LAPACK-Style Codes for the QR Factorization of Banded Matrices*

Alfredo Remón[1], Enrique S. Quintana-Ortí[1], and Gregorio Quintana-Ortí[1]

Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I
12.071–Castellón, Spain
{remon,quintana,gquintan}@icc.uji.es

Abstract. In this paper we present unblocked and blocked LAPACK-style codes for the computation of the QR factorization of a banded matrix. The new routines are evaluated using highly-tuned multithreaded implementations of BLAS on two shared-memory multiprocessors, revealing the performance of the blocked codes.

Key words: Banded matrices, QR factorization, Householder transformations, LAPACK, multithreaded BLAS, Symmetric Multiprocessor (SMP).

1 Introduction

The QR factorization is an important numerical tool for the solution of linear systems and linear least squares problems [3]. When the coefficient matrix in these problems is banded, huge savings can be gained in operations and storage by exploiting the structure of the problem. Problems with banded coefficient matrix appear, e.g., in static and dynamic analyses and linear equations in structural engineering, finite element analysis in structural mechanics, and domain decomposition methods for partial differential equations in civil engineering.

In this paper we describe the design, implementation, and evaluation of new unblocked and blocked Fortran-77 routines for the QR factorization of a banded matrix. (No routine exist in LAPACK [1] which covers this functionality.) The routines address the LAPACK storage scheme for banded matrices, and reduce the number of operations by exploiting the nonzero pattern of the problem. Furthermore, Householder reflectors are used to attain high performance in the factorization.

The next two sections contain the description of the new routines, and a few experimental results which confirm the performance of the new codes.

* This research was supported by the CICYT project TIN2005-09037-C02-02 and FEDER, and project No. P1B-2008-19 of the *Fundación Caixa-Castelló/Bancaixa and UJI*.

2 Computing the QR Factorization of a Banded Matrix

Given a matrix $A \in \mathbb{R}^{m \times n}$, with $m \geq n$, its QR factorization is given by $A = QR$, where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular. When A presents lower and upper bandwidths k_l and k_u, respectively, a careful computation of the factorization leads to an upper triangular factor with upper bandwidth $k_l + k_u$, as described next.

2.1 Householder transformations and the QR Factorization

Consider the real-valued vector $x = \left(\frac{\chi_0}{x_1}\right)$, where χ_0 equals the first element of x. The Householder vector associated with x is defined as $u = \left(\frac{1}{x_1/\nu_0}\right)$, where $\nu_0 = \chi_0 + \text{sign}(\chi_0)\|x\|_2$. If $\beta = \frac{2}{u^T u}$, then $(I - \beta u u^T)x$ annihilates all but the first element of x, which becomes $\eta = -\text{sign}(\chi_0)\|x\|_2$. The transformation $I - \beta u u^T$ is referred to as a Householder transformation or *reflector*.

Let us introduce the notation $[u, \eta, \beta] := h(x)$ as the computation of the above mentioned u, η, and β from vector x. Assume that the first k columns of the matrix A have already been factorized and the corresponding Householder transformations have been applied, and consider the partitionings

$$A = \left(\begin{array}{c|c|c} A_{TL} & A_{TM} & \\ \hline A_{ML} & A_{MM} & A_{MR} \\ \hline & A_{BM} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c|c|c} A_{00} & a_{01} & A_{02} & & \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T & & \\ \hline A_{20} & a_{21} & A_{22} & & \\ \hline & & A_{32} & \alpha_{33} & \\ \hline & & A_{42} & a_{43} & A_{44} \end{array}\right), \tag{1}$$

where $A_{TL} \in \mathbb{R}^{k \times k}$, $A_{MM} \in \mathbb{R}^{k_l+1 \times k_u+k_l+1}$, and α_{11}, α_{33} are both scalars. Then, in order to factorize one more column of the matrix, during the current iteration the following computations have to be performed:

$$\begin{aligned} &\left[\left(\frac{1}{a_{21}}\right), \alpha_{11}, \beta_1\right] = \left[\left(\frac{1}{u_2}\right), \eta, \beta_1\right] := h\left(\frac{\alpha_{11}}{a_{21}}\right) \\ &w^T := a_{12}^T + u_2^T A_{22} \\ &\left(\frac{a_{12}^T}{A_{22}}\right) := \left(\frac{a_{12}^T - \beta_1 w^T}{A_{22} - \beta_1 u_2 w^T}\right) \end{aligned}$$

As a result of these operations, a_{12}^T / A_{22} become a fully populated vector/block, leading to the upper triangular factor R of bandwidth $k_l + k_u$.

Figure 1 illustrates the packed storage scheme used for band matrices in LAPACK and how this scheme accommodates for the result of the QR factorization.

2.2 A blocked routine for the QR Factorization

For linear algebra codes, portable high performance can be attained on current desktop computers by casting most of the computations in terms of the matrix-matrix product. The computation of the QR factorization is no exception and

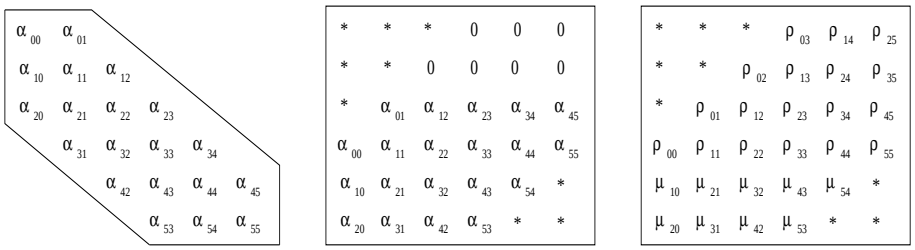

Fig. 1. 6×6 band matrix with upper and lower bandwidths $k_l = 2$ and $k_u = 1$, respectively (left); packed storage scheme used in LAPACK (center); result of the QR factorization where $\rho_{i,j}$ and $\mu_{i,j}$ stand, respectively, for the entries of the upper triangular factor R and the Householder vectors of the corresponding columns, u.

the use of aggregated Householder reflectors [2] usually yields good results for dense matrices. Here, we have extended this technique to banded matrices, taking care to avoid unnecessary fill-in in the matrix. In order to do so, b columns are factorized per iteration and the corresponding block transformation is applied in terms of matrix-matrix operations.

3 Experimental Results

All experiments in this section were performed using IEEE double-precision (real) arithmetic, and band matrices of order $n = 5,000$ with $k_l = k_u$. For each bandwidth dimension, we employed values from 1 to 200 to determine the optimal block size, but only those results corresponding to the best block size are shown.

We report the performance of the unblocked and blocked codes, `DGBQR2` and `DGBQRF` respectively, on two SMP architectures with 2 and 4 processors. Details on the hardware and software employed in the evaluation are given in Table 1.

Figure 2 illustrates the performance, in terms of GFLOPs (10^9 floating-point arithmetic operations per second), of the unblocked and blocked codes using a single thread and multiple threads. In the latter case, 2 and 4 threads are employed on XEON and ITANIUM, respectively. The results show that the blocked routines outperform their unblocked versions starting from a given bandwidth dimension which depends on the cache dimension of the architecture. The difference between the performance of the blocked routines when using GotoBLAS or MKL are minor. Finally, there is a limited amount of parallelism in the algorithm which is reflected in the asymptotic GFLOPs rate that is attained.

References

1. E. Anderson, Z. Bai, J. Demmel, J. E. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. E. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992.

Platform	Architecture	#Proc.	Frequency (GHz)	L2 cache (KBytes)	L3 cache (MBytes)	RAM (GBytes)
XEON	Intel Xeon	2	2.4	512	–	1
ITANIUM	Intel Itanium2	4	1.5	256	4	4

Platform	BLAS	Compiler	Optimization Flags	Operating System
XEON	GotoBLAS 1.15 MKL 8.1	`gcc` 3.3.5	`-O3`	Linux 2.4.27
ITANIUM	GotoBLAS 1.15 MKL 8.0	`icc` 9.0	`-O3`	Linux 2.4.21

Table 1. SMP architectures (top) and software (bottom) employed in the evaluation.

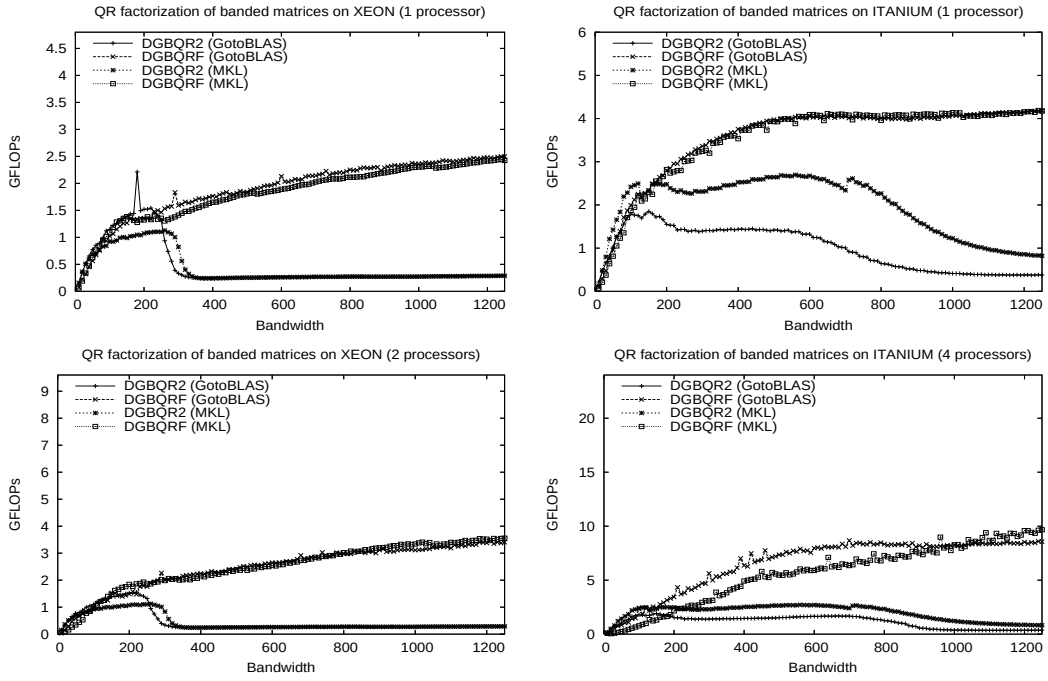

Fig. 2. Performance of the banded QR factorization codes using 1 thread (top) and multiple threads (bottom).

2. Christian Bischof and Charles Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):s2–s13, Jan. 1987.
3. Ake Björk. *Numerical Methods for Least Squares Problems*. SIAM, 1996.