SIAM J. SCI. COMPUT.
Vol. 23, No. 6, pp. 2165–2182
© 2002 Society for Industrial and Applied Mathematics

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

A BLOCK ORTHOGONALIZATION PROCEDURE WITH CONSTANT SYNCHRONIZATION REQUIREMENTS*

ANDREAS STATHOPOULOS† AND KESHENG WU‡

Abstract. First, we consider the problem of orthonormalizing skinny (long) matrices. We propose an alternative orthonormalization method that computes the orthonormal basis from the right singular vectors of a matrix. Its advantages are that (a) all operations are matrix-matrix multiplications and thus cache efficient, (b) only one synchronization point is required in parallel implementations, and (c) it is typically more stable than classical Gram–Schmidt (GS). Second, we consider the problem of orthonormalizing a block of vectors against a previously orthonormal set of vectors and among itself. We solve this problem by alternating iteratively between a phase of GS and a phase of the new method. We provide error analysis and use it to derive bounds on how accurately the two successive orthonormalization phases should be performed to minimize total work performed. Our experiments confirm the favorable numerical behavior of the new method and its effectiveness on modern parallel computers.

Key words. Gram–Schmidt, orthogonalization, Householder, QR factorization, singular value decomposition, Poincaré

AMS subject classification. 65F15

PII. S1064827500370883

1. Introduction.

Computing an orthonormal basis from a given set of vectors is a basic computation, common to most scientific applications. Often, it is also one of the most computationally demanding procedures because the vectors are of large dimension, and because the computation scales as the square of the number of vectors involved. Further, among several orthonormalization techniques the ones that ensure high accuracy are the more expensive ones.

Skinny (or long) matrices, whose row dimension far exceeds their column dimension, arise naturally in various scientific contexts. Examples include statistical analysis, where there are many more observations than variables, and iterative methods that use a small subspace of vectors to span the required solutions. For a variety of reasons, an orthonormal basis of these vectors must be computed. Traditionally, this is obtained through the QR factorization, even though quite often the matrix R is not of primary interest, but rather the orthonormal basis Q. The QR factorization is computed through Householder transformations (we call this method simply QR) or through classical Gram–Schmidt (GS) or modified Gram–Schmidt (MGS). Although less stable numerically, GS with reorthogonalization is usually preferred to the QR method because of better computational properties.

Yet, all of these methods have performance limitations on modern, cache based processors and parallel computers. Their implementations are based on level 1 or level 2 BLAS operations [8, 9, 15], which have low cache reuse. Level 3 BLAS implementations are possible, but they are not suitable for skinny matrices. On parallel

*Received by the editors April 17, 2000; accepted for publication (in revised form) December 4, 2001; published electronically May 15, 2002. This research was supported by NERSC and performed using computational facilities at NERSC and the College of William and Mary. The latter facility was supported by grants from the National Science Foundation (EIA-9977030) and Sun Microsystems (SAR EDU00-03-793).

http://www.siam.org/journals/sisc/23-6/37088.html

†Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 (andreas@cs.wm.edu).

‡NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (kwu@lbl.gov).

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

platforms, such as the increasingly popular clusters of workstations, reduction in communication and synchronization overheads has not kept up with the explosive growth of network bandwidth and processor speed [21]. As a result, the global synchronization required by frequent inner products does not scale with the number of processors. A method is still needed that operates only on blocks of vectors and requires a number of synchronizations that is independent of the size of the matrix.

In many applications, orthonormalization occurs in an incremental fashion, where a new set of vectors (we call this *internal set*) is orthogonalized against a previously orthonormal set of vectors (we call this *external*), and then among themselves. This computation is typical in block Krylov methods, where the Krylov basis is expanded by a block of vectors [11, 12]. It is also typical when certain external orthogonalization constraints have to be applied to the vectors of an iterative method. Locking of converged eigenvectors in eigenvalue iterative methods is such an example [19, 22]. The nature of these applications suggests that the internal set is usually a skinny matrix with fewer vectors than the external set.

Conceptually, this problem can be viewed as an update of a QR factorization that has already produced the orthonormal set of external vectors which should not be modified. Computationally, however, the problem is usually tackled as a two phase process; first, orthogonalizing the internal vectors against the external ones (external phase), and second, orthogonalizing the internal vectors among themselves (internal phase). For the external phase, block GS and MGS are the most competitive choices, while for the internal phase an efficient orthonormalization procedure for skinny matrices is needed.

Most of the previous efforts to address the above two problems considered blocks of vectors, and used hybrids of the more scalable GS across blocks, and the more accurate MGS within blocks [2, 14]. Performance improves, but the number of synchronization points is still linear to the number of vectors, and BLAS level 2 kernels are still dominant despite blocking. Interestingly, such efforts have focused on a full QR factorization of a set of vectors, rather than on the two phase problem.

In this paper, we introduce a method based on the singular value decomposition (SVD) that uses the right singular vectors to produce an orthonormal basis for a given skinny matrix. The idea itself is not new, dating back at least to Poincaré, and it is sometimes encountered in chemistry and wavelet literature [5, 6, 16, 17, 20]. However, it has not received any attention as a computationally viable orthogonalization method, and to our knowledge there is no analysis of its numerical properties. The method, which we call SVQB, uses exclusively level 3 BLAS kernels, and it has a constant number of synchronization points. We show that it is not as accurate numerically as MGS, but it is better than GS in the absence of special sparsity structure. More interestingly, we show that more stable alternatives, such as MGS or Householder, are an overkill for our two phase problem. Coupling the SVQB method for the internal phase with a block GS with reorthogonalization for the external phase results in a method also with constant synchronization requirements.

The paper is organized as follows. First we describe the SVQB method for skinny matrices, we analyze its numerical stability, and we confirm our theory through numerical experiments and comparisons with other methods. Second, we couple SVQB with a block GS for the two phase problem. We analyze the numerical interaction between the two methods, and based on this theory we tune the two phases to avoid unnecessary reorthogonalizations. Following this, we present timings from a series of experiments on the Cray T3E, IBM SP-2, and on a cluster of SUN workstations.

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

These verify that the block computations and the small number of synchronizations help the new method achieve accurate orthogonality faster than other competitive methods.

2. The problem(s). Let $V \in \Re^{n\times k}$ be a set of orthonormal vectors, and $W \in \Re^{n\times m}$ be a set of vectors, where $k + m \leq n$. In practice, we expect $m < k$ and $m \ll n$. When $m \ll n$, W is often referred to as a skinny matrix. The first problem we consider is that of obtaining an orthonormal set of vectors Q such that span(Q)=span(W). The second problem is again to obtain an orthonormal Q such that span($[V\ W]$)=span($[V\ Q]$) and $Q \perp V$. In both problems Q can be any orthonormal basis, not necessarily from a QR factorization. In finite precision, the equality of the spans can be relaxed, but the orthonormality requirement must remain.

The distinction between the two problems is made for computational reasons. First, orthonormalizing skinny matrices is a problem important in itself, which allows for efficient solutions without a QR factorization. In the presence of an external matrix V, methods like the classical GS would orthogonalize each vector of W against all previous orthogonal vectors in both W and V. In that case, the difference between the two phases is blurred. However, such algorithms allow only for level 2 BLAS computational kernels and introduce at least $O(m)$ number of synchronization points on parallel computers. To improve computational performance we need to consider W as a block (or subblocks within W). A block GS method would orthogonalize the block against V (and other previous subblocks in W). In this case, the orthogonality among the vectors within the block must be resolved at a different time in a distinct internal phase. Finally, because V cannot be modified, distinguishing between the problems allows non-QR factorization methods to be used for the internal phase.

For the internal phase, GS and MGS are popular QR factorization methods which both incur the same number of arithmetic operations, they are based on level 2 BLAS kernels, and they can be implemented in parallel with a modest number of $m + 1$ synchronization points [11, 24]. MGS is more numerically stable with the error in the orthogonality of Q bounded by $\epsilon\kappa(W)$, where $\kappa(W)$ is the condition number of W [1, 3]. Householder reflections yield a matrix Q which is orthogonal to machine precision (ϵ) but require twice the arithmetic of (M)GS. In practice, for most matrices, a second orthogonalization with GS is typically enough for producing orthogonality to machine precision [7], and thus GS is often preferred over other methods.

In the context of the two phase problem, producing an internal set of vectors Q with orthogonality close to machine precision is unnecessary because of the interdependence of the phases. For example, external orthogonalization against V may spoil the internal orthogonality of W, and vice versa. Therefore, this two phase problem obviates the use of expensive but stable methods such as Householder.

3. The SVQB method. An especially interesting orthonormal basis of the span(W) is the one derived from the right singular vectors of W. Assume that the vectors in W are normalized. The singular values of W are the square roots of the eigenvalues of $S = W^TW$, and the right singular vectors are the corresponding eigenvectors of S. Let $SU = U\Lambda$ be the eigendecomposition of S and define $Q = WU\Lambda^{-1/2}$. Obviously, span(Q) = span(W) and $Q^TQ = I$. If the W vectors are not normalized, the diagonal of S, $D = \text{diag}(S)$, contains the squares of their norms ($S_{ii} = W_i^TW_i$). Therefore, we can implicitly work with the normalized $WD^{-1/2}$ by scaling the columns and rows of S. This is inexpensive and is as numerically stable as explicit normalization. The resulting factorization is not a QR but rather a "QB"

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

factorization, where Q is orthonormal and B a full matrix. In exact arithmetic, the algorithm for this singular vector QB factorization (which we call SVQB) follows.

ALGORITHM 3.1. $Q = \text{SVQB}(W)$.

1. $S' = W^T W$.
2. *Scale* $S = D^{-1/2} S' D^{-1/2}$, *with* $D = diag(S')$.
3. *Solve* $SU = U\Lambda$, *for all eigenpairs.*
4. *Compute* $Q = W D^{-1/2} U \Lambda^{-1/2}$.

When some of the vectors in W are linearly dependent, one or more of the eigenvalues and their corresponding vectors in Q are zero. In finite precision, a similar effect is caused by almost linearly dependent vectors and eigenvalues close to zero. Because of numerical noise, such eigenvalues cannot be bounded away from zero. To prevent normalization overflows, and to avoid an explicit computation of the norm of the vectors Q, we set a minimum threshold for eigenvalues. If ϵ is the machine precision, we insert the following two steps:

3.1. $\tau = \epsilon \ \max_i(\Lambda_{ii})$.

3.2. If $\Lambda_{ii} < \tau$, set $\Lambda_{ii} = \tau$ for all i.

Other strategies for dealing with linear dependencies are also possible. For example, we could consider only those eigenvectors with eigenvalues greater than some "safe" threshold. The resulting basis is then smaller, but it is guaranteed to be orthonormal and to numerically span a subspace of the original vectors. Finally, because of finite precision arithmetic, the algorithm may have to be applied iteratively ($Q^{(i+1)} = \text{SVQB}(Q^{(i)})$) until an orthonormal set Q is obtained.

The solution of the eigenvalue problem and the implicit normalization involve only $m \times m$ matrices (S and U), and thus they are inexpensive. On parallel computers these can be duplicated on each processor. The matrix-matrix multiplication for computing S and the multiplication of W with U each contribute $2nm^2$ floating point operations, which makes the algorithm twice as expensive as GS. However, these operations are level 3 BLAS kernels and can be performed efficiently on cache based computers. Alternatively, the matrix multiplication for computing the symmetric S can be performed with half the operations, but level 2 BLAS kernels will have to be used. Moreover, a parallel implementation of the SVQB method requires only one synchronization point when computing the matrix S.

We note that our interest in the singular vectors stems only from the computational efficiency of SVQB. A standard bidiagonalization kernel for computing the SVD of W directly could offer better numerical stability [4]. However, its cache utilization and parallel efficiency is similar to that of QR (with Householder transformations), without yielding the same level of machine precision orthogonality. We have not considered such methods, as they offer no advantages over the QR method.

3.1. The Cholesky QR method. Similarly to the SVQB method, we can derive a block QR factorization based on the Cholesky factorization. Note that if $S = W^T W = R^T R$, where R is the Cholesky factor, $Q = WR^{-1}$ defines the QR factorization for W [11]. Although this method (denoted as CholQR) is rarely used computationally, it has some attractive characteristics: it is a QR factorization; it is based on a level 3 BLAS kernel and a triangular system solution; it involves only 50% more arithmetic than GS; and it also requires one synchronization point in parallel implementations. Researchers have noticed that it is not as stable as MGS, but it is often more stable than GS [2, 10]. One of the problematic issues with CholQR is that the more ill conditioned S is, the less stable the Cholesky factorization becomes. Regularizing it effectively is not as straightforward as in the case of SVQB, where the

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

smallest singular pairs can simply be left out of the computation. Finally, the cache performance of CholQR is usually inferior to that of SVQB because of the triangular solve.

4. Stability analysis of SVQB. For many applications, such as block Krylov iterative methods, the orthonormality of the resulting vectors Q, not the upper triangular R of the QR factorization, is of importance. For example, Krylov methods will still make progress, albeit a slower one, if provided with a slightly different orthonormal set Q. Thus, as it is common in the literature, we measure stability as the departure of the resulting Q from orthonormality rather than its backward error. Because of its block, nonsequential nature, we expect the SVQB procedure to be less stable than MGS. However, as we show below, it is often more stable than GS.

THEOREM 4.1. *Let W be a set of m linearly independent vectors of $\Re^n$. Let $\bar{Q}$ be the floating point representation of the matrix computed by applying the SVQB procedure on W. If $\kappa(W)$ is the condition number of W, then*

$$\|I - \bar{Q}^T\bar{Q}\| \;\leq\; c_0 \;\min\left(\,\epsilon\;\kappa(W)^2,\;1\right),$$

where $\|.\|$ denotes the 2-norm, ϵ is the machine round off, and c_0 is a constant depending on n and m.

Proof. Let $S = W^TW$, and let $\tilde{S}$ be the floating point representation of S. Then

$$\tilde{S} = S + \delta S, \;\text{ with }\; \|\delta S\| \leq c_1\;\epsilon\;\|S\|. \tag{4.1}$$

We can further write this as $\|\delta S\| \leq c_1\;\epsilon\;\|W\|^2$. The effect of performing scaling on the matrix S corresponds to each vector in W having norm 1 implicitly, and, in that case, $\|\delta S\| \leq c_1\;\epsilon\;m$.

Let $\bar{U}$, with $\bar{U}^T\bar{U} = I$, and $\bar{\Lambda}$ be the computed eigenvectors and eigenvalues of the small $m \times m$ symmetric matrix $\tilde{S}$. From standard backward error analysis, these can be considered an exact eigendecomposition of a nearby matrix $\bar{S} = \bar{U}\bar{\Lambda}\bar{U}^T$. Using relation (4.1) we can express the error in $\bar{S}$ as

$$\bar{S} = \tilde{S} + \delta\tilde{S}, \;\text{ with }\; \|\delta\tilde{S}\| \leq c_2\;\epsilon\;\|S\| + O(\epsilon^2). \tag{4.2}$$

From the above, and by letting $c_3 = c_1 + c_2$, the matrices S and $\bar{S}$ are related by

$$\|\bar{S} - S\| = \|\delta\tilde{S} + \delta S\| \leq c_3\;\epsilon\;\|S\|. \tag{4.3}$$

Let $\bar{\lambda}_{\min} = \min_i \bar{\Lambda}_{ii}$, and $\bar{\lambda}_{\max} = \max_i \bar{\Lambda}_{ii}$, and consider a similar notation for eigenvalues of other matrices. Because our algorithm sets eigenvalues $\bar{\lambda}_i$ that are smaller than $\epsilon\bar{\lambda}_{\max}$ equal to this threshold, we define a diagonal matrix $\hat{\Lambda}$ such that

$$\hat{\Lambda}_{ii} = \begin{cases} \bar{\Lambda}_{ii} & \text{if } \bar{\Lambda}_{ii} > \epsilon\bar{\lambda}_{\max}, \\ \epsilon\bar{\lambda}_{\max}, & \text{if } \bar{\Lambda}_{ii} \leq \epsilon\bar{\lambda}_{\max}. \end{cases} \tag{4.4}$$

Let $\bar{Q} = Q + \delta Q = W\bar{U}\hat{\Lambda}^{-1/2} + \delta Q$ be the floating point representation of the matrix returned by the SVQB procedure. Then, $\|\delta Q\| \leq c_4\,\epsilon\,\|W\|\,\|\bar{U}\|\,\|\hat{\Lambda}^{-1/2}\|$. If we denote by λ_i the exact eigenvalues of S, with $\lambda_{\max}$ the largest one, then $\|W\| = \sqrt{\lambda_{\max}}$. Note also that $\bar{U}^T\bar{U} = I$, and thus $\|\bar{U}\| = 1$. From (4.4) we have

$$\|\hat{\Lambda}^{-1}\| \leq\; \min\left(\frac{1}{\bar{\lambda}_{\min}}, \frac{1}{\epsilon\bar{\lambda}_{\max}}\right). \tag{4.5}$$

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Because for symmetric eigenproblems the error in the eigenvalue is bounded by the error in the matrix, for any $\bar{\lambda}_i$ we have $\left|\bar{\lambda}_i - \lambda_i\right| \leq \|\bar{S} - S\| \leq c_3\, \epsilon\, \|S\| = c_3\, \epsilon\, \lambda_{\max}$. Then, there are constants c_5 and c_6, such that

$$\bar{\lambda}_{\min} = \lambda_{\min} + c_5\, \epsilon\, \lambda_{\max} \quad \text{and} \quad \bar{\lambda}_{\max} = \lambda_{\max} + c_6\, \epsilon\, \lambda_{\max}. \tag{4.6}$$

We note that $\kappa(W) = \sqrt{\kappa(S)} = \sqrt{\lambda_{\max}/\lambda_{\min}}$, and, because $\epsilon + c_6\, \epsilon^2 = O(\epsilon)$, a substitution of (4.6) into (4.5) yields

$$\|\hat{\Lambda}^{-1}\| \leq \min\left(\frac{1/\lambda_{\min}}{1 + c_5\, \epsilon\kappa(S)},\ \frac{1}{\epsilon\lambda_{\max}}\right). \tag{4.7}$$

The first term is chosen as the minimum if none of the $\bar{\Lambda}_{ii}$ is in the order of $\epsilon\bar{\lambda}_{\max}$ or smaller. This means that all the singular values of W, $\sqrt{\lambda_i}$, must be larger than $\sqrt{\epsilon}$, because they can be represented by eigenvalues of $\bar{S}$, despite the squaring. Therefore, $\kappa(S) < O(1/\epsilon)$, and $1 + c_5\, \epsilon\kappa(S) = O(1)$. Thus, the bound (4.7) becomes

$$\|\hat{\Lambda}^{-1}\| \leq c_7\, \min\left(\frac{1}{\lambda_{\min}},\ \frac{1}{\epsilon\lambda_{\max}}\right). \tag{4.8}$$

By setting $c_8 = c_4 c_7$, we can give a bound for $\|\delta Q\|$, as well as for $\|Q\| = \|W\bar{U}\hat{\Lambda}^{-1/2}\|$:

$$\|\delta Q\| \leq c_8\, \min\left(\, \epsilon\, \kappa(W),\ \sqrt{\epsilon}\, \right), \tag{4.9}$$

$$\|Q\| \leq c_8\, \min\left(\, \kappa(W),\ 1/\sqrt{\epsilon}\, \right). \tag{4.10}$$

We emphasize that the above is not the backward error for the exact result of SVQB but for the product of computed matrices that yields $\bar{Q}$. The exact backward error is not relevant because the orthogonality of $\bar{Q}$ is of interest.

Let us consider the departure of the computed $\bar{Q} = Q + \delta Q$ from orthonormality:

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &= \|I - \hat{\Lambda}^{-1/2}\bar{U}^T W^T W \bar{U}\hat{\Lambda}^{-1/2} + \delta Q^T Q + Q^T \delta Q\| + O(\delta Q^2) \\ &\leq \|I - \hat{\Lambda}^{-1/2}\bar{U}^T S \bar{U}\hat{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\|. \end{aligned} \tag{4.11}$$

From relations (4.1)–(4.4) and the orthonormality of $\bar{U}$, this becomes

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\leq \|I - \hat{\Lambda}^{-1/2}\bar{U}^T\bar{S}\bar{U}\hat{\Lambda}^{-1/2} + \hat{\Lambda}^{-1/2}\bar{U}^T(\delta\tilde{S} + \delta S)\bar{U}\hat{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\| \\ &\leq \|I - \hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2}\| + \|\hat{\Lambda}^{-1}\|\|\delta\tilde{S} + \delta S\| + 2\|\delta Q\|\|Q\|. \end{aligned} \tag{4.12}$$

From definition (4.4), the first term is zero if $\kappa(S) < O(1/\epsilon)$. Otherwise, it is equal to $\max(1 - \bar{\lambda}_i/\epsilon\bar{\lambda}_{\max},\ \text{for } \bar{\lambda}_i \leq \epsilon\bar{\lambda}_{\max})$. However, this is less than one, and in that case the other terms are also $O(1)$. From bounds (4.3) and (4.8)–(4.10), and by setting $c_0 > c_3 c_7 + 2c_8^2$, we obtain

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\leq c_3 c_7\, \min\left(\, \epsilon\kappa(W)^2,\ 1\right) + 2c_8^2 \min\left(\, \epsilon\kappa(W)^2,\ 1\right) \\ &\leq c_0\, \min\left(\, \epsilon\, \kappa(W)^2,\ 1\right). \end{aligned} \tag{4.13}$$

∎

Next, we bound $|\kappa(\bar{Q}) - 1|$, thus showing that when applying SVQB iteratively, $\bar{Q}$ converges fast to an orthonormal basis. We first state the following lemma (see [13]).

LEMMA 4.2.

1. *If* $\|I - Q^T Q\| \leq \alpha$, *then* $\|Q^T Q\| \leq \|Q\|^2 \leq 1 + \alpha$.
2. *If* $\|I - Q^T Q\| \leq \alpha < 1$, *then* $\|I - (Q^T Q)^{-1}\| \leq \frac{\alpha}{1-\alpha}$.

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

PROPOSITION 4.3. *If* $\|I - Q^TQ\| \leq \alpha < 1$*, then the condition number* $\kappa(Q)$ *satisfies*

$$\kappa(Q) \leq \sqrt{\frac{1+\alpha}{1-\alpha}}.$$

Proof. By definition, $\kappa(Q) = \sqrt{\|Q^TQ\|\|(Q^TQ)^{-1}\|}$. The proof follows from Lemma 4.2, since $\|Q^TQ\| \leq \|Q\|^2 \leq 1+\alpha$, and $\|(Q^TQ)^{-1}\| = \|I - I + (Q^TQ)^{-1}\| \leq 1 + \|I - (Q^TQ)^{-1}\| \leq \frac{1}{1-\alpha}$. ∎

THEOREM 4.4. *Let* W *be a set of* m *linearly independent vectors of* $\Re^n$*, with condition number* $\kappa(W)$*. Let* $\bar{Q}$ *be the floating point representation of the matrix computed by applying the SVQB procedure on* W*. If* ϵ *is the machine round off, then*

$$\text{if } \sqrt{\epsilon}\ \kappa(W) < c < 1, \qquad \kappa(\bar{Q}) \leq 1 + O(\epsilon\ \kappa(W)^2)).$$

Proof. If we let $\alpha = O(\epsilon\ \kappa(W)^2) < c^2 < c < 1$, according to Theorem 4.1, $\|I - \bar{Q}^T\bar{Q}\| < \alpha$, and by using Proposition 4.3, we have $\kappa(\bar{Q}) \leq \sqrt{\frac{1+\alpha}{1-\alpha}} \leq \sqrt{1 + \frac{2\alpha}{1-\alpha}} \leq 1 + \frac{\alpha}{1-\alpha}$. This proves the bound because, in this case, α is bounded away from 1. ∎

The case where $\kappa(W) \geq \frac{1}{\sqrt{\epsilon}}$ cannot be bounded in the general case because numerical error dominates. However, some intuition can be gained by considering the structure of $\bar{Q}^T\bar{Q}$ as obtained from (4.12) and the bounds (4.9)–(4.10) and (4.3):

$$\bar{Q}^T\bar{Q} = \hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2} + \hat{\Lambda}^{-1/2}\Delta S\hat{\Lambda}^{-1/2} + \Delta Q,$$

where $\Delta S = O(\epsilon\bar{\lambda}_{\max})$ and $\Delta Q = O(1)$ element-wise. Note that after scaling, $\hat{\Lambda}^{-1/2}\Delta S\hat{\Lambda}^{-1/2}$ also becomes element-wise $O(1)$. From the definition of $\hat{\Lambda}$, the entries of the diagonal matrix $\hat{\Lambda}^{-1/2}\bar{\Lambda}\hat{\Lambda}^{-1/2}$ are 1 for all eigenvalues above the $\epsilon\bar{\lambda}_{\max}$ threshold, and $\bar{\lambda}_i/(\epsilon\bar{\lambda}_{\max}) = 1/(\epsilon\kappa(\bar{S}))$ for the rest. Thus, the eigenvalues of $\bar{Q}^T\bar{Q}$ are O(1) perturbations of the these diagonal values, so the smallest eigenvalue cannot be bounded away from zero. If we assume that the only finite precision errors occur from the inability to represent eigenvalues of $\bar{S}$ smaller than $\epsilon\lambda_{\max}$, i.e., $\Delta S = 0$ and $\Delta Q = 0$, then, obviously, $\kappa(\bar{Q}^T\bar{Q}) = \epsilon\kappa(\bar{S}) < \epsilon\kappa(S)$, and thus $\kappa(\bar{Q}) < \sqrt{\epsilon}\kappa(W)$. Note that in this case the vectors of $\bar{Q}$ are exactly orthogonal to each other, yet their condition number is far from 1.

Although $\kappa(\bar{Q}) < \sqrt{\epsilon}\kappa(W)$ cannot be proved in general, we have observed it in all our numerical experiments. A plausible explanation is that $O(1)$ perturbations after vector scaling introduce random noise which we expect to be in linearly independent and relatively well-conditioned directions.

4.1. Convergence comparisons. The above theorems suggest that in most situations, applying SVQB once or twice should produce orthogonal vectors. In the case of extremely ill-conditioned vectors, a third application of the procedure might be necessary. This is akin to the behavior of GS with reorthogonalization [7, 13, 18], but it is expected to be better than iterative GS without internal reorthogonalization.

If an accurate Cholesky decomposition can be computed, the CholQR procedure should be identical to SVQB. In fact, it might be possible to prove bounds for CholQR similar to the ones in the previous section. However, even with an accurate decomposition, for very large condition numbers we expect CholQR to be less stable than the eigenvalue-based SVQB method.

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 4.1
W = Krylov(A, 30), where A is a 2-D Laplacean of size 1089×1089, and an initial vector of all ones. $\kappa(W) = 3e+20$. However, after scaling, because of numerical error, $\kappa(W)$ becomes: $6e+16$.

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(Q)$	$\kappa(Q'_u)$	$\kappa(Q)$	$\kappa(Q)$	$\kappa(Q)$
1	6e+16	1e+09	4e+08	1e+09	2e+16	2e+01
2	4e+08	1e+01	4e+00	6e+01	1e+14	3e-14
3	4e+00	$1+\epsilon$	$1+\epsilon$	1+7e-12	8e+10	$1+\epsilon$
4	–	–	–	$1+\epsilon$	3e+06	–
5	–	–	–	–	1+1e−03	–
6	–	–	–	–	$1+\epsilon$	–

TABLE 4.2
W = Hilbert matrix of size(100). $\kappa(W) = 2e+19$.

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(Q)$	$\kappa(Q'_u)$	$\kappa(Q)$	$\kappa(Q)$	$\kappa(Q)$
1	2e+19	3e+11	9e+10	2e+12	2e+19	7e+02
2	9e+10	2e+03	7e+02	6e+04	4e+16	1+2e-13
3	8e+02	1+5e-11	1+2e-11	1+2e-07	2e+14	$1+\epsilon$
4	1+2e-11	$1+\epsilon$	$1+\epsilon$	$1+\epsilon$	4e+12	–
5	–	–	–	–	4e+10	–
6	–	–	–	–	4e+08	–
7	–	–	–	–	2e+00	–
8	–	–	–	–	$1+\epsilon$	–

TABLE 4.3
*W = [ones(1, 30), diag(rand(30, 1)*eps*eps*eps)].*

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(Q)$	$\kappa(Q'_u)$	$\kappa(Q)$	$\kappa(Q)$	$\kappa(Q)$
1	2e+49	3e+41	4e+34	4e+34	2e+01	$1+\epsilon$
2	4e+34	7e+25	4e+19	4e+19	$1+\epsilon$	–
3	4e+19	3e+11	2e+07	4e+06	–	–
4	2e+07	1+1e−03	1+1e−04	1+4e−03	–	–
5	1+1e−04	$1+\epsilon$	$1+\epsilon$	$1+\epsilon$	–	–

To demonstrate the relative effectiveness of these methods, we apply them on three sets of vectors and report the improvements on their condition numbers. The first set is the 30 Krylov vectors generated from a vector of all ones and the two-dimensional (2-D) Laplacean on a regular, finite difference, square mesh with Neumann conditions. The dimension of the matrix is 1089, and the initial vector is not considered among the set of 30. The second set consists of the columns of the Hilbert matrix of size 100. The third set is rather artificial, and it has been used to show the benefits of MGS over GS [2, 13, 14]. We use the following variation, shown in MATLAB notation: `W = [ ones(1,30); diag( rand(30,1)*eps*eps*eps ) ]` . All tests are run in MATLAB on a SUN Ultra-2 workstation with $\epsilon = 2.2e{-}16$. The condition numbers are computed by the Matlab function `cond` and therefore could be inaccurate whenever they exceed 10^{16}. The results for these three cases are shown in Tables 4.1, 4.2, and 4.3, respectively.

We compare SVQB against CholQR, GS, and MGS by printing the condition number $\kappa(\bar{Q})$ of the vectors that these methods produce after each iteration. Since

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

the implicit normalization in step 4 of SVQB does not guarantee normality for ill-conditioned problems, we also print the condition number $\kappa(Q_u)$ of the unscaled vectors, $Q_u = W\bar{U}$, and the condition number of the same vectors after explicitly scaling them by their norms, $\kappa(Q'_u)$. Note that at any iteration, $\kappa(Q_u)$ is equal to $\kappa(Q'_u)$ of the previous iteration. For example, after the first iteration $\kappa(Q_u) = \kappa(W)$. This verifies that explicit normalization is not needed. If scaling by $\bar{\Lambda}^{-1/2}$ does not produce normal vectors, another iteration must be performed either way, during which the implicit normalization of columns and rows of S in step 2 of SVQB has the same effect as explicit normalization. In addition, explicit normalization would introduce additional work, and, more importantly, an additional synchronization point.

The results in all tables confirm the developed theory. When the condition number is smaller than $1/\sqrt{\epsilon}$, the reduction obeys closely the bound in Theorem 4.4. The application of one step of SVQB reduces the condition number of a set of vectors by at least $\sqrt{\epsilon}$. This reduction is sharp for the examples in Tables 4.1 and 4.2, but if the vectors are explicitly normalized the reduction could be larger (see Table 4.3).

As expected, the CholQR method behaves similarly to SVQB (Table 4.3). In some cases, the orthogonality of CholQR is inferior to that of SVQB (see Table 4.2), and thus it is possible that it takes more iterations to produce a fully orthonormal set (see Table 4.1). This is in spite of the fact that in our implementation, we first compute the lowest eigenvalue of W^TW and shift it so that the Cholesky decomposition is applied on a numerically positive definite matrix.

As discussed earlier, GS without reorthogonalization for each vector is not effective for large condition numbers. In such cases, GS may offer no improvement between successive iterations (see the first GS iteration in Table 4.2), or it may require many iterations to produce a set with a relatively small conditioner number (see Tables 4.1 and 4.2). Once this is achieved, however, one or two further iterations provide a fully orthonormal set. An exception is the example in Table 4.3 for which GS requires only one reorthogonalization. The reason is that GS takes advantage of the sparse structure of the matrix, performing computations only among very small elements, thus achieving low *relative* error.

MGS is clearly more stable than the rest of the methods. However, because the departure from orthogonality for MGS is bounded by $\epsilon\kappa(W)$ [1], even for relatively small $\kappa(W)$, a second MGS is often needed (see Tables 4.1 and 4.2). Note that the $\epsilon\ \kappa(W)^2$ bound for SVQB is virtually identical to that of MGS, if $\kappa(W)$ is close to 1, thus diminishing any advantages over SVQB.

5. The two phase problem. The above theory and examples establish that SVQB is a competitive choice for the internal orthonormalization of the two phase problem. If we denote as $W' = \text{Ortho}(V, W)$ any orthonormalization procedure for the external phase, $W' = (I - VV^T)WD^{-1/2}$, where $D^{1/2}$ is a diagonal matrix with the normalizing norms of the vectors, the two phase algorithm can be described as follows.

ALGORITHM 5.1. $Q = \text{Ortho-SVQB}(V, W)$.

1. $W' = \text{Ortho}(V, W)$.
2. $Q = \text{SVQB}(W')$.

The most common choices for Ortho() are GS and MGS. Because efficiency is important, especially when the number of vectors in V, k, is large, block GS with some form of reorthogonalization is usually employed. As we show below, accuracy close to machine precision is less critical because the orthogonality achieved in one phase may not be preserved in the other.

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

In finite precision, the above algorithm does not always compute an accurately orthonormal set Q, and thus it has to be applied in an iterative fashion. To develop an efficient iterative version of Ortho-SVQB, we must first examine the numerical interaction between the two phases.

5.1. Numerical interplay of the phases. The reason that full orthogonality is not always necessary in each phase is that SVQB procedure may destroy previous orthogonality against V, and Ortho(V, W) may destroy the orthogonality in W.

LEMMA 5.1. *Let W be a set of vectors with $\|V^T W\| \leq \mu \|W\|$. Let $\bar{Q} =$ SVQB(W) be the result of the internal step of the Ortho-SVQB algorithm. Then,*

$$\|V^T \bar{Q}\| = O\left((\mu + \epsilon) \ \min\left(\kappa(W), \frac{1}{\sqrt{\epsilon}}\right)\right).$$

Proof. Following the notation of Theorem 4.1, let $\bar{Q} = W\bar{U}\bar{\Lambda}^{-1/2} + \delta Q$. Using bounds (4.8) and (4.9) we have $\|V^T\bar{Q}\| = \|V^T W\bar{U}\bar{\Lambda}^{-1/2} + V^T\delta Q\| \leq \mu\|W\|\|\bar{\Lambda}^{-1/2}\| + O(\|\delta Q\|) = O\left((\mu+\epsilon)\min(\kappa(W), 1/\sqrt{\epsilon})\right)$. ∎

The lemma states that even when W is exactly orthogonal to V, i.e., $\mu = 0$, the SVQB procedure at the next step may destroy that orthogonality up to a maximum of $\sqrt{\epsilon}$. For example, consider the following matrices:

$$V = \begin{bmatrix} 0.17164335073404 \\ 0.00000000003278 \\ -0.17164335076682 \end{bmatrix} \text{ and } W = \begin{bmatrix} 1 & 1 \\ 1 & 1+10^{-6} \\ 1 & 1 \end{bmatrix}.$$

A Matlab computation shows that $\|V^T W\| =$ 3.2e-17, and $\kappa(W) =$ 4.2e+6. After the step $Q =$ SVQB(W), we observe that $\|V^T Q\| =$ 4e-11, which agrees with our lemma. Therefore, it is not important to choose one of the more accurate Ortho() methods, such as MGS, to obtain good orthogonality against V, as this may be lost later.

The Ortho() procedure in the first step of the Ortho-SVQB algorithm has an even worse effect on the orthogonality of the W vectors.

LEMMA 5.2. *Let $V^T V = I$, and W a set of normal vectors with $\|W^T W - I\| = \nu$, and $\|V^T W\| = \delta < 1$. Let $Q =$ Ortho$(V, W) = (W - VV^T W)D^{-1/2}$ be the normalized result of the external orthogonalization. Assume that there is no floating point error in computing Q. Then,*

$$\|Q^T Q - I\| \leq \frac{\nu + 2\delta^2}{1 - \delta^2}.$$

Proof. If we let $S = V^T W$, we have $D_{ii} = w_i^T w_i - w_i^T VV^T w_i = 1 - e_i^T S^T S e_i$. Note that for all diagonal elements of $S^T S$, it holds that $e_i^T S^T S e_i \leq \|S^T S\| = \delta^2 < 1$. As a result, for all diagonal elements of D, we have $D_{ii} > 1 - \delta^2$. This holds for the $\min(D_{ii})$ too, and therefore $\|D^{-1}\| < 1/(1-\delta^2)$. In addition, we see that for all i, $1/D_{ii} - 1 < \delta^2/(1-\delta^2)$. From the above we can compute

$$\begin{aligned} \|Q^T Q - I\| &= \|D^{-1/2}(W^T W - I)D^{-1/2} + D^{-1} - I - D^{-1/2}(W^T V)(V^T W)D^{-1/2}\| \\ &\leq \nu\|D^{-1}\| + \|D^{-1} - I\| + \|D^{-1}\|\|S^T S\| \\ &\leq \nu/(1-\delta^2) + \delta^2/(1-\delta^2) + \delta^2/(1-\delta^2) = (\nu + 2\delta^2)/(1-\delta^2). \end{aligned}$$ ∎

The lemma states that even when the vectors W are orthonormal, i.e., $\nu = 0$, they will lose their mutual orthogonality after the Ortho() step if W is not sufficiently

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

orthogonal against V (i.e., $\|V^T W\| > \sqrt{\epsilon}$). In finite precision, additional orthogonality loss is expected. The bound above is sharp within a constant. For example, consider

$$V = \begin{bmatrix} a \\ a \\ \sqrt{1-2a^2} \end{bmatrix} \text{ and } W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Initially, $W^T W = I$, but after $Q = \text{Ortho}(V, W)$, we can verify that $\|Q^T Q - I\| = a^2/(1-a^2)$. The lemma gives a bound of $4a^2/(1-2a^2)$, which for small a it is four times larger than the actual loss of orthogonality.

6. The iterative GS-SVQB algorithm. Section 5.1 suggests that GS is sufficient for the external phase, so the rest of the paper focuses on the GS-SVQB algorithm. Figure 6.1 shows four possible GS-SVQB implementations, based on which of the two steps (GS or SVQB) is carried out iteratively. We choose the most appropriate algorithm based on computational considerations and on the developed theory.

Algorithm 1: **repeat** $Q^{(i)} = \text{GS}(V, W^{(i-1)})$ $W^{(i)} = \text{SVQB}(Q^{(i)})$ **until** $\left(W^{(i)T} W^{(i)} = I \text{ and } W^{(i)} \perp V\right)$	Algorithm 2: **repeat** **repeat** $Q^{(i)} = \text{GS}(V, W^{(i-1)})$ **repeat** $W^{(i)} = \text{SVQB}(Q^{(i)})$ **until** $\left(W^{(i)T} W^{(i)} = I \text{ and } W^{(i)} \perp V\right)$
Algorithm 3: **repeat** **repeat** $Q^{(i)} = \text{GS}(V, W^{(i-1)})$ $W^{(i)} = \text{SVQB}(Q^{(i)})$ **until** $\left(W^{(i)T} W^{(i)} = I \text{ and } W^{(i)} \perp V\right)$	Algorithm 4: **repeat** $Q^{(i)} = \text{GS}(V, W^{(i-1)})$ **repeat** $W^{(i)} = \text{SVQB}(Q^{(i)})$ **until** $\left(W^{(i)T} W^{(i)} = I \text{ and } W^{(i)} \perp V\right)$

FIG. 6.1. *Four possible iterative implementations of the GS-SVQB algorithm. The outer loop is repeated until W becomes numerically orthonormal and orthogonal to V. The inner loops could be repeated until full orthogonalization is achieved or for a specified number of steps. Our theory suggests that Algorithm* 4 *is the most preferable.*

First, we note that GS is expensive because the size of V is usually much larger than that of W, so we try to minimize the number of times it is repeated. Second, two or three applications of GS are usually sufficient to produce full orthogonality against V. However, according to Lemma 5.1, such an orthogonality could be wasted by as much as $\epsilon\kappa(W)$ in the SVQB step. Thus, Algorithms 2 and 3 that apply GS repeatedly are inappropriate. Algorithm 1 also may result in wasted work when SVQB and GS do not reach synergistic levels of accuracy.

Algorithm 4 seems the most appropriate choice. Note that iterating SVQB to produce good orthogonality within W is justified, since at the following outer step GS can destroy it only by $O(\|V^T W\|^2)$ (Lemma 5.2). Especially if $\|V^T W\| \leq O(\sqrt{\epsilon})$ is obtained at the current step, orthogonality within W will be maintained fully.

6.1. Tuning the algorithm. The next step is to identify efficient and practical conditions for terminating the outer and inner repeat loops of Algorithm 4. To avoid unnecessary work, Lemmas 5.1 and 5.2 suggest that the two steps, GS and SVQB, must be balanced by keeping both $\kappa(W)$ and $\|V^T W\|$ comparably small.

First, we seek the conditions under which the final outer iteration i does not require SVQB applications. Because $Q^{(i)}$ must be orthonormal, $i > 1$, and $Q^{(i)} =$

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

GS$(V, W^{(i-1)})$ must not have destroyed the internal orthogonality of $W^{(i-1)}$. Thus, Lemma 5.2 implies test (6.1), which can be checked inexpensively as a GS by-product:

$$\|V^T W^{(i-1)}\| < \sqrt{\epsilon}. \tag{6.1}$$

We also need to test whether $W^{(i-1)}$ was orthonormal before the GS step. Since $\kappa(W^{(i-1)})$ is not known yet, we use the singular values of $Q^{(i-1)}$ obtained in the last SVQB. Theorem 4.1 implies that if SVQB produced an orthonormal $W^{(i-1)}$, then $\kappa(Q^{(i-1)}) = O(1)$. Therefore, this condition must be tested along with (6.1):

$$\textbf{if } \|V^T W^{(i-1)}\| < \sqrt{\epsilon} \textbf{ and } \kappa(Q^{(i-1)}) = O(1) \textbf{ then exit} \tag{6.2}$$

The outer loop should repeat if the GS procedure has not managed to orthogonalize $W^{(i-1)}$ against V. We perform reorthogonalization according to a popular test due to Daniel et al. [7], whenever the norm of $Q^{(i)}$ becomes less than 0.7 times the norm of $W^{(i-1)}$. However, SVQB also can cause loss of orthogonality against V.

Assume that the inner loop of SVQB produces $W^{(i)}$ with $\kappa(W^{(i)}) > O(1)$. Obviously, GS at the next outer iteration will not reduce the condition number of $Q^{(i+1)} = \text{GS}(V, W^{(i)})$. However, the next application of SVQB, $W^{(i+1)} = \text{SVQB}(Q^{(i+1)})$, will destroy orthogonality versus V by as much as $\epsilon\kappa(Q^{(i+1)})$ (Lemma 5.1), making a third $(i+2)$ outer iteration necessary. To avoid this, the SVQB inner loop should be iterated to produce at least $\kappa(W^{(i)}) = O(1)$. The inner loop executes at least once, to guarantee full orthonormality of $W^{(i)}$, when no second outer iteration is needed.

We summarize the above analysis into the following algorithm. The condition number κ_{last} is computed from the eigenvalues of the matrix $S = Q^{(j-1)T} Q^{(j-1)}$ and thus corresponds to the matrix before the application of the last SVQB. A bound on the resulting $\kappa(W^{(i)})$ can be inferred through Theorem 4.4, and this is what the until condition checks. Finally, note that $\|V^T W^{(i-1)}\|$ can be computed during GS.

ALGORITHM 6.1. $Q = \text{iGS-SVQB}(V, W)$ *(iterative GS-SVQB method).*

$W^{(0)} = W$
$\kappa_{\text{last}} =$ *large number*
$i = 1$
repeat
  $\delta = \|V^T W^{(i-1)}\|$
  $W^{(i)} = \text{GS}(V, W^{(i-1)})$
  if $(\delta < \sqrt{\epsilon})$ **and** $(\kappa_{\text{last}} = O(1))$
    break *(skip final* SVQB*)*
  $Q^{(0)} = W^{(i)}$
  $j = 1$
  Reortho = ((*Daniel's test*) **or** $(\kappa(W^{(i)}) > O(1))$)
  repeat
    $\kappa_{\text{last}} = \kappa(Q^{(j-1)})$
    $Q^{(j)} = \text{SVQB}(Q^{(j-1)})$
    $j = j + 1$
  until $(\kappa_{\text{last}} < O(1/\sqrt{\epsilon}))$
  $W^{(i)} = Q^{(j-1)}$
  $i = i + 1$
until (*Reortho* = *false*)

7. Further optimizations. The above is a block algorithm that targets performance. However, the tests it performs are pessimistic as they apply on the block

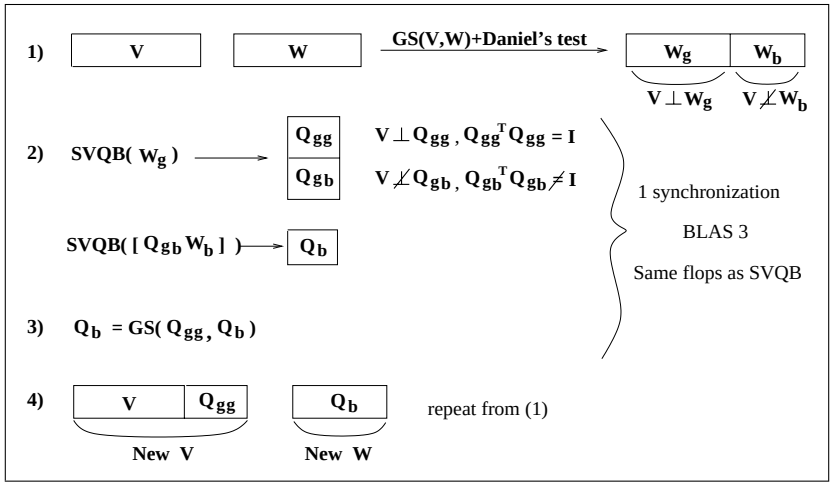

FIG. 7.1. *A practical implementation of the SVQB algorithm.*

W as a whole. This may be wasteful, since individual vectors in W may become orthogonal to V and to other vectors in W. Further reorthogonalizations should exempt these vectors, performing computations on a smaller block. Fortunately, we can perform tests on individual vectors and adjust the block dynamically, without affecting the block structure or the number of synchronizations of the algorithm. Interestingly, large bounds in Lemmas 5.1 and 5.2 can be the result of only a couple of ill-conditioned vectors. By grouping vectors based on individual tests, the bounds still apply, only for smaller blocks, and thus provide better direction to the algorithm.

Specifically, after the GS phase and without additional computation, we can separate those vectors that do not need reorthogonalization (good vectors) and those that need it (bad vectors), $W = [W_g W_b]$. However, some of the good vectors may be very close to other vectors in W. Moreover, because the SVQB phase mixes all vectors, the identity of the good ones will disappear.

We can solve this problem inexpensively. As in the regular SVQB, we compute $S = \left[\begin{smallmatrix} S_g & S_{bg}^T \\ S_{bg} & S_b \end{smallmatrix}\right] = [W_g\ W_b]^T[W_g\ W_b] = W^TW$. First we perform an eigenvalue decomposition of S_g. This will subdivide the good group into $W_g = [Q_{gg}\ Q_{gb}]$. The group Q_{gg} consists of all the singular vectors corresponding to large singular values of S_g. These Q_{gg} vectors are orthogonal both to V and to each other and can be appended to V in future iterations. Note that Q_{gg} has at least one vector. The group Q_{gb} consists of all the singular vectors with small singular values, and so they may have lost their orthogonality against V as well. Therefore, we should combine the Q_{gb} with the W_b vectors and apply $Q_b = \text{SVQB}([Q_{gb}\ W_b])$. Finally, the resulting Q_b needs to be orthogonalized versus Q_{gg}. The many implementation details fall beyond the scope of this paper. Figure 7.1 shows the conceptual steps of this algorithm.

Notice that all of the above eigenvalue decompositions, orthogonalizations, vector groupings, and tests are performed not on the W vectors, but on $m \times m$ matrices and their eigenvectors, with negligible computational cost. The main operation is still $S = W^TW$, which is BLAS 3 and incurs only one synchronization.

7.1. Variable block algorithm. When the number of vectors in W is large, applying the SVQB method on the full block may not always be cache efficient or

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

numerically stable. Typically, there is an optimal block size beyond which cache performance decreases. In addition, the conditioning of W is bound to deteriorate with block size. For these reasons, we want a variable block size that can be tuned according to the machine and the problem.

Let b be the desirable block size. We partition W into $p = m/b$ sets of vectors, $W = [W_1, \ldots, W_p]$, and apply the iGS-SVQB on each one individually. After a W_i subblock is made orthonormal and orthogonal to V and to previous W_j, $j < i$, it is locked with V and the next W_{i+1} is targeted. The algorithm follows.

ALGORITHM 7.1. $Q = \text{bGS-SVQB}(V, W, b)$ *(variable block* iGS-SVQB *method).*
$p = m/b$
Partition $W = [W_1, \ldots, W_p]$
$Q = [\,]$, $Z = V$
for $i = 1, p$
  $Q_t = \text{iGS-SVQB}(Z, W_i)$
  $Z = [Z, Q_t]$
  $Q = [Q, Q_t]$

The number of synchronization points in the bGS-SVQB algorithm is $O(p)$, and a larger percentage of the computation is spent on the GS procedure. For $b = 1$, the algorithm reduces to the classical GS method, while for $b = m$ the algorithm is the iGS-SVQB method. We expect to identify a range of block sizes for which the cache performance is optimal, while the synchronization requirements are not excessive.

8. Timing experiments. We have tested our implementation of bGS-SVQB against a variety of orthogonalization alternatives. To provide a common comparison framework for all methods, we use a "bGS-Method" algorithm that is identical to our bGS-SVQB, except that a different method is used to orthogonalize the block.

The first method is the classical GS algorithm with reorthogonalization. This is the only method whose structure differs slightly from the "bGS-Method." For GS, it is more efficient to orthogonalize each vector in the block at once against all V vectors and all previously orthogonalized vectors in W. Thus, block size does not affect the behavior of GS, and in the figures we simply refer to it as GS.

We also compare against the QR factorization with Householder reflections. The method is denoted as bGS-QR and uses the QR implementation from the ScaLAPACK library [4] for both single and multiprocessor platforms. Finally, we compare against the computationally similar CholQR method. The method, denoted as bGS-CholQR, uses the (sequential) Cholesky decomposition in the ScaLAPACK library.

All algorithms have been implemented in Fortran 90, using MPI, and run on the Cray T3E 900 and the IBM SP2 parallel computers at NERSC National Lab, and on a 64-node cluster of SUN Ultra 5s at the College of William and Mary. 256 MB of memory are available on each node of all machines, while on the SP2 the nodes are two-processor SMP nodes, but they are assigned individual MPI processes. The T3E network is considerably faster than the SP2 and the COW networks (Power Switch and Fast Ethernet, respectively). On the NERSC platforms we link with the MPICH libraries and we use the machine optimized libraries for ScaLAPACK and BLAS. On the SUN cluster, we use LAM MPI and BLAS 3/2 kernels automatically optimized with ATLAS from the University of Tennessee [23].

Our first numerical example is an easily reproducible set of 30 Krylov vectors of the diagonal matrix `A = diag([1:n])` (in Matlab notation), and the initial vector `x = [ 1 log([2:n]) ]'`, with `n = 500000`. We let $V = \emptyset$ and build W as the set of normalized vectors, $W = \left[x, Ax, A^2x, \ldots, A^{29}x\right]$. The condition number of the result-

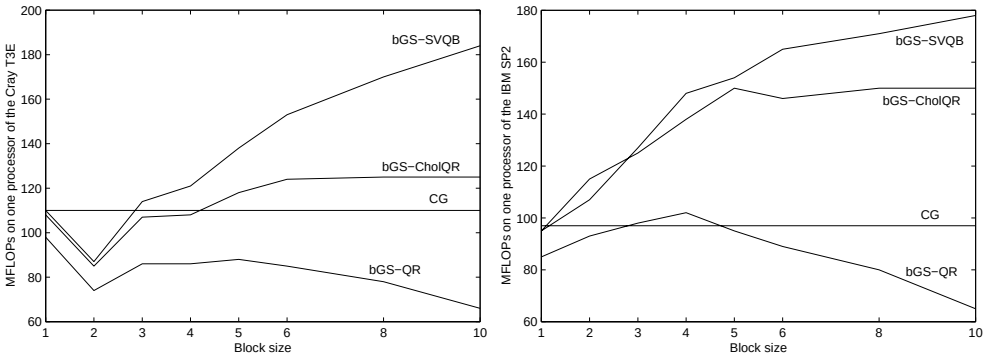

FIG. 8.1. *Single node MFLOPS as a function of block size for the four methods. The clear performance advantage of block methods is expected to counterbalance the increase in the number of FLOPs. The methods orthonormalize* 30 *vectors of dimension* 500000. *Left graph depicts Cray T*3*E results. Right graph depicts IBM SP*2 *results.*

ing set W, as computed by the Matlab `cond` function, is 1.3892E+20. The goal is to orthonormalize the set W as accurately as possible, through various orthogonalization methods and block sizes: bGS-Method(V, W, b). Initially, most of the computation is spent on the "Method," while as more blocks get orthonormalized GS takes over, reducing the computational differences between methods.

Figure 8.1 illustrates the single-node floating point performance (MFLOP rate) achieved for each of the four algorithms as a function of block size, on the T3E (left graph), and on the SP2 (right graph). As expected, the GS rate is constant regardless of block size. The single-node performance of the bGS-QR does not improve with block size on either machine, which points both to the ScaLAPACK implementation and to the inherent block limitations of the QR. On the other hand, the block structure of SVQB and CholQR allows them to outperform GS significantly, even for small blocks of 8–10 vectors. The Cholesky back-solve implementation seems to better exploit the architecture of the SP2 than the T3E. However, on both platforms, bGS-SVQB improves GS performance by at least 70–80% for these small block sizes. We should mention that the block MGS method in [14] is expected to have worse single-node performance than GS because only one of the two phases involves BLAS 3 kernels.

Good node performance is important only if it leads to accurate and faster orthogonalization. All of the algorithms tested produced a final orthonormal set Q, with $\|Q^TQ - I\| = 10^{-13}$. Figure 8.2 shows that execution times of block methods are superior to the GS method. The graphs plot execution time as a function of block size, for three methods, and for various numbers of processors. The left graph corresponds to the Cray T3E and the right one to the IBM SP2. The bGS-CholQR and bGS-SVQB are consistently faster than GS, for any block size on the SP2, and for block sizes of 4 or above on the T3E. It is also clear, because of the logarithmic time scale, that the relative improvement over the GS timings persists on a large number of processors, despite smaller local problem sizes. bGS-SVQB is 20% faster than GS on the T3E and more than 25% faster on the SP2. Note that the good performance of bGS-CholQR on the SP2 (35% faster than GS) does not carry over to the T3E.

Our next experiment measures the effects of synchronization as the number of nodes increases by fixing the problem size on each processor and using a constant block size of 6. For this test, the set W has 30 Krylov vectors generated by the matrix

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

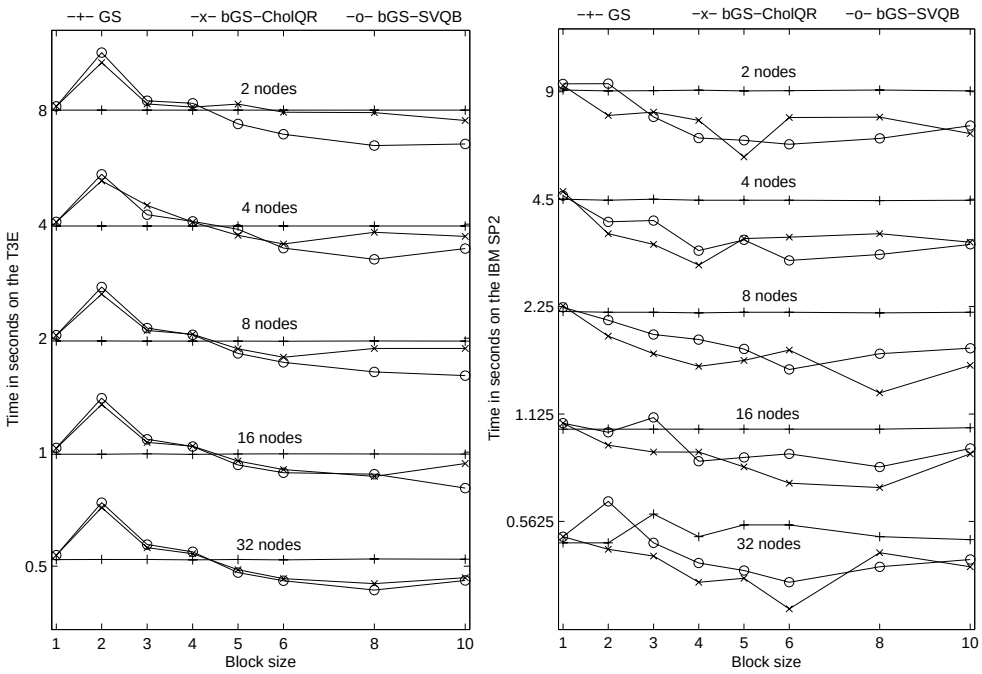

FIG. 8.2. *Time versus block size for three methods and for various numbers of processors. Results obtained on the T3E (left graph) and on the IBM SP2 (right graph). Time scale is logarithmic.*

of the discretized Laplacean on a three-dimensional cube. Every processor holds a $32 \times 32 \times 32$ uniform grid locally (so the matrix size is proportional to the number of nodes) and uses it to create the Krylov space. The W vectors are generated in chunks of six successive Krylov vectors, and each chunk is orthonormalized by a call to bGS-Method$(V, W, 6)$. Figure 8.3 plots the execution times of the four methods over a wide range of processor numbers. The T3E has been used for this experiment because of the large number of available nodes. In the absence of communication/synchronization costs, the times should be equal for all processors. The time increase observed in the figure is relatively small for all methods because of the extremely fast T3E network. However, the effects are more apparent on GS and bGS-QR, as their curves increase faster than the respective ones for bGS-CholQR and bGS-SVQB. Finally, on this problem the bGS-SVQB is more than 30% faster than GS (an improvement over the previous numerical problem).

Because of superscalar processors and a higher-latency network, we expect the block algorithms to perform better on the SUN cluster. We use the same test case of 30 Krylov vectors from the uniform-grid Laplacean, with each processor storing a $32 \times 32 \times 32$ subgrid. The left graph in Figure 8.4 shows the effect of blocking on the single-node execution time of the algorithms. The effects are much more dramatic than on the other machines, as execution time is reduced by about half. This is attributed to the ATLAS fine tuning of the BLAS kernels. Note that CholQR improves single-node performance on this architecture. The time variability is typical of caching effects. The right graph of the figure shows the scalability of the algorithms under constant computational load per processor. A block size of 12 is used, i.e., bGS-Method$(V, W, 12)$. Our proposed methods clearly outperform GS, and their time

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

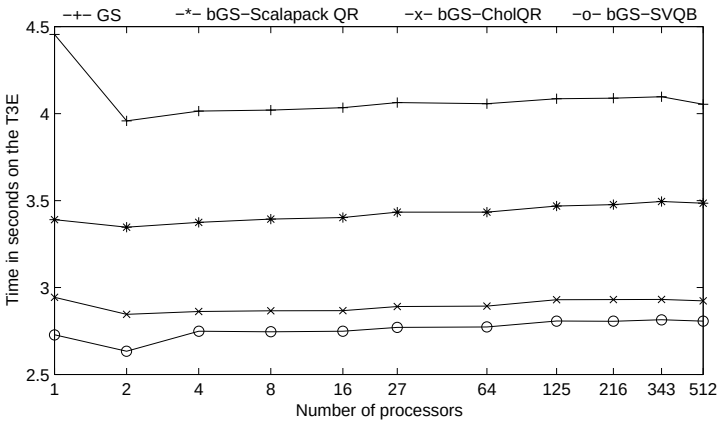

FIG. 8.3. *Scalability of four methods on the T3E, under constant problem size (*$32768 = 32^3$ *vector rows) per processor and block size of* 6. *Ideal scalability would show as a flat horizontal line.*

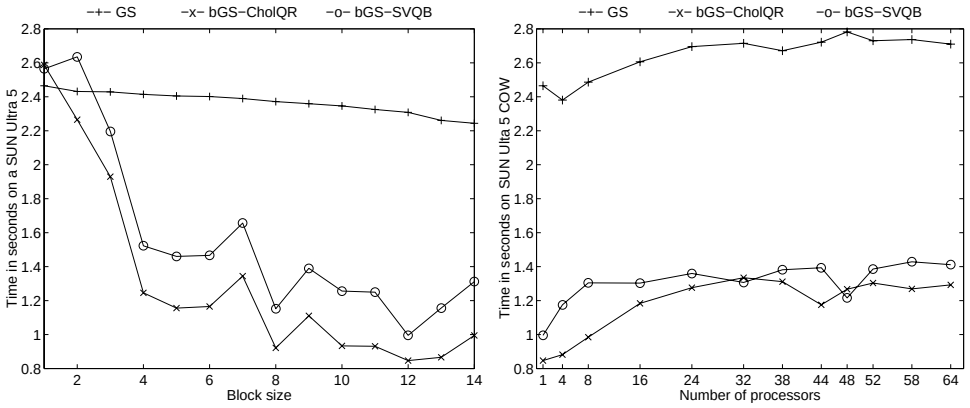

FIG. 8.4. *Left graph: execution time of three methods versus block size on a single SUN Ultra* 5. *BLAS* 3 *libraries are optimized with ATLAS. Right graph: scalability of the three methods on a cluster of* 64 *SUN Ultra* 5*s, under constant problem size (*$32768 = 32^3$ *vector rows) per processor and block size of* 12. *Ideal scalability would show as a flat horizontal line.*

is not only substantially smaller than GS but also seems to increase slower with the number of processors.

9. Conclusions. We have introduced and analyzed a new method, called SVQB, that computes an orthonormal basis of a skinny matrix W from its right singular vectors. The method is attractive computationally because it involves only BLAS 3 kernels and it requires only one synchronization point in parallel implementations. We have proved that the departure from orthonormality of the resulting vector set is bounded by $O(\epsilon\kappa(W)^2)$, if $\kappa(W) < O(1/\sqrt{\epsilon})$. We have also considered the problem of two phase orthonormalization, where a block of vectors is orthonormalized against a previously orthonormal set of vectors with GS and among itself with SVQB. Computational efficiency suggests the independent, block application of each of the methods. However, each phase impairs the orthogonality produced during the other phase. We have provided bounds that describe this numerical interdependence and

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

have used them to balance the work performed by each of the two orthogonalization phases within an iterative scheme. Our Matlab examples have demonstrated that our theoretical bounds are in accordance with practice, and our parallel implementations on the Cray T3E, the IBM SP2, and on a SUN COW have shown that our method improves the performance of other orthonormalization alternatives.

REFERENCES

[1] Å. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[2] Å. Björck, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 198 (1994), pp. 297–316.

[3] Å. Björck and C. C. Paige, *Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 176–190.

[4] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK User's Guide*, SIAM, Philadelphia, 1997.

[5] S. Chaturvedi, A. K. Kapoor, and V. Srinivasan, *A New Orthogonalization Procedure with an Extremal Property*, Technical Report quant-ph/9803073, LACS Stanford, Palo Alto, CA, 1998.

[6] C. K. Chui, *Wavelet Analysis and Its Applications*, Academic Press, San Diego, 1992.

[7] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[8] J. Dongarra, J. DuCrouz, I. Duff, and S. Hammarling, *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.

[9] J. Dongarra, J. DuCrouz, S. Hammarling, and R. Hanson, *An extended set of FORTRAN basic linear algebra subprograms*, ACM Trans. Math. Software, 14 (1988), pp. 1–32.

[10] W. Gander, *Algorithms for the QR Decomposition*, Technical Report TR 80-02, Angewandte Mathematik, ETH Zurich, Zurich, Switzerland, 1980.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[12] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 361–377.

[13] W. Hoffman, *Iterative algorithms for Gram-Schmidt orthogonalization*, Computing, 41 (1989), pp. 335–348.

[14] W. Jalby and B. Philippe, *Stability analysis and improvement of the block Gram–Schmidt algorithm*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1058–1073.

[15] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–325.

[16] P. O. Lödwin, J. Chem. Phys., 18 (1950), p. 365.

[17] P. O. Lödwin, Adv. Quant. Chem., 23 (1992), p. 84.

[18] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1997.

[19] Y. Saad, *Iterative methods for sparse linear systems*, PWS, 1996.

[20] H. C. Schweinler and E. P. Wigner, *Orthogonalization methods*, J. Math. Phys., 11 (1970), p. 1693.

[21] J. P. Singh, D. E. Culler, and A. Gupta, *Parallel Computer Architecture. A Hardware/Software Approach*, Morgan Kaufmann, San Francisco, 1999.

[22] A. Stathopoulos and J. R. McCombs, *A parallel, block, Jacobi-Davidson implementation for solving large eigenproblems on coarse grain environments*, in Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, CSREA Press, 1999, pp. 2920–2926.

[23] R. C. Whaley and J. J. Dongarra, *Automatically Tuned Linear Algebra Software*, Technical Report UT-CS-97-366, University of Tennessee, Knoxville, TN, 1997.

[24] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.

Downloaded 11/25/14 to 141.217.58.222. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php