LBNL-45577

A BLOCK ORTHOGONALIZATION PROCEDURE WITH CONSTANT SYNCHRONIZATION REQUIREMENTS

ANDREAS STATHOPOULOS * AND KESHENG WU †

Abstract. We propose an alternative orthonormalization method that computes the orthonormal basis from the right singular vectors of a matrix. Its advantage are: a) all operations are matrix-matrix multiplications and thus cache-efficient, b) only one synchronization point is required in parallel implementations, c) could be more stable than Gram-Schmidt. In addition, we consider the problem of incremental orthonormalization where a block of vectors is orthonormalized against a previously orthonormal set of vectors and among itself. We solve this problem by alternating iteratively between a phase of Gram-Schmidt 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, Poincare

AMS Subject Classification. 65F15

1. Introduction.

Computing an orthonormal basis from a given set of vectors is a basic computation, common in 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.

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 [12, 11]. 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].

This problem differs from the classical QR factorization in that the external set of vectors should not be modified. Therefore, a two phase process is required; first orthogonalizing the internal vectors against the external, and second the internal among themselves. Usually, the number of the internal vectors is much smaller than the external ones, and significantly smaller than their dimension. Another important difference is that the accuracy of the R matrix of the QR factorization is not of primary interest, but rather the orthonormality of the produced vectors Q. A variety of orthogonalization techniques exist for both phases. For the external phase, Gram-Schmidt (GS) and its modified version (MGS) are the most competitive choices. For the internal phase, QR factorization using Householder transformations is the most stable, albeit more expensive method [11]. When the number of vectors is significantly smaller than their dimension, MGS or GS with reorthogonalization are usually preferred.

Computationally, MGS, GS and Housholder transformations are based on level 1 or level 2 BLAS operations [15, 9, 8]. These basic kernel computations, dot products, vector updates and sometimes matrix-vector operations, cannot fully utilize the

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

†NERSC, Lawrence Berkeley National Laboratory, Berkeley, California 94720, (kwu@lbl.gov).

processor cache, with significant performance shortcomings on most of todays computers. In addition, these methods have high synchronization requirements when implemented on parallel computers. The number of synchronization points of the GS is linear to the number of vectors, while for the rest methods it is quadratic [14]. When the parallel platform involves a high latency network but fast processors, as in the case of the increasingly popular clusters of workstations [21], the number of synchronization points can have an adverse effect in the scalability of the method.

Most of the previous efforts to address these problems considered blocks of vectors, and used hybrids of the more scalable GS for orthogonalization between blocks, and the more accurate MGS for orthogonalization within blocks [14, 2]. By adjusting the block size appropriately, these techniques can produce accurate orthonormal sets while reducing the synchronization requirements of MGS. However, the number of synchronization points is still linear to the number of vectors, and BLAS level 2 kernels are still dominant despite blocking. Finally, most of such efforts have focused on a full QR factorization of a set of vectors, rather than 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 subspace. The idea itself is not new, dating back at least to Poincare, and it is sometimes encountered in chemistry and wavelet literature [5, 20, 16, 17, 6]. 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. Beyond theoretical considerations, the method, which we call SVQB, has some very attractive characteristics. It uses primarily level 3 BLAS kernels, and it has a constant number of synchronization points. As we show in this paper, it is not as numerically accurate as MGS, but it is often better than GS. 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. We first describe the basic idea of the SVQB method, and we couple it with a block GS for the two phase problem. The numerical stability of the method is analyzed in the following section, and some numerical experiments and comparisons with other methods verify its effectiveness. We then analyze the numerical interaction between the methods used in each of the two phases, and based on this theory we tune the two phases so that they do not perform unnecessary reorthogonalizations. Following, we present timings from a series of experiments on the Cray T3E, and on the IBM SP-2. These verify that both the block computations and the small number of synchronizations help the new method achieve accurate orthogonality, faster than other competitive methods.

2. Orthogonalization methods. 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 usually expect $m < k$ and $m \ll n$. The problem is to obtain an orthonormal set of vectors Q, such that span(Q)=span(W) and $Q \perp V$. This problem can be viewed as consisting of two phases: obtaining a Q which is orthogonal to V (external phase), and orthonormalizing the Q vectors among themselves (internal phase). The internal phase problem is most commonly solved through various methods for QR factorization [11, 23].

Gram-Schmidt (GS) is the most well known and also the least computationally expensive method for QR factorization. In addition, GS is based primarily on level

2 BLAS kernels, and it is possible to implement it in parallel with a modest number of $m+1$ synchronization points. Although computationally attractive, GS does not always produce numerically orthogonal vectors [2]. However, when used with reorthogonalization, a second GS iteration is typically enough for producing orthogonality to machine precision [7]. Since a second iteration is not always necessary, GS with reorthogonalization is often preferred over other methods.

Householder reflections, on the other hand, yield a QR factorization that is numerically accurate. By construction, the resulting matrix Q is orthogonal to machine precision (ϵ), and the computed upper triangular matrix R differs from the exact one by less than $\epsilon\|A\|$ [11]. Despite its excellent numerical characteristics, this technique involves twice the arithmetic of the GS. For a slight increase in arithmetic, a block version is also possible that facilitates the use of level 3 BLAS kernels. However, when $m \ll n$, this blocking is harder to obtain and it is less effective. Moreover, the number of synchronization points it requires grows quadratically with m.

Modified Gram-Schmidt (MGS) improves the numerical stability of the GS by orthogonalizing individual pairs of vectors rather than a vector against a block. For MGS, the error in the orthogonality of Q can be bounded by $\epsilon\kappa(W)$, where $\kappa(W)$ is the condition number of W [1, 3]. The operation count for MGS is the same with GS, but working on individual vectors allows only for level 1 BLAS kernels, which impairs significantly the actual performance on cache based computers. Finally, the number of synchronization points grows quadratically with m.

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, if the internal vectors W are orthogonal to machine precision to all external vectors V, the internal orthogonalization could spoil significantly this external orthogonality. Therefore, a second iteration through the algorithm would still be necessary. If W and V are far from orthogonal, obtaining a fully orthonormal Q is also unnecessary because the external orthogonalization will spoil the orthogonality of Q. Therefore, this two phase problem obviates the use of expensive but stable methods such as Householder.

3. The SVQB method. As with the two phase problem, it is often the case that an orthonormal basis of the span(W) is needed, rather than a QR factorization of W. There are various ways of obtaining such a basis, but an especially interesting one 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, $\mathrm{span}(Q) = \mathrm{span}(W)$ and $Q^TQ = I$. If the W vectors are not normalized, the diagonal of S, $D = \mathrm{diag}(S)$, contains the squares of their norms ($S_{ii} = W_i^TW_i$). Therefore, we can perform a normalization implicitly by scaling the columns and rows of S, and work with the implicitly normalized $WD^{-1/2}$. This implicit normalization is not only inexpensive, but also as numerically stable as explicit normalization. The resulting factorization is not a QR but rather a 'QB' 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. $Q = \mathrm{SVQB}(W)$
1. $S' = W^TW$
2. *Scale* $S = D^{-1/2}S'D^{-1/2}$, *with* $D = diag(S)$

3. *Solve $Su = u\Lambda$, for all eigenpairs*
4. *Compute $Q = WD^{-1/2}u\Lambda^{-1/2}$*

When some of the vectors in W are linearly dependent, one or more of the eigenvalues are zero, and the scaling in the computation of Q cannot be carried out. In finite precision, the same effect is caused by almost linearly dependent vectors and eigenvalues close to zero. To prevent normalization overflows, 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 ($Q^{(i+1)} = \mathrm{SVQB}(Q^{(i)})$), until an orthonormal set Q is obtained.

Both the solution of the eigenvalue problem and the implicit normalization involve only $m \times m$ matrices (S and u), and thus can be computed inexpensively. On the other hand, 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 both 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 will have to be used. Moreover, if implemented in parallel, the SVQB method method requires only one synchronization point, when computing the matrix S.

3.1. The Choleski QR method. Similarly to the SVQB method, we can derive a block QR factorization based on the Choleski factorization. Note, that if $S = W^TW = R^TR$, where R is the Choleski factor, $Q = WR^{-1}$ defines the QR factorization for W [11]. Although this method is rarely or never 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 frequently it can be more stable than GS [2, 10]. One of the problematic issues with this (denoted as CholQR) method is that the more ill-conditioning S is, the less stable the Choleski factorization becomes. Regularizing it in some efficient and effective way is not as straightforward as in the case of the SVQB, where the smallest singular pairs can be simply left out of the computation. Finally, the cache performance of the CholQR is usually inferior to that of the SVQB, because of the triangular solve.

3.2. The two phase GS-SVQB. In the two phase case, the external vectors V should not be modified, and thus external orthogonalization is performed by either GS or MGS. Better stability is certainly an advocate for MGS. However, in many problems k, the number of vectors in V, is large and performing this operation as efficiently as possible is crucial. In addition, accuracy close to machine precision is less critical because the orthogonality achieved in one phase may not be preserved in the other. For this reason, the block GS with BLAS 3 kernels and some form of reorthogonalization mechanism is usually preferred. If we denote as $Q = \mathrm{GS}(V, W)$ the block GS process, $(I - VV^T)W$ followed by a normalization, the two phase algorithm can be described as:

ALGORITHM 3.2. $Q = \text{GS-SVQB}(V, W)$
1. $W' = \text{GS}(V, W)$
2. $Q = \text{SVQB}(W')$

In practice, 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 GS-SVQB, we must first examine the numerical characteristics of the SVQB algorithm. This is the goal of the following section.

4. Stability analysis of the SVQB. For many applications, such as block Krylov iterative methods, the orthonormality of the resulting vectors Q is more important, than the accuracy of the upper triangular R of the QR factorization. For example, Krylov methods will still make progress, albeit a slower one, if provided with an orthonormal set Q. Moreover, the actual span(W) of an almost linearly dependent W is not well defined numerically. Therefore, in this paper we measure stability as the departure of the resulting Q from orthonormality rather than the backward error. Because of its block, non sequential nature, we expect the SVQB procedure to be less stable than the MGS. But as we show below, it can often be 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*

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

where $\|\| = \|\|_2$, ϵ is the machine round off, and ct is a constant depending on n and m.

Proof. Let $S = W^TW$, and denote by $\tilde{S}$ 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 $\|\delta S\| \leq c_1\; \epsilon\; m$.

From standard backward error analysis, the solution of the small $m \times m$ symmetric eigenvalue problem $\tilde{S}u = u\Lambda$ 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 $\bar{Q} = Q + \delta Q = W\bar{u}\bar{\Lambda}^{-1/2} + \delta Q$ be the floating point representation of the matrix returned by the SVQB procedure. Then, $\|\delta Q\| \;\leq c_4\; \epsilon\; \|W\|\; \|\bar{u}\|\; \|\bar{\Lambda}^{-1/2}\|$. Note that $\|\bar{\Lambda}^{-1/2}\| = \frac{1}{\sqrt{\bar{\lambda}_{min}}}$, where $\bar{\lambda}_{min} = \min_i \bar{\Lambda}_{ii}$, and since $\bar{u}$ are orthonormal eigenvectors, we have:

$$\|\delta Q\| \leq c_4\; \epsilon\; \frac{\|W\|}{\sqrt{\bar{\lambda}_{min}}}. \tag{4.4}$$

It is well known that for symmetric eigenproblems the error in the eigenvalue is bounded by the error in the matrix. Thus, the minimum eigenvalue of $\bar{S}$ could have a large error:

$$\left|\bar{\lambda}_{min} - \lambda_{min}(S)\right| \;\leq\; \|\bar{S} - S\| \;\leq\; c_3\ \epsilon\ \|S\| \;=\; c_3\ \epsilon\ \lambda_{max}(S), \tag{4.5}$$

while the largest eigenvalue is always relatively accurate. Because the eigenvalues of S are the squares of the singular values of W, the eigenvalue bounds reflect also that singular values of W smaller than $\sqrt{\epsilon}$ cannot be represented as eigenvalues of $\bar{S}$. For this reason we have to distinguish between the following two cases for the condition number $\kappa(W) = \sqrt{\kappa(S)} = \sqrt{\lambda_{max}(S)/\lambda_{min}(S)}$:

a. If $\kappa(W) \geq \frac{1}{\sqrt{\epsilon}}$, then $\lambda_{min}(S)$ is not computed accurately in floating point, and $\bar{\lambda}_{min}$ could be $O(\epsilon\ \lambda_{max}(S))$. In fact, our algorithm specifically sets any eigenvalues that are smaller than $\epsilon\ \lambda_{max}(\bar{S})$ equal to this threshold. In this case the ratio is bounded by:

$$\frac{\|W\|}{\sqrt{\bar{\lambda}_{min}}} \;\leq\; \sqrt{\frac{\lambda_{max}(S)}{\epsilon\ \lambda_{max}(S)}} \;=\; O\left(\frac{1}{\sqrt{\epsilon}}\right). \tag{4.6}$$

b. If $\kappa(W) \leq \frac{1}{\sqrt{\epsilon}}$, then from bound (4.5) and since for this case, it holds $1 + c_3\ \epsilon\ \kappa(S) = O(1)$, the ratio is bounded by:

$$\begin{aligned} \frac{\|W\|}{\sqrt{\bar{\lambda}_{min}}} \;&\leq\; \sqrt{\frac{\lambda_{max}(S)}{\lambda_{min}(S) + c_3\ \epsilon\ \lambda_{max}(S)}} \;=\; \sqrt{\frac{\kappa(S)}{1 + c_3\ \epsilon\ \kappa(S)}} \\ &=\; O(\sqrt{\kappa(S)}) \;=\; O(\kappa(W)). \end{aligned} \tag{4.7}$$

Combining the two cases, the bound for the ratio in (4.4) can be written as:

$$\|\delta Q\| \leq c_4\ \min\left(\ \epsilon\ \kappa(W),\ \sqrt{\epsilon}\ \right). \tag{4.8}$$

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

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

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &= \|I - \bar{\Lambda}^{-1/2}\bar{u}^T W^T W \bar{u}\bar{\Lambda}^{-1/2} + \delta Q^T Q + Q^T \delta Q\| + O(\delta Q^2) \\ &\leq \|I - \bar{\Lambda}^{-1/2}\bar{u}^T S \bar{u}\bar{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\|. \end{aligned} \tag{4.9}$$

From relations (4.1–4.3) and (4.9), the orthonormality of $\bar{u}$, and the fact that $\|Q\| = O(1)$, we have:

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\leq \|I - \bar{\Lambda}^{-1/2}\bar{u}^T \tilde{S}\bar{u}\bar{\Lambda}^{-1/2} + \bar{\Lambda}^{-1/2}\bar{u}^T(\delta\tilde{S} + \delta S)\bar{u}\bar{\Lambda}^{-1/2}\| + 2\|\delta Q\|\|Q\| \\ &\leq \|\bar{\Lambda}^{-1}\|\|\delta\tilde{S} + \delta S\| + 2\|\delta Q\|\|Q\| \\ &\leq c_3\ \epsilon\ \frac{\|S\|}{\bar{\lambda}_{min}} + 2\|\delta Q\|. \end{aligned} \tag{4.10}$$

From bounds (4.6) and (4.7), and because $\|S\| \leq \|W\|^2$ we have:

$$\begin{aligned} \|I - \bar{Q}^T\bar{Q}\| &\leq c_3\ \epsilon\ \min\left(\ \kappa(W)^2,\ \frac{1}{\epsilon}\ \right) + 2c_4\ \min\left(\ \epsilon\ \kappa(W),\ \sqrt{\epsilon}\ \right) \\ &\leq ct\ \min\left(\ \epsilon\ \kappa(W)^2,\ 1\right). \end{aligned} \tag{4.11}$$

$\square$

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

LEMMA 4.2.

1. If $\|I - Q^TQ\| \leq \alpha$, then $\|Q^TQ\| \leq \|Q\|^2 \leq 1 + a$.

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

PROPOSITION 4.3. *Assume that for the set of vectors W, with condition number $\kappa(W)$, $\|I - W^TW\| \leq \alpha < 1$. After one application of the SVQB algorithm, the resulting Q matrix has condition number:*

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

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

To avoid extensive use of complexity analysis symbols, we introduce the following intuitive notations $f < O(g) \Leftrightarrow f = o(g)$, $f > O(g) \Leftrightarrow f = \omega(g)$, and $f \geq O(g) \Leftrightarrow f = \Omega(g)$.

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:*

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

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

The case where $\kappa(W) \geq O(\frac{1}{\sqrt{\epsilon}})$ is much harder to bound because the eigenvalues of the $S = W^TW$ are much smaller than ϵ and numerical error dominates. However, a bound can be given with some relatively weak assumptions. We first need the following lemma.

LEMMA 4.5. *Let $A = \begin{bmatrix} G & \sqrt{\epsilon}C \\ \sqrt{\epsilon}C^T & B \end{bmatrix}$, where $\|C\| = O(1)$, G is a symmetric matrix with eigenvalues $\lambda_g = O(1)$, and B is symmetric matrix with eigenvalues $0 < \lambda_b < O(1)$. Then, if there is a $\lambda_b \ll O(1)$, there exists an eigenvalue of A, λ, such that $|\lambda_b - \lambda| = O(\epsilon)$.*

Proof. Let (u_b, λ_b) an eigenpair of B, and define the vector: $u = [u_g^T u_b^T]^T$, where u_g is chosen to satisfy the first block row set of eigenvalue equations for A: $u_g = \sqrt{\epsilon}(\lambda_b I - G)^{-1}Cu_b$. Applying A on u we have: $Au = \lambda_b \begin{bmatrix} u_g \\ u_b \end{bmatrix} + \begin{bmatrix} 0 \\ \epsilon\ C^T(\lambda_b I - G)^{-1}Cu_b \end{bmatrix}$. From our assumption, $\lambda_b \ll O(1)$, which implies $\|(\lambda_b I - G)^{-1}\| = O(1)$, and thus $\|\epsilon\ C^T(\lambda_b I - G)^{-1}Cu_b\| = O(\epsilon)$. This means that the matrix A is only $O(\epsilon)$ away from a matrix that has (u, λ_b) as an eigenpair. $\square$

THEOREM 4.6. *Consider the same assumptions of theorem 4.4. If we assume, in addition, that (a) singular values of W that are smaller than $O(\sqrt{\epsilon})$ are well separated* *from the rest which are $O(1)$, and (b) the smallest singular value of $\bar{Q}$ is not appreciably smaller than the one of W, then:*

$$\text{if } \kappa(W) \geq O(\frac{1}{\sqrt{\epsilon}}), \qquad \kappa(\bar{Q}) \leq O(\sqrt{\epsilon}\ \kappa(W))).$$

Proof. Throughout this proof we assume the notation of theorem 4.1. We introduce $Q_u = W\bar{u}$, the unscaled version of $\bar{Q}$. The error in the floating point representation of Q_u is δQ_u with $\|\delta Q_u\| \leq \epsilon \|W\| = O(\epsilon\sqrt{\lambda_{max}})$, where $\lambda_{max} \equiv \lambda_{max}(S)$. We now rewrite relation (4.9) using this unscaled Q_u:

$$\bar{Q}^T\bar{Q} = \bar{\Lambda}^{-1/2}\left(\bar{u}^T S \bar{u} + \delta Q_u^T Q_u + Q_u^T \delta Q_u\right)\bar{\Lambda}^{-1/2}. \tag{4.12}$$

Let $\Delta Q = \delta Q_u^T Q_u + Q_u^T \delta Q_u$, and observe that $\|\Delta Q\| \leq 2\|\delta Q_u\|\|Q_u\| = O(\epsilon\lambda_{max})$. Because $\bar{u}$ are orthonormal vectors, $Q_u^T Q_u = \bar{u}^T S \bar{u}$ has the same eigenvalues as S. We will show that scaling from left and right with $\bar{\Lambda}^{-1/2}$ reduces $\kappa(Q_u^T Q_u + \Delta Q)$ by a factor of $O(\epsilon)$. Thus, we focus on the exact eigenvalues of the exact product $\bar{Q}^T\bar{Q}$ of the computed vectors $\bar{Q}$.

Since $\kappa(W) \geq O(\frac{1}{\sqrt{\epsilon}})$, there is a set of eigenvalues of S that are smaller than ϵ. Standard eigensolvers view these eigenvalues as numerical multiplicities, and thus they produce a basis for their subspace, rather than their exact eigenvectors. On the other hand, the separation assumption (a) states that the approximations for the rest of the eigenvectors have no mixing with these almost degenerate ones. If we order the eigenvalues of $\bar{S}$ in decreasing size, we can partition the corresponding eigenvectors into two groups, a "good" group u_g and a "bad" group u_b:

$$\begin{array}{ll} u_g = \bar{u}_i, & \text{for } i = 1, \ldots, l \\ u_b = \bar{u}_i, & \text{for } i = l+1, \ldots, m \end{array}, \text{ and } \bar{\lambda}_l > O(\epsilon\lambda_{max}).$$

The fact that there is no mixing between the good and the bad groups translates to $u_g^T u_b = O(\epsilon)$. Because of the separation and the fact that $\|\bar{S} - S\| = O(\epsilon\lambda_{max})$, the eigenvectors of $\bar{S}$ produced by eigensolvers are accurate approximations of the eigenspaces of S within each of the two groups. Therefore, the eigenvalues of $\bar{u}_g^T S \bar{u}_g$ are $\lambda_i + O(\epsilon\lambda_{max})$, $i = 1, \ldots, l$, and the eigenvalues of $\bar{u}_b^T S \bar{u}_b$ are $\lambda_i + O(\epsilon\lambda_{max})$, $i = l+1, \ldots, m$. Thus, we can rewrite the unscaled part of equation (4.12) as:

$$\bar{u}^T S \bar{u} + \Delta Q = \begin{bmatrix} G & 0 \\ 0 & B \end{bmatrix} + O(\epsilon\lambda_{max}), \tag{4.13}$$

where $G = \bar{u}_g^T S \bar{u}_g$, $B = \bar{u}_b^T S \bar{u}_b$, and the perturbation $O(\epsilon\lambda_{max})$ applies to all the elements.

We now focus on $\bar{Q}^T\bar{Q}$ by scaling the matrix in equation (4.13) from left and right by $\bar{\Lambda}^{-1/2}$. In our algorithm, any $\bar{\lambda}_i < \epsilon\lambda_{max}$ is set equal to $\epsilon\lambda_{max}$, which gives $\bar{\Lambda}$ the following form:

$$\bar{\Lambda} = diag(\bar{\lambda}_1, \ldots, \bar{\lambda}_l, \epsilon\lambda_{max}, \ldots, \epsilon\lambda_{max}).$$

Using these scaling factors we obtain:

$$\bar{Q}^T\bar{Q} = \begin{bmatrix} G' & C \\ C^T & \frac{1}{\epsilon\lambda_{max}}(B + O(\epsilon\lambda_{max})) \end{bmatrix}, \tag{4.14}$$

where G' is $G + O(\epsilon\lambda_{max})$ scaled with the good eigenvalues of $\bar{\Lambda}$, and the i_{th} row of C are the scaled perturbations $O(\epsilon\lambda_{max})$:

$$C_{i,j} = \frac{O(\epsilon\lambda_{max})}{\sqrt{\epsilon\lambda_{max}\bar{\lambda}_i}} = O\left(\sqrt{\epsilon\lambda_{max}}\right), \quad j = l+1, \ldots, m.$$

From the theorem hypothesis, $\bar{\lambda}_i = O(1)$, $i = 1, \ldots, l$, and thus the eigenvalues of $G + O(\epsilon\lambda_{max})$ are $O(1)$. As a result, the eigenvalues of G' are also $O(1)$. On the other hand, the minimum eigenvalue λ_b of $B + O(\epsilon\lambda_{max})$ could be much larger than λ_{min}. We consider the worst case scenario, where $\lambda_b = \lambda_{min} + O(\epsilon\lambda_{max})$. The order notation covers also the case of no perturbation. From this, and from assumption (b), the minimum eigenvalue of $\frac{1}{\epsilon\lambda_{max}}(B + O(\epsilon\lambda_{max}))$ is:

$$\lambda_b' = \frac{\lambda_{min} + O(\epsilon\lambda_{max})}{\epsilon\lambda_{max}}.$$

We distinguish two cases. First, if the perturbation in order notation above is close to $\epsilon\lambda_{max}$, the eigenvalues of $\bar{Q}^T\bar{Q}$ are all $O(1+\sqrt{\epsilon})$. Therefore, its condition number is close to one, and the theorem holds. Second, we assume that the perturbation is much smaller than $\epsilon\lambda_{max}$, and therefore $\lambda_b' \ll 1$. In this case, the assumptions of lemma 4.5 are satisfied for the matrix in equation (4.14), and thus the smallest eigenvalue of $\bar{Q}^T\bar{Q}$ is equal to $\lambda_b' + O(\epsilon\lambda_{max})$. Substituting these eigenvalues in the condition number $\kappa(\bar{Q}^T\bar{Q}) = \kappa(\bar{Q})^2$ completes the proof:

$$\kappa(\bar{Q})^2 = \frac{1}{\frac{1+O(\epsilon\ \kappa(S))}{\epsilon\ \kappa(S)} + O(\epsilon\lambda_{max})} = \frac{\epsilon\ \kappa(S)}{1 + O(\epsilon\ \kappa(S)) + O(\epsilon^2\ \kappa(S))} = O(\epsilon\ \kappa(S)).$$

□

In general, one could circumvent the separation assumption of the theorem by choosing a value of $\tau > 1$ and the "good" eigenvalues $\bar{\lambda}_l > O(\tau\epsilon\lambda_{max})$, to ensure $O(1)$ separation between the good and bad groups. The existence of such separator is guaranteed by the small number (m) of eigenvalues that span the interval between 0 and $\lambda_{max} = O(1)$. Although, increasing τ weakens the theoretical bound of the theorem, our numerical experiments have shown a consistent reduction of $\sqrt{\epsilon}$ in the condition number of $\bar{Q}$.

The second assumption of the theorem is necessary since, in general, we cannot guarantee that the smallest singular value of $\bar{Q}$ is not smaller than $\sigma_{min}(W)$ or even zero. However, such a situation is rather improbable for two reasons. First, one expects the SVQB process to produce better conditioned, if not orthogonal, $\bar{Q}$ vectors. Second, since computing $\bar{Q}$ involves a matrix multiplication and a scaling operation, one expects the numerical error to be in linearly independent directions, or to be a small relative error. If the elements of the W and $\bar{u}$ vectors are $O(1)$, numerical error is expected to mount close to ϵ, and thus cancellations should not reduce the lowest eigenvalue of S, which is much smaller than ϵ. If there is a special structure of either W or $\bar{u}$ with elements smaller than ϵ, numerical error will affect the computations with those elements in a relative sense, and thus the smallest eigenvalue may not be reduced significantly.

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 akin to the behavior of the iterative GS method [13, 18, 7]. However, unlike block GS without internal re-orthogonalization, SVQB guarantees an improvement of $\sqrt{\epsilon}$ at every iteration, even for $\kappa(W) > 1/\sqrt{\epsilon}$.

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

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 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 [14, 13, 2]. 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 $\epsilon = 2.2\text{e}{-16}$. 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.

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
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 4.1

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

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
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 4.2

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

	SVQB			CholQR	GS	MGS
Iteration	$\kappa(Q_u)$	$\kappa(\bar{Q})$	$\kappa(Q'_u)$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$	$\kappa(\bar{Q})$
1	2e+49	3e+41	4e+34	4e+34	2e+01	1+ϵ
2	4e+34	7e+25	4e+19	4e+19	1+ϵ	–
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+ϵ	1+ϵ	1+ϵ	–	–

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

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 the implicit normalization in SVQB does not guarantee normality for ill-conditioned problems, we print also the condition number of the unscaled vectors Q_u, $\kappa(Q_u)$, and the condition number of the same vectors after explicitly scaling them by their norms, $\kappa(Q'_u)$. Note that after each iteration $\kappa(Q_u)$ is equal to the condition number of the vectors before the application of SVQB. For example, after the first iteration $\kappa(Q_u) = \kappa(W)$.

The results in all tables confirm theorems 4.4 and 4.6. The application of one step of SVQB reduces the condition number of a set of vectors at least by $\sqrt{\epsilon}$. 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). When the condition number is smaller than $1/\sqrt{\epsilon}$, the reduction obeys closely the bound in theorem 4.4.

As expected, the CholQR method behaves similarly to SVQB (table 4.3). In some cases, the orthogonality of the CholQR is inferior to that of the 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 Choleski decomposition is applied on a numerically positive definite matrix.

As discussed earlier, block GS without re-orthogonalization for each vector is not effective for large condition numbers. In such cases, GS may offer no improvement between successive iterations (see first GS iteration in table 4.2), or it may require many iterations to produce a set with 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 re-orthogonalization. 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 the 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. Orthogonalizing against V. The above theory and examples establish that SVQB is a competitive technique for orthogonalizing a set of vectors W among themselves. In practice, however, full orthogonality among W vectors may not be necessary in all iterations of algorithm GS-SVQB. The reason is the interplay between the block GS and the SVQB at the first and second step of the algorithm respectively. The GS destroys the orthogonality of W, and the SVQB procedure may destroy the orthogonality against V.

LEMMA 5.1. *Let $W = GS(V, W')$ be the set of vectors resulting after the first step of the GS-SVQB(V, W') algorithm, and assume that $\|V^TW\| \leq \mu \ \|W\|$. Let $\bar{Q} = SVQB(W)$ be the result of the second step of the GS-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 the error bound on δQ in (4.4), and the bounds for the two different cases in (4.6) and (4.7), we have: $\|V^T\bar{Q}\| = \|V^TW\bar{u}\bar{\Lambda}^{-1/2} + V^T\delta Q\| \leq \mu\|W\|\|\bar{\Lambda}^{-1/2}\| + O(\|\delta Q\|) = O\left((\mu+\epsilon)\min(\kappa(W), 1/\sqrt{\epsilon})\right)$. ☐

The above lemma states that even when GS produces W that is exactly orthogonal to V, i.e., $\mu = 0$, the SVQB procedure may destroy that orthogonality up to a maximum of $\sqrt{\epsilon}$. Although lemma 5.1 is given only for the SVQB, similar results apply if any other method for QR decomposition is used. In our numerical experiments, we have observed consistently the bound $(\mu \ \kappa(W))$ for the loss of orthogonality against V, regardless of the method used to orthogonalize the W vectors among themselves. This is an additional reason diminishing the importance of using the more accurate MGS, instead of SVQB, in our procedure.

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

LEMMA 5.2. *Let $V^TV = I$, and W be a set of normal vectors with $\|W^TW - I\| \leq \nu$. Let $Q = (W - VV^TW)D^{-1/2}$ be the normalized result of the block GS, where D is a diagonal matrix with the squares of the normalizing norms of the vectors. Assume that there is no floating point error in computing Q. If $\|V^TW\| = \delta < 1$, then:*

$$\|Q^TQ - I\| = O\left(\frac{\nu+\delta^2}{1-\delta^2}\right).$$

Proof. If we let $S = V^TW$, we have $D_{ii} = w_i^Tw_i - w_i^Tvv^Tw_i = 1 - e_i^TS^TSe_i$. Note that for all diagonal elements of S^TS, it holds $e_i^TS^TSe_i \leq \|S^TS\| = \delta^2 < 1$. As a result, for all diagonal elements of D, we have : $D_{ii} > 1 - \delta^2$. This holds for the 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:

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

☐

The lemma states that if the vectors W are not orthogonal enough to V (i.e., if $\|V^TW\| > \sqrt{\epsilon}$), they will also lose their orthogonality among each other after the GS step. This lemma applies to any procedure that may be used instead of GS.

6. The iterative GS-SVQB algorithm. Theorems 4.1, 4.4 and 4.6 and the well studied behavior of the GS algorithm suggest that the GS-SVQB should be applied iteratively. Figure 6.1 shows four possible implementations, based on which of the two steps (GS/SVQB) is carried out iteratively.

Algorithm 1: **repeat** $W = \mathrm{GS}(V, W)$ $W = \mathrm{SVQB}(W)$ **until** $(\|W^TW - I\|, \|V^TW\| = O(\epsilon))$	Algorithm 2: **repeat** **repeat** $W = \mathrm{GS}(V, W)$ **repeat** $W = \mathrm{SVQB}(W)$ **until** $(\|W^TW - I\|, \|V^TW\| = O(\epsilon))$
Algorithm 3: **repeat** **repeat** $W = \mathrm{GS}(V, W)$ $W = \mathrm{SVQB}(W)$ **until** $(\|W^TW - I\|, \|V^TW\| = O(\epsilon))$	Algorithm 4: **repeat** $W = \mathrm{GS}(V, W)$ **repeat** $W = \mathrm{SVQB}(W)$ **until** $(\|W^TW - I\|, \|V^TW\| = O(\epsilon))$

FIG. 6.1. *Four possible iterative implementations of the GS-SVQB algorithm. The outer loop is repeated until W becomes 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.*

Algorithm 1 is a straight forward iterative implementation of GS-SVQB, but it does not take into account the fact that SVQB and GS may require different number of steps to orthogonalize W among themselves and against V respectively. Forcing both methods to take the same number of steps may cause a lot of work to be wasted. This applies especially to the GS, which performing twice is usually enough [18]. Moreover, because usually the size of V is much larger than that of W, we try to minimize the number of times that GS has to be repeated.

Algorithms 2 and 3 apply GS repeatedly on W and thus they also are inappropriate for similar reasons. In addition, although the resulting W might be orthogonal to V after the GS, applying the SVQB on W destroys that orthogonality by as much as $\epsilon\kappa(W)$ (from lemma 5.1). This completely obviates more than one GS application.

Algorithm 4 seems the most appropriate choice for an iterative implementation of GS-SVQB. An additional argument in favor of this choice is that GS destroys the orthogonality among the W vectors only by $O(\|V^TW\|^2)$ (see lemma 5.2). The presence of the square often alleviates this loss, especially when the GS has already been applied once.

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. We observe that the iterative application of SVQB in the inner loop should not necessarily produce a fully orthonormal set W. Lemmas 5.1 and 5.2 suggest that the two steps, GS and SVQB, must be balanced so that work is not wasted in obtaining full orthogonality when this is not needed. They also suggest that both $\kappa(W)$ and $\|V^TW\|$ must be kept comparably small.

We start by determining the number of times the inner repeat loop needs to be executed. First, we seek the conditions under which the final outer iteration does not require any SVQB applications. In this case, one or more outer iterations have been performed already, because W must be orthonormal. In addition, orthonormality of W must not be destroyed by the GS at this last outer iteration. Thus, lemma 5.2 implies that before this GS it holds:

$$\|V^TW\| < \sqrt{\epsilon}. \tag{6.1}$$

Although, this condition can be checked inexpensively as a by-product of GS, the

$\kappa(W)$ cannot be known exactly without additional synchronization and computation. To decide whether to skip the final SVQB, we can instead use theorem 4.1 and the singular values of W obtained in the last SVQB of the previous outer iteration. According to the theorem, the SVQB that produced the latest orthonormal set W must have been applied to a previous set with condition number $O(1)$. Therefore, besides (6.1), we should also test for:

$$\kappa(W_{\text{before last SVQB}}) = O(1). \tag{6.2}$$

Next, we examine the number of inner iterations required at any step of algorithm 4. Because applying GS twice is usually enough [18], we claim that the inner iteration should produce a set of W with at least $\kappa(W) = O(1)$. If the inner iteration produced W with $\kappa(W) > O(1)$, then at the next outer iteration, the orthogonality that the next GS achieves against V will be on the order of $\epsilon\ \kappa(W)$ (lemma 5.1). Thus, a third outer iteration would be necessary, which could have been avoided. As a result, we only need to distinguish between those cases for which the inner iteration should produce a completely orthonormal set W or one with $\kappa(W) = O(1)$. For this, we need to take into account the reorthogonalization requirement of GS.

To test whether the GS procedure requires reorthogonalization, we have used a popular test due to Daniel et al. [7]. Reorthogonalization is needed whenever the norm of W after GS becomes less than .7 times its original norm before the GS. In our algorithm, the SVQB also can cause loss of orthogonality against V. Let Q be the set resulting from GS on W, and consider the following cases.

Assume that Daniel's test does not require reorthogonalization. If $\kappa(Q) = O(1)$, then one application of the SVQB will produce a fully orthonormal Q, without destroying its orthogonality against V. If $\kappa(Q) > O(1)$, then SVQB will destroy the orthogonality against V and orthogonalization must be repeated. The SVQB should be carried out until the $\kappa(Q)$ becomes $O(1)$, so that the SVQB at the following outer iteration does not affect V^TQ.

Next, assume that Daniel's test requires reorthogonalization. If $\kappa(Q) = O(1)$, still we need to apply SVQB once. This will probably orthonormalize Q and if orthogonality against V is relatively good, a final SVQB may not be needed at the next outer step. If $\kappa(Q) > O(1)$, as in the previous case, we iterate the SVQB until the $\kappa(Q)$ becomes $O(1)$.

At this point we can summarize our above observations and analysis into the following algorithm. The algorithm simplifies our previous discussion by noting that repeat-until loops execute always at least once. Thus, it is sufficient that the 'until' condition for SVQB checks whether the condition number of the resulting Q is $O(1)$. Note, that the condition number of the set W is computed from the eigenvalues of the matrix $S = W^TW$ in the SVQB algorithm. Therefore, it corresponds to the set W before the application of the SVQB. Because a bound on the resulting $\kappa(Q)$ can be inferred through theorems 4.4 and 4.6, the until condition checks if the condition number before the last SVQB was less than $1/\sqrt{\epsilon}$. Finally, note that $\|V^TW\|$ can be computed during the GS, and it corresponds to the overlap of the two matrices before the GS.

Algorithm 6.1. $Q = \text{iGS-SVQB}(V, W)$ *(iterative GS-SVQB method)*
$i = 1$
$W^{(0)} = W$
$\kappa_{\text{last}} =$ *large number*

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

7. Further optimizations. Because of its block character, the iGS-SVQB algorithm iterates on the whole set of vectors W. Often, this may be wasteful because individual vectors in W may be almost orthogonal to V and also to any other vector in W. Clearly, further reorthogonalizations should exempt these vectors, performing computations on a smaller block.

Specifically, after the GS phase, we can easily separate those vectors that do not need reorthogonalization (good vectors) and those that need it (bad vectors), $W = [W_g W_b]$. Unfortunately, not all of the good vectors will turn out to have large singular values in the SVQB phase. Moreover, the SVQB phase will mix all these good and bad vectors and the identity of the good ones will disappear.

To solve this problem, first we perform an eigenvalue decomposition of $S_g = W_g^T W_g$, for the W_g vectors alone. This, in turn, will subdivide the good group into two groups $W_g = [W_{gg} W_{gb}]$. The first group, which we call good-good group, consists of all the singular vectors W_{gg} corresponding to large singular values of S_g. Therefore, these W_{gg} vectors are orthogonal both to V and to each other and can be appended to V in future iterations. Note also that W_{gg} has at least one vector. The second group W_{gb}, which we call good-bad, are all the vectors that did not need reorthogonalization originally, but because they were too close to other vectors in W_g they may have lost their orthogonality against V as well. Therefore, the W_{gb} vectors should be combined with the W_b vectors, and an SVQB should be applied on $[W_{gb} W_b]$.

Still, a few more subtleties need to be resolved. First note that one synchronization is still enough, since the computation of $S_b = W_b^T W_b$ can be computed at the same time as S_g. Second, neither of W_{gb} and W_b vector sets are orthogonal to W_{gg}, and moreover the W_b set has not been orthogonalized against W_{gg} yet. If the angles between W_b and W_{gg} are very small, extra iterations of the expensive, outer loop of the iGS-SVQB algorithm may be needed. Therefore, we have to perform at least one orthogonalization of the $[W_{gb} W_b]$ against W_{gg}. Interestingly, these orthogonalizations can be performed on $m \times m$ matrices, by orthogonalizing the short singular vectors instead of the corresponding W vectors. In this way, the computation of $S_b' = [W_{gb} W_b]^T [W_{gb} W_b]$ can also be performed inexpensively using $m \times m$ matrices, and without additional synchronization points.

The theory supporting the above optimizations, as well as the details of the implementation fall beyond the scope of this paper, and will be the focus of an upcoming technical report. In summary, we can say that at each iteration, our iGS-SVQB implementation identifies orthonormal vectors in W that have been orthonormalized effectively, so that no further computations are wasted on them.

When the number of vectors in W is very large, applying the SVQB method on the full block may not always be efficient or numerically stable. For example, the better cache performance of the SVQB may not be enough to counterbalance the increase in the arithmetic over the GS, or even the solution of the $m \times m$ SVD problem. Moreover, in most situations increasing the block beyond an optimal size may even decrease cache performance. Finally, 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 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 increases linearly with p, and a larger percentage of the computation is spent on the GS procedure. In the extreme case of $b = 1$, the algorithm reduces to the classical GS method, while for $b = m$ the algorithm is simply the iGS-SVQB method. With bGS-SVQB, we expect to identify a range of block sizes for which the cache performance is optimal, while the synchronization requirements are not excessive. Finally note that since bGS-SVQB is based on the iGS-SVQB, it will iterate to guarantee orthogonality to any user specified level.

8. Timing experiments. We have extensively tested our implementation of the bGS-SVQB method 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.

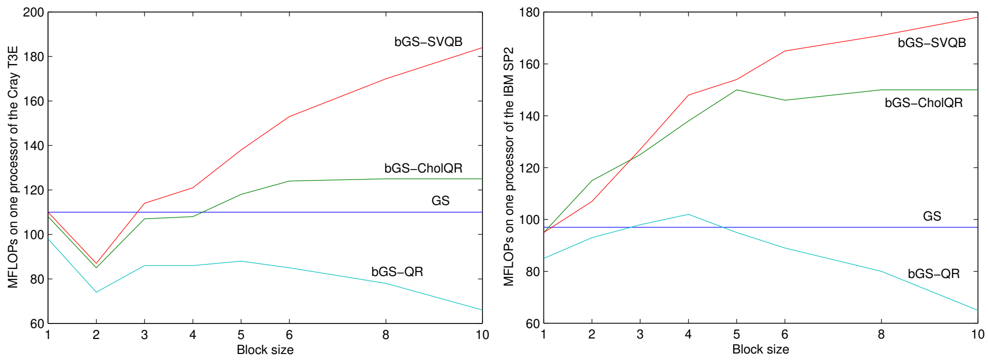

FIG. 8.1. *Single node MFLOPs as a function of block size for the four methods. The methods orthonormalize thirty vectors of dimension 500000. Left graph depicts Cray T3E results. Right graph depicts IBM SP2 results.*

Finally, we compare against with the computationally similar method CholQR. The method is denoted as bGS-CholQR, and uses the Choleski decomposition provided in the ScaLAPACK library. Note that all "bGS-Method" variants reduce to GS when the block size is equal to one.

All algorithms have been implemented in Fortran 90 using the MPI interface, and run on the Cray T3E 900 and the IBM SP2 parallel computers at NERSC National lab. 256 MB of memory are available on each node of both 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 one, while the SP2 processors are slightly faster than the SP2 ones. On both parallel platforms we link with the MPICH libraries, and we make use of the machine optimized libraries for ScaLAPACK and BLAS.

Our first numerical example was chosen to be easily reproducible as a set of thirty Krylov vectors of a diagonal matrix. We use the 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 resulting 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, by applying different orthogonalization variants and for a range of block sizes: bGS-Method(V, W, b). Note that, initially, most of the computation is spent on the "Method", while as more blocks get orthonormalized GS takes over.

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 CG rate is constant regardless of block size. The single-node performance of the bGS-QR does not scale 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 the SVQB and CholQR allows them to outperform GS significantly, even for small blocks of 8-10 vectors. The Choleski back-solve implementation seems to exploit better the architecture of the SP2 than of 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 the GS because only one of the two phases involves BLAS 3 kernels.

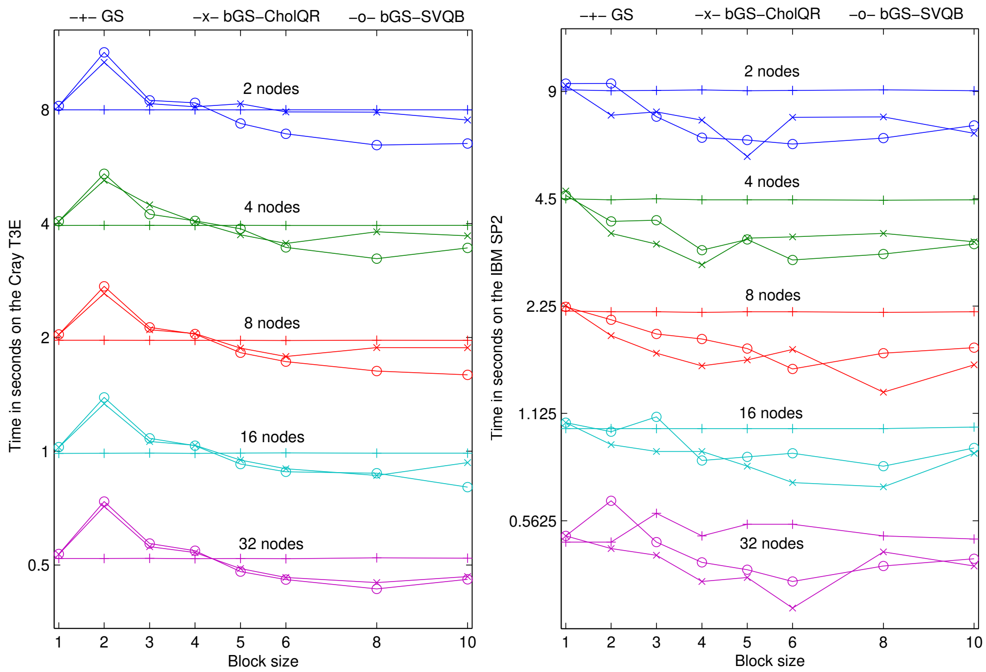

FIG. 8.2.

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 the 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 large number of processors, despite smaller local problem sizes. On the T3E, bGS-SVQB is 20% faster than the GS method, and on the SP2, it is more than 25% faster. Note that although bGS-CholQR is 35% faster than GS on the SP2, this impressive performance does not carry over to the T3E.

Finally, we test a different scalability. We measure 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 of the discretized Laplacean on a 3 dimensional cube. Every processor holds a $32 \times 32 \times 32$ 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 6 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

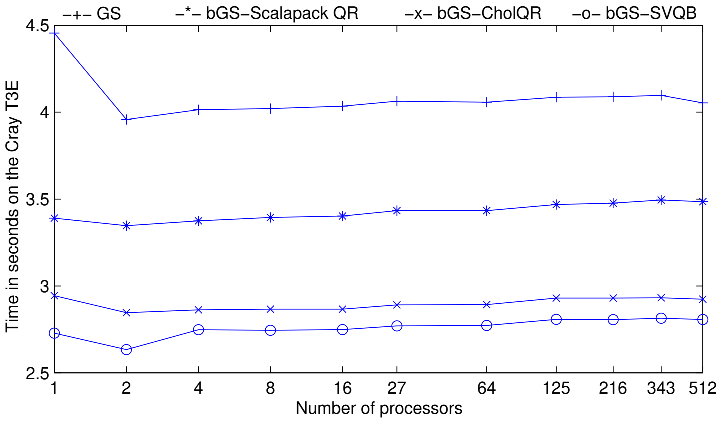

FIG. 8.3.

extremely fast T3E network. However, the effects are more apparent on the GS and the bGS-QR, as their curves increase faster than the respective ones for bGS-CholQR and bGS-SVQB. These differences are expected to be significant if the experiment is run on a high-latency network cluster. Finally, we observe that on this problem the bGS-SVQB is more than 30% faster than the GS (an improvement over the previous numerical problem). This is especially important because the GS is one of the most computationally efficient methods.

9. Conclusions. We have introduced and analyzed a new orthonormalization method, called SVQB, that computes the orthonormal basis from the right singular vectors of a matrix. 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 if the condition number of the vector set is originally $\kappa(W) < O(1/\sqrt{\epsilon})$, the departure from the orthonormality of the resulting vector set is $\|I - QQ^T\| < O(\epsilon\kappa(W))$. Moreover, when $\kappa(W) > O(1/\sqrt{\epsilon})$, and under some weak assumptions, the algorithm guarantees that each iteration reduces the condition number of the set by $\sqrt{\epsilon}$. We have also considered the problem of incremental orthonormalization where a block of vectors is orthonormalized against a previously orthonormal set of vectors with GS, and among itself with SVQB. The independent application of each of the methods destroys the orthogonality produced by the other method. We have provided bounds that describe this numerical interdependence and have used them to balance the work performed by each of the two orthogonalization phases of 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 and the IBM SP2 have shown that our method improves the performance of other orthonormalization alternatives.

10. Acknowledgements. This work was supported by the College of William and Mary and the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

REFERENCES

[1] Å. Björck. Solving linear least squares problems by gram-schmidt orthogonalization. *BIT*, 7:1–21, 1967.

[2] Å. Björck. Numerics of gram-schmidt orthogonalization. *Linear Algebra and its Applications*, 198:297–316, 1994.

[3] Å. Björck and C. Paige. Loss and recapture of orthogonality in the modified gram-schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13(1):176–190, 1992.

[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, 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, 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(136):772–795, October 1976.

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

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

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

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, MD 21211, 1989.

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

[13] W. Hoffman. Iterative algorithms for gram-schmidt orthogonalization. *Computing*, 41:335–348, 1989.

[14] W. Jalby and B. Philippe. Stability analysis and improvement of the block gram-schmidt algorithm. *SIAM J. Sci. Statist. Comput.*, 12(5):1058–1073, 1991.

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

[16] P. O. Lödwin. *J. Chem. Phys.*, 18:365, 1950.

[17] P. O. Lödwin. *Adv. Quant. Chem.*, 23:84, 1992.

[18] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA, 1998.

[19] Yousef Saad. *Iterative methods for sparse linear systems*. PWS Publishing Company, 1996.

[20] H. C. Schweinler and E. P. Wigner. *J. Math. Phys.*, 11:1693, 1970.

[21] J. P. Singh, D. E. Culler, and A. Gupta. *Parallel Computer Architecture. A Hardware/Software Aproach*. Morgan Kaufmann Publishers, Inc, 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 *1999 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 2920–2926. CSREA Press, 1999.

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