Anthony A. Maciejewski
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Charles A. Klein
Department of Electrical Engineering
The Ohio State University
Columbus, Ohio 43210

The Singular Value Decomposition: Computation and Applications to Robotics

Abstract

The singular value decomposition has been extensively used for the analysis of the kinematic and dynamic characteristics of robotic manipulators. Due to a reputation for being numerically expensive to compute, however, it has not been used for real-time applications. This work illustrates a formulation for the singular value decomposition that takes advantage of the nature of robotics matrix calculations to obtain a computationally feasible algorithm. Several applications, including the control of redundant manipulators and the optimization of dexterity, are discussed. A detailed illustration of the use of the singular value decomposition to deal with the general problem of singularities is also presented.

1. Introduction

In recent years the singular value decomposition (SVD) has become a popular tool for analyzing the kinematic and dynamic properties of robotic manipulators (Yoshikawa 1985a; Yoshikawa 1985b). It plays a particularly prominent role with regard to redundant manipulators, both in terms of analyzing the significance of the extra degrees of freedom (Klein and Huang 1983) and in specifying a side criterion that can be optimized using these redundant degrees of freedom. In many cases, these side criteria are some quantitative measure of the qualitative concept of dexterity. Most of the dexterity measures proposed are some function of the singular values of the Jacobian matrix. The most common of these is perhaps the manipulability measure proposed by Yoshikawa (1984) that is defined as the square root of the determinant of the matrix $\boldsymbol{J}\boldsymbol{J}^T$, which is simply the product of the singular values of $\boldsymbol{J}$. Other proposed measures include the trace of the above matrix (Baillieul 1987), the minimum singular value of the Jacobian (Klein and Blaho 1987), the compatibility index (Chiu 1987), and isotropy (all equal singular values; Salisbury and Craig 1982).

While all of the above measures have a physical significance and justification for their use, the key point here is that they are all closely linked to the SVD. Yet in spite of this fact, the full decomposition is usually limited to the analysis of manipulator configurations and is not considered for implementation in on-line control. This is exemplified by the popularity of the manipulability measure, since its major justifications are that it is numerically simple to compute and that its zeros coincide with the singularities of the Jacobian. The implication is that one would really like information about singularities; however, that would require calculation of the SVD, which has a reputation for being numerically expensive to compute. Unfortunately, the determinant gives no information about the absolute proximity to singularities, since the minimum singular value is the only reliable measure of this quantity. In addition, the calculation of the matrix product $\boldsymbol{J}\boldsymbol{J}^T$ squares the condition number, which reduces the accuracy of the result.

This work is concerned with demonstrating that, with the right formulation, the SVD is computationally feasible for use in real-time control. Traditionally, the computation of the SVD of an arbitrary matrix is an iterative procedure, so that the exact number of

The International Journal of Robotics Research,
Vol. 8, No. 6, December 1989,

computations cannot be known a priori. However, the control of robotic systems is not based on the solution of arbitrary matrix equations but quite frequently involves the solution of equations based on the Jacobian matrix. The current Jacobian for a system can be regarded as a perturbation of a previously known matrix for which perturbation bounds on the singular values and singular vectors can be established. It will be shown how this knowledge of the previous state can be exploited during the current computation of the SVD in order to reduce the overall computational burden. The above point will be emphasized in this work, which illustrates a computational scheme capable of calculating the SVD of the Jacobian for use in real-time control of manipulators. The implications of such an algorithm for several applications, including the utilization of redundancy and the optimization of manipulator dexterity, are then discussed. A detailed examination of the advantages of using the SVD for dealing with manipulator singularities is illustrated through computer simulations of a PUMA robot, using both rate and acceleration control.

2. Overview of SVD Algorithms

2.1. The Golub-Reinsch Algorithm

The most popular technique for computing the SVD was first proposed in Golub and Kahan (1965) with source code published in Businger and Golub (1969). It is now commonly referred to as the Golub-Reinsch algorithm (Golub and Reinsch 1970) and is available in many linear algebra software packages such as EISPACK, LINPACK, and IMSL. This algorithm is composed of two distinct steps, namely, transforming the given matrix into bidiagonal form using a series of Householder transformations and then applying an iterative procedure designed to use orthogonal transformations to produce bidiagonal matrices that are successively more diagonal. The procedure used is a variant of the QR algorithm (Watkins 1982), with origin shifts designed to improve the convergence properties (Stewart 1970).

The Golub-Reinsch algorithm is generally regarded as the most efficient and numerically stable technique for computing the SVD of an arbitrary matrix. However, there are two aspects of the algorithm that make it less attractive for the problem at hand. The first relates to the fact that the first step in the algorithm requires a fixed computation to bidiagonalize the given matrix so that a slightly perturbed matrix must still undergo this operation. The second aspect relates to the relatively serial nature of the technique, thus making it difficult to utilize parallel computing structures. For these reasons, the Golub-Reinsch algorithm will not be considered as the basis for implementing a real-time SVD algorithm. The following algorithm, based exclusively on Givens rotations, is more suited to take advantage of incremental perturbations and parallel architectures.

2.2. Algorithms Based on Givens Rotations

Givens rotations are orthogonal transformations of the form

$$Q = \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \cos(\theta) & & & & -\sin(\theta) & & & i \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & \sin(\theta) & & & & \cos(\theta) & & & j \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \\ & & & i & & & & j & & & \end{bmatrix} \quad (1)$$

where all other elements not shown are zero. This transformation can be geometrically interpreted as a plane rotation of θ in the i-j plane. These transformations are also known as Jacobi rotations since they were first described by Jacobi (1846). In contrast to Householder reflections, Givens rotations only affect two rows or columns of the matrix with which they

are multiplied, a property that is useful when designing parallel computing structures based on this transformation. One popular application of Givens rotations is for the computation of the QR decomposition. While this requires $2mn^2 - 2n^3/3$ floating-point operations as compared to $mn^2 - n^3/3$ floating point operations for the Householder version of the QR decomposition (Golub and Van Loan 1983), the Givens rotations can be done in parallel and can therefore be computed in $O(m)$ units of time with an appropriately connected array of $O(n^2)$ processors (Luk 1986a).

The most important property of Givens rotations for the problem at hand is their ability to orthogonalize the two rows or columns on which they operate. This forms the basis of an SVD algorithm that relies solely on Givens rotations (Hestenes 1958; Nash 1975). In particular, consider an orthogonal matrix $\boldsymbol{V}$, composed of successive Givens rotations, such that

$$\boldsymbol{AV} = \boldsymbol{B} \tag{2}$$

where the columns of $\boldsymbol{B}$ are orthogonal. If the columns of $\boldsymbol{B}$ are orthogonal, then it can be written as the product of an orthogonal matrix $\boldsymbol{U}$ and a diagonal matrix $\boldsymbol{D}$

$$\boldsymbol{B} = \boldsymbol{UD} \tag{3}$$

by letting the columns of $\boldsymbol{U}$ be equal to normalized versions of the columns of $\boldsymbol{B}$,

$$\mathbf{u}_i = \frac{\mathbf{b}_i}{\|\mathbf{b}_i\|} \tag{4}$$

and defining the diagonal elements of $\boldsymbol{D}$ to be equal to the norm of the columns of $\boldsymbol{B}$

$$d_{ii} = \|\mathbf{b}_i\|. \tag{5}$$

By substituting (3) into (2) and solving for $\boldsymbol{A}$, one obtains

$$\boldsymbol{A} = \boldsymbol{UDV}^T \tag{6}$$

which is the SVD of $\boldsymbol{A}$.

The critical step in the above procedure for calculating the SVD is determining the orthogonal mtrix $\boldsymbol{V}$ that will orthogonalize the columns of $\boldsymbol{A}$. This matrix is usually formed as a product of Givens rotations, each of which is designed to orthogonalize two columns. Considering the current ith and jth columns of $\boldsymbol{A}$, multiplication by a Givens rotation results in the new columns, $\mathbf{a}_i'$ and $\mathbf{a}_j'$ given by

$$\mathbf{a}_i' = \mathbf{a}_i \cos(\theta) + \mathbf{a}_j \sin(\theta) \tag{7}$$

$$\mathbf{a}_j' = \mathbf{a}_j \cos(\theta) - \mathbf{a}_i \sin(\theta). \tag{8}$$

The constraint that these columns be orthogonal results in

$$\begin{aligned}\mathbf{a}_i'^T\mathbf{a}_j' = 0 = \mathbf{a}_i^T\mathbf{a}_j[\cos^2(\theta) - \sin^2(\theta)] \\ + (\mathbf{a}_j^T\mathbf{a}_j - \mathbf{a}_i^T\mathbf{a}_i)\sin(\theta)\cos(\theta).\end{aligned} \tag{9}$$

The terms in the Givens rotation matrix to achieve orthogonality can be computed by using the formulas given in Nash (1979), which are based on the quantities

$$p = \mathbf{a}_i^T\mathbf{a}_j \tag{10}$$

$$q = \mathbf{a}_i^T\mathbf{a}_i - \mathbf{a}_j^T\mathbf{a}_j \tag{11}$$

$$v = \sqrt{4p^2 + q^2} \tag{12}$$

so that for $q \geq 0$

$$\cos(\theta) = \sqrt{\frac{v+q}{2v}} \quad \text{and} \quad \sin(\theta) = \frac{p}{v\cos(\theta)} \tag{13}$$

and for $q > 0$

$$\sin(\theta) = \operatorname{sgn}(p)\sqrt{\frac{v-q}{2v}} \quad \text{and} \quad \cos(\theta) = \frac{p}{v\sin(\theta)} \tag{14}$$

where

$$\operatorname{sgn}(p) = \begin{cases} 1 & \text{if } p \geq 0 \\ -1 & \text{if } p < 0 \end{cases}. \tag{15}$$

The two sets of formulas are given so that ill-conditioned equations resulting from the subtraction of nearly equal numbers can always be avoided.

The preceding discussion shows how to determine a single Givens rotation that will orthogonalize two columns of a given matrix. It still needs to be shown how the matrix $\boldsymbol{V}$ can be computed from these ele-

mentary rotations. If the Givens rotation to orthogonalize columns i and j is denoted by V_{ij}, then the product of a set of $n(n-1)/2$ rotations denoted by

$$V_k = \prod_{i=1}^{n-1} \left(\prod_{j=i+1}^{n} V_{ij} \right) \tag{16}$$

is referred to as a sweep (Golub and Van Loan 1983). Unfortunately, a single sweep will not, in general, orthogonalize all of the columns of a matrix, since subsequent rotations can destroy the orthogonality produced by previous ones. However, the procedure can be shown to converge (Nash 1975) so that V can be obtained from

$$V = \prod_{k=1}^{l} V_k \tag{17}$$

where the number of sweeps l is not known a priori. Convergence of the algorithm is based on completing an entire sweep with all of the columns being orthogonal. Orthogonality is measured by the parameter α defined as

$$\alpha = \frac{(\mathbf{a}_i^T \mathbf{a}_j)^2}{(\mathbf{a}_i^T \mathbf{a}_i)(\mathbf{a}_j^T \mathbf{a}_j)} \tag{18}$$

dropping below a preset threshold. If for two columns α is below the threshold, then the rotation is not performed.

The above algorithm, by virtue of being composed exclusively of Givens rotations, can be highly parallelized, a task that has already been performed for implementation on the ILLIAC IV (Luk 1980). Architectures specifically designed for this algorithm have also been proposed (Luk 1986b, Schimmel and Luk 1986) and can operate at about $2fn^2$ units of time per sweep for an $n \times n$ matrix where f is the time required for a floating-point multiply and add. It has been shown that the number of sweeps required in the above algorithm is approximately $\log_2 n$. In the following sections it will be shown how perturbation bounds on the singular values and vectors of the Jacobian can be incorporated into this algorithm in order to reduce the number of sweeps required as well as the computational complexity of each sweep.

2.3. Pertubation Bounds on the Singular Value Decomposition

The algorithm for computing the SVD using Givens rotations outlined above uses successive sweeps in order to make the columns of a matrix more orthogonal. The more orthogonal the columns are to begin with, the fewer the number of sweeps required for convergence. If one considers the current manipulator Jacobian to be a pertubation of the previous Jacobian

$$J(t + \Delta t) = J(t) + \Delta J(t), \tag{19}$$

the SVD of which is known and given by

$$J(t) = U(t)D(t)V^T(t), \tag{20}$$

then the matrix $J(t + \Delta t)V(t)$ will have nearly orthogonal columns, provided the perturbation $\Delta J(t)$ is small relative to $J(t)$. The foundation of the above lies in the fundamentally well-behaved nature of the SVD of a matrix. The perturbation bounds on singular values are very well known and easy to show (Forsythe, Malcolm, and Moler 1977):

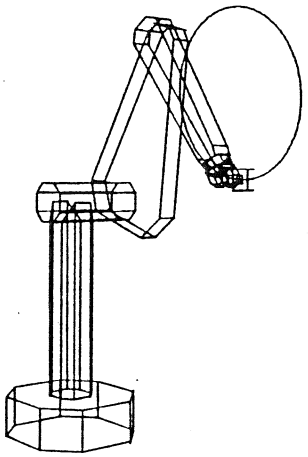
$$|\sigma_i(J(t + \Delta t)) - \sigma_i(J(t))| \leq \|\Delta J(t)\| \tag{21}$$

The perturbation bounds on the rotation of subspaces defined by singular vectors are not as widely known but are also well behaved (Davis and Kahan 1970; Wedin 1972).

3. Implementation of a Real-Time SVD Algorithm

The implementation of the SVD algorithm using Givens rotations and the subsequent refinements were all done in PASCAL on a VAX 785. Substantial testing for a variety of trajectories using a simulation of the PUMA robot have been conducted. Three representative examples are given in Figures 1, 2, and 3, illustrating the starting configuration of the PUMA robot, the desired end-effector trajectory, and the sin-

Fig. 1. Initial configuration and desired end-effector positions for trajectory A, along with the singular values of the Jacobian for each point along this trajectory.



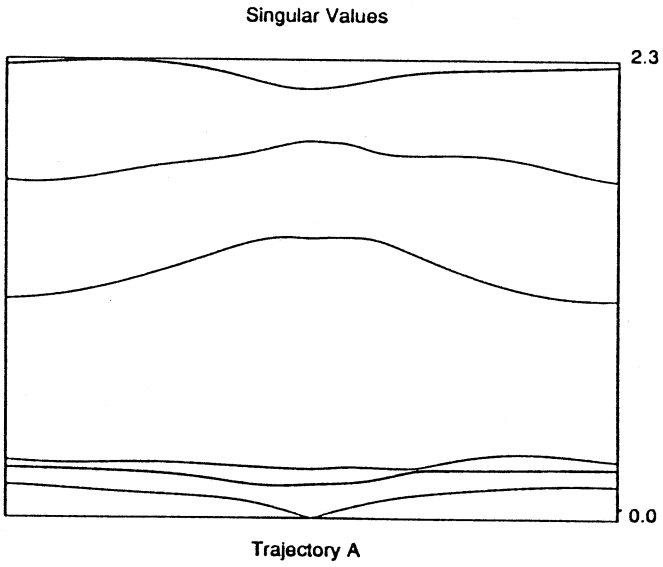
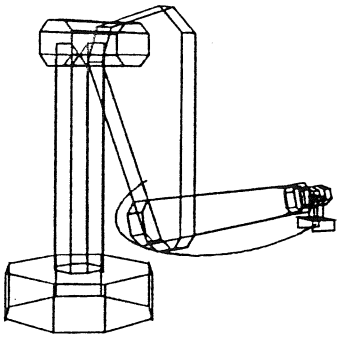
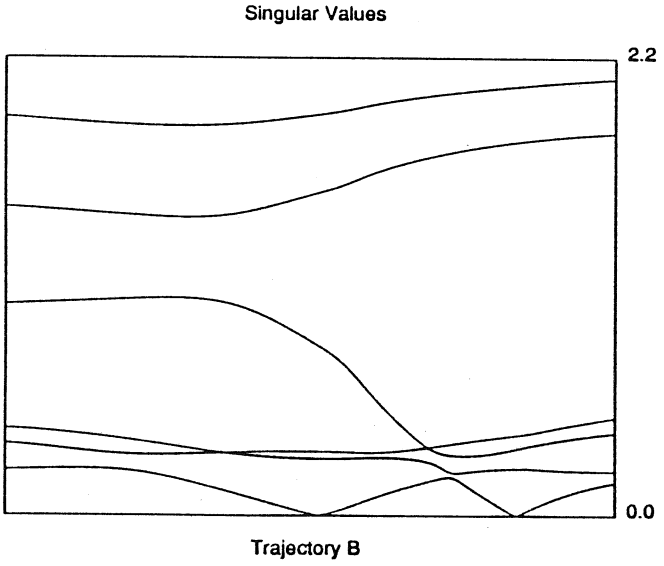


Fig. 2. Initial configuration and desired end-effector positions for trajectory B along with the singular values of the Jacobian for each point along this trajectory.

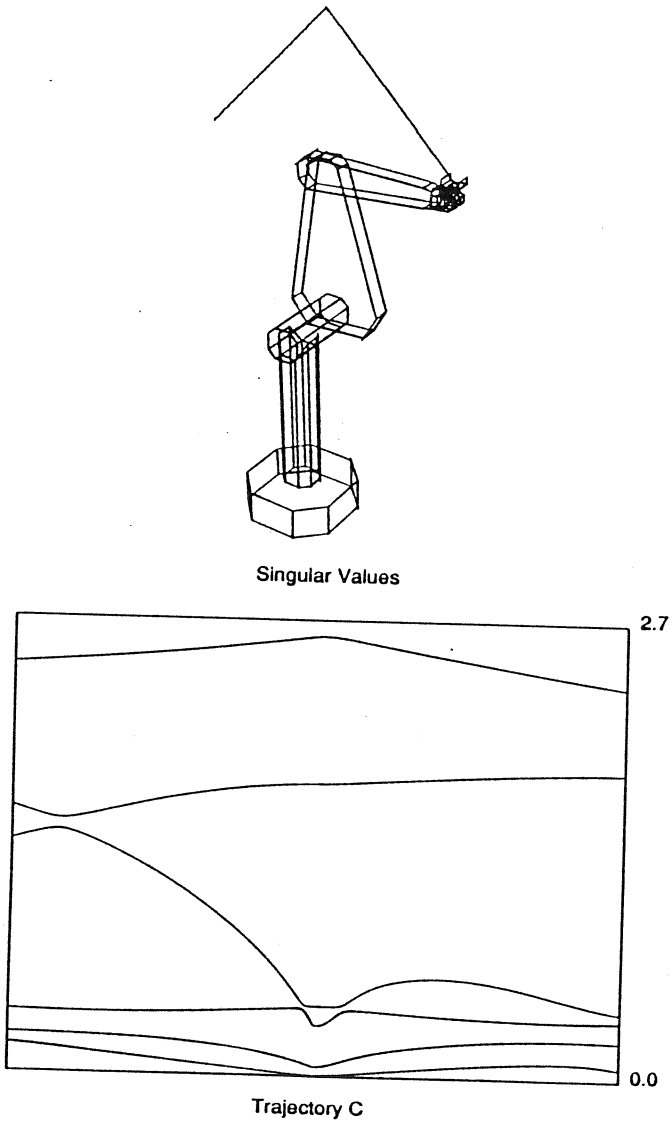




gular values of the Jacobian computed for each point along the trajectory. As can be seen from the singular value plots, these examples all pass near singularities at the midpoint of their trajectories. These singularities are the well-known wrist, shoulder, and elbow singularities that occur in trajectories A, B, and C, respectively. Trajectory C is unique in that it approaches a triple singularity at its midpoint. These examples have been chosen to illustrate the advantages of having the SVD available during the control of a manipulator, since conventional algorithms provide unsatisfactory performance near singular configurations.

In order to illustrate the computational requirements of the basic algorithm, which does not use previous information, and to provide a basis for comparison with the modified algorithm, data on the number of sweeps and plane rotations required to reach convergence is presented in Table 1. Figure 4 shows a plot of these quantities for trajectory B as a representative example. The number of sweeps is the actual number of sweeps in which rotations are performed and does not include the final sweep in which all the columns are checked and determined to be orthogonal. The maximum number of rotations per sweep is 15 since the Jacobian in this case is a 6×6 matrix. Note that the computational expense is fairly uniform over various configurations of the manipulator, with the slight

Fig. 3. Initial configuration and desired end-effector positions for trajectory C along with the singular values of the Jacobian for each point along this trajectory.



variation in the total number of rotations primarily introduced by the final sweep before convergence.

3.1. Incorporating the Previous SVD

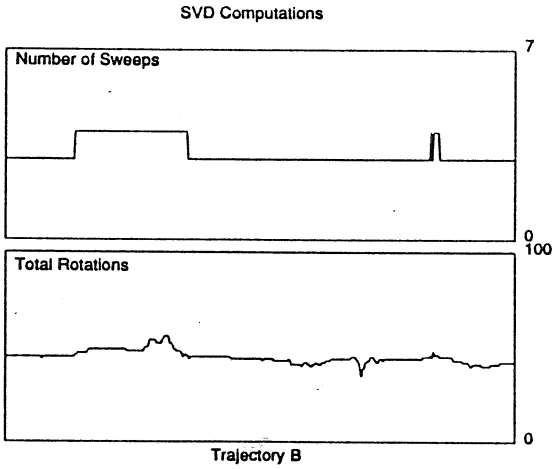
The above experimental data backed by the analytical results pertaining to the perturbation bounds on the rotation of singular vectors seems to indicate that the majority of the computation involved in calculating the SVD is redundant over a given trajectory. Each time the SVD is computed, the columns are orthogonalized by building the matrix V from scratch. If, on the other hand, the value of $V(t + \Delta t)$ is initialized to $V(t)$, then a substantial portion of the work required to compute the current SVD can be eliminated. The results of using this past information in the SVD calculation are presented in Table 2, which illustrates the dramatic reduction in computational expense. The number of rotations is down by about a factor of three to approximately 15, with convergence obtained in virtually one sweep. The nearly uniform requirement of one sweep for convergence suggests removing the iterative nature of the algorithm by fixing the number of sweeps at one. This provides the additional computational advantage of removing convergence tests.

There will still exist cases where a single sweep will not result in convergence, thus introducing error into the calculated SVD. A graph of this error for SVD calculations along trajectory B is presented in Figure 5. Only data for trajectory B is plotted, since the SVD calculations along both trajectories A and C only require a maximum of one sweep for convergence. Several error measures have been computed in order to differentiate the type of error and its source. The singular value error, denoted here by σ_{err}, is a measure of the error between the calculated singular values, σ_c, and the actual singular values, σ_a (computed using the Golub-Reinsch algorithm in the IMSL package), de-

Table 1. Summary of the Computational Requirements for Computing the SVD Using Givens Rotations and No Previous Information

	Rotations			Sweeps		
Trajectory	Min.	Max.	Avg.	Min.	Max.	Avg.
A	39	52	45.7	3	4	3.63
B	35	56	44.9	3	4	3.24
C	32	45	40.2	3	3	3.00
Total	32	56	43.6	3	4	3.29

Fig. 4. The number of sweeps and rotations required to compute the SVD of the Jacobian along trajectory B.



fined by the equation

$$\sigma_{err} = \sum_{i=1}^{6} (\sigma_{a_i} - \sigma_{c_i})^2. \tag{22}$$

The error in the input and output singular vectors, denoted by V_{err} *and* U_{err}, respectively, is computed using the equations

$$V_{err} = \max(\|I - VV^T\|, \|I - V^TV\|) \tag{23}$$

and

$$U_{err} = \max(\|I - UU^T\|, \|I - U^TU\|) \tag{24}$$

where the spectral norm is used (Lawson and Hanson 1974). Finally, the error in the Jacobian, denoted by J_{err}, is computed using

$$J_{err} = \frac{\|J - UDV^T\|}{\|J\|}, \tag{25}$$

which is the normalized error in using the computed SVD as a representation for the Jacobian.

Table 2. Computational Requirements for Computing the SVD Using the Previous Estimate of the Matrix V

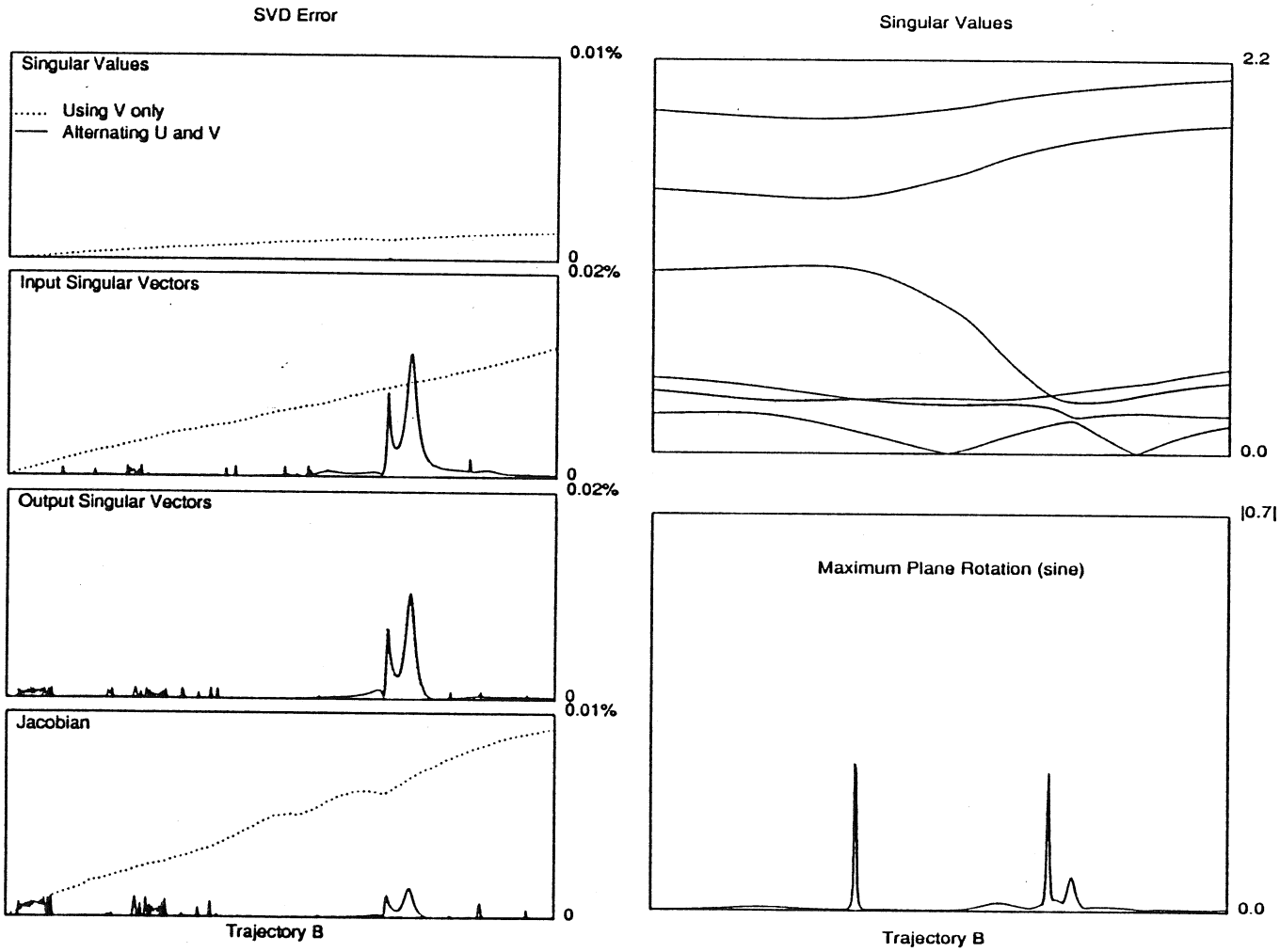
	Rotations			Sweeps		
Trajectory	Min.	Max.	Avg.	Min.	Max.	Avg.
A	14	15	14.975	1	1	1.00
B	14	17	14.970	1	2	1.07
C	13	15	14.831	1	1	1.00

The first point to note about the error terms plotted in Figure 5 is that they are all very small in magnitude, with a maximum on the order of 0.01%. The second important characteristic is the fundamental difference between the monotonically increasing error of the input singular vectors, singular values, and the Jacobian, as compared to the error plot of the output singular vectors. The monotonically increasing error in the input singular vectors is a result of compounding the roundoff error of previous computations by initializing V to its value from the previous computation interval. This error in V is in turn responsible for the error in the singular values and the Jacobian. This error, however, is not carried over into the output singular vectors due to the fundamental difference in the way that they are computed. While V is computed as a product of successive plane rotations, U is computed by normalization of the orthogonal columns of B (see eq. (4)). Therefore, since the compounded error in V does not affect the orthogonality of the columns of B, the error in U is not monotonically increasing, but results from the additional sweep that would be required for convergence. Analysis of the sweep data shows that the two peaks in the output singular vector error correspond to those Jacobians that required two sweeps in order to orthogonalize their columns. It is important to note, however, that this error is subsequently reduced to its previous small value.

The monotonically increasing error due to using the previous V matrix is still of some concern. While the magnitude of this error is small for this trajectory, it will grow without bound. Note that for repetitive tasks in conservative systems this is not a problem; when the manipulator returns to its initial starting configuration, the true SVD is assumed to be known, and V

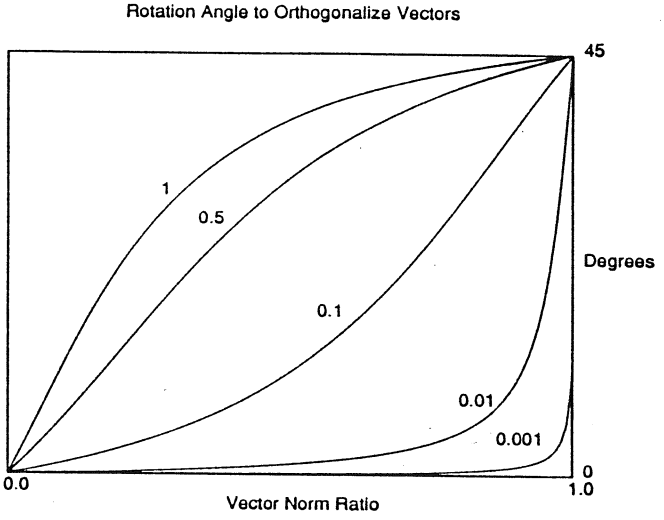
Fig. 5. Error in computation of the SVD introduced from incorporating the previous V matrix and using a single sweep.

Fig. 6. The singular values of the Jacobian for each point along trajectory B along with the maximum magnitude of the plane rotation required during the computation of the SVD.



can be reset. For arbitrary open trajectories, however, this will not be the case. A simple solution would be to periodically recompute the SVD with V reset to I; however, this will take between three and four sweeps and result in a nonuniform computation time interval. Fortunately, there is an alternative technique that takes advantage of the contrast between the error in computing V and U. Since U is not corrupted by compounded error, it can be used to reset V by carrying out the plane rotations on the rows instead of the columns of J. Thus if $U^T J = B$ where the rows of B are orthogonal, then $B = DV^T$ where the rows of V^T are normalized rows of B, so that once again $J = UDV^T$. Since V has now been computed by the normalization of orthogonal rows, it is not affected by the compounded error in the previous V. By alternating this procedure each interval, applying plane rotations first from the right on the columns of J and the next time from the left on the rows, one can effectively eliminate the buildup of error along the trajectory. A plot of the SVD error terms for trajectory B using this technique is given in Figure 5. As expected, the monotonically increasing error in the input singular vectors has been eliminated and is therefore also no longer reflected in the computed singular values or in J_{err}. The error terms are now all of the same form, with peaks at those configurations where two sweeps would be required to orthogonalize the rows or columns of the Jacobian.

Fig. 7. A graph of the plane rotation angle required to orthogonalize two vectors versus the relative length of the vectors. This function is plotted for various values of the angle between the two vectors.



3.2. Small Angle Approximations

A further reduction in the computational complexity of the equations for computing the SVD can be gained by examining the nature of the plane rotations required. A plot of the maximum rotation angle required during the computation of the SVD of the Jacobian for each point along trajectory B is presented in Figure 6. Note that for the majority of the trajectory the maximum rotation angle is very small; however, there exist three peaks, two of which are very sharp. From comparing the position of these peaks to the spacing of the singular values also presented in Figure 6, it is clear that very large rotations are required when there is a crossing of adjacent singular values. By comparing the position of the second two peaks in Figure 6 with those in the error plots of the singular vectors in Figure 5, one can see that they coincide. Thus the large angles in the plane rotations result in an extra sweep being required for the algorithm to converge, thus introducing the error. It may at first seem curious that the largest of the peaks, the first, does not produce any peak in the error plot. The reason for this apparent anomaly is that a large rotation angle is a necessary but not a sufficient condition to require an additional sweep. From examining the spacing of the singular values at the position of the first peak, one can see that the two equal singular values, σ_4 and σ_5, are separated from the remaining singular values. This implies that the large rotation required is restricted to the plane defined by their associated singular vectors and can therefore be successfully completed within a single sweep. In contrast, the final two peaks that result from the crossing of singular values σ_3 with σ_4 and σ_5 with σ_6, respectively, occur at a point where these four singular values are closely spaced. This results in a greater interaction between plane rotations, therefore creating the necessity of an additional sweep for convergence and introducing error into the single sweep approximation.

Since the maximum plane rotations required during the algorithm are very small in magnitude outside of a few isolated peaks, a small angle approximation is useful in reducing the computational effort required in computing the SVD. The plane rotation required to orthogonalize two vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ is obtained by satisfying eq. (9). By applying the double angle formulas, the above equation can be solved for θ resulting in

$$\theta = \tfrac{1}{2} \tan^{-1} \frac{2\mathbf{a}_i^T\mathbf{a}_j}{\mathbf{a}_j^T\mathbf{a}_j - \mathbf{a}_i^T\mathbf{a}_i}. \tag{26}$$

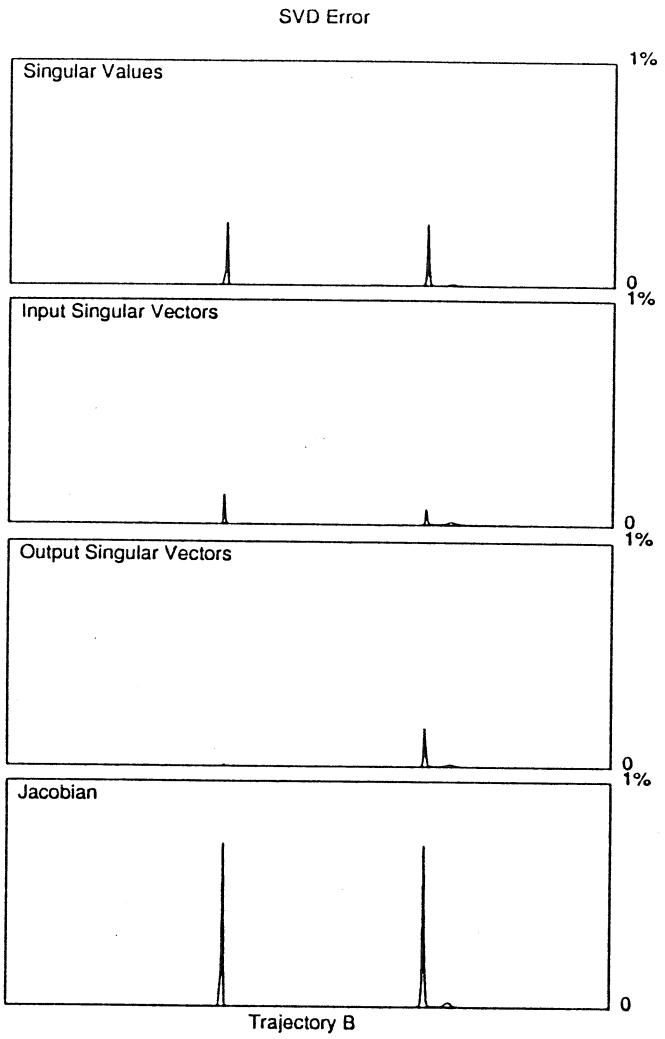
Dividing both the numerator and the denominator of the right hand side of eq. (26) by $\|\mathbf{a}_j\|^2$ results in the still exact equation

$$\theta = \tfrac{1}{2} \tan^{-1} \frac{2\cos(\phi)\dfrac{\|\mathbf{a}_i\|}{\|\mathbf{a}_j\|}}{1 - \left(\dfrac{\|\mathbf{a}_i\|}{\mathbf{a}_j\|}\right)^2} \tag{27}$$

where ϕ is the angle between $\mathbf{a}_i$ and $\mathbf{a}_j$. This form of the equation makes explicit the dependence of the plane rotation angle on both the non-orthogonality of the two vectors involved as well as their relative length. A plot of the required plane rotation to orthogonalize two vectors versus their relative lengths for various values of $\cos(\phi)$ is given in Figure 7.

One interesting point about the graph in Figure 7 is that when the two vectors are of equal length they will be orthogonalized by a rotation of 45 degrees regardless of the angle between them. The most important point in terms of using a small angle approximation,

Fig. 8. Error in computation of the SVD introduced from using a small angle approximation.



however, is that for nearly orthogonal vectors the rotation angle will be small, except when there are nearly equal singular values. Using the previous estimate of the SVD to pre-orthogonalize the current Jacobian guarantees that this will be true. Therefore, if the two vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ are not equal in length, then θ will be small. For small θ the approximations

$$\cos(\theta) \approx 1 \quad \text{and} \quad \sin(\theta) \approx \theta \tag{28}$$

are valid. The advantages of using this approximation are two-fold. First, the solution of eq. (26) is vastly simplified to

$$\theta = \frac{a_i^T \mathbf{a}_j}{\mathbf{a}_j^T \mathbf{a}_j - \mathbf{a}_i^T \mathbf{a}_i} \tag{29}$$

so that the expensive computations of eqs. (12)–(14) are no longer required. Second, the calculations to compute the results of this plane rotation previously required two floating point multiplies and one addition per element. By using this small angle approximation, eqs. (7) and (8) now become

$$\mathbf{a}_i' = \mathbf{a}_i + \mathbf{a}_j\theta \quad \text{and} \quad \mathbf{a}_j' = \mathbf{a}_j - \mathbf{a}_i\theta \tag{30}$$

so that the number of floating point multiplies required has been cut in half. One must still consider, however, the case where the two vectors are of nearly equal length (i.e., nearly equal singular values). As discussed above, under these circumstances not only is the small angle approximation not valid, but the angle of rotation is at its maximum value of 45°. Fortunately, the cosine and the sine of 45° are equal, so that the reduction in floating point multiplies can still be achieved. Therefore, if $\|\mathbf{a}_i\| \approx \|\mathbf{a}_j\|$ then

$$\cos(\theta) = \sin(\theta) = \sqrt{2}/2 \tag{31}$$

and eqs. (7) and (8) become

$$\mathbf{a}_i' = \frac{\sqrt{2}}{2}(\mathbf{a}_i + \mathbf{a}_j) \quad \text{and} \quad \mathbf{a}_j' = \frac{\sqrt{2}}{2}(\mathbf{a}_j - \mathbf{a}_i), \tag{32}$$

which still only require one floating point multiply and addition per element.

Simulation results showing the error in the computed SVD using the small angle approximations discussed above are presented for trajectory B in Figure 8. Trajectory B is presented as an example, since it represents the worst case of the three trajectories. The form of the error terms now more closely reflects the plot of Figure 6, since approximations for large rotations will always introduce error regardless of whether they are restricted to a single plane. The peak errors are all well within 1%, with nominal values at around 0.001%. The peak values simply reflect the inherently ill-conditioned nature of trying to define singular vectors for nearly equal singular values. The small angle approximation primarily introduces error to the norm

of the rows and columns of the rotation matrix but not to their orthogonality. Thus a renormalization of the resultant rotation matrix significantly reduces the errors introduced by the small angle approximation. While this degree of accuracy should be sufficient for most applications, it is important to note that manipulator configurations that produce peaks in the error terms are easily identified by examining the spacing of the singular values. Therefore, the small angle approximations can be abandoned when there are closely spaced singular values, thus obtaining an even higher degree of accuracy while retaining the computational advantages of the approximation throughout the rest of the trajectory.

3.3. Numerical Evaluation

A test was performed to evaluate the actual CPU time needed to compute the SVD of the Jacobian matrix for a six-degree-of-freedom manipulator. The algorithm using small angle approximations and the previous estimate of the singular vectors, coded in PASCAL and executed on a VAX785 computer, required 5.13 ms. In comparison, the execution time for the unmodified algorithm (without small angle approximations or the use of previous estimates) required an average of 27.2 ms for a typical Jacobian. This figure is comparable to the time required by the Golub-Reinsch algorithm in the IMSL package (32.3 ms). These figures, however, do not reflect the advantage of the algorithm in terms of its parallelism, since they are coded and executed on a serial machine. By using a simple mesh connection of processing elements capable of executing an addition or multiplication, the plane rotations can be computed in parallel. Such architectures suggest that a computation time of well within 1 ms are easily achievable (Luk 1986b; Schimmel and Luk 1986).

4. Applications

The ability to calculate the SVD of the Jacobian in real time has a number of possible different applications. As mentioned above, several proposed dexterity measures are closely linked to the singular values and vectors of the Jacobian. Thus the kinematic and static force capabilities of the current manipulator configuration can be compared to the requirements of an assigned task. Inconsistencies between the physical capabilities of the manipulator and the assigned task can be addressed by the manipulator itself by determining a more suitable configuration. This results in more autonomous behavior, which can be utilized for both automated motion planning and work-cell design as well as for the operation of robotic manipulators in unstructured environments.

Real-time computation of the SVD also enhances the utilization of redundancy in robotic systems. In terms of the resolved motion rate control formulation (Whitney 1969),

$$J\dot{\theta} = \dot{\mathbf{x}} \tag{33}$$

one of the most common techniques for using the redundant degrees of freedom within the system is to use the projection operator formulation proposed in Liegeois (1977)

$$\dot{\theta} = J^{+}\dot{\mathbf{x}} + (I - J^{+}J)\mathbf{z} \tag{34}$$

where J^{+} is the pseudoinverse of J and $\mathbf{z}$ is an arbitrary vector in $\dot{\theta}$ space. This formulation has been used to optimize a number of secondary criteria under the constraint of a specified end-effector trajectory including joint availability, torque minimization (Hollerbach and Suh 1987), and obstacle avoidance (Maciejewski and Klein 1985; Nakamura, Hanafusa, and Yoshikawa 1987). With the complete SVD available, the projection operation becomes trivial, since the singular vectors $\mathbf{v}_i$ for $r > i \geq n$ specify an orthonormal basis for the null space. Thus the relative advantages of using the homogeneous solution for alternate secondary criteria can be easily evaluated.

The remainder of this work will consider the application of the real-time SVD algorithm to the fundamental problem of singular configurations. With the exception of Cartesian positioning robots, all articulated manipulators can be shown to possess singular configurations that limit the effective number of independent degrees of freedom (Baker and Wampler

1987). A considerable amount of effort (Asada and Cro Granito 1985; Aboaf and Paul 1987; Dubey and Luh 1987; Mayorga and Wong 1987; Sampei and Furuta 1987) has been devoted to either avoiding or dealing with operations at singularities due to the high joint velocities and spurious motions that can result.

4.1. Damped Least-Squares Solutions

The effects of singularities are frequently presented with respect to the resolved motion rate control formulation given in eq. (33). Singularities are identified by a mathematical change of rank in J, which physically represents the inability of the manipulator to achieve an arbitrary end-effector velocity. For these cases, inverses are not defined, and even pseudoinverse solutions such as eq. (34) are unsatisfactory since there is an undesirable discontinuity at the singularity that can result in oscillations and unacceptably high joint velocities. These difficulties are not unique to the resolved motion rate formulation but are an inherent part of the transformation between Cartesian and joint space.

A general approach to resolving the discontinuity at singular configurations and maintaining a well-conditioned formulation that results in physically meaningful joint velocities is to use the damped least-squares formulation independently proposed in Nakamura and Hanafusa (1986) and Wampler (1986). The damped least-squares solution of eq. (33) is the solution that minimizes the sum $\|\dot{\mathbf{x}} - \boldsymbol{J}\dot{\boldsymbol{\theta}}\| + \lambda\|\dot{\boldsymbol{\theta}}\|$ so that the end-effector tracking error is weighted against the norm of the joint velocity by using λ, also known as the damping factor. This solution is typically obtained by solving an equation of the form

$$(\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})\dot{\boldsymbol{\theta}} = \boldsymbol{J}^T\dot{\mathbf{x}}. \tag{35}$$

This solution is guaranteed to be the minimal residual solution over all solutions of equal or smaller norms. Unfortunately, the norm of the solution cannot be determined a priori for a given damping factor.

The specification of the problem one would like to solve is the minimization of the residual $\|\dot{\mathbf{x}} - \boldsymbol{J}\dot{\boldsymbol{\theta}}\|$, which defines the end-effector tracking accuracy under the constraint $\|\dot{\boldsymbol{\theta}}\| \leq \dot{\theta}_{max}$ where $\dot{\theta}_{max}$ is the physical limit on the manipulator's joint velocity. The desired solution can therefore be obtained by using the damped least-squares solution for an appropriate value of the damping factor. Intuitively, if the value of $\|\dot{\boldsymbol{\theta}}\|$ for which the residual is equal to zero is less than $\dot{\theta}_{max}$, then $\lambda = 0$; otherwise λ would take on the value that results in $\|\dot{\boldsymbol{\theta}}\| = \dot{\theta}_{max}$. In physical terms, if the joint velocity that exactly tracks the desired end-effector trajectory is physically achievable, then it should be used; otherwise the optimal solution requires that the joint velocity norm be at its limit.

The damped least-squares solution of eq. (33), which will be denoted by $\dot{\boldsymbol{\theta}}^{(\lambda)}$ in order to denote its explicit dependence on the damping factor, is given by

$$\dot{\boldsymbol{\theta}}^{(\lambda)} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i\mathbf{u}_i^T\dot{\mathbf{x}} \tag{36}$$

where

$$\boldsymbol{J} = \sum_{i=1}^{n} \sigma_i\mathbf{u}_i\mathbf{v}_i^T \tag{37}$$

is the SVD of the Jacobian. The solution norm is, therefore, given by

$$\|\dot{\boldsymbol{\theta}}^{(\lambda)}\|^2 = \sum_{i=1}^{r} \left[\frac{\dot{x}_i\sigma_i}{\sigma_i^2 + \lambda^2}\right]^2 \tag{38}$$

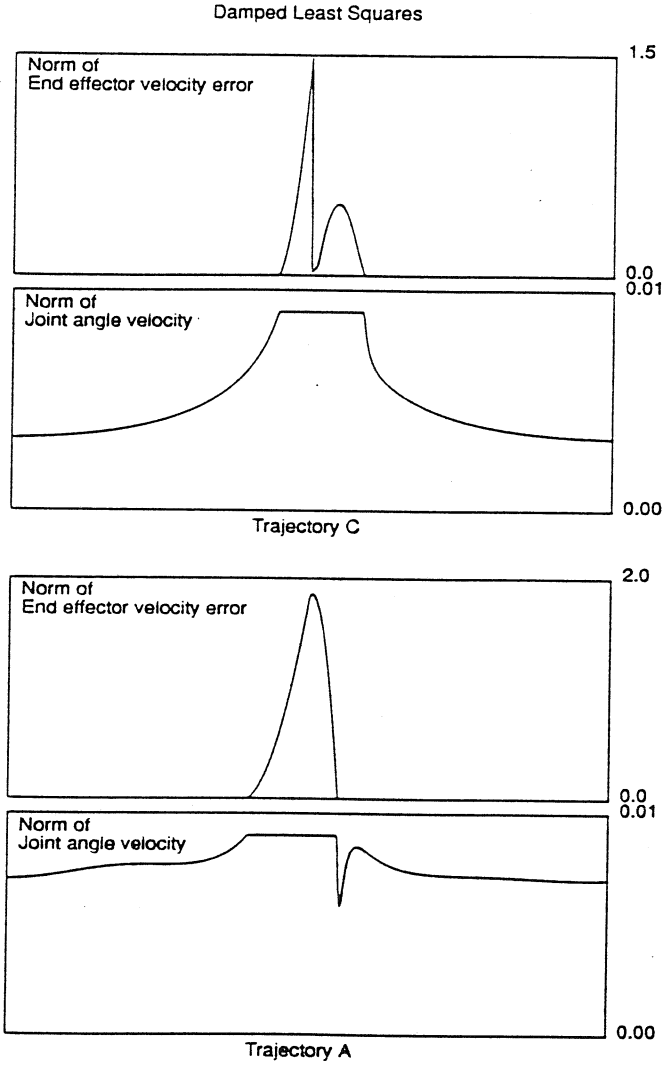
where

$$\dot{x}_i = \mathbf{u}_i^T\dot{\mathbf{x}}. \tag{39}$$

Equation (38) is the nonlinear equation in λ that must be solved in order to find the optimal solution when $\|\dot{\boldsymbol{\theta}}^{(\lambda)}\| = \dot{\theta}_{max}$. An efficient technique for doing so is to use Newton's method, which requires the derivative of eq. (38) with respect to the damping factor, as discussed in Lawson and Hanson (1974).

The above technique was implemented in a simulation of the PUMA robot for the three trajectories presented in Figures 1, 2, and 3. As mentioned previously, these trajectories are chosen to go through singular configuration in order to illustrate the properties of the damped least-squares solution. The results

Fig. 9. The norm of the end-effector velocity tracking error and the joint angle velocity for trajectory A and C using the damped least-squares solution.



of the simulation, both the end-effector velocity tracking error and joint angle velocity norm, are plotted for the three trajectories in Figures 9 and 10. The maximum joint velocity norm $\dot{\theta}_{max}$ was set at 0.009 radians per computation interval for all three trajectories. The end-effector tracking error is zero at all points along the trajectory, except where the Jacobian becomes nearly singular and the desired end-effector velocity has a component in the direction of the lost degrees of freedom. At these points the characteristic jump in the joint velocity is observed but is effectively clamped at the maximum allowable value. This prevents the spurious motions that are typical of manipulators passing through singular configurations using other methods of inverse kinematics. Thus the solutions are physically meaningful even near singular configurations, and, in fact, they result in the minimum amount of end-effector tracking error. This error can actually be zero if the commanded end-effector velocity does not have a component in the direction of the singular vectors associated with the small singular values (notice the notch in Fig. 9). This, in fact, is the advantage of having the complete SVD available instead of only limited information on singularities. It should be noted that, as a result of passing through a singular configuration, it is possible for the manipulator to switch solution branches (for example, going from an elbow-up to an elbow-down configuration).

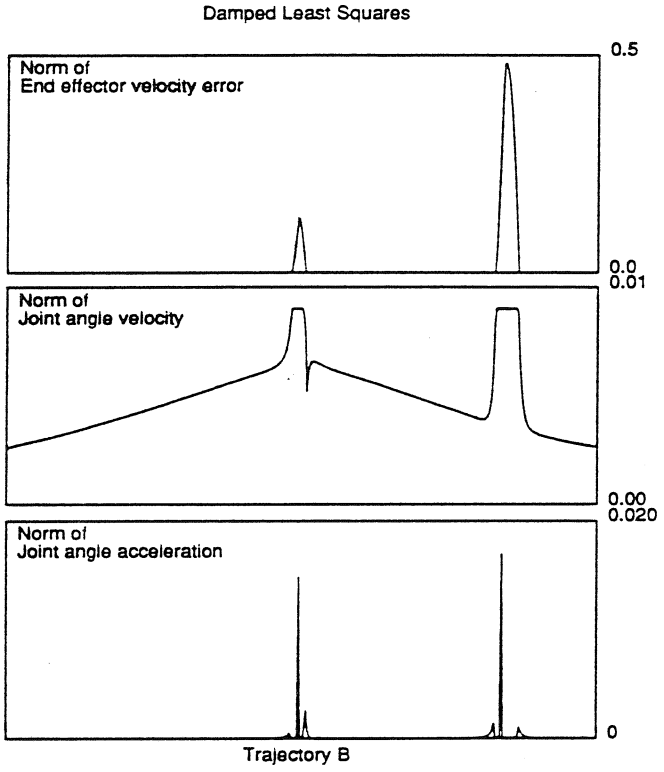
4.2. A Continuous Version of the Truncated SVD Solution

The use of damped least squares provides the optimal solution for tracking a given end-effector trajectory under the physical constraints on the joint motions. While this solution is optimal, it may be undesirable to implement due to the iterative nature of calculating the appropriate damping factor λ. It can be shown that the characteristics of the damped least-squares solution are very similar to those obtained using the truncated SVD solution. The truncated SVD solution of a linear system of equations described by eq. (33), denoted here by $\dot{\theta}^{(k)}$, is defined as

$$\dot{\theta}^{(k)} = \sum_{i=1}^{k} \frac{\dot{x}_i}{\sigma_i} \mathbf{v}_i \tag{40}$$

where k is an integer less than or equal to the rank r. The truncated SVD reduces the solution norm by removing all components of the solution that correspond to small singular values while retaining all of those associated with larger singular values. The parameter k is used to define small and large such that σ_i for $i \leq k$ are large, and σ_i for $i > k$ are considered small. It can be shown that $\dot{\theta}^{(k)}$ is the minimum resid-

Fig. 10. The norm of the end-effector velocity tracking error, joint velocity, and joint accelerations for trajectory B using the damped least-squares solution at the velocity level.



ual solution for all $\dot{\theta}$ in the k-dimensional subspace spanned by $\mathbf{v}_i$ for $i \leq k$ (Marquardt 1970). For cases where $\sigma_k \gg \sigma_{k+1}$ and λ falls between the two largely separated singular values σ_k and σ_{k+1}, the results for the two types of solutions will be approximately the same. By modifying the truncated SVD to be continuous instead of a stepwise function of k, a solution that is virtually identical to the damped least-squares solution can be obtained. This type of solution will be denoted $\dot{\theta}^{(c)}$ and is defined by

$$\dot{\theta}^{(c)} = \sum_{i=1}^{k} \frac{\dot{x}_i}{\sigma_i} \mathbf{v}_i + \frac{(c-k)\dot{x}_{k+1}}{\sigma_{k+1}} \mathbf{v}_{k+1} \qquad (41)$$

where c is a real number less than or equal to the rank, and k is the greatest integer less than or equal to c. The norm of this type of solution is given by

$$\|\dot{\theta}^{(c)}\|^2 = \sum_{i=1}^{k} \left(\frac{\dot{x}_i}{\sigma_i}\right)^2 + \left(\frac{(c-k)\dot{x}_{k+1}}{\sigma_{k+1}}\right)^2, \qquad (42)$$

since singular vectors are mutually orthogonal unit vectors.

The advantages of using this form of a solution are that, when the SVD is available, it is extremely easy to compute. The norms of the truncated singular value solutions $\|\dot{\theta}^{(i)}\|$, obtained from eq. (40), are computed as i is incremented until either i is equal to the rank or the norm is greater than $\dot{\theta}_{max}$. The latter case is the one of interest since it represents reaching the physical constraint on the joint velocity. In this case k is now known, namely $i-1$, and the desired quantity $(c-k)$ for which $\|\dot{\theta}^{(c)}\|$ is equal to $\dot{\theta}_{max}$ can be easily obtained from (42). This value is then used to obtain the continuous truncated singular value solution defined by (41). An implementation of this algorithm was used to simulate control of a PUMA robot for the three trajectories presented above. In all three cases the resultant joint trajectories were within 1% of those obtained using the damped least-squares solution. A detailed error analysis of the difference between the continuous truncated SVD solution and the damped least-squares solution can be found in Maciejewski (1987).

4.3. Resolved Acceleration Control

The above sections have discussed how to deal with hard constraints on the joint angle velocities in the presence of singularities by removing those components associated with small singular values. In many practical cases, however, the joint accelerations will be the limiting factor. The same techniques are equally applicable, since resolved acceleration control (Luh, Walker, and Paul 1980) still requires a solution based on some sort of inverse of the Jacobian. This is easily seen by differentiating eq. (33) to obtain

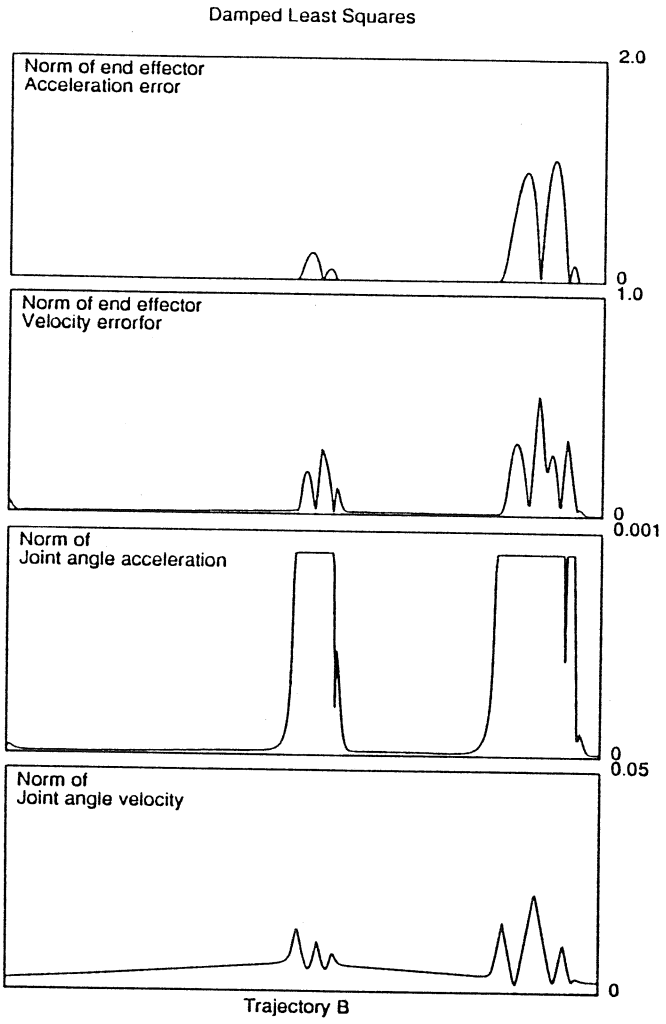
$$\boldsymbol{J}\ddot{\theta} + \dot{\boldsymbol{J}}\dot{\theta} = \ddot{\mathbf{x}}. \qquad (43)$$

Thus for a given state of the manipulator, the joint accelerations required to achieve a desired end-effector acceleration can be computed. These joint accelerations can become infinite in the presence of singularities, so that once again a practical solution is to minimize the residual $\|\ddot{\mathbf{x}} - (\boldsymbol{J}\ddot{\theta} + \dot{\boldsymbol{J}}\dot{\theta})\|$ under the constraint

of physically achievable accelerations defined by $\|\ddot{\theta}\| \leq \ddot{\theta}_{max}$. As an example, consider the use of velocity control for tracking trajectory B given in Figure 10. A plot of the joint accelerations required to maintain the desired velocities is also given in this figure. The form of the joint acceleration norm is typical in that a very large spike is located at the singularity, with two smaller peaks on either side. Note that this type of behavior is not immediately apparent if one considers simply differentiating the norm of the joint velocities, but it becomes clear after considering the effect of passing through a singularity. In particular, the first acceleration peak is a result of the joints accelerating to match the large velocity required due to the component that is in the direction associated with the small singular value. The acceleration then goes to zero since the hard constraint on the velocities limits the effect of this singular component. As the manipulator passes through the singularity, however, the component along the small singular value switches sign so that those joints required to match it must decelerate to zero and then accelerate in the opposite direction, thus resulting in the large spike in the acceleration curve. There is no spike in the velocity curve at this point, since the velocity constraint is still in effect, but it is now limiting the velocity in the other direction. The final peak in the acceleration is then the result of leaving the singular region so that the joints decelerate, since large velocities are not required outside of singular regions.

In order to obtain an optimum solution in the presence of hard constraints on the joint accelerations, the continuous truncated SVD technique was applied to the solution of eq. (43). The resulting joint velocities and accelerations along with the end-effector tracking errors are presented in Figure 11. As expected, the end-effector acceleration tracking error is zero, except near the singularities, where the hard constraint on the acceleration is encountered. This constant limit on the acceleration results in the triangular shape of the joint velocity curves, which alternate between acceleration and deceleration. Imposing such an acceleration constraint effectively increases the region in which the effect of the singularity is felt. This results in an end-effector velocity tracking error that is non-zero for a larger portion of the trajectory around the two singular configurations. This tracking error, however, is the minimum achievable for the given physical limit on the acceleration of the manipulator.

Fig. 11. The norm of the end-effector velocity and acceleration tracking error and joint angle velocity and acceleration norms for trajectory B using the damped least-squares solution for resolved acceleration control.



5. Conclusions

The conclusions of the work presented here can be divided into those relating to calculation of the SVD in real time and those pertaining to the advantages of having the SVD of the Jacobian available for the control of robotic manipulators. With regard to the first

point, the major advantage of the implementation of the SVD algorithm presented here is that by using the known SVD of a previous Jacobian, the computational effort of calculating the SVD of the current Jacobian can be greatly reduced. By applying the rotation implied by the previously known singular vectors, a nearly orthogonal matrix results, which will typically converge in a single sweep. This fact alone reduces the computation time by greater than a factor of three, since it has been shown that the original algorithm will require between three and four sweeps to converge. In addition, the computation time is now known a priori, also removing the need for convergence tests. The potential difficulty of accumulating the error of previous computations is removed by alternating the use of the input and output singular vectors with which to apply the rotations. This modification, while improving the error characteristics, does not affect the computational requirements. Finally, by taking advantage of small angle approximations, the number of floating point multiplications required in the plane rotations is cut in half, with the total number of floating point operations cut by a third.

The availability of a computationally efficient algorithm for computing the SVD of the Jacobian results in a large number of potential applications, including the real-time evaluation of dexterity and utilization of redundant degrees of freedom. This work illustrates the application of the SVD to the fundamental problem of dealing with singularities. The optimal solution to the damped least-squares formulation can be easily and efficiently obtained when the SVD is available. Furthermore, it has been shown how a continuous form of the truncated SVD solution can be used in place of the damped least-squares solution. The use of this formulation at either the velocity or acceleration level allows operation through singular configurations without violating physical constraints or generating spurious motions.

Acknowledgments

This work was supported by the National Science Foundation under grant DMC-8421321.

References

Aboaf, E. W., and Paul, R. P. 1987 (March 31–April 3, Raleigh, N.C.). Living with the singularity of robot wrists. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1713–1717.

Asada, H., and Cro Granito, J. A. 1985 (March 25–28, St. Louis). Kinematic and static characterization of wrist joints and their optimal design. *Proc. 1985 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 244–250.

Baillieul, J. 1987 (March 31–April 3, Raleigh, N.C.). A constraint oriented approach to inverse problems for kinematically redundant manipulators. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1827–1833.

Baker, D. R., and Wampler, C. W. 1987 (March 31–April 3, Raleigh, N.C.). Some facts concerning the inverse kinematics of redundant manipulators. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 604–609.

Businger, P. A., and Golub, G. H. 1969. Algorithm 358: Singular value decomposition of a complex matrix. *Communicat. ACM.* 12(10):564–565.

Chiu, S. L. 1987 (March 31–April 3, Raleigh, N.C.). Control of redundant manipulators for task compatibility. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1718–1724.

Davis, C., and Kahan, W. M. 1970. The rotation of eigenvectors by a perturbation. *SIAM J. Numer. Anal.* 7(1):1–46.

Dubey, R., and Luh, J. Y. S. 1987 (March 31–April 3, Raleigh, N.C.). Redundant robot control for higher flexibility. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1066–1072.

Forsythe, G. E., Malcolm, M. A., and Moler, C. B. 1977. *Computer Methods for Mathematical Computations.* Englewood Cliffs, N.J.: Prentice-Hall.

Golub, G. H., and Kahan, W. M. 1965. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.* 2(2):205–224.

Golub, G. H., and Reinsch, C. 1970. Singular value decomposition and least squares solutions. *Numer. Math.* 14(5):403–420.

Golub, G. H., and Van Loan, C. F. 1983. *Matrix Computations.* Baltimore: Johns Hopkins University Press.

Hestenes, M. R. 1958. Inversion of matrices by biorthogonalization and related results. *J. Soc. Ind. Appl. Math.* 6(1):51–90.

Hollerbach, J. M., and Suh, K. C. 1987. Redundancy resolution of manipulators through torque optimization. *IEEE J. Robot. Automat.* RA-3(4):308–316.

Jacobi, C. G. J. 1846. Uber ein leichtes verfahren die in der theorie der sacularstorungen vorkommendern gleichungen numerisch aufzulosen. *Crelles's J.* 30:51–94.

Klein, C. A., and Blaho, B. E. 1987. Dexterity measures for the design and control of kinematically redundant manipulators. *Int. J. Robot. Res.* 6(2):72–83.

Klein, C. A., and Huang, C. H. 1983. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. Sys. Man Cybernet.* SMC-13(3):245–250.

Lawson, C. L., and Hanson, R. J. 1974. *Solving Least Squares Problems.* Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Liegeois, A. 1977. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Sys. Man Cybernet.* SMC-7(12):868–871.

Luh, J. Y. S., Walker, M. W., and Paul, R. P. C. 1980. Resolved-acceleration control of mechanical manipulators. *IEEE Trans. Automat. Control* AC-25(3):468–474.

Luk, F. T. 1980. Computing the SVD on the ILLIAC IV. *ACM Trans. Math. Software* 6(4):524–539.

Luk, F. T. 1986a. A rotation method for computing the QR-decomposition. *SIAM J. Sci. Stat. Comput.* 7(2):452–459.

Luk, F. T. 1986b. A triangular processor array for computing singular values. *Linear Algebra Appls.* 77(5):259–273.

Maciejewski, A. A. 1987. The analysis and control of robotic manipulators operating at or near kinematically singular configurations. Ph.D. thesis, Department of Electrical Engineering, The Ohio State University.

Maciejewski, A. A., and Klein, C. A. 1985. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int. J. Robot. Res.* 4(3):109–117.

Marquardt, D. W. 1970. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics* 12(3):591–612.

Mayorga, R. V., and Wong, A. K. C. 1987 (March 31–April 3, Raleigh, N.C.). A singularities avoidance method for the trajectory planning of redundant and nonredundant robot manipulators. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1707–1712.

Nakamura, Y., and Hanafusa, H. 1986. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME J. Dyn. Sys. Meas. Control* 108(3):163–171.

Nakamura, Y., Hanafusa, H., and Yoshikawa, T. 1987. Task-priority based redundancy control of robot manipulators. *Int. J. Robot. Res.* 6(2):3–15.

Nash, J. C. 1975. A one-sided transformation method for the singular value decomposition and algebraic eigenproblem. *Comp. J.* 18(1):74–76.

Nash, J. C. 1979. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation.* Bristol: A. Hilger.

Salisbury, J. K., and Craig, J. J. 1982. Articulated hands: Force control and kinematic issues. *Int. J. Robot. Res.* 1(1):4–12.

Sampei, M., and Furuta, K. 1987 (March 31–April 3, Raleigh, N.C.). Robot control in the neighborhood of singular points. *Proc. 1987 IEEE Int. Conf. Robot. Automat.* Silver Spring, Md.: IEEE Computer Society Press, pp. 1696–1700.

Schimmel, D. E., and Luk, F. T. 1986. A new systolic array for the singular value decomposition. In C. E. Leiserson (ed.): *Advanced Research in VLSI.* Cambridge, Mass.: MIT Press, pp. 205–217.

Stewart, G. W. 1970. Incorporating origin shifts into the QR algorithm for symmetric tridiagonal matrices. *CACM* 13(6):365–367.

Wampler, C. W. 1986. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Trans. Sys. Man Cybernet.* SMC-16(1):93–101.

Watkins, D. S. 1982. Understanding the QR algorithm. *SIAM Review* 24(4):427–440.

Wedin, P. A. 1972. Perturbation bounds in connection with singular value decomposition. *BIT* 12(1):99–111.

Whitney, D. E. 1969. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man–Mach. Sys.* MMS-10(2):47–53.

Yoshikawa, T. 1984. Analysis and control of robot manipulators with redundancy. In M. Brady and R. Paul (eds.): *Robotics Research: The First International Symposium.* Cambridge, Mass.: MIT Press, pp. 735–747.

Yoshikawa, T. 1985a. Manipulability of robotic mechanisms. *Int. J. Robot. Res.* 4(2):3–9.

Yoshikawa, T. 1985b. Dynamic manipulability of robot manipulators. *J. Robot. Sys.* 2(1):113–124.