Accurate SVDs of Structured Matrices

James Demmel*

October 9, 1997

Abstract

We present new $O(n^3)$ algorithms to compute very accurate SVDs of Cauchy matrices, Vandermonde matrices, and related "unit-displacement-rank" matrices. These algorithms compute all the singular values with guaranteed relative accuracy, independent of their dynamic range. In contrast, previous $O(n^3)$ algorithms can potentially lose all relative accuracy in the tiniest singular values.

LAPACK Working Note 130
University of Tennessee Computer Science Report ut-cs-97-375

1 Introduction

The *singular value decomposition (SVD)* of a real matrix G is the factorization $G = U\Sigma V^T$ where U and V are orthogonal matrices and Σ is nonnegative and diagonal. If G is m-by-n, with $m \geq n$ (otherwise transpose G), then U is m-by-n, $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_n)$ with $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$, and V is n-by-n. We call the columns u_i of $U = [u_1, ..., u_n]$ the *left singular vectors*, the columns v_i of $V = [v_1, ..., v_n]$ the *right singular vectors*, and the σ_i the *singular values*.

Our goal is to compute the SVD (i.e. the u_i, v_i and σ_i) to *high relative accuracy*, and with cost $O(n^3)$, i.e. roughly the same cost as prior, less accurate dense matrix algorithms. By high relative accuracy we mean the following:

- The error $|\sigma_i - \hat{\sigma}_i|$ in the computed singular value $\hat{\sigma}_i$ is bounded by $O(\varepsilon)\sigma_i$, where ε is machine precision, i.e. the relative error is small.
- The angle $\theta(u_i, \hat{u}_i)$ between the true left singular vector u_i and the computed vector $\hat{u}_i$ is bounded by $O(\varepsilon)/\mathrm{relgap}_i$, where $\mathrm{relgap}_i = \min_{j\neq i} |\sigma_j - \sigma_i|/\sigma_i$ is the relative gap between σ_i and the nearest other singular value. An analogous statement holds for the computed right singular vectors $\hat{v}_i$.

*Computer Science Division and Mathematics Dept., University of California, Berkeley, CA 94720 (demmel@cs.berkeley.edu). This material is based in part upon work supported by the Advanced Research Projects Agency contract No. DAAH04-95-1-0077 (via subcontract No. ORA4466.02 with the University of Tennessee), the Department of Energy grant No. DE-FG03-94ER25219, and contract No. W-31-109-Eng-38 (via subcontract Nos. 20552402 and 941322401 with Argonne National Laboratory), the National Science Foundation grant ASC-9313958, and NSF Infrastructure Grant Nos. CDA-8722788 and CDA-9401156.

In contrast, conventional numerical algorithms (QR iteration, divide and conquer, bisection and inverse iteration, traditional Jacobi, ...) only compute the SVD with *high absolute accuracy.* This means that the error in the singular value is only bounded by $O(\varepsilon)\sigma_1$ instead of $O(\varepsilon)\sigma_i$. Thus the largest singular values are computed with high relative accuracy, but the tinest singular values (those near or less than $\varepsilon\sigma_1$) may have no relative accuracy at all. Similarly, the error in the computed singular vectors can also be σ_1/σ_i times as large.

In an earlier paper [8] we showed that if we were given a *rank-revealing decomposition (RRD)* $G = XDY^T$, i.e. a decomposition where D is diagonal and X and Y are well-conditioned (but otherwise arbitrary), then we could compute the SVD of G to high relative accuracy from X, D and Y in $O(n^3)$ time. For this to work, the computed $\hat{X}$, $\hat{D}$ and $\hat{Y}$ must be sufficiently accurate:

- Each entry of $\hat{D}$ is known to high relative accuracy: $|D_{ii} - \hat{D}_{ii}| = O(\varepsilon)|D_{ii}|$.
- $\hat{X}$ and $\hat{Y}$ have small norm errors, i.e. $\|X - \hat{X}\| = O(\varepsilon)\|X\|$ and $\|Y - \hat{Y}\| = O(\varepsilon)\|Y\|$.

To summarize, our high accuracy SVD algorithm consists of two steps:

1. Compute the RRD $G = XDY^T$ sufficiently accurately.
2. Compute the SVD of XDY^T using the algorithm from [8] (see also section 2 below).

In [8] a variety of matrix classes were described that permitted the RRD $G = XDY^T$ to be computed sufficiently accurately. In particular we considered the RRD provided by Gaussian elimination with complete pivoting (GECP): X and Y are both unit triangular matrices with off-diagonals bounded by one in magnitude. However, the usual implementation of GECP will often not compute X, D and Y as accurately as required; a new way to compute the triangular factors is needed.

In particular, suppose C is a Cauchy matrix: $C_{ij} = 1/(x_i + y_j)$, where the x_i and y_j are given floating point numbers. There is a well-known formula for the determinant of C:

$$\det(C) = \frac{\prod_{1\le i<j\le n}(x_j - x_i)(y_j - y_i)}{\prod_{1\le i,j\le n}(x_i + y_j)} \quad . \tag{1}$$

Every factor $x_j - x_i$, $y_j - y_i$, or $x_i + y_j$ is computable to high relative accuracy, as well as their products and quotients, in floating point arithmetic. Thus $\det(C)$ is computable to high relative accuracy. Since every submatrix of a Cauchy matrix is Cauchy, every minor of C can be computed to high relative accuracy. The same is true of the *Cauchy-like matrix* $G = D_1CD_2$, where D_1 and D_2 are diagonal. Finally, since every entry of every Schur complement of G is the quotient of minors of G (or just a minor), all the intermediate and final results of GECP on G can be computed to high relative accuracy using equation (1). In other words, we can compute the triangular factors of G, and thus its SVD, to high relative accuracy, where G is defined by the x_i, y_j, and diagonal entries of D_1 and D_2. This does *not* mean that small relative changes in these parameters cause small relative changes in the singular values. Indeed, the 1-by-1 example with $x_1 = 1$ and $y_1 = -1 + \varepsilon$ shows that this is not true. It does mean that our algorithm is forward stable, and simply gets an accurate answer for the floating point values of the parameters stored in the machine.

Unfortunately, using equation (1) in the most straightforward way to implement GECP would cost $O(n^5)$ operations. The first main result in this paper is an algorithm that reduces the cost from $O(n^5)$ to $\frac{4}{3}n^3$. This was inspired by work in predictive pivoting for low displacement rank matrices

[4, 14, 3, 26, 17] but can also be derived straightforwardly from equation (1), which is the approach we take here. Among other examples, we compute the SVD of a 100-by-100 Hilbert matrix. The computed singular values range over 150 orders of magnitude, and agree to 15 decimal digits with the singular values computed using a conventional algorithm with 200-decimal digit arithmetic; the singular vectors agree to 14 or more digits. This result is presented in section 4.

The second result is a similar high accuracy RRD for Vandermonde matrices. It is based on the observation that if V is Vandermonde, so that $V_{ij} = x_i^{j-1}$, and $F_{ij} = \exp(2\sqrt{-1}\pi(i-1)(j-1)/n)/\sqrt{n}$ is the discrete Fourier transform matrix, then VF is (essentially) Cauchy-like, and the parameters determining the Cauchy-like structure can be computed to sufficient accuracy to use the previous algorithm. The observation that VF is Cauchy-like was also exploited in the displacement rank literature, but we again derive it from first principles. This result is presented in section 5.

We observe more generally that if G is defined by the Sylvester equation (also called *displacement equation*)

$$XG + GY = d_1 d_2^T \tag{2}$$

where

- X and Y are diagonalizable matrices with accurately known eigendecompositions, and
- d_1 and d_2 are column vectors

then G can be accurately transformed into a Cauchy-like matrix, and the above techniques applied. (If X and Y are normal matrices, this process is slightly simpler than if they are not.) This result is presented in section 6.

A matrix satisfying equation (2) is said to have *displacement rank* equal to 1, because the right hand side has rank 1. It is natural to ask about the possibility of computing accurate SVDs of matrices with higher displacement rank. For example, Toeplitz matrices have displacement rank 2. We do not currently see any way to extend our results to higher displacement rank than 1. In section 6 we arge that no high accuracy SVD for Toeplitz matrices can exist, in the sense that they exist for Cauchy and Vandermonde matrices, and describe conditions under which such algorithms can exist.

In section 7 we decribe a simple estimator for the relative error in the SVD we compute, in the case where the input data is uncertain.

Finally, section 2 also presents an improvement on the algorithm from [8] for the SVD of XDY^T. The improvement eliminates a factor from the error bound in [8] and reduces it to the minimum possible: $O(\varepsilon(\kappa(X) + \kappa(Y))$.

2 Computing an Accurate SVD of $G = XDY^T$

For completeness, we restate the algorithm from [8] (see also [10]) for computing a high accuracy SVD of the RRD $G = XDY^T$:

Algorithm 1. Compute a high accuracy SVD of $G = XDY^T$.

(1) Perform QR factorization with pivoting on XD to get $XD = QRP$, where P is a permutation. Thus $G = QRPY^T$.
(2) Multiply to get $W = RPY^T$. This must be *conventional* matrix multiplication,

e.g. Strassen's method [18] may not be used. Thus $G = QW$.

(3) Compute the SVD of $W = \bar{U}\Sigma V^T$ using one-sided Jacobi [9]. Thus $G = Q\bar{U}\Sigma V^T$.

(4) Multiply $U = Q\bar{U}$. Thus $G = U\Sigma V^T$ is the desired SVD.

The statement of Theorem 3.1 from [8] which describes the accuracy of this algorithm is as follows:

Theorem 3.1 from [8]. *Let D' be a diagonal matrix, chosen so that $R' = D'^{-1}R$ is as well conditioned as possible. We can always choose D' so that $\kappa(R')$ is bounded by $O(2^n)$, and it is usually much smaller. Then in floating point arithmetic with machine precision ε, the above algorithm computes the SVD of G with relative accuracy $\eta = O(\varepsilon\kappa(R') \cdot \max(\kappa(X), \kappa(Y)))$.*

In theory [10, 25, 9, 24, 11] the Jacobi rotations during the one-sided Jacobi in step (3) must be applied to the right side of W to guarantee the bounds. But in practice, applying them from the left is significantly faster, and has never been found to be significantly less accurate (see the comments on speed in section 4.1 below).

The factor $\kappa(R')$ in the error bound depends on how well the pivoting during the QR decomposition of XD "reveals the rank" of XD. The bound $O(2^n)$ comes from the standard column-pivoting algorithm [15] and choosing $D'_{ii} = R_{ii}$, but better alternatives are available [30, 1, 5, 6, 16, 20, 28]. For example, M. Gu has a pivoting scheme that reduces $O(2^n)$ to $O(n^{1+(1/4)\log_2 n})$, analogous to the pivot growth bound for GECP. See also [27].

We can eliminate the factor $\kappa(R')$ by using the following more expensive algorithm:

Algorithm 2. Compute a high accuracy SVD of $G = XDY^T$.

(1) Multiply $W_1 = DY^T$. Thus $G = XW_1$.

(2) Compute the SVD of $W_1 = U_1\Sigma_1 V_1^T$ using one-sided Jacobi. Thus $G = XU_1\Sigma_1 V_1^T$.

(3) Multiply $W_2 = XU_1\Sigma_1$. Thus $G = W_2 V_1^T$.

(4) Compute the SVD of $W_2 = U\Sigma V_2^T$ using one-sided Jacobi. Thus $G = U\Sigma V_2^T V_1^T$.

(5) Multiply $V = V_1 V_2$. Thus the SVD of G is $G = U\Sigma V^T$

However, Algorithm 2 is likely to be twice as expensive as Algorithm 1 because it does two Jacobi SVDs instead of one. Since R' is unlikely to be large, we cannot recommend Algorithm 2 for general use.

3 Assumptions about Floating Point Arithmetic

We use the conventional error model for floating point arithmetic:

$$\mathrm{fl}(a \odot b) = (a \odot b)(1 + \delta) \tag{3}$$

where $\odot \in \{+, -, \times, /\}$, and $|\delta| \le \varepsilon$, where ε is machine precision. In particular, we assume that neither overflow nor underflow occur, since both can destroy relative accuracy in the result.

This assumption implies that expressions like the one in equation (1) can be evaluated to high relative accuracy. Products and quotients of many floating point numbers such as in equation (1) can in principle be quite susceptible to over/underflow; this was not a problem in any test that we ran (once we replaced expressions like $(\prod_i a_i)/(\prod_i b_i)$ with $\prod_i (a_i/b_i)$) but could be guarded against by testing and scaling, or use of IEEE over/underflow flags.

Furthermore, for our algorithm for Vandermonde matrices we further assume that floating point arithmetic is implemented with a guard digit, so that $\mathrm{fl}(a \pm b)$ is exactly $a \pm b$ when cancellation occurs, i.e. when $.5 < a/b < 2$ in binary arithmetic. Almost all existing machines (except the Cray T90 and its predecessors and emulators) satisfy these conditions.

When a and b are complex numbers, we assume that equation (3) continues to hold, with δ a tiny complex number bounded by $|\delta| = O(\varepsilon)$ [18]. This is true for any reasonable implementation of complex arithmetic. Note that this is weaker than requiring that both real and imaginary parts be computed to high relative accuracy.

4 Cauchy-like Matrices

We begin with the simple observation that GECP on G can be implemented as follows. We write this decomposition as $G = XDY^T$, where X and Y are (possibly row permuted) unit lower triangular matrices. Suppose without loss of generality that the first k pivot rows and columns have been chosen as the leading k rows and columns. Then the k-th Schur complement $S^{(k)} = G_{22} - G_{21}G_{11}^{-1}G_{12}$ satisfies

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ G_{21}G_{11}^{-1} & I \end{bmatrix} \cdot \begin{bmatrix} G_{11} & G_{12} \\ 0 & S^{(k)} \end{bmatrix}$$

The next pivot row and column is chosen by finding the largest entry in $S^{(k)}$; assume without loss of generality that it is in the (1,1) position of $S^{(k)}$. Finally, $D_{k+1,k+1} = S_{11}^{(k)}$, the $k+1$-st column of X below the diagonal is first column of $S^{(k)}$ divided by $S_{11}^{(k)}$, and the $k+1$-st row of Y^T to the right of the diagonal is the first row of $S^{(k)}$ divided by $S_{11}^{(k)}$. Therefore, if we can derive a formula for computing all the entries of $S^{(k)}$ in c operations per entry, we will have an algorithm requiring $\sum_{i=1}^{n} ci^2 = \frac{c}{3}n^3 + O(n^2)$ operations to compute the same triangular factorization as GECP.

We use the fact that

$$S_{rs}^{(k)} = \det(G([1:k,r],[1:k,s]))/\det(G(1:k,1:k))$$

where we use Matlab-notation for submatrices of $G = D_1CD_2$. Substituting equation (1) into this expression and canceling terms, we get that

$$\begin{aligned} S_{rs}^{(k)} &= D_{1,r} \cdot D_{2,s} \cdot \frac{\prod_{1\le i\le k}(x_r - x_i)(y_s - y_i)}{(x_r + y_s)\prod_{1\le i\le k}(x_i + y_s)(x_r + y_i)} \\ &= S_{rs}^{(k-1)} \frac{(x_r - x_k)(y_s - y_k)}{(x_k + y_s)(x_r + y_k)} \end{aligned}$$

where $S_{rs}^{(0)} = D_{1,r}D_{2,s}/(x_r + y_s)$ is the original matrix entry. Thus, updating each entry of the Schur complement from the previous Schur complement costs $c = 8$ operations, or few as $c = 4$ operations if all $2n^2$ possible values of $x_r - x_s$, $y_r - y_s$, and $1/(x_r + y_s)$ are precomputed. This makes the overall cost of the algorithm range from $\frac{8}{3}n^3$ (when done in place, with no extra storage) to $\frac{4}{3}n^3$ (with an extra $2n^2$ storage). The $\frac{8}{3}n^3$ operations include $\frac{2}{3}n^3$ divisions, which are very expensive, whereas the $\frac{4}{3}n^3$ operations are all multiplications. To complete the algorithm, each time the Schur complement is updated, it must be pivoted so that $S_{11}^{(k)}$ is the largest entry; this requires $\frac{1}{3}n^3$ comparisons. Also, after completing the factorization the k-th row and column must be divided by k-th diagonal entry. The entire algorithm is shown below:

Algorithm 3. High Accuracy GECP of a Cauchy-like matrix.

for $r = 1 : n$ and $s = 1 : n$
    $G_{rs} = D_{1,r} D_{2,s} / (x_r + y_s)$
endfor
for $k = 1 : n - 1$
    Find the largest absolute entry in $G(k : n, k : n)$
    Swap rows and columns of G, and entries of x and y, so that G_{kk} is largest
    for $r = k + 1 : n$ and $s = k + 1 : n$ ... overwrite $G(k + 1 : n, k + 1 : n)$ by $S^{(k)}$
        $G_{rs} = G_{rs}(x_r - x_k)(y_s - y_k)/[(x_k + y_s)(x_r + y_k)]$
    endfor
endfor
$D = \mathrm{diag}(G)$
$X = \mathrm{tril}(G) * D^{-1} + I$ ... tril means strict lower triangle
$Y = (D^{-1} * \mathrm{triu}(G) + I)^T$ ... triu means strict upper triangle

All operations involve multiplication, division, and sums and differences of input data, and so are accurate to high relative accuracy.

4.1 Numerical Experiments

Our challenge is to test the accuracy of a routine that we claim is more accurate than any other. We do this in several ways:

1. We can compute the SVD using a conventional algorithm implemented in very high precision arithmetic, and compare the answers to our new algorithm (implemented in Matlab on a Sparc 10, running IEEE double precision arithmetic with $\varepsilon = 2^{-53} \approx 10^{-16}$). We have only performed this rather expensive test for the the 100-by-100 Hilbert matrix, using Mathematica [33] with 200-decimal digit software floating point, and rounding the final answers back to 16 decimal digits, using the command
```
N[SingularValues[N[Table[1/(i+j-1),{i,100},{j,100}],200],Tolerance->10^(-200)],16].
```
The singular values range from about 2.2 down to $5.8 \cdot 10^{-151}$, as shown in the top of Figure 1. If we were to plot the pivots from Algorithm 3 (the $D_{i,i}$) on the same graph, they would be visually nearly indistinguishable; the ratios $D_{i,i}/\sigma_i$ of pivots to singular values all lie in the range [1/16, 16], and 86% are in [1/4,4]. The relative gaps all exceed .6, so we expect all singular vectors from both Mathematica and from our new algorithm to be very accurate. The largest relative difference between a singular value from Mathematica and from our new algorithm was less than than $4 \cdot 10^{-15}$, or about 34ε. The largest angle between a right singular vector from Mathematica and from our new algorithm was less than $6.5 \cdot 10^{-15}$, or 58ε. The largest angle between a left singular vector from Mathematica and from our algorithm was less than $2.8 \cdot 10^{-14}$, or 255ε. A surface plot of the singular vector (or equivalently eigenvector) matrix is shown in the bottom of Figure 1, which reveals interesting patterns of "parallel valleys". This is shown in a different way in Figure 2, which plots dark dots for positive eigenvector components, and leaves white space for the negative eigenvectors components. This amounts to a low resolution contour plot of the eigenvector matrix. The top of Figure 2

shows the results from Mathematica. Although it is somewhat difficult to see, the number of sign changes in eigenvector j (the number of transitions from a positive (black) component to a negative (white) component, or white to black, in column j) is equal to $j-1$. For example, the first column is entirely black (no sign changes) and the last column alternates signs (99 sign changes). Furthermore, the sign changes in eigenvector j interlace those in eigenvector $j+1$ (in the sense that the zero crossings of the piecewise linear curve with corners $(i, U_{i,j})$ interlace the zero crossings of the curve with corners $(i, U_{i,j+1})$). These properties are as predicted by the theory of totally positive (or oscillation) matrices [13, 22], of which the Hilbert matrix is an example. (Totally nonnegative (positive) matrices are those matrices whose every minor is nonnegative (positive). Oscillation matrices are totally nonnegative matrices A such that A^k is totally positive for some positive k.) The bottom of Figure 2 shows the results from the new algorithm. They agree everywhere except for the components whose magnitudes are less than 10^{15}, which are shown in red. The magnitudes of the components quickly drop from 10^{15} to near 10^{-74} in this part of the eigenvector matrix. The 200-decimal digit computation with Mathematica computed all these correctly, but the new algorithm has an $O(\varepsilon)$ error in each component, so the signs of such small components are uncertain.

In contrast, a conventional SVD algorithm in double precision arithmetic (as implemented in Matlab) loses all significant figures for the 79 singular values less than ε, and most of these are wrong by many orders of magnitude.

The next 7 tests use only conventional floating point.

2. We confirm that we have computed an accurate SVD in the conventional sense by measuring

$$Q_1 = \frac{\|G - U\Sigma V^T\|}{\varepsilon\|G\|} + \frac{\|UU^T - I\|}{\varepsilon} + \frac{\|VV^T - I\|}{\varepsilon}$$

and confirming that it is $O(1)$ or perhaps $O(n)$.

3. We compute the SVD of G using a conventional algorithm, and confirm that the singular values agree with the singular values computed by our new algorithm to within $\pm O(\varepsilon)\sigma_1$. We measure this as follows:

$$Q_2 = \max_i \frac{|\sigma_{conventional,i}(G) - \sigma_{new,i}(G)|}{\varepsilon\sigma_{conventional,1}(G)} \quad .$$

Note that Q_2 should be $O(1)$ to confirm that our new algorithm computes the largest singular values (those between σ_1 and $\varepsilon\sigma_1$) at least as accurately as a conventional algorithm. We also confirm that the singular vectors are at least as accurate as those from conventional algorithm, by measuring whether Q_3 is $O(1)$:

$$Q_3 = \max_i \frac{\max(\theta(u_{conventional,i}(G), u_{new,i}(G)), \theta(v_{conventional,i}(G), v_{new,i}(G)))}{\varepsilon / \left(\frac{\min_{j\neq i}|\sigma_{new,i}(G) - \sigma_{new,j}(G)|}{\sigma_{new,1}(G)}\right)} \quad .$$

4. We compute each entry of G^{-1} to high relative accuracy using a well-known formula similar to (1), and compute the SVD of G^{-1} using a conventional algorithm. We confirm that these

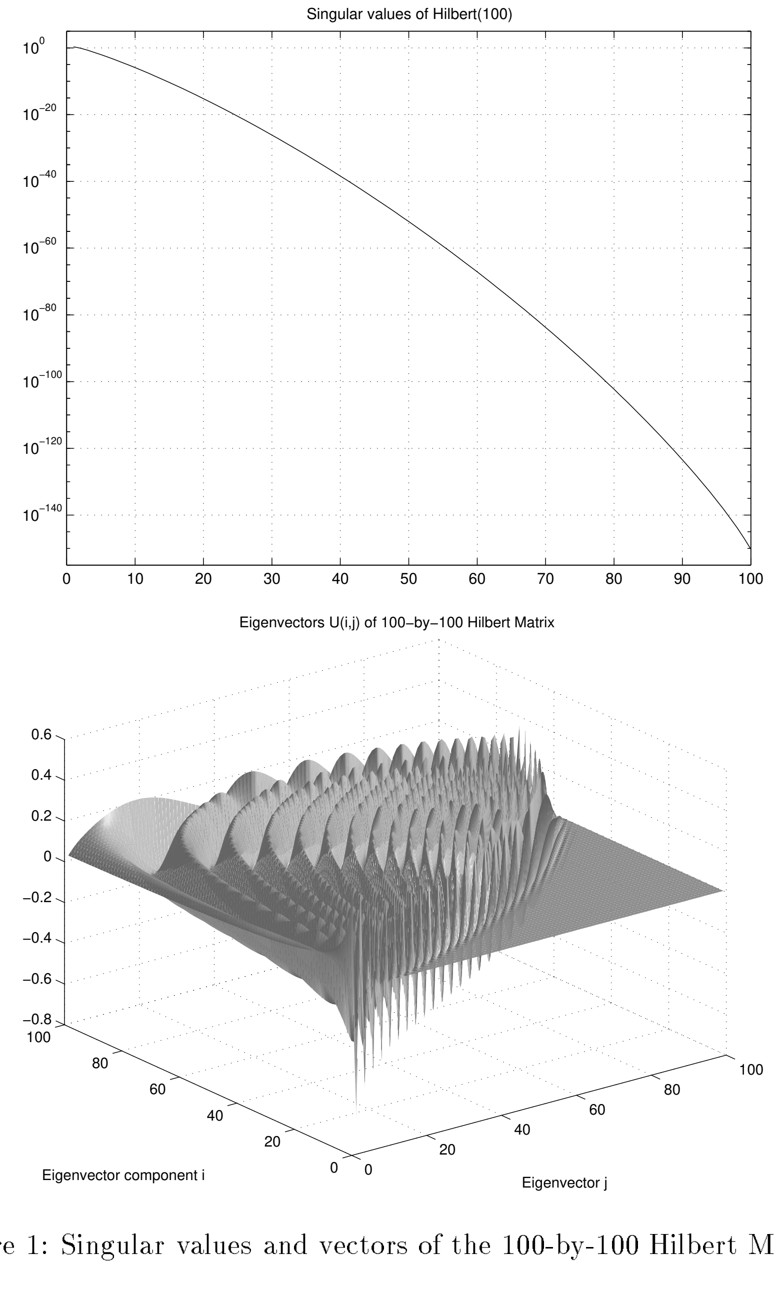

Figure 1: Singular values and vectors of the 100-by-100 Hilbert Matrix

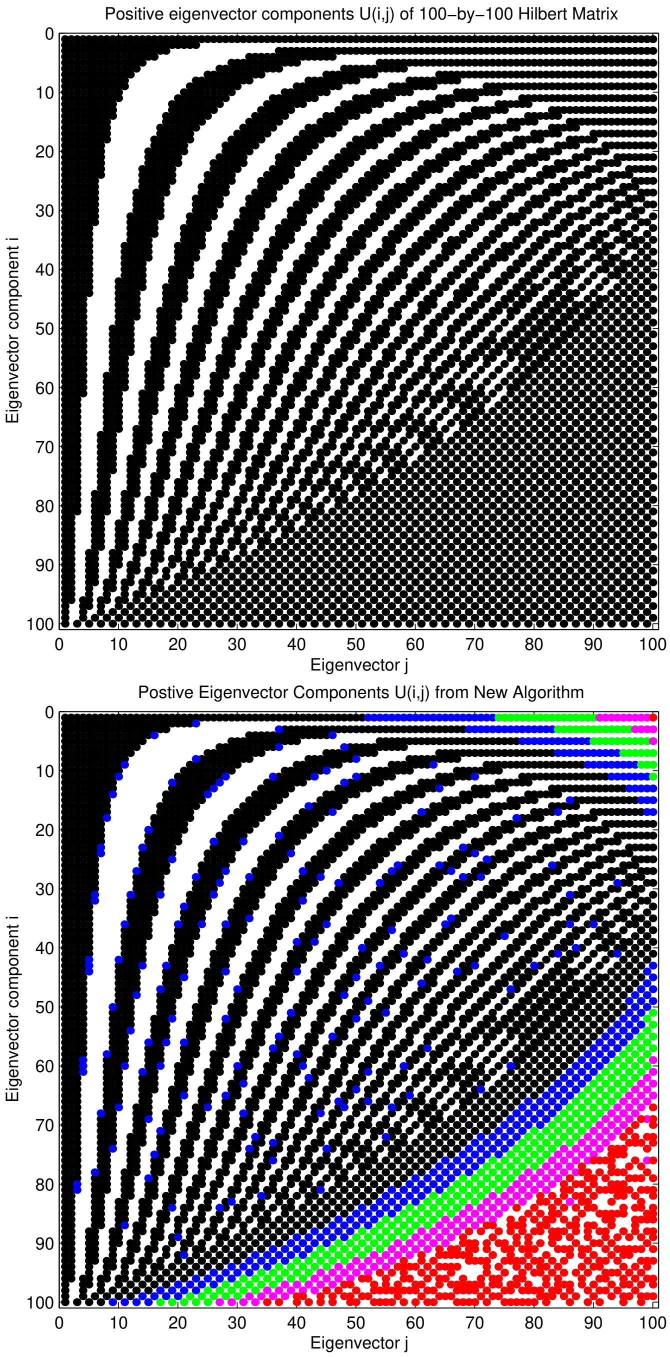

Figure 2: Positive Components of the Singular Vectors of the 100-by-100 Hilbert Matrix. The top picture shows the results from Mathematica, and the lower picture from the new algorithm. In the lower picture, entries are colorcoded by magnitude: Entries in the range (1,.01] are black, in (.01,10^{-5}] are blue, in (10^{-5},10^{-10}] are green, in (10^{-10},10^{-15}] are magenta, and in (10^{-15},0] are red. The $O(10^{-15})$ errors can make the red entries have incorrect signs.

singular values agree with the reciprocals of the eigenvalues computed by our new algorithm to within $\pm O(\varepsilon)/\sigma_n$, by measuring

$$Q_4 = \max_i \frac{|1/\sigma_{new,i}(G) - \sigma_{conventional,i}(G^{-1})|}{\varepsilon/\sigma_{new,n}(G)}$$

to see if it is $O(1)$. This confirms that the smallest singular values of G (those between σ_n and σ_n/ε) are computed accurately. We also confirm that the singular vectors are at least as accurate as those from the conventional algorithm, by confirming that the following quantity is $O(1)$:

$$Q_5 = \max_i \frac{\max(\theta(u_{conventional,i}(G^{-1}), v_{new,i}(G)), \theta(v_{conventional,i}(G^{-1}), u_{new,i}(G)))}{\varepsilon/\left(\frac{\min j\neq i|\sigma^{-1}_{new,i}(G)-\sigma^{-1}_{new,j}(G)|}{\sigma^{-1}_{new,n}(G)}\right)} \ .$$

5. We use the fact that G^{-1} is Cauchy-like to compute its SVD using our new algorithm, and confirm that it agrees (after reciprocating the singular values) with the SVD computed from G. This is measured by

$$Q_6 = \max_i \frac{|\sigma_{new,i}(G) - \sigma^{-1}_{new,i}(G^{-1})|}{\varepsilon\sigma_{new,i}(G)} \ .$$

We also confirm that the singular vectors agree to within the error bound determined by the relative gap, via

$$Q_7 = \max_i \frac{\max(\theta(u_{new,i}(G^{-1}), v_{new,i}(G)), \theta(v_{new,i}(G^{-1}), u_{new,i}(G)))}{\varepsilon/\left(\frac{\min j\neq i|\sigma^{-1}_{new,i}(G)-\sigma^{-1}_{new,j}(G)|}{\sigma^{-1}_{new,i}(G)}\right)} \ .$$

Here are our test cases:

1. 10 Hilbert matrices of dimensions 10, 20, 30, ... , 100 ($x_i = i$, $y_i = i-1$, $D_{1,i} = D_{2,i} = 1$).

2. 10 Hilbert matrices of dimensions 10, 20, ... , 100, with strong diagonal scaling ($x_i = i$, $y_i = i-1$, $D_{1,i} = 10^{20u}$, $D_{2,i} = 10^{20u}$, where u is a random number uniformly distributed from 0 to 1).

3. 10 random Cauchy matrices of dimensions 10, 20, ... , 100 ($x_i = u$, $y_i = -u$, $D_{1,i} = D_{2,i} = 1$, where u is a random number uniformly distributed from 0 to 1).

4. 10 random Cauchy matrices of dimensions 10, 20, ... , 100 ($x_i = 10^{10u}$, $y_i = -10^{10u}$, $D_{1,i} = D_{2,i} = 1$, where u is a random number uniformly distributed from 0 to 1).

5. 10 random Cauchy matrices of dimensions 10, 20, ... , 100 with strong diagonal scaling ($x_i = 10^{10u}$, $y_i = -10^{10u}$, $D_{1,i} = 10^{10u}$, $D_{2,i} = 10^{10u}$, where u is a random number uniformly distributed from 0 to 1).

The results were as follows (given as the maximum values over all 50 test cases). As can be seen, the results were about as accurate as expected.

$\max Q_1$	$\max Q_2$	$\max Q_3$	$\max Q_4$	$\max Q_5$	$\max Q_6$	$\max Q_7$
2752	21	30	21	101	98	137

In addition, we measured some other quantities of numerical interest. The condition numbers of the matrices tested ranged from about 10^8 to 10^{182}, a very wide range. The relative gaps all exceeded 10^{-3}, meaning that all singular vectors were computed with small norm errors, about $10^3\varepsilon$. The condition numbers of the triangular factors X and Y, which limit the accuracy of Algorithm 1, never exceeded 379. Indeed, in tens of thousands of other examples run using a direct search technique [19] in an attempt to maximize $\mu = \max(\max_{ij} |X^{-1}|_{ij}, \max_{ij} |Y^{-1}|_{ij})$, μ never exceeded 7, for $n \leq 20$. Another factor that limits the accuracy of Algorithm 1, the condition number of R' in the statement of Theorem 3.1, never exceeded 5. The number of Jacobi sweeps required for convergence in Algorithm 1 depends very much on how we do one-sided Jacobi. The matrix W to which we apply one-sided Jacobi consists of a well-conditioned matrix premultiplied by a diagonal matrix with widely varying entries, so that the rows of W vary widely in norm. If we apply Jacobi rotations to W from the left (the algorithm analyzed in [9]), then convergence is rapid, with never more than 8 Jacobi sweeps required, and an average of 4.6 sweeps (including one just to confirm convergence, which does not apply any rotations). If we apply Jacobi rotations to W from the right, which has a more satisfying theoretical analysis [10, 25, 24], then convergence is much slower, with the number of sweeps being 50 for modest sized examples, and growing rapidly with n. The above results were run with Jacobi rotations applied on the left.

4.2 The Symmetric Positive Definite Case

Significant simplifications are possible in this case, although the overall algorithm is still $O(n^3)$. If G is symmetric and positive definite, the largest pivot is always on the diagonal. Thus, only the diagonal entries of the $S^{(k)}$ need to be computed to determine the pivot order. This leads to the following algorithm:

Algorithm 4. High Accuracy SVD of a Symmetric Positive-Definite Cauchy-like matrix G.

1. Compute only the diagonals of each $S^{(k)}$, and use this to determine the pivot order. This costs just $O(n^2)$.

2. Reorder the data determining G in the correct order, and compute the reordered G's LDU factorization using Algorithm 2.5 of [3]. This costs $O(n^2)$, and is forward stable for the same reasons as Algorithm 3.

3. Rewrite $LDU = \hat{L}\hat{L}^T$, the Cholesky factorization. This also costs $O(n^2)$.

4. Apply one-sided Jacobi to $\hat{L}$ to compute its SVD. The left singular vectors of L are the eigenvectors of G. This costs $O(n^3)$.

Algorithm 2.5 in [3] and Algorithm 3 above are quite similar, but Algorithm 2.5 takes advantage of a fixed pivot order to compute the final entries of L column by column, and final entries of U row by row, rather than computing all the $S^{(k)}$. To see how, note that all the following factors of entries of $S^{(k)}$ can be computed and then combined in $O(n^2)$ time:

$$\begin{array}{rll}
\prod_{1\le i\le k} (x_r - x_i) & \text{for all} & 1 \le k < r \le n \\
\prod_{1\le i\le k} (y_s - y_i) & \text{for all} & 1 \le k < s \le n \\
x_r + y_s & \text{for all} & 1 \le r, s \le n \\
\prod_{1\le i\le k} (x_i + y_s) & \text{for all} & 1 \le k < s \le n \\
\prod_{1\le i\le k} (x_r + y_i) & \text{for all} & 1 \le k < r \le n
\end{array}$$

Algorithm 2.5 uses only $O(n)$ extra storage, rather than $O(n^2)$.

It is possible to recognize symmetric positive definite generalized Cauchy matrices quite easily: Suppose $D_1 = D_2$ and $x_i = y_i$ for all i, so that G is symmetric. (This last condition can be weakened slightly, since replacing each x_i by $x_i - c$ and each y_i by $y_i + c$ leaves G unchanged. But this must be done without rounding error to guarantee accuracy. This is possible for the Hilbert matrix, with $c = .5$.) Then G is positive definite if and only if all $x_i > 0$, as can be seen from equation (1) applied to all leading principal submatrices. Furthermore, G's rows and columns can be symmetrically permuted (so $x_i < x_{i+1}$) and multiplied by -1 (if any $D_{1,i} < 0$) to make G totally positive. We note that this order is in general not the same as the order determined by pivoting during the algorithm.

A similar approach was also applied to the Hilbert matrix in [25], but using a formula for the Cholesky factor of the Hilbert matrix without pivoting. When $n = 100$, the corresponding L has a condition number in excess of 10^{20}, so relative accuracy is not guaranteed. Complete pivoting is essential.

5 Vandermonde matrices

Let V be an n-by-n Vandermonde matrix, so that $V_{ij} = x_i^{j-1}$. Let F be the n-by-n discrete Fourier transform matrix, so that $F_{ij} = \omega^{(i-1)(j-1)}/\sqrt{n}$, where $\omega = \exp(2\pi\sqrt{-1}/n)$ is a primitive n-th root of unity. Assume for now that no x_i is equal to one of the roots of unity in F; we eliminate this assumption later. Then it is easy to see that VF is Cauchy-like by computing

$$\begin{aligned}
(VF)_{ij} &= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} x_i^k \omega^{(j-1)k} \\
&= \frac{1}{\sqrt{n}} \frac{1 - (x_i\omega^{j-1})^n}{1 - x_i\omega^{j-1}} \\
&= \frac{1 - x_i^n}{\sqrt{n}\omega^{j-1}} \frac{1}{\omega^{1-j} - x_i} \quad .
\end{aligned}$$

Thus, provided we can compute all of $D_{1,i} = 1 - x_i^n$, $D_{2,j} = \omega^{1-j}/\sqrt{n}$, $\omega^{1-j} - x_i$ and $\omega^{1-j} - \omega^{1-k}$ to high relative accuracy, we can use Algorithm 3 to compute the triangular factorization of VF to high relative accuracy, and then the SVD $VF = U\Sigma V^T$ to high relative accuracy. We compute the final SVD via $V = U\Sigma(FV)^T$; multiplying FV increases the angular error in the right singular vectors by only $O(\varepsilon)$.

Now consider that case when some x_i is a root of unity appearing in F. Since x_i is a (real or complex) floating point number, i.e. rational, the only possible roots of unity that x_i can equal are ± 1 and $\pm\sqrt{-1}$. In this case, the i-th row of VF will contain all zeros except for one entry equal to $\sqrt{n}$; thus VF is not Cauchy-like. Here is how to incorporate such rows into the GECP process. Since all entries but one are zero, the row will not participate in any operations until $\sqrt{n}$ is itself selected as the pivot. This simply results in eliminating all other entries in its column, without updating any other nonzero entries. The remaining matrix is Cauchy-like (with possibly other rows with a single nonzero), so the elimination can proceed as before.

Now we consider how to compute $\omega^{1-j} - \omega^{1-k}$ to high relative accuracy. Given a table of powers of ω, we can simply compute $\omega^{1-j} - \omega^{1-k}$ in the obvious way. Since $|\omega^{1-j} - \omega^{1-k}| \geq 2\sin(\pi/n)$ the difference cannot be too small (unless n is truly enormous), so it is computed with relative accuracy $O(n\epsilon)$. One can do better by using formulas like $\cos(\alpha) - \cos(\beta) = 2\sin(\frac{\beta+\alpha}{2})\sin(\frac{\beta-\alpha}{2})$.

Next we discuss how to compute $x_i^n - 1$ and $\omega^{1-j} - x_i$ to high relative accuracy. We distinguish between the cases of x_i real and x_i complex, and deal with the easier real case first. First consider $x_i^n - 1$ when x_i is real. When $x_i^n \leq 0$, we just compute $x_i^n - 1$ straightforwardly, since there is no cancellation. When $x_i^n > 0$, we can either use the formula $(|x_i| - 1)\sum_{j=0}^{n-1} |x_i|^j$ for an $O(n)$ cost, or use a repeated-squaring-like algorithm for $O(\log n)$ cost [21], or use $\mathrm{expm1}(n \cdot \log_e(x_i))$ for $O(1)$ cost, provided that good implementations of $\mathrm{expm1}(z) \equiv e^z - 1$ and $\log_e(z)$ are available [31, 32]. Next consider $\omega^{1-j} - x_i$. Assume that we have a table of values of ω^{1-j} which are accurate to high relative accuracy in the sense of equation (3) (with a tiny complex δ); Assume further that the real values of ω^{1-j} ($+1$ or -1) are *exact* in the table; the complex values can be computed using any reasonable implementations of sine and cosine. Then $\omega^{1-j} - x_i$ can be computed straightforwardly to high relative accuracy for the following reason: If ω^{1-j} is real, then it is exact, and the difference is computed to high relative accuracy. If ω^{1-j} is complex, its imaginary part is at least $\sin(2\pi/n)$, so $\omega^{1-j} - x_i$ cannot be any smaller in magnitude, so the difference is again small in the sense of a tiny complex δ in equation (3).

When x_i is complex, the situation is more complicated. We assume that we have a table of powers of ω, represented as follows: for each ω^k we store $w_{1,k} = \mathrm{round}(\omega^k)$ (the real and imaginary components being rounded to the *nearest* floating point numbers; a slightly larger error can be tolerated too) and $w_{2,k} = \mathrm{round}(\omega_k - w_{1,k})$ (rounding here may be less careful; a few units of error in the last place is tolerable). Later we will return to the computation of this table.

Using this table, we compute $\omega^k - x_i$ as $d_{k,i} = \mathrm{round}(\mathrm{round}(w_{1,k} - x_i) + w_{2,k})$, i.e. with two floating point operations. We claim that $d_{k,i}$ is accurate to high relative accuracy in both real and imaginary components independently. To see this, there are 3 cases:

1. If $x_i = w_{1,k}$ exactly, then $\mathrm{round}(w_{1,k} - x_i) = 0$. Then $d_{k,i} = w_{2,k}$ is correct to as many digits as $w_{2,k}$.

2. If $.5 < w_{1,k}/x_i < 2$ but $x_i \neq w_{1,k}$, then cancellation occurs in $\mathrm{round}(w_{1,k} - x_i)$. Because we assume arithmetic has a guard digit, this nonzero difference is exact, and at least about twice as large in magnitude as $w_{2,k}$ (we use the fact that $w_{1,k} = \mathrm{round}(\omega^k)$ is (nearly) the *nearest* floating point number to ω^k). This means that there is no significant cancellation in $\mathrm{round}(\mathrm{round}(w_{1,k} - x_i) + w_{2,k}) = \mathrm{round}((w_{1,k} - x_i) + w_{2,k})$, so it is computed to high relative accuracy.

3. If $w_{1,k}/x_i \le .5$ or $2 \le w_{1,k}/x_i$, then round$(w_{1,k} - x_i)$ is accurate to high relative accuracy, and since it is at least half as large in magnitude as $w_{1,k}$, adding $w_{2,k}$ does not change it significantly.

To compute $x_i^n - 1$ accurately, we can evaluate it as $\prod_{i=0}^{n-1}(x_i - \omega^i)$, evaluating each difference as described above.

Now we return to the computation of the table of values of $w_{1,k}$ and $w_{2,k}$. The obvious algorithm is to compute $w = \text{round\_to\_double}(\omega^k)$ to double precision, let $w_{1,k} = \text{round\_to\_single}(w)$, and $w_{2,k} = \text{round\_to\_single}(w - w_{1,k})$. The trouble is that $w_{2,k}$ may be much smaller than εw, so that it may have few or no significant digits when computed this way, whereas we rely on it being accurate to single precision. For example, $\cos(2 \cdot 99 \cdot \pi/3565) \approx .9848162531852694$. Rounding this to single yields the real part of $w_{1,99} \approx .9848162531852722$, and subtracting to get the real part of $w_{2,99}$ yields $\approx -2.776 \cdot 10^{-15}$, which only has 5 significant bits. In other words, high precision arithmetic may be needed to compute the table. But since we can empirically confirm that the above example is the worst case up to $n = 10000$, we can assert that *triple* precision is enough up to $n = 10000$, for input data in IEEE single precision. On the other hand, the table only needs to be computed once and for all for each n, so its cost should not be considered part of the algorithm.

The following observation [2] indicates why we expect triple precision to be enough except perhaps in rare cases. By the theory of continued fractions [23, Thm. 19], a rational approximation $p/q \approx \Re\omega^k$ (the real part of ω^k) can only satisfy $|p/q - \omega^k| < 1/(2q^2)$ if p/q is a *convergent* of $\Re\omega^k$, i.e. a truncation of the infinite continued fraction expansion of $\Re\omega^k$ (the same is true for the imaginary part). So unless the floating point approximation $\Re w_{1,k}$ happens to equal this convergent, which seems unlikely, we will have

$$|\Re w_{2,k}| \approx |\Re w_{1,k} - \Re\omega^k| \ge 1/(2q^2) \ .$$

Since $\Re w_{1,k} = p/q$ is a floating point number, $q = 2^e$ where $e \approx \lfloor \log_2 \Re\omega^k \rfloor + b$, where b is the number of bits in the floating point fraction, and $p \approx 2^b$, since the floating point fraction is normalized. The nontrivial values of $\Re\omega^k$ satisfy $|\Re\omega_k| \gtrapprox 2\pi/n$, so

$$\frac{|\Re w_{2,k}|}{|\Re\omega^k|} \gtrapprox \frac{1}{(2q^2)} \cdot \frac{q}{p} = \frac{1}{2pq} \gtrapprox 2^{-2b}(\pi/n) \ .$$

This says that computing ω_k to triple precision means that $\Re w_{2,k}$ will be determined with relative error at most $(n/\pi)2^{-b}$, i.e. nearly single precision.

It is an open problem to avoid the need for this high precision table entirely.

6 Unit Displacement Rank Matrices

The most general situation for which the above high accuracy SVD technique works is described as follows. Assume that G satisfies the Sylvester equation (or *displacement equation* [4, 14, 3, 26])

$$XG + GY = d_1 d_2^T$$

where X and Y are diagonal matrices with diagonal entries x_i and y_j respectively, and d_1 and d_2 are column vectors. Then it is immediate to see that G is Cauchy-like with $G_{ij} = d_{1i}d_{2j}/(x_i + y_j)$.

Now suppose that X and Y are normal matrices with eigendecompositions $X = U_X D_X U_X^T$ and $Y = U_Y D_Y U_Y^T$. Then we can premultiply the above equation by U_X^T and postmultiply it by U_Y to transform it to

$$D_X(U_X^T G U_Y) + (U_X^T G U_Y) D_Y = (U_X^T d_1)(U_Y^T d_2)^T$$

or

$$D_X G' + G' D_Y = d_1' d_2'^T$$

so that G' is Cauchy-like. Thus, if this transform can be performed accurately enough, then we can compute the SVD of $G' = U \Sigma V^T$, and backtransform using U_X and U_Y to get the SVD of $G = (U_X U)\Sigma(U_Y V)^T$.

For example, Vandermonde matrices satisfy the displacement equation with X diagonal and Y a circular shift. The eigenvector matrix of Y is the discrete Fourier transform matrix F, and its eigenvalues are roots of unity. The entries of d_1 are $x_i^n - 1$. This is the transformation used in the previous section.

One can imagine other possibilities besides X and Y diagonal or circular shifts, but it is important that $d_1' = U_X^T d_1$ and $d_2' = U_Y^T d_2$ be computable to high relative accuracy (alternatively, one could say that d_1' and d_2' are the parameters of G that determine its SVD to high relative accuracy).

Now consider the case of X and Y diagonalizable, but not necessarily normal, so that $X = U_X D_X U_X^{-1}$ and $Y = U_Y D_Y U_Y^{-1}$. Then given the SVD $G' = U\Sigma V^T$, $G = (U_X U)\Sigma(U_Y^{-T} V)^T$ is *not* the SVD of G, because U_X and U_Y are not unitary. Instead, we simply convert the RRD $G' = XDY^T$ obtained by Algorithm 3 into the RRD $G = (U_X X) D (U_Y^{-T} Y)^T$, and then apply Algorithm 1 [12].

What about matrices satisfying displacement equations with higher than rank-1 matrices on the right hand side? For example, Toeplitz matrices satisfy such an equation with rank 2. We argue that no high relative accuracy algorithm like ours can exist for Toeplitz matrices, and we believe it is unlikely that they exist for higher displacement rank in general.

It is tricky to argue impossibility results in floating point arithmetic, because one can simulate arbitrary precision floating point using just standard floating point, without "bit-fiddling" [29]. Still, we may argue informally as follows. Consider just a 2-by-2 Toeplitz matrix:

$$T = \begin{bmatrix} a & b \\ c & a \end{bmatrix}$$

If we could compute the singular values of T to high relative accuracy, then by multiplying them we could compute the determinant of T to high relative accuracy (actually just its absolute value, but that is enough for this argument). In other words, we would have a floating point algorithm for evaluating $a^2 - b \cdot c$ to high relative accuracy, without any tricks like splitting a, b or c into half-precision parts and combining them separately. But since $a^2 - b \cdot c$ does not factor into products and quotients of factors we can compute to high relative accuracy (like $x_i - x_j$ for Cauchy and Vandermonde matrices) – indeed, it does not factor at all – we believe that there is no forward stable SVD algorithm for Toeplitz matrices.

To illustrate the difficulties in formalizing this argument, consider a matrix of the form $G_{ij} = d_{1i} d_{2j} / (x_i w_j + y_j z_i)$. This is Cauchy-like, since we can also write $G_{ij} = (d_{1i})/z_i)(d_{2j}/w_j)/((x_i/z_i) + (y_j/w_j))$. The sums and differences that we would need to compute to high relative accuracy in the algorithm of section 4 look like $(x_i/z_i) + (y_j/w_j) = (x_i w_j + y_j z_i)/(w_j z_i)$. These can be evaluated to high relative accuracy using double precision, or in single precision using the techniques in [29].

More generally, if *every* minor of G factors into products and quotients of 1 or 2 term polynomials of degree at most d in the original parameters ($d = 1$ for the Cauchy-like matrices in section 4, $d = 1$ for acyclic matrices [8, 7], and $d = 2$ for the matrix in the last paragraph), then a high accuracy SVD algorithm requiring at most d-tuple precision (for the LU factorization) exists. Whether an acceptably fast algorithm ($O(n^3)$ or better) exists to compute the LU factorization this way is a different question that depends on other structure of the matrix, like sparsity pattern or having low displacement rank.

7 Estimating the Error when the Input is Uncertain

So far we have considered only exact inputs, and shown that we compute the SVD to high relative accuracy, as defined in the introduction. When the parameters x_i, y_i, D_{1i} and D_{2i} defining a Cauchy-like matrix are themselves uncertain, then we may easily compute a relative error bound on the SVD using Theorem 2.1 from [8]:

Theorem 2.1. from [8]. *Let $G = XDY^T$ be an RRD with SVD $G = U\Sigma V^T$, and let $\hat{G} = \hat{X}\hat{D}\hat{Y}^T$ with SVD $\hat{G} = \hat{U}\hat{\Sigma}\hat{V}^T$, where $\hat{X}$, $\hat{D}$ and $\hat{Y}$ are defined as follows:*

$$\begin{aligned}
\hat{X} &= X + \delta X \quad \text{where} \quad \frac{\|\delta X\|}{\|X\|} \leq \epsilon \\
\hat{D} &= D + \delta D \quad \text{where} \quad \delta D \ \text{is diagonal and} \ \frac{|\delta D_{ii}|}{|D_{ii}|} \leq \epsilon \\
\hat{Y} &= Y + \delta Y \quad \text{where} \quad \frac{\|\delta Y\|}{\|Y\|} \leq \epsilon
\end{aligned}$$

where $0 \leq \epsilon < 1$. Let $\eta = \epsilon(2+\epsilon)\max(\kappa(X), \kappa(Y))$ and $\eta' = 2\eta+\eta^2$, where $\kappa(Z) = \sigma_{\max}(Z)/\sigma_{\min}(Z)$ is the condition number of Z. Then the difference between the singular values of G and $\hat{G}$ is bounded as follows

$$\frac{|\sigma_i - \hat{\sigma}_i|}{\sigma_i} \leq \eta' \ . \tag{4}$$

Furthermore, the angle θ between u_i and $\hat{u}_i$ (or between v_i and $\hat{v}_i$) is bounded by

$$\sin\theta \leq \sqrt{2}\left(\frac{1+\eta'}{1-\eta'} \cdot \frac{\eta'}{\min(\mathrm{relgap}_i, 2) - \eta'} + \eta\right) \tag{5}$$

provided that $\min(\mathrm{relgap}_i, 2) \geq \eta'$.

It is easy to see that relative errors of size ϵ in the D_{1i} and D_{2i} can increase the relative errors in the singular values by at most ϵ, and the errors in the singular vectors by at most about ϵ divided by the relative gap. Given uncertainty bounds $x_i \pm \delta x_i$ and $y_i \pm \delta y_i$, we can also easily bound the maximum relative error in any factor like $x_i + y_j$ (namely $\varepsilon + (|\delta x_i| + |\delta y_j|)/|x_i + y_j|$) appearing in the algorithm in section 4. The maximum such factor-wise relative error can be used as an estimate of ϵ in Theorem 2.1 above (this ignores factors depending on the dimension n, which are often pessimistic anyway).

From this point of view the Hilbert matrix is very well-conditioned, since small relative changes in the values of $x_i = i$ and $y_i = i - 1$ cause only small relative changes in the SVD.

Acknowledgements

The author thanks Ming Gu and Vadim Olshevsky for discussion of and pointers to relevant displacement rank literature, Doug Priest, W. Kahan and Peter Tang for discussions of floating point issues, Zlatko Drmač for simplifying and improving some of the algorithms, and David Blackston for pointing out the connection with continued fractions.

References

[1] C. Bischof and G. Quintana-Orti. Computing rank-revealing QR factorizations of dense matrices. Argonne Preprint ANL-MCS-P559-0196, Argonne National Laboratory, 1996.

[2] D. Blackston. private communication, 1997.

[3] T. Boros, T. Kailath, and V. Olshevsky. Pivoting and backward stability of fast algorithms for solving Cauchy linear equations. http://WWW-ISL.Stanford.EDU/people/olshevsk/papers.html, 1994.

[4] T. Boros, T. Kailath, and V. Olshevsky. The fast Björck-Pereyra-type algorithm for parallel solution of Cauchy linear equations. http://WWW-ISL.Stanford.EDU/people/olshevsk/papers.html, 1995.

[5] T. Chan. Rank revealing QR factorizations. *Lin. Alg. Appl.*, 88/89:67–82, 1987.

[6] S. Chandrasekaran and I. Ipsen. On rank-revealing QR factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15, 1994.

[7] J. Demmel and W. Gragg. On computing accurate singular values and eigenvalues of acyclic matrices. *Lin. Alg. Appl.*, 185:203–218, 1993.

[8] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. http://www.netlib.org/lapack/lawns/lawn119.ps, to appear in Lin. Alg. Appl., 1997.

[9] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Mat. Anal. Appl.*, 13(4):1204–1246, 1992. (also LAPACK Working Note #15).

[10] Z. Drmač. Accurate computation of the product induced singular value decomposition with applications. Tech Report CU-CS-816-96, Computer Science Dept., University of Colorado at Boulder, July 1995. submitted to SINUM.

[11] Z. Drmač. On the condition behaviour in the Jacobi method. *SIAM J. Mat. Anal. Appl.*, 17(3):509–514, July 1996.

[12] Z. Drmač. Fast and accurate algorithms for canonical correlations, weighted least squares and related generalized eigenvalue and singular value decompositions. Department of Computer Science, University of Colorado at Boulder, Technical report CU-CS-833-97; submitted to SIAM J. Matrix Anal. Appl., March 1997.

[13] F. Gantmacher and M. Krein. *Oszillationsmatrizen, Oszillationskerne, und kleine Schwingungen mechanischer Systeme.* Akademie-Verlag, Berlin, 1960.

[14] I. Gohberg, T. Kailath, and V. Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995.

[15] G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[16] M. Gu and S. Eisenstat. An efficient algorithm for computing a strong rank–revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848 – 869, 1996.

[17] N. J. Higham. Stability analysis of algorithms for solving confluent Vandermonde-like systems. *SIAM J. Mat. Anal. Appl.*, 11(1):23–41, 1990.

[18] N. J. Higham. *Accuracy and stability of numerical algorithms.* SIAM, 1996.

[19] Nicholas J. Higham. Optimization by direct search in matrix computations. *SIAM J. Mat. Anal. Appl.*, 14(2):317–333, April 1993.

[20] P. Hong and C. T. Pan. The rank revealing QR and SVD. *Math. Comp.*, 58:575–232, 1992.

[21] W. Kahan. private communication, 1997.

[22] S. Karlin. *Total Positivity.* Stanford University Press, 1968.

[23] A. Khinchin. *Continued Fractions.* University of Chicago Press, 1964.

[24] W. Mascarenhas. A note on Jacobi being more accurate than QR. *SIAM J. Mat. Anal. Appl.*, 15(1):215–218, 1993.

[25] R. Mathias. Accurate eigensystem computations by Jacobi methods. *SIAM J. Mat. Anal. Appl.*, 16(3):977–1003, 1996.

[26] V. Olshevsky. Pivoting on structured matrices and applications. http://WWW-ISL.Stanford.EDU/people/olshevsk/papers.html, 1997.

[27] C. T. Pan. On the existence and computation of rank-revealing LU factorizations. submitted to Math. Comp., 1996.

[28] C. T. Pan and P. Tang. Bounds on singular values revealed by QR factorization. Technical Report MCS-P332-1092, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[29] D. Priest. Algorithms for arbitrary precision floating point arithmetic. In P. Kornerup and D. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–145, Grenoble, France, June 26-28 1991. IEEE Computer Society Press.

[30] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Mat. Anal. Appl.*, 14(2):494–499, April 1993.

[31] P. T. P. Tang. Table-driven implementation of the logarithm function for IEEE floating-point arithmetic. *ACM Trans. Math. Soft.*, 16(4):378–400, Dec 1990.

[32] P. T. P. Tang. Table-driven implementation of the Expm1 function for IEEE floating-point arithmetic. *ACM Trans. Math. Soft.*, 18(2):211–222, June 1992.

[33] S. Wolfram. *Mathematica: A system for doing mathematics by computer.* Addison-Wesley, 1988.