2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Accurate Matrix Singular Values

Takeshi Ogita†

†Division of Mathematical Sciences, School of Arts and Sciences
Tokyo Woman's Christian University
and
JST, CREST
2-6-1 Zempukuji, Suginami-ku, Tokyo 167-8585, Japan
Email: ogita@lab.twcu.ac.jp

Abstract—In this paper, an algorithm for accurately calculating singular values of matrices is proposed. The proposed algorithm can treat the cases where the matrices are extremely ill-conditioned, i.e. their condition numbers are allowed to go far beyond the bounds of base precision such as IEEE standard 754 double precision. The algorithm requires standard numerical algorithms, which are commonly implemented in several numerical libraries such as BLAS and LAPACK, and an algorithm for accurate matrix multiplication. Numerical results are presented for illustrating the performance of the proposed algorithm.

1. Introduction

This paper is concerned with accurate singular value decomposition (SVD) of an extremely ill-conditioned matrix: Let $A \in \mathbb{R}^{n\times n}$. Then the SVD of A is expressed by

$$A = U\Sigma V^T, \quad U, V, \Sigma \in \mathbb{R}^{n\times n}$$

where both U and V are orthogonal matrices consisting of all singular vectors, and Σ is a diagonal matrix such that

$$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n).$$

Here σ_i for $i = 1, 2, \ldots, n$ are singular values of A.

We regard A as ill-conditioned, if condition number of A is very large. Here the condition number of A is defined by

$$\kappa(A) = \|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max} := \max_{1\le i\le n} \sigma_i$ and $\sigma_{\min} := \min_{1\le i\le n} \sigma_i$.

Let $\mathbb{F}$ be a set of floating-point numbers. Let $\mathbf{u}$ denote the unit round-off of floating-point arithmetic. In IEEE 754 double precision (binary64), $\mathbf{u} = 2^{-53} \approx 1.1 \times 10^{-16}$. In general, if $\kappa(A)$ is large, then errors of results obtained by floating-point arithmetic for linear systems, eigenvalue problems and singular value problems also become large. For example, let us consider a linear system $Ax = b$ with A being nonsingular. Let $x^* = A^{-1}b$, which is the exact solution of $Ax = b$. Let $\tilde{x}$ denote an approximate solution of $Ax = b$ by a standard numerical algorithm such as Gaussian elimination. Then it holds that [5]

$$\frac{\|x^* - \tilde{x}\|}{\|x^*\|} \approx O(\mathbf{u})\kappa(A), \quad 1 \le i \le n.$$

Therefore, if $\kappa(A) > \mathbf{u}^{-1}$, then we cannot expect the accuracy of $\tilde{x}$. It is also true for eigenvalue problems and singular value problems.

The purpose of this article is to propose an algorithm for accurately calculating singular values of a matrix whose condition number goes far beyond $\mathbf{u}^{-1}$. Such cases have been treated in [7, 8] for accurate matrix factorizations.

We also consider symmetric eigenvalue problems: Let $A = A^T \in \mathbb{R}^{n\times n}$. Then the eigenvalue decomposition of A is expressed by

$$A = XDX^T, \quad X, \Sigma \in \mathbb{R}^{n\times n}$$

where X is an orthogonal matrix and consists of all eigenvectors of A, and D is a diagonal matrix such that

$$D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n).$$

Here λ_i for $i = 1, 2, \ldots, n$ are eigenvalues of A.

From the relation between eigenvalues and singular values of a symmetric matrix, there is some k satisfying $|\lambda_k| = \sigma_i$, $i = 1, 2, \ldots, n$. Throughout the paper, we assume that

$$\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_n > 0.$$

2. Accurate matrix multiplication

Suppose for $x, y \in \mathbb{R}^n$ a dot product $x^T y$ can be computed in arbitrarily high accuracy. Namely, for any k, ℓ $(k \ge \ell)$ we can obtain $s_\ell := \sum_{i=1}^{\ell} s^{(i)}$ with $s^{(i)}$ being floating-point numbers satisfying

$$\left|\frac{x^T y - s_\ell}{x^T y}\right| \le O(\mathbf{u}^\ell) + O(\mathbf{u}^k)\mathrm{cond}(x^T y), \quad (x^T y \ne 0). \quad (1)$$

Here $\mathrm{cond}(x^T y)$ denotes condition number of dot product [9] such that

$$\mathrm{cond}(x^T y) := 2\frac{|x^T||y|}{|x^T y|}, \quad x^T y \ne 0.$$

Therefore, Eq. (1) means that we need to calculate $x^T y$ in k-fold base precision and round the result into ℓ-fold base precision. To satisfy Eq. (1), we can use algorithms in [9,

10, 11, 13, 14], that are based on error-free transformation of floating-point arithmetic.

We extend it to matrix multiplication. We use the following notation

$$C_\ell = \{A \cdot B\}_k^\ell,$$

which means for $A \in \mathbb{R}^{m\times p}$ ٮ $B \in \mathbb{R}^{p\times n}$ it holds that

$$|AB - C_\ell| \le O(\mathbf{u}^\ell)|AB| + O(\mathbf{u}^k)|A||B|.$$

Here $|A|$ denotes the matrix of taking the absolute value of A componentwise. Moreover, inequality for matrices such as $A \le B$ means $A_{ij} \le B_{ij}$ for all (i, j). If $\ell = 1$, then we abbreviate $C = \{A \cdot B\}_k$.

3. Algorithm for accurate SVD

Several algorithms for SVD [1, 2, 3, 6] have been proposed. Most of the SVD algorithms are backward stable; Let $\tilde{\sigma}_i$ denote approximations of σ_i. Then it holds that

$$\frac{|\sigma_i - \tilde{\sigma}_i|}{\sigma_1} \approx O(\mathbf{u}), \quad 1 \le i \le n. \tag{2}$$

It is equivalent to

$$\frac{|\sigma_i - \tilde{\sigma}_i|}{\sigma_n} \approx O(\mathbf{u})\kappa(A), \quad 1 \le i \le n.$$

Thus, if $\kappa(A) = \sigma_1/\sigma_n$ is large, then the accuracy of relatively small singular values become worse.

We present the following algorithm for accurate SVD. When we input $A \in \mathbb{R}^{n\times n}$ and a tolerance $\varepsilon_{\rm tol} > \mathbf{u}$, the algorithm outputs $\tilde{\sigma}_i \in \mathbb{F}$, $1 \le i \le n$ satisfying

$$\frac{|\sigma_i - \tilde{\sigma}_i|}{\sigma_i} \lesssim \varepsilon_{\rm tol}, \quad 1 \le i \le n.$$

Algorithm 1 *Accurate singular value decomposition.*

0: *Input $A \in \mathbb{R}^{n\times n}$ and tolerance $\varepsilon_{\rm tol} > \mathbf{u}$.*
1: *Put $U_0 = V_0 = I$ and $k = 1$.*
2: $T \leftarrow \{U_{k-1}^T \cdot A\}_k$.
3: $B_k \leftarrow T \cdot V_{k-1}$.
4: $\tilde{\sigma}_i = (B_k)_{ii}$, $g_i = \sum_{j\ne i} |B_k|_{ij}$ *for all i.*
5: *If $\varepsilon_{\rm tol} \cdot \tilde{\sigma}_i \ge g_i$ for all i, then*
$U = U_k$, $V = V_k$, $\Sigma = \mathrm{diag}(\tilde{\sigma}_i)$ and stop.
6: *SVD of B_k: $B_k \approx W_k \Sigma_k V_k^T$.*
7: $U_k \leftarrow \{U_{k-1} \cdot W_k\}_k^k$.
8: *Update $k \leftarrow k + 1$ and return to 2.*

Higher precision arithmetic is needed only in Steps 2 and 7. For the SVD algorithm in Step 6, it is necessary to satisfy Eq. (2).

4. Algorithm for accurate eigenvalue decomposition

Let $A = A^T \in \mathbb{R}^{n\times n}$. Let σ_i and λ_i, $i = 1, 2, \ldots, n$ be singular values and eigenvalues of A, respectively. Suppose

$$|\lambda_1| \ge |\lambda_2| \ge \ldots \ge |\lambda_n| > 0.$$

Then it holds that

$$\sigma_i = |\lambda_i|, \quad i = 1, 2, \ldots, n.$$

For the singular value decomposition $A = U\Sigma V^T$ and the eigenvalue decomposition $A = XDX^T$, there exists a diagonal matrix $S \in \mathbb{R}^{n\times n}$ satisfying

$$\Sigma = DS, \quad |S_{ii}| = 1, \ i = 1, 2, \ldots, n.$$

Here eigenvectors and singular vectors of A are equivalent because A is symmetric, so that we can put $X = U$. Then, we have

$$A = U\Sigma V^T = UDSV^T.$$

If D is nonsingular, then $SV^T = X^T = U^T$, which yields

$$S = U^T V.$$

Since S is diagonal, it is sufficient to compute

$$S_{ii} = \sum_{k=1}^{n} U_{ki}V_{ki}, \quad i = 1, 2, \ldots, n$$

and set

$$\lambda_i = S_{ii}\sigma_i, \quad i = 1, 2, \ldots, n.$$

We present the following algorithm for accurate eigenvalue decomposition. When we input $A = A^T \in \mathbb{R}^{n\times n}$ and a tolerance $\varepsilon_{\rm tol} > \mathbf{u}$, the algorithm outputs $\tilde{\lambda}_i \in \mathbb{F}$, $1 \le i \le n$ satisfying

$$\left|\frac{\lambda_i - \tilde{\lambda}_i}{\lambda_i}\right| \lesssim \varepsilon_{\rm tol}, \quad 1 \le i \le n.$$

Algorithm 2 *Accurate eigenvalue decomposition for symmetric matrices.*

0: *Input $A = A^T \in \mathbb{R}^{n\times n}$ and tolerance $\varepsilon_{\rm tol} > \mathbf{u}$.*
1: *Compute an accurate SVD of A s.t.*
$A \approx U\Sigma V^T$ by Algorithm 1.
2: *for $i = 1 : n$*
$s_i = \mathrm{sign}\left(\sum_{k=1}^n U_{ki}V_{ki}\right)$;
$\tilde{\lambda}_i = s_i \cdot \Sigma_{ii}$;
end
3: *$X = U$, $D = \mathrm{diag}(\tilde{\lambda}_i)$ and stop.*

In Step 2, we use the sign function sign for enforcing $s_i = \pm 1$, because U and V are not exactly orthogonal due to the rounding errors, and the computation $s_i = \sum_{k=1}^n U_{ki}V_{ki}$ also involves rounding errors.

5. Numerical results

We evaluate the performance of the proposed algorithm (Algorithm 2), which also includes Algorithm 1. We use a PC with 1.86 GHz Intel Core 2 Duo CPU and Matlab 2009a with INTLAB 5.5 [12]. All computations are done in IEEE 754 double precision arithmetic ($\mathbf{u} = 2^{-53} \approx 10^{-16}$). To generate test matrices, we adopt the following Matlab function:

```
function A = randmatsym(n,cnd)
% A real symmetric n-by-n matrix A is
% generated with a specified condition
% number cnd.

d = logspace(1,log10(cnd),n);
D = diag((-1).^[1:n].*d);
X = randorth(n);  % random orthogonal
k = ceil(log(cnd)/log(2^53));

% Accurate computation of X*D*X' with
% k-fold working precision
T = acc_mm(D,X',{0 k});
A = acc_mm(X,T,{0 k});

% Enforce symmetricity of A
if iscell(A)
    for i=1:length(A)
        L = tril(A{i},-1);
        A{i} = L + L' + diag(diag(A{i}));
    end
else
    A = 0.5*(A + A');
end
```

Using this function, we can generate a random symmetric matrix A with specified dimension n and condition number cnd. Moreover, the matrix A has almost one half each of positive and negative eigenvalues. If cnd is greater than $\mathbf{u}^{-1}$, we use a cell array to express A as

$$\mathtt{A} = \mathtt{A\{1\}} + \mathtt{A\{2\}} + \ldots + \mathtt{A\{k\}}$$

to avoid the reduction of condition number due to the rounding errors.

First, we treat a matrix with condition number being almost the limit of double precision arithmetic. We set n = 5 and cnd = 10^{15} as follows:

```
>> n=5; cnd=1e15; A=randmatsym(n,cnd);
>> d=sort(eig(sym(A,'f'))) % symbolic comput.
d =
  -1000000000000000.175990519298626
 -100000000.005539146395567419559 06
  -10.010092311006842997511963193 67
  31622.778285161926155448656332726
  316227766016.827399314774880992 94
```

Here d is obtained by Symbolic Math Toolbox, so that it has very high accuracy. Thus, we can regard d as the exact eigenvalues of A.

We compute eigenvalues by a Matlab's built-in function eig, which calls LAPACK's DSYEV using a standard numerical algorithm for symmetric eigenvalue problems.

```
>> d1=sort(eig(A)) % Matlab built-in (LAPACK)
d1 =
    -1.000000000000000e+15
    -9.999999994796611e+07
    -9.995486841962371e+00
     3.162278320369064e+04
     3.162277660168737e+11
>> abs(double((d-d1)./d)) % relative error
ans =
     1.759905192986260e-16
     5.757303270581678e-10
     1.459074361223595e-03
     1.555375265686662e-07
     1.464733711631263e-13
```

It can be seen from this result that by the standard numerical algorithm, the accuracy of relatively large computed eigenvalues is high in terms of relative error, while that of relatively small ones is low. Therefore, if the condition number becomes larger, then it is estimated that the standard numerical algorithm cannot compute a good approximation for relatively small eigenvalues.

We next compute eigenvalues by the proposed algorithm (Algorithm 2), which is implemented for Matlab as a function acceig.

```
>> d2=sort(acceig(A)) % proposed algorithm
k = 2
k = 3
d2 =
    -1.000000000000000e+15
    -1.000000000055391e+08
    -1.001009231100685e+01
     3.162277828516193e+04
     3.162277660168274e+11
>> abs(double((d-d2)./d)) % relative error
ans =
     1.759905192986260e-16
     4.205310678490546e-16
     2.378162025519442e-16
     1.341712723442831e-16
     2.130315742304067e-17
```

It can be seen that by the proposed algorithm, highly accurate results are obtained, even for relatively small eigenvalues.

As the second example, we fix n to 100 and vary cnd as 10^{20}, 10^{40}, ..., 10^{100}. Table 1 displays computing time and the maximum relative error for approximate eigenvalues obtained by the proposed algorithm. Moreover, the number of iteration k in Algorithm 1, which is called from Algortihm 2, is also shown. We set $\varepsilon_{\mathrm{tol}} = 10^{-6}$ as a tolerance for the proposed algorithm.

Here the proposed algorithm increases the number of iterations adapting to the condition number. It turns out that highly accurate eigenvalues can efficiently be computed by the proposed algorithm even for ill-conditioned problems.

Table 1: Results by the proposed algorithm: Computing time t, maximum relative error and the number of iterations k ($\mathtt{n} = 100$).

$\kappa(A)$	t (sec)	Maximum relative error	k
10^{20}	0.24	2.78×10^{-11}	3
10^{40}	1.02	6.22×10^{-13}	5
10^{60}	1.96	2.98×10^{-13}	6
10^{80}	3.51	5.11×10^{-11}	7
10^{100}	6.87	4.17×10^{-13}	9

6. Conclusion

In this paper, we proposed accurate numerical algorithms for singular value problems and symmetric eigenvalue problems. In general, the condition number of a given matrix is not known in advance, so that we do not know how high computational precision is required. Therefore, it is difficult for a standard numerical algorithm to compute accurate results for all eigenvalues. Using the proposed algorithms, it is possible to do it adaptively increasing the computational precision.

The proposed algorithms consist of standard numerical algorithms, that are implemented in LAPACK, and algorithm for accurate matrix multiplication. Moreover, it is shown in [10, 11] that accurate matrix multiplication can efficiently be implemented using BLAS. Thus, the proposed algorithms have scalability and portability.

To prove the convergence of the proposed algorithms remains to be solved.

References

[1] E. Anderson et al.: LAPACK User's Guide, Third Edition, SIAM, Philadelphia, 1999.

[2] J. Demmel, W. Kahan: Computing small singular values of bidiagonal matrices with guaranteed high relative accuracy, SIAM J. Sci. Statist. Comput., 11 (1990), 873–912.

[3] K. V. Fernando, B. N. Parlett: Accurate singular values and differential qd algorithms, Numer. Math., 67 (1994), 191–229.

[4] G. Golub, W. Kahan: Calculating the singular values and pseudo-inverse of a matrix J. Soc. Indus. Appl. Math.: Ser. B, 2 (1965), 205–224.

[5] N. J. Higham: Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM Publications, Philadelphia, 2002.

[6] M. Iwasaki, Y. Nakamura: Accurate computation of singular values in terms of shifted integrable schemes, Japan J. Indust. Appl. Math. 23 (2006), 239–259.

[7] T. Ogita: Accurate matrix factorization: Inverse LU and inverse QR factorizations, SIAM J. Matrix Anal. Appl., to appear.

[8] T. Ogita, S. Oishi: Accurate and robust inverse Cholesky factorization, submitted for publication.

[9] T. Ogita, S. M. Rump, S. Oishi: Accurate sum and dot product, SIAM J. Sci. Comput., 26:6 (2005), 1955–1988.

[10] K. Ozaki, T. Ogita, S. Oishi: Tight and Efficient Enclosure of Matrix Multiplication by Using Optimized BLAS, Numerical Linear Algebra with Applications, accepted for publication.

[11] K. Ozaki, T. Ogita, S. Oishi, S. M. Rump: Error-Free Transformation of Matrix Multiplication by Using Fast Routines of Matrix Multiplication and its Applications, submitted for publication.

[12] S. M. Rump: *INTLAB – INTerval LABoratory*, Developments in Reliable Computing (Tibor Csendes ed.), Kluwer Academic Publishers, Dordrecht, 1999, 77–104.
`http://www.ti3.tu-harburg.de/~rump/intlab/`

[13] S. M. Rump, T. Ogita, S. Oishi: Accurate floating-point summation part I: Faithful rounding, SIAM J. Sci. Comput., 31:1 (2008), 189-224.

[14] S. M. Rump, T. Ogita, S. Oishi: Accurate floating-point summation part II: Sign, K-fold faithful and rounding to nearest, SIAM J. Sci. Comput., 31:2 (2008), 1269-1302.