INVERSE KINEMATIC ALGORITHMS FOR REDUNDANT SYSTEMS

H. Das, J-J. E. Slotine and T. B. Sheridan

Man-Machine Systems Lab., M.I.T.,
Cambridge, Mass. 02139.

Abstract

An iterative method of computing the solution of the inverse kinematic problem is developed for redundant systems using the transpose of the Jacobian matrix instead of the pseudoinverse. The solutions may be optimized on a criterion function or on physical constraints such as obstacle avoidance. Stability and convergence of the method are shown. Although its convergence rate is about twice slower than Newton's method, the advantage of the method is that it remains easily tractable close to singular configurations of the manipulator. A hybrid method combining the Jacobian transpose and Newton's methods is proposed. Results of the the application of the method on a 10 link manipulator in 2-D space are shown.

1. Introduction

The forward kinematics (i.e. the determination of the end-effector position from known joint positions) is simple for most manipulators and is of the form

$$\mathbf{x} = f(\mathbf{q}) \tag{1.1}$$

where $\mathbf{q}$ is a vector of joint positions and $\mathbf{x}$ is a vector of end-effector coordinates.

However, it is the inverse of (1.1) that is of more practical use because usually the desired end-effector position is known and the corresponding joint positions have to be found to attain the desired end-effector position. An additional problem is presented by a class of systems called *kinematically redundant systems*. These systems have more degrees of freedom than necessary to satisfy the end-effector position specification. For example, positioning an end-effector in three dimensional space requires six degrees of freedom. A manipulator with more than six degrees of freedom is kinematically redundant. Such a manipulator will have an infinite number of configurations that satisfy the end-effector position constraint. The problem associated with redundant systems is to pick a solution that is in some sense the best.

The solution of this combined problem can be formulated as a *constrained optimization* i.e. optimizing on some function while satisfying the desired end-effector position constraint. A number of mathematical methods have been derived[1] [2] to solve the problem although these methods have not seen much use in the solution of the inverse kinematic problem. Chang[3] also finds the constrained optimum using Lagrange multipliers to generate additional equations so as to fully constrain the problem, and a Newton iteration to solve the set of equations.

Alternatives to the iterative solution of the problem have been pursued by researchers. For simple manipulator systems, closed-form solutions may be found analytically.[4] Resolved motion rate control is a another method, that solves the linearized kinematic equations for velocity instead of position.[5] [6] The resulting joint velocity may be integrated to obtain position. Accuracy is sacrificed for speed with this method. The approach of using control algorithms, such as impedance control, that avoid explicit inverse kinematics and perform control, is yet another solution method.[7] [8]

In this study, an iterative algorithm suggested by Asada and Slotine[9] and Slotine and Yoerger[10] [11] is developed. The scheme also presents similarities with those by Wolovich and Elliott[12] and Sciavicco and Siciliano.[13] The above are all closely related to the method of steepest descent found in the solution of optimization problems.[1] The approach being presented here formulates the problem as a constrained optimization and uses a method of steepest descent to solve it. The method that has been developed is based on optimizing on physical parameters such as obstacle avoidance and a prefered manipulator configuration although optimizing on a criterion function is also possible. An interactive program has been written using the scheme that allows the operator to obtain solutions of joint position trajectories while being able to change optimization and constraint parameters on-line in the numerical computation.

Section 2 presents a description of the numerical

algorithm, refered to here as the *Jacobian transpose method*, and some of the issues to be considered are presented in the following section. The method is compared to Newton's method and results of the algorithm applied to a 10 link manipulator in 2 D space are shown. Future work using the algorithm in a trajectory planning program is discussed in section 3.

2. Stability Proof for the Jacobian Transpose Inverse Kinematic Algorithm

The *Jacobian transpose* method of solving the inverse kinematics can be represented as a control problem with no uncertainty.[10] Since the forward kinematics is trivial, it is used with a control algorithm to solve the inverse kinematics. The block diagram below is a physical analogy of the method showing the basic form, without optimization.

The parameters in the diagram are:

- $\mathbf{x}_{des}$ the desired end-effector position,
- K_p a end point stiffness matrix,
- $\mathbf{F}$ a force at the tip of the end-effector,
- $\mathbf{J}^T$ is the transpose of the Jacobian matrix,
- τ the equivalent torques at the jonts,
- I the identity matrix and
- $\dot{\mathbf{q}}$ the joint velocities,

Physically, the idea can be thought of as applying a force to the end of a kinematically equivalent but dynamically simple manipulator in the direction of the desired position of the end-effector. The magnitude of the force is proportional to the distance between the desired and the actual end position. When the end-effector is finally at the desired end position, the joints are measured to obtain the solution of the inverse kinematic problem. For a kinematically redundant system, other forces and torques can be applied to obtain an "optimal" solution.

2.1 Continuous Time Stability Proof

The following analysis proves stability of the method in continuous time.[10] [11] The method could be made to optimize on a criterion function. The update law in such a case is

$$\dot{\mathbf{q}} = -\alpha \mathbf{S}\mathbf{J}^T K_p \tilde{\mathbf{x}} - \beta \{I - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\}\nabla \mathbf{H}(\mathbf{q}) \qquad (2.1)$$

where

- $\mathbf{H}(\mathbf{q})$ is the function to be minimized.
- $\mathbf{S}$ is a positive definite symmetric matrix determined from the discrete time analysis,
- $\tilde{\mathbf{x}}$ is equal to $\mathbf{x} - \mathbf{x}_{desired}$
- α and β are positive scalars determined from the discrete time analysis,

The first term $-\alpha \mathbf{S}\mathbf{J}^T K_p \tilde{\mathbf{x}}$ in the update law is analogous to a force at the tip of the simulated system proportional to the distance of the tip from the desired position of the tip. Thus, the force drives the manipulator tip to the desired end position. The second term optimizes on the criterion function in the null space of the Jacobian matrix. Alternatively, obtaining the desired end position, obstacle avoidance along with physically shaping the system, is shown below. The update law chosen to iteratively determine the joint positions that result in the desired end position is

$$\dot{\mathbf{q}} = -\alpha \mathbf{S}\mathbf{J}^T K_p \tilde{\mathbf{x}} - \beta \{I - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\} \{ K\tilde{\mathbf{q}} + K_{obs} \sum_{j=1}^{j=\eta} \mathbf{J}_j^T \sum_{i=1}^{i=\kappa} \left(\frac{\tilde{\mathbf{x}}_{ij}}{\tilde{\mathbf{x}}_{ij}^T \tilde{\mathbf{x}}_{ij}}\right)\} \qquad (2.2)$$

where

- K is the joint stiffness matrix,
- $\tilde{\mathbf{q}}$ is equal to $\mathbf{q} - \mathbf{q}_{desired}$,
- K_{obs} is a repulsion factor of joints from obstacles,
- $\mathbf{J}_j$ is the Jacobian matrix of the manipulator at joint j,
- $\tilde{\mathbf{x}}_{ij}$ is the vector from obstacle i to joint j,
- κ, η are respectively the number of obstacles and joints.

The second term in the update, $-\beta \{\cdots\}\{\cdots\}$, optimizes on other constraints. $K\tilde{\mathbf{q}}$ can be seen as the application of torques at the manipulator joints to attain a desired angle. The K_{obs} $\cdots$ term is equivalent to the application of forces on each joint from each obstacle. $\{I - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\}$ projects the vector following it on to the nullspace of the manipulator Jacobian matrix. An alternative method of projection using singular value decomposition is possible when the manipulator is in a singular configuration. With this update law, optimization is performed without affecting the motion of the end of the manipulator.

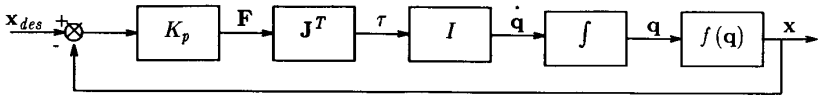

Figure 1.1 Block Diagram of the Jacobian Transpose Method

Stability for the first case is shown using a Lyapunov stability proof. The proof also applies to the second optimizing function. Let us pick a candidate Lyapunov function

$$\mathbf{V} = \frac{1}{2}\tilde{\mathbf{x}}^T K_p \tilde{\mathbf{x}} + \mathbf{V}_2 \tag{2.3}$$

where

$$\mathbf{V}_2 = \frac{1}{2}\beta^2 \dot{\mathbf{q}}^T K \dot{\mathbf{q}}. \tag{2.4}$$

$\mathbf{V}_2$ is included to ensure that the system does not go unstable in the nullspace motion. It must be shown that:

1) $\dot{\mathbf{V}}$ is negative definite, i.e. the tip always moves to the desired end position and

2) at steady state, the tip is at the desired end position.

If K_p is chosen to be diagonal and if $\mathbf{x}_{desired}$ is constant,

$$\dot{\mathbf{V}} = \tilde{\mathbf{x}}^T K_p \dot{\mathbf{x}} + \dot{\mathbf{V}}_2 \tag{2.5}$$

Using the relation $\dot{\mathbf{x}} = \mathbf{J}\,\dot{\mathbf{q}}$ to replace $\dot{\mathbf{x}}$, and substituting in the update law for $\mathbf{q}$,

$$\dot{\mathbf{V}} = \tilde{\mathbf{x}}^T K_p \mathbf{J} \{ -\alpha \mathbf{S}\mathbf{J}^T K_p \tilde{\mathbf{x}} - \beta \{I - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\} \nabla \mathbf{H}(\mathbf{q}) \} + \dot{\mathbf{V}}_2 \tag{2.6}$$

Since K_p is symmetric and the optimizing part of the control is in the nullspace of the Jacobian matrix,

$$\dot{\mathbf{V}} = -\alpha \tilde{\mathbf{x}}^T K_p^T \mathbf{J}\mathbf{S}\mathbf{J}^T K_p \tilde{\mathbf{x}} + \dot{\mathbf{V}}_2 \tag{2.7}$$

The algorithm is stable and the end point will be at the desired position at steady state if β is chosen such that $\dot{\mathbf{V}}_2$ is always negative. It is theoretically possible for the system to get stuck in some singular position where the force applied at the end tries to move the "system" in a direction in which it cannot move. In practice, torques and forces acting on the "system" for shaping or obstacle avoidance will move the "system" out of such configurations.

2.2 Discrete Time Stability Proof

The algorithm to solve the inverse kinematics is implemented by nature on a digital computer. Performance of the algorithm may therefore be improved by studying the discrete system directly. The same update law is used:

$$\Delta \mathbf{q}_k = -\alpha_k \Delta t \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k - \beta_k \Delta t \{I - \mathbf{J}_k^T(\mathbf{J}_k \mathbf{J}_k^T)^{-1}\mathbf{J}_k\} \nabla \mathbf{H}(\mathbf{q}_k) \tag{2.8}$$

where the subscript k denotes the time step of length Δt.

The stability analysis is repeated with the previous Lyapunov function,

$$\mathbf{V}_k = \frac{1}{2}\tilde{\mathbf{x}}_k^T K_p \tilde{\mathbf{x}}_k + \frac{1}{2}\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k \tag{2.9}$$

Instead of taking the time derative of the Lyapunov function, the difference between successive time steps is found. For stability, the function should always be decreasing. Intuitively, the problem of a discrete implementation can be seen to be the trade-off between a fast convergence to the solution and the resulting numerical instability.

The difference $\mathbf{V}_{k+1} - \mathbf{V}_k$ can always be made less than zero by suitably picking α, β and $\mathbf{S}$. These parameters are obtained from a discrete time stability analysis to enforce stability and improve the rate of convergence. The calculation of α and β is detailed in Appendix A while the determination of $\mathbf{S}$ is discussed in Appendix B. At steady state, the end point should be at the desired position. The joint angles at steady-state are the solution of the inverse kinematics.

The Jacobian transpose inverse kinematic method is general and applies to 3 dimensional space without modification. The only issue is handling orientation in 3-D. Yoerger and Slotine [11] discuss the problem and how it may be solved.

2.3 Comparison with Newton's Method

A Newton's iteration to solve the same problem has the update

$$\Delta \mathbf{q}_k = -\Delta t \mathbf{J}_k^{\dagger} \tilde{\mathbf{x}}_k - \Delta t \{I - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\} \nabla \mathbf{H}(\mathbf{q}). \tag{2.10}$$

where $\mathbf{J}_k^{\dagger}$ is the pseudoinverse of the Jacobian matrix of the manipulator.

Locally, the Jacobian transpose and Newton's methods produce the same results as does Chang's method.[14] The Jacobian transpose method with variable step length for maximum rate of convergence to the solution is exactly equivalent to the Newton's method for a one degree of freedom system. With multiple degrees of freedom, Newton's method has a convergence rate about twice as fast as the Jacobian transpose method. The reason is that although the *Jacobian transpose* method requires less computation per iteration step, the method takes more iterations to converge to a solution for most manipulator configurations. However, Newton's method is not suitable when the manipulator is close to singular configurations. A hybrid method combining Newton's method and the Jacobian transpose method is therefore suggested as the algorithm to be used in the solution of the inverse kinematics.

2.4 Results

The Jacobian transpose method has been implemented on the IRIS 2400 Graphics Workstation for a few example redundant systems. The sequence of diagrams below show the graphics output for a ten link manipulator in two dimensional space moving through an obstacle field. The solutions were made to optimize on obstacle avoidance and mantaining zero joint angles (i.e. keeping the manipulator as close to a straight line as possible) while following the desired end-effector trajectory entered by the operator.

3. Conclusions

It is intended, in the study of this inverse kinematic method, to use the algorithm as an aid to a human operator in the trajectory planning stage of manipulator control. The algorithm may, in simple task situations, be used as an on-line inverse kinematic procedure, the output of which can be

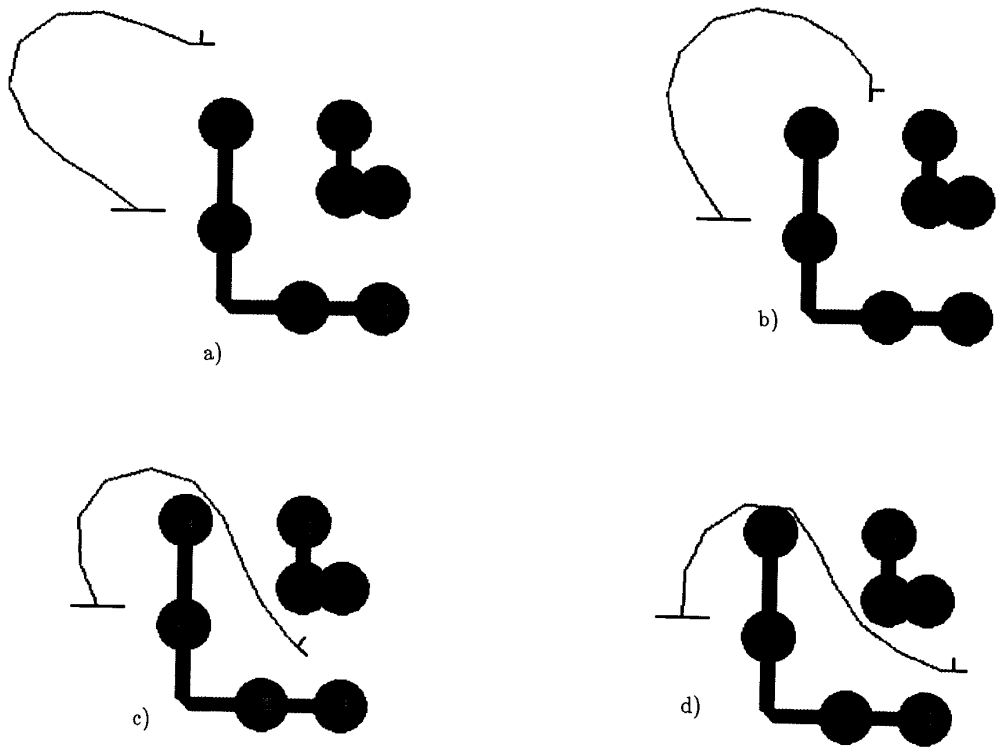

Figure 2.2. Solutions for a 10 link Manipulator in 2-D Moving through an Obstacle Field.

directly entered as joint commands to the controllers. Furthermore, a human aided iterative process using this algorithm on a computer with graphic display of the results is proposed for planning the trajectories of manoeuvering the manipulator through a complex obstacle field. The operator may attempt a variety of end-effector paths to achieve a task before committing to a particular trajectory. The aim is to have the computer perform the computation intensive solution of the inverse kinematics while the human operator makes higher level decisions on end-effector trajectory, specifing prefered joint positions and obstacle positions.

Features to be included in the trajectory planning program are the ability of the operator to make changes in obstacle positions, shapes and sizes, and to specify desired joint positions on-line. The end-effector can be guided through the desired path with a joystick while the algorithm automatically computes the joint positions required to attain the end-effector position and displays the manipulator on the graphics output.

Appendix A

The conditions on stability for a discrete implementation of the method are derived here. The proof is similar to that for the continuous time case in Section 2.1. The update law is

$$\dot{\mathbf{q}}_k = -\alpha_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k - \beta_k \{I - \mathbf{J}_k^T (\mathbf{J}_k \mathbf{J}_k^T)^{-1} \mathbf{J}_k\} \nabla \mathbf{H}(\mathbf{q}). \quad \text{A.1}$$

The Lyapunov function chosen is

$$\mathbf{V}_k = \frac{1}{2} \tilde{\mathbf{x}}_k^T K_p \tilde{\mathbf{x}}_k + \frac{1}{2} \beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.2}$$

Now, for stability, $\mathbf{V}$ must decrease with time i.e. $\mathbf{V}_{k+1}$ must be less than $\mathbf{V}_k$.

$$\mathbf{V}_{k+1} - \mathbf{V}_k = \frac{1}{2} \tilde{\mathbf{x}}_{k+1}^T K_p \tilde{\mathbf{x}}_{k+1} - \frac{1}{2} \tilde{\mathbf{x}}_k^T K_p \tilde{\mathbf{x}}_k + \frac{1}{2} \beta_{k+1}^2 \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} - \frac{1}{2} \beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.3}$$

The term $\tilde{\mathbf{x}}_{k+1}$ can be represented using the Taylor's series expansion

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \Delta t \dot{\mathbf{x}}_k + \frac{\Delta t^2}{2} \ddot{\mathbf{x}}_k + \cdots \quad \text{A.4}$$

The matrix $\mathbf{S}$, determined in Appendix B, can be used

to make the terms in Δt^2 and above small. Ignoring those terms,

$$\mathbf{V}_{k+1} - \mathbf{V}_k = \Delta t \tilde{\mathbf{x}}_k^T K_p \dot{\mathbf{x}}_k + \frac{1}{2}\Delta t^2 \dot{\mathbf{x}}_k^T K_p \dot{\mathbf{x}}_k + \frac{1}{2}\beta_{k+1}^2 \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} - \frac{1}{2}\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.5}$$

Now, substituting the relation,

$$\dot{\mathbf{x}}_k = \mathbf{J}_k \dot{\mathbf{q}}_k \quad \text{A.6}$$

in the above gives

$$\mathbf{V}_{k+1} - \mathbf{V}_k = \Delta t \tilde{\mathbf{x}}_k^T K_p \mathbf{J}_k \dot{\mathbf{q}}_k + \frac{1}{2}\Delta t^2 \dot{\mathbf{q}}_k^T \mathbf{J}_k^T K_p \mathbf{J}_k \dot{\mathbf{q}}_k + \frac{1}{2}\beta_{k+1}^2 \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} - \frac{1}{2}\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.7}$$

Note that the optimizing vector is in the nullspace of $\mathbf{J}_k$. So,

$$\mathbf{J}_k \dot{\mathbf{q}}_k = -\alpha \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k - \beta_k \mathbf{J}_k \{I - \mathbf{J}_k^T (\mathbf{J}_k \mathbf{J}_k^T)^{-1} \mathbf{J}_k\} \nabla \mathbf{H}(\mathbf{q}) = -\alpha \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k. \quad \text{A.8}$$

Using the above,

$$\mathbf{V}_{k+1} - \mathbf{V}_k = -\alpha_k \Delta t \tilde{\mathbf{x}}_k^T K_p \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k + \frac{1}{2}\alpha_k^2 \Delta t^2 \tilde{\mathbf{x}}_k^T K_p^T \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_p + \frac{1}{2}\beta_{k+1}^2 \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} - \frac{1}{2}\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.9}$$

Let

$$N = \Delta t \tilde{\mathbf{x}}_k^T K_p \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k \quad \text{A.10}$$

and

$$D = \frac{1}{2}\Delta t^2 \tilde{\mathbf{x}}_k^T K_p \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T K_p \tilde{\mathbf{x}}_k. \quad \text{A.11}$$

Then

$$\mathbf{V}_{k+1} - \mathbf{V}_k = -\alpha_k N + \alpha_k^2 D + \frac{1}{2}\beta_{k+1}^2 \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} - \frac{1}{2}\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k. \quad \text{A.12}$$

Let $\mathbf{C} = -\alpha_k N + \alpha_k^2 D$. This quadratic equation in α_k has a minimum and always negative value of $\mathbf{C}$ when $\alpha = \dfrac{N}{2D}$. The sum of the α terms in (A.12) have been made negative. The sum of the β terms can also be made negative by choosing β_{k+1} such that

$$\beta_{k+1} = \frac{\beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k}{\dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1}} \quad \text{if } \dot{\mathbf{q}}_{k+1}^T \dot{\mathbf{q}}_{k+1} \text{ is greater than } \beta_k^2 \dot{\mathbf{q}}_k^T \dot{\mathbf{q}}_k$$

$$= 1 \quad \text{otherwise}$$

to force the stability condition on the last two terms in (A.12) by making the first term always smaller than the second.

Appendix B

In the derivation in Appendix A, higher order terms in the expansion of $\tilde{\mathbf{x}}_{k+1}$ were assumed to be negligible. In this appendix, a matrix, $\mathbf{S}$, is derived to force that condition. The derivation is possible for two reasons. The first is that Euler integration is used in the algorithm. The second is the special nature of the Jacobian matrix. The Taylor series expansion of $\tilde{\mathbf{x}}_{k+1}$ is

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \Delta t \dot{\tilde{\mathbf{x}}} + \frac{1}{2}\Delta t^2 \ddot{\tilde{\mathbf{x}}} + \cdots \quad \text{B.1}$$

The derivatives of $\tilde{\mathbf{x}}_k$ may be replaced by,

$$\dot{\tilde{\mathbf{x}}}_k = \mathbf{J}_k \dot{\mathbf{q}}_k \quad \text{B.2}$$

$$\ddot{\tilde{\mathbf{x}}}_k = \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k + \mathbf{J}_k \ddot{\mathbf{q}}_k \quad \text{B.3}$$

and so on. In the discrete implementation, Euler integration is used i.e.

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t \dot{\mathbf{q}}_k \quad \text{B.4}$$

so $\mathbf{q}_k$ and its higher derivatives between time steps have been set to zero implicitly. Thus,

$$\ddot{\tilde{\mathbf{x}}}_k = \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k \quad \text{B.5}$$

$$\dddot{\tilde{\mathbf{x}}}_k = \ddot{\mathbf{J}}_k \dot{\mathbf{q}}_k \quad \text{B.6}$$

and so on.

From here on an order of magnitude argument is used. For pure prismatic type manipulator joints, the Jacobian matrix terms are independent of the joint positions so that the derivatives of $\mathbf{J}$ are zero. The situation is more involved for a manipulator with revolute type joints. Let us first define σ_{Jmax} to be the maximum singular value of the Jacobian matrix for all posible values of $\mathbf{q}$. Since the terms in the Jacobian matrix are sums of sines and cosines of the joint positions, the time derivative of the Jacobian matrix are the sums of sines and cosines multiplied by the joint velocities. Thus, the order of magnitude of $\dot{\mathbf{J}}_k \dot{\mathbf{q}}_k$ can be represented by $\sigma_{Jmax} \|\dot{\mathbf{q}}_k\| \mathbf{q}_k$. Or, even more conservatively,

$$\ddot{x}_{j,k} = \sigma_{Jmax} \|\dot{\mathbf{q}}_k\|^2 \quad \text{B.7}$$

where $\ddot{x}_{j,k}$ is the jth element of $\ddot{\mathbf{x}}_k$. Similarly,

$$\dddot{x}_{j,k} = \sigma_{Jmax} \|\dot{\mathbf{q}}_k\|^3 \quad \text{B.8}$$

and so on.

The expansion of the higher order terms may be rewritten as

$$order(\frac{1}{2}\Delta t^2 \ddot{x}_{j,k} + \cdots) = order \sigma_{Jmax}(\frac{1}{2}\Delta t^2 \|\dot{\mathbf{q}}_k\|^2 + \frac{1}{6}\Delta t^3 \|\dot{\mathbf{q}}_k\|^3 + \cdots). \quad \text{B.9}$$

Letting Q_k equal $\Delta t \|\dot{\mathbf{q}}_k\|$,

$$order(\frac{1}{2}\Delta t^2 \ddot{x}_{j,k} + \cdots) = order\ \sigma_{Jmax}\ (e^{Q_k} - 1 - Q_k). \quad \text{B.10}$$

Notice that the sum of the higher order terms of $\tilde{\mathbf{x}}$ will be small if Q is small. Note also that while the terms in $\dot{\mathbf{x}}$ are linearly dependent on Q, the sum of the higher order terms are exponentially related to Q indicating that by keeping Q small, the higher order terms can be make negligible. A limit on the maximum value of Q bounds the sum of all the higher order terms of the expansion. This may be achieved, for example, by selecting $\mathbf{S}_k$ such that whenever Q exceeds a limiting value,

$$\mathbf{S}_k = \begin{bmatrix} 1 & & & & \\ & 1 & & 0 & \\ & & \ddots & & \\ & & & \frac{1}{Q_k} & \\ & & 0 & & \frac{1}{Q_k} \end{bmatrix} \qquad \text{B.11}$$

where the diagonal element is unity corresponding to prismatic joints and $\frac{1}{Q_k}$ for revolute joints. $\mathbf{S}_k$ is set to the identity matrix when the limit is not exceeded. The matrix $\mathbf{S}_k$ insures that the revolute joint velocities are kept small. The effect of using such a **S** matrix is that when the desired end effector position is distant from the actual end effector position, the prismatic joints will respond to reduce the difference. For a vehicle-manipulator system, the vehicle will approach the target before the hand tries to move to the desired position.

REFERENCES

1. **Scales, L. E.,** 'Introduction to Non-Linear Optimization', Springer-Verlag, New York, 1985.
2. **Fletcher, R.,** 'Generalized Inverses for Nonlinear Equations and Optimization' in 'Numerical Methods for Nonlinear Algebraic Equations', ed. P. Rabinowitz, Gordon and Breach Science Publishers, London, 1970.
3. **Chang, P. H.,** 'A Closed Form Solution for Inverse Kinematics of Redundant Manipulators', Proc. of the IEEE Int'l Conf. on Robotics and Automation, San Francisco, CA, Apr. 1986.
4. **Featherstone, R.,** 'Position and Velocity Transformations between Robot End-Effector Coordinates and Joint Angles', Int'l J. of Robotics Research, Vol. 2, #2, 1983.
5. **Whitney, D. E.,** 'Resolved Motion Rate Control of Manipulators and Human Prostheses', IEEE Trans on Man-Machine Systems, Vol MMS-10, #2, June 1969.
6. **Klein, C. A. and C.-H. Huang,** 'Review of Pseudo-inverse Control for use with Kinematically Redundant Manipulators', IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-13, #3, Mar./Apr. 1983.
7. **Khatib, O.,** 'Real-Time Obstacle Avoidance for Manipulators and Mobile Robots', Int'l J. of Robotics Research, Vol. 5, #1, Spring 1986.
8. **Hogan, N.,** 'Impedance Control: An Approach to Manipulation Parts 1, 2, 3', J. of Systems, Measurement and Control, Vol 107 Mar. 1985.
9. **Asada, H. and J-J. E. Slotine,** 'Robot Analysis and Control', J. Wiley and Sons, 1986.
10. **Slotine, J-J. E. and D. R. Yoerger,** 'A Rule Based Inverse Kinematics Algorithm for Redundant Systems', Int'l J. of Robotics and Automation, 2 (3), 1987.
11. **Yoerger D. R. and J-J. E. Slotine,** 'Task Resolved Motion Control of Vehicle-Manipulator Systems', Int. J. Robotics and Automation, 2 (3), 1987.
12. **Wolovich, W. A. and H. Elliott,** 'A Computational Technique for Inverse Kinematics', Proc. 23rd Conf. on Decision and Control, Las Vegas, NV, 1984.
13. **Sciavicco L. and B. Siciliano,** 'A Dynamic Solution to the Inverse Kinematic Problem of Redundant Manipulators', IEEE Int'l Conf. on Robotics and Automation, Mar 31- Apr 3, 1987, Raleigh, NC.
14. **Chang, P. H.,** 'Analysis and Control of Robot manipulators with Kinematic Redundancy', Ph. D. thesis, MIT, June, 1987.