LAPACK-Style Codes for the QR Factorization of Banded Matrices

Alfredo Remón-Gómez, Enrique S. Quintana-Ortí,
Gregorio Quintana-Ortí
{remon,quintana,gquintan}@icc.uji.es

Universidad Jaume I de Castellón (Spain)

9th International Workshop on State-of-the-Art in Scientific and Parallel Computing

Outline

QR factorization

QR factorization for general matrices

Given $A \in \mathbb{R}^{m \times n}$, with $m \geq n$, its QR factorization is given by

$$A = Q \cdot R,$$

Where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is singular

QR factorization for general band matrices

QR factorization for general band matrices

Given $A \in \mathbb{R}^{m \times n}$, with upper and lower bandwidth ku and kl respectively, then if QR factorization with Householder transformations is computed carefully,

- Q is lower triangular with lower bandwidth kl
- R is upper triangular with upper bandwidth $ku + kl$

Householder transformations

The Householder transformations can be employed to annihilate elements from a vector or matrix

Householder transformation

- Consider vector $x = \begin{pmatrix} \chi_0 \\ x_1 \end{pmatrix}$
- Then, $\left(I - \beta u u^T\right) x = \begin{pmatrix} \bar{\chi_0} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} -sign(\chi_0)\|x\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

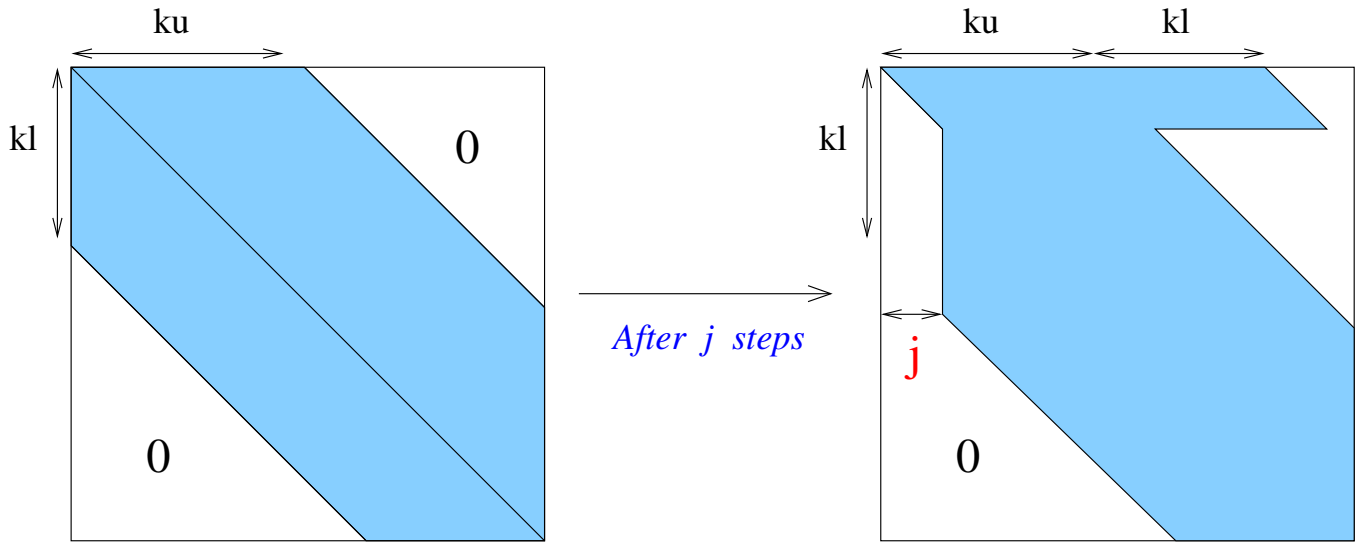
With $\beta = \frac{2}{u^T u}$, $u = \begin{pmatrix} 1 \\ x_1/v_0 \end{pmatrix}$ and $v_0 = \chi_0 + sign(x_0)\|x\|_2$

Where $\left(I - \beta u u^T\right) x$ is the Householder transformation

Householder transformations for the QR factorization

Objective

Transform the general band matrix A into an upper triangular band matrix

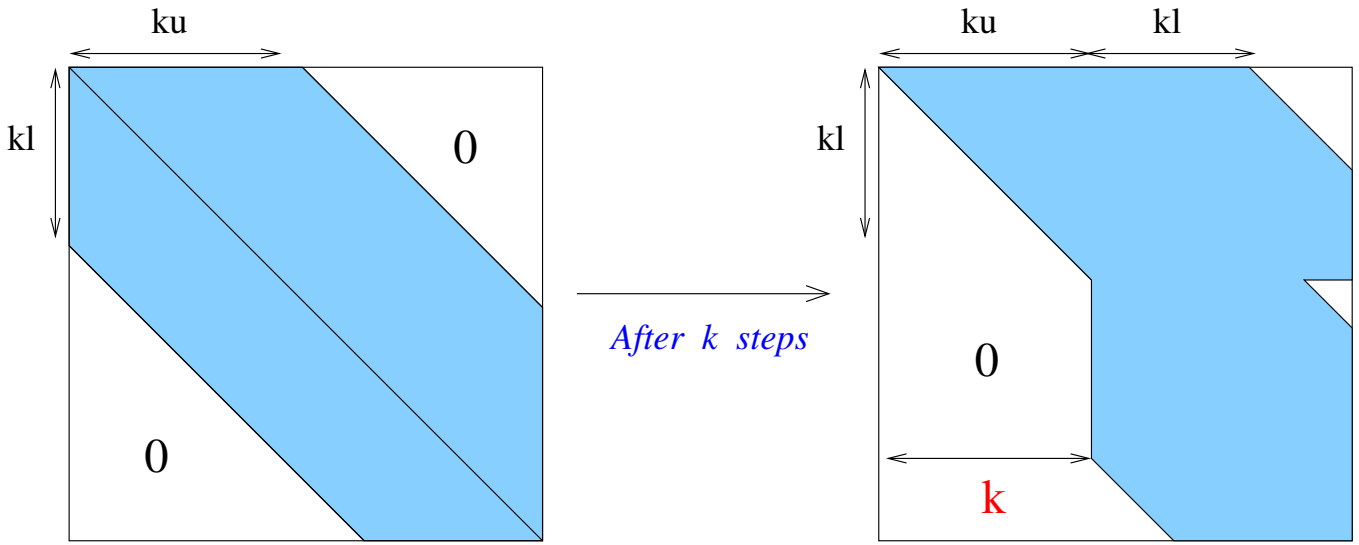


$$\left(I - \beta u_1 u_1^T\right)\left(I - \beta u_2 u_2^T\right) \ldots \left(I - \beta u_j u_j^T\right) \cdot A$$

Householder transformations for the QR factorization

Objective

Transform the general band matrix A into an upper triangular band matrix

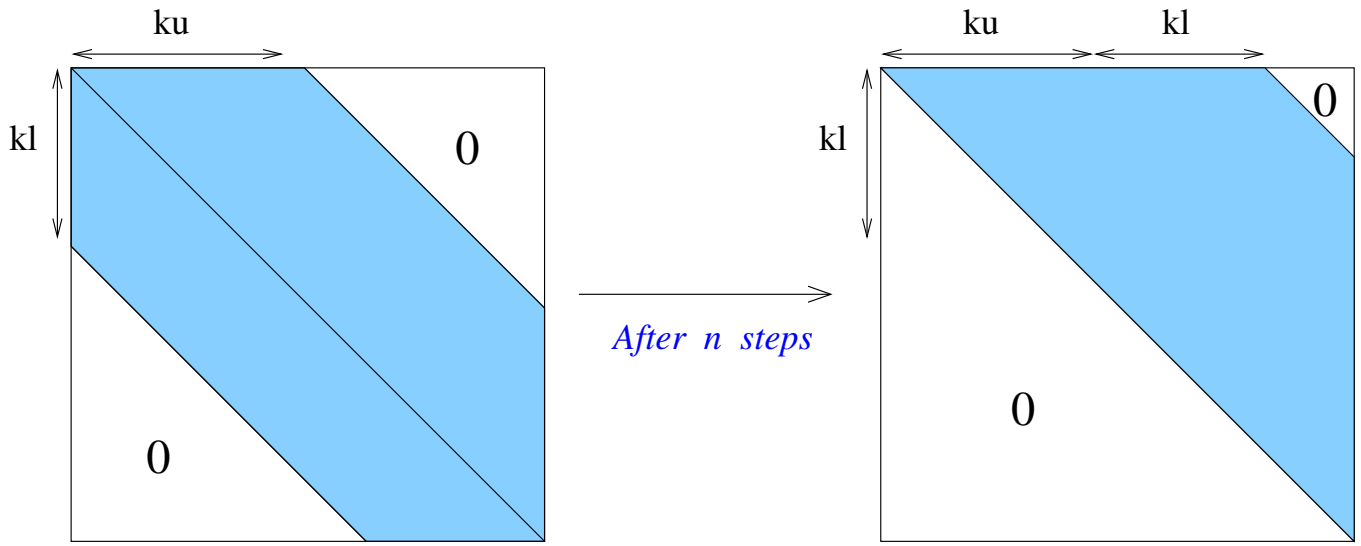


$$\left(I - \beta u_1 u_1^T\right) \cdots \left(I - \beta u_j u_j^T\right) \cdots \left(I - \beta u_k u_k^T\right) \cdot A$$

Householder transformations for the QR factorization

Objective

Transform the general band matrix A into an upper triangular band matrix



$$\left(I - \beta u_1 u_1^T\right)\left(I - \beta u_2 u_2^T\right) \ldots \left(I - \beta u_n u_n^T\right) \cdot A$$

New routines for the QR factorization of general band matrices

New routines implemented

- GBBQR2
  - Unblocked routine
  - Based on BLAS-2 kernels
- GBBQRF
  - Blocked routine
  - Based on BLAS-3 kernels

We will focus on the blocked routine GBBQRF

Routine GBBQRF

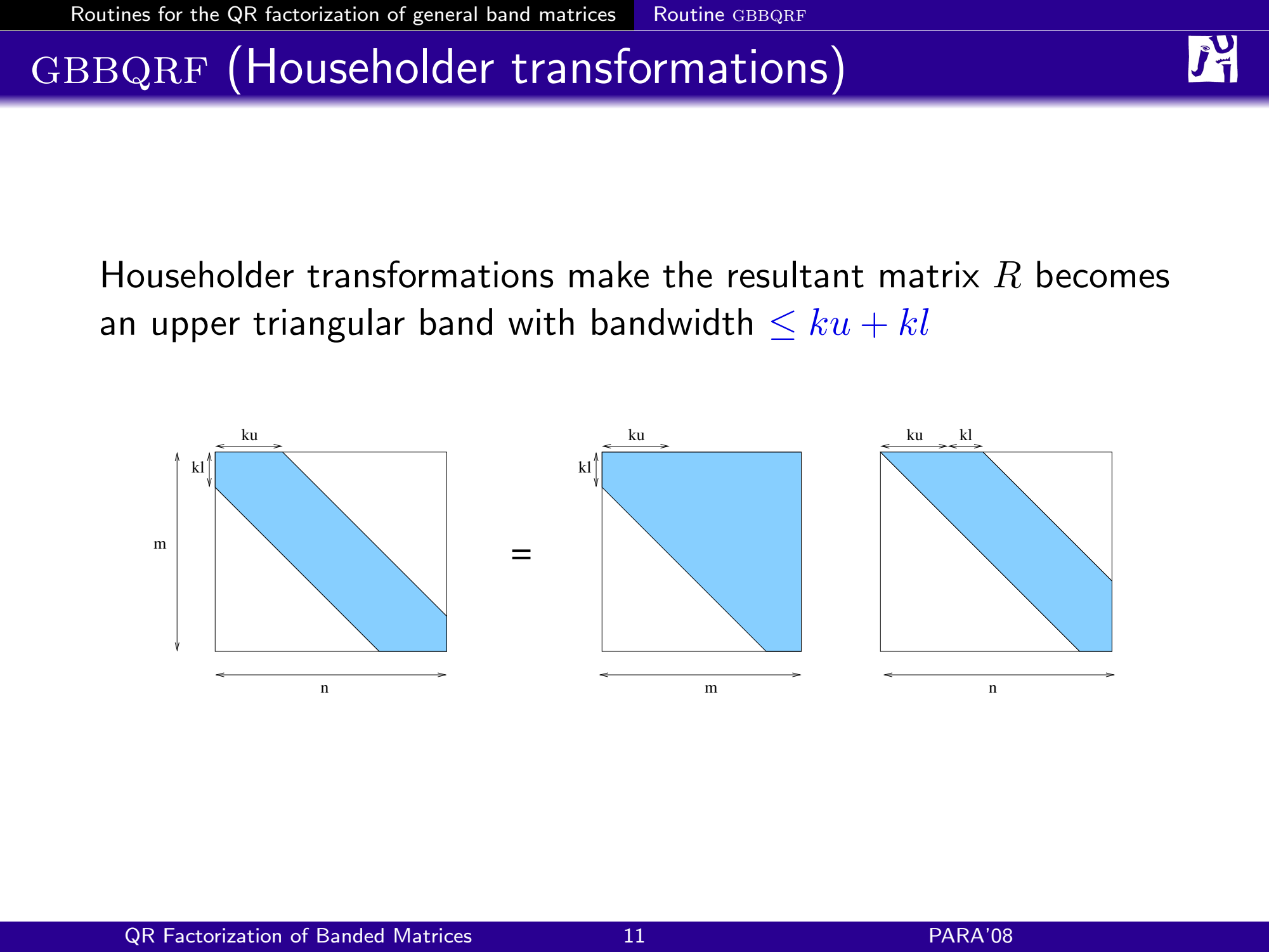
Functionality

Computes the QR factorization of a general band matrix

Main characteristics

- Exploits the matrix structure so the memory requirements and the computational cost are reduced
- Use of Householder transformations
- Based on BLAS-3 operations

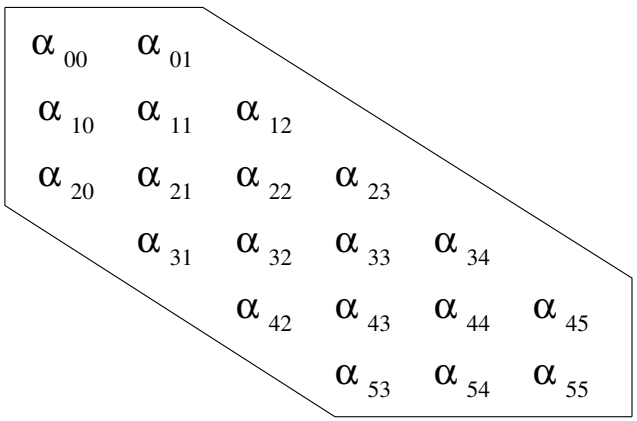
GBBQRF (Householder transformations)

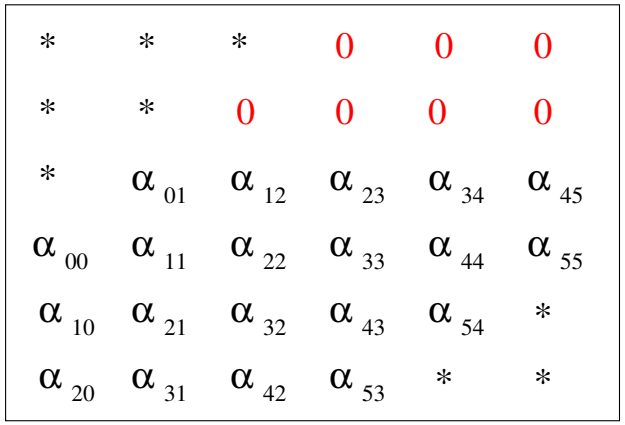


Householder transformations make the resultant matrix R becomes an upper triangular band with bandwidth $\leq ku + kl$

GBBQRF (Storage scheme)

- Matrix A is stored following a variant of the compact storage scheme employed by BLAS and LAPACK routines
- In this variant, kl extra upper diagonals are stored, initially padded with zeros





$m = n = 6$, $kl = 2$ and $ku = 1$

- This scheme is analogous to the one employed in LAPACK routines for the LU factorization

GBBQRF (Storage scheme)

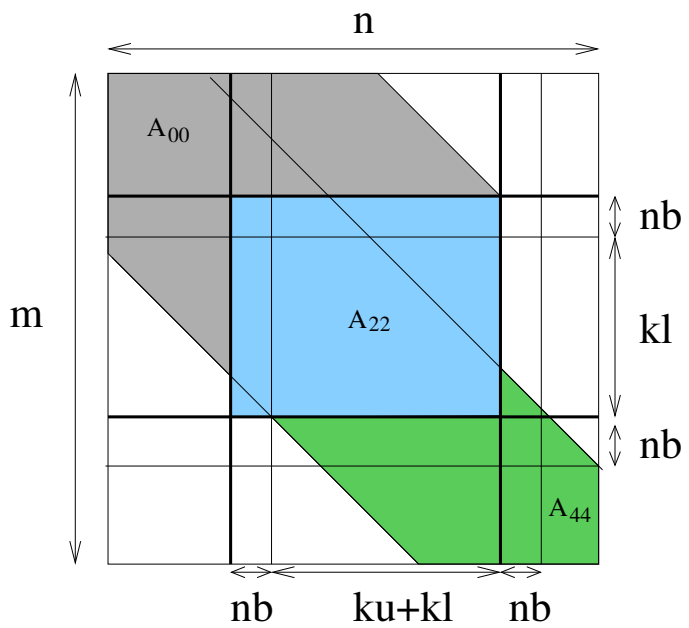
- Following the LAPACK-Style, matrices the reflectors and R replace matrix A
- The extra kl upper-diagonals padded with zeros are necessary to store all the elements of R

*	*	*	0	0	0
*	*	0	0	0	0
*	α_{01}	α_{12}	α_{23}	α_{34}	α_{45}
α_{00}	α_{11}	α_{22}	α_{33}	α_{44}	α_{55}
α_{10}	α_{21}	α_{32}	α_{43}	α_{54}	*
α_{20}	α_{31}	α_{42}	α_{53}	*	*

*	*	*	ρ_{03}	ρ_{14}	ρ_{25}	
*	*	ρ_{02}	ρ_{13}	ρ_{24}	ρ_{35}	R
*	ρ_{01}	ρ_{12}	ρ_{23}	ρ_{34}	ρ_{45}	
ρ_{00}	ρ_{11}	ρ_{22}	ρ_{33}	ρ_{44}	ρ_{55}	
μ_{10}	μ_{21}	μ_{32}	μ_{43}	μ_{54}	*	Q
μ_{20}	μ_{31}	μ_{42}	μ_{53}	*	*	

GBBQRF (Algorithm)

GBBQRF implements an iterative algorithm with a 5×5 partitioning

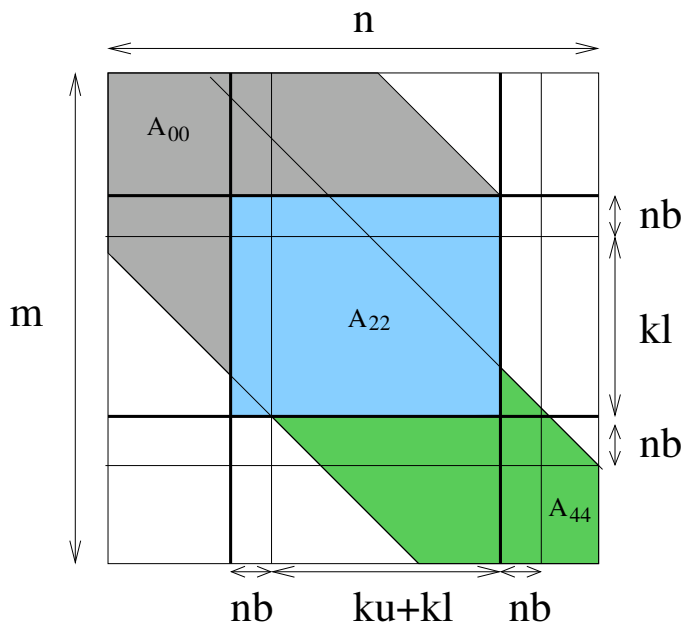


The operations performed in each iteration are

- The QR factorization of nb columns of A is computed
- The next $ku + kl$ columns of A are updated

GBBQRF (Algorithm)

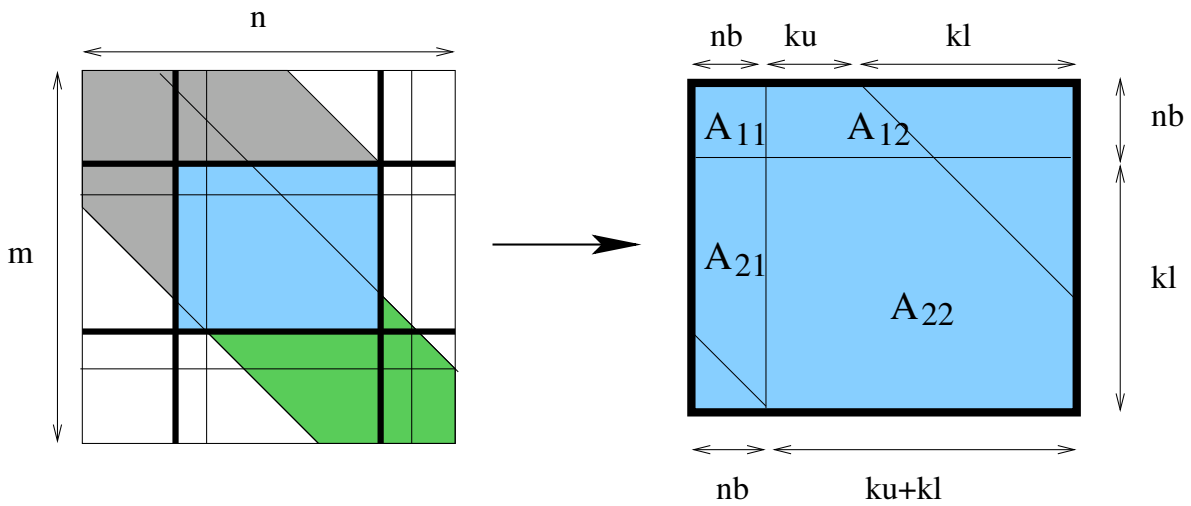
GBBQRF implements an iterative algorithm with a 5×5 partitioning



- Blocks in gray have been computed
- Blocks in blue form the active region (updated during the iteration)
- Blocks in green have not been accessed

GBBQRF (Algorithm)

During the current iteration, only blocks in the active region will be accessed

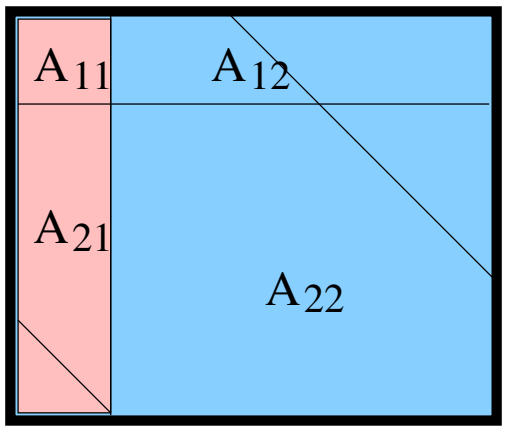


Note that the lower triangular part of the $(nb \times nb)$ block positioned at the bottom of A_{21} is out of the band

GBBQRF (Algorithm)

Step 1: QR factorization of blocks A_{11} and A_{21}

- Using the new non-blocked BLAS-2 routine GBBQR2, the QR factorization of blocks A_{11} and A_{21} is obtained

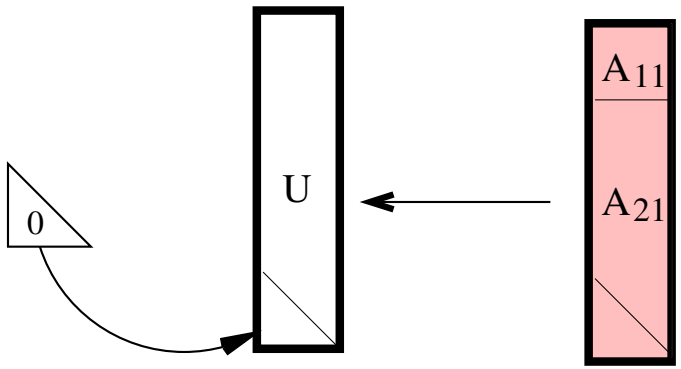


$$(\text{GBBQR2}) \quad \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} = \text{QR}\left(\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}, \; r \right),$$

GBBQRF (Algorithm)

Step 2.1: Apply the Householder transformations to blocks A_{12} and A_{22}

- Copy A_{11} and A_{21} into a rectangular workspace where the whole block A_{21} is stored

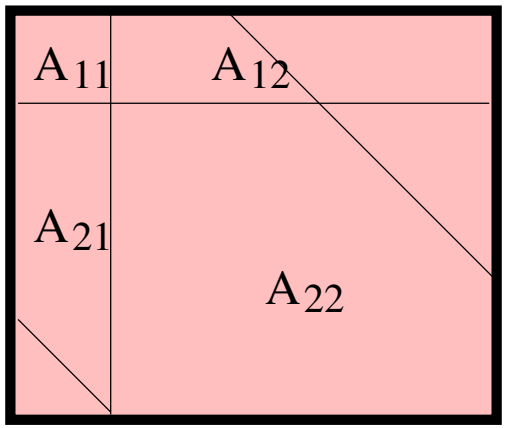


$$
\begin{aligned}
U &= \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}, \\
\text{STRIL}(U) &= 0,
\end{aligned}
$$

GBBQRF (Algorithm)

Step 2.2: Apply the Householder transformations to blocks A_{12} and A_{22}

- Compute of the Householder transformation and update of A_{12} and A_{22}



$$
\begin{aligned}
&(\text{LARFT}) \quad T = \text{LARFT}(U,\ r),\\
&(\text{LARFB}) \quad \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix} = (I - UTU^T) \cdot \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix},
\end{aligned}
$$

GBBQRF (Algorithm)

- The lower triangular part of the $nb \times nb$ block at the bottom of A_{21} contains only null elements
- No attemption to exploit this structure is done because:
  - As $b \ll kl$ no huge savings are expected Dealing with this structure would increment the number of invocations to BLAS routines

Step 3: Move boundaries

- The active region moves along the diagonal nb rows and columns

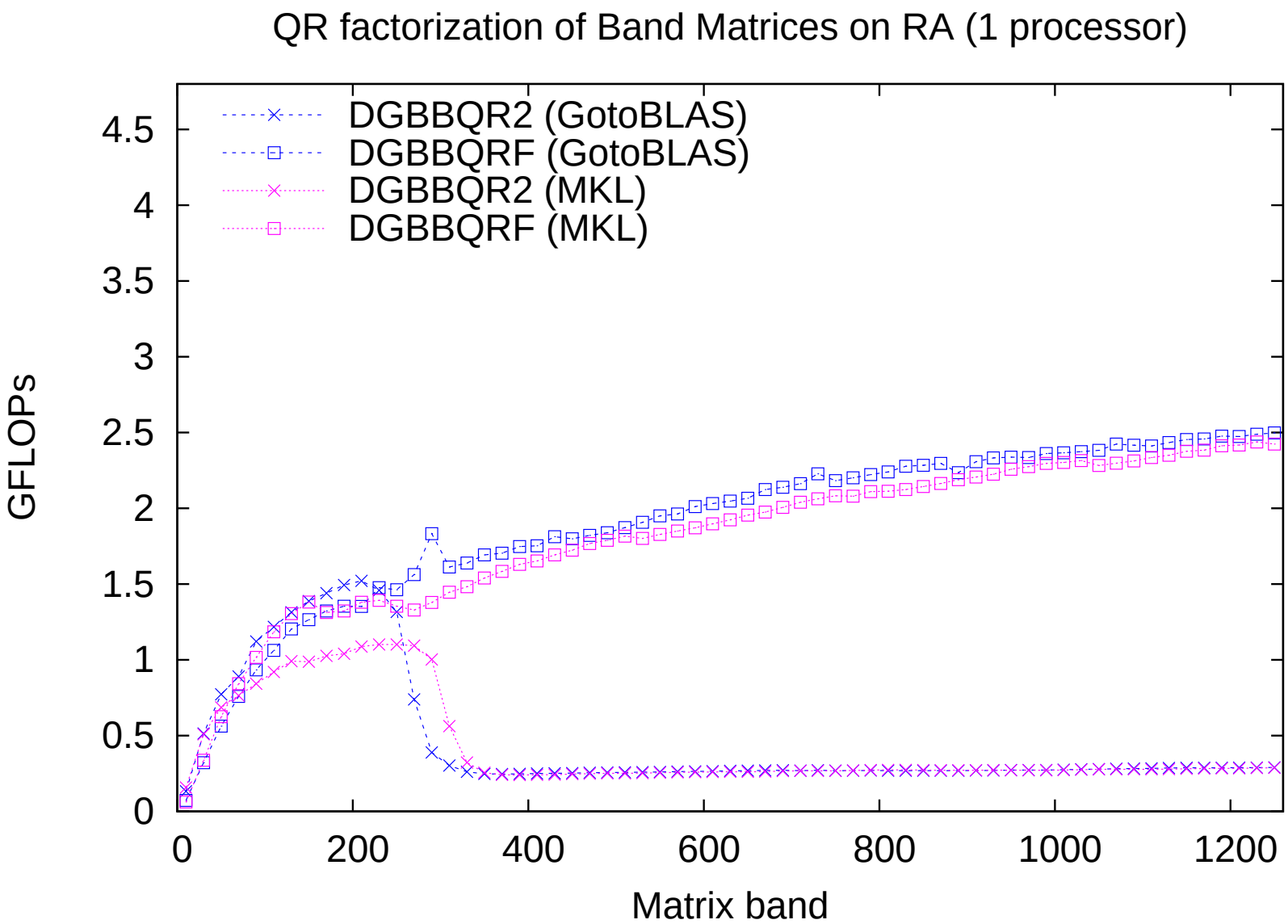
Experimental Results

Platform	Architecture	Frequency (GHz)	L2 cache (KBytes)	L3 cache (MBytes)	RAM (GBytes)
RA	Intel Xeon	2.4	512	–	1
CATON	Intel Itanium2	1.5	256	4	4

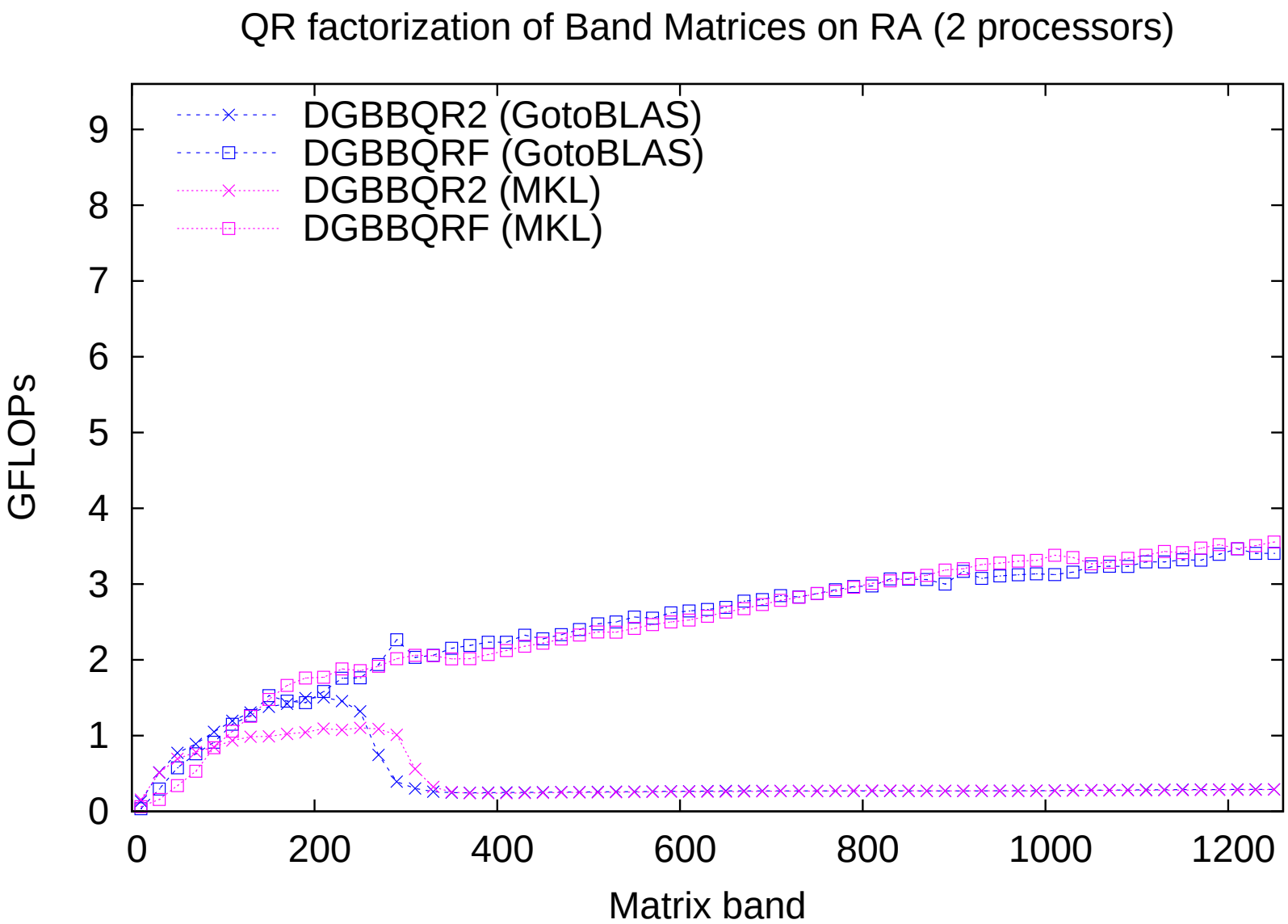
Platform	Compiler	Optimization flags	BLAS	Operating System
RA	gcc 3.3.5	-O3	Goto BLAS 1.15 MKL 8.1	Linux 2.4.27
CATON	`icc` 9.0	-O3	Goto BLAS 1.15 MKL 8.0	Linux 2.4.21

Results correspond to matrices of order n=5,000 with different bandwidths (from 10 to 1250) and block sizes (from 1 to 100)

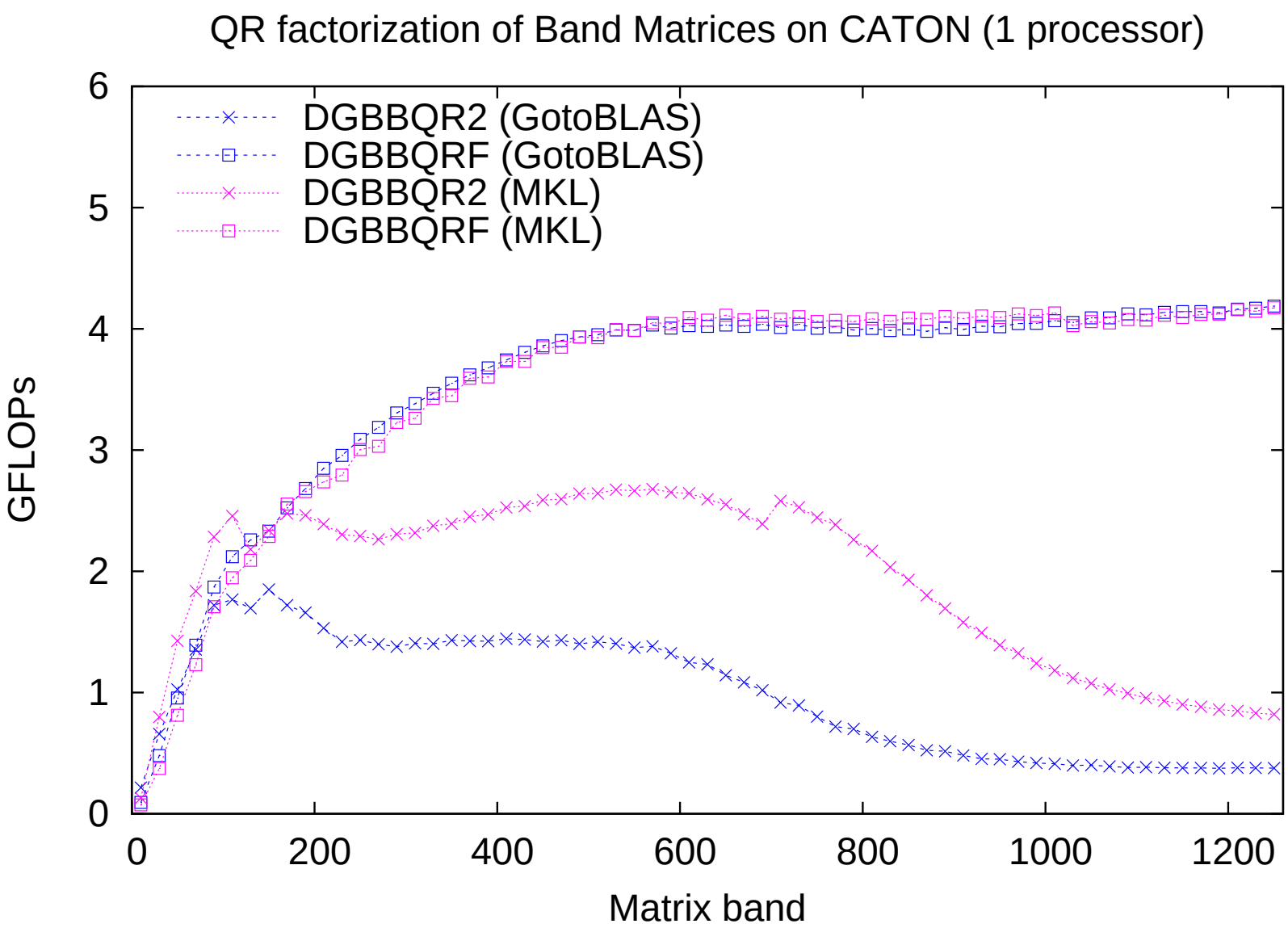
Experimental results RA (1 proc.)



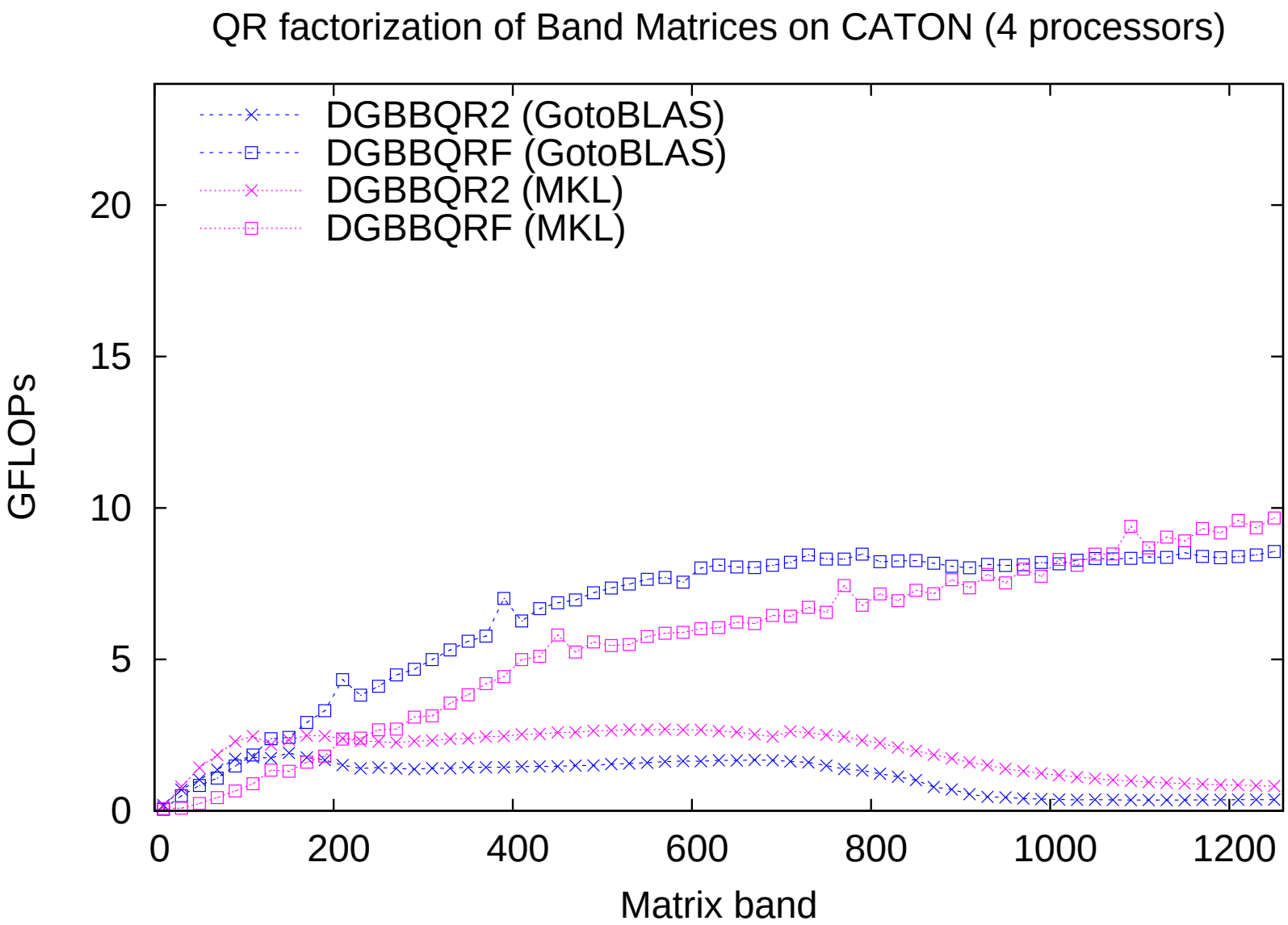
Experimental results RA (2 proc.)



Experimental results CATON (1 proc.)



Experimental results CATON (4 proc.)



Conclusions

- Two new LAPACK-Style routines for the computation of the QR factorization of band matrices have been presented
- Both routines exploit the matrix structure to reduce the memory requirements and the number of computations
- The new codes have been evaluated in two current platforms, showing the higher performance of the blocked routine
- Results show that the operation presents a limited degree of parallelism

Thanks.