A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators

Li-Chun Tommy Wang and Chih Cheng Chen

Abstract—**A new method for computing numerical solutions to the inverse kinematics problem of robotic manipulators is developed in this paper. The proposed method is based on a combination of two nonlinear programming techniques and the forward recursion formulas, with the joint limitations of the robot being handled implicitly as simple boundary constraints. This method is numerically stable since it converges to the correct answer with virtually any initial approximation, and it is not sensitive to the singular configurations of the manipulator. In addition, this method is computationally efficient and can be applied to serial manipulators having any number of degrees of freedom.**

I. Introduction

THE transformation of the position and orientation of a manipulator end-effector from Cartesian coordinates to joint coordinates is known as the inverse kinematics problem. The problem is important for off-line robot trajectory planning, motion control, and work space analysis.

Closed-form solutions to this problem are available only for certain classes of industrial robots with simplified structures [4], [10][1] In addition, these solutions are not only manipulator dependent but are also subject to uncertainty due to manufacturing errors [18]. Therefore, for the development of a general-purpose computer-aided robot design and analysis program, a numerical approach is essential.

There are basically two types of numerical methods that have been developed for solving the inverse kinematics problem of manipulators and general spatial mechanisms. The first type uses either the Newton–Raphson method to solve the nonlinear kinematic equations [1], [21] or predictor-corrector-type algorithms to integrate the differential kinematic equations [7], [19]. The major difficulty with this method is that, when the Jacobian matrix is singular (or ill-conditioned), it does not find a solution. In addition, if the initial approximation of the solution vector (i.e., the vector of joint variables) is not sufficiently accurate, this method may become unstable. Several modifications have been developed to overcome these difficulties [7], [19], [20]; however, when the manipulator is in an *exact* singular configuration, these algorithms still fail to converge [12].

The second type is based on optimization techniques. Instead of solving the inverse kinematics problem directly, this method uses gradient-based nonlinear programming (NP) algorithms to solve an equivalent minimization problem [6], [7], [9]. In general, since the inverse Jacobian matrix is not used, this method is numerically more stable than the first method. The complexity of the minimization problem, is highly dependent on the formulation of the objective function. In addition, due to the highly nonlinear nature of the manipulator kinematic equations, the gradient vector of the objective function is usually evaluated numerically [9]. This, however, may increase the computation time considerably.

There also exist algorithms that are based on *heuristic* direct search techniques (which do not require the gradient information [12], [14]), but the *local* convergence rate of these methods are generally expected to be much slower than that of the gradient-based method [13], [15].

The method developed in this paper is based on the concept of combined optimization. It uses the cyclic coordinate descent (CCD) method [13] to rapidly find a feasible point that is near to the true solution and then uses the Broyden–Fletcher–Shanno (BFS) variable metric method [13], [15] to obtain a solution at the desired degree of precision. The joint variable limitations of the robot are handled implicitly as simple boundary constraints. This method is not sensitive to the initial or the singular configurations of the manipulator, and it fully exploits the strength of both the CCD and the BFS methods. In addition, instead of using the conventional Denavit–Hartenberg matrices [3], the kinematic equations and all related vectors are computed based on the more efficient forward recursion formulas developed by Wang and Ravani [23]. Therefore, this method is not only numerically stable but also computationally efficient.

The organization of this paper is as follows: The formulation of the problem and the objective function are presented in the next section. Section III gives a detailed derivation of the CCD method together with a complete solution procedure. The analytic form of the gradient vector and an efficient line search technique for computing the optimum search step size for the BFS method are developed in Section IV. Section V presents various numerical examples to demonstrate the

Manuscript received April 5, 1989; revised December 18, 1990. This work was supported by the National Science Committee of the Republic of China under Grant NSC78-0422-E001-06.

The authors are with the Department of Mechanical Engineering and Technology, National Taiwan Institute of Technology, Taipei 10772, Taiwan, Republic of China.

IEEE Log Number 9144670.

[1]It should be noted that there exist methods for finding the closed-form inverse kinematics solutions for more general robot configurations [18]. Such methods, however, are very elaborate and do not seem to be useful for real-time applications.

1042-296X/91/0800-0489$01.00

More — on the outside.

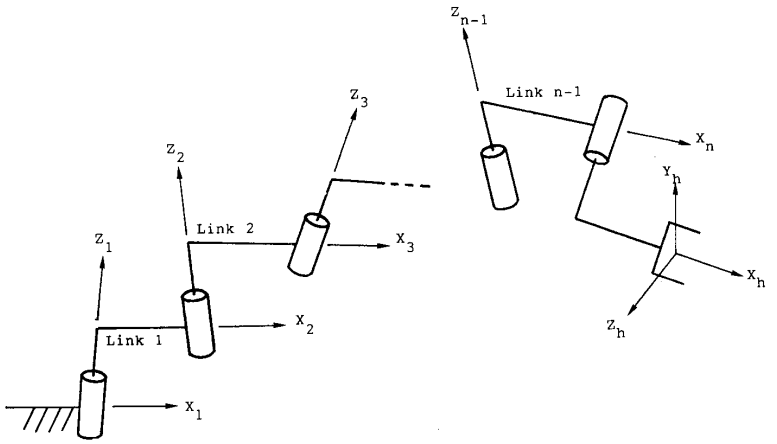


Fig. 1. Schematic diagram of a general n DOF manipulator.

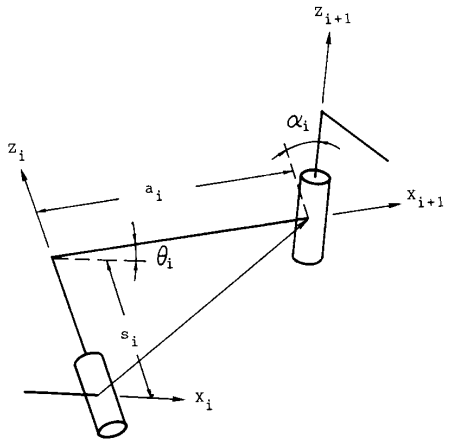


Fig. 2. Definition of link parameters.

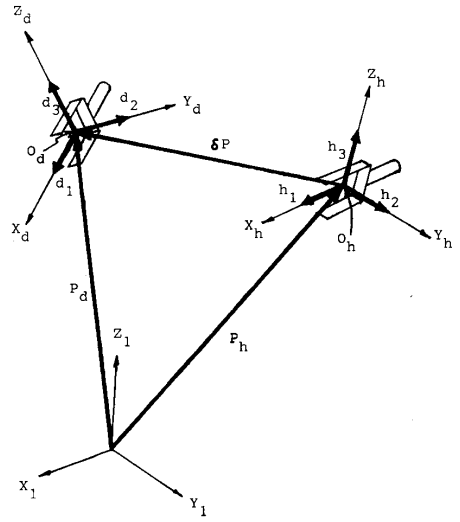


Fig. 3. The current and the desired end-effector configurations.

stability and efficiency of the method. The conclusions from this work are provided in the last section.

II. Problem Formulation

In this paper we are only concerned with manipulators with rigid binary links, open-loop kinematic structures, and single-degree-of-freedom joints. Fig. 1 shows such a manipulator with n degrees of freedom. There are a total of $n + 1$ coordinate systems attached to the manipulator. The first (base) coordinate system is used as a frame of reference and is fixed to the ground. The last coordinate system (x_h y_h z_h) is attached to the end-effector. The other coordinate systems are attached to the links, noting that the one attached on link i is the $i + 1$th coordinate system. The definitions and symbols used in this paper for link parameters are the same as those used in Wang and Ravani's study [23], as illustrated in Fig. 2. It should be noted that the joint axes are aligned with the z axes of the coordinate systems; if joint i is a rotational joint, then the rotation angle θi is the joint variable; if joint i is a translational (or prismatic) joint, then the joint variable is the link offset s_i.

The Cartesian coordinates of the end-effector are defined as the position vector $\boldsymbol{P}_h(\boldsymbol{q})$, and the orientation matrix $[\boldsymbol{R}_h(\boldsymbol{q})]$. where

- $\boldsymbol{P}_h(\boldsymbol{q})$ is the current position of the origin of the coordinate system, (x_h y_h z_h);
- $[\boldsymbol{R}_h(\boldsymbol{q})]$ $= [\boldsymbol{h}_1(\boldsymbol{q}) \mid \boldsymbol{h}_2(\boldsymbol{q}) \mid \boldsymbol{h}_3(\boldsymbol{q})]$, where $\boldsymbol{h}_1(\boldsymbol{q})$, $\boldsymbol{h}_2(\boldsymbol{q})$ and $\boldsymbol{h}_3(\boldsymbol{q})$ are unit vectors along the x_h, y_h, and z_h axes, as shown in Fig. 3; and
- $\boldsymbol{q}$ $= [q_1, q_2, \cdots, q_n]^t$ is the $n \times 1$ vector of the joint variables.

The inverse kinematics problem can be stated as follows:

Problem P1: Given the desired Cartesian coordinates of the end-effector, $\boldsymbol{P}_d$ and $[\boldsymbol{R}_d] = [\boldsymbol{d}_1 \mid \boldsymbol{d}_2 \mid \boldsymbol{d}_3]$, where $\boldsymbol{d}_j$ (j = 1 to 3) are unit vectors along the x_d, y_d, and x_d axes (see Fig. 3). Find the joint variable vector $\boldsymbol{q}$ such that $\boldsymbol{P}_h(\boldsymbol{q}) = \boldsymbol{P}_d$ and $[\boldsymbol{R}_h(\boldsymbol{q})] = [\boldsymbol{R}_d]$. In addition, the solution vector should satisfy the physical limitations of the joints, i.e., $q_i^l \le q_i \le$

q_i^u for $i = 1$ to n. Where q_i^l and q_i^u are, respectively, the lower and upper bounds of the ith joint variable.

For any given q, the Cartesian location (position and orientation) of the end-effector can be computed by using the forward recursion formulas [23]. If q is not a solution of problem P1, then the computed location will not agree with the desired location, as shown in Fig. 3. However, if we can find a correction vector δq such that $q^* = q + \delta q$ is a solution to problem P1, then point o_h should coincide with o_d and the orientation of the two coordinate systems should be identical. This implies that $\| \delta P(q^*) \| = \| P_d - P_h(q^*) \| = 0$ and $d_j \cdot h_j(q^*) = 1$ for $j = 1$ to 3, where $\| \cdot \|$ denotes the Euclidean norm of a vector and a $(\cdot)$ denotes the vector dot product. Therefore, the errors between the current and the desired locations of the end-effector can be described by the following positive scalar functions of q:

$$\text{Position error:} \quad \Delta P(q) = \delta P(q) \cdot \delta P(q) \tag{1}$$

$$\text{Orientation error:} \quad \Delta O(q) = \sum_{j=1}^{3} (d_j \cdot h_j(q) - 1)^2 \tag{2}$$

$$\text{The total error:} \quad E(q) = \Delta P(q) + \Delta O(q) \tag{3}$$

and the original inverse kinematics problem can be transformed into the following minimization problem:

Problem P2: Find q^* such that $E(q^*) \min \{ E(q) \mid q_i^l \le q_i \le q_i^u,\ i = 1 \text{ to } n \}$ and $E(q^*) \le \epsilon$ where $\epsilon \to 0$ is a prescribed small tolerance.

Note that (2) gives equal weighting of the orientation error to each of the three directions; this, however, may not be necessary in some practical applications. In order to handle such situations, (2) can be modified to

$$\Delta O(q) = \sum_{j=1}^{3} \sigma_j (d_j \cdot h_j(q) - 1)^2$$

where

$$\sigma_j = \begin{cases} 1 & \text{if the } j\text{th direction needs specifying} \\ 0 & \text{otherwise.} \end{cases}$$

For example, when a robot is used for arc welding, only the direction of the electrode (or the welding torch) is important. For this case, we can arrange the electrode to coincide with the z_h axis of the end-effector and then assign $\sigma_1 = \sigma_2 = 0$ and $\sigma_3 = 1$.

III. The Solution Procedure

The forenamed minimization problem (P2) is an n-dimensional nonlinear programming (NP) problem. In order to solve this problem efficiently, the solution procedure developed in this section consists of two phases. The first phase uses the cyclic coordinate descent (CCD) method, and the second phase uses the Broyden–Fletcher–Shanno (BFS) method.[2]

[2] In fact, any NP algorithm that has good local convergence properties can be used. The BFS method is selected because numerical experiments indicate that it is less dependent upon an exact line search [15].

A. The Cyclic Coordinate Descent (CCD) Method

The CCD method is a heuristic direct search method [13], [15]. Each cycle of this method consists of n steps; at the ith (i varying from n to 1) step, only the ith joint variable is allowed to be changed to minimize the objective function. The configuration of the robot is updated *after* each completed cycle. This cyclic process is continued until the value of the objective function reaches a predetermined small tolerance.

It should be pointed out that Kazerounian [12] has also developed a direct search algorithm to solve the inverse kinematics problem based on the same concept developed in the CCD method. However, while Kazerounian's method uses zero position analysis and forward cycles (i.e., i varying from 1 to n), the CCD method developed here uses forward recursion formulas and backward cycles. In addition, the objective function formulation of the two methods are quite different. Consequently, the computations required by the CCD method are considerably less than those required by Kazerounian's method, as shown in Table I.

Fig. 4 illustrates one step in the CCD method. Here, vector P_i indicates the current position of point o_i (the origin of the ith coordinate frame), P_{ih} is the vector from o_i to the current position of the end-effector, and p_{id} is a vector from o_i to the *desired* position of the end-effector. At the ith step, vectors P_{id} and P_i remain constant, and in order to minimize the position and orientation error of the end-effector, vector P_{ih} and the coordinate system (x_h, y_h, z_h) are allowed to either rotate about or translate along the z_i axis depending on the type of joint i. Hence, two cases must be considered.

Case A—Joint i Is a Rotational Joint

In this case, the joint variable q_i is the rotation angle θ_i. Since the other joint variables $(q_k,\ k = 1 \cdots i-1,\ i+1 \cdots n)$ are not allowed to be changed, P_{ih} can be considered as a free vector. If we rotate it about the z_i axis with an angle ϕ, then the rotated vector can be written as

$$P_{ih}'(\phi) = [R(z_i,\phi)] P_{ih} \tag{4}$$

where $[R(z_i,\phi)]$ is the 3×3 spatial rotation matrix, and z_i is an unit vector along the z_i axis (expressed with respect to the base coordinate frame). Consequently, the position error becomes a *single* variable function of ϕ

$$\Delta p(\phi) = (P_{id} - P_{ih}'(\phi)) \cdot (P_{id} - P_{ih}'(\phi)). \tag{5}$$

Substituting (4) into (5) and noting that $[R(z_i,\phi)]$ is an *orthogonal* matrix, one obtains

$$\begin{aligned} \Delta p(\phi) &= P_{id} \cdot P_{id} + P_{ih} \cdot \left([R(z_i,\phi)]' [R(z_i,\phi)] P_{ih} \right) \\ &\quad - 2 P_{id} \cdot ([R(z_i,\phi)] P_{ih}) \\ &= P_{id} \cdot P_{id} + P_{ih} \cdot P_{ih} - 2 P_{id} \cdot ([R(z_i,\phi)] P_{ih}) \end{aligned}$$

where $[\cdot]'$ indicates the transposed matrix. Since $P_{id} \cdot P_{id}$ and $P_{ih} \cdot P_{ih}$ are positive constants, to minimize $\Delta p(\phi)$ would be the same as to *maximize*

$$g_1(\phi) = P_{id} \cdot [R(z_i,\phi)] P_{ih}. \tag{6}$$

TABLE I
COMPARISON OF COMPUTATIONAL REQUIREMENTS FOR KAZEROUNIAN'S METHOD AND THE CCD METHOD

Method		ith Step*	One Cycle*	$n = 6$
Kazerounian	Mult.	$18(n - i) + 212$	$9n^2 + 209n$	1578
	Add.	$15n - 9i + 181$	$10\frac{1}{2}n^2 + 176\frac{1}{2}n + 3$	1440
CCD	Mult.	141	$171n - 5$	1021
	Add.	101	$119n + 3$	717

*The computational requirements are for rotational joints only.

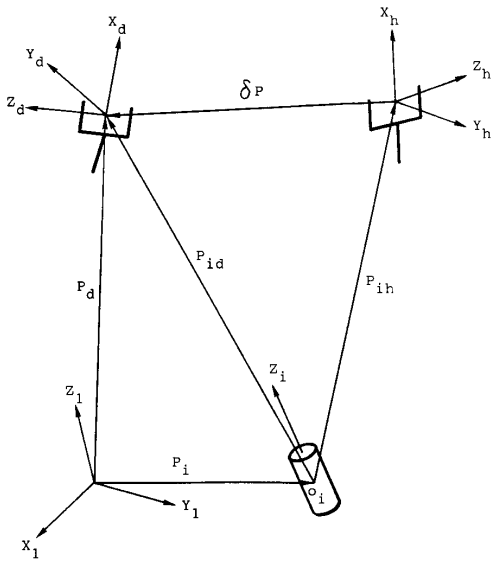


Fig. 4. One step of the CCD method.

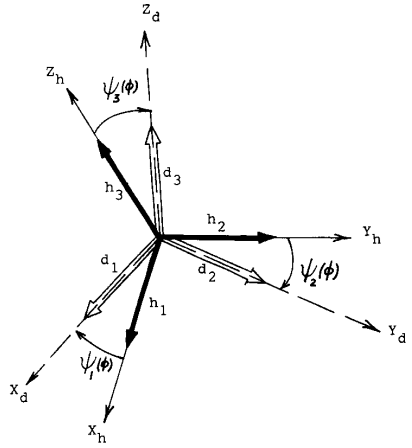


Fig. 5. Definition of the direction angles.

Similarly, if we rotate the last coordinate system about the z_i axis with an angle ϕ then the orientation vectors become

$$\boldsymbol{h}_j'(\phi) = \left[\boldsymbol{R}(z_i,\phi)\right]\boldsymbol{h}_j \qquad \text{for } j = 1 \text{ to } 3 \tag{7}$$

and the orientation error (see (2)) becomes

$$\Delta o(\phi) = \sum_{j=1}^{3} \left(\boldsymbol{d}_j \cdot \boldsymbol{h}_j'(\phi) - 1\right)^2 \tag{8}$$

which is also a function of ϕ only. Noting that both $\boldsymbol{d}_j$ and $\boldsymbol{h}_j'(\phi)$ are unit vectors; thus

$$\boldsymbol{d}_j \cdot \boldsymbol{h}_j'(\phi) = \cos \psi_j(\phi)$$

where $\psi_j(\phi)$ is the direction angle between vectors $\boldsymbol{d}_j$ and $\boldsymbol{h}_j$, as shown in Fig. 5. Consequently, (8) can be written as

$$\Delta o(\phi) = \sum_{j=1}^{3} \left(\cos \psi_j(\phi) - 1\right)^2. \tag{9}$$

The physical meaning of (9) can be interpreted with the help of Fig. 5. It is clear from this figure that the orientation error between the two coordinate systems would be minimized as $\psi_j(\phi)$ approaches zero for all j, or when the three direction cosines, $\cos \psi_j(\phi)$, simultaneously approach one; this can be achieved by minimizing (9). Alternatively, since $\cos \psi_j(\phi)$ is bounded between ± 1, the orientation error can also be effectively reduced by *maximizing*

$$g_2(\phi) = \sum_{j=1}^{3} \cos \psi_j(\phi) = \sum_{j=1}^{3} \boldsymbol{d}_j \cdot \boldsymbol{h}_j'(\phi). \tag{10}$$

It should be pointed out that it is not exactly equivalent to minimize (9) or maximize (10), since the optimum solutions for ϕ in the two functions, in general, would be different. However, as ϕ is iteratively adjusted so that $\cos \psi_j(\phi)$ coverges to one for all j, the solutions are asymptotically equivalent. The reason for using (10) instead of (9) to formulate the orientation error is that the computations involved in minimizing (9) are much more complicated than those involved in maximizing (10), since the main purpose of the CCD method is to efficiently find a good starting value for the joint variables for use in the BFS method; therefore, in order to increase computational efficiency, (10) is, in general, a more suitable alternative. Also notice that (10) gives equal weighting of the orientation error to each of the three directions of the end-effector coordinate system, and, if desired, it can be easily modified to

$$g_2(\phi) = \sum_{j=1}^{3} \sigma_j \boldsymbol{d}_j \cdot \boldsymbol{h}_j'(\phi)$$

where σ_j is equal to either 1 or 0, depending on whether the jth direction needs to be specified or not.

Combining (6) and (10), the objective function (for this case) can be defined as

$$g(\phi) = w_p g_1(\phi) + w_o g_2(\phi) \tag{11}$$

where w_p and w_0 are weighting factors that are arbitrary positive real numbers.[3] Now the problem becomes:

Problem P3: Find ϕ^*, such that $g(\phi^*) = \max\{g(\phi) \mid \phi^l \le \phi \le \phi^u\}$, where $\phi^l = \theta_{ic} - \theta_i^l$, $\phi^u = \theta_i^u - \theta_{ic}$, θ_i^l and θ_i^u are the lower and upper bounds of the joint variable θ_i, respectively, and θ_{ic} is the current value of θ_i.

The analytical solution of problem P3 can be easily derived. According to the vector form of Rodrigues' equation [23], it can be shown that

$$[\boldsymbol{R}(z_i,\phi)]\boldsymbol{P}_{ih} = z_i(\boldsymbol{P}_{ih} \cdot z_i)(1 - \cos\phi) + \boldsymbol{P}_{ih}\cos\phi + (z_i \times \boldsymbol{P}_{ih})\sin\phi \tag{12}$$

where $\times$ denotes the vector cross product. Substituting (12) into (6), $g_1(\phi)$ becomes

$$g_1(\phi) = (\boldsymbol{P}_{id} \cdot z_i)(\boldsymbol{P}_{ih} \cdot z_i)(1 - \cos\phi) + (\boldsymbol{P}_{id} \cdot \boldsymbol{P}_{ih})\cos\phi + \boldsymbol{P}_{id} \cdot (z_i \times \boldsymbol{P}_{ih})\sin\phi.$$

Similarly, (10) can be expanded into

$$g_2(\phi) = \sum_{j=1}^{3} \{(\boldsymbol{d}_j \cdot z_i)(\boldsymbol{h}_j \cdot z_i)(1 - \cos\phi) + (\boldsymbol{d}_j \cdot \boldsymbol{h}_j)\cos\phi + \boldsymbol{d}_j \cdot (z_i \times \boldsymbol{h}_j)\sin\phi\}$$

Substituting these expressions into (11) after combining terms, the objective function becomes

$$g(\phi) = k_1(1 - \cos\phi) + k_2\cos\phi + k_3\sin\phi$$

where k_1, k_2, and k_3 are *constant* coefficients given by

$$k_1 = w_p(\boldsymbol{P}_{id} \cdot z_i)(\boldsymbol{P}_{ih} \cdot z_i) + w_0\sum_{j=1}^{3}(\boldsymbol{d}_j \cdot z_i)(\boldsymbol{h}_j \cdot z_i)$$

$$k_2 = w_p(\boldsymbol{P}_{id} \cdot \boldsymbol{P}_{ih}) + w_0\sum_{j=1}^{3}(\boldsymbol{d}_j \cdot \boldsymbol{h}_j)$$

$$k_3 = z_i \cdot \left[w_p(\boldsymbol{P}_{ih} \times \boldsymbol{P}_{id}) + w_0\sum_{j=1}^{3}(\boldsymbol{h}_j \times \boldsymbol{d}_j)\right].$$

If there are no boundary constraints imposed on ϕ, then $g(\phi)$ is maximized when

$$\frac{dg(\phi)}{d\phi} = (k_1 - k_2)\sin\phi + k_3\cos\phi = 0$$

and

$$\frac{d^2g(\phi)}{d\phi^2} = (k_1 - k_2)\cos\phi - k_3\sin\phi < 0$$

A unique value of ϕ, denoted as $\tilde{\phi}$, can be determined from these conditions. The actual solution of problem P3, however, should also satisfy the boundary constraints. Thus, if $\phi^l \le \tilde{\phi} \le \phi^u$ then $\phi^* = \tilde{\phi}$ (notice that the periodic solutions $\tilde{\phi} \pm 2\pi$ should also be checked to see if they are inside the boundary constraints). Otherwise, if $g(\phi^u) > g(\phi^l)$, then $\phi^* = \phi^l$; if $g(\phi^u) > (\phi^l)$, then $\phi^* = \phi^l$.

Case B—Joint i Is a Translational Joint

For this case, only the position error can be reduced because the orientation of the end-effector is independent of the joint variable s_i. Denoting the change of s_i as λ (measured from s_{ic}, the current value of s_i), the position error becomes

$$\Delta p(\lambda) = (\boldsymbol{P}_{id} - (\boldsymbol{P}_{ih} + z_i\lambda)) \cdot (\boldsymbol{P}_{id} - (\boldsymbol{P}_{ih} + z_i\lambda)) = \delta\boldsymbol{P} \cdot \delta\boldsymbol{P} - 2(\delta\boldsymbol{P} \cdot z_i)\lambda + \lambda^2$$

where $\delta\boldsymbol{P} = \boldsymbol{P}_{id} - \boldsymbol{P}_{ih}$, as shown in Fig. 4, and the problem becomes:

Problem P4: Find λ^*, such that $\Delta p(\lambda^*) = \min\{\Delta P(\lambda) \mid \lambda^l \le \lambda \le \lambda^u\}$, where $\lambda^l = s_{ic} - s_i^l$, $\lambda^u = s_i^u - s_{ic}$, and s_i^u and s_i^l are the upper and lower bounds of s_i, respectively.Since

$$\frac{d\Delta p(\lambda)}{d\lambda} = 2(\lambda - \delta\boldsymbol{P} \cdot z_i)$$

and

$$\frac{d^2\Delta p(\lambda)}{d\lambda} = 2 > 0.$$

Thus, if there are no boundary constraints imposed on λ, then $\Delta p(\lambda)$ is minimized when $\lambda = \tilde{\lambda} = \delta\boldsymbol{P} \cdot z_i$. Similar to the previous case, the actual solution of problem P4 should satisfy the boundary constraints. Thus, if $\lambda^l \le \tilde{\lambda} \le \lambda^u$, then $\lambda^* = \tilde{\lambda}$. Otherwise, if $\Delta p(\lambda^u) < \Delta p(\lambda^l)$, then $\lambda^* = \lambda^u$; if $\Delta p(\lambda^l) < \Delta p(\lambda^u)$, then $\lambda^* = \lambda^l$.

Hence, at each step in the CCD method, for either case, the original n-dimensional minimization problem is reduced to a simple one-dimensional problem. In addition, the analytical solution of the one-dimensional problem can be easily computed. Furthermore, since the value of the objective function is reduced with each step, the global convergency of this method is guaranteed. It also is not sensitive to the singular configuration of the manipulator. Therefore, the initial approximation of the solution vector can be arbitrary. However, due to the heuristic nature of this method, the rate of convergency is highly dependent on the structure of the manipulator. Our experience shows, in general, when the initial approximation is far away from the true solution, applying only a few cycles of this method will bring the solution vector into the neighborhood of the true solution, but it may become very slow thereafter. On the other hand, when the solution vector is near the true solution, the BFS method

[3]The following weighting factors are suggested in this paper: $w_0 = 1$, $w_p = \alpha(1 + \rho)$, where α is a sizing factor depending on the dimension of the link lengths, and

$$\rho = \min(\|\boldsymbol{P}_{id}\|, \|\boldsymbol{P}_{ih}\|)/\max(\|\boldsymbol{P}_{id}\|, \|\boldsymbol{P}_{ih}\|).$$

is generally expected to give a superlinear or a near-quadratic convergence rate [15]. Therefore, the CCD method is especially suitable for finding good starting values for the BFS method. The criterion for switching between the two methods can be designated as follows:

After each completed cycle of the CCD method, compare the current value of the objective function E_c with the one computed from the previous cycle, E_p. If $E_p < \beta$ and $E_c > E_p^2$ (where $\beta \leq 1$ is a small positive constant to indicate the closeness of the solution vector, the value can be designated according to the dimension of the manipulator), then the solution vector is in the neighborhood of the true solution and the convergence rate is less than quadratic; switch to the BFS method.

B. Numerical Algorithm

The complete solution procedure is summarized below:

Step 1: Input the link parameters of the robot and the boundary constraints q_i^l and q_i^u ($i = 1$ to n), specify the desired location of the end-effector, $\boldsymbol{P}$ and $[\boldsymbol{R}_d]$, the termination criterion ϵ, and the initial guess for the joint variable vector.

Phase I

Step 2: Compute $\boldsymbol{P}_i$, $\boldsymbol{P}_{i-1}^*$, x_i, y_i, z_i, for $i = 2$ to $n + 1$ using the forward recursion formulas. Also compute $\boldsymbol{P}_{ih} = \boldsymbol{P}_{n+1} - \boldsymbol{P}_i$, for $i = n$ to 1.
Set $\boldsymbol{h}_1 = \boldsymbol{x}_{n+}\mathbf{1}$, $\boldsymbol{h}_2 = y_{n+1}$, $\boldsymbol{h}_3 = z_{n+1}$ and $\boldsymbol{P}_h = \boldsymbol{P}_{n+1}$.
Compute the current total error E_c by using (1), (2), and (3).
If $E_c \leq \epsilon$, then stop; otherwise, go through the switch method criterion:
If $E_p < \beta$ and $E_c > E_p^2$, then go to step 4 (the initial value of E_p can be specified by a large real number); otherwise, set $E_p = E_c$ and go to the next step.

Step 3: For $i = n$ to 1 **do**
if joint i is a rotation joint
then compute ϕ^* by using the method developed in Case A of Section III-A. Update $q_i = q_i + \phi^*$, and $\boldsymbol{P}_{ih} = \boldsymbol{P}_{ih}'(\phi^*)$, $\boldsymbol{h}_j = \boldsymbol{h}_j'(\phi^*)(j = 1$ to 3) by using (4) and (7).
else (joint i is a translational joint)
compute λ^* by using the method developed in Case B of Section III-A. Update $q_i = q_i + \lambda^*$, $\boldsymbol{P}_{ih} = \boldsymbol{P}_{ih} + \lambda^* z_i$
endif;
If $i > 1$ then set $\boldsymbol{P}_{i-1}$,h $= \boldsymbol{P}_{ih} + \boldsymbol{P}_{i_1}^*$
end do;
go to Step 2.

Phase II

Step 4: Start the BFS method. The BFS algorithm used in this paper is based on the one given in [13] with a self-scaling and restart feature. However, in order to handle the joint variable constraints, minor modifications are required (see the Appendix).

The above algorithm is robust and converges rapidly for all the problems that we have tested. However, it should be pointed out that there are two difficulties in using this algorithm. One difficulty is that it seldom converges to a local minimum point (i.e., the objective function stops decreasing, but is still greater than ϵ), which does not help to solve the inverse kinematics problem. Another difficulty is that for manipulators that have multiple solutions (each solution corresponding to one of the global minimum points), this method does not guarantee the finding of all possible solutions. These difficulties are generic to all iterative algorithms of solving systems of nonlinear equations [9]. Nevertheless, both of the difficulties are not severe since they can usually be resolved by perturbing the initial guesses, as illustrated by a numerical example presented in Section V.

IV. Computational Considerations

It is well known that the most time-consuming steps of the BFS algorithm (or any other gradient-based NP algorithm) are the computation of the gradient vector of the objective function and the evaluation of the optimum step size in the search direction [2], [13], [15].

In this section, the analytical form of the gradient vector and an effective method to approximate the optimum search step size are developed. As a result, the computation load of the BFS method is significantly reduced.

A. The Gradient Vector

The objective function $\boldsymbol{E}(\boldsymbol{q})$ defined in the second section can be written as

$$E(\boldsymbol{q}) = (\boldsymbol{P}_d - \boldsymbol{P}_h(\boldsymbol{q})) \cdot (\boldsymbol{P}_d - \boldsymbol{P}_h(\boldsymbol{q})) + \sum_{j=1}^{3} (\boldsymbol{d}_j \cdot \boldsymbol{h}_j(\boldsymbol{q}) - 1)^2. \quad (13)$$

The gradient vector $\nabla \boldsymbol{E}(\boldsymbol{q})$ is defined by

$$\nabla E(\boldsymbol{q}) = \left[\frac{\partial E(\boldsymbol{q})}{\partial q_1}, \cdots \frac{\partial E(\boldsymbol{q})}{\partial q_i}, \cdots \frac{\partial E(\boldsymbol{q})}{\partial q_n} \right]^t.$$

According to (13), the elements of the gradient vector are

$$\frac{\partial E(\boldsymbol{q})}{\partial q_i} = 2\left\{ [\boldsymbol{P}_h(\boldsymbol{q}) - \boldsymbol{P}_d] \cdot \frac{\partial \boldsymbol{P}_h(\boldsymbol{q})}{\partial \boldsymbol{q}_i} + \sum_{j=1}^{3} [\boldsymbol{d}_j \cdot \boldsymbol{h}_j(\boldsymbol{q}) - 1] \left[\boldsymbol{d}_j \cdot \frac{\partial \boldsymbol{h}_j(\boldsymbol{q})}{\partial q_i} \right] \right\}$$

for $i = 1$ to n. (14)

It can be shown that [22]

$$\frac{\partial \boldsymbol{P}_h(\boldsymbol{q})}{\partial q_i} = \boldsymbol{b}_i \quad (15)$$

where

$$b_i = \begin{cases} z_i \times P_{ih} & \text{if joint } i \text{ is a rotational joint} \\ z_i & \text{if joint } i \text{ is a translational joint} \end{cases}$$

and

$$\frac{\partial h_i(q)}{\partial q_i} = Q_i \tag{16}$$

where

$$Q_i = \begin{cases} z_i \times h_j(q) & \text{if joint } i \text{ is a rotational joint} \\ 0 & \text{if joint } i \text{ is a translational joint.} \end{cases}$$

Substituting (15) and (16) into (14), after rearranging and combining terms, one obtains

$$\frac{\partial E(q)}{\partial q_i} = \begin{cases} 2z_i \cdot \left[(P_d - P_h(q)) \times P_{ih} + \sum_{j=1}^{3} (d_j \cdot h_j(q) - 1)(h_j(q) \times d_j) \right] \\ \quad \text{if joint } i \text{ is a rotational joint} \\ 2z_i \cdot (P_h(q) - P_d) \\ \quad \text{if joint } i \text{ is a translational joint.} \end{cases}$$

Given q, the vectors $P_h(q)$, $h_j(q)$ (j = 1 to 3), P_{ih} and z_i (i = 1 to n) can be recursively computed by using the forward recursion formulas. Consequently, the gradient vector $\nabla E(q)$ can be efficiently evaluated.

B. Line Search Technique

The BFS method uses an approximated Hessian matrix and the gradient vector to obtain the search direction. After the search direction is obtained, the objective function (see (1)) is reduced to a single variable function

$$E(\alpha) = \Delta P(\alpha) + \Delta O(\alpha) \tag{17}$$

where

$$\Delta P(\alpha) = (P_d - P_h(q_c + \alpha u)) \cdot (P_d - P_h(q_c + \alpha u)) \tag{18}$$

and

$$\Delta O(\alpha) = \sum_{j=1}^{3} \{ d_j \cdot h_j(q_c + \alpha u) - 1 \}^2 \tag{19}$$

where u is the search direction, q_c is the current joint positions, and α is the search step size. The line search problem is to find α^* such that α^* minimizes $E(\alpha)$.

Due to the highly nonlinear nature of the manipulator kinematic equations, the exact solution of this problem is difficult to obtain. Nevertheless, an approximated solution can be derived. In addition, this approximation is well justified when the solution vector is in the neighborhood of the true solution; this is the case for the algorithm developed in this paper since the BFS method is always started from a feasible point that is close to the true solution.

If α is sufficiently small, then the differential kinematic equations of the manipulator can be written as

$$\delta X = \alpha [J_c] u$$

or in partitioned form

$$\begin{bmatrix} \delta p \\ \delta o \end{bmatrix} = \alpha \begin{bmatrix} [J_{pc}] \\ [J_{oc}] \end{bmatrix} u$$

where $[J_c]$ is the $6 \times n$ Jacobian matrix (evaluated with respect to the *current* configuration of the manipulator), $[J_{pc}]$ and $[J_{oc}]$ are the upper and lower $3 \times n$ submatrices of $[J_c]$; δp and δo are, respectively, the corresponding Cartesian space differential displacement and rotation of the end-effector.

Therefore, the position vector $P_h(q_c + \alpha u)$ can be approximated by

$$\begin{aligned} P_h(q_c + \alpha u) &= P_h(q_c) + \delta p \\ &= P_{hc} + \alpha s_{pc} \end{aligned}$$

where $P_{hc} = P_h(q_c)$ and $s_{pc} = [J_{pc}]u$, and the position error (see (18)) becomes

$$\Delta P(\alpha) = (\delta P_c - \alpha s_{pc}) \cdot (\delta P_c - \alpha s_{pc}) \tag{20}$$

where $\delta P_c = P_d - P_{hc}$.

Noting that vector δo represents a differential rotation of the end-effector coordinate system, the orientation vectors $h_j(q_c + \alpha u)$ (j = 1 to 3) can be approximated by [16], [17]

$$\begin{aligned} h_j(q_c + \alpha u) &= h_j(q_c) + \delta o \times h_j(q_c) \\ &= h_{jc} + \alpha (s_{oc} \times h_{jc}) \end{aligned}$$

where $h_{jc} = h_j(q_c)$ and $s_{oc} = [J_{oc}]u$, and the orientation error (see (19)) becomes

$$\Delta O(\alpha) = \sum_{j=1}^{3} \{ d_j \cdot h_{jc} + d_j \cdot (s_{oc} \times h_{jc})\alpha - 1 \}^2. \tag{21}$$

Substituting (20) and (21) into (17), after rearranging and simplifying terms, (17) becomes

$$E(\alpha) = A - 2B\alpha + C\alpha^2$$

where

$$A = \delta P_c \cdot \delta P_c + \sum_{j=1}^{3} (h_{jc} \cdot d_j - 1)^2$$

$$B = s_{pc} \cdot \delta P_c + s_{oc} \cdot \left\{ \sum_{j=1}^{3} (h_{jc} \cdot d_j - 1)(d_j \times h_{jc}) \right\}$$

$$C = s_{pc} \cdot s_{pc} + \sum_{j=1}^{3} \{ (s_{oc} \times h_{jc}) \cdot d_j \}^2.$$

TABLE II
THE LINK PARAMETERS OF THE PUMA 560 ROBOT

Joint	Link Length a (meters)	Twist Angle α (degrees)	Offset Length s (meters)	Rotation Angle θ (degrees)
1	0	90	0.6604	q_1
2	0.432	0	0.200	q_2
3	0	90	−0.0505	q_3
4	0	−90	0.432	q_4
5	0	90	0.0	q_5
6	0	0	0.0565	q_6

TABLE III
THE JOINT VARIABLE LIMITATIONS OF THE PUMA 560 ROBOT

Joint	Lower Bound q^l (degrees)	UpperBound q^u (degrees)
1	−160	160
2	−225	45
3	−45	225
4	−110	170
5	−100	100
6	−266	266

Since $\dfrac{d^2E(\alpha)}{d\alpha^2} = 2C > 0$, $E(\alpha)$ is minimized when

$$\frac{dE(\alpha)}{d\alpha} = 2(-B + C\alpha) = 0$$

and the optimum search step size is

$$\alpha^* = \frac{B}{C}.$$

Given $\boldsymbol{q}_c$ and $\boldsymbol{u}$, the Jacobian matrix and vectors $\boldsymbol{P}_{hc}$ and $\boldsymbol{h}_{jc}$ (j = 1 to 3) can be recursively computed by using the algorithms developed by Wang and Ravani [23]. Consequently, the (approximated) optimum step size α^* can be efficiently evaluated.

V. NUMERICAL EXAMPLES

A computer program for solving the inverse kinematics problem for general manipulators has been developed by the authors. Users of this program can interactively input the structure of the manipulator to be studied and choose any one of the following four methods to solve their problem, namely, the CCD method, the BFS method, the combined CCD–BFS method, and the Newton–Raphson (NR) method (the original NR algorithm has been modified to handle the joint limits by using an algorithm similar to the one presented in the Appendix that restricts the step size). This program is used here to compare the reliability and efficiency of these methods.

Example 1: The structure of the manipulator used in this example is based on the PUMA 560 robot [11]. This robot is chosen simply because a closed-form solution can be obtained [4], which can be used for checking the accuracy of the numerical solution. The link parameters and the joint limits of this robot are listed in Tables II and III.

The convergence criterion for this example is defined by $\epsilon = 10^{-k}$, where k varies from 2 to 8. For each value of k, 100 problems were given to each of the four methods to solve. Both the initial and the desired configurations of the problems were randomly generated within the working space of the robot. The program was run on a VAX 780 computer, and the average execution (CPU) times of the *solved* problems are plotted in Fig. 6. From this figure, the NR method appears to be the most efficient method. However, while the other three methods were able to solve *all* of the given problems to the desired precision, the NR method diverged when trying to solve approximately 30% of the problems. Since numerical stability is an important consideration, therefore, the combined CCD–BFS method may be the most suitable as a general-purpose program.

Example 2: This example is similar to the first example except that while all of the desired configurations were randomly generated within the working space of the robot, the initial configurations were intentionally given at an exact singular configuration ($q_1 = q_2 = \cdots = q_6 = 0$). Since the modified NR method is unable to solve problems of this kind, only the other three methods are compared. It turned out that three methods were still able to solve all of the problems to the desired degree of precision. The average CPU times are plotted in Fig. 7. It is clear from this figure that the combined CCD-BFS method is the most efficient method, especially when the required degree of precision is high.

Example 3: The PUMA 560 robot can have at most eight solutions when there are no joint limits imposed [4]. Therefore, in order to demonstrate that the proposed method has the potential of finding *all* of the multiple solutions, the joint limitations were released in this example. The desired configuration of the end-effector was given by: $\boldsymbol{P}_d$ = [0.740 m, 0.331 m, 0.788 m]^t, and $\boldsymbol{d}_1$ = [−0.636, 0.771, −0.008]^t, $\boldsymbol{d}_2$ = [0.022, 0.029, 0.999]^t, $\boldsymbol{d}_3$ = [0.771, 0.635, −0.036]^t, which corresponds to an exact solution of $\boldsymbol{q}$ = [10°, 20°, 30°, 40°, 50°, 60°]. The algorithm had been modified to run iteratively. The initial guesses of the joint variables were

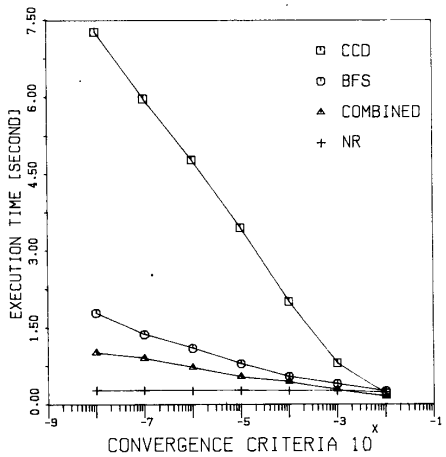


Fig. 6. Average execution time for numerical example 1.

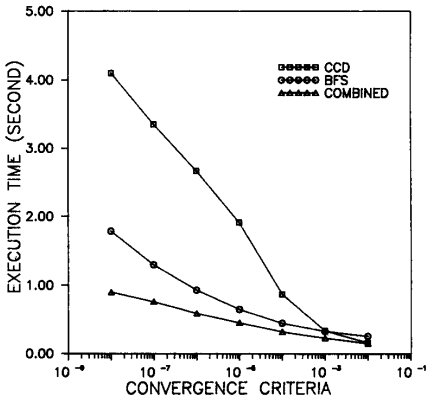


Fig. 7. Average execution time for numerical example 2.

TABLE IV
THE MULTIPLE SOLUTIONS OF THE PUMA 560 ROBOT

Solution Number	q_1 (degrees)	q_2 (degrees)	q_3 (degrees)	q_4 (degrees)	q_5 (degrees)	q_6 (degrees)
1	10.0264	19.9956	29.9771	39.3377	49.9444	60.3699
2	10.0342	−40.0295	150.0456	118.6216	33.4653	−35.3132
3	10.0253	19.9661	29.9964	−140.7245	−50.2606	−119.7849
4	−147.0222	159.9971	150.0357	8.8718	−42.6454	−96.8326
5	−147.0244	159.9910	150.0411	−171.0534	42.6024	82.1593
6	10.0408	−40.0320	149.9682	−62.8692	−33.5067	146.6351
7	−147.0298	−139.9917	29.9429	161.1484	−19.2311	106.8562
8	−147.0981	−139.9995	30.0623	−22.4287	18.7000	−69.6390

assigned as zeros in the first iteration, then perturbed by an increment of 7 degrees for the successive iterations. All eight solutions were found successfully (with $\epsilon = 10^{-8}$) after 95 iterations (see Table IV). The total execution time was about 60 s.

Example 4: This example demonstrates that the proposed method can also be used for continuous joint space trajectory planning. The robot structure for this example is based on the Stanford arm; the link parameters are given in Table V. The desired trajectory of the wrist center point is a circle centered at (0.2 m, −0.05 m, 0.5 m) with respect to the base coordinate frame and a radius equal to 0.2 m as shown in Fig. 8. The trajectory is discretized into 72 equally spaced points. The initial guesses of the joint variables used for the first point were zeros. Noting that multiple solutions do exist for the Stanford arm, and hence, in order to prevent a sudden jump to another solution branch, the initial guess for the solution vector for each of the successive points was given by the solution of the preceding point. Note that the convergence rate of the algorithm can also be significantly improved by using this scheme. The computed joint trajectories are plotted in Figs. 9 and 10. The *total* computation time for this example on the VAX 780 was only about 8 s.

Example 5: This example illustrates the ability of the proposed method for solving the inverse kinematics of redundant manipulators (i.e., $n > 6$). The link parameters of a redundant robot ($n = 7$) are given in Table VI. The desired configuration of the end-effector was given by: $\boldsymbol{P}_d = [-0.2$ m, 0.6 m, 0.5 m$)^t$, and $\boldsymbol{d}_1 = [1, 0, 0]^t$, $\boldsymbol{d}_2 = [0, 1, 0,]^t$, $\boldsymbol{d}_3 = [0, 0, 1]^t$. The initial guesses of the joint variables were assigned as zeros. The algorithm successfully converged ($\epsilon = 10^{-8}$) to a solution of $\boldsymbol{q} = [-181.914, -26.682, 137.791, 163.214, -75.440, -66.823, -32.284]^t$ (degrees). It should be noted that this is just one solution among an infinite number of solutions, but it can be used as an initial feasible solution in the search for the "optimum" solution [5].

TABLE V
THE LINK PARAMETERS OF THE STANFORD ROBOT

Joint	Link Length a (meters)	Twist Angle α (degrees)	Offset Length s (meters)	Rotation angle θ (degrees)
1	0	−90	0	q_1
2	0.15	90	0	$q2$
3	0	0	q_3	0
4	0	−90	0	q_4
5	0	90	0	q_5
6	0	0	0	q_6

VI. CONCLUSION

In this paper, a combined optimization method to find the numerical solutions to the inverse kinematics problem of

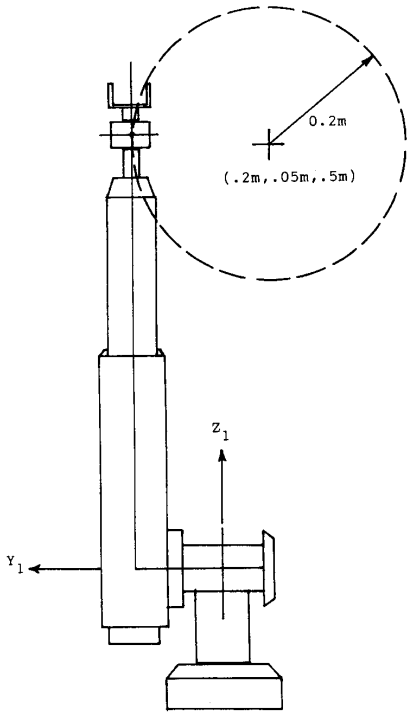


Fig. 8. The Stanford Arm and a circular trajectory.

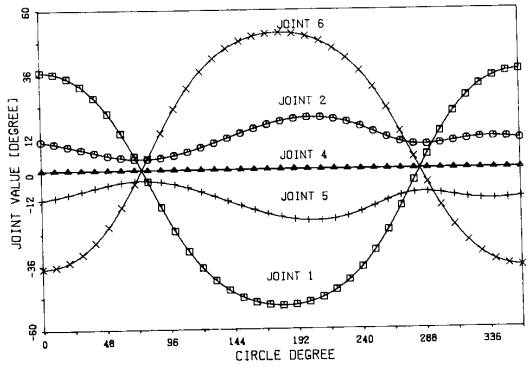


Fig. 9. Computed trajectories of the revolute joints—example 4.

mechanical manipulators has been developed. The major advantages of this method are that it is not sensitive to the initial and singular configurations of the manipulator, and it implicitly handles the joint variable boundary constraints. In addition, it can be applied to serial manipulators with any degree of freedom. This method is also computationally efficient and can be used for finding multiple solutions for the manipulator as well as for continuous trajectory planning, as indicated by the numerical examples presented in the last section.

With minor modifications, this method can also be applied to simple closed (single loop) spatial linkages for displacement analysis. Therefore, a promising future research topic is to extend this method to manipulators that have combined open-loop and closed-loop structures.

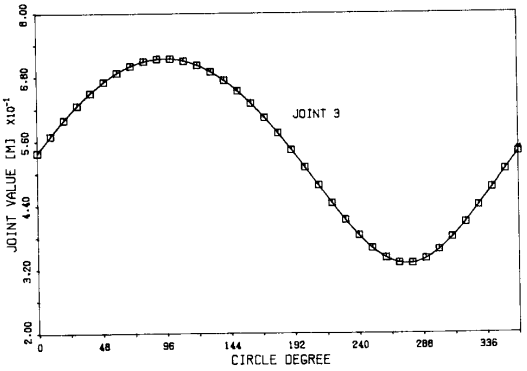


Fig. 10. Computed trajectory of the prismatic joint—example 4.

TABLE VI
THE LINK PARAMETERS OF THE REDUNDANT ROBOT

Joint	Link Length a (meters)	Twist Angle α (degrees)	Offset Length s (meters)	Rotation Angle θ (degrees)
1	0.700	90	0.0	q_1
2	0	90	0.6604	q_2
3	0.432	0	0.200	q_3
4	0	90	−0.0505	q_4
5	0	−90	0.432	q_5
6	0	90	0.0	q_6
7	0	0	0.0565	q_7

APPENDIX

In order to apply the BFS method to problems with simple boundary constraints, two modifications are necessary [15].

A. The Search Direction

Denote the unconstrained search direction as $\boldsymbol{u} = [u_1, u_2, \cdots, u_n]^t$, for the (simple) constrained problem; this direction vector should be modified to

$$\boldsymbol{u}_c = [u_{1c}, u_{2c}, \cdots, u_{nc}]^t$$

where

$$u_{ic} = \begin{cases} 0 & \text{if } q_{ic} \geq q_i^u \text{ and } u_i > 0 \\ 0 & \text{if } q_{ic} \leq q_i^l \text{ and } u_i < 0 \\ u_i & \text{otherwise} \end{cases} \quad \text{for } i = 1 \text{ to } n$$

where q_{ic} is the current value of the ith joint variable, and q_i^l and q_i^u are the lower and upper bounds of q_i.

B. The Search Step Size

The optimum search step size α^* (see Section IV-B) should be checked (and modified if necessary) using the following procedure.

Step 1 set counter $i = 1$

Step 2 if $q_{ic} + \alpha^* u_{ic} \geq q_i^u$ then $\alpha^* = (q_i^u - q_{ic})/u_{ic}$.
else
if $q_{ic} + \alpha^* u_{ic} \leq q_i^l$ then $\alpha^* = (q_i^l - q_{ic})/u_{ic}$

Step 3 if $i = n$ then output α^* and update the joint variables accordingly, else set $i = i + 1$ and go to step 2.

REFERENCES

[1] J. Angeles, "On the numerical solution for the inverse kinematic problem," *Int. J. Robotics Res.*, vol. 4, no. 2, pp. 21-37, 1985.

[2] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming, Theory and Algorithms.* New York: Wiley, 1979.

[3] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME J. Appl. Mechan.*, vol. 22, no. 2, pp. 215-221, June 1955.

[4] S. Elgazzar, "Efficient kinematic transformations for the PUMA 560 robot," *IEEE J. Robotics Automat.*, vol. RA-1, no. 3, pp. 142-151, Sept. 1985.

[5] R. G. Fenton, B. Benhabib, and A. A. Goldenberg, "Optimal point-to-point motion control of robots with redundant degrees of freedom," *ASME J. Eng. Industry,* vol. 108, pp. 120-126, May 1986.

[6] A. A. Goldenberg and D. L. Lawrence, "A generalized solution to the inverse kinematics of robotic manipulators," *ASME J. Dynamic Syst., Measure., Control,* vol. 107, pp. 103-106, Mar. 1985.

[7] A. A. Goldenberg, J. A. Apkarian, and H. W. Smith "A new approach to kinematic control of robot manipulators," *ASME J. Dynamic Syst., Measure., Control,* vol. 109, pp. 97-103, June 1987.

[8] K. C. Gupta and K. Kazerounian, "Improved numerical solutions of inverse kinematics of robots," presented at Int. Conf. Robotics Automat., St. Louis, MO, Mar. 1985.

[9] A. S. Hall, Jr., R. R. Root, and E. Sandgren, "A dependable method for solving matrix loop equations for the general three-dimensional mechanism," *ASME J. Eng. Industry,* pp. 547-550, Aug. 1977.

[10] J. M. Hollerback and G. Sahar, "Wrist-partitioned, inverse kinematic accelerations and manipulator dynamics," *Int. J. Robotics Res.*, vol. 2, no. 4, pp. 61-76, 1983.

[11] R. Ibarra and N. D. Perreira, "Determination of linkage parameter and pair variable errors in open chain kinematic linkages using a minimal set of pose measurement data," *ASME J. Mechanisms, Transmissions, Automat. Design,* vol. 108, pp. 159-166, June 1986.

[12] K. Kazerounian, "On the numerical inverse kinematics of robotic manipulators," *ASME J. Mechanisms, Transmissions, Automat. Design,* vol. 109, pp. 8-13, Mar. 1987.

[13] D. G. Luenberger, *Linear and Nonlinear Programming.* Reading, MA: Addison-Wesley, 1984.

[14] S. Mahalingam and A. M. Sharan, "The nonlinear displacement analysis of robotic manipulators using the complex optimization method," *Mechanism Machine Theory,* vol. 22, no. 1, pp. 89-95, 1987.

[15] G. V. Reklaitis *et al., Engineering Optimization Methods and Applications.* New York: Wiley, 1983.

[16] R. M. Rosenberg, *Analytical Dynamics of Discrete Systems.* New York: Plenum, 1980.

[17] I. H. Shames, *Engineering Mechanics, Vol. II, Dynamics.* Englewood Cliffs, NJ: Prentice-Hall, 1966, p. 738.

[18] L. W. Tsai and A. P. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods," *ASME J. Mechanisms, Transmissions, Automat. Design,* vol. 107, pp. 189-200, June 1985.

[19] Y. T. Tsai and E. D. Orin, "A strictly convergent real-time solution for inverse kinematics of robot manipulators," *J. Robotic System.*, vol. 4, no. 4, pp. 477-501, 1987.

[20] M. Tucker and N. D. Perreira, "Generalized inverses for robotic manipulators," *Mechanism and Machine Theory,* vol. 22, no. 6, pp. 507-514, 1987.

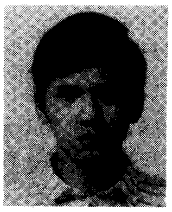
[21] J. J. Uicker, Jr., J. Denavit, and R. S. Hartenberg, "An iterative method for the displacement analysis of spatial mechanisms," *ASME J. Appl. Mechanics,* vol. 107, pp. 189-200, 1954.

[22] L. T. Wang and D. Kohli, "Closed and expanded form of manipulator dynamics using lagrangian approach," *ASME J. Mechanisms, Transmissions, Automat. Design,* vol. 107, no. 2, pp. 223-225, June 1985.

[23] L. T. Wang and B. Ravani, "Recursive computations of kinematic and dynamic equations for mechanical manipulators," *IEEE J. Robotics Automat.*, vol. RA-1, no. 3, pp. 124-131, Sept. 1985.

Li-Chun Tommy Wang received the B.E. degree in 1978 from Chung Yuan University, Chungli, Taiwan, Republic of China, the M.S. degree in 1983 from the University of Wisconsin, Milwaukee, and the Ph.D. degree in 1986 from the University of Wisconsin, Madison, all in mechanical engineering.

From 1986 to 1987, he was a Visiting Associate Professor with the Department of Mechanical Engineering at National Sun Yat-Sen University, Kaohsiung, Taiwan. In 1987, he joined the Department of Mechanical Engineering and Technology, National Taiwan Institute of Technology, Taipei, Taiwan, where he is currently an Associate Professor. His research interests include robotics, the computational kinematics and dynamics of multibody systems, design automation, and computer graphics and animation.



Chih Cheng Chen received the B.S. degree in 1985 from National Cheng-Kung University, Tainon, Taiwan, and the M.S. degree in 1987 from National Taiwan University, Taipei, Taiwan, both in mechanical engineering. He is currently working toward the Ph.D. degree in mechanical engineering at the National Taiwan Institute of Technology.

His research interests include robotics and the kinematics and dynamics of mechanisms.