LAPACK WORKING NOTE 167: SUBSET COMPUTATIONS WITH THE MRRR ALGORITHM[§]

OSNI A. MARQUES[*], BERESFORD N. PARLETT[†], AND CHRISTOF VÖMEL[‡]

Technical Report UCB//CSD-05-1392
Computer Science Division
UC Berkeley

Abstract. The main advantage of inverse iteration over the QR algorithm and Divide & Conquer for the symmetric tridiagonal eigenproblem is that subsets of eigenpairs can be computed at reduced cost.

The MRRR algorithm (MRRR = Multiple Relatively Robust Representations) is a clever variant of inverse iteration without the need for reorthogonalization. STEGR, the current version of MRRR in LAPACK 3.0, does not allow for subset computations. The next release of STEGR is designed to compute a (sub-)set of k eigenpairs with $\mathcal{O}(kn)$ operations.

Because of the special way in which eigenvectors are computed, MRRR subset computations are more complicated than when using inverse iteration. Unlike the latter, MRRR sometimes cannot ignore the unwanted part of the spectrum.

We describe the problems with what we call 'false singletons'. These are eigenvalues that appear to be isolated with respect to the wanted eigenvalues but in fact belong to a tight cluster of unwanted eigenvalues. This paper analyzes these complications and ways to deal with them.

Key words. Multiple relatively robust representations, numerically orthogonal eigenvectors, symmetric tridiagonal matrix, subset computation, false singleton.

AMS subject classifications. 15A18, 15A23.

Monday 26$^{\text{th}}$ September, 2005, 10:54am

1. Introduction. The algorithm of Multiple Relatively Robust Representations (MRRR) [13, 14, 15, 16, 17, 8] is a sophisticated variant of inverse iteration for a symmetric tridiagonal matrix T [7, 11] that avoids explicit orthogonalization even for tightly clustered eigenvalues.

The first release of MRRR in the current version 3.0 of LAPACK [1] only computed the full set of eigenpairs. Here, we describe the inclusion of a subset functionality into the MRRR algorithm, similar to the existing one in inverse iteration. The ability to compute subsets of eigenpairs at reduced cost represents a significant advantage over the QR algorithm [12] and the Divide & Conquer method [4, 5, 10] which cannot take advantage of the subset situation.

In order to compute orthogonal eigenvectors without having to resort to Gram-Schmidt orthogonalization, MRRR takes note of any clusters of close eigenvalues. For each of these clusters, MRRR computes an appropriate relatively robust representation $LDL^T = T - \sigma I$ from which it computes the eigenvectors, see [14, 15, 16, 17, 8].

When only a subset of eigenvalues is desired, the situation of an end of the wanted subset being part of a different cluster can pose considerable difficulties. An eigenvalue at one end of the set of wanted eigenvalues might appear isolated within in the set of

*Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50F-1650, Berkeley, CA 94720, USA. oamarques@lbl.gov

†Mathematics Department and Computer Science Division, University of California, Berkeley, CA 94720, USA. parlett@math.berkeley.edu

‡Computer Science Division, University of California, Berkeley, CA 94720, USA. voemel@eecs.berkeley.edu

§This work has been supported by a grant from the National Science Foundation (Cooperative Agreement no. ACI-9619020).

wanted eigenvalues but belong to a cluster of close but unwanted eigenvalues. We call such an eigenvalue a *false singleton* because it is only a singleton among the wanted part, not the whole spectrum. We describe the difficulties that such a false singleton poses to MRRR and how to overcome them.

Section 2 compares standard inverse iteration and the MRRR algorithm. We emphasize differences in the vector computation that can lead to problems for MRRR when only a subset of the spectrum is known.

In Section 3, we discuss four different issues when the MRRR algorithm is applied to subset computations in a way that mimics standard inverse iteration. We also describe how to address theses issues.

In Section 4, we compare our subset approach to a different one that is used in the parallel MRRR algorithm [3, 2]. There, one not only works on the wanted subset of eigenpairs but embeds the subset computation into one for an isolated superset of the spectrum. This idea guarantees orthogonality between eigenvectors belonging to non-overlapping subsets of eigenvalues and is essential for the parallel computation.

We discuss the overhead of this approach which makes it unattractive for the sequential LAPACK algorithm.

2. The differences in computing eigenvectors by inverse iteration and MRRR. In this section, we describe the algorithmic differences in computing an eigenvector of a symmetric tridiagonal matrix T by standard inverse iteration (LAPACK's STEIN) and by the MRRR algorithm (STEGR).

Standard inverse iteration embodied in LAPACK's STEIN treats subset computations in the same way as the full spectrum case. All eigenvectors are computed from the matrix T and its eigenvalues. When an eigenvalue is too close to its neighbors, it is perturbed by a small relative amount. Then one step of inverse iteration with a random right-hand side is performed. The resulting vector is kept orthogonal to all previously computed eigenvectors that belong to close eigenvalues. It is scaled and used as new right-hand side until the iteration converges. See [11, 7] for details.

In contrast, STEGR computes a shift σ defining $T-\sigma I = LDL^T$ with the following properties:

- Small relative changes in entries of L and D cause small relative changes in λ. We call such an (L, D) a Relatively Robust Representation (RRR) for λ.
- The (local shifted) eigenvalue approximation $\hat{\lambda}, |\lambda - \hat{\lambda}| = \mathcal{O}(\epsilon|\lambda|)$ is of about the same size as its distance to the next closest neighbor. Precisely, let $\text{gap}(\hat{\lambda}) = \min\left\{|\hat{\lambda} - \mu| : \lambda \neq \mu, \mu \in \text{spectrum}(LDL^T)\right\}$, then the shift σ is chosen such that $\text{relgap}(\hat{\lambda}) = \text{gap}(\hat{\lambda})/|\lambda|$ is larger than a threshold.

Then the key step is to find a vector v with a small *relative* residual

$$\|(LDL^T - \hat{\lambda}I)v\| = \mathcal{O}(n\epsilon|\lambda|). \tag{2.1}$$

The reward is revealed by the classical gap theorem [6, 12]. Let z denote the true eigenvector, then

$$|\sin \angle(v, z)| \leq \frac{\|(LDL^T - \hat{\lambda}I)v\|}{\text{gap}(\hat{\lambda})} = \frac{\mathcal{O}(n\epsilon)}{\text{relgap}(\hat{\lambda})}. \tag{2.2}$$

We note that the shift σ is chosen incrementally. If the relative gaps of a group of eigenvalues are too small, then they can be increased by shifting the origin close to one end of the group. Furthermore, the procedure can be repeated for clusters within clusters of close eigenvalues.

This procedure is best represented by a rooted tree [16]. Each node of the graph represents the RRR for a group Γ of eigenvalues. The root node of this *representation tree* is the initial representation that is an RRR for all the wanted eigenvalues, each leaf corresponds to a a 'singleton', that is a (shifted) eigenvalue with a large relative gap.

We now come to the central part of this exposition, the differences in the two algorithms when computing the eigenvector. A summary of these differences is given in Table 2.1.

	STEIN	STEGR
Matrix	tridiagonal T	(shifted) LDL^T, twisted $N_r \Delta_r N_r^T$
Right-hand side	(scaled) random vector b	scaled column of the identity (corresp. to large entry of true eigenvector)
Eigenvalue	fully accurate (perturbed if necessary)	accurate to some figures (refined by Rayleigh-Quotient)
Iteration	multiple steps $\hat{\lambda}$ unchanged (reorthogonal. if needed)	one step with current $\hat{\lambda}$ gradually refine $\hat{\lambda}$ to high relative accuracy
Convergence criterion	Norm growth in solving $(T - \hat{\lambda} I)v = b$	relatively small residual (see (2.9))
Vector support	full	small if warranted

TABLE 2.1

Differences between standard inverse iteration (STEIN) and the MRRR algorithm (STEGR).

We have already explained that the MRRR algorithm does not use the original matrix T but a suitable LDL^T factorizations instead. As to the choice of the right-hand side, the MRRR algorithm need not use a random vector because it is possible to find the index r of the largest component of the true wanted eigenvector using a double factorization

$$(LDL^T - \hat{\lambda} I) = L^+ D^+ (L^+)^T = U^- D^- (U^-)^T \tag{2.3}$$

with $\mathcal{O}(n)$ work [14]. By using L^+, D^+, U^-, D^-, it is possible to find the index r of the largest entry in the (true) eigenvector associated with $\hat{\lambda}$, provided that $\hat{\lambda}$ is accurate enough.

One major difference between the MRRR algorithm and inverse iteration lies in the fact that the MRRR algorithm works with local eigenvalue approximations of (shifted) RRRs. However, when bisection is used, it is more efficient to compute eigenvalues only to a modest number of digits initially, enough to recognize clusters. The eigenvalue needs to be known to high relative accuracy only at the point when the eigenvector is computed. At intermediate stages, while the representation tree is constructed, eigenvalues only need to be accurate enough to distinguish between large and small relative gaps. Once the (local) eigenvalue approximation is relatively isolated from its neighbors, the Rayleigh-Quotient can be used to refine the eigenvalue more efficiently than bisection.

In order to compute the vector v from the approximation $\hat{\lambda}$, the MRRR algorithm solves

$$(LDL^T - \hat{\lambda} I)v = e_r \gamma_r, \;\; v(r) = 1, \tag{2.4}$$

with a normalization factor γ_r and $\gamma_r^{-1} = [(LDL^T - \hat{\lambda}I)^{-1}]_{rr}$. In order to guarantee (2.1), we solve (2.4) not by standard back substitution but with a *twisted factorization*

$$LDL^T - \hat{\lambda}I = N_r \Delta_r N_r^T. \tag{2.5}$$

where

$$N_r = \begin{pmatrix} 1 & & & & & & & & & \\ L^+(1) & 1 & & & & & & & & \\ & \ddots & \ddots & & & & & & & \\ & & L^+(r-2) & 1 & & & & & & \\ & & & L^+(r-1) & 1 & U^-(r) & & & & \\ & & & & & 1 & U^-(r+1) & & & \\ & & & & & & 1 & \ddots & & \\ & & & & & & & \ddots & U^-(n-1) & \\ & & & & & & & & 1 & \end{pmatrix},$$

and

$$\Delta_r = \mathrm{diag}(D^+(1), \ldots, D^+(r-1), \gamma_r, D^-(r+1), \ldots, D^-(n)).$$

Since $N_r e_r = e_r$ and $\Delta_r e_r = \gamma_r e_r$, the solution of (2.4) is equivalent to solving

$$N_r^T v = e_r, \;\; v(r) = 1, \tag{2.6}$$

so that

$$v(i) = \begin{cases} -L^+(i)v(i+1) & , \; i = r-1, \ldots, 1, \\ -U^-(i-1)v(i-1) & , \; i = r+1, \ldots, n; \end{cases} \tag{2.7}$$

this procedure solely uses multiplications, no additions or subtractions are necessary.

The index r in (2.4) is chosen according to

$$r = \arg \min_{1 \le k \le n} |\gamma_k|, \gamma_k^{-1} = [(LDL^T - \hat{\lambda}I)^{-1}]_{kk}, \tag{2.8}$$

so that when $\hat{\lambda}$ is a good approximation, r corresponds to an above average component in the true eigenvector.

Standard inverse iteration tries to guarantee a small residual by monitoring the growth rate in the vector norm when solving the singular system. The MRRR algorithm does not need this. From (2.4) follows immediately that the residual norm of $(\hat{\lambda}, v)$ is given by $|\gamma_r|/\|v\|_2$, and that the Rayleigh Quotient correction to $\hat{\lambda}$ is $|\gamma_r|/\|v\|_2^2$. Thus, the iteration stops when

$$|\gamma_r|/\|v\|_2 = \mathcal{O}\left(n\epsilon \mathrm{gap}(\hat{\lambda})\right) \tag{2.9}$$

which guarantees a small angle $\mathcal{O}(n\epsilon)$ between the computed and the true eigenvector according to (2.2); or it terminates when $|\gamma_r|/\|v\|_2^2 = \mathcal{O}(\epsilon|\hat{\lambda}|)$, in which case the the Rayleigh Quotient correction cannot improve $|\hat{\lambda}|$. If the residual is not small enough and the Rayleigh Quotient correction improves the initial approximation, it is applied and (2.4) is solved with the new improved $\hat{\lambda}$. When $\hat{\lambda}$ is relatively isolated, the previously found index r need not be recomputed.

Lastly, the MRRR algorithm can 'cut off' the computation of the eigenvector entries by (2.7) whenever the estimated change of the residual relative to the gap becomes marginal. Thus, if in the recursion for $v_{r+1}, \ldots, v_n$, we find

$$(|v(k)| + |v(k+1)|)L^+(k)D^+(k) = \mathcal{O}(\epsilon \text{gap}(\hat{\lambda})), \tag{2.10}$$

the algorithm sets $v(k+1:n) \equiv 0$. An analogous formula is used for the recursion for $v_{r-1}, \ldots, v_1$. For large values of n and 'localized' eigenvectors, this device avoids unnecessary arithmetic effort for negligible parts of the vector.

3. Illustration of possible subset problems. To illustrate the issue, we show in Figure 3.1 the representation tree of a sample matrix of dimension $n = 11$. Recall that the tree represents the cluster structure of the eigenvalues. The root node at the top of the figure represents an RRR for all eigenvalues which come in several clusters. For each of these clusters, a new RRR is computed, depicted as the descendants in the Figure. The leaf nodes at the bottom of the figure are the singletons for which an eigenvector is computed.

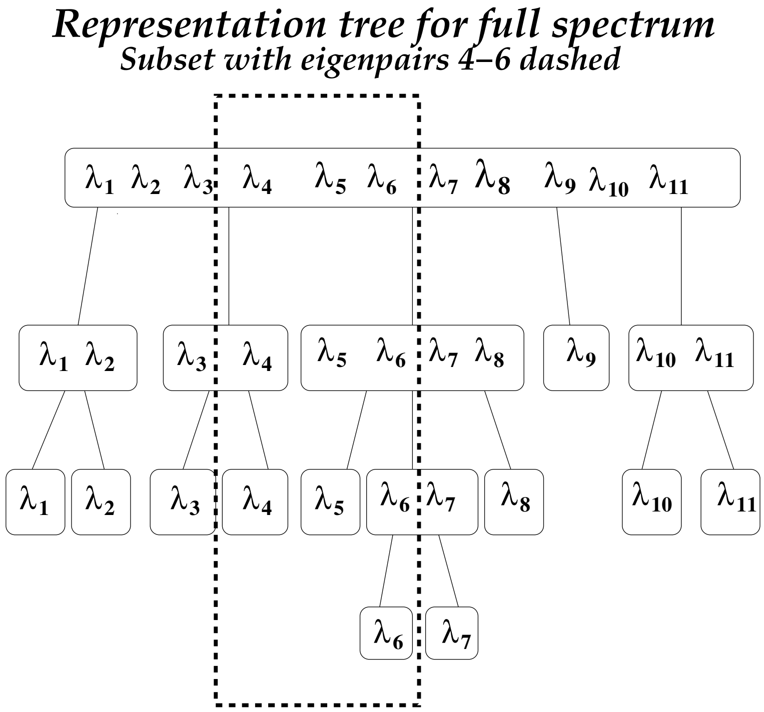

FIG. 3.1. *The representation tree for the whole matrix. Square boxes correspond to singletons, rectangular ones correspond to eigenvalue groups for which an individual representation is needed to improve relative gaps.*

In Figure 3.2, we show the representation tree for a chosen subset $\{4, 5, 6\}$. In this case, the root node is an RRR for the wanted eigenvalues only.

Note that with respect to the representation tree of the whole matrix, eigenvalue 4 is part of a cluster that needs to be refined. The (local) eigenvalue 4 finally becomes a singleton with respect to the child representation. Furthermore, eigenvalues 5, 6 are part of a larger cluster. With respect to the representation of this cluster, on the next deeper level of the representation tree, (local) eigenvalue 5 is a singleton and 6 belongs to a smaller cluster that needs another step of refinement.

In contrast, with respect to the representation tree for the subset $\{4, 5, 6\}$, the eigenvalue 4 is a singleton. It is relatively isolated from cluster $\{5, 6\}$ that, on the next level of the subset representation tree, has local eigenvalues that are relatively isolated. These differences are summarized in tabular form in Figure 3.3.

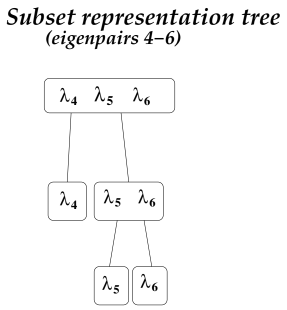

FIG. 3.2. *The representation tree for the spectrum subset of the matrix.*

Eigenvalue (index)	Representation full spectrum	Representation subset
4	2	1
5	2	2
6	3	2

FIG. 3.3. *For each eigenvalue, we show the tree level at which it is a singleton, once with respect to the representation tree for the full spectrum and once for the subset. Level 0 denotes the root, level 1 the descendants of the root, and so on. The level number also corresponds to the number of refinements of an eigenvalue before the vector is computed.*

In our tests, we have encountered four different problems related to this situation. We explain them by the example of eigenvalues 4 and 6 of the sample matrix. Two issues are related to the fact that the subset algorithm does not have knowledge about unwanted eigenvalues and thus not about the gap, either.

- **(A)** The convergence of the eigenvalue cannot be reliably judged by (2.9) if only the one-sided gap to the known part of the spectrum is taken into account. The (unknown) left gap of eigenvalue 4 or the right gap of eigenvalue 6 can be much smaller and thus the Rayleigh Quotient iteration might stop prematurely.
- **(B)** The criterion (2.10) for cutting off the computation of the eigenvector by (2.7) depends on gap$(\hat{\lambda})$. The one-sided gap could be much larger than the true gap and so give the vector too small a support.

These two problems are not extremely serious. The algorithm could easily compute a 'sentinel' eigenvalue on either side of the wanted subset and thus ensure that the criteria for convergence and support are using the correct gap value.

In order to describe the other two problems, we note that with respect to the RRR in the subset representation tree, eigenvalues 4 and 6 are each a 'false' singleton; both have a relative gap to a neighbor that is not large as can be seen from inspecting the full tree.

- **(C)** The bound from (2.2) guarantees a small angle of the computed to the true vector whenever the residual divided by the gap is of order $\mathcal{O}(n\epsilon)$. The MRRR algorithm achieves this by computing a vector with a small relative residual satisfying (2.1), where the local eigenvalue is so small that its relative gap is larger than the required threshold. This is *not* guaranteed for the extremal eigenvalues 4 and 6 in our example; indeed, in the full spectrum case, the algorithm would need to compute a new representation to make the relative gap larger than the threshold. Even with a residual being relatively small compared to the current local eigenvalue, the ratio of residual to the true gap can be significantly larger than $\mathcal{O}(n\epsilon)$.

- **(D)** The formula (2.8) yields an index r with the desirable property $|v(r)| = \|v\|_{infty}$ when $|\hat{\lambda} - \lambda|$ is small enough. Here, 'small enough' depends on how isolated λ is from the other eigenvalues. When λ is part of a cluster and $\hat{\lambda}$ is not much closer to λ than to the other cluster members, then γ_i will correspond to a linear combination of the eigenvectors associated with the cluster. As a consequence, $|v(r)|$ may not be large enough to guarantee that the error angle of the computed vector will be small.

In order to remedy problem (C), we compute the subset RRR very close to the extremal eigenvalue. This increases its relative gap.

Problem (D) can be overcome by recomputing the minimum γ_r and thus the twist index. Another way of reducing the contributions of unwanted eigenvectors in the computation would be subsequent steps of inverse iteration, however, this would involve additions and subtractions in the computation and thus a small relative residual could no longer be guaranteed. We remark that when, in a given matrix, the right-hand side is very sensitive to the accuracy in $\hat{\lambda}$, then the support of the computed vector can be sensitive, too. In such cases, we have to ensure that no data from previous iterations is stored in entries that should be zero.

At the end of this section, we stress that the aforementioned problems are very relevant in actual computations. While we have used the sample matrix from Figures 3.1 and 3.2 for illustration of the issues, we actually have encountered the described problems in tests on real matrices. We report some examples in Table 3.1. The first two tridiagonal matrices were obtained from running the Lanczos algorithm, with a starting vector filled with ones and no reorthogonalization, on the matrices BCSSTK01/BCSSTM01 and BCSSTK10/BCSSTM10, respectively. These are structural engineering matrices and part of the Rutherford-Boeing Sparse Matrix Collection [9]. The third matrix stems from computational quantum chemistry, we obtained it from G. Fann.

Matrix	Dimension	Subset	Failure type
T_bcsstkm01	48	7 - 19	**(A),(D)**
T_bcsstkm10	1086	501 - 985	**(B)**
Fann07 (SINGLE)	120	52 - 59	**(C)**

TABLE 3.1
Examples of failures encountered in MRRR subset computations.

4. Embedding the subset representation tree into the tree of the whole matrix. In the previous section, we have discussed the issues of the MRRR algorithm being applied to subset computations in a way that mimics standard inverse iteration. This section briefly compares our approach to the so-called 'conformal embedding' of the subset representation tree that is used in the parallel MRRR algorithm [3, 2]

In Figure 4.1, we illustrate this approach by the example from Figure 3.1.

The algorithm starts with an isolated superset of the wanted eigenpairs. For simplicity of presentation, we assume it to be the full spectrum. Then, the *relevant* part of the full representation tree is constructed. We compute those representations that define at least one of the wanted eigenvalues. The procedure is then repeated. The subset algorithm thus mimics the version for the full spectrum while omitting the computation of irrelevant representations.

The advantage of the embedded approach is that the algorithm produces mu-

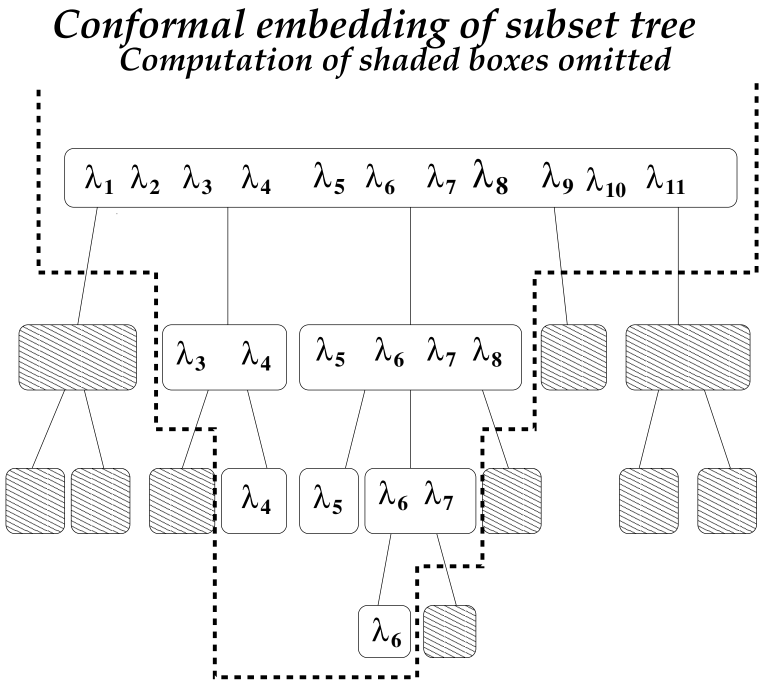

FIG. 4.1. *Illustration of the conformal embedding of the subset representation tree into the representation tree of the whole matrix (compare to Figures 3.1 and 3.2).*

tually orthogonal sets of eigenvectors for non-overlapping subsets of eigenvalues. In particular, the eigenvectors from the subset computation are consistent with those computed from the full spectrum. This is *not* guaranteed by standard inverse iteration like STEIN. In particular, the (parallel) ScaLAPACK version P_SYEVX does not guarantee orthogonality between subsets of eigenvectors on different processors.

However, there are two serious drawbacks that make, in our opinion, the embedded approach prohibitive for the sequential algorithm.

- By comparison, we can see that the embedded tree from Figure 4.1 can be deeper than the subset representation tree from Figure 3.2. Thus the overhead in computing intermediate RRRs is larger.
- The embedded algorithm requires us to find a superset of the wanted eigenvalues that is well isolated so that the embedding is conformal to the tree for full spectrum. Depending on the matrix and the wanted subset, this overhead can be substantial compared to $\mathcal{O}(nk)$ operations for the algorithm from Section 3 that focuses on the wanted eigenpairs only.

5. Summary and conclusions. In this paper, we described four different issues and pitfalls of the MRRR algorithm when it mimics the inverse iteration approach for subset computations. Problems arise when an end of the wanted subset is a false singleton, that is it is part of a separate cluster.

We have discussed two possible modifications to the original algorithm. Both take note of the cluster structure around the set of wanted eigenvalues. The first approach only requires knowledge of the nearest neighbors to the wanted subset. The second approach is based on a conformal embedding of the subset representation tree.

These two approaches have different advantages and drawbacks.

The first approach guarantees that the computed subset vectors are orthogonal to each other and have a small residual. This is similar to the promise made by inverse iteration: LAPACK's STEIN, at the additional cost of Gram-Schmidt orthogonalization, only guarantees numerical accuracy within the wanted subset. There is no guarantee that the computed subset eigenvectors are consistent with those computed from the full matrix spectrum.

The second approach makes this guarantee and is thus the method of choice for the parallel implementation. However, we have pointed out that the embedding requires a greater overhead because it includes a (potentially much larger) subset of the wanted eigenvalues.

Taking into account all these arguments, we chose the first approach for the MRRR subset algorithm because of its small extra cost. The second approach has its place in the parallel MRRR algorithm.

Acknowledgments. We are grateful to Paul Willems and Paolo Bientinesi for their comments on a first draft of this paper.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, 3. edition, 1999.

[2] D. Antonelli and C. Vömel. LAPACK working note 168: PDSYEVR. ScaLAPACK's parallel MRRR algorithm for the symmetric eigenvalue problem. Technical Report UCBCSD-05-1399, University of California, Berkeley, 2005.

[3] P. Bientinesi, I. S. Dhillon, and R. A. van de Geijn. A Parallel Eigensolver for Dense Symmetric Matrices based on Multiple Relatively Robust Representations. Technical Report TR-03-26, University of Texas, Austin, 2003. To appear in SIAM Journal on Scientific Computing.

[4] J. Bunch, P. Nielsen, and D. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31:31–48, 1978.

[5] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[6] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM J. Numer. Anal.*, 7(1):1–47, 1970.

[7] I. S. Dhillon. Current inverse iteration software can fail. *BIT*, 38:4:685–704, 1998.

[8] I. S. Dhillon, B. N. Parlett, and C. Vömel. LAPACK working note 162: The design and implementation of the MRRR algorithm. Technical Report UCBCSD-04-1346, University of California, Berkeley, 2004.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis. The Rutherford-Boeing Sparse Matrix Collection. Technical Report RAL-TR-97-031, Atlas Centre, Rutherford Appleton Laboratory, 1997. Also Technical Report ISSTECH-97-017 from Boeing Information & Support Services and Report TR/PA/97/36 from CERFACS, Toulouse.

[10] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, 16(1):172–191, 1995.

[11] I. C. F. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39(2):254–291, 1997.

[12] B. N. Parlett. *The symmetric eigenvalue problem*. Prentice Hall, Englewood Cliffs, NJ, 1980.

[13] B. N. Parlett. *Acta Numerica*, chapter The new qd algorithms, pages 459–491. Cambridge University Press, 1995.

[14] B. N. Parlett and I. S. Dhillon. Fernando's solution to Wilkinson's problem: an application of double factorization. *Linear Algebra and Appl.*, 267:247–279, 1997.

[15] B. N. Parlett and I. S. Dhillon. Relatively robust representations of symmetric tridiagonals. *Linear Algebra and Appl.*, 309(1-3):121–151, 2000.

[16] B. N. Parlett and I. S. Dhillon. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra and Appl.*, 387:1–28, 2004.

[17] B. N. Parlett and I. S. Dhillon. Orthogonal eigenvectors and relative gaps. *SIAM J. Matrix Anal. Appl.*, 25(3):858–899, 2004.