Available online at www.sciencedirect.com
ScienceDirect
Applied Numerical Mathematics 58 (2008) 871–883
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

Updating the QR decomposition of block tridiagonal and block Hessenberg matrices

Martin H. Gutknecht [a,*], Thomas Schmelzer [b]

[a] *Seminar for Applied Mathematics, ETH Zurich, ETH-Zentrum HG, CH-8092 Zurich, Switzerland*
[b] *Oxford University Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1 3UQ, United Kingdom*

Available online 29 April 2007

Abstract

We present an efficient block-wise update scheme for the QR decomposition of block tridiagonal and block Hessenberg matrices. For example, such matrices come up in generalizations of the Krylov space solvers MINRES, SYMMLQ, GMRES, and QMR to block methods for linear systems of equations with multiple right-hand sides. In the non-block case it is very efficient (and, in fact, standard) to use Givens rotations for these QR decompositions. Normally, the same approach is also used with column-wise updates in the block case. However, we show that, even for small block sizes, block-wise updates using (in general, complex) Householder reflections instead of Givens rotations are far more efficient in this case, in particular if the unitary transformations that incorporate the reflections determined by a whole block are computed explicitly. Naturally, the bigger the block size the bigger the savings. We discuss the somewhat complicated algorithmic details of this block-wise update, and present numerical experiments on accuracy and timing for the various options (Givens vs. Householder, block-wise vs. column-wise update, explicit vs. implicit computation of unitary transformations). Our treatment allows variable block sizes and can be adapted to block Hessenberg matrices that do not have the special structure encountered in the above mentioned block Krylov space solvers.

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. Introduction

We investigate algorithms for computing and updating the full QR decomposition of block Hessenberg matrices and block triangular matrices with upper triangular or upper trapezoidal subdiagonal blocks. For example, such block Hessenberg matrices of increasing size come up in the block GMRES method introduced by Vital [22]. Likewise, block triangular matrices are successively built up in the block versions of MINRES and SYMMLQ investigated in detail in [18], and in the block versions of QMR[1] introduced by various authors [4,7,20,1]. We will briefly sketch this application first in order to motivate the treatment of matrices with this structure and to fix our notation. Matrices with full subdiagonal blocks can be dealt with by either modifying our procedures to this case or by (possibly parallel)

* Corresponding author.
E-mail addresses: mhg@sam.math.ethz.ch (M.H. Gutknecht), thoms@comlab.ox.ac.uk (T. Schmelzer).
URLs: http://www.sam.math.ethz.ch/~mhg (M.H. Gutknecht), http://web.comlab.ox.ac.uk/thomas.schmelzer (T. Schmelzer).

[1] We assume QMR without look-ahead here.

0168-9274/$30.00
doi:10.1016/j.apnum.2007.04.010

preliminary computation of QR decompositions of these subdiagonal blocks, which allow us to reduce the general case to the one with upper triangular or upper trapezoidal blocks treated here.

Block Krylov space solvers can be used for solving linear systems of equations with multiple right-hand sides, which we write as $\mathbf{A}\mathbf{x} = \mathbf{b}$ with

$$\mathbf{A} \in \mathbb{C}^{N\times N}, \quad \mathbf{x} \in \mathbb{C}^{N\times s}, \quad \mathbf{b} \in \mathbb{C}^{N\times s}. \tag{1}$$

We refer to $\mathbf{x}$ and $\mathbf{b}$ as *block vectors* and denote their columns by $\mathbf{x}^{(i)}$ and $\mathbf{b}^{(i)}$ $(i = 1, \ldots, s)$, respectively. The nth approximations $\mathbf{x}_n^{(i)}$ of the s solutions of $\mathbf{A}\mathbf{x}_n^{(i)} = \mathbf{b}^{(i)}$ are chosen so that

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

where $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ is the initial block residual and

$$\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0) := \operatorname{block\,span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{n-1}\mathbf{r}_0\} \tag{2}$$

$$= \mathcal{K}_n\big(\mathbf{A}, \mathbf{r}_0^{(1)}\big) + \cdots + \mathcal{K}_n\big(\mathbf{A}, \mathbf{r}_0^{(s)}\big) \subseteq \mathbb{C}^N \tag{3}$$

is the sum of the s individual nth Krylov subspaces

$$\mathcal{K}_n\big(\mathbf{A}, \mathbf{r}_0^{(i)}\big) := \operatorname{span}\{\mathbf{r}_0^{(i)}, \mathbf{A}\mathbf{r}_0^{(i)}, \ldots, \mathbf{A}^{n-1}\mathbf{r}_0^{(i)}\}.$$

The block algorithms mentioned above apply the block Arnoldi process (in the non-Hermitian case) or the block Lanczos process (in the Hermitian case) to create block vectors $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{n-1}$ whose orthonormal columns are a basis for $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$. Here, when constructing $\mathbf{y}_n$, we have to delete those columns of $\mathbf{A}\mathbf{y}_{n-1}$ that are already contained in $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$. This is called *deflation*. It is crucial for the stability of block Lanczos, and it saves memory space and computing time in block Arnoldi. Deflation is accounted for, but not further investigated in this paper. As a consequence, the block vector $\mathbf{y}_i$ has s_i columns, where $s \geqslant s_0 \geqslant s_i \geqslant s_{i+1}$, $i = 1, 2, \ldots$. In practice, some rank-revealing QR decomposition has to be applied to determine the columns that are deleted. This decomposition implicitly determines an $s_n \times s_n$ permutation matrix $\boldsymbol{\pi}_n$ that accounts for the column pivoting.

Let us denote by $\mathsf{t}_n(\mathbf{A}, \mathbf{r}_0)$ the dimension of $\mathcal{B}_n(\mathbf{A}, \mathbf{r}_0)$, so that

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

Then, assuming exact deflation only, we can summarize the action of n steps of the *block Arnoldi process* as

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{H}}_n, \tag{4}$$

where $\mathbf{Y}_n := (\mathbf{y}_0\ \mathbf{y}_1\ \ldots\ \mathbf{y}_{n-1}) \in \mathbb{C}^{N\times \mathsf{t}_n}$ contains the orthonormal basis vectors of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$, and $\underline{\mathbf{H}}_n$ is an *extended block Hessenberg matrix* with an additional block row:

$$\underline{\mathbf{H}}_n := \left(\begin{array}{cccc} & \mathbf{H}_n & & \\ \hline \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{\eta}_{n,n-1} \end{array}\right) := \left(\begin{array}{cccc} \boldsymbol{\eta}_{0,0} & \boldsymbol{\eta}_{0,1} & \cdots & \boldsymbol{\eta}_{0,n-1} \\ \boldsymbol{\eta}_{1,0} & \boldsymbol{\eta}_{1,1} & \cdots & \boldsymbol{\eta}_{1,n-1} \\ & \boldsymbol{\eta}_{2,1} & \ddots & \vdots \\ & & \ddots & \boldsymbol{\eta}_{n-1,n-1} \\ \hline & & & \boldsymbol{\eta}_{n,n-1} \end{array}\right) \in \mathbb{C}^{\mathsf{t}_{n+1}\times \mathsf{t}_n}. \tag{5}$$

Here the above mentioned permutation matrices are encapsulated in the block coefficients $\boldsymbol{\eta}_{n,n-1}$. If we let $\mathbf{P}_n := \operatorname{block\,diag}(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n)$ be the permutation matrix that describes all these permutations, the block Hessenberg matrix $\underline{\mathbf{H}}_n\mathbf{P}_n$ has subdiagonal blocks $\boldsymbol{\eta}_{n,n-1}\boldsymbol{\pi}_n =: (\boldsymbol{\rho}_n\ \boldsymbol{\rho}_n^{\square})$ of upper trapezoidal shape, with a square, upper triangular $\boldsymbol{\rho}_n$.

We assume here that the block Arnoldi process generates the block vectors $\mathbf{y}_n$ at once and not column by column. This has two important advantages: it allows us to compute in the nth step s_{n-1} matrix-vector products at once and to permute the columns of the newly constructed block vector – a prerequisite for applying a rank-revealing QR decomposition for determining $\mathbf{y}_n$.

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.

For summarizing the resulting *symmetric block Lanczos process* the Arnoldi relation (4) is replaced by the Lanczos relation

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n, \tag{6}$$

where

$$\underline{\mathbf{T}}_n := \left(\begin{array}{c} \mathbf{T}_n \\ \hline \mathbf{0} \quad \ldots \quad \mathbf{0} \quad \beta_{n-1} \end{array}\right) := \left(\begin{array}{ccccc} \boldsymbol{\alpha}_0 & \boldsymbol{\beta}_0^{\mathsf{H}} & & & \\ \boldsymbol{\beta}_0 & \boldsymbol{\alpha}_1 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \boldsymbol{\beta}_{n-2}^{\mathsf{H}} \\ & & & \boldsymbol{\beta}_{n-2} & \boldsymbol{\alpha}_{n-1} \\ \hline & & & & \boldsymbol{\beta}_{n-1} \end{array}\right) \in \mathbb{C}^{\mathsf{t}_{n+1}\times \mathsf{t}_n} \tag{7}$$

and $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i^{\mathsf{H}}$ for all $i = 1, 2, \ldots, n-1$.

Nonsymmetric block Lanczos algorithms also make use of relation (6), but the columns of $\mathbf{Y}_n$ are in general no longer orthonormal and 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 [5,21,11,9]. The aim was to compute multiple eigenvalues and corresponding eigenspaces. The nonsymmetric block version was defined in [14] and has since been addressed in a number papers, e.g., [2]. For the symmetric case the column-wise version was proposed by Ruhe [16], for the nonsymmetric block Lanczos method it was advocated in various papers in the 1990ies, in particular [7,1,6]. The column-wise "Ruhe version" of block Arnoldi was stated in [17].

In this paper we describe in detail a block update procedures for the QR decomposition of $\underline{\mathbf{T}}_n\mathbf{P}_n$ and $\underline{\mathbf{Q}}_n\mathbf{P}_n$ based on Householder reflections and block operations. We will see – not unexpectedly – that the block-wise construction of the block Krylov space basis is not only preferable due to the possible parallelization of matrix–vector products and the option of pivoting, but that the corresponding block update procedure for the QR decomposition is far faster than column-wise updates. The best results are obtained with Householder reflections that are block-wise assembled to a unitary matrix applied at once (instead of applying a sequence of single Householder reflections). Typically, in the application of a block Krylov method the QR decomposition of $\underline{\mathbf{T}}_n\mathbf{P}_n$ or $\underline{\mathbf{Q}}_n\mathbf{P}_n$ is only a small cost factor, but when the block size is large, the savings may be substantial.

2. An update scheme for the QR decomposition of block tridiagonal matrices and upper block Hessenberg matrices

We turn now to the main subject: an efficient recursive QR decomposition of the extended block tridiagonal and block Hessenberg matrices $\underline{\mathbf{T}}_n$ and $\underline{\mathbf{H}}_n$. While in the case of a single system an extremely efficient update algorithm exists that needs only one new Givens rotation per step, the block case requires $\mathcal{O}(s^2)$ rotations per block step. We will show that for this block-wise update it is more efficient to apply Householder reflections instead. The major portion of the computing time reduction comes from the usage of BLAS2 or even BLAS3 block operations and is both software and hardware dependent. Using BLAS3 operations requires explicitly computing the unitary transformations incorporating the Householder reflections of whole blocks.

In the next four sections we describe in full detail this block update procedure that uses, in general, complex Householder reflections. The underlying ideas are quite straightforward generalizations and modifications of the QR update algorithms for the non-block cases, but the details are quite tedious and require a careful implementation. Finally, in Section 5, accuracy and timing experiments that confirm the superiority of the new approach will be presented.

As we mentioned before, the subdiagonal blocks of $\underline{\mathbf{H}}_n\mathbf{P}_n$ and $\underline{\mathbf{T}}_n\mathbf{P}_n$ are upper trapezoidal, a fact we will capitalize upon. We treat the block tridiagonal matrices $\underline{\mathbf{T}}_n$ generated by the symmetric Lanczos process first since their structure is more special. The generalization to the block Hessenberg matrices $\underline{\mathbf{H}}_n$ will be easy.

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

$$\underline{\mathbf{T}}_n\mathbf{P}_n = \left(\begin{array}{ccccc} \boldsymbol{\alpha}_0\boldsymbol{\pi}_1 & \boldsymbol{\beta}_0^{\mathsf{H}}\boldsymbol{\pi}_2 & & & \\ \boldsymbol{\beta}_0\boldsymbol{\pi}_1 & \boldsymbol{\alpha}_1\boldsymbol{\pi}_2 & \ddots & & \\ & \boldsymbol{\beta}_1\boldsymbol{\pi}_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n \\ & & & \ddots & \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n \\ \hline & & & & \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{array}\right) \in \mathbb{C}^{\mathsf{t}_{n+1}\times\mathsf{t}_n}, \tag{8}$$

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, $\boldsymbol{\beta}_i\boldsymbol{\pi}_{i+1} =: (\boldsymbol{\rho}_{i+1}\ \boldsymbol{\rho}_{i+1}^{\square})$

with a square, upper triangular $\boldsymbol{\rho}_{i+1} \cdot \underline{\mathbf{R}}_n$ has the form

$$\underline{\mathbf{R}}_n := \left(\begin{array}{ccccc} \tilde{\boldsymbol{\alpha}}_0 & \tilde{\boldsymbol{\beta}}_0 & \tilde{\boldsymbol{\gamma}}_0 & & \\ & \tilde{\boldsymbol{\alpha}}_1 & \tilde{\boldsymbol{\beta}}_1 & \ddots & \\ & & \ddots & \ddots & \tilde{\boldsymbol{\gamma}}_{n-3} \\ & & & \ddots & \tilde{\boldsymbol{\beta}}_{n-2} \\ & & & & \tilde{\boldsymbol{\alpha}}_{n-1} \\ \hline & & \mathbf{0}_{s_n\times\mathsf{t}_n} & & \end{array}\right) \in \mathbb{C}^{\mathsf{t}_{n+1}\times\mathsf{t}_n}, \tag{9}$$

where

$\tilde{\boldsymbol{\alpha}}_i$ is an $s_i \times s_i$ upper triangular block with full rank,
$\tilde{\boldsymbol{\beta}}_i$ is an $s_i \times s_{i+1}$ block, and
$\tilde{\boldsymbol{\gamma}}_i$ 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 need not decrease monotonously.

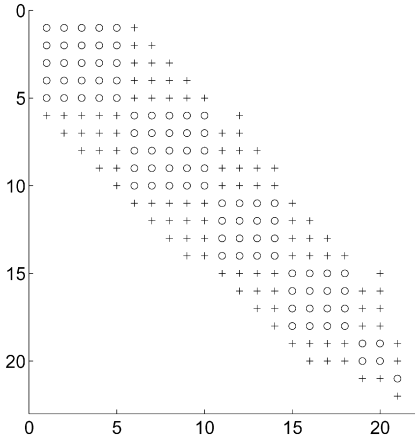

Fig. 1. The block tridiagonal structure of a matrix $\underline{\mathbf{T}}_n\mathbf{P}_n$.

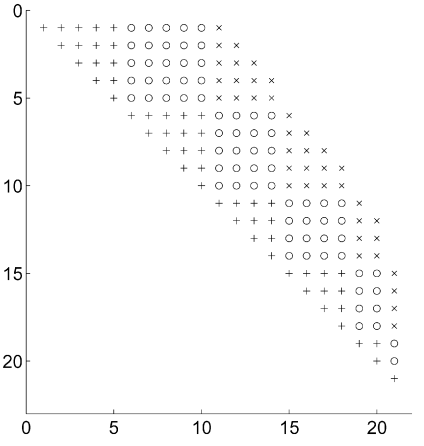

Fig. 2. The corresponding block structure of the matrix $\underline{\mathbf{R}}_n$.

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

$$\mathbf{Q}_{n+1} := \left(\begin{array}{c|c} \mathbf{Q}_n & \mathbf{0}_{\mathsf{t}_n \times s_n} \\ \hline \mathbf{0}_{s_n \times \mathsf{t}_n} & \mathbf{I}_{s_n} \end{array}\right) \mathbf{U}_n, \tag{10}$$

where

$$\mathbf{U}_n := \left(\begin{array}{c|c} \mathbf{I}_{\mathsf{t}_{n-1}} & \mathbf{0}_{\mathsf{t}_{n-1} \times (s_{n-1}+s_n)} \\ \hline \mathbf{0}_{(s_{n-1}+s_n) \times \mathsf{t}_{n-1}} & \widehat{\mathbf{U}}_n \end{array}\right). \tag{11}$$

Here $\widehat{\mathbf{U}}_n$ is a unitary matrix of order $s_{n-1} + s_n$ 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 $\mathsf{t}_n \times \mathsf{t}_n$ matrix $\mathbf{Q}_n$ has the form

$$\mathbf{Q}_n =: (\mathbf{q}_0\ \mathbf{q}_1\ \ldots\ \mathbf{q}_{n-2}\ \tilde{\mathbf{q}}_{n-1}),$$

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

$$\widehat{\mathbf{U}}_n := \left(\begin{array}{c} \widehat{\mathbf{U}}_{n,u} \\ \hline \widehat{\mathbf{U}}_{n,d} \end{array}\right), \tag{12}$$

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

$$\mathbf{Q}_{n+1} = \left(\begin{array}{ccc|c} \mathbf{q}_0 & \cdots & \mathbf{q}_{n-2} & \tilde{\mathbf{q}}_{n-1}\widehat{\mathbf{U}}_{n,u} \\ \hline \mathbf{0} & \ldots & \mathbf{0} & \widehat{\mathbf{U}}_{n,d} \end{array}\right). \tag{13}$$

In particular, the matrix $\mathbf{Q}_{n+1}$ has a trapezoidal structure.

In most applications the explicit computation of $\mathbf{Q}_{n+1}$ is neither needed nor recommended. However, if for some reason it is desired, the following update algorithm, which follows from (13) and is here described using MATLAB notation, could be used.

Algorithm 1 *(Explicit computation of $\mathbf{Q}_{n+1}$).*

Let $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_n$ be the sequence of unitary matrices appearing in (11), and let $\mathbf{Q}_{n+1} := \mathbf{I}_{\mathsf{t}_{n+1}}$. For recursively constructing $\mathbf{Q}_{n+1}$ apply, for $i = 1, \ldots, n$,

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

$$\mathbf{Q}_{n+1}\big(1 : \mathsf{t}_i, \mathsf{t}_{i-1} + 1 : \mathsf{t}_{i+1}\big) := \mathbf{Q}_{n+1}\big(1 : \mathsf{t}_i, \mathsf{t}_{i-1} + 1 : \mathsf{t}_i\big)\widehat{\mathbf{U}}_{i,u}, \tag{14}$$

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

$$\mathbf{Q}_{n+1}\big(\mathsf{t}_i + 1 : \mathsf{t}_{i+1}, \mathsf{t}_{i-1} + 1 : \mathsf{t}_{i+1}\big) := \widehat{\mathbf{U}}_{i,d}. \tag{15}$$

In view of (10) and (11) 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.,

$$\begin{pmatrix} \mathbf{Q}_n^{\mathsf{H}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{s_n} \end{pmatrix} \underline{\mathbf{T}}_n\mathbf{P}_n = \mathbf{U}_n\underline{\mathbf{R}}_n = \left(\begin{array}{ccccccc} \tilde{\boldsymbol{\alpha}}_0 & \tilde{\boldsymbol{\beta}}_0 & \tilde{\boldsymbol{\gamma}}_0 & & & & \\ & \tilde{\boldsymbol{\alpha}}_1 & \tilde{\boldsymbol{\beta}}_1 & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \tilde{\boldsymbol{\gamma}}_{n-3} & \\ & & & & \tilde{\boldsymbol{\alpha}}_{n-2} & \tilde{\boldsymbol{\beta}}_{n-2} & \\ \hline & & & & & \boldsymbol{\mu}_n & \\ & & & & & \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n & \end{array}\right). \tag{16}$$

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

$$\begin{pmatrix} \tilde{\boldsymbol{\gamma}}_{n-3} \\ \tilde{\boldsymbol{\beta}}_{n-2} \\ \boldsymbol{\mu}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{pmatrix} = \left(\begin{array}{cc|c} \mathbf{I}_{s_{n-3}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \left(\begin{array}{cc|c} \widehat{\mathbf{U}}_{n-2}^{\mathsf{H}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{s_{n-1}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I}_{s_n} \end{array}\right) \begin{pmatrix} \mathbf{0}_{s_{n-3}\times s_{n-1}} \\ \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n \\ \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n \\ \hline \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{pmatrix}. \tag{17}$$

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

$$\begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix} := \begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n \end{pmatrix} = \widehat{\mathbf{U}}_n \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{n-1} \\ \mathbf{0}_{s_n\times s_{n-1}} \end{pmatrix}, \tag{18}$$

where $\tilde{\boldsymbol{\alpha}}_{n-1}$ 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 (18) 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 as $\underline{\mathbf{T}}_n\mathbf{P}_n$ has full column rank. Altogether, the relations (17) and (18) yield the following algorithm.

Algorithm 2 *(Block update scheme for QR decomposition of $\underline{\mathbf{T}}_m$).*

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 (8) the upper triangular matrix $\mathbf{R}_m$ of (9) is recursively constructed as follows. For $n=1,\ldots,m$:

(1) Let $\tilde{\boldsymbol{\alpha}} := \boldsymbol{\alpha}_{n-1}\boldsymbol{\pi}_n$, and let $\tilde{\boldsymbol{\beta}} := \boldsymbol{\beta}_{n-2}^{\mathsf{H}}\boldsymbol{\pi}_n$ if $n>1$.
If $n>2$, apply $\widetilde{\mathbf{U}}_{n-2}^{\mathsf{H}}$ to two blocks of the new last block column of $\underline{\mathbf{T}}_n$:

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

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

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

(2) Compute $\widehat{\mathbf{U}}_n$ and $\tilde{\boldsymbol{\alpha}}_{n-1}$ by the QR decomposition (18) (as described in Section 4).
(3) If needed, construct $\mathbf{Q}_{n+1}$ according to Algorithm 1.

There are various ways to construct the QR decomposition (18). Assuming $s_{n-1}=s_n=s$ and a parallel computer, Vital [22, 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 [7] 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 [8]. 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.

For block GMRES we need to QR-decompose a block Hessenberg matrix however, but the modification needed in the above treated procedure is minor. The block matrix that needs to be QR-decomposed is no longer $\underline{\mathbf{T}}_n\mathbf{P}_n$ of (8) but

$$\underline{\mathbf{H}}_n\mathbf{P}_n = \left(\begin{array}{cccc} \boldsymbol{\eta}_{0,0}\boldsymbol{\pi}_1 & \boldsymbol{\eta}_{0,1}\boldsymbol{\pi}_2 & \cdots & \boldsymbol{\eta}_{0,n-1}\boldsymbol{\pi}_n \\ \boldsymbol{\eta}_{1,0}\boldsymbol{\pi}_1 & \boldsymbol{\eta}_{1,1}\boldsymbol{\pi}_2 & \cdots & \boldsymbol{\eta}_{1,n-1}\boldsymbol{\pi}_n \\ & \boldsymbol{\eta}_{2,1}\boldsymbol{\pi}_2 & \ddots & \vdots \\ & & \ddots & \boldsymbol{\eta}_{n-1,n-1}\boldsymbol{\pi}_n \\ \hline & & & \boldsymbol{\eta}_{n,n-1}\boldsymbol{\pi}_n \end{array}\right) \in \mathbb{C}^{\mathsf{t}_{n+1}\times\mathsf{t}_n}, \tag{19}$$

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

$$\underline{\mathbf{R}}_n := \left(\begin{array}{cccc} \tilde{\boldsymbol{\eta}}_{0,0} & \tilde{\boldsymbol{\eta}}_{0,1} & \cdots & \tilde{\boldsymbol{\eta}}_{0,n-1} \\ & \tilde{\boldsymbol{\eta}}_{1,1} & \cdots & \tilde{\boldsymbol{\eta}}_{1,n-1} \\ & & \ddots & \vdots \\ & & & \tilde{\boldsymbol{\eta}}_{n-1,n-1} \\ \hline & & \mathbf{0}_{s_n\times \mathtt{t}_n} & \end{array}\right) \in \mathbb{C}^{\mathtt{t}_{n+1}\times \mathtt{t}_n}. \tag{20}$$

In Algorithm 2 is was enough to apply at most two transformations $\widehat{\mathbf{U}}_{n-1}^{\mathsf{H}}$ and $\widehat{\mathbf{U}}_{n-2}^{\mathsf{H}}$ each step. Here we have to apply all previously constructed transformations $\widehat{\mathbf{U}}_1^{\mathsf{H}}, \ldots, \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}}$ in every step.

Algorithm 3 *(Block update scheme for QR decomposition of $\underline{\mathbf{H}}_m$).*

By applying a sequence of unitary matrices $\widehat{\mathbf{U}}_1, \ldots, \widehat{\mathbf{U}}_m$ to the block Hessenberg matrix $\underline{\mathbf{H}}_m\mathbf{P}_m$ of (19) the upper triangular matrix $\mathbf{R}_m$ of (20) is recursively constructed as follows. For $n = 1, \ldots, m$:

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

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

(2) Let $\boldsymbol{\mu}_n := \tilde{\boldsymbol{\eta}}_{n-1,n-1}$, $\boldsymbol{\nu}_n := \boldsymbol{\eta}_{n,n-1}\boldsymbol{\pi}_n$, and compute $\widehat{\mathbf{U}}_n$ and $\tilde{\boldsymbol{\alpha}}_{n-1}$ by the QR decomposition (18) (as described in Section 4).
(3) If needed, construct $\mathbf{Q}_{n+1}$ according to Algorithm 1.

---

3. Complex Householder reflections

In this section, following Wilkinson [23, 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

$$\mathbf{H}_{\mathbf{v}} := \mathbf{I}_n - 2\frac{\mathbf{v}\mathbf{v}^{\mathsf{H}}}{\langle \mathbf{v}, \mathbf{v}\rangle} = \mathbf{I}_n + \beta\mathbf{v}\mathbf{v}^{\mathsf{H}} \tag{21}$$

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

$$\mathbf{H}_{\mathbf{v}}\mathbf{y} = \mathbf{y} + \beta\mathbf{v}\langle \mathbf{v}, \mathbf{y}\rangle. \tag{22}$$

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

$$\mathbf{H}_{\mathbf{v}}\mathbf{y} = \alpha\mathbf{e}_1 \tag{23}$$

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 (22) yields

$$\mathbf{y} - \alpha\mathbf{e}_1 = -\beta\langle \mathbf{v}, \mathbf{y}\rangle\mathbf{v}. \tag{24}$$

In particular $\mathbf{v} \in \mathsf{span}\{\mathbf{y} - \alpha\mathbf{e}_1\}$. As $\mathbf{H}_{\mathbf{v}} = \mathbf{H}_{\lambda\mathbf{v}}$ for all $\lambda \in \mathbb{C}\backslash\{0\}$ we can choose $\mathbf{v} = \mathbf{y} - \alpha\mathbf{e}_1$ without loss of generality. By (24) this choice implies

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

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

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

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

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

If $y_1 = 0$, then $\alpha\mathbf{e}_1 \perp \mathbf{y}$ and the value of α has no effect on $\langle \mathbf{v}, \mathbf{y}\rangle = \|\mathbf{y}\|_2^2$. So choosing, e.g., $e^{i\theta} = 1$ is fine. Finally, a straightforward but lengthy calculation shows that the just determined values of α and β yield indeed (23).

Algorithm 4 *(Implicit construction of* $\mathbf{H}_\mathbf{v}$*).*

Given $\mathbf{y} = (y_1 \ \dots \ y_n)^\mathsf{T} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$ 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

$$\alpha := -\|\mathbf{y}\|_2 e^{i\theta}, \qquad \beta := \frac{-1}{\|\mathbf{y}\|_2(\|\mathbf{y}\|_2 + |y_1|)}. \tag{25}$$

- 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 might be more economical and accurate to store only $\mathbf{v}$ and β and to apply (22). Parlett [15] presents a thorough discussion of the choice of the sign of α when computing Householder reflections. Lehoucq [13] 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 [10] also discuss the real case at length.

4. QR decomposition of a lower banded matrix

In this section we describe a particularly efficient way of computing the QR decomposition (18) of two blocks from the diagonal and subdiagonal of $\underline{\mathbf{T}}_n\mathbf{P}_n$ or $\underline{\mathbf{H}}_n\mathbf{P}_n$. It capitalizes on the trapezoidal structure of the matrix $\boldsymbol{\nu}_n := \boldsymbol{\beta}_{n-1}\boldsymbol{\pi}_n$. 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 $\mathbf{H}_{1,n}, \dots, \mathbf{H}_{s_{n-1},n}$ such that

$$\begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{n-1} \\ \mathbf{0}_{s_n \times s_{n-1}} \end{pmatrix} = \underbrace{\mathbf{H}_{s_{n-1},n} \dots \mathbf{H}_{1,n}}_{\widehat{\mathbf{U}}_n^\mathsf{H}} \begin{pmatrix} \boldsymbol{\mu}_n \\ \boldsymbol{\nu}_n \end{pmatrix}, \tag{26}$$

where $\tilde{\boldsymbol{\alpha}}_{n-1}$ is an upper triangular matrix. 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 $l_{3,n} = 5 - 3 + 1 + 3 = 6$ and $e_{3,n} = 6 + 3 - 1 = 8$. Hence the Householder reflection that has to be applied to the ith column is given by

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

where $\widehat{\mathbf{H}}_{i,n}$ is a Householder reflection in the sense of our original definition (21): 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 5 *(QR decomposition of a lower banded matrix).*

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, $\mathbf{U}$ the identity matrix of dimension $s_n + s_{n-1}$ and

$$\mathbf{M} := \begin{pmatrix}\boldsymbol{\mu}_n\\ \boldsymbol{\nu}_n\end{pmatrix}. \tag{27}$$

For implicitly constructing s_{n-1} Householder reflections such that (26) holds we proceed for $i = 1, \ldots, s_{n-1}$ as follows:

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

$$l_{i,n} := s_{n-1} - i + 1 + \min(i, s_n), \quad e_{i,n} := l_{i,n} + i - 1. \tag{28}$$

- Construct by Algorithm 4 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

$$\mathbf{y} := \mathbf{y}_{i,n} := \mathbf{M}(i : e_{i,n}, i). \tag{29}$$

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

$$\mathbf{M}(i : e_{i,n}, i+1 : s_{n-1}) = \mathbf{M}(i : e_{i,n}, i+1 : s_{n-1}) + \beta \mathbf{v}\big(\mathbf{v}^{\mathsf{H}} \mathbf{M}(i : e_{i,n}, i+1 : s_{n-1})\big). \tag{30}$$

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

$$\mathbf{U}(i : e_{i,n}, i : i + l_{i,n} - 1) = \mathbf{U}(i : e_{i,n}, i : i + l_{i,n} - 1) + \beta \mathbf{v}\big(\mathbf{v}^{\mathsf{H}} \mathbf{U}(i : e_{i,n}, i : i + l_{i,n} - 1)\big). \tag{31}$$

Finally we let $\tilde{\boldsymbol{\alpha}}_{n-1} := \mathbf{M}(1 : s_{n-1}, 1 : s_{n-1})$ and $\widehat{\mathbf{U}}_n^{\mathsf{H}} := \mathbf{U}$.

In practice, whenever $\mathbf{Q}_{n+1}^{\mathsf{H}}$ has to be applied to a vector or a block vector, we can apply it in the factored form defined by (10) and (11) with the matrices $\widehat{\mathbf{U}}_n$ constructed here with the recursion (31) of Algorithm 5; see Step (1) of Algorithm 3. Of course, we could split up $\widehat{\mathbf{U}}_n$ further into a product of Householder operations or, likewise, in a product of Givens rotations. But by applying $\mathbf{Q}_{n+1}^{\mathsf{H}}$ in the factored form with explicitly computed factors $\widehat{\mathbf{U}}_n$ to several column vectors at once we can make use of BLAS3 operations, while splitting up $\widehat{\mathbf{U}}_n$ into Householder reflections would lead to BLAS2 operations only. Using a product of Givens rotations we would even end up with BLAS1 operations. The unitary matrix $\widehat{\mathbf{U}}_n$, which is a product of s_{n-1} Householder reflections, is an example of a block reflector. The efficiency of such block reflectors, in particular on parallel computers, has been pointed out and investigated before: see, e.g., Schreiber and Parlett [19], where similar block reflectors are used for the reduction to block Hessenberg form and Bischof and van Loan [3], where a special representation for block reflectors was introduced.

5. Numerical experiments on Householder reflections vs. Givens rotations

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. Given an upper trapezoidal matrix $\mathbf{M}$ defined by (27) with $s_{n-1} = s_n = w$, a total of w Householder reflections, applied as described in Section 4, 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. 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 (31).

The time spent for the decomposition of the matrices $\underline{\mathbf{H}}_n$ or $\underline{\mathbf{T}}_n$ is in many but not all cases almost negligible compared with the duration of the Arnoldi process dominated by the matrix vector multiplications and the inner products to orthogonalize the basis vectors. In fact the decomposition takes place completely in the projected coordinate space and hence solely depends on the dimension of the block Krylov space rather than the size or structure of $\mathbf{A}$. The relative cost of the QR decomposition clearly depends strongly on the problem and the implementation of its solution method, as well as on the hardware. Therefore, we only quantify the acceleration of the QR decomposition, and for this we work with matrices $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ filled with random entries.

In Algorithm 3 we have to apply each of $n-1$ unitary transformations $\widehat{\mathbf{U}}_1^{\mathsf{H}}, \ldots, \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}}$. These unitary transformations are available as products of Householder reflections or Givens rotations and could be applied in this way. However, it is more efficient to compute them explicitly by applying the unitary transformations to an identity matrix as in (31). Storing these unitary transformation requires about four times the space needed for storing just the Householder transformations, but this is still much less than storing the product $\mathbf{Q}_{n+1}$ of *all* so far computed Householder reflections. In other words, we want to apply the factorization of $\mathbf{Q}_{n+1}$ described in Algorithm 1.

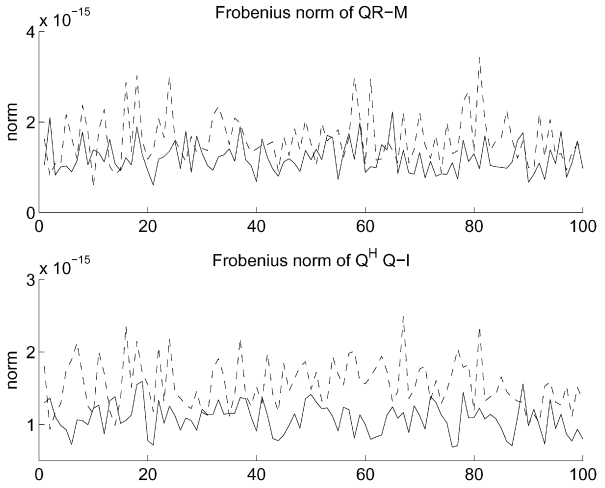

Fig. 3. Experiment 1: Accuracy of the QR decomposition of 100 random 10×5 upper trapezoidal matrices $\mathbf{M}$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations.

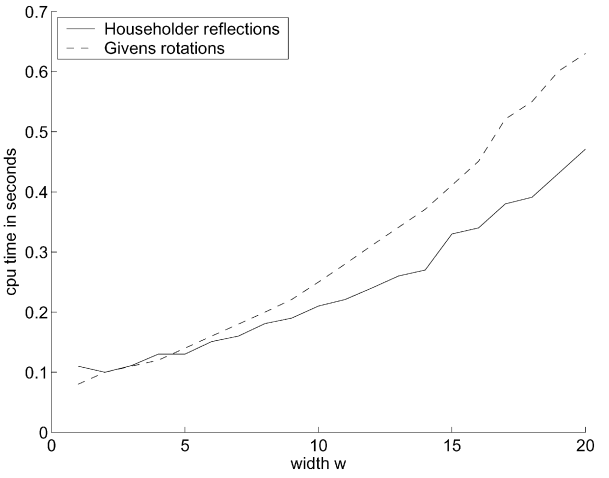

Fig. 4. Experiment 3: Computation time for the QR decomposition of $51w \times 50w$ block Hessenberg matrices $\underline{\mathbf{H}}_n$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations. In both cases the matrices $\widehat{\mathbf{U}}_i^{\mathsf{H}}$ were constructed explicitly.

Experiment 2. Given a block upper Hessenberg matrices $\underline{\mathbf{H}}_n$ of size $51w \times 50w$ with $w = 10$ we apply Algorithm 3 with and without explicit construction and application of the matrices $\widehat{\mathbf{U}}_i^{\mathsf{H}}$. The following average times in seconds have been measured for a set of (random) matrices.

Householder reflectors, explicit construction of $\widehat{\mathbf{U}}_i^{\mathsf{H}}$	0.21 sec.
Householder reflectors, implicit construction of $\widehat{\mathbf{U}}_i^{\mathsf{H}}$	0.96 sec.
Givens rotations, explicit construction of $\widehat{\mathbf{U}}_i^{\mathsf{H}}$	0.26 sec.
Givens rotations, implicit construction of $\widehat{\mathbf{U}}_i^{\mathsf{H}}$	1.41 sec.

The results show that the multiplication by individual Givens or Householder transformations is not as effective as the explicit construction of the matrices $\widehat{\mathbf{U}}_1^{\mathsf{H}}, \ldots, \widehat{\mathbf{U}}_{n-1}^{\mathsf{H}}$.

The experiment above also gives a first indication that Householder reflectors are indeed faster than Givens rotations. Using explicit construction we compare both methods for upper block Hessenberg matrices of different size.

Experiment 3. Given block upper Hessenberg matrices $\underline{\mathbf{H}}_n$ of size $51w \times 50w$ we apply Algorithm 3 with both methods for the QR decomposition for variable w. In both cases we construct the matrices $\widehat{\mathbf{U}}_i^{\mathsf{H}}$ explicitly. The results are shown in Fig. 4.

The difference between both methods is not very pronounced as a large fraction is spent in the first step of Algorithm 3 where both methods do not differ at all – as we apply the explicit construction. The situation is a bit different for block tridiagonal Lanczos matrices where the first step is less pronounced.

Experiment 4. Given block tridiagonal Lanczos matrices $\underline{\mathbf{T}}_n$ of size $51w \times 50w$ (for variable w) we apply Algorithm 2 with both methods for the QR decomposition. In both cases we use the explicit construction of $\widehat{\mathbf{U}}_i^{\mathsf{H}}$. See Fig. 5.

In all experiments so far we have assumed a block-wise construction of $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ as opposed to the column-wise construction of Ruhe [16] or Freund and Malhotra [7]. This block-wise construction enables us to update the QR decomposition of $\underline{\mathbf{H}}_n$ and $\underline{\mathbf{T}}_n$ block-wise and to apply pivoting in the Arnoldi or Lanczos process that yields these matrices.

In our last experiment we mimic this column-wise approach by replacing every operation acting on various columns by an explicit loop over the columns in which the operation is applied to just one column at a time.

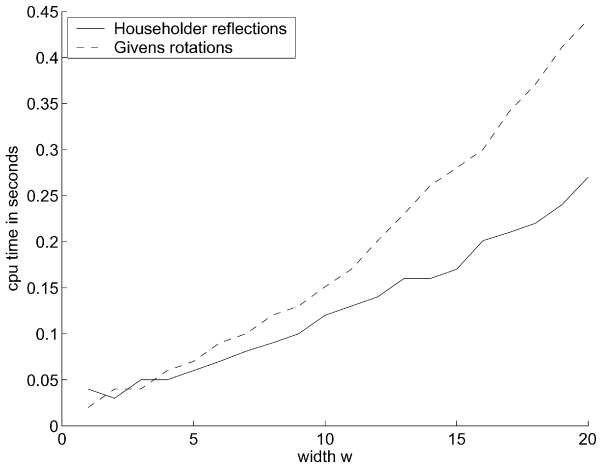

Fig. 5. Experiment 4: Computation time for the QR decomposition of $51w \times 50w$ block tridiagonal Lanczos matrices $\underline{\mathbf{T}}_n$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations. In both cases the matrices $\widehat{\mathbf{U}}_i^{\mathsf{H}}$ were constructed explicitly.

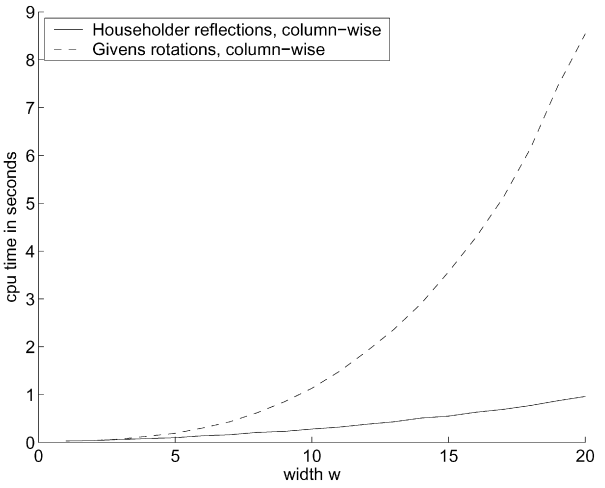

Fig. 6. Experiment 5: Computation time for the column-wise QR decomposition of $51w \times 50w$ block tridiagonal Lanczos matrices $\underline{\mathbf{T}}_n$. The solid line represents results gained by using Householder reflections. The dashed line corresponds to Givens rotations.

Experiment 5. Given block tridiagonal Lanczos matrices $\underline{\mathbf{T}}_n$ of size $51w \times 50w$ we mimic, for variable block width w, the column-wise computation or update of the QR decomposition using either one of the two methods. The results are shown in Fig. 6.

We see that here Householder reflections are much faster than Givens rotations; e.g., by a factor of about 8.5 for $w = 20$. However, noting that the scale on the y-axes in Figs. 5 and 6 differs by a factor of 20, we also see that the efficiency gain by using the block method of Algorithm 2 (or of Algorithm 3 in case of a block Hessenberg matrix) is of the same order of magnitude: e.g., if $w = 20$, we gain a factor of about 20 when using Givens rotations, and a factor of about 4 when using Householder reflections. In total, by replacing column-wise Givens rotations by block-wise assembled Householder rotations we gain a factor of more than 30 if $w = 20$.

The performance differences we noticed will be even much more pronounced if we have block sizes of the order of 100 or even more, as in some of Langou's examples [12].

6. Conclusions and generalizations

The standard approach for computing or updating the QR decomposition of a block tridiagonal matrix $\underline{\mathbf{T}}_n\mathbf{P}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n$ or a block Hessenberg matrix has been a column-wise update algorithm based on Givens rotations, a generalization of the well known update algorithm for tridiagonal or Hessenberg matrices. In the literature on block Krylov space methods it has also been recommended to compute $\underline{\mathbf{T}}_n$ column by column.

As is well known, QR factorizations based on either Givens rotations or Householder reflections produce the Q-factor successively in factored form. We show here that for highest efficiency one should not use the fully factored forms, as one loses parallel efficiency due to small grain parallelism. Explicitly multiplying together certain partial products of Householder or Givens transformations allows us to apply BLAS3 operations instead of BLAS2 (Householder) or BLAS1 (Givens) operations.

So we promote here a block-wise construction of $\underline{\mathbf{T}}_n$ and, for the QR decomposition, a block-wise update algorithm based on assembling the Householder reflections that belong to every block. 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. For example, for a block size of 20, the computing time for the decomposition was reduced by a factor of more than 30 in our simulations. Moreover, the block-wise execution of the block Arnoldi or block Lanczos process allows us to profit from the parallel computation of matrix–vector products.

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.

Acknowledgement

The authors are indebted to Walter Gander for pointing out the references [19] and [3].

References

[1] 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.
[2] Z. Bai, D. Day, Q. Ye, ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems, in: Sparse and Structured Matrices and Their Applications, Coeur d'Alene, ID, 1996, SIAM J. Matrix Anal. Appl. 20 (4) (1999) 1060–1082 (electronic).
[3] C. Bischof, C. Van Loan, The WY representation for products of Householder matrices, in: Parallel Processing for Scientific Computing, Norfolk, VA, 1985, SIAM J. Sci. Statist. Comput. 8 (1987) S2–S13.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] R.W. Freund, QR Zerlegung im Lanczos Prozess, private note, 2004.
[9] G.H. Golub, R. Underwood, The block Lanczos method for computing eigenvalues, in: Mathematical Software, III, Proc. Sympos., Math. Res. Center, Univ. Wisconsin, Madison, WI, 1977, Publ. Math. Res. Center, vol. 39, Academic Press, New York, 1977, pp. 361–377.
[10] G.H. Golub, C.F. van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[11] 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.
[12] J. Langou, For a few iterations less, talk presented at the Eighth Copper Montain Conference on Iterative Methods, Copper Mountain, CO, March 28–April 2, 2004.
[13] R.B. Lehoucq, The computations of elementary unitary matrices, ACM Trans. Math. Software 22 (1996) 393–400.
[14] D.P. O'Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322.
[15] B.N. Parlett, Analysis of algorithms for reflectors in bisectors, SIAM Rev. 13 (1971) 197–208.
[16] A. Ruhe, Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, Math. Comp. 33 (146) (1979) 680–687.
[17] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, Boston, 1996.
[18] T. Schmelzer, Block Krylov methods for Hermitian linear systems, Diploma thesis, Department of Mathematics, University of Kaiserslautern, Germany, 2004.
[19] R. Schreiber, B. Parlett, Block reflectors: theory and computation, SIAM J. Numer. Anal. 25 (1) (1988) 189–205.
[20] V. Simoncini, A stabilized QMR version of block BICG, SIAM J. Matrix Anal. Appl. 18 (2) (1997) 419–434.
[21] R. Underwood, An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems, Ph.D. thesis, Stanford University, Stanford, CA, 1975.
[22] B. Vital, Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur, Ph.D. thesis, Université de Rennes, 1990.
[23] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, Ely House, London, 1965.