Algorithm 853: an Efficient Algorithm for Solving Rank-Deficient Least Squares Problems

LESLIE FOSTER
and
RAJESH KOMMU
San Jose State University

Existing routines, such as xGELSY or xGELSD in LAPACK, for solving rank-deficient least squares problems require $O(mn^2)$ operations to solve $\min \|b - Ax\|$ where A is an m by n matrix. We present a modification of the LAPACK routine xGELSY that requires $O(mnk)$ operations where k is the effective numerical rank of the matrix A. For low rank matrices the modification is an order of magnitude faster than the LAPACK code.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classification—*Fortran 77*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms, Performance

Additional Key Words and Phrases: least squares, QR factorization, rank-deficient

1. INTRODUCTION

The solution of the least squares problems

$$\min_x \|b - Ax\| \tag{1}$$

where A is an $m \times n$ rank-deficient or nearly rank-deficient matrix and $\| \ \|$ indicates the two-norm is important in many applications. For example such ill-posed problems arise in the form of inverse problems in areas of science and engineering such as acoustics, astrometry, tomography, electromagnetic scattering, geophysics, helioseismology, image restoration, remote sensing, inverse scattering, and the study of atmospheres [Hansen 1998; Enting 2002]. LAPACK [Anderson et al. 1999] is the most widely used computational tool to solve (1) and LAPACK has two recommended routines for solving (1). The routine xGELSD is based on the singular value decomposition (SVD) and the routine xGELSY is based on a complete orthog-

This work was supported in part by the Woodward bequest to the Department of Mathematics, San Jose State University.
Authors' addresses: L. Foster, Department of Mathematics, San Jose State University, San Jose, CA 95192; R. Kommu, Department of Physics, San Jose State University, San Jose, CA, 95192 (currently at Department of Physics, University of California at Davis, Davis, CA, 95616).

onal decomposition calculated using the QR factorization with column interchanges. Both of these routines require $O(mn^2)$ floating point operations (flops). We present a modification, which we call xGELSZ, of xGELSY that requires $O(mnk)$ flops where k is the effective numerical rank. For low rank problems the new routine is an order of magnitude faster than xGELSY or xGELSD and for high rank problems the routine xGELSZ requires essentially the same time as xGELSY and less time than xGELSD. This paper presents an implementation of an algorithm similar to an algorithm discussed in [Foster 2003].

The article is structured as follows. In Section 2 we discuss some of the theory behind our algorithm. In Section 3 we describe the usage of xGELSZ and its associated routines. Section 4 presents some results comparing xGELSZ, xGELSY and xGELSD.

2. TRUNCATED QR FACTORIZATIONS

The routine xGELSY in LAPACK is based on the QR factorization with column interchanges

$$A = QRP^T = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} P^T. \tag{2}$$

In these equations Q is an m by m orthogonal matrix (we assume for the moment that A is real and that $m \geq n$), P is a n by n permutation matrix chosen by standard column interchanges [Businger and Golub 1965], R_{11} is a $k \times k$ upper triangular nonsingular matrix, R_{12} is a $k \times (n-k)$ matrix, and R_{22} is an $(m-k) \times (n-k)$ matrix which is upper triangular. In xGELSY the effective numerical rank k is chosen so that R_{11} is the largest leading submatrix whose estimated condition number is less than a user chosen tolerance. Following the computation of the decomposition (2) the LAPACK algorithm truncates the QR factorization by considering R_{22} to be negligible. The matrix R_{12} is annihilated by orthogonal transformations from the right, arriving at the complete orthogonal factorization

$$A \cong Q \begin{pmatrix} T_{11} & 0 \\ 0 & 0 \end{pmatrix} ZP^T \tag{3}$$

where Z is an orthogonal matrix and T_{11} is a $k \times k$ upper triangular nonsingular matrix. LAPACK then calculates the minimum norm solution

$$x = PZ^T \begin{pmatrix} T_{11}^{-1} Q_1^T b \\ 0 \end{pmatrix}. \tag{4}$$

where Q_1 consists of the first k columns of Q.

The algorithm in xGELSZ is based on the simple observation that the calculated solution x is unchanged if the matrix R is triangular or if R is simply block triangular where R_{11} is triangular but R_{22} is not triangular. This follows since the matrix R_{22} is not used in the calculation of x in (4) and since Q_1, T_{11} and PZ^T are unchanged by a factorization of R_{22}. In order to carry out the partial factorization of A after k columns and rows of A have been factored so that R_{11} is $k \times k$ and R_{12} is $k \times n - k$ xGELSZ does the following:

(1) A column is chosen among columns $k+1$ to n using the criteria of [Businger and Golub 1965] and pivoted to column $k+1$.

(2) A Householder transformation is constructed to zero out column $k+1$ below the diagonal and the transformations is applied to column $k+1$.
(3) The condition estimator of [Bischof 1990] is applied to the next larger, $k+1 \times k+1$, triangular matrix . If the estimated condition number is greater than a user supplied tolerance the factorization is stopped and the effective rank is set to k. Otherwise the Householder transformation is applied to form row $k+1$ of the partially factored matrix, k is increased by one and these steps are repeated.

This requires applying $k+1$ Householder transformations to A: k to form the $k \times k$ triangular matrix R_{11} and one additional transformation is needed to construct the next larger triangular matrix. We should add that this extra transformation can be discarded since it does not affect x as calculated by (4). The routine xGELSY, on the other hand, factors A completely before applying the condition estimator of [Bischof 1990] and uses n Householder transformations.

For $m \geq n$ if R is triangular, as in xGELSY, the flop count to calculate x for a single right hand side b is approximately

$$2mn^2 - 2n^3/3 + 2nk^2 - 2k^3 \tag{5}$$

whereas if R is block triangular as in xGELSZ the flop count of the algorithm is approximately

$$4mnk - 2k^2m - 2k^3/3. \tag{6}$$

The count in (6) is never larger that the count in (5) and it is much smaller for $k << n$. In addition to reducing the work in the factorization the routine xGELSZ uses only k Householder transformations in forming $Q_1^T b$ whereas xGELSY applies n Householder transformations in forming $Q_1^T b$. This leads to an additional significant reduction in work in the case that (1) must be solved for many right hand sides b. In the above discussion we have assumed, for simplicity, that $m \geq n$. If $m < n$ there are similar saving in floating point operations.

The efficiency of an algorithm is influenced by the effective use of cache memory, as well as the number of flops in the algorithm. To use cache memory efficiently it is important that an algorithm uses level 3 (matrix-matrix) basic linear algebra subroutines, BLAS, as much as possible. In both xGELSY and xGELSZ level 3 BLAS calculations are used to update, periodically, the matrix R_{22} using the block representation of a sequence of Householder transformations [Golub and VanLoan 1996, p. 213]. Although the estimation of the condition numbers of the triangular factors R_{11} is done at different locations in the two algorithms this does not affect the level 3 BLAS calculations used to update R_{22} and, prior to the termination of the factorization in xGELSZ, these level 3 BLAS calculations are identical in xGELSZ and xGELSY. Therefore the performance gains due to the use of level 3 BLAS are similar and in both algorithms approximately half of the floating point operations in the factorizations are done using level 3 BLAS routines.

There is no loss of accuracy in using xGELSZ in comparison with xGELSY. Mathematically (in exact arithmetic) they produce the same solution x. Computationally, in inexact floating point calculations, the two algorithms are different in the location of the calculations for terminating the factorization but the calculations that affect the computed matrix R_{11} and the calculated numerical rank k are

identical. If the same precision is used throughout the computations the calculated numerical ranks and the calculated matrices R_{11} will be identical. The order of the remaining calculations to determine x may be different in the two algorithms since the orderings of the columns of R_{12} may be different. Therefore the computed solutions x, although typically quite close, are usually not identical. However for the calculation of x there is no reason that the ordering of the columns in R_{12} in one algorithm is better than the ordering in the other and the accuracy of the two algorithms will be very similar.

It is also of interest to compare the accuracy of xGELSZ with xGELSD. A useful comparison is made in [Foster 2003]. Let x_{QR} be the solution calculated by the algorithm used in [Foster 2003] and x_{SVD} be the solution calculated by xGELSD. In [Foster 2003], if x_0 solves $Ax_0 = b_0$ for an ill-conditioned matrix A and if $Ax = b = b_0 + \delta b$ is solved then x_{QR} and x_{SVD} are compared by comparing $\|x_{QR} - x_0\|$ with $\|x_{SVD} - x_0\|$. One conclusion from [Foster 2003] is that if the numerical rank is chosen at a sufficiently large gap in the singular value spectrum and if the QR factorization is rank-revealing then approximately half the time the solutions x_{QR} are closer to the desired solution x_0 than are the singular value decomposition (SVD) solutions x_{SVD}. Conversely, the SVD solutions will be closer approximately half the time and in this case overall the two algorithms are very similar in accuracy. The algorithm used in [Foster 2003] is slightly different, as described below, than the algorithm used in xGELSZ. However the analysis of [Foster 2003] applies to xGELSZ and the above conclusions are true if x_{QR} represents the result calculated by xGELSZ.

Some of the ideas used in the xGELSZ algorithm have been discussed earlier in the literature. The algorithm used by LAPACK in xGELSY is discussed in [Quintana-Orti et al. 1998]. The code for xGELSZ is a modificiation of the xGELSY code. For low rank problems it is well known [Golub and VanLoan 1996, p. 250] that truncating the QR factorization reduces the flop count in the factorization to approximately $4mnk$ for small k. Stewart [Stewart 1998, p. 385] also notes that for small k the savings in stopping the reduction early are substantial. As mentioned earlier this paper presents an implementation of an algorithm similar to an algorithm in [Foster 2003]. The primary difference is that xGELSZ checks for termination after each column of A is factored and the algorithm of [Foster 2003] tests for termination after nb columns are factored where nb is the block size used by LAPACK (a typical value of nb is 32). For very low rank problems xGELSZ will be more efficient than the code discussed in [Foster 2003]. We should also note that [Bischof and Quintana-Orti 1998] and [Eisenstat and Gu 1996] discuss algorithms which construct QR factorizations and can be used to solve (1). The code provided in [Bischof and Quintana-Orti 1998] does not truncate the QR factorization early and is slower than xGELSZ for low-rank problems. The paper [Eisenstat and Gu 1996] does discuss truncating the factorization early. However as reported in [Eisenstat and Gu 1996] their strong rank-revealing QR (SRRQR) algorithm is approximately 50% slower than the LAPACK 2.0 routine xGEQPF. Our code is faster than xGEQPF.

3. USE OF XGELSZ AND ASSOCIATED ROUTINES

Each of the routines in the package comes in four precisions – single, double, complex and complex double. Following the LAPACK convention this is indicated by the first letter, S, D, C or Z, respectively, of the associated routine. We use xGELSZ to refer to any of these precisions.

The routine xGELSZ is the driver routine for computing the minimum norm solution to the rank-deficient least squares problem (1). Its use and parameters are identical to the use and parameters of xGELSY, as described in the LAPACK User's Guide [Anderson et al. 1999], except that the parameter specifying the workspace size must be slightly larger. The routine xGELSZ needs adequate workspace to simultaneously factor the matrix and to estimate the condition number of the triangular matrices R_{11} whereas xGELSY finishes the factorization prior to estimating the condition numbers and can use the same workspace for both portions of the computation. For this reason for most problems the recommended workspace for xGELSZ is $2\min(m,n)$ greater than the recommended workspace of $\min(m,n) + 2 * n + nb(n+1)$ for xGELSY. Here nb is the block size used by LAPACK. The documentation for xGELSZ contains additional details.

The routine xGELSZ uses existing LAPACK routines and three new auxiliary routines xLAQP3, xLAQPP and xLAQPQ. The routine xLAQP3 compute the partial QR factorization of the matrix A, calculating P, Q, R_{11}, and R_{12} in (2). The matrix Q is represented as the product of k Householder transformations that are stored in the matrix A.

The routine xLAQP3 uses either xLAQPP, which uses level 3 BLAS, or xLAQPQ, which uses level 2 (matrix-vector) BLAS, for the factorization of A. The choice of whether to use level 2 BLAS routines or level 3 BLAS routines and the choice of the block size depends on the matrix size and memory available, following the guidelines incorporated in the LAPACK routine ILAENV. The routines xLAQPP and xLAQPQ use a partial QR factorization with column pivoting applied to a portion of A. Additional detailed documentation of the new auxiliary routines and the routine xGELSZ is contained in the documentation that comes with the code in the xGELSZ package.

4. COMPARISON OF XGELSD, XGELSY AND XGELSZ

We carried out timing studies of xGELSD, xGELSY and xGELSZ on three different platforms: a 2.6 GHertz Intel computer running Windows XP, a 1.5 GHertz Intel computer running Redhat Linux, and a SUN Computer running SUN-OS 5.8. We generated matrices of different effective ranks using LAPACK's routine xQRT15, modified slightly to allow the effective rank to be set to any user specified value. The routine xQRT15 is used by LAPACK to generate matrices for its timing and accuracy tests. In Figure 1 we summarize runs using our variation of DQRT15 to generate 1600 by 1600 matrices of various ranks. For each run we used BLAS routines supplied by the computer vendor. We also ran the three programs for smaller 100 by 100 matrices and our results are pictured in Figure 2.

For these 100 by 100 matrices there is a jump in the time for DGELSY or DGELSZ for the computers running Windows or SUN-OS, near a numerical rank of 32. For smaller matrices such as these 100 by 100 matrices it is generally faster

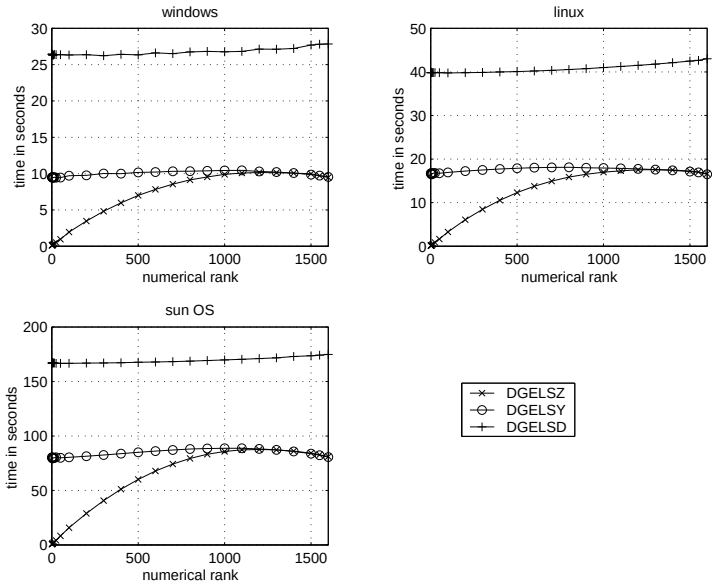

Fig. 1. Timings for DGELSZ, DGELSY and DGELSD for 1600 by 1600 matrices with a variety of numerical ranks on three computers.

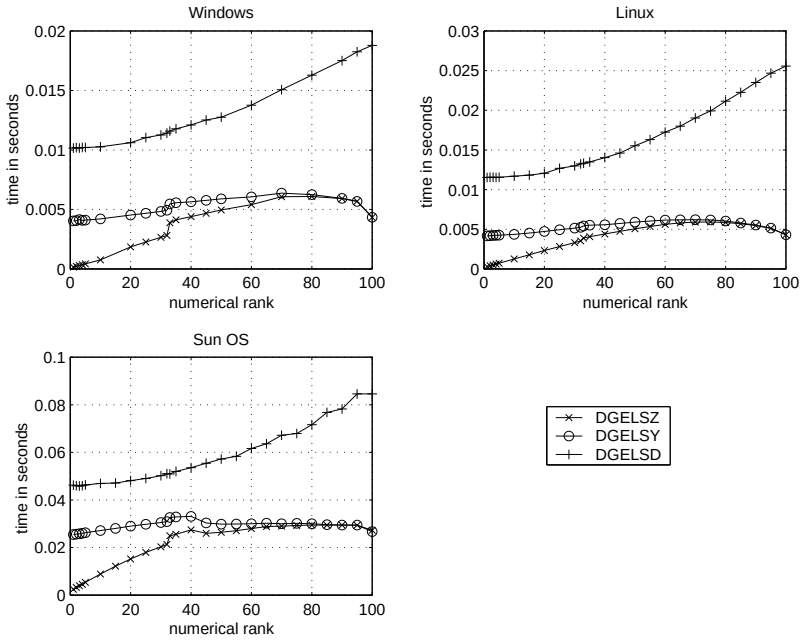

Fig. 2. Timings for DGELSZ, DGELSY and DGELSD for 100 by 100 matrices with a variety of numerical ranks on three computers.

to use an unblocked, level 2 BLAS based routines rather than blocked, level 3 BLAS based routines and most LAPACK routines used by DGELSY and DGELSZ use unblocked code for matrix sizes less than a crossover point of 128. However the LAPACK routines DORMQR and DORMRZ, which are called by DGELSZ and DGELSY, use blocked code when the numerical rank is greater than 32. For DGELSZ and DGELSY this choice is not optimal for smaller matrices on these computers. We did not modify the LAPACK routines DORMQR or DORMRZ.

The advantage of DGELSZ is similar for 100 by 100 matrices and for the larger 1600 by 1600 matrices. For high rank problems DGELSZ is approximately two to six times faster than DGELSD and DGELSZ requires essentially the same time as DGELSY. Here we use the term "essentially the same time" to indicate that in our testing for a variety of high rank matrices with dimensions between 100 by 100 and 1600 by 1600 that the average differences in times have been less than a few percent on all the platforms and precisions discussed and less than a fraction of a percent for larger matrices. For low rank problems DGELSZ is much faster than DGELSY or DGELSD. For example on the computer running Windows for the 1600 by 1600 matrices and for numerical ranks of 5, 100 and 300 DGELSZ is at least 60, 5 and 2, respectively, times faster than DGELSY and at least 125, 10 and 4 times faster than DGELSD. The results are similar for each of the three computers.

In addition to these timing tests we have done other tests. For example we have timed runs for single precision, complex and double precision complex on all three computers and we have timed runs using classes of matrices other than those generated by xQRT15. The results are consistent with the results presented above.

To check that xGELSZ correctly solves (1) we used the LAPACK test routines xCHKAA and xDDRVLS, modified to include tests of xGELSZ. The routines xCHKAA and xDDRVLS are used by LAPACK to verify that the existing least squares solvers xGELSY and xGELSD are correct. The test routines calculate a variety of ratios for a set of test matrices generated by LAPACK. The ratios should not be large, LAPACK uses a limit of 30, if the the code is correctly solving (1). In each of the four precisions and on each of the three computers discussed earlier xGELSZ passed the LAPACK test suite.

Related tests support our earlier comments that the accuracy of xGELSZ will be very similar to that of xGELSY. In Figure 3 for a sample of 5000 random 250 by 300 matrices generated by our modification of DQRT15 on the computer running Windows XP we have plotted histograms of four of the test ratios used in the LAPACK testing routines. The sample includes 200 matrices for each rank that is a multiple of 10 from 10 and 250. The labels on the plots describe each ratio where ϵ is relative machine precision, k is the effective numerical rank of A, $s_{1:k}$ indicates a vector of the first k singular values as specified in DQRT15 of A, svals(T_{11}) is a vector of the singular values of T_{11} and $r = b - Ax$. The vector b is chosen to be in the range space of A. The ratio in the lower right hand plot is defined in LAPACK's routine DQRT17 which calculates this ratio.

The horizontal axis in each plot is large enough to include the largest value over the 5000 samples of each ratio for DGELSZ. For least squares problems LAPACK accepts the accuracy of a routine if all the above ratios are less than 30 and in these calculations all the ratios for DGELSZ, as well as for DGELSY and DGELSD, were

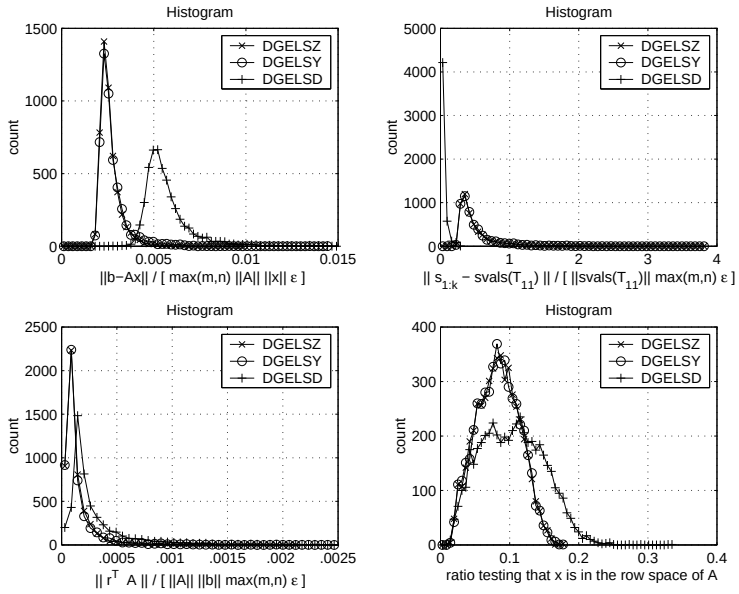

Fig. 3. Histograms for 5000 random 250 by 300 matrices of four ratios used in the LAPACK test routines.

well under this limit. Also note that the histograms for DGELSY and DGELSZ are almost identical which indicates that the accuracies of the two routines are very similar. Similar conclusions were obtained for other platforms and precisions.

REFERENCES

Anderson, E., Bai, Z., Bischof, C., , Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. 1999. *LAPACK Users' Guide, Third Edition*. SIAM, Philadelphia.

Bischof, C. H. 1990. Incremental condition estimation. *SIAM J. Matrix Annal. App. 11*, 312–322.

Bischof, C. H. and Quintana-Orti, G. 1998. Algorithm 782: Codes for rank-revealing qr factorizations of dense matrices. *ACM Transactions on Mathematical Software 24*, 254–257.

Businger, P. and Golub, G. H. 1965. Linear least squares solutions by Householder transformations. *Numerische Mathematik 7*, 269–276.

Eisenstat, S. and Gu, M. 1996. Efficient algorithms for computing a strong rank-revealing qr factorizations. *SIAM J. Sci. Computing 17*, 848–869.

Enting, I. G. 2002. *Inverse Problems in Atmospheric Constituent Transport*. Cambridge University Press, Cambridge.

Foster, L. V. 2003. Solving rank-deficient and ill-posed problems using UTV and QR factorizations. *SIAM J. Matrix Annal. App. 25*, 582–600.

Golub, G. and VanLoan, C. F. 1996. *Matrix Computations*. John Hopkins, Baltimore.

Hansen, P. C. 1998. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia.

Quintana-Orti, G., Sun, X., and Bischof, C. H. 1998. A BLAS-3 version of the QR factorization with column pivoting. *SIAM J. Sci. Comput. 19*, 1486–1494.

Stewart, G. W. 1998. *Matrix Algorithms Volume 1: Basic Decompositions*. SIAM, Philadelphia.

Received xxx; xxx; accepted xxx