# Simple Algorithms for the Partial Singular Value Decomposition

J. C. NASH
*Faculty of Administration, University of Ottawa, Ottawa, Ontario K1N 9B5, Canada*

S. SHLIEN
*Department of Communications, Government of Canada, Ottawa, K2H 8S2, Canada*

***The singular value decomposition (svd) is usually calculated by the Golub/Kahan QR algorithm. Two simple alternative methods for the partial or complete svd are described which are suited for special applications. Both of these can be implemented easily by the user, even on low-capacity microcomputers. Various properties and limitations of these two methods are discussed.***

*Received May 1985*

## 1. INTRODUCTION

Though direct methods of solving linear systems of equations and linear least-squares problems are computationally efficient, they may give unexpected results for ill-conditioned systems.[7, 22] The singular value decomposition (svd) provides a numerically stable means of solving such systems with additional information to diagnose the condition of the problem. The svd is particularly suited to certain applications in image processing,[3, 13] geophysical models,[24] data compression[1] and statistical analysis.[15]

Stable methods for computing the svd are usually based on Golub and Kahan's[10] application of Householder transformations to reduce a matrix to bidiagonal form, followed by QR transformations to obtain the diagonal matrix of singular values. Despite the efficiency and reliability of the Golub/Kahan method and its implementations following Golub and Reinsch,[12] it is not easily adapted to special problems or unconventional computing environments. In particular, there are applications where only a few of the dominant singular values/vectors of large matrices are of interest.[1] Simpler algorithms have merit in such situations where they may provide a more suitable approach given the human or machine resources available. Since the better-known software packages are largely directed to FORTRAN computing environments,[6, 8, 15] users of other programming languages may also find the Golub/Reinsch algorithm less readily accessible.

In many situations where the svd is computed, only the first few singular components are desired; smaller singular values are treated as negligible. The problem of interest is that of finding the $k$ largest singular values and associated vectors of a matrix. Several algorithms based on the Lanczos process have been developed;[5, 11] other methods have been proposed for use on computers which have special architectures.[16] Here we describe two algorithms for this task, either of which can be programmed in less than an hour. The first algorithm, which is based on the power method for eigenvalues and eigenvectors,[2] allows the computation of the singular values and vectors, one set at a time, in order of decreasing size of the singular values. The second algorithm is based on the Hestenes/Nash one-sided transformation method,[17, 18] but employs a new structure and angle calculation. The power method is particularly suited to large matrices of which only the dominant singular components are of interest. The transformation method is appropriate when the dimensions of the matrix are not too large and accurate calculation of the singular vectors is desired. Step-and-description pseudocodes for these algorithms are given in an appendix and several practical details and properties are illustrated by example.

## 2. THE SINGULAR VALUE DECOMPOSITION

The singular value decomposition (svd) of a real $m$ by $n$ matrix $A$ with $m > n$ can be written as the product

$$A = USV^T, \tag{2.1}$$

where the $m$ by $n$ matrix $U$ and the $n$ by $n$ matrix $V$ satisfy

$$U^TU = V^TV = VV^T = 1_n \tag{2.2}$$

and $S$ is an $n$ by $n$ diagonal matrix whose elements can be ordered without loss of generality into decreasing order of magnitude. The diagonal elements of $S$ are the singular values, and the column vectors of $U$ and $V$ corresponding to each singular value are the associated left and right singular vectors of matrix $A$. If $u_i$ and $v_i$ are the $i$th column vectors of $U$ and $V$ respectively, then the singular value decomposition of $A$ may also be written as the outer product expansion

$$A = \sum_{i=1}^{n} S_{ii}\, u_i\, v_i^T \tag{2.3}$$

where the rank-one matrix

$$S_{ii}\, u_i\, v_i^T$$

is called the $i$th singular plane. The series can be truncated to yield the least-squares approximation to $A$ with rank $K$ less than $n$. Other outer product expansions exist.[20]

It should be noted that there are other forms for the $U$, $S$ and $V$ matrices in the singular value decomposition. In particular, $U$ may be written as a full $m$ by $m$ orthogonal matrix so that $S$ must be augmented by $(m-n)$ rows of zeros and becomes an $m$ by $n$ matrix. Our methods do not compute this form of the svd. The problem of interest is that of finding the $k$ largest singular values and associated vectors of matrix $A$ for $k < n \leqslant m$.

## 3. POWER METHOD

The computation of singular values and vectors is closely related to the matrix eigenvalue problem for a real, symmetric matrix. Indeed, the svd may be rephrased as an eigenvalue/eigenvector decomposition

$$\begin{bmatrix} 0_m & A \\ A^T & 0_n \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} S \tag{3.1}$$

where $0_m$ and $0_n$ are $m$ by $m$ and $n$ by $n$ matrices of zeros. Since the matrix

$$G = \begin{bmatrix} 0_m & A \\ A^T & 0_n \end{bmatrix} \tag{3.2}$$

is symmetric, its eigenvalues are all real, so that difficulties associated with complex eigenvalues are not present in the svd process. A single eigensolution of $G$ satisfies the pair of equations

$$\begin{aligned} Av &= us \\ A^T u &= vs \end{aligned}, \tag{3.3}$$

where $u$ and $v$ are of unit length. There may be up to $n$ solutions of the form $(u, v, s)$ and we distinguish these by subscripts where necessary in the work below. The method[23] determines the solution by performing the following iteration

$$\begin{aligned} u^{(k+1)} &= \frac{Av^{(k)}}{\|Av^{(k)}\|} \\ v^{(k+1)} &= \frac{A^T u^{(k+1)}}{\|A^T u^{(k+1)}\|} \\ s^{(k+1)} &= \|A^T u^{(k+1)}\| \end{aligned} \tag{3.4}$$

starting with a random guess $v^{(0)}$ and continuing until

$$\|u^{(k+1)} - u^{(k)}\| \tag{3.4a}$$

is small.

To compute other singular planes, a deflation is performed. To find the second singular plane we form

$$A' = A - S_{11} u_1 v_1^T, \tag{3.5}$$

where the subscript 1 refers to the first set of solution vectors and singular value found by the power method, so that by expansion (2.3) $A'$ no longer includes the first singular plane. The second singular plane is computed by performing the same iteration as above with the deflated matrix $A'$. By virtue of Equation (3.4), the deflation guarantees

$$A'^T u^{(k+1)} = 0, \tag{3.6}$$

whether or not the power iteration (3.4) was carried to convergence, so that the remaining singular vectors $u$ will be orthogonal to the last $u$ iterate, that is,

$$u^{(k+1)}.$$

Without iterating (3.4) to convergence, it follows that $A$ can be factored into the product

$$A = \tilde{U}\tilde{S}\tilde{V}^T, \tag{3.7}$$

where $\tilde{U}$ is orthogonal and $\tilde{S}$ is diagonal. The tilde ($\tilde{}$) is used to show that the matrices do **not** give the true svd. $\tilde{V}$ approaches orthogonality with convergence of (3.4$a$) as illustrated below. Let

$$s^{*(k+2)} = \|Av^{(k+1)}\| \tag{3.8}$$

then

$$\left.\begin{aligned} A'v^{(k+1)} &= (A - s^{(k+1)} u^{(k+1)} v^{T(k+1)}) v^{(k+1)} \\ &= Av^{(k+1)} - s^{(k+1)} u^{(k+1)} \\ &= s^{*(k+2)} u^{(k+2)} - s^{(k+1)} u^{(k+1)} \\ &\simeq s^{(k+1)} (u^{(k+2)} - u^{(k+1)}) \end{aligned}\right\} \tag{3.9}$$

which approaches zero as the power iteration process converges. Thus $\tilde{U}$ is orthogonal, $\tilde{S}$ is diagonal, but the orthogonality of $\tilde{V}$ depends on the convergence criterion chosen.

The rate of convergence of the iteration, like the power method for the equivalent eigenproblem, depends on the ratio of adjacent singular values.[23] For low-rank matrices, this ratio is usually far from unity (though this cannot be guaranteed), suggesting rapid convergence. The convergence rate is not overly sensitive to the dimensions of the matrix, so that the method is particularly suited to problems involving large matrices with only a few non-zero singular values.[23] Such matrices arise in image processing and data compression applications noted above. Experimental results also suggest that convergence errors do not build up in the deflation procedure but are removed as subsequent singular planes are computed.

A potential problem with this power method for the svd is the deflation process if the small singular values and their associated vectors are required. After the major singular planes have been removed, the deflated matrix may be dominated by round-off noise. This will happen whenever the ratio of the smallest to the largest of the singular values approaches the precision of the arithmetic. This prevents the accurate determination of the svd. Thus, the power method is not recommended for computing the generalized inverse of a matrix, but it is suitable for data compression applications, for example, image encoding.[1]

## 4. PLANE ROTATION METHOD

The Hestenes/Nash one-sided transformation method[17, 18] for the complete svd can be modified to compute a partial svd when the matrix $A$ is rank-deficient. In a manner analogous to the Jacobi method for the eigensolutions of a real, symmetric matrix, this method operates on the matrix $A$ with a sequence of plane rotations to make the columns of the resulting matrix orthogonal. The implementation described here departs from the two-by-two Jacobi steps employed by Brent, Luk and Van Loan for parallel computations.[4] Let the rotation matrix to rotate columns $i$ and $j$ of $A$ by angle $\phi$ be called $R$, then the four elements of $R$ which are different from those of an identity matrix are

$$\left.\begin{aligned} R_{ii} &= R_{jj} = \cos(\phi) \\ R_{ij} &= -R_{ji} = \sin(\phi) \end{aligned}\right\} \tag{4.1}$$

The product of the rotations becomes the orthogonal matrix $V$, so that

$$AV = B, \tag{4.2}$$

where $B$ has orthogonal columns. The angle $\phi$ in the rotation is calculated so that in operating on columns $i$ and $j$ of the (intermediate) matrix $A$ with $i < j$, the transformed column $i$ has magnitude at least as great as its untransformed counterpart.[17] This has the effect of ordering the magnitudes of the columns of $B$, leaving

$$B^T B = S^2. \tag{4.3}$$

The singular values are the column norms of matrix $B$ and the $U$ matrix is obtained by simple normalisation of $B$. Therefore

$$AV = US, \tag{4.4}$$

which is equivalent to (2.1).

Naturally, if one of the singular values is zero, say the $k$th, then the $k$th column of $B$, call it $b_k$, cannot be normalised. However, this can only be the case if this column is null and the matrix $A$ is rank-deficient. This provides a mechanism by which the computation of the svd in such cases may be accelerated. The algorithm is organised to perform the rotations in a pattern or sweep which covers the whole matrix. A common sweep pattern is the 'row cyclic' choice, which takes column pairs

$$\begin{array}{l}(1,2)(1,3)\ldots\quad(1,n)\\ \qquad(2,3)\ldots\quad(2,n)\\ \qquad\qquad\ldots\\ \qquad\qquad\qquad(n-1,n)\end{array} \tag{4.5}$$

That is, $n(n-1)/2$ rotations are performed in an entire sweep.

Convergence is assumed when the angle of rotation is 'small' for all rotations in one sweep. If the magnitude of a column is zero, the rotation angle will be zero if this column is the right-hand member of the current column pair. The rotation angle for such a column will be $\pi 1/2$ if it is the left-hand member of a pair of columns and the other is not null. After a complete sweep, such a column will be at the right-hand side of the working matrix. Since it can no longer contribute to rotations, it is now possible to reduce the number of rotations in a sweep by reducing the column dimension of the working matrix

$$n' = n-1, \tag{4.6}$$

thus saving $(n-1)$ rotations in a sweep.

In the angle calculation, care has been taken to avoid unnecessary subtractions of nearly equal quantities. Rotation angles may exceed the usual $\pi/4$ limit suggested in proofs of convergence of the Jacobi algorithm.[7] Recently Hoy Booker of American University has shown that in all situations where a tolerance is used to judge if the rotation angle $\phi$ is 'small', the algorithm will converge. (This corresponds to the 'threshold' Jacobi method.) He has also shown some cases of the exact algorithm (that is, with the rotation ignored only when $\phi = 0$ precisely, and all arithmetic to infinite precision) to be convergent.

## 5. PERFORMANCE, PROPERTIES AND MODIFICATIONS

As discussed in Section 3, convergence properties of the power method are affected by the distribution of the singular values but are less sensitive to the size of the matrix. To illustrate the convergence of the two methods presented here, test matrices of various orders were produced by application of pseudo-random plane rotation sequences to diagonal matrices whose elements (singular values) follow a geometric progression.[19] Both the ratio of adjacent singular values and the size of the matrices were varied for this study, but all matrices reported are square ($m = n$). Computations were performed on a PDP 11/60 computer with floating-point hardware using FORTRAN-coded programs. Similar results were obtained in BASIC on a North Star Horizon microcomputer.

The convergence properties were measured by determining the degree of non-orthonormality of the $u$ and $v$ singular-vector approximations as a function of the number of iterations or sweeps. The measure used is the mean square of the off-diagonal elements of the inner-product matrices

$$U^TU \text{ and } V^TV. \tag{5.1}$$

The exact orthonormality of the singular vectors is important for the computation of the generalised inverse or the solution of a system of linear equations.

From Fig. 1($a$) we see that the error in the power method is greatest when the singular values have a wide range of values (left-hand side of graph). This error, due to the deflation technique, is not dependent on the number of iterations used. Fig. 1($b$) shows the same deflation problem at the left-hand side of the graph, while the relatively large errors at the right-hand side are due to slow convergence of the power method when the matrix has approximately equal singular values. The slowing of convergence with approximately equal singular values follows the parallel analysis for the Jacobi eigenvalue algorithm (Ref. 21, page 182).

From the results shown in Fig. 1 and 2, we see that the Hestenes/Nash method is preferable when orthogonality of the $v$ vectors is important, as suggested in Section 3. Futhermore, for small matrices, Table 2 suggests this method to be the most efficient. Note that the use of the reduction in the column dimension of the working matrix is an important factor in reducing the computational workload and consequent execution time.

The convergence of the power method can be enhanced by replacing the random starting vector by an estimate of the singular vector. The vector formed by the difference

$$v^{(k+1)} - v^{(k)} \tag{5.2}$$

tends to point in the direction of the next singular vector $v$ as the iterations progress. This difference is also of utility in testing for convergence of the iteration process. Using a starting vector derived from this difference, processing time for the power method was reduced by as much as a factor of 2. This technique is described in step 6 of the pseudocode in the appendix. A comparison of the convergence properties for one example matrix is given in Table 1 and Fig. 2.

Table 2 presents a comparison of execution times for the two algorithms proposed here and the Golub/Reinsch[12] algorithm as presented by Lawson and Hanson.[15] The Golub/Reinsch method is generally faster for this particular full svd, though the number of singular planes computed never exceeded the mantissa length of 24 bits, in keeping with the singular values decreasing in magnitude with the fixed ratio of 0.5 in the test matrix.

Table 3 presents a comparison of the important features of the three approaches.

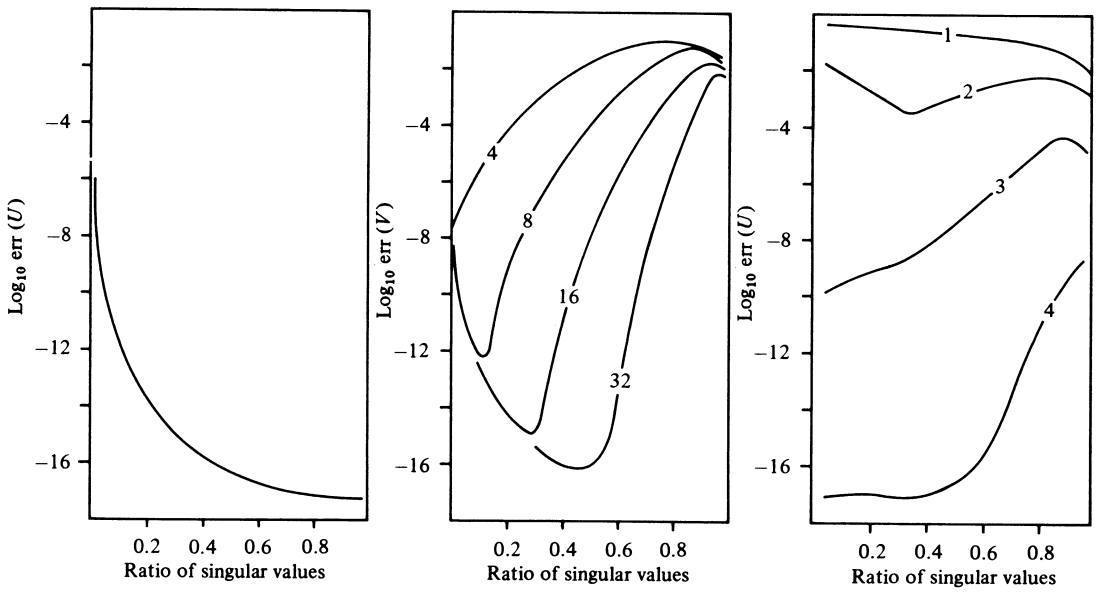
To illustrate the properties of the algorithms in an extreme situation, we compute the vector

$$x = L^{-1}L(1,1,\ldots 1)^T \tag{5.3}$$

where

$$L_{ij} = \exp(-t(i-1)j) \tag{5.4}$$

and $t$ is some parameter. The resulting system of equations is highly ill-conditioned, so that neither direct matrix inversion nor the svd should be considered




Off-diagonal mean-square errors for the svd of an 8 by 8 matrix generated by random plane rotations applied to a diagonal matrix whose elements are

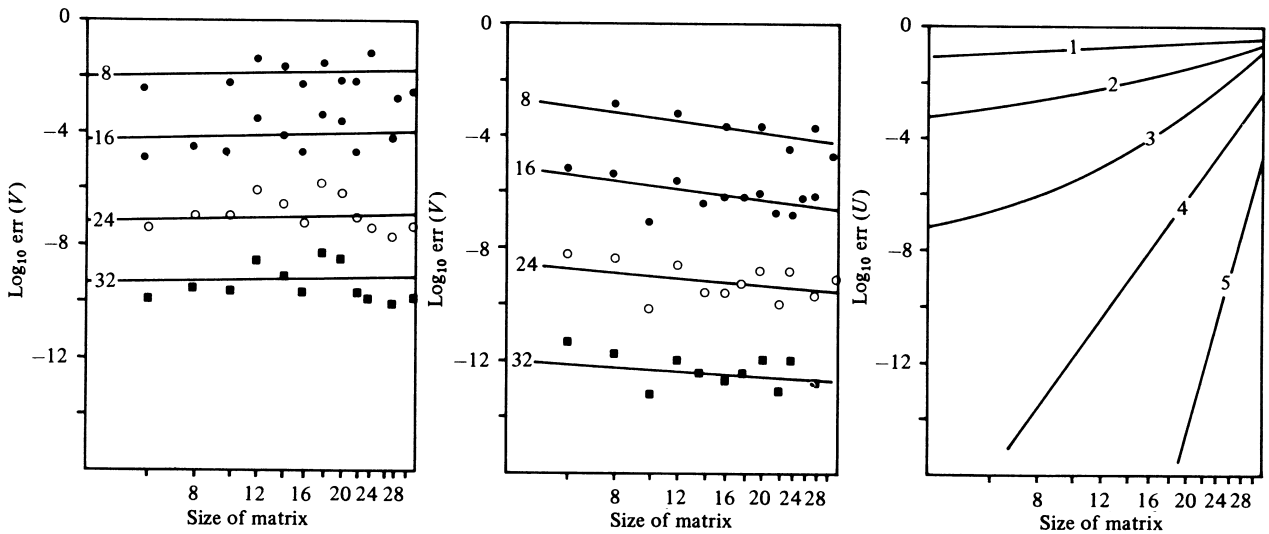
$$A(i, i) = r ** (i-1)$$

(*a*) Power method. Base 10 logarithm of the mean-square off-diagonal error for the $U$-matrix versus the ratio of the singular values, $r$.

(*b*) Power method. Base 10 logarithm of the mean-square off-diagonal error for the $V$-matrix versus $r$ for selected numbers of iterations.

(*c*) Transformation method. Base 10 logarithm of the mean-square off-diagonal error for the $U$-matrix versus $r$ for 1 to 4 sweeps.

**Fig. 1. Condition number effects.**




Convergence of svd for a square matrix generated by random plane rotations applied to a diagonal matrix with elements $0.7^{i-1}$, $i = 1, 2, \ldots, n$ versus $n$, the order of the matrix.

(*a*) Power method with random starting vector. Base 10 logarithm of the mean-square off-diagonal error for the $V$-matrix versus $n$ for selected numbers of iterations.

(*b*) Power method with estimated starting vector. Base 10 logarithm of the mean-square off-diagonal error for the $V$-matrix versus $n$ for selected numbers of iterations.

(*c*) Transformation method. Base 10 logarithm of the mean-square off-diagonal error for the $U$-matrix versus $n$ for 1 to 5 sweeps.

**Fig. 2. Size effects.**

appropriate for this type of problem except as a source of test results. Letting

$$y = L(1, 1, \ldots, 1)^T$$

$x$ is given in terms of the singular planes as

$$x = \sum_{i=1}^{n} v_i u_i^T \frac{y}{S_{ii}}, \qquad (5.5)$$

where the singular values and vectors are those for the matrix $L$. As the summation above is performed, the typical approximation to $x$ first approaches the desired result (all ones), then begins to diverge as poorly determined singular planes are included, that is, as an incorrect decision as to the effective rank of $L$ is made. Table 4 compares results for the Golub/Reinsch code along with the two methods of this paper for this test

**Table 1. Convergence of the $v$-vector in the power method as measured by the norm of the difference between successive $v$-vectors**

Note that the use of an estimate of 'guess' for the starting $v$-vector reduces the number of iterations to reduce this norm below a tolerance of E-6. Calculations are for an 8 by 8 matrix generated by applying random plane rotations to a diagonal matrix whose elements are $2^i$, $i = 1, 2, \ldots, 8$.

| Singular plane | Random starting vector | Estimated starting vector |
|---|---|---|
| 1 | 0.3 | 0.3 |
| | 0.07 | 0.07 |
| | 0.02 | 0.02 |
| | 0.004 | 0.004 |
| | 0.001 | 0.001 |
| | 0.0003 | 0.0003 |
| | 0.00006 | 0.00006 |
| | 0.00002 | 0.00002 |
| | 0.000004 | 0.000004 |
| | 0.000001 | 0.000001 |
| | 0.0000003 | 0.0000003 |
| 2 | 0.1 | 0.00000007 |
| | 0.02 | |
| | 0.004 | |
| | 0.0009 | |
| | 0.0002 | |
| | 0.00006 | |
| | 0.00001 | |
| | 0.000004 | |
| | 0.0000008 | |
| 3 | 0.04 | 0.000002 |
| | 0.01 | 0.0000004 |
| | 0.003 | |
| | 0.0006 | |
| | 0.0002 | |
| | 0.00004 | |
| | 0.00001 | |
| | 0.000002 | |
| | 0.0000006 | |
| 4 | 0.13 | 0.001 |
| | 0.03 | 0.0002 |
| | 0.008 | 0.00006 |
| | 0.002 | 0.000004 |
| | 0.0005 | 0.0000009 |
| | 0.0001 | |

**Table 2. Comparison of processing times (seconds) to compute the complete svd for matrices generated by applying random plane rotations to diagonal matrices having elements $2^{-i}$, for $i = 1, 2, \ldots, n$, where $n$ is the order of the matrix**

Calculation performed on a PDP 11/60 with floating-point processor (approximately 10,000 additions or multiplications per second) in FORTRAN under the RSX11M operating system. Convergence tolerances equivalent to 8 significant figures.

| Order $n$ | Power method | Plane rotation method | Golub/Reinsch method |
|---|---|---|---|
| 8 | 2.75 | 1.22 | 1.43 |
| 12 | 7.7 | 4.4 | 4.2 |
| 16 | 17.7 | 10.4 | 9.2 |
| 20 | 31.5 | 25.4 | 16.3 |
| 24 | 57.3 | 42.1 | 25.9 |
| 28 | 83.3 | 57.9 | 38.9 |
| 32 | 98.5 | 78.3 | 57.0 |
| 36 | 134 | 103 | 79.5 |
| 40 | 148 | 130 | 107 |
| 44 | 177 | 169 | 142 |
| 48 | 212 | 208 | 185 |

**Table 3. Comparison of properties of svd algorithms**

| | Algorithm | | |
|---|---|---|---|
| Property | Power method | Hestenes/Nash | Golub/ Reinsch |
| Reconstruction of $A = USV^T$ | Good | Good | Good |
| Orthogonality of $U$-matrix | Good if deflation error does not dominate | Error proportional to largest singular value | Good |
| Orthogonality of V-matrix | Good if deflation error does not dominate | Good | Good |
| Run-time efficiency | Best for low-rank large matrices | Best for low-rank and/or small matrices | Best for general matrices |

**Table 4. Absolute value of element of maximum modulus in the vector $L^{-1}LE - E$ where $E = (1, 1, \ldots, 1)^T$ where $L$ is the matrix whose $(i, j)$ element is $\exp(-t\,(i-1)\,j)$ and $t$ is some parameter (given below)**

All calculations on a PDP 11/60 (see Table 2). Figure in parentheses after element size is the number of singular planes calculated.

| Order of matrix | $t$ | Direct inversion approach | svd Power method | svd Plane rotation method | svd Golub/ Reinsch method |
|---|---|---|---|---|---|
| 4 | 1.0 | 0.00025 | 0.013 (4) | 0.00007 (4) | 0.0002 (4) |
| | 2.0 | 0.015 | 0.17 (3) | 0.016 (4) | 0.0017 (4) |
| | 3.0 | 7.0 | 0.95 (3) | 0.05 (3) | 0.05 (2) |
| 6 | 1.0 | 0.5 | 0.33 (4) | 0.38 (5) | 0.39 (5) |
| 8 | 0.2 | 0.33 | 0.02 (5) | 0.004 (7) | 0.005 (7) |
| | 0.4 | 0.31 | 0.20 (5) | 0.23 (7) | 0.020 (7) |
| | 0.6 | 200.0 | 0.5 (5) | 0.11 (7) | 0.23 (7) |
| 16 | 0.1 | 'Singular matrix' | 0.05 (6) | 0.002 (8) | 0.005 (8) |
| 32 | 0.05 | — | 0.07 (7) | 0.002 (10) | 0.005 (10) |

example. The Hestenes/Nash method appears to be the most reliable for this particular application.

## 6. CONCLUSION

Two simple methods for the partial svd of a matrix have been presented. These are easily adapted to unusual problems or computer environments. While their general application will yield results inferior to the Golub/Reinsch method, in a number of situations they may be preferred, particularly when adaptation is necessary to accommodate special situations facing the user.

**Acknowledgements**

We are indebted to Dr J. H. Wilkinson for his comments on the Hotelling deflation we have used, and for pointing out that the full error analysis does not appear to have been published.

## REFERENCES

1. N. N. Abdelmalek, T. Kasvand and D. Goupil, Fingerprint data compression. *Seventh International Conference on Pattern Recognition. IEEE Computer Society*, Silver Spring, MD, 834–836 (1984).
2. F. E. Acton, *Numerical Methods That Work*. Harper & Row, New York (1970).
3. W. C. Andrews and C. L. Patterson, Singular value decomposition and digital image processing. *IEEE Trans. Acoustics, Speech and Signal Processing*, **ASSP-24**, 26–53 (1976).
4. R. Brent, F. T. Luk and C. Van Loan, Computation of the singular value decomposition using mesh-connected processors. *J. VLSI and Computer Systems 1*, 242–270 (1985).
5. J. Cullum and R. A. Willoughby, *Computing Singular Values and Corresponding Singular Vectors of Large Matrices by Lanczos Tridiagonalization*. Report RO 8200, IBM Thomas J. Watson Research Center, Yorktown Heights (1980).
6. J. J. Dongarra *et al. LINPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia (1980).
7. G. E. Forsythe and P. Henrici, The cyclic Jacobi method for computing the principal values of a complex matrix. *American Mathematical Society Transactions*, **94**, 1–23 (1960).
8. B. S. Garbow *et al. Matrix Eigensystem Routines – EISPACK Guide Extension* (Lecture notes in Computer Science, vol. 51). Springer Verlag, New York (1977).
9. N. Garguir, Comparative performance of SVD and adaptive cosine transforms in coding images. *IEEE Transactions on Communications*, **27**, 1230–1234 (1979).
10. G. H. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, series B, **2**, 205–224 (1965).
11. G. H. Golub, F. T. Luk and M. L. Overton, A block Lanczos method for computing singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software* **7**, 149–169 (1981).
12. G. H. Golub and O. Reinsch, Singular value decomposition and least squares solutions. *Numerische Mathematik* **14**, 403–420 (1970).
13. T. S. Huang and P. M. Narendra, Image restoration by singular value decomposition. *Applied Optics* **14**, 2213–2216 (1975).
14. E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. Wiley, New York (1966).
15. C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ (1974).
16. F. T. Luk, Computing the singular value decomposition on the Illiac IV. *ACM Transactions on Mathematical Software* **7**, 524–539 (1980).
17. J. C. Nash, A one-sided transformation method for the singular value decomposition and algebraic eigenproblem. *The Computer Journal* **18**, 75–76 (1975).
18. J. C. Nash, *Compact Numerical Methods for Computers*. Adam Hilger, Bristol; Halsted Press, New York (1979).
19. J. C. Nash and R. L. C. Wang, Subroutines for testing progams which compute the generalized inverse of a matrix, accepted *ACM Transactions on Mathematical Software* (1986).
20. D. P. O'Leary and S. Peleg, Digital image compression by outer product expansion. *IEEE Transactions on Communications* **COM-31** (3), 441–444 (1983).
21. B. N. Parlett, *The Symmetric Eignenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ (1980).
22. G. W. Stewart, *Introduction to Matrix Computations*. Academic Press, New York (1973).
23. S. Shlien, A method for computing the partial singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-4**, 671–676 (1982).
24. R. A. Wiggins, The general linear inverse problem: implication of surface waves and free oscillations for earth structure. *Review of Geophysics and Space Physics* **18**, 251–285 (1972).

## APPENDIX

### A1. POWER METHOD FOR THE PARTIAL SVD OF AN *M* BY *N* MATRIX *A*

Step Description

0 Comment: initialisation
Provide program with
*A*, the matrix
*m*, *n*, row and column size of *A*
eps, the machine precision = radix ** (1-number of radix digits)
tol, a tolerance for deciding that a singular value is zero
delta, a convergence criterion (see note 1)
climit, the maximum number of iterations per singular plane
klimit, the maximum number of singular planes to compute
*p*, *q*, working vectors of length *m*
*r*, working vector of length *n*
*sv*, *U*, *V*, vector and arrays to store the singular values and vectors, dimensioned klimit, *m* by klimit, *n* by klimit respectively.
Set $k = 1$; comment; for first singular plane.
Let $sv(1) = 0$; comment: to avoid undefined value in step 7.

1 Initialise $r(i)$ = random (0, 1) for $i = 1, 2, \ldots, n$.
Let $s = sqrt(r\text{-transpose} * r)$
Let $p(i) = r(i)/s$ for $i = 1, 2, \ldots, n$. Comment: normalise *r* into *p*. Vector *p* is the initial guess to the right singular vector.

2 Let counter = 0; comment: iteration counter for *k*th plane.

3 Let counter = counter + 1.
If counter > climit, abort; comment: too many

Step Description

iterations. (May instead go to step 7, if current approximation to the $k$th singular plane is considered adequate)

4. Compute $q = A * p$; comment: left matrix multiplication.
Let $s = sqrt(q\text{-transpose} * q)$
Let $q(i) = q(i)/s$ for $i = 1, 2, \ldots,$ m. Comment: normalize $q$ on to itself. $q$ eventually becomes the left singular vector.

5 Compute $r = A\text{-transpose} * q$; comment: matrix multiplication.
Let $s = \text{sqrt}(r\text{-transpose} * r)$
Let $r(i) = r(i)/s$ for $i = 1, 2, \ldots, n$: comment: normalize $r$ on to itself, $r$ eventually becomes the right singular vector.

6 Comment: convergence test.
Let $d = 0$.
For $i = 1$ to $n$; let $d =$ maximum $(d, \text{abs}(p(i) - r(i)))$; end loop on $i$. If d $\leqslant$ delta, goto step 8; comment: difference between iterates small, so assume vector converged. Here we test only on the $v$ vector. If $d > 10 * \text{eps}$, then let $w(i) = p(i) - r(i)$ for $i = 1, 2, \ldots, n$: comment: save for starting vector of next singular plane. The value 10 is a heuristic choice.
Let $p = r$; comment: copy current approximation to $v$ vector. This can be included in loop for $w$.
Goto step 3.

7 Comment: test for rank-deficient matrix.
If $s/(sv(1) + \text{tol}) < \text{tol}$, then go to step 12; comment: note the use of tol in the denominator to avoid a zero divide. A null matrix may yield a small $s$. Since $sv(1)$ has been intialised to zero, this test will perform correctly, with neither over- nor under-flow, given a reasonable value for tol, and a matrix $A$ with reasonable scaling.

8 Comment: convergence achieved, store results.
$su(k) = s$; comment: save singular value.
For $i = 1$ to $m$; let $U(i, k) = q(i)$; end loop on $i$.
For $i = 1$ to $n$: let $V(i, k) = r(i)$; end loop on $i$.
Comment: save singular vectors.

9 If $k \geqslant$ klimit. goto step 12: comment: computed all requested singular planes.

10 Comment: deflate matrix. Hotelling's deflation is used here for simplicity. Other deflations are possible (Ref. 22, ch. 5).
For $i = 1$ to $m$
For $j = 1$ to $n$
Let $A(i, j) = A(i, j) - s * q(i) * r(j)$
End loop on $j$
End loop on $i$.

11 Comment: normalise starting vector for next iteration.
Let $s = sqrt(w\text{-transpose} * w)$
If $s <$ eps, goto step 1; comment: watch out for noise only.
Let $p(i) = w(i)/s$ for $i = 1, 2, \ldots, n$.
Goto step 2.

12 Stop. Comment: may now test results.

## A2. HESTENES-NASH PLANE ROTATION METHOD FOR SVD OF $M$ BY $N$ MATRIX $A$

Step Description

0 Comment: initialisation
Provide program with
$A$, the matrix
$m$, $n$, row and column size of $A$
eps, the machine precision = radix ** (1-number of radix digits)
slimit, the maximum number of sweeps allowed. A suggested value of slimit is max ([$n$/4], 6).
Let $nt = n$; comment: current estimate of rank.
Let tol = eps; comment: suggested rank test tolerance.

1 Comment: initialise $V$ matrix to identity.
For $i = 1$ to $n$.
For $j = 1$ to $n$: let $V(i, j) = 0$; end loop on $j$.
Let $V(i, i) = 1$.
End loop on $i$.

2 Let rcount $= nt * (nt - 1)/2$; comment: counter set to number of plane rotations performed in a sweep. Counts down from maximum.
Let scount = scount + 1
If scount > slimit, then stop. Comment: too many sweeps.

3 Comment: main body of procedure to step 15.
For $j = 1$ to $(nt - 1)$.

4 For $k = (j + 1)$ to $nt$. Comment: inner sweep loop.

5 Let $p = 0$; let $q = 0$; let $r = 0$.
Comment: compute required column sums. Note that some machines do not underflow to zero without error trap. For such machines special code must be used to avoid the error trap. Note the use of multiplication instead of powering for the square of numbers for speed and reliability with interpretive language processors.
For $i = 1$ to $m$.
Let $p = p + A(i,j) * A(i,k)$
Let $q = q + A(i,j) * A(i,j)$
let $r = r + A(i,k) * A(i,k)$
End loop on $i$.

6 Comment: save squares of singular value approximations.
Let $z(j) = q$; let $z(k) = r$.

7 If $q \geqslant r$ then goto step 9, Comment: column magnitudes are in order if test is true.

8 Comment: angle calculations when magnitudes of columns are not in order.
Let $q = (q/r) - 1$: comment: this will be negative.
Let $p = p/r$.
Let $vt = sqrt(4 * p * p + q * q)$
Let $s = sqrt(0.5 * (1 - q/vt))$
If $p < 0$, then let $s = -s$; comment; sine of rotation angle.
Let $c = p/(vt * s)$: comment: cosine.
Goto step 11; comment: proceed to rotation, which must be performed since columns not in order.

9 Comment: order of column magnitudes is correct, so test for convergence, first on column magnitude, then on orthogonality.
If $q * r \leqslant \text{eps} * \text{eps}$, then goto step 13. Comment:

Step Description

may give trouble on machines which do not underflow to zero. Not scaled to norm of matrix $A$, so may give incorrect results for matrices with large or small elements.

If $(p/q)*(p/r) \leqslant$ eps, then goto step 13. Comment: test for unnecessary rotation. The tolerance may be adjusted to force a more or less stringent test.

10 Comment: angle calculation when no exchange of order is required.

Let $r = 1-(r/q)$; comment: safe division.

Let $p = p/q$.

Let $vt = sqrt(4*p*p+r*r)$

Let $c = sqrt(0.5*(1+r/vt))$; comment: cosine of angle.

Let $s = p/(vt*c)$; comment: sine of angle.

11 Comment: rotation of columns of matrix $A$.

For $i = 1$ to $m$.

Let $r = A(i,j)$; let $A(i,j) = c*r+s*A(i,k)$; let $A(i,k) = -s*r+c*A(i,k)$.

End loop on $i$.

12 Comment: rotation of columns of matrix $V$.

For $i = 1$ to $n$.

Let $r = V(i,j)$; let $V(i,j) = c*r+r*V(i,k)$; let $V(i,k) = -s*r+c*A(i,k)$.

End loop on $i$.

Goto step 14; comment: end of rotation.

13 Let rcount = rcount $-1$; comment: decrement rotation count since rotation has not been necessary.

14 End loop on $k$: comment: end of inner loop.

15 End loop on $j$: comment: end outer loop.

16 Print 'end of sweep', scount, 'number of rotations performed = ', rcount.

Step Description

17 If $nt < 3$, then goto step 19. Comment: before trying to reduce rank, must be sure it is not already small. Also, we must force the column sums $z(i)$, $i = 1, 2, \ldots, n$ to be computed for $m$ by 2 matrices via a second sweep.

18 Comment: is rank reduction possible?

If $z(nt)/(z(1)+\text{tol}) > \text{tol}$, then goto step 19.

Comment: cannot reduce rank if test true. Note that the test works even if matrix is null. Tolerances should be adjusted for the norm of $A$. The test works on the squares of the singular values.

Else let $nt = nt-1$; the goto step 17. Comment: rank estimate reduced.

19 If rcount $> 0$, then goto step 2; comment: try again if any rotations performed in the current sweep.

20 Comment: complete the decomposition for non-zero singular values.

For $j = 1$ to $nt$; comment: loop over singular values found to be non-zero.

Let $q = sqrt(z(j))$; comment: singular value.

For $i = 1$ to $m$; let $U(i,j) = A(i,j)/q$: end loop on $i$.

Comment: normalise left-hand singular vector.

(This may overwrite A. For least squares calculations or computation of the pseudo-inverse, step 20 may be omitted and the decomposition in terms of the current $A$ matrix, matrix $V$, and the squares of the singular values $Z$ used.)

End loop on $j$.

21 Stop.

# Announcement

12–14 OCTOBER 1987

**Astronomy from Large Databases: Scientific Objectives and Methodological Approaches,** Garching bei München, West Germany

Ever-increasing quantities of data are available to the astronomer, generated by satellite-borne and by terrestrial instruments. Of necessity, large amounts of data demand special methods for their analysis. It is additionally the case that there is much to be gained from studying large, diverse datasets.

The aim of this conference, to be hosted by the Space Telescope – European Coordinating Facility, is to review and analyse new possibilities offered to astronomers by the current and future availability of (and easy access to) large astronomical databases. The conference if focused especially on methods applicable to such data collections, and on the astronomical problems which are most intimately related to large amounts of data.

The *Proceedings* will be published by the European Southern Observatory.

**Topics**

1. *Astronomical databases: current trends.* A review both of the infrastructure – storage media, networking – and of the principles of constructing astronomical databases and standards.

2. *Statistical analysis of complex databases.* The application of statistical and pattern recognition methods to complement the astrophysical analysis by running a preliminary investigation of the data, sorting out ideas, shedding a new 'objective' or 'independent' light on a problem, or pointing out aspects which would not come to light in a classical approach.

3. *Expert systems around astronomical databases.* As with the first topic above, this is to be a review of new developments in this field.

4. *Object classification problems.* Creating taxonomies of objects, studying spectral classification schemes, optimally discriminating between galaxies and stars, and other problems requiring classification methods as a tool.

5. *Astrophysics from large databases.* Survey work can be considered here, and also topics such as: variability, distance scale, large scale structures, and cosmological applications. As with other topics, this list is non-exhaustive.

*Scientific Committee*

P. Benvenuti (ST-ECF, Munich) (Chairperson)
P. Grosbøl (ESO, Munich)
A. Heck (CDS, Strasbourg) (Editor, *Proceedings*)
C. Jaschek (CDS, Strasbourg)
M. Johnston (STScI, Baltimore)
J. Mead (NASA, Goddard)
F. Murtagh (ST-ECF, Munich) (Editor, *Proceedings*)
B. G. Taylor (ESA, Noordwijk)
U. Zimmermann (MPE, Munich)

*Further details are available from*:

F. Murtagh, ST-ECF/ESA/ESO, Karl-Schwarzschild-Strasse 2, D-8046 Garching-bei-München, FRG. (Telephone: +49 89 32006298. Telex: 528 282 22 EO D.)