[17] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Proc. Robotics Research: 1st Int. Symp.*, M. Brady and R. Paul, Eds. Cambridge, MA: MIT Press, 1984, pp. 735–748.



Pyung H. Chang (S'86–M'86) was born in Pusan, Korea, in 1951. He received the B.S.M.E. and M.S.M.E. degrees from Seoul National University (SNU), Seoul, Korea, in 1974 and 1977, respectively. Since 1980, he has been at the Massachusetts Institute of Technology (MIT), Cambridge, where he is a Ph.D. candidate in mechanical engineering and a member of the MIT Artificial Intelligence Laboratory.

His research interests span robotics, systems and control, and machine vision.