FAST PARALLEL COMPUTATION OF REDUNDANCY

Daniel Boley
Department of Computer Science
University of Minnesota

ABSTRACT

For redundant manipulators, the inverse kinematics equations are typically solved by taking the pseudo-inverse of the Jacobian, or by adding enough constraints to algebraically remove the redundancy. Many popular measures of redundancy also depend on computing the algebraic rank or determinant of the Jacobian. We present fast, parallelizable algorithms to dynamically update all the pseudo-inverse, determinant, rank, and many other redundancy measures and properties as the arm goes through its motions.

INTRODUCTION

When designing a task for a manipulator, the fundamental kinematic equation is the relation between the position and/or orientation of the end effector and the joint angles, according to the equation

$$[x, y, z, \alpha, \beta, \gamma]^{\mathrm{T}} \equiv \mathbf{r} = \mathbf{F}(\boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}$ is the vector of joint angles, or more generally a point in *configuration space*, and $\mathbf{r}$ is the 6-vector of end-effector position/orientation in *cartesian space*. The inverse kinematic problem is to solve for the joint angles needed to achieve a given position/orientation of the end-effector. The solution of the nonlinear system of equations (1) by Newton's method, a simple and effective method to use, requires the use of the Jacobian $J(\boldsymbol{\theta})$ of F. Then each Newton iteration is computed by solving the set of linear equations for the correction $\boldsymbol{\theta}_{\text{new}}-\boldsymbol{\theta}_{\text{old}}$:

$$J(\boldsymbol{\theta}_{\text{old}}) \cdot (\boldsymbol{\theta}_{\text{new}}-\boldsymbol{\theta}_{\text{old}}) = F(\boldsymbol{\theta}_{\text{old}}). \tag{2}$$

When the manipulator is following a path in cartesian space, $\mathbf{r}$ and $\boldsymbol{\theta}$ become functions of time t. It is then necessary to control the joint *velocities* $\dot{\boldsymbol{\theta}}$, related by

$$\dot{\mathbf{r}} = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}. \tag{3}$$

Both (2) and (3) require the ''inversion'' of the Jacobian, or rather the solution of a set of linear equations whose matrix of coefficients is J.

The theory described by (2) or (3) breaks down when (a) the Jacobian is singular (i.e. the manipulator is in a singular configuration), or (b) the Jacobian is not square (i.e. the number of joints does not match the degrees of freedom in cartesian space). For the latter case, if there are too many joints, the robot is said to be *redundant*, and then equations (2) or (3) will have multiple solutions. The problem is then to select among all the solutions that particular solution that meets some secondary criterion, either a constraint on some of the variables or optimality of some performance index. All these involve the solution of systems of linear equations which are overdetermined, underdetermined, or singular. The object of this paper is to explore the efficient algorithms currently in use for such problems.

For example, for a redundant manipulator, the kinematic mapping of joint angles to cartesian coordinates will have a non-square Jacobian with more columns than rows. One method to invert the kinematic mapping that has been suggested is the use of the generalized (Moore-Penrose) inverse (see, e.g. [3, p243]), computed explicitly. The Moore-Penrose inverse is a matrix J is defined as the unique matrix $J^{\#}$ which satisfies the four properties

$$\text{(i) } JJ^{\#}J = J \quad \text{(ii) } J^{\#}JJ^{\#} = J^{\#} \quad \text{(iii) } [JJ^{\#}]^{\mathrm{T}} = JJ^{\#} \quad \text{(iv) } [J^{\#}J]^{\mathrm{T}} = J^{\#}J.$$

The Moore-Penrose inverse is used to parametrize all solutions to the joint equations (3) by

$$\dot{\boldsymbol{\theta}} = J^{\#}\dot{\mathbf{r}} + (I - J^{\#}J)\mathbf{y} \tag{4}$$

where the first term $J^{\#}\dot{\mathbf{r}}$ specifies the satisfaction of the primary objective (3) and the second term represents all the excess degrees of freedom. The free vector $\mathbf{y}$ is then determined to satisfy a secondary objective, which may be a constraint on some internal joints or may be to minimize a cost functional representing joint velocities or distance to obstacles. One main purpose of this paper is to evaluate different methods to solve (4).

COMPUTATIONAL MATRIX METHODS

As noted in [3, p230f], there are methods for solving a rectangular system of equations that are more accurate and faster than methods based on computing $J^{\#}$ explicitly. Typically, all these methods proceed by finding a vector $\mathbf{x}$ that minimizes $\|A\mathbf{x}-\mathbf{b}\|_2$, and selecting the (almost) smallest norm solution in case there is a choice. Two such methods are the Orthogonal (QR) Decomposition and the Singular Value Decomposition (SVD). The SVD of am $n\times m$ matrix A is $A = U\Sigma V^{\mathrm{T}}$, where U,V are orthogonal matrices of appropriate dimension, and Σ is a $n\times m$ diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, where $p = \min\{n,m\}$. The use of the SVD in the analysis of robotic manipulators has been already investigated, but it has not been very popular because of its reputation for being expensive.

Another decomposition, which is computationally simpler, is the QR Decomposition. If A is a full rank matrix, then the QR Decomposition, also known as an orthogonal triangularization procedure, reduces A to upper triangular form by a series of orthogonal transformations such as Householder transformations: $I-2\mathbf{u}\mathbf{u}^{\mathrm{T}}$ where $\mathbf{u}$ is a unit vector [3]. The result is the matrix factorization $A = QR$, where R is upper triangular and Q is the orthogonal matrix consisting of the accumulation of all the individual Householder transformations. If A is rectangular with more rows than columns, the factorization will have the form

$$A = QR = [Q_1,Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1R_1, \tag{5}$$

where R_1 is square, upper triangular, and has the same rank as A.

In the case that the matrix A is singular or rank deficient, then the QR factorization is not sufficient to solve the least squares problem. However, the QR factorization can be extended to include column pivoting at little extra cost [3, p235]. The resulting factorization is the simplest example of a so-called *rank-revealing* factorization, and has the form

$$A = QRP^{\mathrm{T}} = [Q_1,Q_2]\begin{bmatrix} R_1 & S \\ 0 & R_2 \end{bmatrix}P^{\mathrm{T}}, \tag{6}$$

where Q is an orthogonal matrix as before, P is a permutation matrix encapsulating the column interchanges, and R is an upper triangular, *graded* matrix, which means that $r_{ii}^2 \geq r_{ik}^2 + \cdots + r_{kk}^2$ for all $k>i$ and all i. Intuitively, this means that any small diagonal entries of R will be pushed to the lower right, ‘‘revealing’’ any rank deficiency. It is then possible to solve the rank deficient least squares problem

$$\min\|A\mathbf{x}-\mathbf{b}\| = \|QRP^{\mathrm{T}}\mathbf{x}-\mathbf{b}\| \tag{7}$$

by treating $R_2 \approx 0$ and using the formula [3]

$$\mathbf{x} = P\begin{bmatrix} R_1^{-1}Q^{\mathrm{T}}\mathbf{b} \\ 0 \end{bmatrix}. \tag{8}$$

By allowing P to be a general orthogonal matrix, we may further reduce (6) to the *complete orthogonal decomposition* (COD):

$$A = QRP^{\mathrm{T}} = [Q_1,Q_2]\begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}\begin{bmatrix} P_1^{\mathrm{T}} \\ P_2^{\mathrm{T}} \end{bmatrix} \approx Q_1R_1P_1^{\mathrm{T}}, \tag{9}$$

where R_1 is square and invertible and $\|R_2\|$ is small enough to be treated as zero. With this decomposition, the solution (8) is then the smallest norm solution to (7). We note that the SVD is a special case of the COD where R_1, R_2 are diagonal, but obtained at considerable extra expense. This will be explored further below. The Moore-Penrose inverse can also be computed from (9) (with the COD or the SVD) by

$$A^{\#} = P\begin{bmatrix} R_1^{-1} & 0 \\ 0 & 0 \end{bmatrix}Q^{\mathrm{T}} = P_1R_1^{-1}Q_1^{\mathrm{T}}.$$

For the case of redundant robots, J will be $n{\times}m$ rectangular with $n<m$. It has already been noted that the QR decomposition can be used to solve (3) when full rank [1]. Here, we outline how the COD can be used to solve the problem when rank deficient. When J has full rank, the following prescription applies in the same way, except that we can replace the COD (9) with the QR decomposition (5). We compute the COD of $A = J^{\mathrm{T}}$, obtaining the form (9). Then $J^{\#} = Q_1R_1^{-\mathrm{T}}P_1^{\mathrm{T}}$, and (4) can be written

$$\dot{\boldsymbol{\theta}} = Q_1R_1^{-\mathrm{T}}P_1^{\mathrm{T}}\dot{\mathbf{r}} + Q_2Q_2^{\mathrm{T}}\mathbf{y},$$

where we have used the facts $J^{\#}J = Q_1Q_1^{\mathrm{T}}$ and $Q_1Q_1^{\mathrm{T}} + Q_2Q_2^{\mathrm{T}} = I$. At this point, we may replace $\mathbf{y}$ with a quantity defined by our secondary objective. If the secondary objective is to minimize a potential function p, then we may simply set $\mathbf{y} = -k(\partial p/\partial\boldsymbol{\theta})^{\mathrm{T}}$ in (5) [8, p131]. If the secondary objective is the satisfaction of a second set of cartesian constraints $\dot{\mathbf{r}}^{(2)} = J^{(2)}\boldsymbol{\theta}$, then the solution is [8, p129]

$$\dot{\boldsymbol{\theta}} = J^{\#}\dot{\mathbf{r}} + \hat{J}(\dot{\mathbf{r}}^{(2)} - J^{(2)}J^{\#}\dot{\mathbf{r}}) + (I{-}J^{\#}J)(I{-}\hat{J}^{\#}\hat{J})\mathbf{z},$$

where $\hat{J} = J^{(2)}Q_1Q_1^{\mathrm{T}}$. We compute the COD of $\hat{J} = \hat{Q}_1\hat{R}_1^{-\mathrm{T}}\hat{P}_1^{\mathrm{T}}$ to obtain the solution

$$\dot{\boldsymbol{\theta}} = Q_1R_1^{-\mathrm{T}}P_1^{\mathrm{T}}\dot{\mathbf{r}} + \hat{Q}_1\hat{R}_1^{-\mathrm{T}}\hat{P}_1^{\mathrm{T}}(\dot{\mathbf{r}}^{(2)}{-}J_2Q_1R_1^{-\mathrm{T}}P_1^{\mathrm{T}}\dot{\mathbf{r}}) + Q_2Q_2^{\mathrm{T}}\hat{Q}_2\hat{Q}_2^{\mathrm{T}}\mathbf{z},$$

parametrized by the free vector **z**, representing any degrees of freedom left over.

The same techniques used to solve the non-square or singular systems can also be used to estimate many other quantities in kinematics and dynamics. For example, many of the measures of dexterity that have been proposed in the literature can be computed using the COD. Examples of such measures of dexterity that have been proposed include a measure based on the determinant of $JJ^{\#}$ [10], the trace [2], minimum singular value [6], etc.), but they all amount to various ways to determine the algebraic singularity of the matrix. Though the COD itself may be sufficient to expose the rank of the matrix, these other measures can be easily estimated from the COD, or more computed more precisely from the SVD.

Besides finding the least squares solution, many methods have been proposed based on the use of the Moore-Penrose pseudo-inverse of the Jacobian to achieve combined compliance control [9], or the nullspace of the Jacobian to find the "best" solution within the space of all solutions [5], or the projectors corresponding to the range and nullspaces [4], as examples. All of these tasks can be computed via the COD. Many of these tasks, as well as others, involve the re-computation of the Jacobian as the configuration changes, hence there is a need for efficient algorithms for the factorization of the Jacobian as well as its derived forms: pseudo-inverse, nullspace, projectors, as the case may be. It is also useful to be able to update the factorizations as the Jacobian undergoes small changes. We explore this in the next section.

A FAST PARALLEL ALGORITHM AND UPDATING PROCEDURE

The use of an efficient updating algorithm for the SVD for robotics applications was proposed in [7]. They showed that by using an adaptive iterative algorithm, based on the *Kogbetliantz method* (derived from the *Jacobi method* for the symmetric eigenproblem) one can substantially reduce the computational effort to obtain the SVD at many points along a trajectory by a factor of 5 or 6 on a serial computer.

Unlike many other methods, the Kogbetliantz method for the computation of the SVD does not require an initial reduction to any particular form. To compute the SVD of J, it applies a sequence of orthogonal transformations from the right, where each orthogonal transformation is based on a plane rotation $Q = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$, where $c^2 + s^2 = 1$. Each rotation acts on two columns of J (costing $2n$ operations), and is constructed to make those two columns mutually orthogonal. A *sweep* is complete when every column has been paired exactly once with every other column. Of course, each rotation destroys the orthogonality created by the previous rotation, but it can be shown that after several sweeps, the matrix gradually converges to $A^{(\infty)}$ with all columns mutually orthogonal. Typical numerical experience indicates that the number of sweeps needed to converge is proportional to $\log n$ for an $n \times n$ matrix. On an $n \times n$ mesh of parallel processors, one can order the rotations so that an entire sweep can be completed in $n-1$ steps [3, p450].

At convergence, we have the relation $JV = A^{(\infty)}$, where V is the accumulation of all the individual orthogonal transformations, and $A^{(\infty)}$ has mutually orthogonal columns. Then the SVD of J can be recovered by scaling the columns of $A^{(\infty)}$ to unit length, obtaining $JV = U\Sigma$ where Σ is the diagonal matrix of scale factors, which are exactly the singular values.

A robot following a path gives rise to a slowly varying Jacobian, and [7] proposed to take advantage of this fact to speed up the convergence of the Kogbetliantz method. If we have the SVD of a given $J = U\Sigma V^{\mathrm{T}}$, then we can obtain the SVD of a slightly modified $J + \Delta J$ by forming the matrix $A^{(0)} = (J + \Delta J)V$. This matrix will have columns that are almost mutually orthogonal, so often one sweep of the Kogbetliantz method will be sufficient for convergence, depending on the size of the perturbation ΔJ and the existence of a singularity.

COST COMPARISON

We compare the cost of the various methods using the following ‘‘model problem:’’

$$\min \|\dot{\boldsymbol{\theta}}\|_2^2 \quad \text{s.t.} \quad J\dot{\boldsymbol{\theta}} = \dot{\mathbf{r}}, \tag{10}$$

where J is the $p \times n$ Jacobian matrix with rank $r \le p \le n$. Intuitively, we want to find the joint velocities $\dot{\boldsymbol{\theta}}$ which achieve a given end-effector velocity $\dot{\mathbf{r}}$, and which has the smallest norm among all such solutions. Because space does not permit a fuller analysis here, we restrict our attention to the model problem and assume for simplicity that the desired end-effector velocity is feasible.

In general, the cost associated with each method is heavily dominated by the computation of the associated factorization. The basic cost in floating point operations for each associated factorization is reported in Table 1 [3, p239, 248, 450]. The extra costs for using the factorization to solve the problem are much smaller. We present three typical situations: (i) a square, nonsingular, nonredundant system, (ii) a highly redundant but full rank system, and (iii) a nonredundant but singular system. For the full rank situations (i) and (ii), the COD reduces to the QR decomposition. This table shows that the optimal method depends on the situation. For (i), any of the first three methods have comparable cost, though it is well known that the normal equations suffers in accuracy [3, p230]. For a highly redundant situation (ii), the normal equations is the cheapest, but if we are near a singularity, it tends to break down, so we must use the COD. For the singular case (iii), only the last three methods can be used, and of those three, the COD is the cheapest. There are two limitations to the COD. First, the ‘‘rank revealing nature’’ of the COD is only a heuristic which can fail. It is possible to make it more robust at some minor extra expense. Second, the SVD (either full or updated) is required for the norm constrained least squares problem [7]

$$\min \|J\dot{\boldsymbol{\theta}} - \dot{\mathbf{r}}\|_2 \quad \text{s.t.} \quad \|\dot{\boldsymbol{\theta}}\|_2 \le \dot{\theta}_{\max}.$$

Method	(i) $n = p = r$	(ii) $n \gg p = r$	(iii) $n = p \gg r$
Normal Equations	$^4/_3 n^3$	$np^2 + {}^1/_3 p^3$	(not applicable)
QR Decomposition	$^4/_3 n^3$	$2np^2 - {}^2/_3 p^3$	(not applicable)
COD	$^4/_3 n^3$	$2np^2 - {}^2/_3 p^3$	$4n^2 r - {}^8/_3 r^3$
Full SVD	$12n^3$	$2np^2 + 11p^3$	$12n^3$
Updated SVD	$4n^3$	$4np^2$	$4n^3$

Table 1. Cost to solve model problem (10) by different methods under various conditions.

CONCLUSIONS

In comparing the matrix computation algorithms applicable to kinematic problems, we have found that the choice of method depends critically on the degree of redundancy of the manipulator and how close to singularity is the configuration. One can achieve a high degree of robustness using the SVD, at a large computational cost. But the other methods proposed may achieve accuracy quite sufficient for mechanical systems.

On a parallel computing architecture with sufficient number of processors, the SVD can be reduced to $O(n)$ cost using the method outlined above. Similar reductions can be achieved for the QR decomposition, but of the methods applicable to singular problems, the SVD is the most easily parallelizable, and thus may the choice for real-time applications.

REFERENCES

[1] **K. Anderson, J. Angeles,** Kinematic Inversion of Robotic Manipulators in the Presence of Redundancies, *Intl. J. Robotics Res.* 8, pp. 80-97, 1989.

[2] **J. Baillieul,** A Constraint Oriented Approach to Inverse Problems for Kinematically Redundant Manipulators, in *Proc 1985 IEEE Intl. Conf. Robot. Automat.*, pp. 244-250, 1987.

[3] **G. H. Golub, C. Van Loan,** *Matrix Computations 2/e,* Johns Hopkins, 1989.

[4] **M. Z. Huang, K. R. Harendra Varma,** Optimal Rate Allocation in Kinematically Redundant Manipulators -- The Dual Projection Method, in *IEEE Int’l Conf. Robotics and Automation*, pp. 702-707, April 9-11, 1991, Sacramento CA, 1991.

[5] **K. R. Harendra Varma, M. Z. Huang,** A Jacobian Minor-Based Algorithm for Rate Coordination of Redundant Serial Chain Manipulators, in *IEEE/RSJ IROS Conference*, pp. 1741-1746, July 7-10, 1992, Raleigh NC, 1992.

[6] **C. A. Klein, B. E. Blaho,** Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators, *Int. J. Robotics Res.* 6, pp. 72-83, 1987.

[7] **A. A. Maciejewski, C. A. Klein,** The SVD: Computation and Application to Robotics, *Int. J. Robot. Res.* 8, pp. 73-78, 1989.

[8] **Y. Nakamura,** *Advanced Robotics, Redundancy and Optimization,* Addison Wesley, 1991.

[9] **K. Yokoi, H. Maekawa, K. Tanie,** A Method of Compliance Control for a Redundant Manipulator, in *IEEE/RSJ IROS Conference*, pp. 1927-1934, July 7-10, 1992, Raleigh NC, 1992.

[10] **T. Yoshikawa,** Analysis and Control of Robot Manipulators With Redundancy, in *Robotics Research: The First International Symposium* (M. Brady, R. Paul ed.), pp. 735-747, MIT Press, 1984.