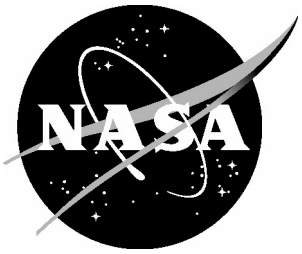
NASA/CR-2005-213500



A Survey of Singular Value Decomposition Methods and Performance Comparison of Some Available Serial Codes

Gerald E. Plassman
Raytheon Technical Services, Hampton, Virginia

July 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

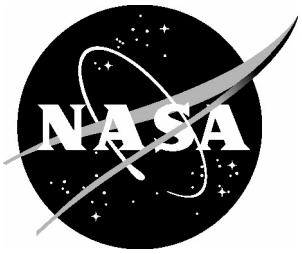
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR-2005-213500



A Survey of Singular Value Decomposition Methods and Performance Comparison of Some Available Serial Codes

Gerald E. Plassman
Raytheon Technical Services, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Purchase Order L-70750D

July 2005

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

A Survey of Singular Value Decomposition Methods and Performance Comparison of Some Available Serial Codes

Gerald E. Plassman, Raytheon

Introduction

A growing need for accurate and robust noise measurement has emerged over the past several years. To meet this need, NASA Langley Research Center has developed and tested instrumentation incorporating two-dimensional directional acoustic sensor arrays and post-processing procedures involving classical beam-forming of a discrete two-dimensional acoustic image based on array steering followed by image deconvolution to obtain high resolution maps of source noise.[1] Deconvolution for this application represents an inverse problem that typically requires the solution of a large and very ill-conditioned linear system characterized by a real-valued, dense, and mildly non-symmetric system matrix, A. While DAMAS[2], a most recent and promising Langley developed algorithm for solving this problem, incorporates an iterative solution method, singular value decomposition (SVD) remains a solution method of interest. In this report, the SVD requirement for a recently tested alternative source image deconvolution algorithm is addressed, and the computational bottleneck it represents in that application motivates a corresponding interest in identifying the most efficient methods and implementations for achieving SVD.

The serial computation of a complete SVD of a square matrix of order n by the classic method of Golub-Kahan-Reisch (GKR) requires $O(n^3)$ time.[3] The Numerical Recipes[4] implementation of this method, routine SVDCMP, is currently being used and, for problems with A of order n=1000, represents a computational bottleneck in SVD-based implementations of the deconvolution algorithm. In this algorithm, A is defined in terms of array steering vectors which depend on the size and relative orientation of the acoustic source grid with respect to the sensor array, the shear effects of the flow field, and the frequency of acoustic response addressed.[1] The desire to address sequences of many test problems with A of larger order, together with the knowledge that, given the above dependencies, A will generally not be identical over these sequences, identifies the criticality of identifying a more efficient SVD method.

The addressed alternative algorithm for source image deconvolution first applies SVD and then computes a regularized solution subject to a non-negativity constraint. SVD of A allows for solution expression as a singular value (SV) expansion. For such an A, solution regularization eliminates or reduces the magnitude, and noise-induced distorting effects, of all but the small number of leading (dominant SV) terms of a SV solution expansion. The non-negativity constraint adjusts the regularized solution if necessary to remove any physically non-meaningful negative elements or source pressure values.

This alternative deconvolution algorithm, using a truncated solution expansion form of regularization followed by a non-negative solution primal method by deVilliers etal.[5], produces encouraging results when tested on idealized point source synthetic data. A user-supplied threshold on SV magnitude determines the truncation point. The deVilliers non-negative solution method is formulated as a constrained quadratic programming problem on the coefficients of a full SV solution expansion, wherein the leading term coefficients are held fixed and the coefficients of the reintroduced non-leading terms are selected to yield the minimum 2-norm solution such that the sum of all their associated expansion terms are non-negative. This non-

negative solution method demonstrates the advantages of reproducing the original truncated expansion solution whenever that is non-negative, adding a minimum 2-norm extra solution component (the remainder of the expansion with altered coefficients) when required to achieve solution non-negativity, and generally improving solution resolution.

The primary purpose of the work reported in this paper is a performance comparison of available alternative complete SVD methods/implementations. A secondary purpose is a survey of a wider scope of alternatives, including partial SVD, special case SVD, and others developed for concurrent processing systems, with particular focus on computer clusters such as Beowulf systems.

The remaining sections of this paper will address the extent of the SVD requirement for the addressed SVD-based alternative deconvolution algorithm, a brief and selective summary of SVD methods currently available, a description of the serial full SVD method implementations selected for comparison with SVDCMP, a description of the test problems addressed during this comparison, and the results of that testing including the identification of Lapack 3.0 routine DGESDD as a recommended full SVD replacement for SVDCMP.

SVD Requirement

This section describes the extent of the SVD requirement for the addressed SVD-based alternative deconvolution algorithm.

The SVD of a real-valued square A, say n by n, the type of A obtained for the addressed application, can be expressed as the factorization, $A = U*W*V^T$, where W is diagonal with monotonically non-increasing positive elements, U and V are orthogonal (i.e. columns U_i of U are orthonormal and columns V_i of V are orthonormal), and all three are n by n. Since V is orthogonal, V^T, the transpose of V is also the inverse, V^{-1}, of V. Similarly, $U^T = U^{-1}$. Therefore, given the linear system Ax = b, where x and b are respectively the n by 1 vectors representing the unknown desired solution and known right hand side (beam-formed acoustic image in the addressed application), we can obtain solution x as follows:

$$Ax = b \rightarrow (U*W*V^T)x = b \rightarrow x = (U*W*V^T)^{-1}b \rightarrow x = V*W^{-1}U^Tb \quad (1)$$

where $W^{-1} = \mathrm{diag}(1/w_i)$ and w_i is the i'th diagonal element (SV) of W. Thus x can be expressed as the SV expansion of scalar multiples of the orthonormal columns of V:

$$x = \sum_i c_i * V_i \quad (2)$$

where scalars $c_i = U_i^T * b/w_i$, that is the inner products of the columns of U and b divided by the associated SV. Thus a truncated expansion, say the first k terms, requires only the first k elements of W and first k columns of U and V. Notice that the leading terms of this expansion are those corresponding to the dominant SVs, and also that the small w_i associated with the later terms will magnify any noise in those terms.

While equation (2) shows that solution regularization by truncated solution expansion requires only a partial SVD, that is the determination of only the leading k SVs (elements of W) and left and right hand singular vectors (columns of U and V), the deVilliers positive solution primal method requires the availability of the remaining n-k columns of V in order to both define the

constraints on the coefficients and evaluate the terms of the altered remainder of the expansion. The use of the classical regularization method of Tikhonov[6], an alternative to regularization by truncated solution expansion, employs a weighting of all the SV expansion terms based on SV magnitude, and therefore requires all W elements and U rows.

An alternative positive solution method defined by deVilliers etal is based on solving the dual of the primal method. This dual method alternative requires the solution of a non-linear problem of order k to obtain new coefficients of the k terms of the truncated expansion for x leading to a non-negative solution. The advantages of this dual method are reduced problem order and elimination of the requirement for columns of V beyond k. Its disadvantage is that computational cost in part scales as k^3 while that of the primal method scales as n. For $k << n$, as is typical for current applications of the addressed deconvolution algorithm, deVilliers etal recommend the dual method. For such cases, deVilliers etal demonstrate the ability of a quasi-Newton implementation of this dual method, including a simple remedy for potential non-convergence, to compute non-negative solutions comparable with the primal method in a fraction of the time.

In summary, the current implementation of the SVD-based alternative deconvolution algorithm, with solution regularization by truncated SV expansion and non-negative solution assurance by the deVilliers primal method, requires an SVD that provides only truncated W elements and U^T rows (i.e. leading U columns) but all (columns) of V, where U truncation corresponds to that of W as established by a user specified threshold on SVs. A new implementation, obtaining non-negative solution assurance by the deVilliers dual method would require only a partial SVD, i.e. only the k leading elements of W and k leading columns of both U and V. This suggests that any further investigation of an SVD-based alternative deconvolution algorithm has the potential to dramatically reduce the SVD computational cost and possibly overall algorithm cost by combining partial SVD computation with the deVilliers dual method.

SVD Methods

A selective literature review of SVD methods and their implementation provided a basic understanding of their applicability to and potential for use in the addressed SVD-based alternative deconvolution algorithm, and guided the selection of alternative codes for a performance evaluation against SVDCMP. This section outlines the knowledge gained by that review as it applies to serial computation of complete or partial SVD of a real-valued square A using direct (or transformational) methods. Methods for obtaining SVD by eigendecomposition of Hermitian (or symmetric for real-valued) matrices related to A are also described. The potential for a parallel implementation of iterative methods is only briefly addressed.

LAPACK/ScaLAPACK

A 1996 survey by Higham[7] of a decade of recent developments in dense linear algebra, including SVD, identifies LAPACK[8] as a state of the art (public domain) package of Fortran linear algebra software and successor to EISPACK and LINPACK. As such, LAPACK serves as a clearinghouse for available implementations of new linear algebra methods. LAPACK routines also make use of partitioned implementations of imbedded vector-vector through matrix-matrix operations, by means of potentially hardware optimized Basic Linear Algebra Subprograms (BLAS), that enable execution efficiency on high performance (hierarchical memory) computers over problem scale by preserving a high ratio of floating point operations to data movement. This

identifies LAPACK as a likely source of efficient alternatives to SVDCMP. LAPACK 3.0, the latest version, offers several such alternatives based on a range of SVD methods.

ScaLAPACK[9] is a scalable version of LAPACK designed for distributed memory parallel architectures, including networked clusters of processors, supporting message passing. ScaLAPACK utilizes a set of Basic Linear Algebra Communication Subprograms (BLACS) along with BLAS to enable efficiency and portability. ScaLAPACK provides only one of the LAPACK SVD methods, but for sufficiently large problems offers a potential for faster computation on parallel architectures with sufficient communication performance.

SVD by Eigendecomposition

A recent SIAM publication[10], serving as a practical guide for the solution of algebraic eigenvalue problems, provides a good description of current methods for both complete and partial SVD. It also identifies those that are available in LAPACK 3.0 or anticipated in a future version. It similarly addresses methods for computing the eigenvalues and eigenvectors of Hermitian (or symmetric for real-valued) matrices. These eigendecomposition methods can be used to obtain the SVD of any real-valued square A by computing the eigendecompostion of the symmetric matrix $S = A^T*A$, which is:

$$S = A^T*A = (U*W*V^T)^T*(U*W*V^T) = (V*W^T*U^T)*(U*W*V^T) = V*(W*W)*V^T \quad (3)$$

and is always symmetric and positive semidefinite. Thus the singular values, w_i, of A are the square roots of the non-negative eigenvalues, w_i^2, of A^T*A, while the right singular vectors, V_i, of A are the eigenvectors of A^T*A. The left singular vectors, U_i, of A can then be computed as $U_i = A*V_i/w_i$. SVD computation based on eigendecompostion is somewhat less efficient than SVD specific methods, but supplements the set of available codes, particularly with respect to partial SVD computation. A similar approach involves the eigendecomposition of $A*A^T$. These two approaches do not increase the order of S beyond that of A, but require significant pre- and post-processing (e.g. computations of $S = A^T*A$ and $U_i = A*V_i/w_i$, i =1,n) and may reduce the accuracy of computed small SV and associated singular vectors.

An alternative choice for the symmetric matrix S from which eigendecomposition methods can be used to obtain the SVD of any real-valued square A is S = H(A) which is defined as:

$$S = H(A) = \begin{vmatrix} 0 & A \\ A^T & 0 \end{vmatrix} = Q*Z*Q^T \quad (4)$$

Where: $Q = 1/\mathrm{sqrt}(2)*\begin{vmatrix} U & -U \\ V & V \end{vmatrix}$, $Z = \begin{vmatrix} W & 0 \\ 0 & -W \end{vmatrix}$; Z and W diagonal

This approach does not require significant pre- and post-processing nor reduces the accuracy of computed small SV and associated singular vectors, but results in an S of order twice that of A.

Direct SVD Options

Direct or transformational SVD specific methods, those intended for dense, typically non-symmetric (and in our case real square) matrices, A, typically involve the following three phases

when implemented for serial computation; where subscripts 1 and 2 on U and V indicate distinct orthogonal matrices rather than columns of U and V, and I is the identity matrix:

1) $U_1^T*A*V_1 = B$, $U_1^T*U_1 = V_1^T*V_1 = I$; orthonormal reduction to bidiagonal form, B
2) $U_2*W*V_2^T = B$, $U_2^T*U_2 = V_2^T*V_2 = I$; an SVD of bidiagonal form B
3) $U = U_1*U_2$ and $V = V_1*V_2 \rightarrow A = U*W*V^T$ an SVD of A

Phase 1 is typically computed by Householder reflections at a cost of $(8/3)*n^3$ floating point operations (flops). Phase 3 cost is also $O(n^3)$ flops. Phase 2 cost, however, depends both on the method used and the character of matrix A. A similar phase and associated flops cost breakdown is applicable to the related eigendecomposition for A^T*A; where phase 1 now results in a symmetric tridiagonal matrix at a cost of $(4/3)*n^3$ flops.

We now briefly describe the standard available phase 2 alternatives, some of which are only suitable for complete SVD. We also identify available LAPACK/ScaLAPACK implementations, limiting ourselves to those addressing the SVD of A when available, and identifying those addressing the SVD of A^T*A when that is the only option. A doctoral dissertation by Dhillon[11] provides (eigendecomposition) flops analyses of these standard phase 2 methods and presents a very promising new one, suitable for both partial and complete SVD and based on eigendecompostion of related hermitian matrices, which has the potential for O(n) cost per SVD triplet (w_i and associated U_i and V_i), compared with near $O(n^2)$ for existing alternatives. Reference 10 also provides flops analysis, including those for SVD directly from A.

Classic Complete SVD. Use of the classic QR algorithm for phase 2 of the complete SVD of A, long considered the best serial implementation, incurs a cost of $O(n^3)$ flops ($6*n^3$ when addressing A^T*A), including $O(n^2)$ expensive square root operations. This QR algorithm is implemented in the LAPACK computational routines xBDSQR (x=S or D for single or double precision) and called by their SVD driver's xGESVD. These drivers implement the classic GKR method for SVD by combining Householder based phase 1 with QR based phase2. This is the only direct SVD method supported by ScaLAPACK.

Divide and Conquer Based Complete SVD. A more recent phase 2 algorithm for complete SVD, based on a divide and conquer technique proposed by Cuppen for parallel implementation of the related hermitian (or symmetric) eigenproblem (eq. 3), is typically faster than other complete SVD phase 2 algorithms on serial computers as well. This divide and conquer approach recursively applies a technique of independently diagonalizing two equal sized sub-tridiagonal matrices followed by a coupling step. Recursion is continued until a sub-matrix order of about 25 is reached, at which point the classic QR is applied. The smaller serial cost stems from deflation, or the common recurrence of identical eigenpairs (eigenvalue and eigenvector) in the successive sub-bidiagonal/tridiagonal matrices. For the worst case, when no deflation occurs, $O(n^3)$ flops are required, but the typical flops requirement with deflation is much less. While the accuracy of classic QR is not guaranteed, it is typically achieved in practice. Mild assumptions about the implementation of floating point arithmetic are made, but modern processors typically meet them. This phase 2 algorithm is the fastest currently available in LAPACK specifically for the complete SVD problem. It is most efficient when eigenvalues are clustered. It is implemented in xBDSDC and called by SVD driver routines xGESDD. One drawback of this phase 2 algorithm is the significant amount (more than $8n^2$) of floating point workspace required, which could become prohibitive for sufficiently large problems.

Standard Partial SVD. When only a partial SVD is required, a bisection method of determining singular values followed by an inverse (power) iteration method to find the corresponding

singular vector pairs becomes attractive. When the desired singular values are isolated, this approach to solving the phase 2 algorithm can cost as little as O(n) flops per singular triplet. However, this per-triplet cost can increase to $O(nk^2)$, in achieving singular vector orthogonality, when the desired singular value is within a cluster of k-1 others. When many singular values are clustered, the total cost could approach $O(n^3)$ and singular vector orthogonality is not guaranteed. This phase 2 algorithm is not available in LAPACK for direct SVD of A, but SVD by the (eq. 3) related eigenproblem solution of A^T*A or the (eq. 4) related eigenproblem solution of H(A) by this algorithm is supported by the LAPACK driver xSYEVX, with the bisection and inverse iteration steps respectively performed by the computation routines xSTEBZ and xSTEIN. When eigenvalues are well separated each eigen-value/vector pair can be computed independently, yielding a simple and embarrassingly parallel implementation. Thus, despite the potential for heavy communication when computing orthogonal eigenvectors associated with closely clustered eigenvalues, the bisection/inverse iteration algorithm for the phase 2 solution remains popular for parallel implementation. ScaLAPACK supports such an implementation in the driver PxSYTRD and the computational routines PxSTEBZ and PxSTEIN.

Ultimate Partial/Full SVD. The "ultimate", $O(n^2)$ cost, phase 2 solution for complete SVD is theoretically achieved by both the fast multipole divide and conquer method of Carrier, Greengard, and Rokhlin and the recently developed algorithm by Dhillon[11]. No software for performing SVD with the former method has been identified. The Dhillon algorithm, also called the Relatively Robust Representation (RRR) algorithm, however, is implemented in the LAPACK computational routines xSTEGR in support of the hermitian (or symmetric) eigenproblem which can address A^T*A or H(A) to obtain an SVD of A. This RRR algorithm implementation is usually faster than the "divide and conquer" method implemented in xSTEDC (and in xBDSDC for SVD), and is expected to directly support SVD as well in a future release of LAPACK. While RRR is capable of supporting both partial and complete eigenproblems, LAPACK xSTEGR currently addresses only the complete case. The RRR algorithm represents an alternative way of performing inverse iteration, a very fast method for isolated eigenvalues that avoids the high expense of re-orthogonalizing the eigenvectors associated with clusters of eigenvalues. When clusters of eigenvalues are encountered, RRR abandons inverse iteration in favor of a matrix shift operation that permits a triangular factorization such that the small shifted eigenvalues, those in the cluster, are determined to high relative accuracy. Their eigenvectors are then accurately obtained by an optimal twisted (or double ended triangular) factorization. The low cost of these computations relies on recently discovered (differential qd) algorithms that produce the twisted factorizations without computing matrix products.

Jacobi SVD Methods

Jacobi methods for computing SVD are fundamentally different than those already discussed in that they seek (for real valued A) to obtain the factorization $A = U*W*V^T$ directly, to within some convergence criteria, by iterative application of elementary orthogonal transformations or Jacobi rotations. In effect it combines phases 1 and 2 of the identified direct SVD methods. The orthogonal transformations applied in phase 1 of these direct SVD methods retain introduced zero elements and thus attain the bidiagonal (or tridiagonal) B in a fixed number of steps determined by n, the order of the A. The phase 2 objective of achieving a diagonal W to some convergence criteria is then achieved by applying to B one of the methods described above. Each Jacobi rotation eliminates (zeros out) a single element of the matrix to which it is applied. Successive Jacobi rotations each eliminate some other single element, while possibly reintroducing non-zero values in the place of elements eliminated by preceding Jacobi rotations, but doing so in a manner that reduces the sum of the squares of all off-diagonal elements. Thus a sufficient but unknown

number of iterations (Jacobi rotations) will yield a diagonal W to within some convergence criteria in the least squared sense. Serial implementations of Jacobi methods are slower than serial implementations of any of the direct SVD methods, but have the potential to determine much more accurately the smallest SV and their associated singular vectors and are more amenable to parallelization.

Selected Codes

Code selection for a comparison with SVDCMP (NRsvdcmp) in terms of complete SVD computation accuracy and cost was limited to four routines. These routines, like SVDCMP, do not support partial SVD computation. The testing of partial SVD computation is not addressed in this paper. The first is the routine F02WEF (NAGf02wef) from the licensed NAG Fortran Library. The remaining three, all from the public domain LAPACK 3.0 library, include DGESVD (LPdgesvd), DGESDD (LPdgesdd) and DSYEVR (LPdsyevr). The alternate names given in parentheses identify the labels used in plotted test results to identify SVD computation method by source library and routine name. SVDCMP is a single precision code with critical computations implemented to retain double precision accuracy. The four routines selected for comparison are all implemented in double precision. All testing was performed on a DEC Alpha machine. The three LAPACK selections include the use of generic BLAS routines. The use of DEC Alpha optimized BLAS may result in lower computation costs for these LAPACK selections than those reflected in the reported test results.

Table 1 summarizes the characteristics of the six tested methods of SVD computation. All methods are suitable for the complete SVD of the order n, non-symmetric, square, and real-valued test matrix, A, typical of the addressed test problems. The first four methods listed represent the use of distinct routines addressing a direct SVD formulation. The last two listed represent the use of the same routine addressing alternate symmetric matrix, S, eigendecomposition formulations of the SVD computation, one addressing an order n $S = A^T*A$ and the other addressing an order 2*n S = H(A). The first method, SVDCMP, is the classic SVD implementation currently used. The second and third methods represent alternate implementations of the classic SVD method also employing the QR algorithm during phase 2. They were selected to ascertain the relative efficiency of SVDCMP. The last three methods employ algorithms that have the potential for significantly faster SVD computation. Their selection represents likely candidates for more efficient SVD computation. The first of these, using routine DGESDD, applies the divide and conquer algorithm in phase 2 of the SVD computation and requires significantly more workspace. The last two make use of the most promising RRR algorithm in the eigendecomposition of S, but require the overhead of forming S and retrieving the components of the SVD of A from the eigendecomposition of S. For the last method, there is also the significant impact of the doubled order of S with respect to A.

Table 1. Summary Characteristics of Tested SVD Computation Methods

Routine	Formulation	Phase 2 Method	Matrix	Order	Work Storage Flt/Int
NRsvdcmp	Direct SVD	Classic QR	A	n	0 (F), 0 (I)
NAGf02wef	Direct SVD	Classic QR	A	n	n^2+5n (F), 0 (I)
LPdgesvd	Direct SVD	Classic QR	A	n	5n (F), 0 (I) #
LPdgesdd	Direct SVD	Divide and Conquer	A	n	$8n^2$+4n (F), 8n (I)
LPdsyevr	Eigendecomposition	RRR (Dhillon)	$S=A^T*A$	n	n^2+6n (F), 12n (I)
LPdsyevr	Eigendecomposition	RRR (Dhillon)	S=H(A)	2n	$4n^2$+12n (F), 24n (I)

n^2+5n (F) provided since storage beyond the required 5n amount may improve performance

Test Problems

The test problems for evaluating the performance of the selected SVD computation methods identified in table 1 require the complete SVD of synthetic real-valued, dense, square, and generally non-symmetric matrices, A, of varied order, n, and condition number. All elements of the main diagonal of these test matrices, a(k,k), k=1,n are set equal to one, while the lower and upper off diagonal elements are defined as:

$$a(i,j) = 1/(2*(i-j))^{pl} \quad \text{for } i > j \text{ (lower)} \qquad (5)$$
$$a(i,j) = 1/(2*(j-i))^{pu} \quad \text{for } i < j \text{ (upper)}$$

The lower and upper power parameters pl and pu are positive and generally taken from the open interval (1,0). Those used were limited to reciprocals of integer powers of 10, with powers ranging from –2 to –9. For a given matrix order, smaller values of pl and pu yield a more ill-conditioned matrix. Equal values for pl and pu result in a symmetric matrix, while larger differences in pl and pu result in greater matrix asymmetry. As matrix order increases, the maximum SV approaches that order, while the minimum SV approaches the smaller of pl and pu. The ratio of maximum over minimum SV represents the condition number.

Two primary sets of test matrices, ranging in order from 100 to 1200, are addressed. Both are mildly asymmetric (magnitudes of pl and pu differ by a single order of magnitude). One uses pl = $1/10^2$ and pu = $1/10^3$ and is considered mildly ill-conditioned, with condition numbers ranging approximately from 10^5 to 10^6. The other uses pl = $1/10^6$ and pu = $1/10^7$ and is considered highly ill-conditioned and to approach the ill-conditioning typical of inverse problems such as those associated with acoustic source image deconvolution. Condition numbers in this case ranged approximately from 10^9 to 10^{10}. These primary sets of test matrices are summarized in table 2. Additional symmetric (pl = pu) and mildly asymmetric (|pl-pu| = 1) test matrices using other values of pl and pu were used to ascertain SVD method performance over a range of level of ill-conditioning.

The largest (order 1200) test matrices addressed represent, with one exception, the largest that all tested SVD computation methods could address given the memory size available on the DEC Alpha host test machine (array1.larc.nasa.gov). The exception is the computational method applying eigendecomposition to the symmetric matrix S = H(A), for which an order 1000 A (and order 2000 S) is the largest addressable integer multiple of 100. Note that the total memory requirement for all tested SVD computation methods includes, in addition to table 1 identified work space, a common requirement of order*(2*order+1), where order equals 2*n rather than n when S = H(A) is addressed.

Table 2. Characteristics of two primary test matrix sets addressed

Ill-Conditioning	Symmetry	Order Range	pl	pu	Comparison with deconvolution
Mild	Mildly asymmetric	100-1200 in 100s	10^{-2}	10^{-3}	Much less ill-conditioned
High	Mildly asymmetric	100-1200 in 100s	10^{-6}	10^{-7}	Similar level of ill-conditioning

Test Results

Test results for all selected SVD computation methods when applied to problems addressing the identified mildly and highly ill-conditioned sets of test matrices are presented in this section following discussions of the performance metrics used, compilation optimization used, and a preliminary comparison of methods performance against a range of fixed order alternative symmetric and non-symmetric test problems of varied condition number. Presented results will include, in addition to performance metric comparisons of SVD methods applied to matrices of each test set, the impact of ill-conditioning level on selected SVD methods. The impact, when addressing the eigendecomposition problem formulation required by the two LPdsyevr based methods, of alternative constructions of the symmetric matrix, S, from a test matrix, A, will be separately addressed. As in the previous statement, all references to SVD computation methods in this section are made by the alternative (to routine) names presented under the "Routine" column in table 1. Reference to LPdsyevr without specifying the form of S addressed indicates LPdsyevr applied to $S = A^T*A$. Figure caption references to "easy" or "hard" A (or S) indicate references to tests from the test sets addressing, respectively, mildly or highly ill-conditioned A (or S created from such an A).

Performance Metrics

The two primary test metrics are execution (elapsed) time in seconds for the SVD computation and the residual of the computed SVD. The residual metric is defined as:

$$\Sigma_{i,j} |A(i,j) - U*W*V^T(i,j)|$$

where (i,j) represents the individual elements of original test matrix, A, and the reconstructed A computed as $U*W*V^T$. Ratios of each these primary metrics for pairs of SVD computation methods serve as additional metrics. For the eigendecomposition based SVD computation methods, the total SVD computation time is also broken down into components allowing a comparison, in percents of total time, of performing the SVD of symmetric matrix, S, and the associated pre- and post-processing costs (e.g. computations of $S = A^T*A$ and $U_i = A*V_i/w_i$, i =1,n). Further references to simply "time" will indicate total SVD computation time. Secondary test metrics include the maximum and minimum SV magnitudes and condition number based on their ratio.

A preliminary validation of proper SVD computation by each method compared all components of U, V, and W for an order four problem. Methods varied somewhat in the order and sign of singular vectors (columns of U and V), but all provided a correct SVD.

Selection of Compilation Optimization

Preliminary tests of the four direct SVD methods (NRsvdcmp, NAGf02wef, LPdgesvd, and LPdgesdd) against the order 700 matrix of the mildly ill-conditioned test set were performed with various Fortran compilation optimization options to select the option for all further testing. Figures 1 and 2 respectively compare the time and residual metrics of these four methods for this test matrix over the range of compilation optimization options. Based on the reduced time metric for the "O5" optimization option for the methods LPdgesvd and LPdgesvd, and the negligible differences in the residual metric over all methods and options, the O5 option was chosen. The

slightly reduced residual values for NRsvdcmp and LPdgesdd when using optimization option "mlibfast" did not sway this decision since this option is designed to favor speed over accuracy.

Condition Number Tests

Additional preliminary tests with most SVD computation methods against a range of both symmetric and mildly non-symmetric order 400 test matrices of varied condition number were performed to ascertain the impact of test matrix character on methods performance. The selection of matrix character for both the mildly and highly ill-conditioned sets of test matrices was based on the results of these preliminary tests. The only SVD computation method not addressed by these tests is LPdsyevr applied to S=H(A); LPdsyevr applied to $S=A^T*A$ is addressed.

Table 3 summarizes the definition and character of the addressed matrices. For these order 400 test matrices, condition number generally increases as 1/(pl*pu). The level of ill-conditioning is assigned accordingly. Odd levels represent mildly non-symmetric matrices while even levels represent symmetric matrices. The condition numbers presented are those computed from the ratios of maximum SV over minimum SV (of A) returned by LPdgesvd, the SVD method judged to best approximate the minimum SV. All applied SVD methods returned the same maximum SV for each of the set of increasingly ill-conditioned order 400 test matrices. Figure 3 compares, over this same set, the magnitude of the minimum SV computed by LPdgesvd with those computed by the other tested SVD methods. For matrices up to level 7, all methods but LPdsyevr (applied to $S=A^T*A$) yield similar condition numbers when determined by this computation. SVD method LPdgesdd computes minimum SV nearly equal to those of LPdgesvd through level 8, while NAGf02wef follows suit through level 7. NRsvdcmp computes slightly larger values through level 7. LPdsyevr (which will now imply its application to $S=A^T*A$ unless specified otherwise) fails to provide meaningful values beyond level 3. This is attributable to the necessity to retrieve SV values for A as the square roots of the computed eigenvalues of S.

Table 3. Character of order 400 test matrices of varied condition number

Level	Symmetric	Definition pl	Definition pl	Condition Number
1	No	10^{-2}	10^{-3}	2.90E+05
2	Yes	10^{-3}	10^{-3}	1.65E+06
3	No	10^{-4}	10^{-5}	3.33E+07
4	Yes	10^{-5}	10^{-5}	7.97E+08
5	No	10^{-6}	10^{-7}	1.20E+10
6	Yes	10^{-7}	10^{-7}	5.98E+11
7	No	10^{-7}	10^{-8}	1.36E+11
8	Yes	10^{-8}	10^{-8}	6.08E+17
9	No	10^{-8}	10^{-9}	> 4.00E+19
10	Yes	10^{-9}	10^{-9}	> 4.00E+19

Figures 4 and 5 respectively compare the time and residual of these SVD methods over the same set of increasingly ill-conditioned order 400 test matrices. Figure 4 demonstrates that LPdgesdd time is best (lowest) and very consistent over the entire range of condition level, while LPdsyevr time is less consistent and ranges from 5 to 45 percent slower and averaging about 30 percent slower. Times for the remaining three methods up through condition level 8 are consistently at least twice as slow as those for LPdgesdd, with NAGf02wef about 10 percent faster than LPdgesvd and NRsvdcmp. For the two most ill-conditioned matrices (levels 9 and 10), these three methods demonstrate a dramatic reduction in time, especially NRsvdcmp.

Figure 5 indicates that residual values generally increase by one order of magnitude over the range of condition level. LPdsyevr and NRsvdcmp generally display the lowest residuals, while the fastest method, LPdgesdd generally displays the highest. Note the erratic fluctuation of residual for NAGf02wef over the range of condition level. Note also the dramatic reduction in NRsvdcmp time for levels 9 and 10 is attended at level 10 with a dramatic rise in residual. This suggests that NRsvdcmp may be starting to experience precision problems at these condition levels.

The condition levels chosen for the mildly and highly ill-conditioned sets of test matrices to be addressed during tests over matrix order are 2 and 5 respectively. Given the above discussion, an argument for a higher-level ill-conditioned set, say level 8 or 9, can be made. Indeed, this appears to be a desirable third test set. The more modest level 5, however targets the level where LPdsyevr may begin to have problems, based on it minimum SV performance.

Mildly Ill-conditioned (Easy) Problems

All six of the SVD computation methods identified in table 1 were applied to problems addressing test matrices from the mildly ill-conditioned or "easy" test set. All but LPdsyevr (S=H(A)) were applied to all test set matrices, from order 100 through 1200 in increments of 100. LPdsyevr (S=H(A)) could not be applied to orders 1100 and 1200 due to insufficient host machine memory. The present discussion and figures 6-11 compare the results of the other five methods on these problems. Results comparison with LPdsyevr (S=H(A) will be addressed later.

Figure 6 presents a linear-linear plot of execution time over matrix order by SVD method. LPdgesdd exhibits the best time performance, while LPdsyevr ($S=A^T*A$) is a close second. The remaining three appear to exhibit a higher order time cost growth with respect to matrix order. NRsvdcmp exhibits the highest cost for the higher order problems. Figure 7 presents the same data on a log-log plot. A comparison of the line slopes of LPdgesdd and LPdsyevr ($S=A^T*A$) with those of the other three methods indicates that LPdgesdd time cost is of the same order (cubic with respect to matrix order) as these three but has a smaller cost constant. A similar comparison with the LPdsyevr ($S=A^T*A$) line slope, however, indicates a higher cost constant combined with a lower cost order that is less pronounced as matrix order increases.

Figures 8 and 9 present similar plots for the SVD residual. Methods LPdgesvd, LPdgesdd, and LPdsyevr ($S=A^T*A$) exhibit very similar residual behavior, while methods NRsvdcmp and NAGf02wef are, respectively, slightly better and worse. Figure 9 indicates that, for all five methods, residual error grows roughly as the square of matrix order, or linearly with the number of matrix elements (as one might expect).

Figure 10 compares time cost ratios of LPdgesdd, the best performing, with each of the other four methods. While LPdsyevr ($S=A^T*A$) and NAGf02wef are the worst performers for matrices of order 100, they become the best at order 200. In fact, in this range their behaviors are similar. As matrix order increases, however, the time cost performance of NAGf02wef abruptly changes to that of LPdgesvd and NRsvdcmp. These three methods perform similarly until matrix order 1200 when the cost for NRsvdcmp sharply increases relative to the other two. The dominance of LPdsyevr ($S=A^T*A$) over these three begins at matrix order 300, and increases dramatically as matrix order increases. Compared with LPdgesdd, LPdsyevr ($S=A^T*A$) performance gradually increases with matrix order, surpassing the 80 percent mark with respect to LPdgesdd for matrix order greater than 1000. In general, for matrices of order greater than 400, LPdgesdd is more than

twice as fast as all the other methods except LPdsyevr ($S=A^T*A$), and 20-25 percent faster than that alternative.

For the easy (mildly ill-conditioned) test matrices, all SVD methods returned equal maximum and minimum SV over all orders of matrix A. The log-log presentation of the condition number (|max SV|/|min SV|) in Figure 11 indicates its constant power growth with respect to test matrix order, with that power equal to 1.0. The companion line plot shows that the condition numbers for the addressed test matrices are well approximated by the ratio of matrix order over the smaller of the two power parameters, pl and pu, used to define the off diagonal elements of A. This is a consequence of the observation that the maximum SV is well approximated by matrix order, while the minimum SV is approximated by the smaller of pl and pu.

Highly Ill-conditioned (Hard) Problems

All six of the SVD computation methods identified in table 1 were applied to problems addressing test matrices from the highly ill-conditioned or "hard" test set. All but LPdsyevr (S=H(A)) were applied to all test set matrices, from order 100 through 1200 in increments of 100. LPdsyevr (S=H(A)) could not be applied to orders 1100 and 1200 due to insufficient host machine memory. The present discussion and figures 12-17 compare the results of the other five methods on these problems. Results comparison with LPdsyevr (S=H(A)) will be addressed later. The presentation of data in figures 12-17 is the same as that provided in figures 6-11.

Figures 12 and 13 indicate the relative cost performance, over order of the hard test matrices, of the same five SVD methods addressed in figures 6 and 7 is very similar to that observed for the set of easy test matrices. A similar comparison of the figures 8 and 9 with figures 14 and 15, however, indicates some differences in both relative and absolute residual behavior for these five methods over matrix order. The main relative difference is the superior performance of LPdsyevr ($S=A^T*A$) for matrix order up through 700, when its performance becomes similar to all other methods except NRsvdcmp, which is slightly better. The main absolute difference is a generally more rapid (than square of matrix order) growth of residual error with increased matrix order, approaching the cube of matrix order. The one exception is LPdsyevr ($S=A^T*A$), which appears to maintain a matrix order squared residual growth up through matrix order 800.

Figure 16 indicates that time cost ratios of LPdgesdd with each of the other four methods when addressing hard test matrices are fairly similar to those presented in figure 10 for the easy test matrices. One main difference is a better (80 to 90 percent) LPdsyevr ($S=A^T*A$) efficiency, relative to LPdgesdd, for matrix order of 200 or larger. Another is the reduction (from 90 to 80 percent) in that relative efficiency up through matrix order of 500, at which point relative efficiency increases nearly linearly with still larger matrix order. For the remaining three methods, efficiency relative to LPdgesdd remains a rather constant 40 to 50 percent with respect to matrix orders 500 and greater.

Unlike the results obtained when addressing easy test matrices, not all methods returned equal minimum SV over all orders of matrix A. The SVD method based on LPdsyevr ($S=A^T*A$), which must compute the minimum SV for A as the square root of the much smaller, and in this case zero, minimum eigenvalue for S, returns a zero minimum SV for A, for all orders of A. NRsvdcmp consistently returns one order of magnitude larger minimum SV for A than the other three methods (NAGf02wef, LPdgesvd and LPdgesdd) which return the same values for all matrix orders. The log-log presentation, in Figure 17, of the condition number (|max SV|/|min SV|) dependence on matrix order for both NRsvdcmp and these three methods show a much

noisier linear (in reality constant power) growth. The suspected better estimate (by virtue of agreement) of these three methods, compared with that from NRsvdcmp, indicates a larger (approximately 2.0) constant power dependence of condition number on test matrix order, greater than the power 1.0 predicted by the ratio of matrix order over the smaller of the two power parameters, pl and pu.

Level of Ill-conditioning Impact

The impact of level of ill-conditioning on the performance of NRsvdcmp and the best two identified alternatives, LPdgesdd, and LPdsyevr ($S=A^T*A$), is presented in that order by figures 18-23. The left and right hand (even and odd numbered) figures for each method specific pair compare, respectively, the time and residual metrics for the mildly and highly (easy and hard) sets of test matrices. Figure pair 18 and 19 indicate execution time for NRsvdcmp is generally only slightly less for the hard set of test matrices, while the NRsvdcmp residuals for the hard set, relative to the easy set, increase moderately for matrices up through order 400, but transition to a consistent order of magnitude increase for matrix orders of 700 and larger.

Figure 20 indicates, for LPdgesdd, no significant difference in execution time over all matrix orders when addressing easy and hard test matrices. When a difference does occur, however, it is larger for the hard test matrices. Figure 21 indicates LPdgesdd residuals for the hard set, relative to the easy set, increase, with matrix order, toward a consistent order-of-magnitude increase over all but the smallest tested order of 100. Figure 22 indicates, for LPdsyevr ($S=A^T*A$), a consistently moderate decrease in execution time, for hard test matrices relative to easy, over all matrix orders greater than 500. This decrease is greater than that experienced with NRsvdcmp when making the same comparison. Figure 23 indicates LPdsyevr ($S=A^T*A$) residuals for the hard set remain consistent with those for the easy set for test matrices of up through order 700. Residual values for the hard set transition to one order of magnitude larger than those for the easy set over matrix orders 800 and 900, and consistently maintain that difference for matrix order larger than 900.

Time Cost Breakdown for LPdsyevr ($S=A^T*A$)

Computation of the SVD of a generally non-symmetric matrix, A, by the eigendecomposition of a related symmetric matrix, S, as is required when exploiting the highly efficient RRR algorithm by Dhillon[11] is currently implemented in LAPACK 3.0 only in support of LPdsyevr. The choice of the introduced alternative constructions of S effect both the order of S relative to A and the computational overhead involved, the latter primarily consisting of the construction of S and the extraction of at least a set of singular vectors of A from the eigenvectors of S. The cost of extracting the SV of A from the eigenvalues of S is relatively negligible.

For method LPdsyevr ($S=A^T*A$), the order of S remains equal to the order of A, but the associated overhead costs of computing $S=A^T*A$ and extracting the left singular vectors, U_i, of A by the computations $U_i= A*V_i/w_i$ are both significant. Figure 24 compares as percents of total LPdsyevr ($S=A^T*A$) cost, the component costs of; 1) $S=A^T*A$, 2) the eigendecomposition (EVD) of S, and 3) $U_i= A*V_i/w_i$, i=1,n (U = A*V*invW). Both overhead costs remain very similar in percent of total cost over all orders of A, growing from 10 percent for order 100 to near an asymptotic percent of about 30 for order 700 and larger. In contrast, the percent cost of the eigendecomposition of S (by the very efficient RRR algorithm) drops sharply from a maximum of 80 for matrix order 100 to near an asymptotic percent of 40 for order 700 and larger. This

indicates that overhead costs account for 60 percent of total SVD cost by the LPdsyevr ($S=A^T*A$) method for large order A. Figure 25 simply shows the same component cost information in a cumulative form.

Figure 26 compares the time cost, over easy matrices of varied order, of just the EVD component of computing SVD by LPdsyevr ($S=A^T*A$) with the cost of computing SVD by LPdgesdd. The significantly lower cost of computing the EVD (of S) demonstrates the higher efficiency of the RRR algorithm. Figure 27 shows that the order of EVD cost growth with increase in matrix order is fractionally less than the corresponding order of SVD cost over matrix order realized with SVD by LPdgesdd, particularly for matrix order less than 500 where EVD cost order approximates the theoretical best of 2.0 (i.e. $O(n^2)$ cost). Given this excellent performance of EVD by the RRR algorithm, the proposed plan to implement a direct SVD method based on RRR in a future release of LAPACK, one expected to be less burdened with overhead cost, promises to make available an SVD method with the potential for better total SVD computation performance than the current best method, LPdgesdd.

Alternative LPdsyevr (S=H(A)) Performance

The performance, relative to LPdsyevr ($S=A^T*A$), of the LPdsyevr (S=H(A)) alternative to SVD computation by way of an EVD of S is addressed in figures 28 – 35. A preliminary examination of time cost breakdown for LPdsyevr (S=H(A)) demonstrated that the overhead costs of constructing S=H(A) and extracting SVD components of A (U, V, and W, where $A=U*W*V^T$) from the EVD components of S are negligible. Presuming that the EVD time costs by the RRR algorithm for alternate S constructions of equal order are comparable, LPdsyevr (S=H(A)) cost for equal order S should be approximately 60 percent less than LPdsyevr ($S=A^T*A$) cost. The problem is that, for a given A order, S order for LPdsyevr (S=H(A)) is double that for LPdsyevr ($S=A^T*A$). Figures 28 and 29 respectively compare, for equal order test matrices, A, the time cost of these eigendecomposition formulations of SVD computation for the easy and hard test sets. They illustrate that LPdsyevr (S=H(A)) based SVD cost for equal order A is dramatically higher than that for LPdsyevr ($S=A^T*A$), particularly for hard test matrices; not an unexpected result given the doubled order of S, compared with A, for the LPdsyevr (S=H(A)) SVD method.

Figures 30 and 31 respectively compare, for equal order constructed matrices, S, the time cost of these eigendecomposition formulations of SVD computation for the easy and hard test sets. They show significantly lower LPdsyevr (S=H(A)) time cost, for a given order S, than that for LPdsyevr ($S=A^T*A$); a result anticipated by the equal EVD time cost presumption mentioned in the previous paragraph. This difference in time cost is noticeably smaller, however, for the hard test matrix set, compared with the easy test matrix set.

Figures 32 and 33 compare, for the easy and hard test matrix sets respectively, the residuals for the LPdsyevr ($S=A^T*A$) and LPdsyevr (S=H(A)) methods for SVD computation for the same A matrix over matrix A order. The residuals for LPdsyevr (S=H(A)) are, for both easy and hard test matrix sets, consistently up to an order of magnitude larger for A of order 800 or less. Figure 33 also indicates that LPdsyevr (S=H(A)) method residuals can be several orders of magnitude larger when addressing hard test matrices (reference the data point for order 700 A).

Finally, figures 34 and 35 compare, for time and residual metrics respectively, the performance degradation of LPdsyevr (S=H(A)) when addressing more ill-conditioned test matrices, A. Figure 34 shows this cost performance degradation to increase significantly as the order of A increases. The residual metric plots in figure 35 show that LPdsyevr (S=H(A)) based SVD accuracy

degrades only slightly with more ill-conditioned test matrices, A, and appears to be only slightly more pronounced as the order of A increases. The spike in figure 35 for the hard matrix test set at the order 700 data point again suggests that LPdsyevr (S=H(A)) based SVD has the potential to be quite limited in accuracy when the addressed A is highly ill-conditioned.

All these observations indicate that LPdsyevr ($S=A^T*A$) is by far the preferred alternative for an eigendecomposition formulated SVD method.

Conclusions

Based on the above described test results, the following conclusions are drawn:

- NRsvdcmp is comparable with other readily available direct complete SVD codes, such as NAGf02wef and LPdgesvd, employing some version of the classic algorithm (QR based reduction to SVD from bidiagonal form in phase 2), but may experience precision problems for highly ill-conditioned matrices of order greater than 1000.
- LPdgesdd, employing a divide and conquer technique in phase 2, is the best currently available direct complete SVD code for matrices of order 300 or greater, typically requiring less than half the execution time as any of the codes based on the classic direct complete SVD algorithm, and exhibiting comparable accuracy.
- The one disadvantage of LPdgesdd is its factor-of-eight increase in workspace requirement compared with the classical method based codes.
- A direct complete SVD method based on the most promising Relatively Robust Representation (RRR) algorithm for phase 2 reduction to SVD from bidiagonal form has the potential to be significantly faster than LPdgesdd, but an implementation does not currently appear to be readily available, although a future version of LAPACK is expected to provide one.
- Complete RRR based EVD is significantly faster than the current best (LPdgesdd) method of complete SVD for equal order matrices.
- Regarding SVD computation of a non-symmetric order n square matrix, A, by means of symmetric matrix, S, eigendecomposition, the order n $S = A^T*A$ construction is strongly recommended over the addressed order 2*n alternative of S = H(A).
- The currently available LPdsyevr method for $S = A^T*A$ based SVD computation by eigendecomposition is significantly faster than the classical direct complete SVD methods, but is not as fast as LPdgesdd. This is due to a 60 percent overhead for S construction and SVD component extraction from EVD of S for moderate or larger order A.

Limitations

This evaluation is limited to serial methods for complete SVD. While parallel and partial SVD methods were addressed in the presented survey of SVD methods, they were not tested. All LAPACK implementations have the potential to display performance beyond that documented in this paper when linked with host hardware optimized BLAS. The characteristics of the singular value distributions for the generated test matrices have not been analyzed, may not be similar to those typical of acoustic noise deconvolution problems, and that dissimilarity may have a significant impact on the speed of the newer SVD methods LPdgesdd and LPdsyevr ($S=A^T*A$).

References

1. Humphreys, W. M.; Brooks, T. F.; Hunter, W. W.; and Meadows, K. R.: "Design and Use of Microphone Directional Arrays for Aeroacoustic Measurements", *AIAA Paper 98-0471, AIAA 36st Aerospace Sciences Meeting & Exhibit*, Reno, Nevada, January 12-15, 1998.
2. Brooks, T. F.; Humphreys, W. M.: "A Deconvolution Approach for the Mapping of Acoustic Sources (DAMAS) Determined from Phased Microphone Arrays", *AIAA Paper 2004-2954, 10th AIAA/CEAS Aeroacoustics Conference*, Manchester, UK, May 10-12, 2004.
3. Pan, Y. and Hamdi, M.: "Singular value decomposition on processor arrays with a pipelined bus system", Journal of Network and Computer Applications 19, pp. 235-248, 1996.
4. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P.: *Numerical Recipes in Fortran 77*, 2nd Edition, Vol. 1, Cambridge University Press, 1992.
5. deVilliers, G. D.; McNally, B.; and Pike, E. R.: "Positive solutions to linear inverse problems", Inverse Problems 15, pp. 615-635, 1999.
6. Hanson, P. C.: "The L-curve and its use in the numerical treatment of inverse problems", Department of Mathematical Modeling, Technical University of Denmark, DK-2800 Lyngby, Denmark.
7. Higham, N.J.: "Recent Developments in Dense Numerical Linear Algebra", Numerical Analysis Report No. 288, Manchester Center for Computational Mathematics, April 1996 (revised August 1996).
8. Anderson, E.; Bai, Z.;Bischof, C.; Blackford, S.; Demmel, J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; and Sorensen, D.: *Lapack User's Guide, 3rd Edition*, Society for Industrial and Applied Mathematics (SIAM), 1999.
9. Blackford, S.; Choi, J.; Cleary, A.; D'Azevedo, E.; Demmel, J.; Dhillon, I.; Dongarra, J.; Hammarling, S.; Henry, G.; Petitet, A.; Stanley, K.; Walker, D; and Whaley, R.: *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathematics (SIAM), 1997.
10. Bai, Z.; Demmel, J.; Dongarra, J.; Ruhe, A.; and van der Vorst, H: *Templates for the Solution of Algebraic Eigenvalue Problems, A Practical Guide*, Society for Industrial and Applied Mathematics (SIAM), 2000.
11. Dhillon, I. S: "A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem", Doctoral dissertation, University of California, Berkeley, 1997.

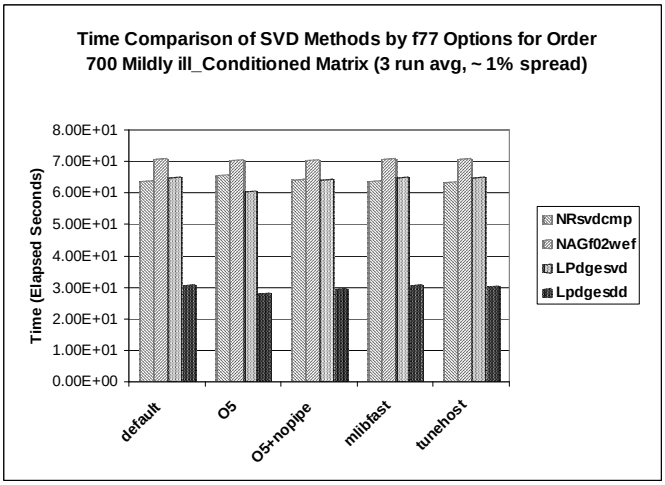


Figure 1. Time by compiler option

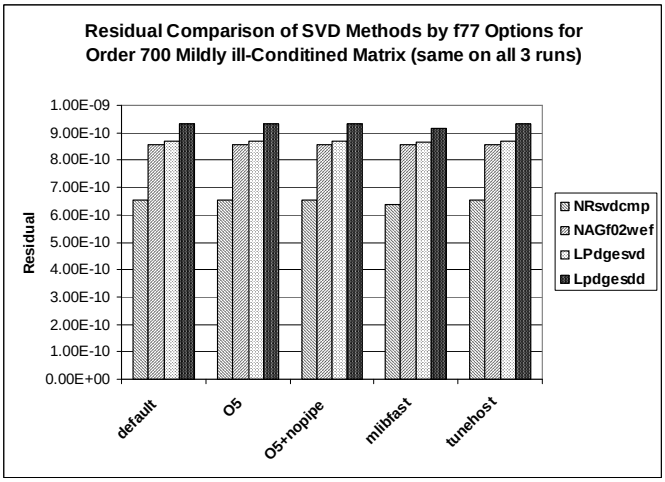


Figure 2. Residual by compiler option

.

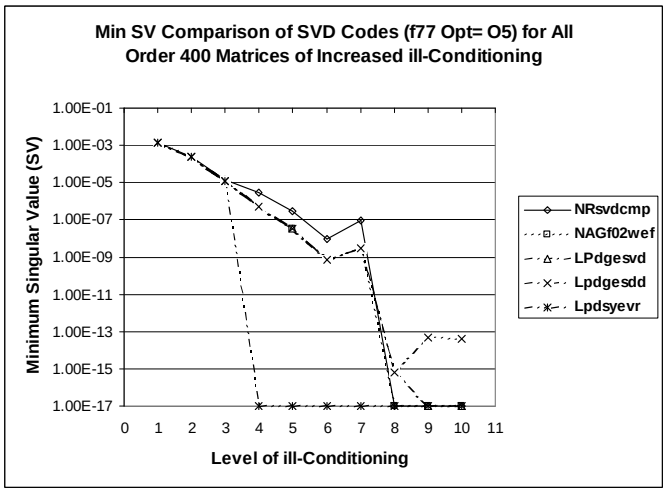


Figure 3. Minimum SV comparison (SV = 10^{-17} represents zero or any smaller value)

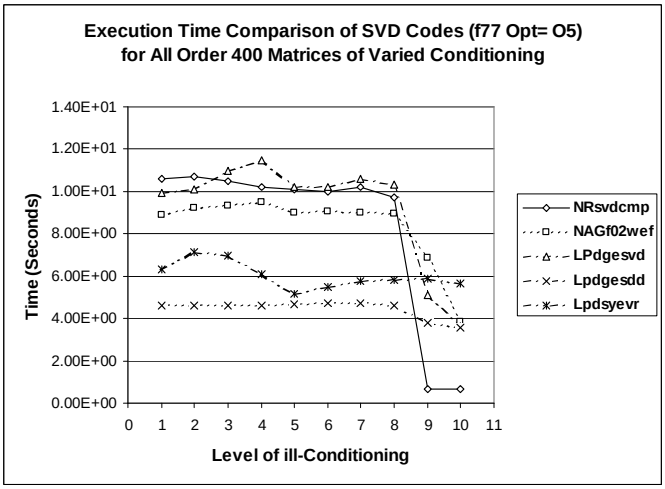


Figure 4. Time by condition

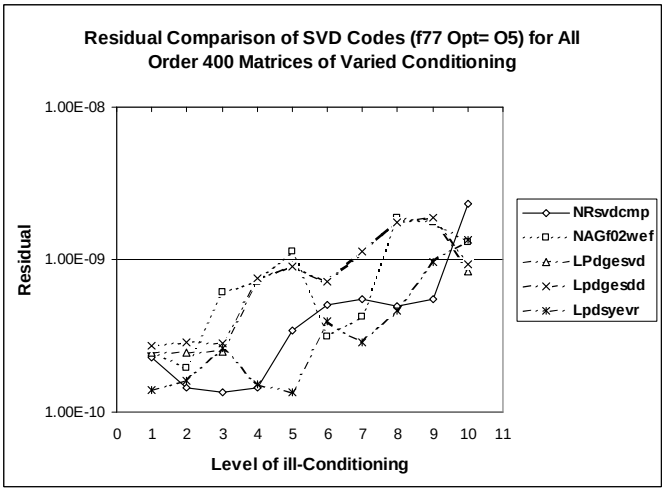


Figure 5. Residual by condition

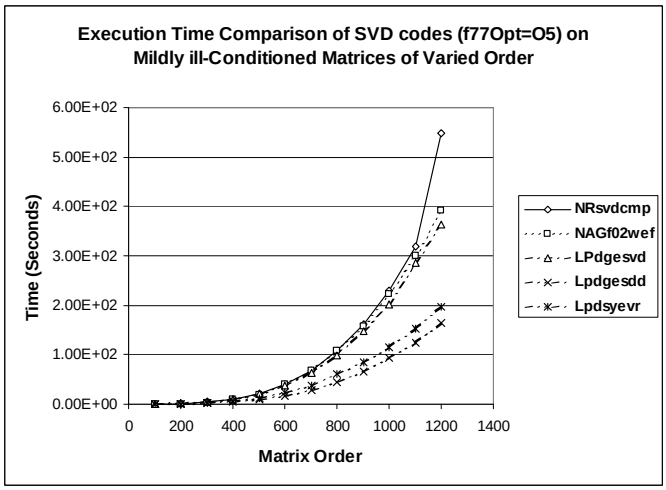


Figure 6. Time by method for easy

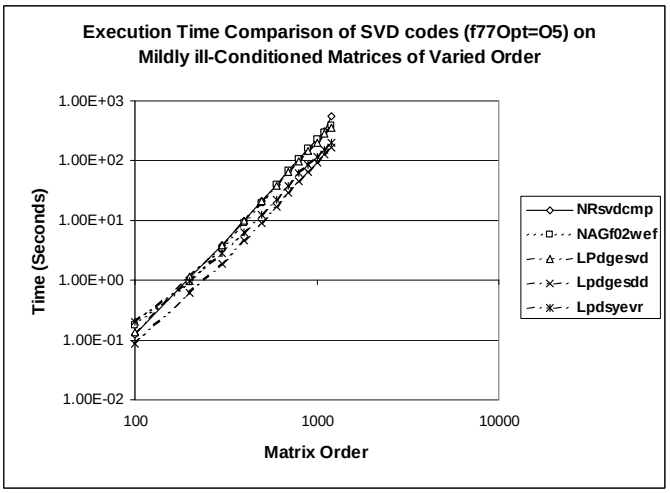


Figure 7. Time order for easy

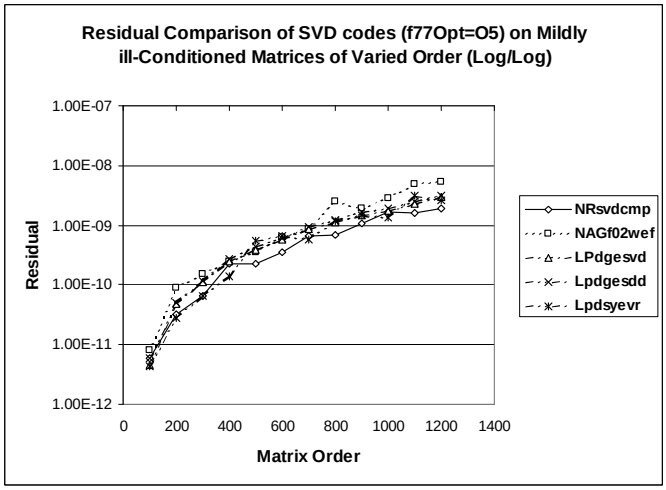


Figure 8. Residual by method for easy

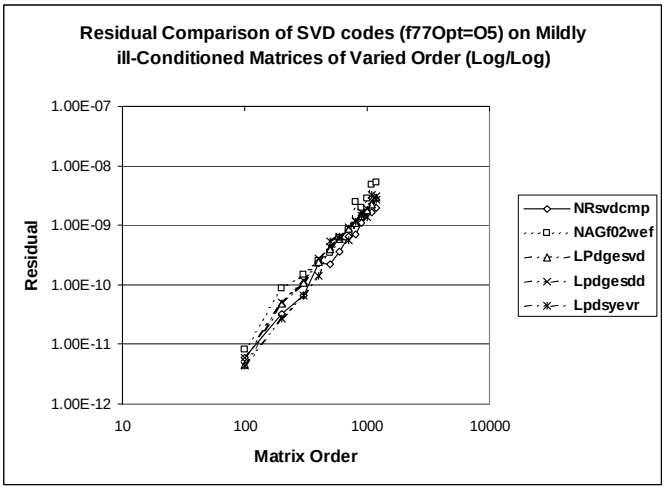


Figure 9. Residual order for easy

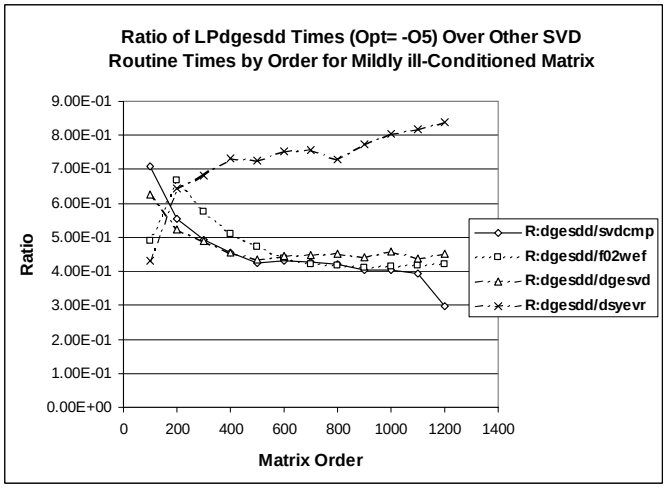


Figure 10. Method time ratios for easy

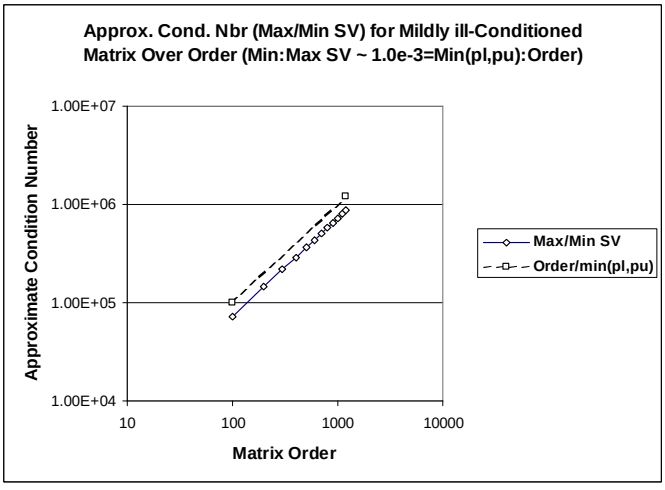


Figure 11. Condition number for easy

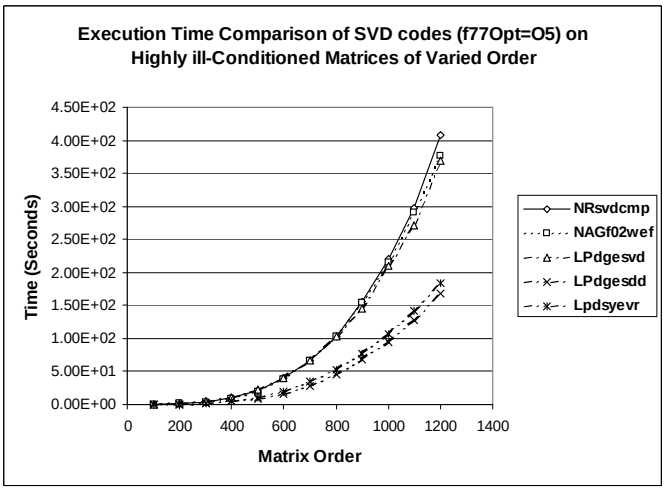


Figure 12. Time by method for hard

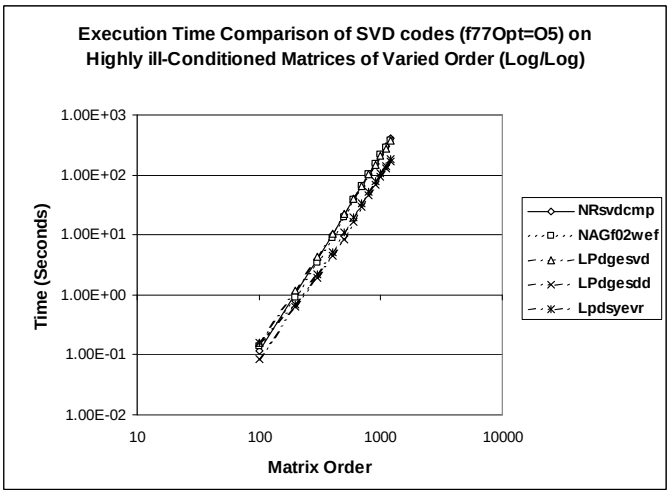


Figure 13. Time order for hard

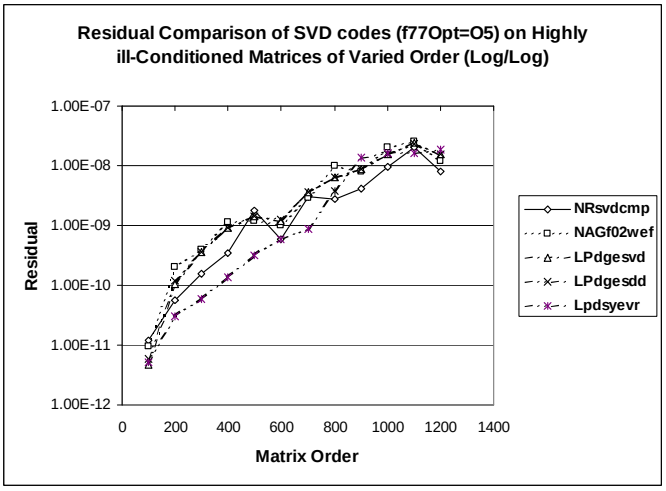


Figure 14. Residual by method for hard

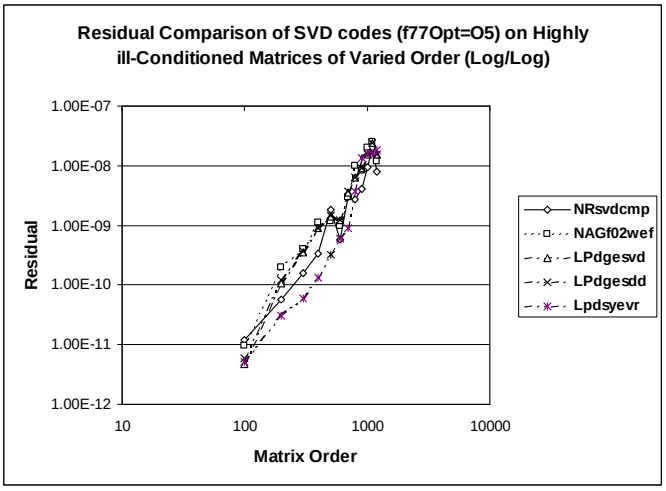


Figure 15. Residual order for hard

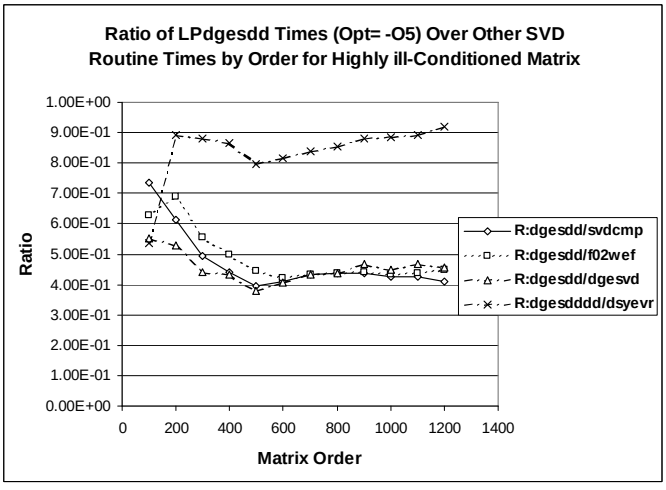


Figure 16. Method time ratios for hard

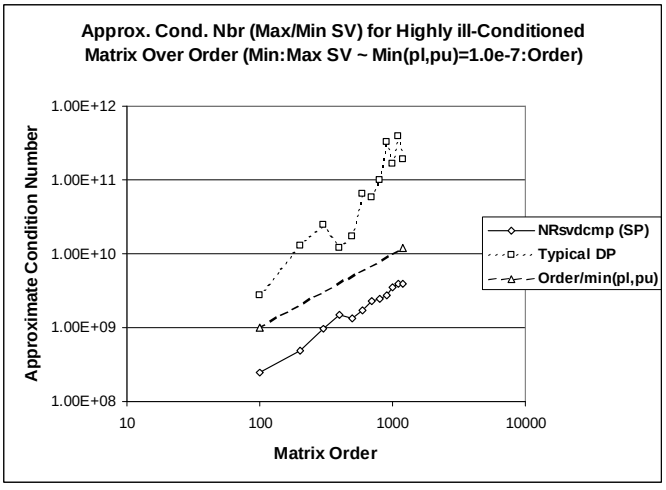


Figure 17. Condition number for hard

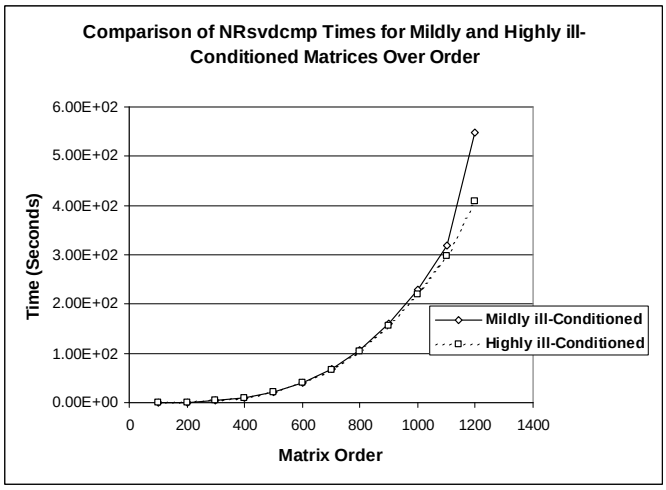


Figure 18. NRsvdcmp easy/hard times

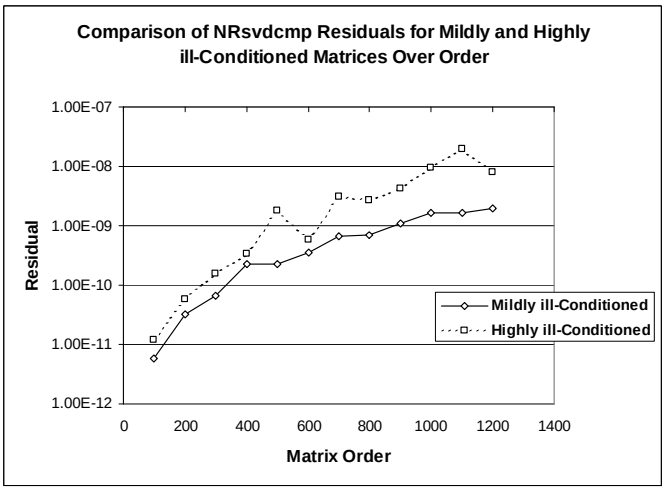


Figure 19. NRsvdcmp easy/hard residuals

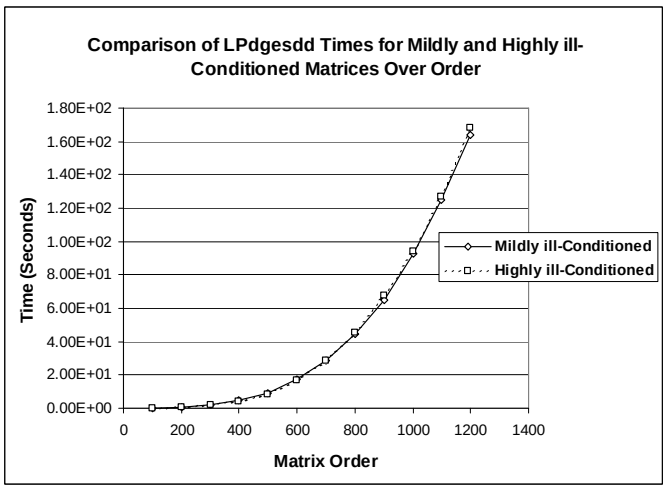


Figure 20. LPdgesdd easy/hard times

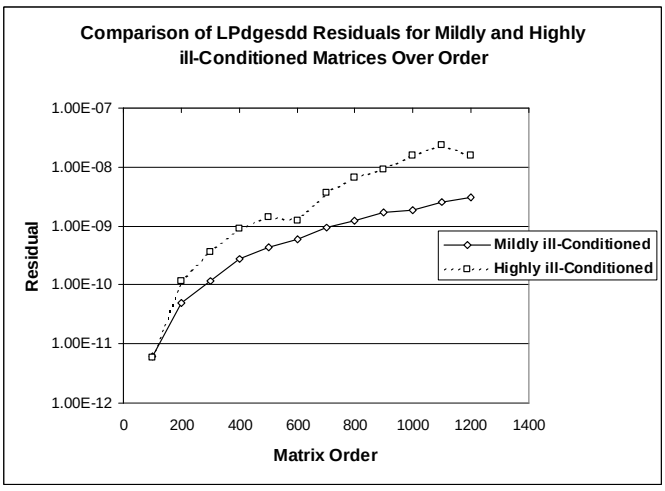


Figure 21. LPdgesdd easy/hard residuals

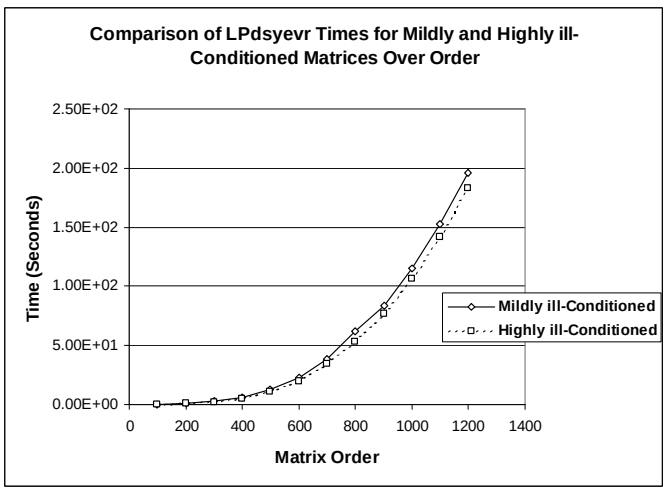


Figure 22. Lpdysevr easy/hard times

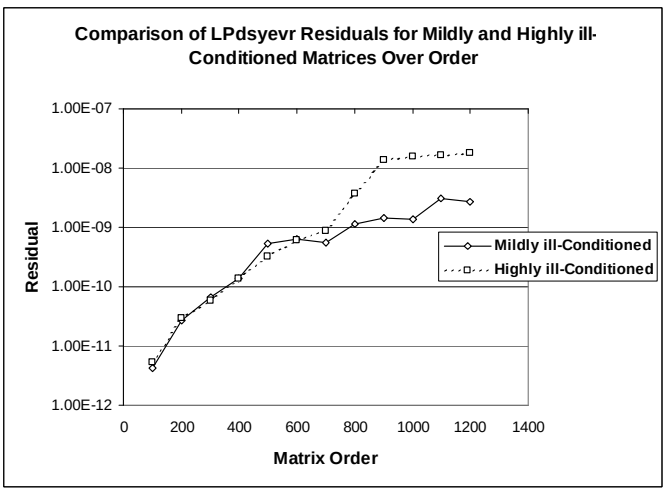


Figure 23. LPdysevr easy/hard residuals

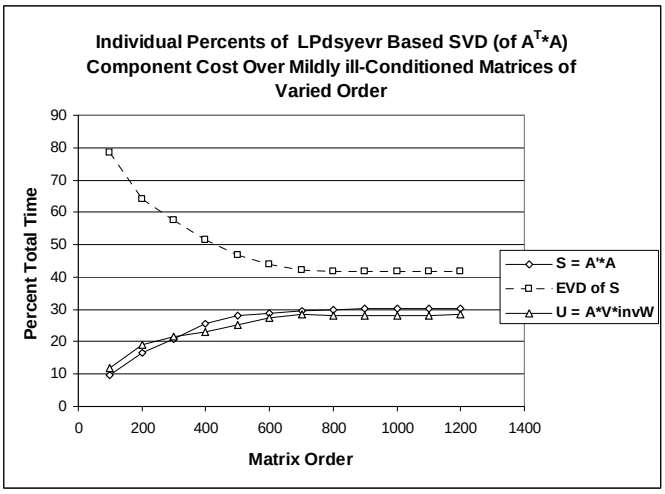


Figure 24. Lpdysevr component times

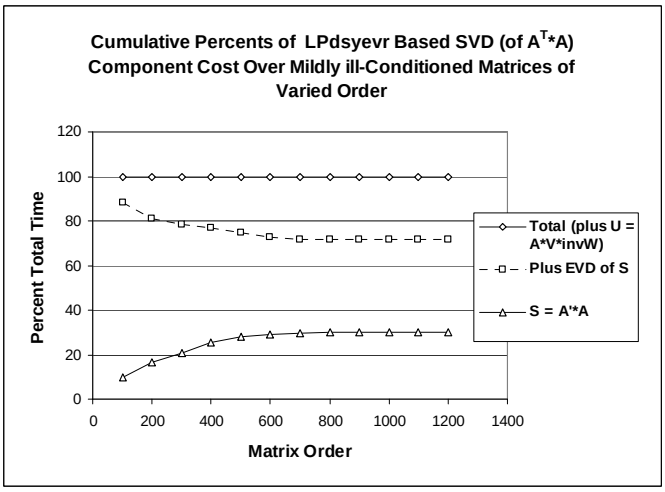


Figure 25. Lpdysevr cumulative times

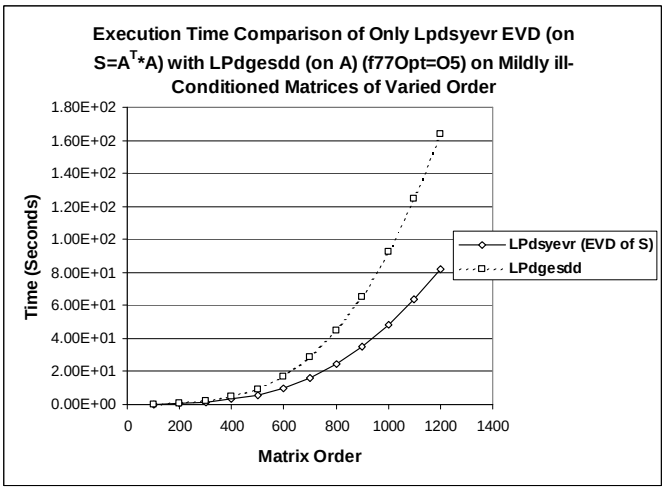


Figure 26. LPdysevr EVD vs LPdgesdd times

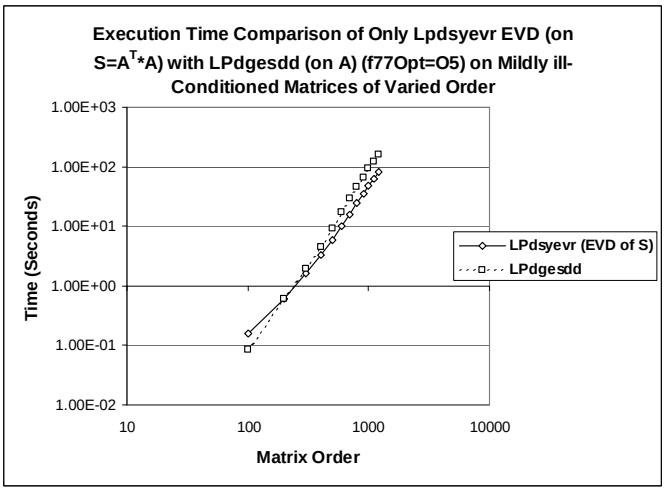


Figure 27. LPdysevr EVD vs LPdgesdd time orders

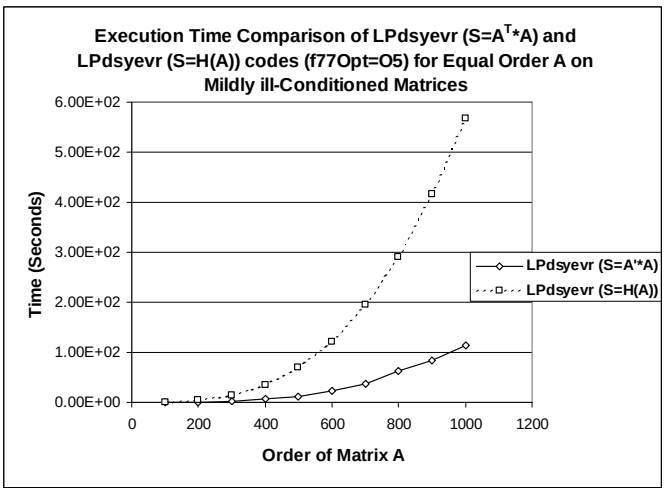


Figure 28. Equal order easy A times

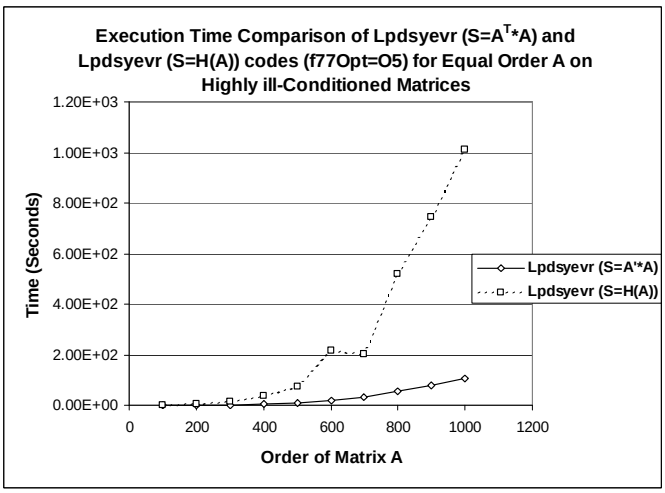


Figure 29. Equal order hard A times

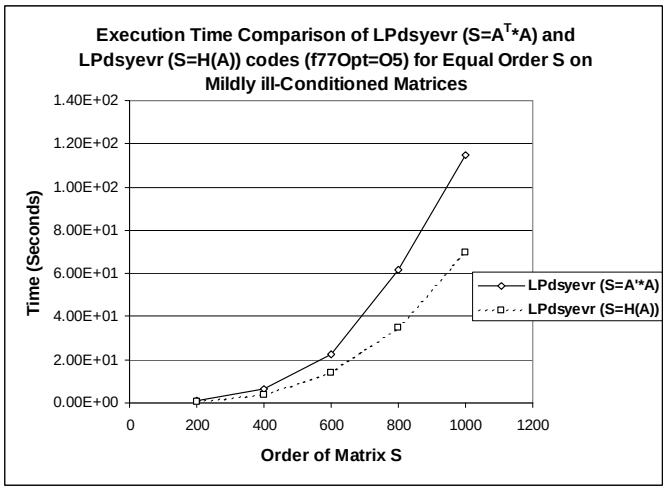


Figure 30. Equal order easy S times

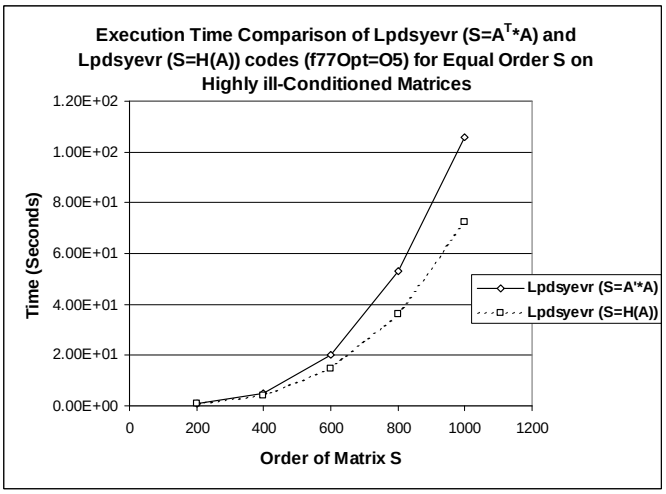


Figure 31. Equal order hard S times

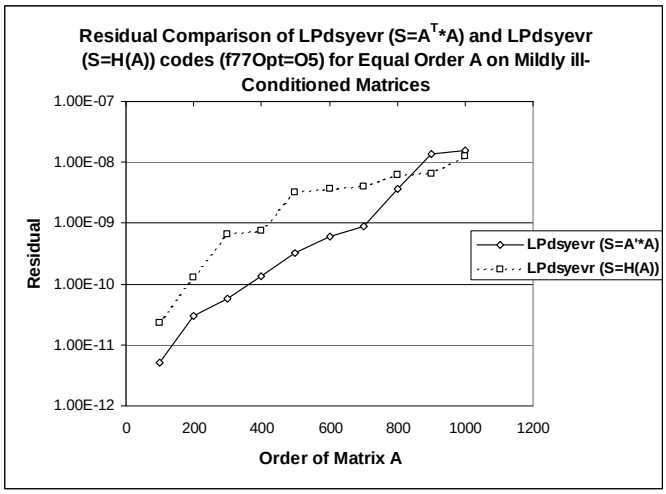


Figure 32. Equal order easy A residuals

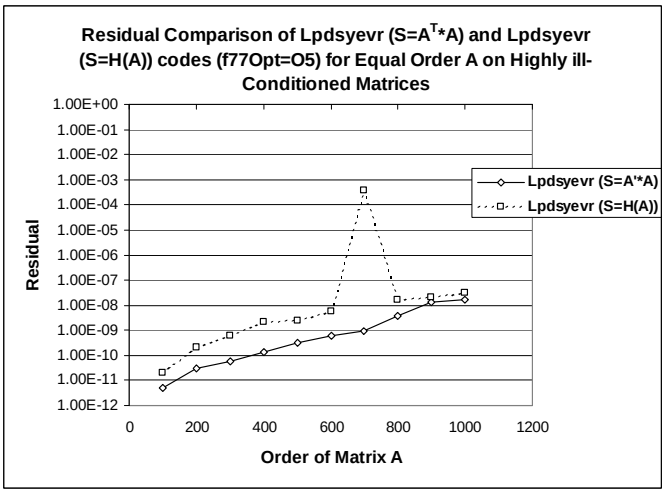


Figure33. Equal order hard A residuals

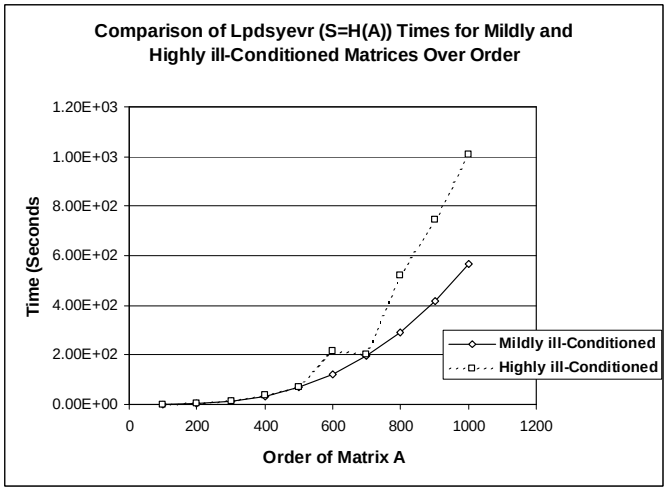


Figure 34. S=H(A) easy/hard times

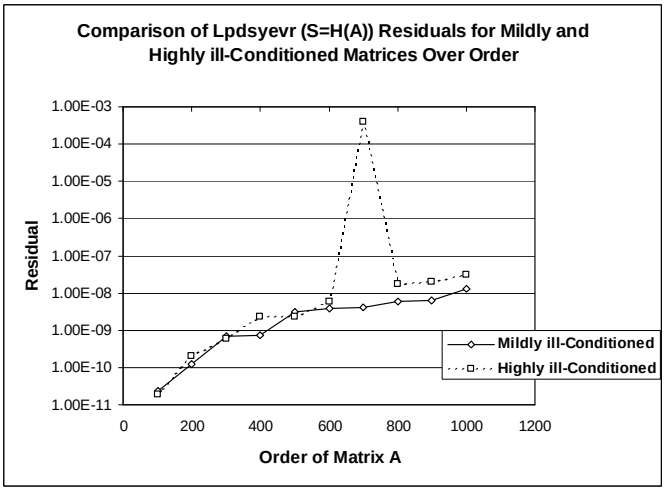


Figure 35. S=H(A) easy/hard residuals

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 07 - 2005	Contractor Report	

4. TITLE AND SUBTITLE

A Survey of Singular Value Decomposition Methods and Performance Comparison of Some Available Serial Codes

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Plassman, Gerald E.

5d. PROJECT NUMBER

L-70750D

5e. TASK NUMBER

5f. WORK UNIT NUMBER

23-781-10-11

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

Raytheon Technical Services Company
Hampton, Virginia 23666

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/CR-2005-213500

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 71
Availability: NASA CASI (301) 621-0390

13. SUPPLEMENTARY NOTES

Prepared for Langley Research Center under GSA Contract GS-00T-99-ALD-0209.
Langley Technical Monitor: William M. Humphreys.
An electronic version can be found at http://ntrs.nasa.gov

14. ABSTRACT

This contractor report describes a performance comparison of available alternative complete Singular Value Decomposition (SVD) methods and implementations which are suitable for incorporation into point spread function deconvolution algorithms. The report also presents a survey of alternative algorithms, including partial SVD's special case SVD's, and others developed for concurrent processing systems.

15. SUBJECT TERMS

Applied mathematics; Data processing; Numerical Analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	27	(301) 621-0390