数理解析研究所講究録
第2005巻 2016年 43-55

Spectral divide-and-conquer algorithms for matrix eigenvalue problems

Yuji Nakatsukasa
Department of Mathematical Informatics, University of Tokyo

Conventional algorithms for the (symmetric or non-symmetric) eigenvalue decomposition and the singular value decomposition (SVD) are based on initially reducing the matrix to a condensed (tridiagonal, Hessenberg or bidiagonal) form. Unfortunately, they are not optimal in view of recent trends in computer architectures, which require minimizing communication along with the arithmetic cost. With collaborators the author has been developing spectral divide-and-conquer algorithms, which can achieve both requirements. Spectral divide-and-conquer algorithms recursively decouple the problem into two smaller subproblems. This report summarizes the developments thus far and gives an overview of spectral divide-and-conquer algorithms for eigenvalue problems and the SVD, and point to ongoing directions. A large bulk of this report consists of collaborations with Nicholas J. Higham [19] and Roland W. Freund [20].

1 Introduction

There has been much recent progress on designing algorithms that reduce communication in addition to the arithmetic cost [5], so that they are well-suited for parallel computing. While in [5] many basic matrix operations–such as matrix multiplication, LU, QR and Cholesky factorizations–are shown to have an implementation that minimizes communication, notable exceptions are the eigenvalue decomposition and the SVD. Conventional algorithms for the (symmetric or non-symmetric) eigenvalue decomposition and the singular value decomposition (SVD) are based on initially reducing the matrix to a condensed form, such as tridiagonal, Hessenberg or bidiagonal [9], and implementing them in a communication-minimizing manner is highly nontrivial.

Ballard, Demmel, and Dumitriu [4] have developed the spectral divide-and-conquer algorithm of Bai, Demmel, and Gu [2], into algorithms that achieve, asymptotically, lower bounds on the costs of communication.

These algorithms, along with other existing spectral divide-and-conquer methods, generally require significantly more arithmetic than the standard algorithms based on reduction to condensed (tridiagonal or bidiagonal) form (the scaled Newton approach is an exception but its stability was worse than standard algorithms in our experiments). In addition, none of them has been proven to be backward stable in floating point arithmetic.

In this line of studies the authors introduce spectral divide-and-conquer algorithms for the symmetric eigenvalue decomposition and the SVD that are

- proven to be backward stable (under mild assumptions that hold in practice),

- minimize communication while having arithmetic costs within a small factor of those for the standard methods,
- parallelizable in many levels.

Experiments in MATLAB with a sequential implementation suggest that the proposed algorithms can potentially outperform standard algorithms, both in speed and stability. It is therefore of much interest to develop codes optimized for a parallel computing architecture and examine its performance.

2 Outline of algorithms

Here we summarize the basis for the algorithms for the symmetric eigenproblem and the SVD.

2.1 Symmetric eigendecomposition

We first explain the essential idea for the symmetric eigenvalue decomposition. Let $A \in \mathbb{R}^{n\times n}$ be symmetric. We describe how to compute an invariant subspace of A corresponding to the positive (or negative) eigenvalues using the polar decomposition (see Section 3 for the polar decomposition and its computation). For simplicity we assume that A is nonsingular; the singular case is discussed in the papers [19, 20]. The first step is to note the connection between the polar decomposition of a symmetric A and its eigendecomposition. Let $A = U_pH$ be the polar decomposition and let $A = V\Lambda V^*$, with $\Lambda = \mathrm{diag}(\Lambda_+, \Lambda_-)$, be an eigendecomposition, where the diagonal matrices Λ_+ and Λ_- contain the positive and negative eigenvalues, respectively. If there are k positive eigenvalues then

$$\begin{aligned} A &= V\mathrm{diag}(\Lambda_+, \Lambda_-)V^* \\ &= V\mathrm{diag}(I_k, -I_{n-k})V^* \cdot V\mathrm{diag}(\Lambda_+, |\Lambda_-|)V^* \\ &\equiv U_pH. \end{aligned} \tag{1}$$

Suppose we have computed U_p in (1) using the QDWH algorithm, and partition $V = [V_1,\ V_2]$ conformably with Λ. Note that

$$U_p + I = [V_1\ V_2]\begin{bmatrix} I_k & 0 \\ 0 & -I_{n-k} \end{bmatrix}[V_1\ V_2]^* + I = [V_1\ V_2]\begin{bmatrix} 2I_k & 0 \\ 0 & 0 \end{bmatrix}[V_1\ V_2]^* = 2V_1V_1^*,$$

so the symmetric matrix $C = \frac{1}{2}(U_p + I) = V_1V_1^*$ is an orthogonal projector onto $\mathrm{span}(V_1)$, which is the invariant subspace corresponding to the positive eigenvalues. Hence we can compute $\mathrm{span}(V_1)$ by computing an orthogonal basis for the column space of C, for which one step of subspace iteration usually suffices.

As proven in [19], the symmetric eigenvalue decomposition computed in this way is backward stable, which means the computed matrix of eigenvalues $\widehat{\Lambda}$ and eigenvector matrix $\widehat{V}$ satisfy $\widehat{V}^*\widehat{V} = I + \epsilon$ and $A - \widehat{V}\widehat{\Lambda}\widehat{V}^* + \epsilon\|A\|$ where ϵ here represents a matrix of norm $\mathcal{O}(u)$, where u is the unit roundoff.

The condition under which backward stability is established is that the polar decomposition is computed in a backward stable manner, and subspace iteration is successful. Both these conditions can be established [18] and easily verified.

2.2 SVD

For the SVD, we basically follow the strategy introduced in [14], replacing the symmetric eigendecomposition by the algorithm just described: We first compute the polar decomposition $A = U_p H$ and then the symmetric eigendecomposition $H = V\Sigma V^*$. Combining the two, the SVD is obtained from $A = (U_p V)\Sigma V^*$.

The SVD computed via their framework is known [14] to be backward stable provided that both the polar decomposition and the eigendecomposition are computed in a backward stable manner.

2.3 Polar decomposition as the key computational kernel

As described above, one natural way to execute spectral divide-and-conquer is via the polar decomposition. For computing the polar decomposition, the scaled Newton and QDWH iterations (which we outline below) are two of the most popular algorithms, as they are backward stable and converge in at most nine and six iterations, respectively.

However, the experiments in [19] indicate that the QDWH-based algorithms are still considerably slower on a sequential machine (by a factor between 2 and 4) than the fastest standard algorithm. Although the relative performance is expected to be better on parallel systems, further improvements to the algorithm itself are therefore much desired. Since the spectral divide-and-conquer algorithms use the polar decomposition as fundamental building blocks, the design of an improved algorithm for $A = U_p H$ would directly lead to improved spectral divide-and-conquer algorithms for the symmetric eigendecomposition and the SVD.

Following this framework, in [20] we propose a new higher-order variant of the QDWH iteration. The key idea of this algorithm comes from approximation theory: we use the best rational approximant for the scalar sign function due to Zolotarev in 1877, which lets the algorithm converge in just *two* iterations, with the whopping rate of convergence *seventeen*. The algorithm employs a high-degree Zolotarev function (best rational approximant) obtained by composing low-degree Zolotarev functions, an extraordinary property enjoyed by the sign function. The resulting algorithms for the polar, symmetric eigenvalue, and singular value decompositions have higher arithmetic costs than the QDWH-based algorithms but are better-suited for parallel computing, and exhibit excellent numerical backward stability. In this report we briefly outline the main ideas. We refer to [20] for details and the analyses.

3 The QDWH algorithm as a rational approximation to $\operatorname{sign}(x)$

Any rectangular matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ has a polar decomposition [12, Thm. 8.1], [15, Sec. 7.3]

$$A = U_p H, \tag{2}$$

where U_p has orthonormal columns and H is symmetric positive semidefinite. The polar decomposition (2) is unique if A has full column rank.

The QDWH algorithm [17] computes the unitary polar factor U_p of a full-rank matrix A as the limit of the sequence X_k defined by

$$X_{k+1} = X_k(a_k I + b_k X_k^* X_k)(I + c_k X_k^* X_k)^{-1}, \qquad X_0 = A/\alpha. \tag{3}$$

Here, $\alpha > 0$ is an estimate of $\|A\|_2$ such that $\alpha \gtrsim \|A\|_2$. Setting $a_k = 3$, $b_k = 1$, $c_k = 3$ gives the Halley iteration, which is the cubically convergent member of the family of principal Padé iterations [12, Sec. 8.5]. The iterates (3) preserve the singular vectors while mapping the singular values by a rational function $R_k(\cdots R_2(R_1(x))$, that is, $X_k = U\Sigma_k V^*$, where $\Sigma_k = R_k(\cdots R_2(R_1(\Sigma))$ is the diagonal matrix with ith diagonal $R_k(\cdots R_2(R_1(\sigma_i))$; equivalently $R(\Sigma)$ denotes the matrix function in the classical sense [12]. The choice of the rational functions $R_k(x)$ is of crucial importance, and in QDWH $R_k(x) = x\frac{a_k+b_k x^2}{1+c_k x^2}$, in which the parameters a_k, b_k, c_k are chosen dynamically to speed up the convergence. They are computed by $a_k = h(\ell_k)$, $b_k = (a_k - 1)^2/4$, $c_k = a_k + b_k - 1$, where $h(\ell) = \sqrt{1+\gamma} + \frac{1}{2}\big(8 - 4\gamma + 8(2-\ell^2)/(\ell^2\sqrt{1+\gamma})\big)^{1/2}$, $\gamma = \big(4(1-\ell^2)/\ell^4\big)^{1/3}$. Here, ℓ_k is a lower bound for the smallest singular value of X_k, which is computed from the recurrence $\ell_k = \ell_{k-1}(a_{k-1} + b_{k-1}\ell_{k-1}^2)/(1 + c_{k-1}\ell_{k-1}^2)$ for $k \geq 1$. Note that all the parameters are available for free (without any matrix computations) for all $k \geq 0$ once we have estimates $\alpha \gtrsim \|A\|_2$ and $\ell_0 \lesssim \sigma_{\min}(X_0)$, obtained for example via a condition number estimator.

With such parameters the iteration (3) is cubically convergent and needs at most six iterations for convergence to U_p with the tolerance u for any matrix A with $\kappa_2(A) \leq u^{-1}$, that is, $\|X_6 - U_p\|_2 = \mathcal{O}(u)$.

The iteration (3) has a mathematically equivalent QR-based implementation, which is numerically more stable (this is the actual QDWH iteration):

$$\begin{bmatrix} \sqrt{c_k}X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, \quad X_{k+1} = \frac{b_k}{c_k}X_k + \frac{1}{\sqrt{c_k}}\left(a_k - \frac{b_k}{c_k}\right) Q_1 Q_2^*, \quad k \geq 0. \tag{4a}$$

Once the computed polar factor $\widehat{U}_p$ is obtained, we compute the symmetric polar factor $\widehat{H}$ by $\widehat{H} = \frac{1}{2}(\widehat{U}_p^* A + (\widehat{U}_p^* A)^*)$ [12, Sec. 8.8].

4 Higher-order variant

In light of the above observation, a natural idea is to consider approximations $R \in \mathcal{R}_{2r+1,2r}$ of the sign function $\operatorname{sign}(x)$ for general $r \geq 1$, with the goal to map all the singular values to 1. Since $\operatorname{sign}(x)$ is an odd function, the optimal approximant in $\mathcal{R}_{2r+1,2r}$ has the form

$R(x) \equiv x\frac{P(x^2)}{Q(x^2)}$, where $P, Q \in \mathcal{P}_r$. As in the QDWH case, one way to obtain P and Q is to solve the max-min problem

$$\max_{P,Q\in\mathcal{P}_r} \min_{\ell\le x\le 1} x\frac{P(x^2)}{Q(x^2)} \tag{5}$$

subject to the constraint $x\frac{P(x^2)}{Q(x^2)} \le 1$ on $[0,1]$. However, solving (5) via extending the approach in [17] for $r = 1$ to $r \ge 2$ seems highly nontrivial.

Fortunately, the max-min problem (5) is equivalent to one of the classical rational approximation problems that Zolotarev [25] solved explicitly in 1877 in terms of elliptic functions. For details on Zolotarev functions see for example [1, 22].

A recent preprint [10] also uses Zolotarev functions for computing partial eigenvaules. Here we are concerned with computing the whole eigendecomposition and the SVD.

To get an idea of the resulting rational functions, below we plot the functions on for varying degrees. Observe how fast the functions visibly converge to the sign function as the degree increases.

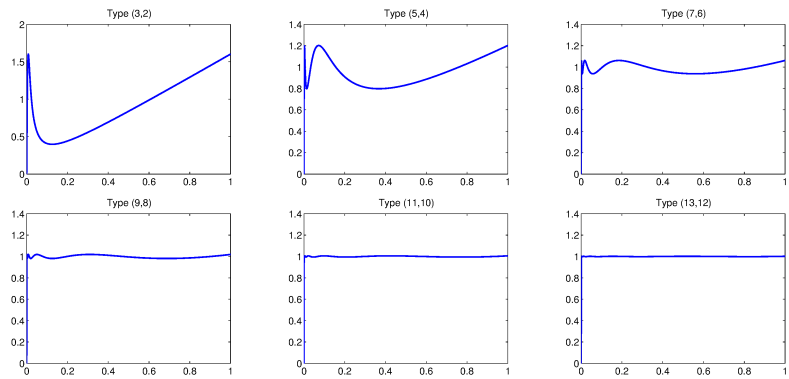

Figure 1: Zolotarev's best rational approximants to the sign function on $[\ell, 1]$ of type $(2r+1, 2r)$ for $r = 1, 2, \ldots, 6$ and $\ell = 10^{-3}$.

There is another remarkable fact about Zolotarev functions: we can compose them to obtain another Zolotarev functions, of much higher degree. For details see [20].

Figure 2 illustrates this fact, in whcih two Zolotarev functions of type $(3, 2)$ are shown in blue and green, and its composition is shown in red. Observe how close the red function (which is a rational function of type $(9, 8)$) is an excellent approximation to the sign function.

Indeed, Zolotarev functions are so powerful that if the degrees are appropriately chosen to be of type $(2r + 1, 2r)$ where $r \in [1, 8]$, then just two iterations would suffice to obtain numerical convergence (i.e. to $\mathcal{O}(u)$) in double precision arithmetic. This runs counter to intuition, since it is known [8] that Abel's impossibility theorem implies that the exact polar decomposition cannot be computed in a finite number of arithmetic operations, an approximation correct to $\mathcal{O}(10^{-16})$ can be obtained.

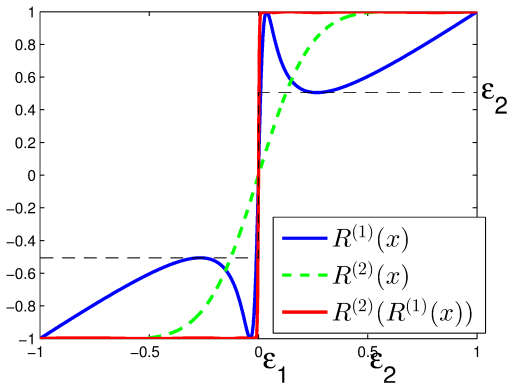

Figure 2: Zolotarev function and its composition.

We call the resulting algorithm for the polar decomposition Zolo-pd. A mind-boggling aspect of Zolo-pd is its rate of convergence. A typical numerical algorithm converges quadratically, that is, with convergence rate 2. Zolo-pd, on the other hand, converges with rate seventeen. Proving this is not difficult, see [20].

Similarly, we call the resulting algorithms for symmetric eigendecomposition and the SVD Zolo-eig and Zolo-SVD, respectively.

4.1 Evaluating a Zolotarev function at matrix arguments

To apply the Zolotarev functions to the singular values of A, we need an efficient and stable way of computing $R(X) := UR(\Sigma_k;\ell)V^*$, where $X_k = U\Sigma_k V^*$ is the SVD. For stability and communication efficiency, we look for an inverse-free implementation, that is, one that does not explicitly invert matrices or require solutions to linear systems; this was the original motivation for the QDWH iteration [17].

Crucial to this task is the following result, which was given in [24], [12, p. 219], and was also used in the QDWH iteration (which is Zolo-pd for the special case $r = 1$).

Lemma 4.1 *Let* $\begin{bmatrix} \eta X \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$ *be the QR decomposition of* $\begin{bmatrix} \eta X \\ I \end{bmatrix}$, *where* $X, Q_1 \in \mathbb{C}^{m\times n}$ *and* $Q_2, R \in \mathbb{C}^{n\times n}$. *Then*

$$Q_1 Q_2^* = \eta X (I + \eta^2 X^* X)^{-1}. \tag{6}$$

We now show that by using the partial fraction representation of Zolotarev functions we can use this lemma for the stable computation of $R(X)$ for any $r \geq 1$.

In general, a partial fraction representation expresses a given rational function in terms of a sum of fractions involving polynomials of low degree. For the Zolotarev function we

obtain the following partial fraction decomposition representation:

$$\frac{\prod_{j=1}^{r}(x^2+c_{2j})}{\prod_{j=1}^{r}(x^2+c_{2j-1})} = 1 - \sum_{j=1}^{r} \frac{a_j}{x^2+c_{2j-1}}, \tag{7}$$

where

$$a_j = \left(\prod_{k=1}^{r}(c_{2j-1}-c_{2k})\right) \cdot \left(\prod_{k=1,k\neq j}^{r}(c_{2j-1}-c_{2k-1})\right). \tag{8}$$

Equation (8) provides a simple and stable way to compute the coefficients a_j in (7), and an easy way to verify (8) is to multiply (7) by x^2+c_{2j-1} and take $x = ic_{2j-1}$.

In matrix form, our task is to compute

$$R(X) = X\prod_{j=1}^{r} P_j(X^*X)\prod_{j=1}^{r}(Q_j(X^*X))^{-1},$$

where $P_j(X^*X) = X^*X + c_{2j}I$ and $Q_j(X^*X) = X^*X + c_{2j-1}I$. Using (7), we see that $R(X)$ can be obtained by computing

$$R(X) = X + \sum_{j=1}^{r} a_j X(X^*X + c_{2j-1}I)^{-1}, \tag{9}$$

which, by Lemma 4.1, is equivalent to

$$\begin{cases} \begin{bmatrix} X \\ \sqrt{c_{2j-1}}I \end{bmatrix} = \begin{bmatrix} Q_{j1} \\ Q_{j2} \end{bmatrix} R_j, \\ R(X) = X + \sum_{j=1}^{r} \frac{a_j}{\sqrt{c_{2j-1}}} Q_{j1}Q_{j2}^*. \end{cases} \tag{10}$$

Note that the r QR factorizations and matrix multiplications $Q_{j1}Q_{j2}^*$ in (10) are completely independent of each other, therefore we can easily compute $Q_{j1}Q_{j2}^*$ in a parallel fashion for $j = 1, \ldots, r$ and compute $R(X)$ simply by adding up the matrices. Furthermore, numerically the evaluation (10) based on partial fractions is much more accurate than a direct evaluation.

We note that the use of partial fractions for Padé-type matrix iterations was employed by Kenney and Laub in [16] for the matrix sign function, and by Higham and Papadimitriou [13] for the polar decomposition. For the action of a matrix function on a vector it was used for example in [7, 23].

4.2 Faster iterations via Cholesky factorization

The QDWH iteration (4) is mathematically equivalent to (3), which can be implemented according to

$$Z_k = I + c_k X_k^* X_k, \quad W_k = \mathrm{chol}(Z_k), \tag{11a}$$

$$X_{k+1} = \frac{b_k}{c_k} X_k + \left(a_k - \frac{b_k}{c_k}\right)(X_k W^{-1})W^{-*}, \tag{11b}$$

where $\mathrm{chol}(Z_k)$ denotes the Cholesky factor of Z_k. Note that (11) involves $m \times n$ matrices, in contrast to (4), which contains an $(m+n)\times n$ matrix. The arithmetic cost of (11) is mn^2 flops for forming the symmetric positive definite matrix Z_k, $n^3/3$ flops for computing its Cholesky factorization, and $2mn^2$ flops for two multiple right-hand side triangular substitutions. Therefore this implementation requires $3mn^2 + n^3/3$ flops, which is cheaper than evaluating (4) by Householder QR factorization and exploiting the structure of $\left[\begin{smallmatrix}\sqrt{c_k}X_k \\ I\end{smallmatrix}\right]$, which needs $5mn^2$ flops. Furthermore, the Cholesky decomposition and triangular substitution both have a known arithmetic and communication-minimizing implementation. Thus (11) is expected to be faster than (4). The same is true of Zolo-pd.

The numerical stability of (11), however, is compromised if Z_k is ill conditioned, as standard error analysis shows that X_{k+1} has errors of order $\kappa_2(Z)\epsilon$ [11]. The implementation (4) is also subject to errors, arising from a QR factorization of the matrix $\left[\begin{smallmatrix}\sqrt{c_k}X_k \\ I\end{smallmatrix}\right]$, but this matrix has 2-norm condition number equal to the square root of that of Z_k, and in any case numerical stability of the polar decomposition computed by (4) is independent of $\kappa_2(Z_k)$, as shown in [18].

To investigate (11) further, we note that $\kappa_2(Z) \le 1 + c_k\|X_k\|_2^2$ and $\|X_k\|_2 \le 1$ (provided that $\alpha \ge \|A\|_2$), and moreover b_k/c_k and $a_k - b_k/c_k$ are both of order 1 for all k [18]. Hence X_{k+1} is computed from (11) with forward error bounded by $c_k\epsilon$. Fortunately, c_k converges to 3 and the convergence is fast enough so that $c_k \le 100$ for $k \ge 2$ for any practical choice $\ell_0 > 10^{-16}$. In QDWH we switch from (4) to (11) once c_k is smaller than 100, which improves the speed and also in practice slightly improves the stability. In particular, if $\ell_0 > 10^{-5}$ then we have $c_k \le 100$ for $k \ge 1$ (for a given k, c_k is a decreasing function of ℓ_0), so we need just one iteration of (4).

For Zolo-pd, on the other hand, checking c_k is not necessary because we know a priori that in the second iteration we have $c_i \le 100$. In other words, no matter what the situation is, in the second iteration we invoke the faster implementation via Cholesky. Moreover, when the input matrix was well-conditioned we can invoke Cholesky right from the start.

We note that there is an alternative, straightforward implementation of (3) which uses matrix inversion or solution to linear systems with multiple right-hand side. The resulting algorithm DWH is unfortunately numerically unstable, as observed in [17], and this instability carries over to Zolo-pd if implemented this way.

4.3 Cost comparison

Here we compare the computational cost of our Zolotarev-based algorithms with the standard algorithms.

Polar decomposition Table 1 compares Zolo-pd with QDWH and the scaled Newton iteration [12, Ch. 8], the two most practical algorithms for the unitary polar factor of $A \in \mathbb{C}^{m\times n}$ (we need $m = n$ for the scaled Newton iteration as it is applicable only to nonsingular matrices). It summarizes the backward stability, the dominant type of operation, the maximum iteration count and arithmetic cost in flops required for $\kappa_2(A) \leq 10^{16}$; for well-conditioned matrices the flop count generally decreases. The arithmetic cost for QDWH is taken from the flop count in [19], and we can also derive that of Zolo-pd similarly. The parenthesized entry at the bottom of the table shows the arithmetic cost along the critical path when the r QR and Cholesky factorizations are computed in parallel. The arithmetic cost of the scaled Newton iteration assumes that matrix inverses are computed in the standard way based on LU factorization with partial pivoting.

Zolo-pd requires more arithmetic cost than QDWH and scaled Newton, by about a factor 3. However, along the critical path it requires the fewest flops, so in a parallel implementation we expect Zolo-pd to be the fastest.

Table 1: Comparison of algorithms for the polar decomposition.

	Zolo-pd	QDWH	scaled Newton
Backward stability	$(\surd)$	$\surd$	$\surd^a$
Dominant operation	QR	QR	inversion
Max. # iterations	2	6	9
Arithmetic cost	$64mn^2 + \frac{8}{3}n^3$ $(8mn^2 + \frac{1}{3}n^3)$	$22mn^2 + \frac{4}{3}n^3$	$18n^3$

[a]Matrix inverses need to be computed in a mixed backward-forward stable manner to prove backward stability of scaled Newton [6, 18]. Using the standard method of LU with partial pivoting for inversion this condition is not guaranteed, and it can be indeed unstable [18]. The parenthesized $(\surd)$ means the stability is observed numerically but not yet established theoretically.

Symmetric eigendecomposition Table 2 compares the spectra-divide-and-conquer algorithms Zolo-eig QDWH-eig, IRS (implicit repeated squaring [3]), and ZZY [24], along with the standard algorithm that performs tridiagonalization followed by the symmetric tridiagonal QR algorithm [21, Ch. 8]. The algorithms compute both the eigenvalues and eigenvectors. In addition to the information shown in Table 1, Table 2 shows whether the algorithm minimizes communication in the asymptotic sense.

Following [19], the arithmetic cost of QDWH-eig and Zolo-eig is obtained assuming that the splitting points σ are chosen such that $\kappa_2(A - \sigma I) \leq 10^5$, for which $r = 5$ is

Table 2: Comparison of algorithms for symmetric eigendecomposition.

	Zolo-eig	QDWH-eig	IRS [3]	ZZY [24]	standard [9, § 8.3]
Min. communication?	$\surd$	$\surd$	$\surd$	$\times$	$\times$
Backward stability	$(\surd)$	$\surd$	conditional	$(\surd)$	$\surd$
Max. # iterations	2	6	53	53	
Arithmetic cost	$39.6n^3$ $(10.25n^3)$	$27n^3$	$\approx 720n^3$	$\approx 370n^3$	$9n^3$

sufficient. We can take a different σ if this does not hold. Note that the arithmetic cost along the critical path of Zolo-eig in a parallel implementation is nearly the same as that of the standard algorithm.

SVD Table 3 compares four SVD algorithms: Zolo-SVD, QDWH-SVD, IRS, and the standard algorithm that performs bidiagonalization followed by bidiagonal QR. The arithmetic cost shows the flop counts for a square $n \times n$ matrix A, and since for Zolo-SVD and QDWH-SVD it depends on the condition number $\kappa_2(A)$, we show the arithmetic cost in the range $\kappa_2(A) = [1.1, 10^{16}]$.

Table 3: Comparison of algorithms for the SVD.

	Zolo-SVD	QDWH-SVD	IRS	standard [9, § 8.6]
Min. communication?	$\surd$	$\surd$	$\surd$	$\times$
Backward stability	($\surd$)	$\surd$	conditional	$\surd$
Max. # iterations	2	6	53	
Arithmetic cost ($m = n$)	$48n^3$–$111n^3$ ($18.5n^3$–$23.5n^3$)	$35n^3$–$52n^3$	$\approx 5700n^3$	$26n^3$

The arithmetic cost along the critical path of Zolo-SVD is slightly smaller than that of the standard algorithm.

In MATLAB experiments, we observe that the sequential runtime of Zolo-eig and Zolo-SVD is slower than the QDWH-based counterparts, and much slower than standard algorithms (or MATLAB's built-in functions `eig`, `svd`). However, along the critical path the runtime is significantly shorter, even ourperforming standard algorithms. This indicates the potential speed when Zolo-based algorithms are implemented in a proper parallel fashion. For numerical experiments and more details we refer to [20]

4.4 Backward stability

As mentioned in the introduction, the backward stability of Zolo-eig and Zolo-SVD rests on that of Zolo-pd for the polar decomposition. For iterations for the polar decomposition, Nakatsukasa and Higham [18] perform stability analysis, who show that the computed polar decomposition is backward stable if two conditions are satisfied: (i) Each iterate is backward-forward stable, that is, the computed approximant $\widehat{Y}$ to $Y = f(X)$ satisfies $\widehat{Y} = f(\widetilde{X}) + \epsilon\|\widehat{Y}\|_2$ where $\widetilde{X} = \widehat{X} + \epsilon\|\widehat{X}\|_2$, where ϵ denotes a matrix whose norm is $\mathcal{O}(u)$. (ii) The function $f(x)$ lies above $y = x$.

Of the two conditions, the second is more nonintuitive and indeed gives insights into some unstable iterations proposed in the literature. It is shown in [18] that QDWH satisfies the two conditions and hence is backward stable if pivoting (row and column) is used for computing the QR factorizations.

For Zolo-pd, it is easy to verify that the second condition is satisfied, because the mapping function $f(x)$ is the best approximant to the sign function, and $y = x$ can be regarded as a member of $(2r + 1, 2r)$ rational functions. However, regarding the first condition, the presence of r QR factorizations seems to make the discussion nontrivial.

Although experiments demonstrate the excellent backward stability of Zolo-pd, its proof therefore remains an open problem.

5 Ongoing work and open problems

As mentioned above, one important open problem is to prove the backward stability of the Zolotarev-based polar decomposition algorithm, which experiments strongly suggest in the affirmative. To do so one would need to extend the discussion in [18] so that the algorithm fits in the framework of the analysis.

The key idea of spectral divide-and-conquer methods is dividing the spectrum into two groups and mapping one to a single value (e.g. 1) and the other to another value (e.g. -1). We have focused on the symmetric eigenvalue decomposition and the SVD, both of which enjoy the numerically attractive property that the decompositions involve only orthogonal and diagonal matrices.

However, the essense of spectral divide-and-conquer is generalizable to other settings, such as nonsymmetric eigenvalue problems and generalized eigenvalue problems. For these problems spectral divide-and-conquer may have even pronounced benefits, as the standard QR and QZ algorithm for solving nonsymmetric (generalized) eigenvalue problems typically perform much worse in speed compared with what the arithmetic cost predicts. These, along with extension to more general problems such as polynomial eigenvalue problems, are among the ongoing projects.

References

[1] N. I. Akhiezer. *Elements of the Theory of Elliptic Functions*, volume 79 of *Translations of Mathematical Monographs*. American Mathematical Society, 1990.

[2] Z. Bai, J. Demmel, and M. Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numer. Math.*, 76(3):279–308, 1997.

[3] G. Ballard, J. Demmel, and I. Dumitriu. Minimizing communication for eigenproblems and the singular value decomposition. Technical Report 237, LAPACK Working Note, 2010.

[4] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in linear algebra. Technical Report 218, LAPACK Working Note, 2009.

[5] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in numerical linear algebra. *SIAM J. Matrix Anal. Appl.*, 32(3):866–901, 2011.

[6] R. Byers and H. Xu. A new scaling for Newton's iteration for the polar decomposition and its backward stability. *SIAM J. Matrix Anal. Appl.*, 30:822–843, 2008.

[7] V. Druskin, S. Güttel, and L. Knizhnerman. Near-optimal perfectly matched layers for indefinite helmholtz problems. MIMS EPrint, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2013.

[8] A. George and K. Ikramov. Is the polar decomposition finitely computable? *SIAM Journal on Matrix Analysis and Applications*, 17:348, 1996.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 4th edition, 2012.

[10] S. Güttel, E. Polizzi, P. Tang, and G. Viaud. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. MIMS EPrint 2014.39, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2014.

[11] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, USA, second edition, 2002.

[12] N. J. Higham. *Functions of Matrices: Theory and Computation.* SIAM, Philadelphia, PA, USA, 2008.

[13] N. J. Higham and P. Papadimitriou. A parallel algorithm for computing the polar decomposition. *Parallel Computing*, (20):1161–1173, 1994.

[14] N. J. Higham and P. Papadimitriou. A new parallel algorithm for computing the singular value decomposition. *In Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 80–84, 1994.

[15] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, second edition, 2012.

[16] C. Kenney and A. Laub. A hyperbolic tangent identity and the geometry of padé sign function iterations. *Numerical Algorithms*, 7:111–128, 1994. 10.1007/BF02140677.

[17] Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM J. Matrix Anal. Appl.*, 31(5):2700–2720, 2010.

[18] Y. Nakatsukasa and N. J. Higham. Backward stability of iterations for computing the polar decomposition. *SIAM J. Matrix Anal. Appl.*, 33(2):460–479, 2012.

[19] Y. Nakatsukasa and N. J. Higham. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM J. Sci. Comp*, 35(3):A1325–A1349, 2013.

[20] Y. Nakatsukasa and R. W. Freund. Using zolotarev's rational approximation for computing the polar, symmetric eigenvalue, and singular value decompositions. manuscript.

[21] B. N. Parlett. *The Symmetric Eigenvalue Problem.* SIAM, Philadelphia, 1998.

[22] P. P. Petrushev and V. A. Popov. *Rational Approximation of Real Functions*, volume 28. Cambridge University Press, 2011.

[23] J. van den Eshof, T. Lippert, A. Frommer, K. Schilling, and H. van der Vorst. Numerical methods for the QCD overlap operator: I. sign-function and error bounds. *Comput. Phys. Comm.*, 146:203–224, 2002.

[24] Z. Zhang, H. Zha, and W. Ying. Fast parallelizable methods for computing invariant subspaces of Hermitian matrices. *Journal of Computational Mathematics*, 25(5):583–594, 2007.

[25] E. I. Zolotarev. Application of elliptic functions to the questions of functions deviating least and most from zero. *Zap. Imp. Akad. Nauk. St. Petersburg,*, 30(5), 1877. In Russian.

Department of Mathematical Informatics
University of Tokyo
Tokyo 113-8656
JAPAN
E-mail address: nakatsukasa@mist.i.u-tokyo.ac.jp