Updating the QR decomposition of block tridiagonal and block Hessenberg matrices generated by block Krylov space methods

M. Gutknecht and T. Schmelzer [1]

Research Report No. 2005-10
November 2005

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

[1] Oxford University Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1 3UQ, United Kingdom

Updating the QR decomposition of block tridiagonal and block Hessenberg matrices generated by block Krylov space methods

M. Gutknecht and T. Schmelzer [1]

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Research Report No. 2005-10 November 2005

Abstract

For MinRes and SymmLQ it is essential to compute the QR decompositions of tridiagonal coefficient matrices gained in the Lanczos process. Likewise, for GMRes one has to find those of Hessenberg matrices. These QR decompositions are computed by an update scheme where in every step a single Givens rotation is constructed.

Generalizing this approach we introduce a block-wise update scheme for the QR decomposition of the block tridiagonal and block Hessenberg matrices that come up in generalizations of MinRes, SymmLQ, GMRes, and QMR to block methods for systems with multiple right-hand sides. Using (in general, complex) Householder reflections instead of Givens rotations is seen to be much more efficient in the block case. Some implementation details and numerical experiments on accuracy and timing are given. In particular, we compare our method with the one based on Givens rotations that has been used in a version of block QMR. Our treatment includes the option of deflation, that is, the successive reduction of the block size due to linear dependencies.

Keywords: block Arnoldi process, block Lanczos process, block Krylov space method, block MinRes, block SymmLQ, block GMRes, block QMR, block tridiagonal matrix, block Hessenberg matrix, QR decomposition, QR factorization

[1] Oxford University Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1 3UQ, United Kingdom

1 Introduction

In 1975 Paige and Saunders [1] proposed two Krylov (sub)space methods called MinRes and SymmLQ for solving sparse Hermitian indefinite linear systems

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

with $\mathbf{A} = \mathbf{A}^{\mathsf{H}} \in \mathbb{C}^{N\times N}$, $\mathbf{x} \in \mathbb{C}^{N}$ and $\mathbf{b} \in \mathbb{C}^{N}$. In contrast to the conjugate gradient (CG) and conjugate residual (CR) methods that had been introduced more than 20 years earlier by Hestenes and Stiefel [2] [3] and are mostly limited to Hermitian positive definite systems (there is a version of CR that can cope with indefinite systems), the new methods made use of a different approach: during the recursive construction of *orthonormal* bases $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1}\}$ for the Krylov subspaces

$$\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) := \mathsf{span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{n-1}\mathbf{r}_0\right\}, \tag{2}$$

for each n, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is projected into the Krylov space $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$ and its projection is solved in terms of the coordinates. So, basically, for each n, a projection of the system is solved in coordinate space. The approach relies on the fact that this can be done very efficiently in a recursive way by updating the LQ or the QR decomposition of a tridiagonal matrix. Despite the different approach, whenever CR and CG do not break down due to the indefiniteness of $\mathbf{A}$, Paige and Saunders' methods produce the same solutions (in exact arithmetic). Their wider applicability is due to the guaranteed existence of a nested set of orthonormal bases, while the collinear orthogonal bases of residual vectors constructed implicitly by CG may not exist due to an infinite scaling factor.

The extension of MinRes to nonsymmetric systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ is straightforward, yet it took another ten years till the GMRes algorithm was introduced by Saad and Schultz [4]. Here a QR decomposition of a Hessenberg matrix needs to get updated in each step. Another generalization to the nonsymmetric case is the QMR method of Freund and Nachtigal [5], which is based on the nonsymmetric Lanczos process and makes again just use of the QR decomposition of a tridiagonal matrix as long as no look-ahead [6] is needed.

The same approach also works (with a number of additional difficulties to cope with), for systems with multiple right-hand sides, which we will also write as (1), but with

$$\mathbf{A} = \mathbf{A}^{\mathsf{H}} \in \mathbb{C}^{N\times N}, \qquad \mathbf{x} \in \mathbb{C}^{N\times s}, \qquad \mathbf{b} \in \mathbb{C}^{N\times s}. \tag{3}$$

The block GMRes method for the nonsymmetric case was introduced by Vital [7], the block versions of MinRes and SymmLQ were investigated in detail in [8], and block versions of QMR were introduced by various authors [9], [10], [11], [12]. In the first method one has to QR-decompose a block Hessenberg matrix with upper triangular (or upper trapezoidal) subdiagonal blocks, in the other methods the matrix is block triangular and has the same type of subdiagonal blocks (if we assume no look-ahead in QMR). In this paper we describe the updating of the QR decomposition of matrices with these structures. While in the case of a single system constructing one Givens rotation per step yields an extremely efficient update algorithm, the block case requires $\mathcal{O}(s^2)$ rotations per block step. We will show that in practice it is much more efficient to apply Householder reflections instead. The major portion of the computing time reduction comes from the usage of BLAS2-type block operations and is hardware dependent.

While in a standard Krylov space method the approximants $\mathbf{x}_n$ of the solution $\mathbf{x}_\star$ of a single system are chosen such that $\mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{r}_0 :\equiv \mathbf{b} - \mathbf{A}\mathbf{x}_0$ is the initial residual, in block Krylov space methods each of the s nth approximations $\mathbf{x}_n^{(i)}$ of the s solutions of $\mathbf{A}\mathbf{x}_n^{(i)} = \mathbf{b}^{(i)}$ $(i = 1, \ldots, s)$ are chosen so that

$$\mathbf{x}_n^{(i)} - \mathbf{x}_0^{(i)} \in \mathcal{B}_n(\mathbf{A}, \mathbf{r}_0),$$

where

$$\begin{aligned}\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0) &:\equiv \text{block span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{n-1}\mathbf{r}_0\right\} \\ &:\equiv \left\{\sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{r}_0 \boldsymbol{\gamma}_k \,;\; \boldsymbol{\gamma}_k \in \mathbb{C}^s \ (k = 0, \ldots, n-1)\right\} \\ &= \mathcal{K}_n\left(\mathbf{A}, \mathbf{r}_0^{(1)}\right) + \cdots + \mathcal{K}_n\left(\mathbf{A}, \mathbf{r}_0^{(s)}\right) \subseteq \mathbb{C}^N\end{aligned}$$

is the sum of the s individual nth Krylov subspaces. But, in general, this is not a direct sum.

In exact arithmetic the block Arnoldi process (in the non-Hermitian case) or the block Lanczos process (in the Hermitian case) create **block vectors** $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1} \in \mathbb{C}^{N \times s}$ whose orthonormal columns are a basis for $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$:

$$\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0) = \text{block span}\left\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1}\right\}.$$

Deflation is crucial for the stability of block Lanczos, and it saves memory space and computing time in block Arnoldi. It is accounted for, but not investigated in detail in this paper. It means to delete those columns of $\mathbf{y}_n$ that are already contained in $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$. As a consequence, the block vector $\mathbf{y}_i$ has s_i columns, where $s \geq s_0 \geq s_i \geq s_{i+1}$, $i = 1, 2, \ldots$.

We denote by $t_n(\mathbf{A}, \mathbf{r}_0)$ the dimension of $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$, which implies that

$$t_n(\mathbf{A}, \mathbf{v}_0) = \sum_{i=0}^{n-1} s_i.$$

Since in the theory of ordinary Krylov spaces the **grade of $\mathbf{A}$ with respect to $\mathbf{r}_0$** is an important notion [1], we introduce a corresponding one for the block case:

Definition 1 *The smallest index n with*

$$t_n(\mathbf{A}, \mathbf{r}_0) = t_{n+1}(\mathbf{A}, \mathbf{r}_0)$$

is called the ***block grade of $\mathbf{A}$ with respect to $\mathbf{r}_0$*** *and denoted by $\bar{\nu}(\mathbf{A}, \mathbf{r}_0)$.*

In this paper we start with a brief review of the block Arnoldi and the symmetric block Lanczos processes. The nonsymmetric block Lanczos process is only referred to briefly. Then we turn to the main subject of recursive QR decomposition of the corresponding block tridiagonal and block Hessenberg matrices. A block update procedure using, in general, complex Householder reflections is proposed and described in full detail. Accuracy and timing experiments that confirm its superiority are also given.

2 A block Arnoldi process

There are various block Arnoldi algorithms. Here we just cite our favorite one, where new columns are added block-wise, s_n at step n, and where orthogonalization is done with the modified block Gram-Schmidt (MBlGS) algorithm. Deflation relies on testing the numerical rank of a block vector by some rank-revealing QR factorization. When we write $\mathsf{rank}\,\widetilde{\mathbf{y}}$ we actually mean this numerical rank of $\widetilde{\mathbf{y}}$ determined by such a factorization.

Algorithm 1 (Block Arnoldi algorithm based on MBlGS)

Let a non-Hermitian matrix $\mathbf{A}$ and an orthonormal block vector $\mathbf{y}_0 \in \mathbb{C}^{N\times s}$ be given. For constructing a nested set of orthonormal block bases $\{\mathbf{y}_0, \dots, \mathbf{y}_{m-1}\}$ for the nested Krylov subspaces $\mathcal{B}_m(\mathbf{A}, \mathbf{y}_0)$ $(m = 1, 2, \dots \leq \bar{\nu}(\mathbf{A}, \mathbf{y}_0))$ proceed, for $n = 1, 2, \dots,$ as follows:

(1) Compute $\mathbf{A}\mathbf{y}_{n-1}$ by evaluating s_{n-1} matrix-vector products (MVs) in parallel:

$$\widetilde{\mathbf{y}} := \mathbf{A}\mathbf{y}_{n-1}. \tag{4}$$

(2) Subtract the projections of $\widetilde{\mathbf{y}}$ on the earlier computed basis vectors, that is, for $k = 0, \dots, n-1$ compute

[1] Wilkinson [13] called it the *grade of $\mathbf{r}_0$ with respect to $\mathbf{A}$*, but like Ilić and Turner [14] we think that *grade of $\mathbf{A}$ with respect to $\mathbf{r}_0$* is the more natural phrase, since this grade is the degree of the minimal polynomial of the restriction of $\mathbf{A}$ to the Krylov space generated by $\mathbf{A}$ from $\mathbf{r}_0$.

$$\begin{aligned}\boldsymbol{\eta}_{k,n-1} &:= \langle \mathbf{y}_k, \widetilde{\mathbf{y}} \rangle\,, & (5)\\ \widetilde{\mathbf{y}} &:= \widetilde{\mathbf{y}} - \mathbf{y}_k\, \boldsymbol{\eta}_{k,n-1}\,. & (6)\end{aligned}$$

(3) Compute the QR factorization of $\widetilde{\mathbf{y}} \perp \mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0)$ *and thereby determine* $\mathsf{rank}\,\widetilde{\mathbf{y}} = s_n \le s_{n-1}$*:*

$$\widetilde{\mathbf{y}} =: \begin{pmatrix} \mathbf{y}_n & \mathbf{y}_n^{\Delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{\rho}_n & \boldsymbol{\rho}_n^{\square} \\ \mathbf{0} & \boldsymbol{\rho}_n^{\Delta} \end{pmatrix} \boldsymbol{\pi}_n^{\mathsf{T}} =: \begin{pmatrix} \mathbf{y}_n & \mathbf{y}_n^{\Delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}_{n,n-1} \\ \boldsymbol{\eta}_{n,n-1}^{\Delta} \end{pmatrix}, \tag{7}$$

where:

$\boldsymbol{\pi}_n$ *is an* $s_{n-1} \times s_{n-1}$ *permutation matrix,*
$\mathbf{y}_n$ *is an* $N \times s_n$ *block vector with full numerical column rank, which goes into the basis,*
$\mathbf{y}_n^{\Delta}$ *is an* $N \times (s_{n-1} - s_n)$ *matrix that will be deflated (deleted),*
$\boldsymbol{\rho}_n$ *is an* $s_n \times s_n$ *upper triangular, nonsingular matrix,*
$\boldsymbol{\rho}_n^{\square}$ *is an* $s_n \times (s_{n-1} - s_n)$ *matrix,*
$\boldsymbol{\rho}_n^{\Delta}$ *is an upper triangular* $(s_{n-1} - s_n) \times (s_{n-1} - s_n)$ *matrix with* $\|\boldsymbol{\rho}_n^{\Delta}\|_F = O(\sigma_{s_n+1})$*, where* σ_{s_n+1} *is the largest singular value of* $\widetilde{\mathbf{y}}$ *smaller or equal to* tol.

The permutations are encapsulated in the block coefficients $\boldsymbol{\eta}_{n,n-1}$. Let

$$\mathbf{P}_n :\equiv \mathsf{block\ diag}\,(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n)$$

be the permutation matrix that describes all these permutations. Note that $\mathbf{P}_n^{\mathsf{T}} = \mathbf{P}_n^{-1}$. If $\mathsf{tol} = 0$ we speak of **exact deflation**. Assuming exact deflation only, it is easy to show that the t_{n-1} columns of

$$\mathbf{Y}_n :\equiv \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \ldots & \mathbf{y}_{n-1} \end{pmatrix} \in \mathbb{C}^{N \times t_n}$$

form indeed an orthonormal basis of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and that for $n < \bar{\nu}^{\square}(\mathbf{A}, \mathbf{y}_0)$ the fundamental Arnoldi relation

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{H}}_n, \tag{8}$$

still holds, where

$$\underline{\mathbf{H}}_n :\equiv \left(\begin{array}{c} \mathbf{H}_n \\ \\ \hline \mathbf{0} \ \ldots \ \mathbf{0} \ \boldsymbol{\eta}_{n,n-1} \end{array}\right) :\equiv \left(\begin{array}{cccc} \boldsymbol{\eta}_{0,0} & \boldsymbol{\eta}_{0,1} & \cdots & \boldsymbol{\eta}_{0,n-1} \\ \boldsymbol{\eta}_{1,0} & \boldsymbol{\eta}_{1,1} & \cdots & \boldsymbol{\eta}_{1,n-1} \\ & \boldsymbol{\eta}_{2,1} & \ddots & \vdots \\ & & \ddots & \boldsymbol{\eta}_{n-1,n-1} \\ \hline & & & \boldsymbol{\eta}_{n,n-1} \end{array}\right) \in \mathbb{C}^{t_{n+1} \times t_n}\,. \tag{9}$$

Alternative block Arnoldi algorithms apply classical (block) Gram-Schmidt (CBlGS) or compute one new column after another using classical or modified Gram-Schmidt or Householder reflections. The column-wise versions have the major disadvantage that only one MV is computed at once.

3 A symmetric block Lanczos process

If $\mathbf{A}$ is Hermitian and either none or only exact deflation occurs, all the blocks $\boldsymbol{\eta}_{k,m}$ with $k < m-1$ in $\underline{\mathbf{H}}_n$ are zero blocks, and the square part $\mathbf{H}_n$ of $\underline{\mathbf{H}}_n$ turns out to be Hermitian too. So, $\mathbf{H}_n$ is a Hermitian block tridiagonal matrix now called $\mathbf{T}_n$, and $\underline{\mathbf{H}}_n$ becomes $\underline{\mathbf{T}}_n$. This reduces the cost of generating the basis dramatically, but it also causes a strong propagation of roundoff errors. To take the changes into account we have to replace the formulas in step (2) of the block Arnoldi algorithm (Algorithm 1) and make a small change of notation in step (3).

If inexact deflation occurs, there are two options. The first consists in explicitly orthogonalizing against any block vector $\mathbf{y}_k$ whose "child" $\widetilde{\mathbf{y}}$, the projection of $\mathbf{A}\mathbf{y}_k$ onto a subspace orthogonal to $\mathsf{span}\begin{pmatrix} \mathbf{y}_k & \mathbf{y}_{k-1} \end{pmatrix}$, has components that were deflated (in analogy to what has been done in [12]). This means that $\mathbf{T}_n$ is no longer block tridiagonal. The second is to accept a loss of orthogonality linked to avoiding this explicit orthogonalization. We will stick to the second option in this section. The other case is later covered by treating the QR decomposition of a block Hessenberg matrix.

Algorithm 2 (Symmetric block Lanczos algorithm)

Let a Hermitian matrix $\mathbf{A}$ *and an orthonormal block vector* $\mathbf{y}_0 \in \mathbb{C}^{N\times s}$ *be given. For constructing a nested set of orthonormal block bases* $\{\mathbf{y}_0, \dots, \mathbf{y}_{m-1}\}$ *for the nested Krylov subspaces* $\mathcal{B}_m(\mathbf{A}, \mathbf{y}_0)$ $(m = 1, 2, \dots \leq \bar{\nu}(\mathbf{A}, \mathbf{y}_0))$ *proceed, for* $n = 1, 2, \dots,$ *as follows:*

(1) Compute $\mathbf{A}\mathbf{y}_{n-1}$ *by applying* s_{n-1} MV*s in parallel:*

$$\widetilde{\mathbf{y}} := \mathbf{A}\mathbf{y}_{n-1}. \tag{10}$$

(2) Subtract the projections of $\widetilde{\mathbf{y}}$ *on the two last computed block vectors, that is compute*

$$\widetilde{\mathbf{y}} := \widetilde{\mathbf{y}} - \mathbf{y}_{n-2}\boldsymbol{\beta}_{n-2}^{\mathsf{H}} \quad \textit{if } n > 1, \tag{11}$$

$$\boldsymbol{\alpha}_{n-1} := \langle \mathbf{y}_{n-1}, \widetilde{\mathbf{y}} \rangle \,, \tag{12}$$

$$\widetilde{\mathbf{y}} := \widetilde{\mathbf{y}} - \mathbf{y}_{n-1}\boldsymbol{\alpha}_{n-1}\,. \tag{13}$$

(3) Compute the QR factorization of $\widetilde{\mathbf{y}} \perp \mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0)$ *with* $\mathsf{rank}\, \widetilde{\mathbf{y}} = s_n \leq s_{n-1}$*:*

$$\widetilde{\mathbf{y}} \;=:\; \begin{pmatrix} \mathbf{y}_n \; \mathbf{y}_n^{\triangle} \end{pmatrix} \begin{pmatrix} \boldsymbol{\rho}_n & \boldsymbol{\rho}_n^{\square} \\ \mathbf{0} & \boldsymbol{\rho}_n^{\triangle} \end{pmatrix} \boldsymbol{\pi}_n^{\mathsf{T}} \;=:\; \begin{pmatrix} \mathbf{y}_n \; \mathbf{y}_n^{\triangle} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta}_{n-1} \\ \boldsymbol{\beta}_{n-1}^{\triangle} \end{pmatrix}, \tag{14}$$

where $\boldsymbol{\pi}_n$, $\mathbf{y}_n$, $\mathbf{y}_n^{\triangle}$, $\boldsymbol{\rho}_n$, $\boldsymbol{\rho}_n^{\square}$, *and* $\boldsymbol{\rho}_n^{\triangle}$ *are defined as in Algorithm* 1.

The Arnoldi relation (8) is now replaced by the Lanczos relation

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n\,, \tag{15}$$

where

$$\underline{\mathbf{T}}_n :\equiv \left(\begin{array}{c} \mathbf{T}_n \\ \hline \mathbf{0} \ldots \mathbf{0}\; \beta_{n-1} \end{array} \right) :\equiv \left(\begin{array}{ccccc} \boldsymbol{\alpha}_0 & \boldsymbol{\beta}_0^{\mathsf{H}} & & & \\ \boldsymbol{\beta}_0 & \boldsymbol{\alpha}_1 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \boldsymbol{\beta}_{n-2}^{\mathsf{H}} \\ & & & \boldsymbol{\beta}_{n-2} & \boldsymbol{\alpha}_{n-1} \\ \hline & & & & \boldsymbol{\beta}_{n-1} \end{array} \right) \in \mathbb{C}^{t_{n+1} \times t_n} \tag{16}$$

and $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i^{\mathsf{H}}$ for all $i = 1, 2, \ldots n-1$, so that $\mathbf{T}_n$ is Hermitian.

The alternative versions of the block Arnoldi algorithm that we mentioned are also available for block Lanczos, except that the difference between CBlGS and MBlGS is a smaller one since we have just a three-term block recurrence.

Nonsymmetric block Lanczos algorithms generate two sets of biorthogonal (or, dual) bases for the block Krylov subspaces $\mathcal{B}_m(\mathbf{A}, \mathbf{y}_0)$ and their dual spaces $\mathcal{B}_m(\mathbf{A}^{\mathsf{H}}, \mathbf{z}_0)$, where $\mathbf{z}_0$ may differ from $\mathbf{y}_0$. The block Lanczos relation (15) holds again (along with an analogue one for the dual space), but the square part of the block tridiagonal matrix $\underline{\mathbf{T}}_m$ is no longer Hermitian. Nevertheless the QR updating scheme developed in the following section would apply also to this case, since, as we will see, the symmetry is not capitalized upon.

The symmetric block Lanczos algorithm was the first block Krylov space method that was introduced [15], [16], [17], [18]. The aim was to compute multiple eigenvalues and corresponding eigenspaces. The nonsymmetric block version was defined in [19] and has since been addressed in a number papers, e.g., [20]. For the symmetric case the column-wise version was proposed by Ruhe [21], for the nonsymmetric block Lanczos method it was advocated in various papers in the 1990ies, in particular [22], [10], [12]. The column-wise

"Ruhe version" of block Arnoldi was stated in [23] [2].

4 An update scheme for the QR decomposition of block tridiagonal matrices

From (8) and (15) it follows that the subdiagonal blocks of $\underline{\mathbf{H}}_n\mathbf{P}_n$ and $\underline{\mathbf{T}}_n\mathbf{P}_n$ are upper trapezoidal. This will be capitalized upon in our update scheme for the QR decomposition of these matrices. We treat the block tridiagonal matrices generated by the symmetric Lanczos process first since their structure is more special. The generalization to block Hessenberg matrices will be easy.

Let $\underline{\mathbf{T}}_n\mathbf{P}_n = \mathbf{Q}_{n+1}\underline{\mathbf{R}}_n$ be the QR decomposition of $\underline{\mathbf{T}}_n\mathbf{P}_n$ so that $\mathbf{Q}_{n+1}$ is a unitary $t_{n+1}\times t_{n+1}$ matrix and $\underline{\mathbf{R}}_n$ is an upper triangular $t_{n+1}\times t_n$ matrix with full column rank. Recall that

$$\underline{\mathbf{T}}_n\mathbf{P}_n = \left(\begin{array}{ccccc} \boldsymbol{\alpha}_0\boldsymbol{\pi}_1 & \boldsymbol{\beta}_0^{\mathsf{H}}\boldsymbol{\pi}_2 & & & \\ \boldsymbol{\beta}_0\boldsymbol{\pi}_1 & \boldsymbol{\alpha}_1\boldsymbol{\pi}_2 & \ddots & & \\ & \boldsymbol{\beta}_1\boldsymbol{\pi}_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n \\ & & & \ddots & \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n \\ \hline & & & & \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right) \in \mathbb{C}^{t_{n+1}\times t_n}, \qquad (17)$$

where

$\boldsymbol{\alpha}_i\boldsymbol{\pi}_{i+1}$ is an $s_i \times s_i$ block and
$\boldsymbol{\beta}_i\boldsymbol{\pi}_{i+1}$ is an $s_{i+1}\times s_i$ upper trapezoidal block, $\boldsymbol{\beta}_i\boldsymbol{\pi}_{i+1} = \begin{pmatrix} \boldsymbol{\rho}_{i+1} & \boldsymbol{\rho}_{i+1}^{\square} \end{pmatrix}$.

$\underline{\mathbf{R}}_n$ has the form

$$\underline{\mathbf{R}}_n :\equiv \left(\begin{array}{ccccc} \widetilde{\boldsymbol{\alpha}}_0 & \widetilde{\boldsymbol{\beta}}_0 & \widetilde{\boldsymbol{\gamma}}_0 & & \\ & \widetilde{\boldsymbol{\alpha}}_1 & \widetilde{\boldsymbol{\beta}}_1 & \ddots & \\ & & \ddots & \ddots & \widetilde{\boldsymbol{\gamma}}_{n-3} \\ & & & \ddots & \widetilde{\boldsymbol{\beta}}_{n-2} \\ & & & & \widetilde{\boldsymbol{\alpha}}_{n-1} \\ \hline & & \mathbf{0}_{s_n\times t_n} & & \end{array}\right) \in \mathbb{C}^{t_{n+1}\times t_n}, \qquad (18)$$

[2] But note that the upper bound in the `for`-loop over j on line 2 of Algorithm 6.23 in [23] should be $mp+p-1$ instead of m.

where

$\widetilde{\boldsymbol{\alpha}}_i$ is an $s_i \times s_i$ upper triangular block with full rank,
$\widetilde{\boldsymbol{\beta}}_i$ is an $s_i \times s_{i+1}$ block, and
$\widetilde{\boldsymbol{\gamma}}_i$ is an $s_i \times s_{i+2}$ lower trapezoidal block.

Examples are depicted in Figs. 1 and 2. Note that the matrix $\underline{\mathbf{T}}_n\mathbf{P}_n$ is, in general, not even structurally symmetric, but all subdiagonal blocks are upper trapezoidal. When some permutations $\boldsymbol{\pi}_i$ differ from the identity, the upper edge of the band needs not decrease monotonously.

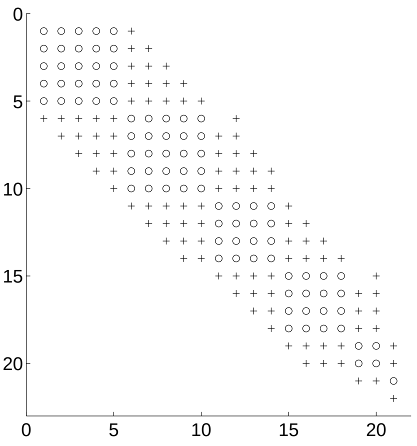
Fig. 1. The block tridiagonal structure of a matrix $\underline{\mathbf{T}}_n\mathbf{P}_n$.

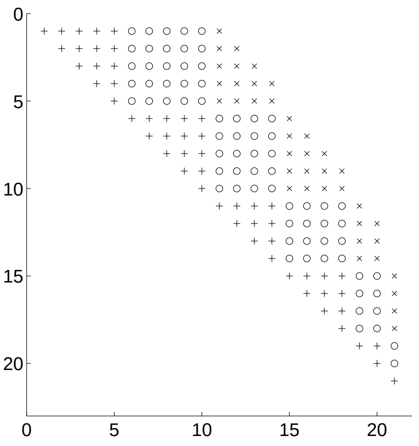
Fig. 2. The corresponding block structure of the matrix $\underline{\mathbf{R}}_n$.

We determine the unitary matrix $\mathbf{Q}_{n+1}$ in its factored form. Starting from $\mathbf{Q}_1 = \mathbf{I}_{s_0}$ we apply the recurrence relation

$$\mathbf{Q}_{n+1} := \left(\begin{array}{c|c} \mathbf{Q}_n & \mathbf{0}_{t_n \times s_n} \\ \hline \mathbf{0}_{s_n \times t_n} & \mathbf{I}_{s_n} \end{array}\right) \mathbf{U}_n \,, \tag{19}$$

where

$$\mathbf{U}_n :\equiv \left(\begin{array}{c|c} \mathbf{I}_{t_{n-1}} & \mathbf{0}_{t_{n-1} \times (s_{n-1}+s_n)} \\ \hline \mathbf{0}_{(s_{n-1}+s_n) \times t_{n-1}} & \widehat{\mathbf{U}}_n \end{array}\right) . \tag{20}$$

Here $\widehat{\mathbf{U}}_n$ is a unitary $(s_{n-1} + s_n) \times (s_{n-1} + s_n)$ matrix, which still needs to be determined. Once the sequence of unitary transformations $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_n$ will be known, it will be possible to compute $\mathbf{Q}_{n+1}$ with a simple scheme. Assume that the $t_n \times t_n$ matrix $\mathbf{Q}_n$ has the form

$$\mathbf{Q}_n \equiv: \begin{pmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \ldots & \mathbf{q}_{n-2} & \widetilde{\mathbf{q}}_{n-1} \end{pmatrix} ,$$

where $\mathbf{q}_i \in \mathbb{C}^{t_n \times s_i}$, and that

$$\widehat{\mathbf{U}}_n :\equiv \left(\begin{array}{c} \widehat{\mathbf{U}}_{n,u} \\ \hline \widehat{\mathbf{U}}_{n,d} \end{array} \right) :\equiv \left(\begin{array}{c|c} \widehat{\mathbf{U}}_{n,u,l} & \widehat{\mathbf{U}}_{n,u,r} \\ \hline \widehat{\mathbf{U}}_{n,d,l} & \widehat{\mathbf{U}}_{n,d,r} \end{array} \right) , \tag{21}$$

where

$\widehat{\mathbf{U}}_{n,u}$ is an $s_{i-1} \times (s_{i-1} + s_i)$ matrix,
$\widehat{\mathbf{U}}_{n,d}$ is an $s_i \times (s_{i-1} + s_i)$ matrix,
$\widehat{\mathbf{U}}_{n,u,l}$ is an $s_{i-1} \times s_{i-1}$ matrix,
$\widehat{\mathbf{U}}_{n,u,r}$ is an $s_{i-1} \times s_i$ matrix,
$\widehat{\mathbf{U}}_{n,d,l}$ is an $s_i \times s_{i-1}$ matrix,
$\widehat{\mathbf{U}}_{n,d,r}$ is an $s_i \times s_i$ matrix,

then

$$\mathbf{Q}_{n+1} = \left(\begin{array}{c|c|c} \mathbf{q}_0 \ \cdots \ \mathbf{q}_{n-2} & \widetilde{\mathbf{q}}_{n-1}\widehat{\mathbf{U}}_{n,u,l} & \widetilde{\mathbf{q}}_{n-1}\widehat{\mathbf{U}}_{n,u,r} \\ \hline \mathbf{0} \ \cdots \ \mathbf{0} & \widehat{\mathbf{U}}_{n,d,l} & \widehat{\mathbf{U}}_{n,d,r} \end{array} \right) . \tag{22}$$

In particular, the matrix $\mathbf{Q}_{n+1}$ has a trapezoidal structure. The last equation yields the following update algorithm, which we describe using MATLAB notation:

Algorithm 3 (CONSTRUCTION OF $\mathbf{Q}_{n+1}$)

Let $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_n$ *be the sequence of unitary matrices appearing in* (20)*, and let* $\mathbf{Q}_{n+1} := \mathbf{I}_{t_{n+1}}$*. For recursively constructing* $\mathbf{Q}_{n+1}$ *apply, for* $i = 1, \ldots, n$*,*

- *the upper part of* $\widehat{\mathbf{U}}_i$*:*

$$\mathbf{Q}_{n+1}\left(1 : t_i, t_{i-1}+1 : t_{i+1}\right) := \mathbf{Q}_{n+1}\left(1 : t_i, t_{i-1}+1 : t_i\right) \widehat{\mathbf{U}}_{i,u} , \tag{23}$$

- *the lower part of* $\widehat{\mathbf{U}}_i$*:*

$$\mathbf{Q}_{n+1}\left(t_i + 1 : t_{i+1}, t_{i-1}+1 : t_{i+1}\right) := \widehat{\mathbf{U}}_{i,d} . \tag{24}$$

In practice, whenever $\mathbf{Q}_{n+1}$ has to be applied to a vector or a block vector we can apply it in the factored form defined by this algorithm.

In view of (19) and (20) the multiplication of $\underline{\mathbf{T}}_n \mathbf{P}_n$ by $\mathsf{block\ diag}\,(\mathbf{Q}_n^{\mathsf{H}}, \mathbf{I}_{s_n})$ annihilates all subdiagonal elements except those below the diagonal of the last s_{n-1} columns; or if we regard $\underline{\mathbf{T}}_n \mathbf{P}_n$ as a matrix with n block columns, it does not annihilate the subdiagonal elements in the last block column, i.e.,

$$\begin{pmatrix} \mathbf{Q}_n^{\mathsf{H}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{s_n} \end{pmatrix} \underline{\mathbf{T}}_n \mathbf{P}_n = \mathbf{U}_n \underline{\mathbf{R}}_n = \left(\begin{array}{cccccc} \widetilde{\boldsymbol{\alpha}}_0 & \widetilde{\boldsymbol{\beta}}_0 & \widetilde{\boldsymbol{\gamma}}_0 & & & \\ & \widetilde{\boldsymbol{\alpha}}_1 & \widetilde{\boldsymbol{\beta}}_1 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \widetilde{\boldsymbol{\gamma}}_{n-3} \\ & & & & \widetilde{\boldsymbol{\alpha}}_{n-2} & \widetilde{\boldsymbol{\beta}}_{n-2} \\ & & & & & \boldsymbol{\mu}_n \\ \hline & & & & & \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right). \tag{25}$$

For the entries in the last block column or the entries in the last s_{n-1} columns, respectively, we have in particular

$$\left(\begin{array}{c} \widetilde{\boldsymbol{\gamma}}_{n-3} \\ \widetilde{\boldsymbol{\beta}}_{n-2} \\ \boldsymbol{\mu}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right) = \left(\begin{array}{cc|c} \mathbf{I}_{s_{n-3}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \left(\begin{array}{cc|c} \widehat{\mathbf{U}}_{n-2}^{\mathsf{H}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{s_{n-1}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \left(\begin{array}{c} \mathbf{0}_{s_{n-3}\times s_{n-1}} \\ \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n \\ \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right). \tag{26}$$

For annihilating all subdiagonal elements we have to construct a unitary matrix $\widehat{\mathbf{U}}_n$ of order $s_{n-1} + s_n$ such that

$$\begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix} :\equiv \begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{pmatrix} = \widehat{\mathbf{U}}_n \begin{pmatrix} \widetilde{\boldsymbol{\alpha}}_{n-1} \\ \mathbf{0}_{s_n\times s_{n-1}} \end{pmatrix}, \tag{27}$$

where $\widetilde{\boldsymbol{\alpha}}_{n-1}$ is an upper triangular nonsingular $s_{n-1} \times s_{n-1}$ block. This is just another QR decomposition. Potentially the left-hand side of (27) could be rank-deficient. For $\boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n$ this is true in case of deflation, but for the whole left-hand side this can be seen to be impossible, see [8]. Altogether, the relations (26) and (27) yield the following algorithm.

Algorithm 4 (Block update scheme for the QR decomposition)

By applying a sequence of unitary matrices $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_m$ to the block tridiagonal matrix $\underline{\mathbf{T}}_m\mathbf{P}_m$ of (17) the upper triangular matrix $\mathbf{R}_m$ of (18) is recursively constructed as follows. For $n = 1, \ldots, m$:

(1) Let $\widetilde{\boldsymbol{\alpha}} := \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n$, and let $\widetilde{\boldsymbol{\beta}} := \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n$ if $n > 1$.

If $n > 2$, *apply* $\widetilde{\mathbf{U}}^{\mathsf{H}}_{n-2}$ *to two blocks of the new last block column of* $\underline{\mathbf{T}}_n$*:*

$$\begin{pmatrix} \widetilde{\boldsymbol{\gamma}}_{n-3} \\ \widetilde{\boldsymbol{\beta}} \end{pmatrix} := \widehat{\mathbf{U}}^{\mathsf{H}}_{n-2} \begin{pmatrix} \mathbf{0}_{s_{n-3}\times s_{n-1}} \\ \widetilde{\boldsymbol{\beta}} \end{pmatrix} ;$$

if $n > 1$, *apply* $\widetilde{\mathbf{U}}^{\mathsf{H}}_{n-1}$ *to two blocks of the last block column of* $\mathbf{U}^{\mathsf{H}}_{n-2}\underline{\mathbf{T}}_n$*:*

$$\begin{pmatrix} \widetilde{\boldsymbol{\beta}}_{n-2} \\ \boldsymbol{\mu}_n \end{pmatrix} := \widehat{\mathbf{U}}^{\mathsf{H}}_{n-1} \begin{pmatrix} \widetilde{\boldsymbol{\beta}} \\ \widetilde{\boldsymbol{\alpha}} \end{pmatrix} .$$

(2) Compute $\widehat{\mathbf{U}}_n$ *and* $\widetilde{\boldsymbol{\alpha}}_{n-1}$ *by the QR decomposition* (27) *(as described in Section* 6*).*
(3) If needed, construct $\mathbf{Q}_{n+1}$ according to Algorithm 3.

There are various ways to construct the QR decomposition (27). Assuming $s_{n-1} = s_n = s$ and a parallel computer, Vital [7, pp. 115-116] suggests to distribute the necessary s^2 Givens rotations in such a way on s processors that at most $2s - 1$ are done on a single processor; but results need to be broadcast after each rotation, so for today's parallel computers the tasks are much too small, that is, the parallelism is much too fine grain to be effective. Freund and Malhotra [10] apply Givens rotations column by column since they construct $\underline{\mathbf{T}}_n$ column by column, and since they want to solve a problem with a single right-hand side as a special case of the general block problem [24]. As mentioned above, in the case of a single right-hand side it is enough to apply one Givens rotation per iteration. However, as we show next, when several (or even many) right-hand sides are treated, it is more efficient to use a product of possibly complex Householder reflections.

5 Complex Householder reflections

In this section, following Wilkinson [13, pp. 49-50], we summarize the basic formulas for complex Householder reflections and mention some of the implementation options.

For any nonzero $\mathbf{v} \in \mathbb{C}^n$ the matrix

$$\mathbf{H}_{\mathbf{v}} :\equiv \mathbf{I}_n - 2\frac{\mathbf{v}\,\mathbf{v}^{\mathsf{H}}}{\langle \mathbf{v}, \mathbf{v} \rangle} = \mathbf{I}_n + \beta\,\mathbf{v}\,\mathbf{v}^{\mathsf{H}} \tag{28}$$

with $\beta = -2/\langle \mathbf{v}, \mathbf{v} \rangle \in \mathbb{R}$, is called a Householder reflection. It describes a reflection at the complimentary subspace orthogonal to $\mathbf{v}$. We note that $\mathbf{H}_{\mathbf{v}}$

is Hermitian and unitary, i.e. $\mathbf{H}_\mathbf{v}^\mathsf{H} = \mathbf{H}_\mathbf{v}$ and $\mathbf{H}_\mathbf{v}\mathbf{H}_\mathbf{v}^\mathsf{H} = \mathbf{I}_n$. A vector $\mathbf{y} \in \mathbb{C}^n$ is mapped to

$$\mathbf{H}_\mathbf{v}\mathbf{y} = \mathbf{y} + \beta\mathbf{v}\left\langle \mathbf{v}, \mathbf{y}\right\rangle. \tag{29}$$

Householder's goal — extended to the complex case — was to choose $\mathbf{v}$ depending on $\mathbf{y}$ such that

$$\mathbf{H}_\mathbf{v}\mathbf{y} = \alpha\mathbf{e}_1 \tag{30}$$

for some $\alpha \in \mathbb{C}$. As $\mathbf{H}_\mathbf{v}$ is unitary, $|\alpha| = \|\mathbf{y}\|_2$. We can exclude that $\mathbf{y}$ is a multiple of $\mathbf{e}_1$, since the choice $\mathbf{v} := \mathbf{0}$, $\mathbf{H}_\mathbf{v} := \mathbf{I}$ would suffice in this case. Inserting (29) yields

$$\mathbf{y} - \alpha\mathbf{e}_1 = -\beta\left\langle \mathbf{v}, \mathbf{y}\right\rangle \mathbf{v}. \tag{31}$$

In particular $\mathbf{v} \in \mathsf{span}\left\{\mathbf{y} - \alpha\mathbf{e}_1\right\}$. As $\mathbf{H}_\mathbf{v} = \mathbf{H}_{\lambda\mathbf{v}}$ for all $\lambda \in \mathbb{C}\setminus\{0\}$ we can choose $\mathbf{v} = \mathbf{y} - \alpha\mathbf{e}_1$ without loss of generality. By (31) this choice implies

$$\left\langle \mathbf{v}, \mathbf{y}\right\rangle = -\beta^{-1} \in \mathbb{R}.$$

What remains is to determine the argument of α. Assume $y_1 \neq 0$ first, and let $y_1 = |y_1|e^{i\theta}$ and $\alpha = \|\mathbf{y}\|_2 e^{i\theta_\alpha}$. Then

$$\left\langle \mathbf{v}, \mathbf{y}\right\rangle = \left\langle \mathbf{y} - \alpha\mathbf{e}_1, \mathbf{y}\right\rangle = \|\mathbf{y}\|_2^2 - \|\mathbf{y}\|_2 e^{-i\theta_\alpha}e^{i\theta}|y_1|.$$

So either $\alpha = +\|\mathbf{y}\|_2 e^{i\theta}$ or $\alpha = -\|\mathbf{y}\|_2 e^{i\theta}$. With the first choice cancellation may occur in the first component of $\mathbf{v}$. The second choice is better and yields

$$-\beta^{-1} = \left\langle \mathbf{v}, \mathbf{y}\right\rangle = \|\mathbf{y}\|_2\left(\|\mathbf{y}\|_2 + |y_1|\right).$$

If $y_1 = 0$, then $\alpha\mathbf{e}_1 \perp \mathbf{y}$ and the value of α has no effect on $\left\langle \mathbf{v}, \mathbf{y}\right\rangle = \|\mathbf{y}\|_2^2$. So choosing, e.g., $e^{i\theta} = 1$ is fine.

Algorithm 5 (Implicit construction of $\mathbf{H}_\mathbf{v}$)

Given $\mathbf{y} = \begin{pmatrix} y_1 & \dots & y_n \end{pmatrix}^\mathsf{T} \in \mathbb{C}^n\setminus\{\mathbf{0}\}$ *that is not a multiple of* $\mathbf{e}_1$*, a Householder reflection* $\mathbf{H}_\mathbf{v}$ *satisfying* $\mathbf{H}_\mathbf{v}\mathbf{y} = \alpha\mathbf{e}_1$ *is constructed as follows:*

- *If* $y_1 \neq 0$*, let* $e^{i\theta} := y_1/|y_1|$*; otherwise let* $e^{i\theta} := 1$.
- *Compute* α *and* β *according to*

$$\alpha := -\|\mathbf{y}\|_2 e^{i\theta}, \qquad \beta := \frac{-1}{\|\mathbf{y}\|_2\left(\|\mathbf{y}\|_2 + |y_1|\right)}. \tag{32}$$

- *Compute* $\mathbf{v} := \mathbf{y} - \alpha\mathbf{e}_1$.

For evaluating $\mathbf{H}_\mathbf{v}\mathbf{y}$ for some $\mathbf{y}$ it is not necessary to compute the actual matrix $\mathbf{H}_\mathbf{v}$. It is much more economical and accurate to store only $\mathbf{v}$, to compute β according to (32), and to apply (29). Parlett [25] presents a thorough discussion of the choice of the sign of α when computing Householder reflections.

Lehoucq [26] compares different variants for the choice of the vector $\mathbf{v}$ and the corresponding coefficient β. He specifically compares the different ways of computing $\mathbf{H}_\mathbf{v}$ in EISPACK, LINPACK, NAG and LAPACK. Golub and van Loan [27] also discuss the real case at length.

6 QR decomposition of a lower banded matrix

In this section we describe a particularly efficient way of computing the QR decomposition (27), which capitalizes on the trapezoidal structure of the matrix $\boldsymbol{\nu}_n :\equiv \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n$. Recall that $\boldsymbol{\mu}_n$ is $s_{n-1} \times s_{n-1}$, while $\boldsymbol{\nu}_n$ is $s_n \times s_{n-1}$. For example, if $s_{n-1} = 5$ and $s_n = 4$,

$$\begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix} = \left(\begin{array}{ccccc} \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \hline \circ & \circ & \circ & \circ & \circ \\ & \circ & \circ & \circ & \circ \\ & & \circ & \circ & \circ \\ & & & \circ & \circ \end{array}\right) .$$

We determine s_{n-1} Householder reflections $\mathbf{H}_{1,n}, \ldots, \mathbf{H}_{s_{n-1},n}$ such that

$$\begin{pmatrix} \widetilde{\boldsymbol{\alpha}}_{n-1} \\ \mathbf{0}_{s_n \times s_{n-1}} \end{pmatrix} = \mathbf{H}_{s_{n-1},n} \ldots \mathbf{H}_{1,n} \begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix}, \tag{33}$$

where $\widetilde{\boldsymbol{\alpha}}_{n-1}$ is an upper triangular matrix. In particular

$$\widehat{\mathbf{U}}_n = \mathbf{H}_{1,n} \ldots \mathbf{H}_{s_{n-1},n} \, . \tag{34}$$

Assume that reflections $\mathbf{H}_{1,n}, \mathbf{H}_{2,n}$ have been computed such that

$$\mathbf{H}_{2,n}\mathbf{H}_{1,n}\begin{pmatrix}\boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n\end{pmatrix} = \left(\begin{array}{ccccc} \circ & \circ & \circ & \circ & \circ \\ & \circ & \circ & \circ & \circ \\ & & \bullet & \circ & \circ \\ & & \bullet & \circ & \circ \\ & & \bullet & \circ & \circ \\ \hline & & \bullet & \circ & \circ \\ & & \bullet & \circ & \circ \\ & & \bullet & \circ & \circ \\ & & & \circ & \circ \end{array}\right).$$

The highlighted section of the third column determines the next Householder reflection. In step i this vector has the size

$$l_{i,n} = \underbrace{s_{n-1} - i + 1}_{\text{size of upper part}} + \underbrace{\min(i, s_n)}_{\text{size of lower part}},$$

and the last entry is in row

$$e_{i,n} = l_{i,n} + i - 1.$$

In this example we have $i = 3$ and

$$l_{3,n} = 5 - 3 + 1 + 3 = 6, \qquad e_{3,n} = 6 + 3 - 1 = 8.$$

Hence the Householder reflection that has to be applied to the ith column is given by

$$\mathbf{H}_{i,n} :\equiv \mathsf{diag}\left(\mathbf{I}_{i-1}, \widehat{\mathbf{H}}_{i,n}, \mathbf{I}_{s_n - \min(i,s_n)}\right),$$

where $\widehat{\mathbf{H}}_{i,n}$ is a Householder reflection in the sense of our original definition (28): a reflection at a hyperplane but in a space of dimension $l_{i,n}$ only. When applying this reflection we only compute those entries that are not invariant. In this example the first two and the last row will not be influenced at all. All we have to do is to apply the reflection $\widehat{\mathbf{H}}_{i,n}$ on the submatrix whose left column is exactly given by the vector generating $\widehat{\mathbf{H}}_{i,n}$. Here this submatrix is highlighted:

$$\left(\begin{array}{ccccc}
\circ & \circ & \circ & \circ & \circ \\
 & \circ & \circ & \circ & \circ \\
 & & \alpha & \bullet & \bullet \\
 & & & \bullet & \bullet \\
 & & & \bullet & \bullet \\
\hline
 & & & \bullet & \bullet \\
 & & & \bullet & \bullet \\
 & & & \bullet & \bullet \\
 & & & \circ & \circ
\end{array}\right)$$

After this submatrix has been updated we proceed with the construction of the next reflection.

Algorithm 6 (Implicit construction of $\widehat{\mathbf{U}}_n$)

Let $\boldsymbol{\mu}_n$ be an $s_{n-1} \times s_{n-1}$ block, $\boldsymbol{\nu}_n$ an upper trapezoidal $s_n \times s_{n-1}$ block, and

$$\mathbf{M} :\equiv \begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix}. \tag{35}$$

For implicitly constructing s_{n-1} Householder reflections such that (33) *holds we proceed for $i = 1, \ldots, s_{n-1}$ as follows:*

- *Compute $l_{i,n}$ and $e_{i,n}$:*

$$l_{i,n} :\equiv s_{n-1} - i + 1 + \min(i, s_n), \quad e_{i,n} :\equiv l_{i,n} + i - 1. \tag{36}$$

- *Construct by Algorithm* 5 the Householder reflection $\widehat{\mathbf{H}}_{i,n}$ that generates zeros in the ith row of $\mathbf{M}$ below the diagonal: i.e., compute β and $\mathbf{v}$ using the vector

$$\mathbf{y} := \mathbf{y}_{i,n} :\equiv \mathbf{M}(i : e_{i,n}, i). \tag{37}$$

- Apply $\widehat{\mathbf{H}}_{i,n}$ to the corresponding submatrix of $\mathbf{M}$:

$$\begin{aligned} &\mathbf{M}(i : e_{i,n}, i+1 : s_{n-1}) \\ &\quad = \mathbf{M}(i : e_{i,n}, i+1 : s_{n-1}) + \beta \mathbf{v} \left(\mathbf{v}^{\mathsf{H}} \mathbf{M}(i : e_{i,n}, i+1 : s_{n-1})\right). \end{aligned} \tag{38}$$

7 Adaptation to block Hessenberg matrices

So far we have only treated the QR decomposition of the block tridiagonal matrix that results from the symmetric (Hermitian) block Lanczos process. As the symmetry of $\underline{\mathbf{T}}_n$ has not been accounted for in our algorithms, all we said is also applicable to the QR decomposition of the block tridiagonal matrix generated by the nonsymmetric Lanczos algorithm and used by the block QMR method, as long as no look-ahead is needed and orthogonalization with respect to block vectors whose "child" was numerically rank deficient and has been reduced by inexact deflation is neglected.

For block GMRes we need to QR-decompose a block Hessenberg matrix however, but the modification needed in the above treated procedure are minor. They are to be discussed in this section. First, the block matrix that needs to be QR-decomposed is no longer $\underline{\mathbf{T}}_n\mathbf{P}_n$ of (17) but

$$\underline{\mathbf{H}}_n\mathbf{P}_n = \left(\begin{array}{cccc} \boldsymbol{\eta}_{0,0}\boldsymbol{\pi}_1 & \boldsymbol{\eta}_{0,1}\boldsymbol{\pi}_2 & \cdots & \boldsymbol{\eta}_{0,n-1}\boldsymbol{\pi}_n \\ \boldsymbol{\eta}_{1,0}\boldsymbol{\pi}_1 & \boldsymbol{\eta}_{1,1}\boldsymbol{\pi}_2 & \cdots & \boldsymbol{\eta}_{1,n-1}\boldsymbol{\pi}_n \\ & \boldsymbol{\eta}_{2,1}\boldsymbol{\pi}_2 & \ddots & \vdots \\ & & \ddots & \boldsymbol{\eta}_{n-1,n-1}\boldsymbol{\pi}_n \\ \hline & & & \boldsymbol{\eta}_{n,n-1}\boldsymbol{\pi}_n \end{array}\right) \in \mathbb{C}^{t_{n+1}\times t_n}\,, \tag{39}$$

and the resulting upper triangular factor $\underline{\mathbf{R}}_n$ is now full:

$$\underline{\mathbf{R}}_n :\equiv \left(\begin{array}{cccc} \widetilde{\boldsymbol{\eta}}_{0,0} & \widetilde{\boldsymbol{\eta}}_{0,1} & \cdots & \widetilde{\boldsymbol{\eta}}_{0,n-1} \\ & \widetilde{\boldsymbol{\eta}}_{1,1} & \cdots & \widetilde{\boldsymbol{\eta}}_{1,n-1} \\ & & \ddots & \vdots \\ & & & \widetilde{\boldsymbol{\eta}}_{n-1,n-1} \\ \hline \multicolumn{4}{c}{\mathbf{0}_{s_n\times t_n}} \end{array}\right) \in \mathbb{C}^{t_{n+1}\times t_n}\,. \tag{40}$$

When the unitary transformations $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_{n-1}$, which make up $\mathbf{Q}_n$ according to (19) and (20), have been applied to $\underline{\mathbf{H}}_n\mathbf{P}_n$, then as in (25), this matrix has been transformed into upper triangular form except for the last block column:

$$\begin{pmatrix} \mathbf{Q}_n^{\mathsf{H}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{s_n} \end{pmatrix} \underline{\mathbf{H}}_n \mathbf{P}_n = \mathbf{U}_n \underline{\mathbf{R}}_n = \left(\begin{array}{ccccc} \widetilde{\boldsymbol{\eta}}_{0,0} & \widetilde{\boldsymbol{\eta}}_{0,1} & \cdots & \cdots & \widetilde{\boldsymbol{\eta}}_{0,n-1} \\ & \widetilde{\boldsymbol{\eta}}_{1,1} & \cdots & \cdots & \widetilde{\boldsymbol{\eta}}_{1,n-1} \\ & & \ddots & & \vdots \\ & & & \widetilde{\boldsymbol{\eta}}_{n-2,n-2} & \widetilde{\boldsymbol{\eta}}_{n-2,n-1} \; \vdots \\ & & & & \boldsymbol{\mu}_n \\ \hline & & & & \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right). \tag{41}$$

However, because the last block column contains now not only four blocks but $n+1$, the computation of this block column, which is the first step in updating $\underline{\mathbf{R}}_n$, requires to apply the transformations $\widehat{\mathbf{U}}_1, \dots, \widehat{\mathbf{U}}_{n-1}$ to this block column from the top to the bottom: instead of (26) we have now

$$\left(\begin{array}{c} \widetilde{\boldsymbol{\eta}}_{0,n-1} \\ \widetilde{\boldsymbol{\eta}}_{1,n-1} \\ \vdots \\ \widetilde{\boldsymbol{\eta}}_{n-2,n-1} \\ \boldsymbol{\mu}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right) = \left(\begin{array}{cc|c} \mathbf{I}_{t_{n-2}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \cdots \left(\begin{array}{cc|c} \widehat{\mathbf{U}}_1^{\mathsf{H}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{t_n - s_0 - s_1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \left(\begin{array}{c} \boldsymbol{\eta}_{0,n-1}\boldsymbol{\pi}_n \\ \boldsymbol{\eta}_{1,n-1}\boldsymbol{\pi}_n \\ \vdots \\ \boldsymbol{\eta}_{n-2,n-1}\boldsymbol{\pi}_n \\ \boldsymbol{\eta}_{n-1,n-1}\boldsymbol{\pi}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right). \tag{42}$$

Here, the last two blocks on the left-hand side are processed as before in (27) and Section 6. Their QR decomposition has to be computed, and we do that by applying s_{n-1} Householder reflections. There is no modification of this part needed. As we have just seen, the adaptation from block tridiagonal to block Hessenberg requires a major modification of Algorithm 4 however. The relations (42) and (27) yield the following new version.

Algorithm 7 (Block update scheme for the QR decomposition)

By applying a sequence of unitary matrices $\widehat{\mathbf{U}}_1, \dots, \widehat{\mathbf{U}}_m$ to the block Hessenberg matrix $\underline{\mathbf{H}}_m \mathbf{P}_m$ of (39) *the upper triangular matrix $\mathbf{R}_m$ of* (40) *is recursively constructed as follows. For $n = 1, \dots, m$:*

(1) Set $\widetilde{\boldsymbol{\eta}}_{k,n-1} := \boldsymbol{\eta}_{k,n-1}\boldsymbol{\pi}_n$ $(k = 0, \dots, n-1)$. If $n > 1$, apply each of the $n-1$ unitary transformations $\widetilde{\mathbf{U}}_1^{\mathsf{H}}, \dots, \widetilde{\mathbf{U}}_{n-1}^{\mathsf{H}}$ to two blocks of the new last block column of $\underline{\mathbf{T}}_n$: for $k = 1, \dots, n-1$, redefine

$$\begin{pmatrix} \widetilde{\boldsymbol{\eta}}_{k-1,n-1} \\ \widetilde{\boldsymbol{\eta}}_{k,n-1} \end{pmatrix} := \widehat{\mathbf{U}}_k^{\mathsf{H}} \begin{pmatrix} \widetilde{\boldsymbol{\eta}}_{k-1,n-1} \\ \widetilde{\boldsymbol{\eta}}_{k,n-1} \end{pmatrix}.$$

(2) Let $\boldsymbol{\mu}_n := \widetilde{\boldsymbol{\eta}}_{n-1,n-1}$, $\boldsymbol{\nu}_n := \boldsymbol{\eta}_{n,n-1}\boldsymbol{\pi}_n$, and compute $\widehat{\mathbf{U}}_n$ and $\widetilde{\boldsymbol{\alpha}}_{n-1}$ by the QR decomposition (27) *(as described in Section* 6*).*

(3) If needed, construct $\mathbf{Q}_{n+1}$ according to Algorithm 3.

8 Numerical experiments on Householder reflections vs. Givens rotations

Given an upper trapezoidal matrix $\mathbf{M}$ defined by (35) with $s_{n-1} = s_n = w$, a total of w Householder reflections, applied as described in Section 6, are an efficient way to construct the QR decomposition of $\mathbf{M}$. An alternative is to apply in a double loop a set of w Givens rotations per column, that is a total of w^2. Most important is that in Algorithm 6 the block-wise update in (38) by a single Householder reflection (applied to a whole submatrix of $\mathbf{M}$) must be replaced by applying w Givens rotations to suitable pairs of rows of this submatrix of $\mathbf{M}$. In this section we present numerical experiments performed with Matlab on an IBM ThinkPad T42 with a 1.7 GHz Centrino processor and 512 MByte of RAM running Windows XP and Matlab 6.5. (On an older IBM ThinkPad T23 with a 1.133 GHz Pentium III mobile processor with 640 MByte of RAM running Windows XP, and Matlab 7.0 the tests took roughly twice as long, but the relative performance was also a bit different.) In a first experiment we compare the accuracy of both approaches.

Experiment 1 *We apply both methods for the QR decomposition — Householder reflections and Givens rotations — to a set of* 100 *random* 10×5 *upper trapezoidal matrices* $\mathbf{M}$*. The results are shown in Fig.* 3. The accuracy of both methods turns out to be on the same level: except in a few cases the Frobenius norms of $\mathbf{M} - \mathbf{QR}$ and of $\mathbf{Q}^{\mathsf{H}}\mathbf{Q} - \mathbf{I}$ are for both methods of the same magnitude. The norm of $\mathbf{Q}^{\mathsf{H}}\mathbf{Q} - \mathbf{I}$ is typically slightly smaller if Householder reflections are used. For the explicit computation of $\mathbf{Q}$ we have applied the unitary transformations to $\mathbf{I}_{10\times 10}$ as in (34).

Next we compare the computing time of the two approaches, but again limited to the decomposition of $\mathbf{M}$, not of the whole matrices $\underline{\mathbf{H}}_n$ or $\underline{\mathbf{T}}_n$. (Recall that in our application $\mathbf{M}$ is a submatrix of $\underline{\mathbf{H}}_n$ or $\underline{\mathbf{T}}_n$.) The time-critical second step of the updates involves the application of the so far constructed unitary transformations to the remaining columns of $\mathbf{M}$. The block Arnoldi and Lanczos processes described in Sections 2 and 3 allow us a block-wise construction of $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ as opposed to the column-wise construction of

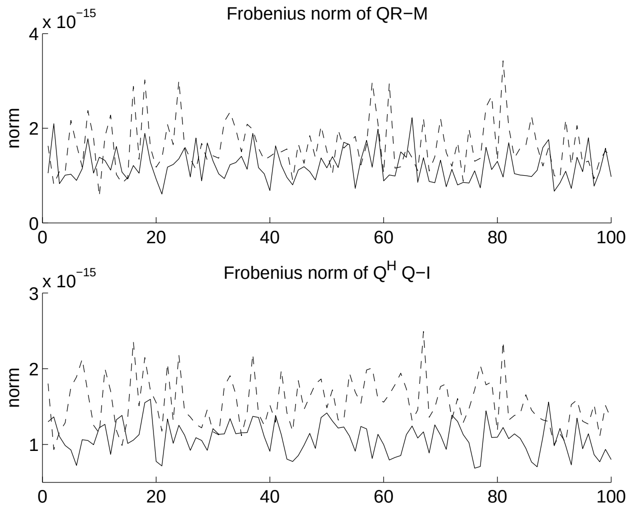

Fig. 3. Experiment 1: Accuracy of the QR decomposition of 100 random 10×5 upper trapezoidal matrices $\mathbf{M}$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations.

Ruhe [21] or Freund and Malhotra [10]. This block-wise construction enables us to update the QR decomposition of $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ block-wise. In particular, $\mathbf{M}$ is constructed at once, and we can profit from the possibility to update during the QR decomposition of $\mathbf{M}$ several columns at once as in (38).

We begin by comparing timings where during the QR decomposition of $\mathbf{M}$ the update of the block consisting of the remaining columns of $\mathbf{M}$ is done block-wise, in the sense that the Givens rotations are always applied to a pair of rows of the block, while the Householder reflections are applied in an explicit for-loop to all the columns of the block. This is an efficient implementation of applying Givens rotations, but not yet the optimal one for the Householder reflections.

EXPERIMENT 2 *In order to compare the speed of the two approaches we measure the* CPU *time for constructing the QR decomposition of* 100 *random matrices M of size $s_1 = s_2 = w$, using Householder reflections or Givens rotations, respectively, both applied block-wise (in the just described sense) to the remaining columns of* $\mathbf{M}$. *See Fig.* 4. For a matrix $\mathbf{M}$ of width 20 the Givens rotations turn out to be about 65% slower, but for small matrices the efficiency difference is small.

In contrast to Experiment 2, in the implementation of block QMR by Freund and Malhotra there is only a routine for mapping a single column vector by a set of Givens rotations, since in block QMR the matrix $\underline{\mathbf{T}}_n$ (and thus $\mathbf{M}$) is generated column-wise, and its QR decomposition is updated whenever a new column of $\underline{\mathbf{T}}_n$ becomes available. In particular, in the first iteration a vector of length $s + 1$ is obtained as first column of $\underline{\mathbf{T}}_n$. A set of s Givens rotations

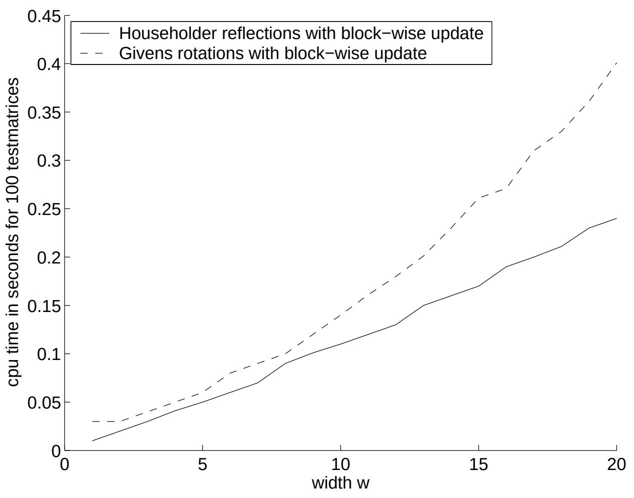

Fig. 4. Experiment 2: Computation time for the QR decomposition of 100 random $2w \times w$ upper trapezoidal matrices $\mathbf{M}$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations. In both cases we have updated the remaining columns block-wise using an explicit loop over the columns or pairs of rows, respectively.

is constructed to annihilate all except the first entry. In the next iteration a second vector of length $s + 2$ appears as second column, etc. At every stage we have to apply the previously constructed rotations (or reflections) to the newly constructed column. In our third experiment we mimic this approach by applying the Givens rotations in an explicit loop over the columns to just one column at a time. We compare this to applying Householder reflections to just one column at a time, which is an alternative for this case where $\underline{\mathbf{T}}_n$ (or $\underline{\mathbf{H}}_n$) is constructed column by column.

Experiment 3 *We measure the* CPU *time for constructing the QR decomposition of* 100 *random matrices* $\mathbf{M}$ *of size* $s_1 = s_2 = w$ *using, on the one hand, Givens rotations to update just one column at a time, and, on the other hand, Householder reflections to update one column at a time. See Fig.* 5. While in the first case the CPU time grows even much stronger with the block width w than in the Givens curve of Fig. 4 (note the different scales), it grows only about twice as fast as in the Householder curve of Fig. 4.

The final experiment concerns the QR decomposition of $\mathbf{M}$ with Householder reflections, where as proposed in this paper, the so far constructed transformations are applied block-wise at once to all the remaining columns of $\mathbf{M}$ using a BLAS2 operation as suggested by (38).

Experiment 4 *We measure the* CPU *time for constructing the QR decomposition of* 100 *random matrices* $\mathbf{M}$ *of size* $s_1 = s_2 = w$ *using Householder reflections with simultaneous update of the remaining columns by a vector-matrix operation as in* (38). *See Fig.* 6, where the result is compared with the

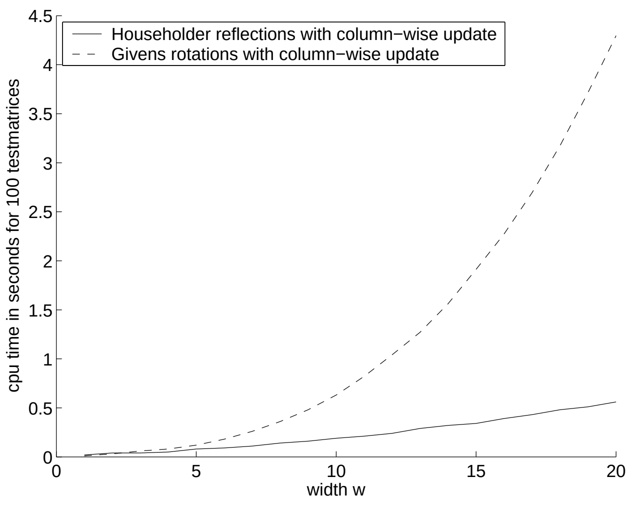

Fig. 5. Experiment 3: Computation time for the QR decomposition of 100 random $2w \times w$ upper trapezoidal matrices $\mathbf{M}$. In contrast to the block-wise updates of Experiment 3 and Fig. 4, timings for column-wise updates are shown here. The dashed line corresponds to using Givens rotations, while the solid line represents results gained by using Householder reflections.

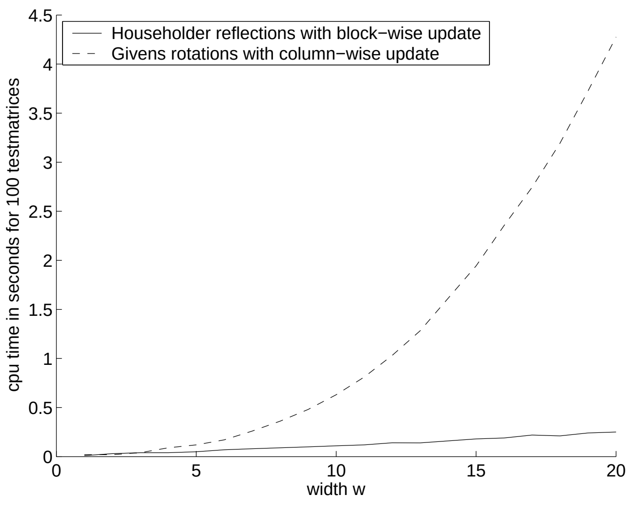

Fig. 6. Experiment 4: Computation time for the QR decomposition of 100 random $2w \times w$ upper trapezoidal matrices $\mathbf{M}$. In contrast to the column-wise updates of Experiment 3 and Fig. 4 (shown here again for Givens rotations as dashed line), the solid line represents now results gained by using Householder reflections and block-wise updates based on a vector-matrix product, that is, a BLAS2 operation.

one of Experiment 3 using Givens rotations to update a single column at a time. With our proposal the CPU time grows only about half as fast as in the Householder curve of Fig. 5. So the improvement over the Givens curve is even more striking (despite the fact that only $\mathbf{M}$ and not the full $\underline{\mathbf{T}}_n$ is decomposed here).

So the procedure proposed here is much faster than its alternatives if $w \geq 5$. The comparison of Experiments 2–4 shows that the difference is mainly due to avoiding column-wise updates with Givens rotations (which requires a double loop), and that the best performance is obtained when the remaining columns of $\mathbf{M}$ are updated by applying Householder reflections as BLAS2 operations. This fastest option does not exist if the columns of $\mathbf{M}$ become available only one at time as in Ruhe's implementation of block Lanczos.

We need to point out that these experiments only show a small part of the effect that can be noticed in the QR decomposition of a large extended block Hessenberg matrix $\underline{\mathbf{H}}_n$ or block tridiagonal matrix $\underline{\mathbf{T}}_n$ since we only treated the QR decomposition of the 2×1 block matrix $\mathbf{M}$ here. Actually, only the first block column of $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ consists just of $\mathbf{M}$. Later, once a new block column of $\underline{\mathbf{H}}_n$ or $\underline{\mathbf{T}}_n$ has been constructed, the so far determined Householder or Givens transformations need to be applied to this block column first (for constructing $\mathbf{M}$) before $\mathbf{M}$ can be QR-decomposed itself. In our approach this updating of the newly found block column is also done by BLAS2 operations analogous to the one in (38). This can be expected to manifest even bigger speedups than the QR decomposition of $\mathbf{M}$, in particular in the case of the block Hessenberg matrix $\underline{\mathbf{H}}_n$ from GMRes, where the new block columns are full. (In the case of a block triangular matrix $\underline{\mathbf{T}}_n$ some of the previously constructed Givens rotations have no effect on a new block column.)

9 Conclusions and generalizations

The symmetric block Lanczos process produces a banded matrix $\underline{\mathbf{T}}_n$, which is bordered in every step and which consists of a symmetric block tridiagonal matrix $\mathbf{T}_n$, extended by some additional rows. Additionally, a permutation matrix $\mathbf{P}_n$ is implicitly determined by the column pivoting. In a related paper [28] we discuss the need for those permutations and deflations in the symmetric block Lanczos process and give details and results for the block MinRes and block SymmLQ algorithms, which require to compute the full QR decomposition $\underline{\mathbf{T}}_n \mathbf{P}_n = \mathbf{Q}_n \underline{\mathbf{R}}_n$. The standard approach for this decomposition has been an update algorithm based on Givens rotations, a generalization of the well known update algorithm for tridiagonal matrices. It has been recommended to compute both $\underline{\mathbf{T}}_n$ and $\mathbf{R}_n$ column by column.

We promote instead a block-wise construction of $\underline{\mathbf{T}}_n$ and a block-wise update algorithm based on Householder reflections for the QR decomposition. It turns out that our QR decomposition is equally accurate as the one based on Givens rotations and that even on a serial computer it is much faster than column-wise updates with Givens rotations, the difference becoming more and more pronounced as the block size grows.

Our approach can be generalized quickly from the symmetric block Lanczos to the unsymmetric block Lanczos and the block Arnoldi processes, and from the QR decomposition of banded symmetric block tridiagonal matrices to the one of banded unsymmetric block tridiagonal matrices or block Hessenberg matrices as they come up in block QMR and block GMRes, respectively. In block GMRes the speedup will be even higher.

References

[1] C. C. Paige, M. A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.

[2] M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bureau Standards 49 (1952) 409–435.

[3] E. Stiefel, Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme, Comm. Math. Helv. 29 (1955) 157–179.

[4] Y. Saad, M. H. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, Math. Comp. 44 (1985) 417–424.

[5] R. W. Freund, N. M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numer. Math. 60 (1991) 315–339.

[6] R. W. Freund, M. H. Gutknecht, N. M. Nachtigal, An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, SIAM J. Sci. Comput. 14 (1993) 137–158.

[7] B. Vital, Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur, Ph.D. thesis, Université de Rennes (1990).

[8] T. Schmelzer, Block Krylov methods for Hermitian linear systems, Diploma thesis, Department of Mathematics, University of Kaiserslautern, Germany (2004).

[9] W. E. Boyse, A. A. Seidl, A block QMR method for computing multiple simultaneous solutions to complex symmetric systems, SIAM J. Sci. Comput. 17 (1) (1996) 263–274.

[10] R. W. Freund, M. Malhotra, A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides, Linear Algebra Appl. 254 (1997) 119–157.

[11] V. Simoncini, A stabilized QMR version of block BICG, SIAM J. Matrix Anal. Appl. 18 (2) (1997) 419–434.

[12] J. I. Aliaga, D. L. Boley, R. W. Freund, V. Hernández, A Lanczos-type method for multiple starting vectors, Math. Comp. 69 (232) (2000) 1577–1601.

[13] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, 1965.

[14] M. Ilić, I. W. Turner, Krylov subspaces and the analytic grade, Numer. Linear Algebra Appl. 12 (1) (2005) 55–76.

[15] J. Cullum, W. E. Donath, A block generalization of the symmetric s-step Lanczos algorithm, Tech. Rep. RC 4845, IBM T.J. Watson Research Center (May 1974).

[16] R. Underwood, An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems, Ph.D. thesis, Stanford University, Stanford, CA (1975).

[17] W. Kahan, B. N. Parlett, How far should you go with the Lanczos process, in: J. Bunch, D. Rose (Eds.), Sparse Matrix Computations, Academic Press, New York, 1976, pp. 131–144.

[18] G. H. Golub, R. Underwood, The block Lanczos method for computing eigenvalues, in: Mathematical Software, III (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, Wis., 1977), Academic Press, New York, 1977, pp. 361–377. Publ. Math. Res. Center, No. 39.

[19] D. P. O'Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322.

[20] Z. Bai, D. Day, Q. Ye, ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems, SIAM J. Matrix Anal. Appl. 20 (4) (1999) 1060–1082 (electronic), sparse and structured matrices and their applications (Coeur d'Alene, ID, 1996).

[21] A. Ruhe, Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, Math. Comp. 33 (146) (1979) 680–687.

[22] R. W. Freund, Computation of matrix Padé approximations of transfer functions via a Lanczos-type process, in: Approximation Theory VIII, Vol. 1 (College Station, TX, 1995), World Sci. Publishing, River Edge, NJ, 1995, pp. 215–222.

[23] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, Boston, 1996.

[24] R. W. Freund, QR Zerlegung im Lanczos Prozess, private note (2004).

[25] B. N. Parlett, Analysis of algorithms for reflectors in bisectors, SIAM Review 13 (1971) 197–208.

[26] R. B. Lehoucq, The computations of elementary unitary matrices, ACM Trans. Math. Software 22 (1996) 393–400.

[27] G. H. Golub, C. F. van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.

[28] T. Schmelzer, M. H. Gutknecht, Block Krylov space methods for indefinite Hermitian linear systems, to appear.