Journal of Computational and Applied Mathematics 236 (2012) 3696–3703

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Computing eigenvectors of block tridiagonal matrices based on twisted block factorizations

Gerhard König [a], Michael Moldaschl [b], Wilfried N. Gansterer [b,*]

[a] University of Vienna, Department of Computational Biological Chemistry, Austria

[b] University of Vienna, Research Group Theory and Applications of Algorithms, Austria

ARTICLE INFO

Article history:
Received 18 September 2010
Received in revised form 1 July 2011

Keywords:
Block tridiagonal matrix
Eigenvector computation
Twisted factorization
Twisted block factorization
Inverse iteration

ABSTRACT

New methods for computing eigenvectors of symmetric block tridiagonal matrices based on twisted block factorizations are explored. The relation of the block where two twisted factorizations meet to an eigenvector of the block tridiagonal matrix is reviewed. Based on this, several new algorithmic strategies for computing the eigenvector efficiently are motivated and designed. The underlying idea is to determine a good starting vector for an inverse iteration process from the twisted block factorizations such that a good eigenvector approximation can be computed with a single step of inverse iteration.

An implementation of the new algorithms is presented and experimental data for runtime behaviour and numerical accuracy based on a wide range of test cases are summarized. Compared with competing state-of-the-art tridiagonalization-based methods, the algorithms proposed here show strong reductions in runtime, especially for very large matrices and/or small bandwidths. The residuals of the computed eigenvectors are in general comparable with state-of-the-art methods. In some cases, especially for strongly clustered eigenvalues, a loss in orthogonality of some eigenvectors is observed. This is not surprising, and future work will focus on investigating ways for improving these cases.

1. Introduction

Block tridiagonal and banded matrices arise in many situations, for example, in the solution of differential equations via finite difference methods or in reduction processes in the context of eigenvalue computations. In the latter case, block tridiagonal matrices can be the intermediate result of a preprocessing step for computing spectral information of general dense matrices, resulting, for example, from a block tridiagonalization process [1] or from a bandwidth reduction process [2,3]. Most existing algorithms for computing spectral information of a band matrix first tridiagonalize the matrix, since many methods are known for efficiently computing eigenvalues and eigenvectors of a tridiagonal matrix. However, the tridiagonalization process tends to dominate the computational cost and has important disadvantages in terms of data locality which make it memory-bound [4]. This motivates our attempt in computing the eigenvectors of a band or block tridiagonal matrix *directly* (without tridiagonalization). One approach for doing this is the block tridiagonal divide-and-conquer (BD and C) method [5,6], which efficiently approximates eigenvalues and eigenvectors of a symmetric block tridiagonal matrix *without* tridiagonalizing it. However, the eigenvector accumulation in the divide-and-conquer process can become the main performance limiting factor of the BD and C method, in particular, in cases where reduced accuracy approximations (with respect to the highest possible accuracy determined by the problem instance and its condition as well

* Corresponding author.
E-mail addresses: gerhard@mdy.univie.ac.at (G. König), a0607892@unet.univie.ac.at (M. Moldaschl), wilfried.gansterer@univie.ac.at (W.N. Gansterer).

0377-0427
doi:10.1016/j.cam.2011.07.010

as by the given floating-point arithmetic) are not sufficient [6]. This motivates efforts in investigating efficient alternatives for directly computing eigenvectors of a symmetric block tridiagonal matrix (without reduction to tridiagonal form), given approximations of the corresponding eigenvalues.

We represent a generic block tridiagonal matrix $W(p)$ as

$$W(p) := \begin{pmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{p-1} & B_{p-1} & C_{p-1} \\ & & & A_p & B_p \end{pmatrix} \in \mathbb{R}^{n\times n}, \tag{1}$$

where $B_i \in \mathbb{R}^{b_i\times b_i}$ for $i = 1,\ldots,p$. The block sizes b_i determine size and shape of the $p-1$ subdiagonal blocks $A_i \in \mathbb{R}^{b_i\times b_{i-1}}$ $(i = 2,\ldots,p)$ and of the $p-1$ superdiagonal blocks $C_i \in \mathbb{R}^{b_i\times b_{i+1}}$ $(i = 1,\ldots,p-1)$. According to this definition, $W(p)$ is in general unsymmetric, but it has identical lower and upper bandwidths. Note that *band* structure can be considered a special case of block tridiagonal structure since any band matrix with identical lower and upper bandwidths has the general form (1) with upper triangular A_i and lower triangular C_i.

In this paper, we consider the task of computing eigenvectors of matrices of the form (1). We focus on *symmetric* $W(p)$ where $B_i = B_i^\top$ for $i = 1,\ldots,p$, and $C_i = A_{i+1}^\top$ for $i = 1,\ldots,p-1$. The approach pursued is based on utilizing *twisted block factorizations*, blocked generalizations of twisted factorizations of tridiagonal matrices (see, for example, [7]) in order to represent shifted $W(p)$ as a product of three matrices (two block tridiagonals with identities along the diagonal and one block diagonal).

First, we briefly review an algorithm for efficiently computing twisted block factorizations of $W(p)$ which we have proposed earlier [8]. Based on these factorizations, we present and experimentally evaluate an algorithm for computing an eigenvector of $W(p)$, given an approximation of the corresponding eigenvalue. The underlying idea is motivated by central components of the MRRR algorithm for computing eigenvectors of a symmetric tridiagonal matrix summarized in [9]. It may not be possible to directly generalize all aspects from the tridiagonal to the block tridiagonal case, but the insights summarized in this paper illustrate that it is worthwhile to pursue an analogous approach for the block tridiagonal case.

The central questions addressed in this paper are (i) how to select a single twisted block factorization of shifted $W(p)$ among all possible ones as the basis for an inverse iteration process, (ii) how to determine a good starting vector for this process, (iii) how the computational efficiency of this approach depends on central problem parameters (such as block sizes, etc.), and (iv) how competitive it is compared to existing approaches. We analytically motivate and compare several algorithmic variants and experimentally study their numerical accuracy as well as their computational performance.

1.1. Related work

Most of the relevant existing work focussed on the computation of eigenvectors of *tridiagonal* matrices. The highly accurate computation of the eigenvalues of a symmetric definite tridiagonal matrix [10,11] is an important building block for the development of very efficient methods for the calculation of eigenvectors of such matrices. Parlett and Dhillon [7] suggested the use of twisted factorizations of tridiagonal matrices for determining a good starting vector for inverse iteration. The underlying idea is that the position of the largest component of the eigenvector sought is associated with the minimal diagonal element of the twisted factorizations. The proper choice of the starting vector based on twisted factorizations leads to a stable and rapidly converging inverse iteration process. A single step of inverse iteration can be sufficient without requiring explicit reorthogonalization of the computed eigenvector [9,11,12,7,13–15].

So far, relatively little is known about how well such strategies generalize to banded or block tridiagonal matrices. Although Parlett and Dhillon [7] briefly mentioned a blocked extension of the tridiagonal case and also suggested a starting vector for the resulting inverse iteration process, they neither investigated algorithmic details nor evaluated this approach quantitatively. More recently, Vömel and Slemons [16] theoretically discussed twisted factorizations of banded or block-tridiagonal matrices. They gave a proof of the existence of two twisted factorizations of banded matrices by using a double factorization of the twisted block. They also summarized the connections to the inverse of the matrix and mentioned the potential use of their twisted factorizations for an inverse iteration process on band matrices—however, again without specifying or evaluating a concrete algorithm.

Vömel and Slemons focussed on *non-blocked* twisted factorizations of a band matrix. When pivoting is introduced for enhancing numerical stability, their approach does in general not preserve block tridiagonal or banded structure due to fill-in. In order to address both aspects—numerical stability and preservation of block tridiagonal structure—we utilize twisted *block* factorizations of $W(p)$ as presented in [8]. Our approach is related to the twisted block factorizations indicated in [7], but beyond that we integrate localized pivoting within blocks in the factorization process *without* causing fill-in.

1.2. Contributions

The approach investigated in this paper comprises three algorithmic components: (i) efficient computation of twisted block factorizations of $W(p)$, (ii) identification of a good starting vector for iterative computation of the desired eigenvector,

1. **initialize** $\tilde{v}_{(0)}$, $i := 0$
2. **repeat**
3. **solve** $\left(W(p) - \tilde{\lambda} I\right) y_{(i+1)} = \tilde{v}_{(i)}$
4. $\tilde{v}_{(i+1)} := y_{(i+1)} / \|y_{(i+1)}\|_2$
5. $i := i + 1$
6. **until convergence**

Fig. 1. Inverse iteration process.

and (iii) an efficient inverse iteration process with this starting vector. We have already discussed the first component earlier [8]. In this paper, we work on new aspects in the second and the third component.

More specifically, in this paper we investigate the following aspects beyond [8]. We motivate and investigate two new strategies (minsvd0 and minsvd2) for determining a good starting vector for the inverse iteration process. We discuss how the inverse iteration process can be performed efficiently for the specific matrix structures arising. We compare previously mentioned and newly developed starting vector selection strategies in terms of numerical properties and in terms of computational performance. Last, but not least, so far no experimental data about the applicability and competitiveness of eigenvector computations for block tridiagonal matrices based on twisted block factorizations can be found in the literature. We fully specify, implement and evaluate a complete algorithm for this task. In addition to our theoretical and analytical investigations we also summarize the results of comprehensive experimental evaluations of different algorithmic variants based on our implementation. Considering a wide range of test matrices, we clearly illustrate for which problem settings our new methods are competitive compared to existing standard approaches.

Synopsis. In Section 2, the process of efficiently computing twisted block factorizations of $W(p)$ is briefly reviewed, since it is one of three main components of the algorithms investigated. The identification of a well suited starting vector for the eigenvector computation is discussed in Section 3. An efficient inverse iteration process using this starting vector is the topic of Section 4. Comprehensive experimental performance evaluations based on an efficient implementation of these concepts are summarized in Section 5. Finally, conclusions and suggestions for future work are given in Section 6.

2. Component I: twisted block factorizations

In analogy to the approach pursued in the MRRR method for tridiagonal matrices [9], the first step of our approach is based on a factorization of block tridiagonal $W(p)$ into the product

$$W(p) = PLU \tag{2}$$

with a permutation matrix P and block tridiagonals L and U. We have presented a method for computing various decompositions of this form based on *twisted block LU factorizations* with local pivoting in [8]. We first briefly summarize this method before we move on to the subsequent components of our approach in the next sections.

The twisted block LU factorizations of $W(p)$ presented in [8] combine forward with backward block elimination steps. Assuming that all factorizations exist, we use the notation TF(f) ($f = 1, 2, \ldots, p$) for a twisted block factorization with $f - 1$ forward $p - f$ backward elimination steps. We denote the diagonal block at position f, where forward and backward elimination steps meet, as "twisted block". As shown in [8], the resulting factors L and U are both block tridiagonals, but only $p - f$ nonzero blocks appear above the block diagonal in L and below the main diagonal in U. For example, TF(3) of $W(4)$ produces

$$W(4) = \begin{pmatrix} P_1^+ & & & \\ & P_2^+ & & \\ & & P_3 & \\ & & & P_4^- \end{pmatrix} \begin{pmatrix} L_1^+ & & & \\ M_2^+ & L_2^+ & & \\ & M_3^+ & L_3 & M_3^- \\ & & & L_4^- \end{pmatrix} \begin{pmatrix} U_1^+ & N_1^+ & & \\ & U_2^+ & N_2^+ & \\ & & U_3 & \\ & & N_4^- & U_4^- \end{pmatrix}. \tag{3}$$

Superscripts are used to distinguish blocks computed in the forward direction ("$^+$") and blocks constructed in the backward direction ("$^-$").

3. Component II: identification of a starting vector

Assuming that we are given an eigenvalue λ of $W(p)$ (or an approximation $\tilde{\lambda}$ thereof, which will be called "shift" in the following), the twisted block factorizations of $W(p) - \tilde{\lambda} I$ can be computed as reviewed in Section 2. Based on these, the next task is to determine a proper starting vector $\tilde{v}_{(0)}$ for an inverse iteration process (cf. Fig. 1).

So far, we have not specified, *which* one of the p possible block twisted factorizations to use in the inverse iteration process. The choice of one of these factorizations also determines the starting vector $\tilde{v}_{(0)}$. In fact, we will utilize the information provided by all the block twisted factorizations of $W(p) - \tilde{\lambda} I$ for determining a suitable starting vector $\tilde{v}_{(0)}$. The idea which motivates this procedure is the connection between the twisted factorizations and the inverse of a matrix.

We review this connection in the following. For a properly chosen starting vector $\tilde{v}_{(0)}$ a few steps of the inverse iteration process should suffice for determining a good approximation of the eigenvector v.

3.1. Analytical motivation for eigenvector approximation

In the following, we first review basic ideas given in [7]. Based on this, we then formulate various concrete algorithmic strategies for determining an eigenvector of $W(p) - \tilde{\lambda} I$ in Section 3.2. In order to keep the notation simple, we assume in the following that all block sizes are equal, i.e., $b_i = b$ for all $i = 1, 2, \ldots, p$.

For each possible blocked twisted factorization TF(k), $1 \leq k \leq p$, define $\Gamma_k \in \mathbb{R}^{b\times b}$ and

$$Z := \begin{pmatrix} Z^+ \\ I_b \\ Z^- \end{pmatrix} \in \mathbb{R}^{n\times b}$$

with $Z^+ \in \mathbb{R}^{(k-1)b\times b}$, $Z^- \in \mathbb{R}^{(p-k)b\times b}$ such that

$$\left(W(p) - \tilde{\lambda} I\right) Z = PLU \begin{pmatrix} Z^+ \\ I \\ Z^- \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \Gamma_k \\ \mathbf{0} \end{pmatrix}. \tag{4}$$

Intuitively, if $\|\Gamma_k\|$ is small, Z contains good approximations to eigenvectors corresponding to λ. By omitting the kth block row in Eq. (4), two *independent* homogeneous systems remain. Denoting with the arguments $i_1 : i_2, i_3 : i_4$ the respective submatrices of P, L and U in Eq. (2) which contain block rows i_1 to i_2 and block columns i_3 to i_4, and introducing the variables $P^u := P(1 : k-1, 1 : k-1)$, $L^u := L(1 : k-1, 1 : k-1)$, $U^u := U(1 : k-1, 1 : k)$, $P^l := P(k+1 : p, k+1 : p)$, $L^l := L(k+1 : p, k+1 : p)$, $U^l := U(k+1 : p, k : p)$ for the respective parts of P, L and U, these two homogeneous systems can be written as

$$P^u L^u U^u \begin{pmatrix} Z^+ \\ I \end{pmatrix} = \mathbf{0} \tag{5}$$

$$P^l L^l U^l \begin{pmatrix} I \\ Z^- \end{pmatrix} = \mathbf{0}. \tag{6}$$

Assuming that the LU factorization exists, the $(k-1)b\times(k-1)b$ matrices P^u and L^u as well as the $(p-k)b\times(p-k)b$ matrices P^l and L^l must be invertible, leaving us with two equations with the system matrices U^u and U^l. The special structures of U^u and U^l allow for computing Z^+ using a blockwise backward substitution process and Z^- using a blockwise forward substitution process.

Using Eq. (3), we again illustrate this for the example $W(4)$ and TF(3): Eq. (5) translates into

$$\begin{pmatrix} U_1^+ & N_1^+ & \mathbf{0} \\ \mathbf{0} & U_2^+ & N_2^+ \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \\ I \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{7}$$

from which Z_2 and Z_1 can be determined using blockwise backward substitution, and Eq. (6) translates into

$$\begin{pmatrix} N_4^- & U_4^- \end{pmatrix} \begin{pmatrix} I \\ Z_4 \end{pmatrix} = \mathbf{0}, \tag{8}$$

from which Z_4 can be determined.

Based on Eq. (1), the omitted kth block row in Eq. (4) yields the following equation

$$A_k Z_{k-1} + B_k + C_k Z_{k+1} = \Gamma_k.$$

We can now substitute Z_{k-1} and Z_{k+1} computed from Eqs. (5) and (6) (compare Eqs. (7) and (8)) yielding

$$-A_k \left(U_{k-1}^+\right)^{-1} N_{k-1}^+ + B_k - C_k \left(U_{k+1}^-\right)^{-1} N_{k+1}^- = \Gamma_k. \tag{9}$$

Recalling from Eq. (3) that $P_k M_k^+ = A_k \left(U_{k-1}^+\right)^{-1}$ and that $P_k M_k^- = C_k \left(U_{k+1}^-\right)^{-1}$, we obtain

$$\Gamma_k = -P_k M_k^+ N_{k-1}^+ + B_k - P_k M_k^- N_{k+1}^-. \tag{10}$$

According to Eq. (3) this means that

$$\Gamma_k = P_k L_k U_k. \tag{11}$$

3.2. Strategies for starting vector selection

The relationships reviewed in Section 3.1 motivate various new strategies for determining the starting vector $\tilde{v}_{(0)}$ for the inverse iteration process based on the twisted block factorizations of $W(p) - \tilde{\lambda}I$. In this section, we present them, and in Section 5 they are evaluated numerically.

As outlined in [7], if Γ^* is the Γ_k with the minimal singular value over all singular values for all possible k and $(\sigma^*_{\min}, u^*, v^*)$ is the corresponding minimal singular triplet, then according to Eq. (4)

$$\left\| \left(W(p) - \tilde{\lambda} I \right) Z v* \right\|_2 = \sigma^*_{\min}. \tag{12}$$

Consequently, if $\sigma^*_{\min}$ is small enough, then Zv^* is a good approximation to an eigenvector of $W(p)$ corresponding to the shift $\tilde{\lambda}$.

Strategies minsvd0, minsvd1, and minsvd2. In these strategies, which are motivated by Eq. (12), the selection of the starting vector is based on the singular values of the subblocks of all twisted factorizations TF(f) of $W(p) - \tilde{\lambda}I$. For strategy *minsvd0*, the singular vector u^* corresponding to the minimal singular value $\sigma^*_{\min}$ of all matrices Γ^f_k ($k = 1, \ldots, p$, $f = 1, \ldots, p$, cf. Eq. (11)) has to be computed. The elements of the starting vector $\tilde{v}_{(0)}$ which are in the rows of Γ^* are set to u^*, all the others to zero. Strategy *minsvd1* (which is the only SVD-based strategy mentioned in [8]) is a computationally cheaper approximation of minsvd0, because it does not require computing any singular vectors: motivated by the localized pivoting done in each block in the twisted block factorization process, the position of the last row of the block Γ^* defines the position in $\tilde{v}_{(0)}$ which is set to one, at all others $\tilde{v}_{(0)}$ is set to zero. In both strategies minsvd0 and minsvd1, after determining $\tilde{v}_{(0)}$ one step of inverse iteration is performed for computing the eigenvector approximation $\tilde{v}_{(1)}$ as summarized in Section 4. Strategy *minsvd2* is directly based on Eq. (12): it determines the matrix Z in Eq. (4) using the blockwise back- and forward substitution processes sketched in Section 3.1, then computes the right singular vector v^* of Γ^*, and finally computes the eigenvector approximation $\tilde{v}_{(1)} := Zv^*$.

In summary: for strategy minsvd1 we need to know $\sigma^*_{\min}$ and its position, and we need to perform one step of inverse iteration. For strategy minsvd0, we need to know $\sigma^*_{\min}$, its position and the corresponding singular vector u^*, and we also need to perform one step of inverse iteration. For strategy minsvd2, we need to know the number of the block row of Γ^* and the singular vector v^* corresponding to $\sigma^*_{\min}$. Then we need to compute the matrix Z and multiply it with v^*.

Strategy minsca. This strategy is a very coarse approximation, but significantly reduces the computational cost compared to the minsvdx strategies, as it does not require the computation of any SVDs. Based on all twisted factorizations $W(p) - \tilde{\lambda}I = P^f L^f U^f$ the position m of the row of the minimum diagonal entry $|U^{f*}_{mm}|$ over all U^f defines the position of the starting vector $\tilde{v}_{(0)}$ which is set to one. The factorization TF(f^*) which contains the minimum diagonal element $|U^{f*}_{mm}|$ is used for computing this eigenvector approximation $\tilde{v}_{(1)}$ as summarized in Section 4.

Strategy random. As a reference strategy, a starting vector with random entries uniformly distributed in [0, 1] has been used.

4. Component III: efficient inverse iteration

In this section, we investigate an inverse iteration process for approximating the eigenvector v of $W(p)$ corresponding to λ based on the starting vector which has been determined according to one of the strategies discussed in Section 3.

If $|\tilde{\lambda} - \lambda|$ is sufficiently smaller than $|\tilde{\lambda} - \mu|$ for all eigenvalues $\mu \neq \lambda$ and if the starting vector $\tilde{v}_{(0)}$ contains a nonzero component in the desired eigenvector v, the inverse iteration process depicted in Fig. 1 will produce an approximation $\tilde{v}$ for the desired eigenvector v.

In general, a random starting vector $\tilde{v}_{(0)}$ is considered appropriate [17]. However, as indicated in [7] and discussed in detail in Section 3, it is possible to determine a better starting vector by using the twisted factorizations of shifted $W(p)$. Next, we discuss how to exploit the special block tridiagonal structure of the factors in the twisted block factorizations of shifted $W(p)$ for efficiently solving the linear systems arising in Line 3. of Fig. 1.

Solution of a block tridiagonal linear system.

Given a twisted block factorization (2) of $W(p) - \tilde{\lambda}I$, three steps are required for solving a linear system with $W(p) - \tilde{\lambda}I$:

a. Apply the inverse of the pivoting matrix P to the right hand side:

$$LUy_{(i+1)} = P^{-1}\tilde{v}_{(i)} =: c.$$

b. Solve $Ly = c$ for y via a *combined* forward and back substitution.
c. Solve $Uy_{(i+1)} = y$ for $y_{(i+1)}$ via *combined* back and forward substitution.

In the following, the combined substitution processes are derived. Without loss of generality, as in Section 2 we use the special case $p = 4$ and TF(3) for illustrating the concept. All vectors involved are partitioned into subvectors of length b_i corresponding to the blocks of the matrix and their indices correspond to the respective row indices of the matrix blocks.

Combined forward/back substitution on L. In the special case considered, Step b. in the solution process above has the following form:

$$\begin{pmatrix} L_1^+ & & & \\ M_2^+ & L_2^+ & & \\ & M_3^+ & L_3 & M_3^- \\ & & & L_4^- \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}.$$

Since both y_2 and y_4 have to be known before we can solve for y_3, it is necessary to start substituting at both ends, gradually solving the equations towards the twisted block. Forward substitution is performed on the forward factorization part marked with the superscripts "+" by first solving $L_1^+ y_1 = c_1$ for y_1 and then $L_2^+ y_2 = c_2 - M_2^+ y_1$ for y_2. The next block is already the twisted block where the forward and backward factorizations meet, thus y_4 is required before we can proceed. In a back substitution step on L, $L_4^- y_4 = c_4$ is solved for y_4. Note that *within* the block, this actually involves a forward substitution process, since L_4^- is lower triangular. Finally, in the block row of the twisted block we can solve $L_3 y_3 = c_3 - M_3^+ y_2 - M_3^- y_4$ for y_3.

Combined back/forward substitution on U. An analogous procedure can be applied to the matrix U for computing $y_{(i+1)}$. By introducing $x := y_{(i+1)}$ in order to simplify notation and by partitioning x appropriately, Step c. in the solution process above translates into solving the linear system

$$\begin{pmatrix} U_1^+ & N_1^+ & & \\ & U_2^+ & N_2^+ & \\ & & U_3 & \\ & & N_4^- & U_4^- \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}.$$

In contrast to the combined forward/back substitution discussed before, this time the substitution process has to start at the twisted block k (in our example, block number three) and proceeds towards the first and last block row of U, since x_k has to be known before the equations in block rows $k-1$ and $k+1$ can be solved. In our example TF(3) of $W(4)$, the combined back/forward substitution takes the following form. First, we solve $U_3 x_3 = y_3$ for x_3 (note that this is a back substitution process *within* the block U_3). Then, x_4 can be computed from the last block equation $U_4^- x_4 = y_4 - N_4^- x_3$ and x_2 from the second block equation $U_2^+ x_2 = y_2 - N_2^+ x_3$. Finally, x_1 can be computed from the first block equation $U_1^+ x_1 = y_1 - N_1^+ x_2$.

5. Experimental evaluation

In this section, we summarize extensive experimental evaluations of the five different strategies presented in Section 3.2 (minsvd0, minsvd1, minsvd2, minsca, random). The resulting algorithms for computing an eigenvector of $W(p)$ are compared in terms of runtime performance as well as in terms of the resulting quality of the eigenvector approximation. For this purpose, we have implemented the methods discussed in this paper in LAPACK-style BLAS-based Fortran routines. In all cases, only *one* step of inverse iteration has been performed.

Test data. Seven different types of symmetric banded test matrices with constant block sizes $b_i = b$ for $i = 1, \ldots, p$ were used in the experiments (this corresponds to upper triangular blocks A_i in Eq. (1)). Matrices of *Type* 0 have random entries uniformly distributed in [0, 1], *Type* 1 matrices have eigenvalues clustered around the machine epsilon $\pm\varepsilon_{mach}$, *Type* 2 matrices have eigenvalues clustered around ± 1, *Type* 3 matrices have eigenvalues geometrically distributed in $[-1, -\varepsilon_{mach}] \cup [\varepsilon_{mach}, 1]$, *Type* 4 matrices have eigenvalues arithmetically distributed in $[-1, -\varepsilon_{mach}] \cup [\varepsilon_{mach}, 1]$, *Type* 5 matrices have eigenvalues whose logarithms are uniformly distributed in $[-1, -\varepsilon_{mach}] \cup [\varepsilon_{mach}, 1]$, and *Type* 6 matrices have random eigenvalues which are uniformly distributed in $[-1, 1]$. Matrix types 1–6 were generated using software written by Yihua Bai.

5.1. Runtime performance

We evaluated the runtime performance of the different strategies on an Intel i7-860 CPU. Comparisons are provided with the most competitive state-of-the-art tridiagonalization-based routines from LAPACK [18]: The routine `dsbevd` reduces $W(p)$ to tridiagonal form, then applies the tridiagonal divide-and-conquer method for computing eigenvalues and eigenvectors, and finally transforms back the eigenvectors. The routine `dsbevr` also reduces $W(p)$ to tridiagonal form, then computes eigenvalues and eigenvectors based on relatively robust representations using the routine `dstemr`, and finally transforms back the eigenvectors. The routines `dsyevd` and `dsyevr` operate analogously, but they treat $W(p)$ as full matrix, thus not exploiting the band structure.

In general, the runtime for the different strategies compared depends on the type of the test matrix. An exception is the strategy *minsca*, where the runtime is independent of the matrix type. Our experiments showed that the LAPACK routines were fastest for matrices of Type 2. Consequently, our runtime comparisons focus on this matrix type, which is in this sense the "most difficult" case in terms of runtime performance for our new approaches.

Fig. 2 shows the runtimes for computing *all* eigenvectors for various matrix sizes with a fixed block size $b = 10$. The eigenvalues required in the approaches based on twisted block factorizations were computed using the routine `LAPACK/dsbevd`

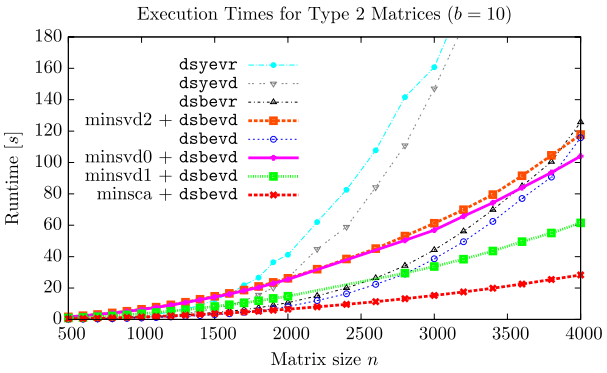

Fig. 2. Runtime comparison for fixed block size b.

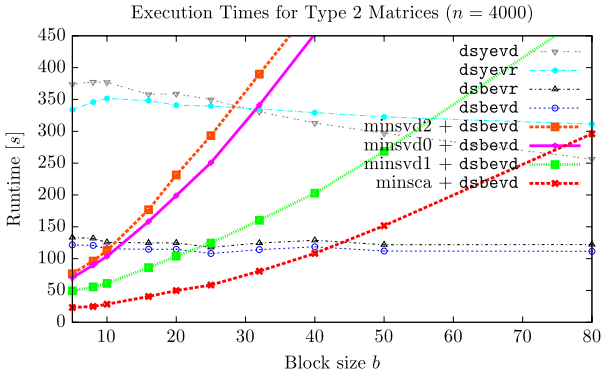

Fig. 3. Runtime comparison for increasing block size b.

in the "eigenvalues only" mode and the sum of the times is shown in Fig. 2 (denoted as "minxxxx + `dsbevd`" in the legend). Fig. 2 clearly illustrates that (i) exploiting the band structure is crucial for good performance, (ii) all methods based on twisted block factorizations are asymptotically competitive with the state-of-the-art tridiagonalization-based methods, and (iii) the strategy *minsca* is the clear winner with high speedups especially for large problems, followed by the strategy *minsvd1*.

Fig. 3 compares the same methods for fixed problem size $n = 4000$ and varying block sizes b. As expected, the performance benefits of the methods based on twisted block factorizations diminish for increasing block sizes. Nevertheless, in particular the strategy *minsca* and to some extent also the strategy *minsvd1* remain very competitive even for larger bandwidths. For small b, all methods based on twisted block factorizations outperform the classical tridiagonalization-based approaches.

5.2. Numerical accuracy

Table 1 summarizes experimental data about residuals

$$\mathfrak{R}_i := \left\| \left(W(p) - \tilde{\lambda}_i I \right) \tilde{v}_{i(1)} \right\|_1 / \left\| W(p) \right\|_1$$

and about eigenvector orthogonality

$$\mathfrak{O}_i := \left\| \left(\tilde{V}^\top \tilde{V} - I \right) (:, i) \right\|_\infty$$

resulting from the five different algorithmic strategies after one step of inverse iteration as percentages of computed eigenpairs where $\mathfrak{R} \leq n\varepsilon_{mach}$ and $\mathfrak{O} \leq n\varepsilon_{mach}$, respectively. We can see that all four strategies based on twisted block factorizations yield mostly very good residuals and perform clearly better than the random strategy. As expected, producing orthogonal eigenvectors within a *single* step of inverse iteration is a very difficult task, in particular when eigenvalues are strongly clustered as it is the case for many of the test matrices, particularly strong in matrix types 1 and 2. Nevertheless, also in terms of eigenvector orthogonality, the strategies based on twisted block factorizations clearly outperform the random strategy. We also would like to emphasize that the *minsca* strategy, which was by far the fastest, is also among the winners in terms of numerical accuracy in most cases.

Table 1
Percentage of computed eigenpairs with a relative residual and worst eigenvector orthogonality not exceeding $n\varepsilon_{mach}$ for the different matrix types ($n = 1700$ and $b = 17$). "mx" stands for the strategy *minsvd*x, "ms" for the strategy *minsca*, and "r" for the random strategy. The best values in each row are highlighted in bold face.

Matrix type	$\mathfrak{R}_i \leq n\varepsilon_{mach}$ (%)					$\mathfrak{O}_i \leq n\varepsilon_{mach}$ (%)				
	m0	m1	m2	ms	r	m0	m1	m2	ms	r
0	**100.0**	**100.0**	99.9	**100.0**	88.7	45.4	39.3	7.0	**47.7**	0.6
1	**100.0**	**100.0**	97.7	**100.0**	99.9	**0.2**	**0.2**	0.0	0.1	**0.2**
2	**100.0**	**100.0**	16.8	**100.0**	1.6	**1.6**	**1.6**	0.0	0.1	**1.6**
3	99.6	99.7	99.8	**99.9**	98.2	18.3	21.8	**29.4**	12.6	0.1
4	99.7	99.7	99.9	**100.0**	95.9	67.9	62.9	72.9	**73.9**	0.5
5	92.0	92.0	**92.1**	84.5	90.8	38.1	35.8	**50.2**	43.4	1.5
6	99.2	99.4	99.6	**99.9**	72.5	85.3	85.7	91.4	**92.6**	0.2
Avg	98.6	**98.7**	86.5	97.8	78.2	36.7	35.3	35.8	**38.6**	0.7

6. Conclusions and future work

Several new algorithmic variants for computing eigenvectors of symmetric block tridiagonal matrices based on twisted block factorizations have been analytically motivated, designed, implemented and evaluated experimentally. It has been shown that for very large problems and/or for small bandwidths, the methods proposed in this paper clearly outperform state-of-the-art tridiagonalization-based methods in terms of runtime. In terms of numerical accuracy, excellent residuals can be achieved within a single step of inverse iteration, but especially for test cases with a tightly clustered spectrum a certain loss of orthogonality in the computed eigenvectors has been observed. The computationally most efficient approximative strategy *minsca* is also among the winners in terms of numerical accuracy.

Due to its high performance potential, the twisted block factorization-based approach is an important and promising building block for alternatives to classical dense tridiagonalization-based eigensolvers. Ways for better handling the cases where a loss of orthogonality has been observed will be investigated in the future.

Acknowledgements

This work was partly supported by the Austrian Science Fund (FWF) under contract S10608-N13 (NFN SISE). We are grateful to Y. Bai for her tool for generating the test matrices.

References

[1] Y. Bai, W.N. Gansterer, R.C. Ward, Block tridiagonalization of effectively sparse symmetric matrices, ACM Trans. Math. Software 30 (2004) 326–352.
[2] C.H. Bischof, B. Lang, X. Sun, Parallel tridiagonalization through two-step band reduction, in: Proceedings of the 1994 Scalable High-Performance Computing Conference, Washington D.C., pp. 23–27.
[3] C.H. Bischof, B. Lang, X. Sun, A framework for symmetric band reduction, ACM Trans. Math. Software 26 (2000) 581–601.
[4] P. Luszczek, H. Ltaief, J. Dongarra, Two-Stage tridiagonal reduction for dense symmetric matrices using tile algorithms on multicore architectures, Technical Report 244, LAPACK Working Note, 2011.
[5] W.N. Gansterer, R.C. Ward, R.P. Muller, An extension of the divide-and-conquer method for a class of symmetric block-tridiagonal eigenproblems, ACM Trans. Math. Software 28 (2002) 45–58.
[6] W.N. Gansterer, R.C. Ward, R.P. Muller, W.A. Goddard III, Computing approximate eigenpairs of symmetric block tridiagonal matrices, SIAM J. Sci. Comput. 25 (2003) 65–85.
[7] B.N. Parlett, I.S. Dhillon, Fernando's solution to Wilkinson's problem: an application of double factorization, Linear Algebra Appl. 267 (1997) 247–279.
[8] W.N. Gansterer, G. König, On twisted factorizations of block tridiagonal matrices, in: Proceedings of the 10th International Conference on Computational Science 2010, Procedia Computer Science 1 (2010) 279–287.
[9] I.S. Dhillon, B.N. Parlett, C. Vömel, The design and implementation of the MRRR algorithm, ACM Trans. Math. Software 32 (2006) 533–560.
[10] J.W. Demmel, W. Kahan, Accurate singular values of bidiagonal matrices, SIAM J. Sci. Stat. Comput. 11 (1990) 873–912.
[11] I.S. Dhillon, B.N. Parlett, Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices, Linear Algebra Appl. 387 (2004) 1–28.
[12] K.V. Fernando, On computing an eigenvector of a tridiagonal matrix. Part I: basic results, SIAM J. Matrix Anal. Appl. 18 (1997) 1013–1034.
[13] B.N. Parlett, For tridiagonals T replace T with LDLt, J. Comput. Appl. Math. 123 (2000) 117–130.
[14] B.N. Parlett, I.S. Dhillon, Relatively robust representations of symmetric tridiagonals, Linear Algebra Appl. 309 (2000) 121–151.
[15] B. Parlett, O. Marques, An implementation of the dqds algorithm (positive case), Linear Algebra Appl. 309 (2000) 217–259.
[16] C. Vömel, J. Slemons, Twisted factorization of a banded matrix, BIT 49 (2009) 433–447.
[17] I.C.F. Ipsen, Computing an eigenvector with inverse iteration, SIAM Rev. 39 (1997) 254–291.
[18] E. Anderson, Z. Bai, C.H. Bischof, S. Blackford, J.W. Demmel, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D.C. Sorensen, Lapack Users' Guide, 3rd ed., SIAM Press, Philadelphia, PA, 1999.