

# A note on using compact WY representations for updating QR decompositions

Daniel Kressner

March 17, 2009

## Abstract

This note is concerned with computing the QR decomposition of a matrix  $\begin{bmatrix} A \\ B \end{bmatrix}$ , where  $A$  is already upper triangular. The compact WY representation for the Householder reflectors generated during such a decomposition have a particular structure, which is used to develop an efficient LAPACK-like implementation. We compare this with an alternative approach solely based on native LAPACK routines and observe that our new implementation may save a nonnegligible fraction of the computational time on an Intel Core2 Duo Processor.

## 1 Introduction

Let  $A$  be an  $n \times n$  upper triangular matrix and  $B$  be an  $m \times n$  matrix. Then this note is concerned with computing a QR decomposition [2]

$$\begin{bmatrix} A \\ B \end{bmatrix} = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad (1)$$

with an  $(m+n) \times (m+n)$  unitary matrix  $Q$  and an  $n \times n$  upper triangular matrix  $R$ . If  $A$  is the triangular factor of a previously computed QR decomposition then (1) amounts to updating this QR decomposition if additional rows are included. The typical scenario giving rise to the need for such updates is that the rows of a matrix are delivered by a batch process, for example when not all rows of the matrix are known in advance or when not all rows of the matrix fit into the main memory at the same time.

In the following, we discuss two different implementations for performing (1), both based on the concept of compact WY representations for products of Householder reflectors. After a short review of compact WY representations in Section 2, we will demonstrate how some recently introduced functionality in LAPACK 3.2 [1] results in a very efficient implementation solely based on native LAPACK routines. This implementation has still some disadvantages, which are addressed by an improved algorithm in Section 4. Unfortunately, the implementation of this new algorithm cannot be casted in terms of native LAPACK routines and requires a particularly tailored implementation of compact WY representations. Finally, in Section 5 we provide some preliminary performance results indicating the potential gains from our new implementations.

## 2 A short review of compact WY representations and the QR decomposition

Let  $H_j = I - \beta_j v_j v_j^T$  with  $\beta_j \in \mathbb{R}$  and  $v_j \in \mathbb{R}^m$  denote a Householder reflector. A product of  $k$  such reflectors can be represented as

$$H_1 H_2 \cdots H_k = I - V T V^T, \quad (2)$$

where  $V = [v_1, \dots, v_k]$  and  $T$  is a  $k \times k$  upper triangular matrix. The representation (2) is called *compact WY representation* and is computed by the following MATLAB function:

```
function T = compactWY(V,beta)
% Computes the compact WY representation of a product of k Householder
% reflectors.
% Input: V - m-by-k matrix containing the vectors v_j
%        beta - vector of length k containing the scalar factors beta_j
% Output: T - upper triangular factor of the compact WY representation
[m,k] = size(V); T = zeros(k);
for j = 1:k,
    T(1:j-1,j) = -beta(j) * T(1:j-1,1:j-1) * ( V(:,1:j-1)'*V(:,j) );
    T(j,j) = beta(j);
end
```

The LAPACK routine `xLARFT` contains a Fortran implementation of this function. Once the WY representation is computed it can be applied to an  $m \times n$  matrix  $B$  as follows:

$$(I - V T V^T) B = B - V (T (V^T B)).$$

This requires one call to `xGEMM` to compute  $W \leftarrow V^T B$ , one call to `xTRMM` to compute  $W \leftarrow T W$ , and another call to `xGEMM` to compute  $B - V W^T$ .

When WY representations are used to compute the QR factorization of a full  $m \times n$  matrix then the entries  $1, \dots, j-1$  of  $v_j$  are zero and the  $j$ th entry of  $v_j$  is one. Hence, the top  $k \times k$  part of  $V$  is a unit lower triangular matrix. This fact is exploited in the LAPACK routine `xLARFB` for applying compact WY representations; each call to `xGEMM` is replaced by a call to `xTRMM` to multiply the upper  $k \times k$  part of  $V$  and a call to `xGEMM` to multiply the remaining  $(m-k) \times k$  part of  $V$ .

## 3 A solution based on native LAPACK routines

With the release of version 3.2, both LAPACK routines `xLARFT` and `xLARFB` scan the trailing parts of  $V$  for zero entries. In particular, when  $V$  happens to have trailing zero rows, these rows are truncated in calls to `xGEMM` in `xLARFB` potentially resulting in a significant decrease of the computational time. To benefit from this effect, the block rows of (1) should be swapped,  $\begin{bmatrix} B \\ A \end{bmatrix}$  takes the following form ( $m = 4, n = 9$ ):

```
b b b b b b b b b
b b b b b b b b b
b b b b b b b b b
```

```

b b b b b b b b b
a a a a a a a a a
0 a a a a a a a a
0 0 a a a a a a a
0 0 0 a a a a a a
0 0 0 0 a a a a a
0 0 0 0 0 a a a a
0 0 0 0 0 0 a a a
0 0 0 0 0 0 0 a a
0 0 0 0 0 0 0 0 a
0 0 0 0 0 0 0 0 a

```

If the LAPACK routine DGEQRF, implementing a block algorithm for computing the QR decomposition, is applied with block size  $k = 3$  to  $\begin{bmatrix} B \\ A \end{bmatrix}$  then the matrix  $V$  in the compact WY representation (2) of the first 3 Householder reflectors takes the following form:

```

1 0 0
v 1 0
v v 1
v v v
v v v
0 v v
0 0 v
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0

```

That is, the sparsity pattern of the first 3 columns in the matrix  $\begin{bmatrix} B \\ A \end{bmatrix}$  is inherited. When applying the compact WY representation, the routine xLARFB will detect that the trailing 6 zero rows of  $V$ . When multiplying with  $V$ , a triangular matrix-matrix multiplication is used for the first three rows and a general matrix-matrix multiplication is used for rows 4, ..., 7. No multiplication is performed with the last 6 zero rows. The approximate computational cost for applying the compact WY representation for general  $k, m, n$  to an  $(m+n) \times p$  matrix  $C$  is as follows:

	Standard	LAPACK 3.2
$W \leftarrow C^T V$	$k^2 p + 2(m+n-k)p$	$k^2 p + 2mp$
$W \leftarrow WT^T$	$k^2 p$	$k^2 p$
$C \leftarrow C - VW^T$	$k^2 p + 2(m+n-k)p$	$k^2 p + 2mp$
total	$3k^2 p + 2(m+n-k)p$	$3k^2 p + 4mp$

These flop counts demonstrate that significant savings can be expected in the practically important case  $n \gg m$ .

As we will see in Section 5, the described strategy is quite effective at reducing the overall computational time for computing the QR decomposition of  $\begin{bmatrix} B \\ A \end{bmatrix}$ . However, there are two (minor) drawbacks:

1. The upper triangular structure in the trailing rows of the matrix  $V$  is not exploited.

2. In a batch process,  $A$  needs to be stored in a larger array (accomodating also  $B$ ) and  $A$  needs to be shifted  $m$  rows downwards after each QR update.

There seems to be no way to avoid these drawbacks when using calls to native LAPACK routines.

## 4 A tailored solution

In the following, we describe an approach that is as effective as the approach described in Section 3 but avoids its drawbacks. For this purpose, it is more convenient to consider the  $\begin{bmatrix} A \\ B \end{bmatrix}$ , which takes the form

```

a a a a a a a a a
0 a a a a a a a a
0 0 a a a a a a a
0 0 0 a a a a a a
0 0 0 0 a a a a a
0 0 0 0 0 a a a a
0 0 0 0 0 0 a a a
0 0 0 0 0 0 0 a a
0 0 0 0 0 0 0 0 a
0 0 0 0 0 0 0 0 0 a
b b b b b b b b b
b b b b b b b b b
b b b b b b b b b
b b b b b b b b b

```

The matrix  $V$  in the compact WY representation (2) of the first 3 Householder reflectors generated by the QR decomposition of  $\begin{bmatrix} A \\ B \end{bmatrix}$  then takes the form

```

1 0 0
0 1 0
0 0 1
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
v v v
v v v
v v v
v v v

```

For applying the compact WY representation to a matrix  $C$ , let us partition

$$V = \begin{bmatrix} I_k \\ 0 \\ V_3 \end{bmatrix}, \quad V_3 \in \mathbb{R}^{m \times k}, \quad C = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}, \quad C_1 \in \mathbb{R}^{k \times p}, \quad C_2 \in \mathbb{R}^{(n-k) \times p}, \quad C_3 \in \mathbb{R}^{m \times p}.$$

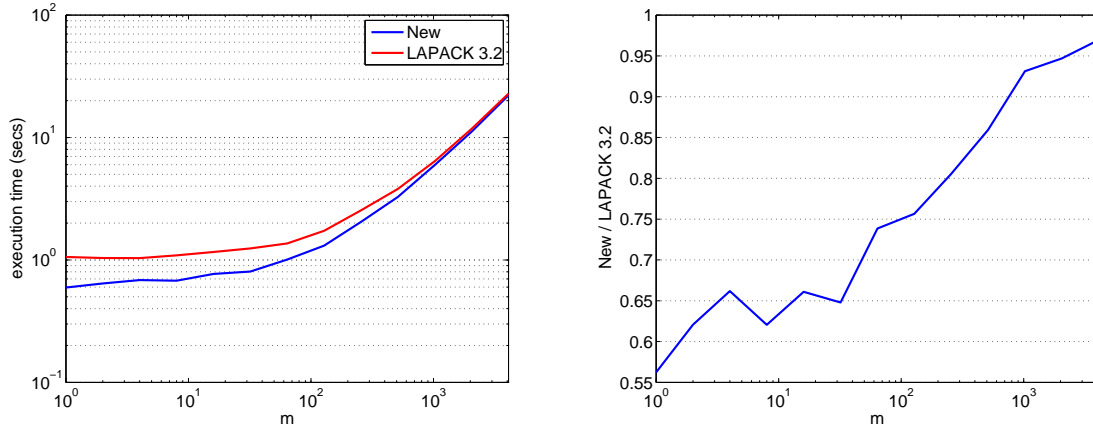


Figure 1: Execution times of our new implementation (Section 4) vs. LAPACK 3.2 (Section 3) for updating a *real* QR decomposition with  $n = 4000$  and  $m = 2^0, 2^1, \dots, 2^{12}$ . The left figure shows the absolute execution times in seconds and the right figure shows the ratio between the execution times.

Then the update  $C \leftarrow (I - VTV^T)C$  is equivalent to  $C_1 \leftarrow C_1 - W^T$  and  $C_3 \leftarrow C_3 - V_3W^T$ , where  $W = (C_1^T + C_3^T V_3)T^T$ . Hence, only two  $k^2p + 4mp$  flops are required, which compares favorably with the native LAPACK approach.

Based on this idea, we have implemented Fortran 77 routines for generating and applying such particularly structured compact WY representations. These routines are available from <http://www.math.ethz.ch/~kressner/qrupdate.php>. Currently, double real and double complex matrices are supported.

## 5 Preliminary performance results

Our Fortran implementation was compiled with the Gnu Fortran compiler (gcc version 4.1.2 20070925) with optimization level 2 and linked with the multi-threaded Goto BLAS 1.26 and LAPACK 3.2. The tests were run on an Intel Core2 Duo with 2.2 GHz. Note that for both LAPACK's `xGEQRF` as well as our new implementation a block size of  $k = 64$  was used. However, in our new implementation the use of WY representation was turned off for  $m$  smaller than  $k/2 = 32$ . Figure 1 displays the obtained execution times for matrices with double real entries while Figure 2 displays the obtained execution times for matrices with double complex entries. As expected, the difference is more notable for smaller  $m$ . In particular, for  $m \leq 100$  we save at least 30% to 40% of the execution time. For larger  $m$ , the difference in execution time becomes negligible.

## References

- [1] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

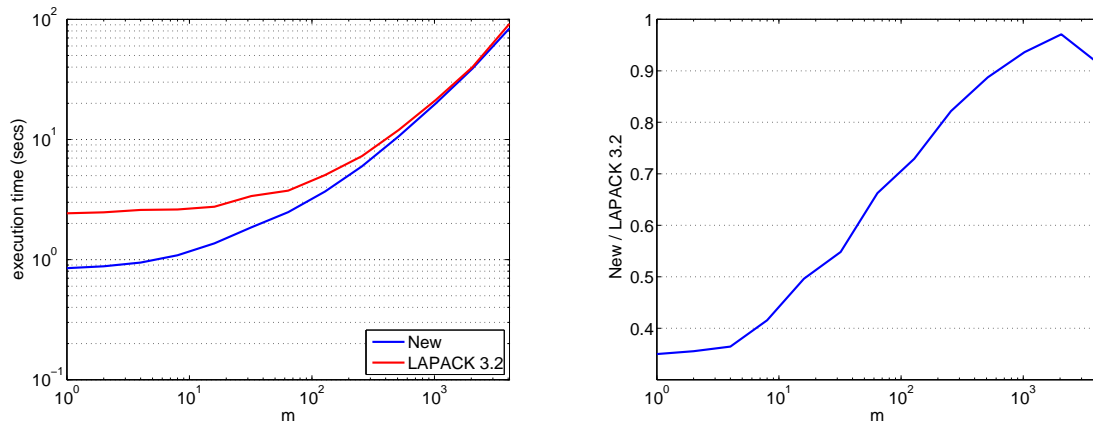


Figure 2: Execution times of our new implementation (Section 4) vs. LAPACK 3.2 (Section 3) for updating a *real* QR decomposition with  $n = 4000$  and  $m = 2^0, 2^1, \dots, 2^{12}$ . The left figure shows the absolute execution times in seconds and the right figure shows the ratio between the execution times.

[2] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.