CholeskyQR2: Cholesky QR factorization with reorthogonalization

November 3, 2015
@ICL, The University of Tennessee

<u>Takeshi Fukaya (Hokkaido Univ. / JST CREST)</u>

joint work with

Yuji Nakatsukasa (The Univ. of Tokyo)
Yuka Yanagisawa (Waseda Unv.)
Yusaku Yamamoto (UEC / JST CREST)

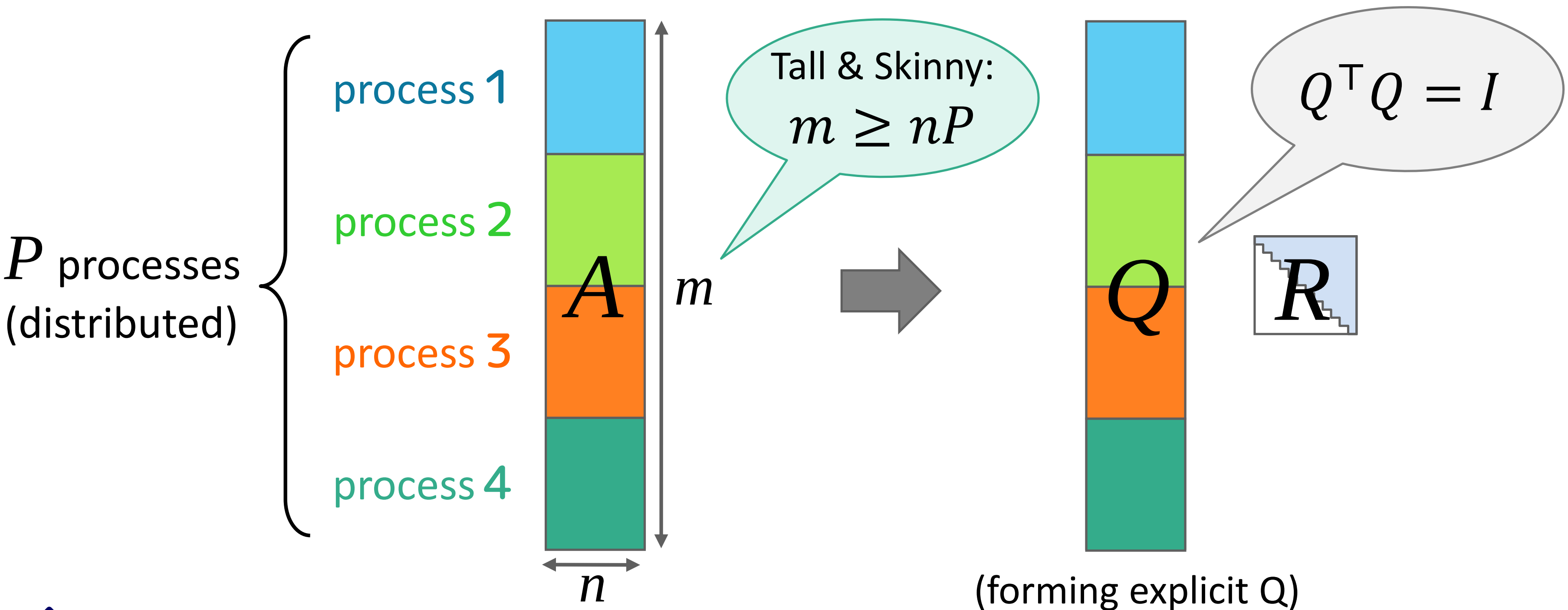
Introduction of our study

Overview

- **Cholesky QR: an algorithm for computing a QR factorization via Cholesky factorization**
  - ✓ very suitable for high-performance computing
  - ✓ simple and easy to implement
  - ✓ numerically unstable
- **Studies aiming for improving its stability**
  - ✓ mixed-precision approach (by I. Yamazaki et al.)
  - ✓ our focus: with reorthogonalization (CholeskyQR2)
    - ➢ theoretical round-off error analysis
    - ➢ performance evaluation on the K computer

Our target problem

- **Tall-skinny QR factorization on a large-scale parallel computer**



- **Applications**
  - computing of an orthogonal basis in block subspace projection methods
  - preprocessing for the SVD of tall and skinny matrices
  - generating an orthogonal transformation for dense-to-band reduction

Efficient parallel algorithm for computing tall-skinny QR fact. is required!

Comm. In conventional algorithms

◆ Gram-Schmidt (CGS)

$$q_1 := \frac{1}{\|a_1\|} a_1$$ (comm.)

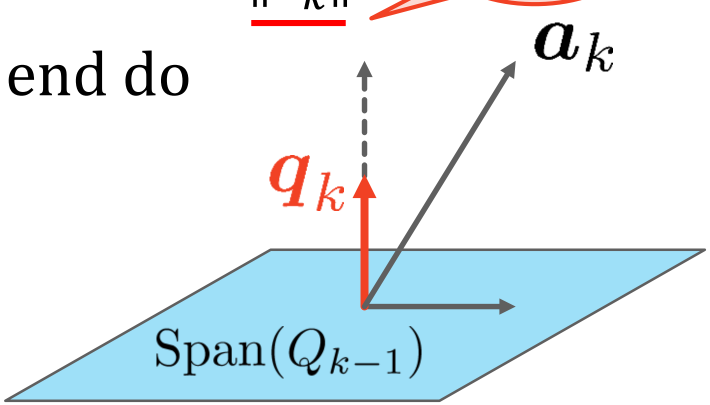
do k=2, n

$$a_k := (I - Q_{k-1}Q_{k-1}^\top) a_k$$ (comm.)

※ $Q_{k-1} := [q_1 \cdots q_{k-1}]$

$$q_k := \frac{1}{\|a_k\|} a_k$$ (comm.)

end do



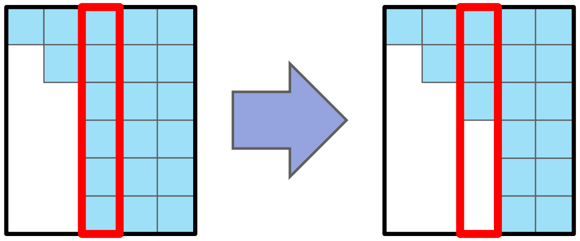
◆ Householder QR

do k=1, n

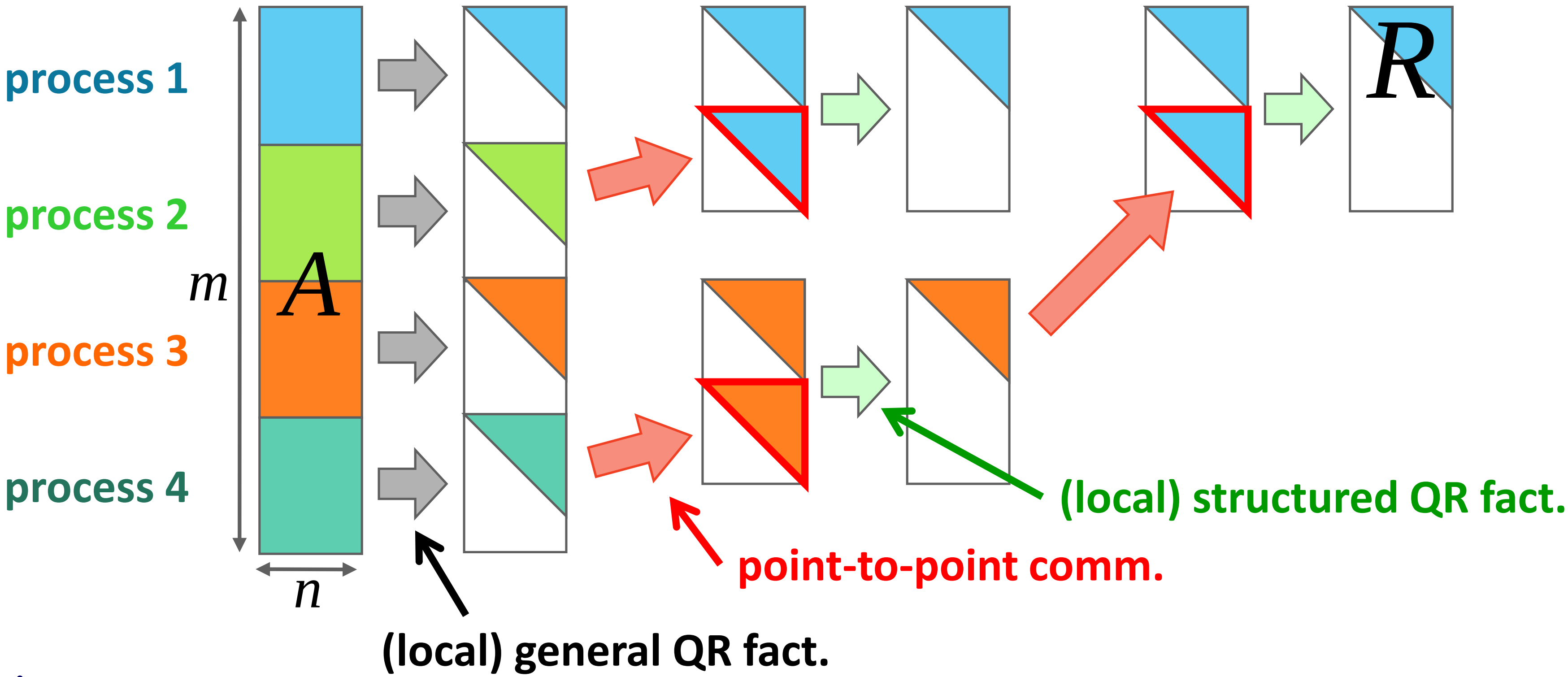
$$[t_k, y_k] := \mathrm{house}(a_k)$$ (comm.)

$$A := (I - t_k y_k y_k^\top) A$$ (comm.)

end do



TSQR algorithm (J. Demmel, et al., 2012)



Features

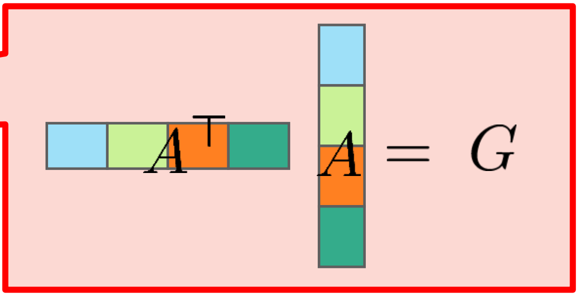
- Comm.-Avoiding: only $\log_2 P$ p-to-p comms. (= 1 global collective comm.)
  (Householder QR requires O(n) collective comms.)
- numerically as stable as Householder QR (and much more than GS algorithms)
- similar computations (but in reverse order) are needed to form the explicit Q.

Cholesky QR algorithm

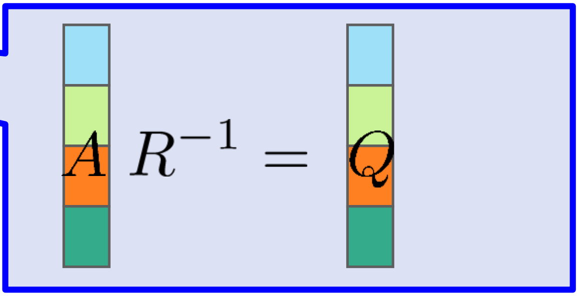
$$\mathrm{CholQR}(A) =: [Q, R]$$

1. $G := A^\top A$
2. $R^\top R := G$ (Cholesky fact. of G)
3. $Q := AR^{-1}$

(local) matrix-matrix mult.
=> collective comm. (Allreduce)



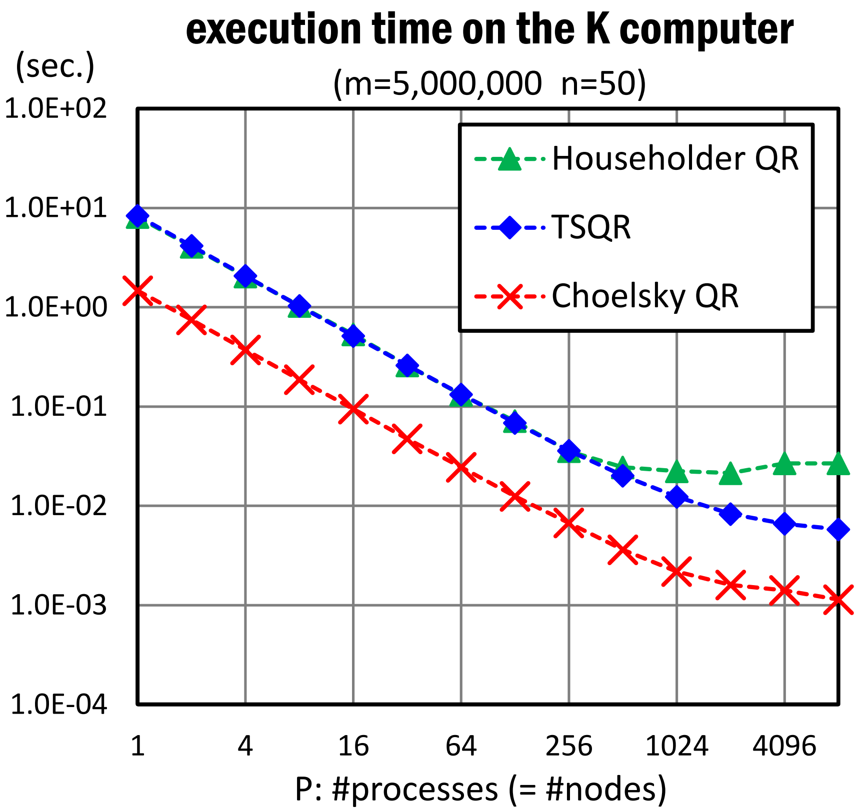
(local) back substitution

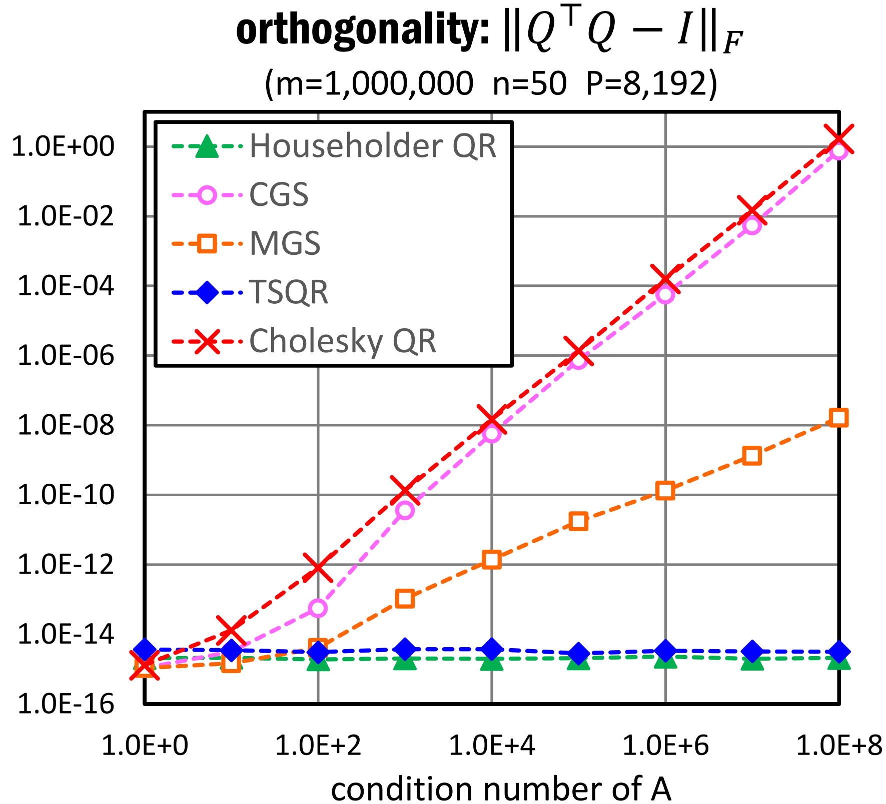


Features

- Comm.-Avoiding: only 1 global collective comm. (Allreduce).
- #flops. is about half that of TSQR (as in Householder QR vs. GS)
- consist of a few fundamental (and usually highly tuned) operations
- known to be **numerically unstable**

TSQR vs. Cholesky QR





- Cholesky QR is much faster than TSQR (and Householder QR).
- Cholesky QR becomes unstable as cond(A) grows (as in CGS).

(residual $\|QR - A\|_F$ is small in every algorithm.)

Cholesky QR is fast but not practical due to its numerical instability.

The CholeskyQR2 algorithm

Motivation

◆ Study on SVQB [A. Stathopoulos & K. Wu, 2002]

- reporting that repeating Cholesky QR improves the numerical stability

◆ Studies on TSQR [J. Demmel, et al., 2012, etc.]

- no comparison with repeated Cholesky QR algorithm

◆ Relationship to Gram-Schmidt type algorithms

- improvement of numerical stability by reorthogonalization (as in CGS2)
- “twice is enough”(W. Kahan & B. Parlett) is applicable?

triangular orthogonalization

$$A \underbrace{\hat{R}_1 \cdots \hat{R}_k}_{\text{upper triangular}} = Q, \quad \left(\hat{R}_1 \cdots \hat{R}_k\right)^{-1} =: R.$$

CGS, MGS, Cholesky QR, ...

orthogonal triangularization

$$\underbrace{\hat{Q}_k \cdots \hat{Q}_1}_{\text{orthogonal}} A = \begin{bmatrix} R \\ O \end{bmatrix}, \quad \left(\hat{Q}_k \cdots \hat{Q}_1\right)^{-1} \begin{bmatrix} I_n \\ O \end{bmatrix} =: Q.$$

Householder QR, TSQR, ...

We focus on an algorithm of repeating Cholesky QR twice (CholeskyQR2).

(CholeskyQR2 can be interpreted as a variant of Cholesky QR with reorthogonalization.)

CholeskyQR2 algorithm

$\mathrm{CholQR2}(A) =: [Q, R]$

1. $[Q_1, R_1] := \mathrm{CholQR}(A)$
2. $[Q, R_2] := \mathrm{CholQR}(Q_1)$
3. $R := R_2 R_1$

reorthogonalization

1. $W := A^\top A$
2. $R_1^\top R_1 := W$
3. $Q_1 := A R_1^{-1}$

1. $W := Q_1^\top Q_1$
2. $R_2^\top R_2 := W$
3. $Q := Q_1 R_2^{-1}$

Features

- Comm.-Avoiding: only 2 global collective comms.
- #msgs. and #words are twice that of Cholesky QR (and TSQR).
- #flops. is as much as TSQR (twice that of Cholesky QR)
- consist of a few fundamental (and usually highly tuned) operations
- can be interpreted as a variant of Cholesky QR with reorthogonalization.

Comparison of parallel execution costs

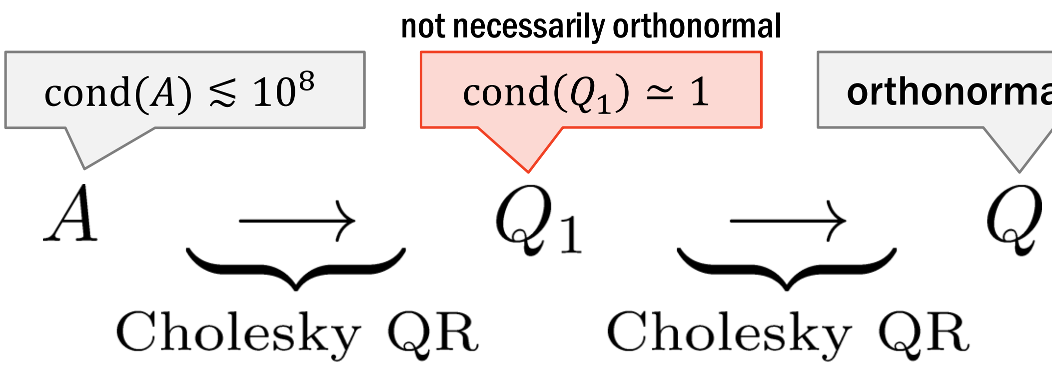
#msgs and #words are converted into point-to-point comm.

	TSQR		CholeskyQR2	
#flops	$2mn^2/P$	general QR fact.	$2mn^2/P$	computing $A^\top A$
	$2mn^2/P$	(general) forming Q	$2mn^2/P$	back substitution
			$2n^3/3$	Cholesky fact.
			$2n^3/3$	forming R
	$(2n^3\log_2 P)/3$	structured QR fact.	$n^2\log_2 P$	in reduction (addition)
	$(2n^3\log_2 P)/3$	(structured) forming Q		
#msgs	$\log_2 P$	for QR fact.	$2\log_2 P$	reduce
	$\log_2 P$	for forming Q	$2\log_2 P$	broadcast
#words	$(n^2\log_2 P)/2$	for QR fact.	$n^2\log_2 P$	reduce
	$(n^2\log_2 P)/2$	for forming Q	$n^2\log_2 P$	broadcast

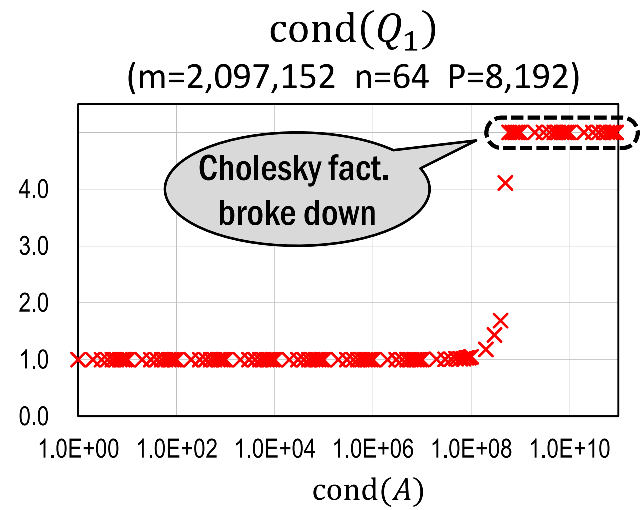
CholeskyQR2 requires cheaper reduction operation but x2 comm. cost.

Theoretical results on stability

◆ Mechanism of CholeskyQR2



(assuming that Cholesky fact. numerically does not break down)



Theorem (Y. Yamamoto, et al., 2015)

Let $\hat{Q}, \hat{R}$ be the computed QR fact. of A by CholeskyQR2 in floating-point arithmetic.

If

$$\mathrm{cond}(A) \le \frac{1}{8\sqrt{(m+n+1)n}} \cdot \frac{1}{\sqrt{\epsilon}} \; (\simeq 10^8),$$

then

$$\|\hat{Q}^\top \hat{Q} - I\|_F \le 6(m+n+1)n\epsilon, \quad \|\hat{Q}\hat{R} - A\|_F \le 5n^2\sqrt{n}\epsilon.$$

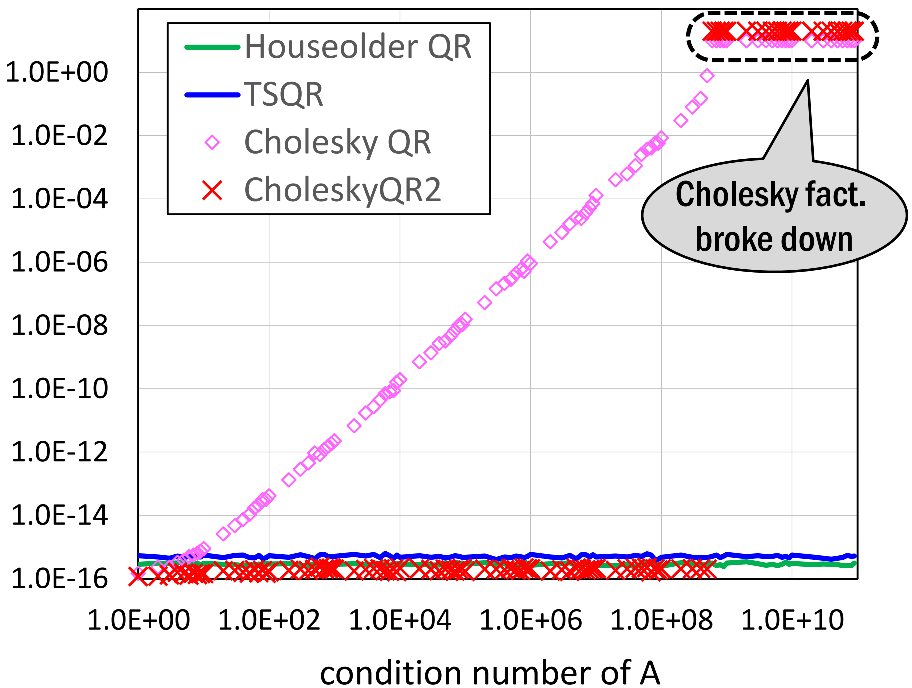
(ϵ is the unit roundoff, and $\epsilon \simeq 10^{-16}$ in double precision.)

Experimental rsults on stability

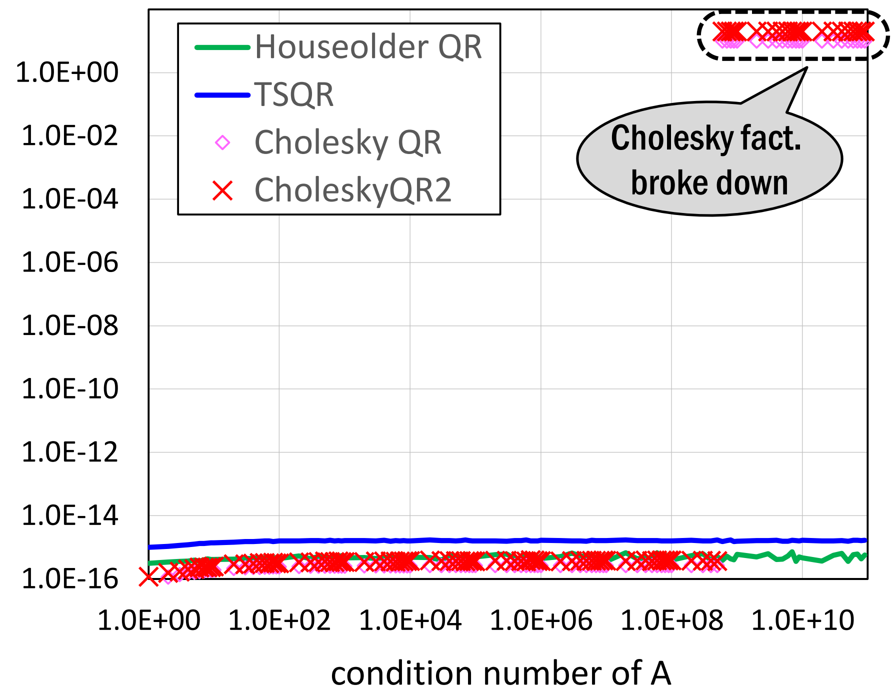
Test matrix: $A = V_1 \Sigma V_2,\ \Sigma = \mathrm{diag}(1, \sigma^{\frac{1}{n-1}}, \ldots, \sigma),\ V_1, V_2$: random orthogonal matrices

(m=2,097,152 n=64 P=8,192 @ Fx10 supercomputing system Oakleaf-FX)

orthogonality: $\|Q^\top Q - I\|_F / \sqrt{n}$



residual: $\|A - QR\|_F / \|A\|_F$



- CholeskyQR2 is as stable as TSQR until Cholesky fact. breaks down.
- Threshold of breakdown is around 10^8 (in which $\mathrm{cond}(A^\top A) = 10^{16}$).

Stability of Cholesky QR is greatly improved (,though still worse than TSQR).

Performance evaluation on the K computer

Evaluation conditions

◆ Computational environment: K computer (RIKEN, Japan)

- SPARC64™ VIIIfx (2.0 GHz, 8 cores) x 1 / node
- 16GB memory / node: DDR3 SDRAM, 64GB/s
- 6D mesh/tours network (Tofu): 5GB/s/link, bidirectional
- assignment: 1 process / node, 8 threads /process
- using MPI & BLAS (thread parallel ver.) by Fujitsu

（http://www.aics.riken.jp/）

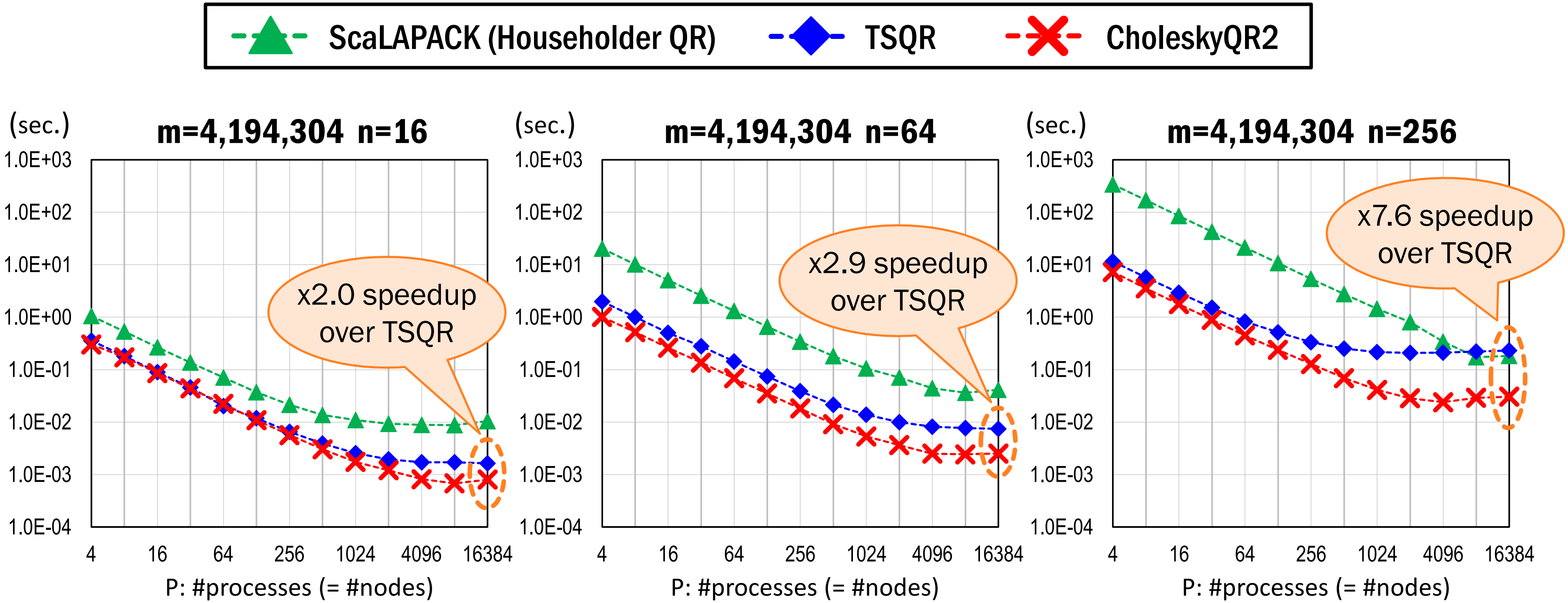
◆ Implementations

- CholeskyQR2
  - ✓ simple implementation based on BLAS, LAPACK routines
  - ✓ using DGEMM for computing $A^{\top}A$ (since DSYRK is less tuned)
- TSQR
  - ✓ general QR fact.: recursive QR based on DGEMM
  - ✓ structured QR fact.: self-coding with simple loop blocking (not using BLAS)
  - ✓ computing compact-WY representations for forming explicit Q

◆ Test problem

- computing the QR fact. (forming explicit Q) of a random matrix

Total exe. time (strong scaling)



- TSQR and CholeskyQR2 are much faster than ScaLAPACK.
  (excepting when n=256 and P is large for TSQR)
- CholeskyQR2 outperforms TSQR.
  (not only when P is large but also when P is small)

CholeskyQR2 is still efficient from the viewpoint of parallel performance.

Breakdown of execution time

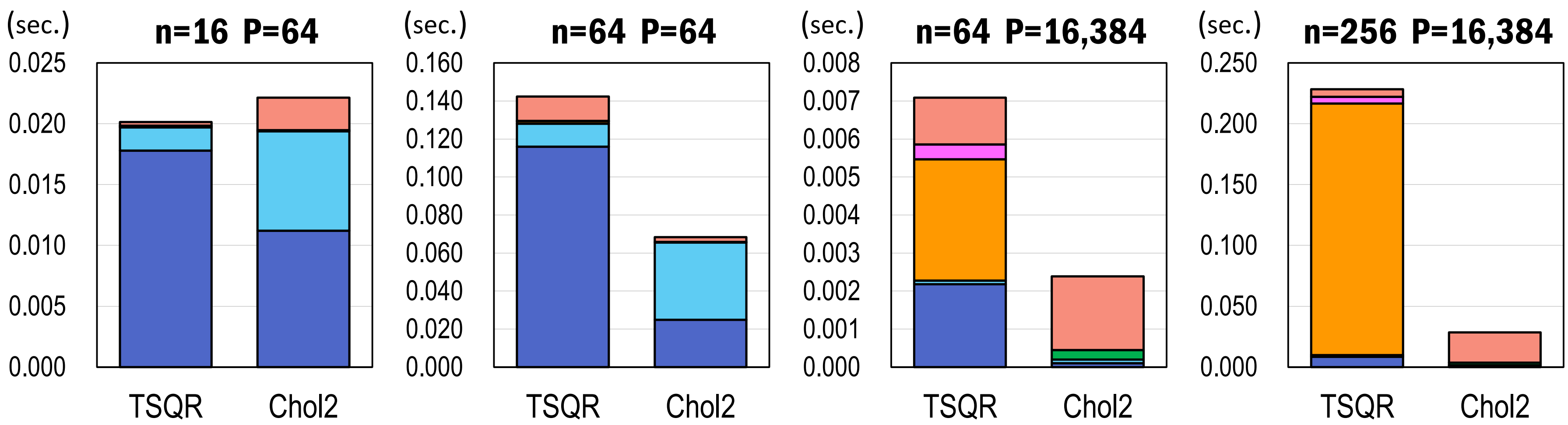
TSQR

- general QR fact.
- forming Q
- structured QR fact.
- (structured) forming Q
- communication

ChokeskyQR2 (Chol2)

- matrix-matrix mult.
- back substitution
- Cholesky fact. & forming R
- communication

(in every case, m=4,194,304)

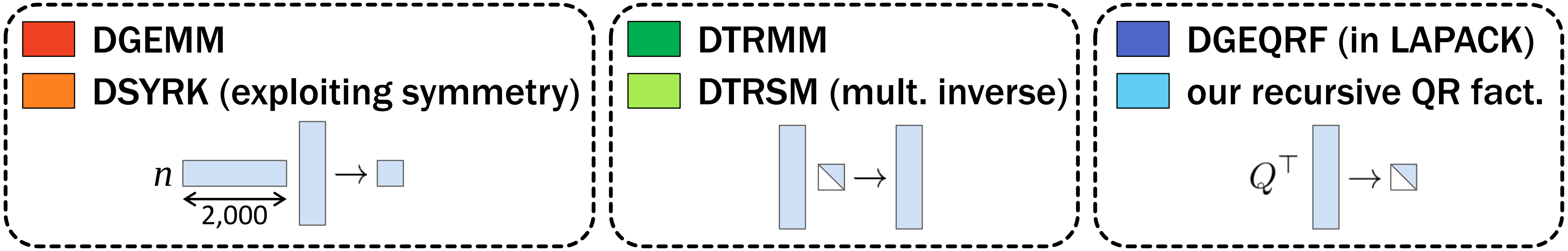


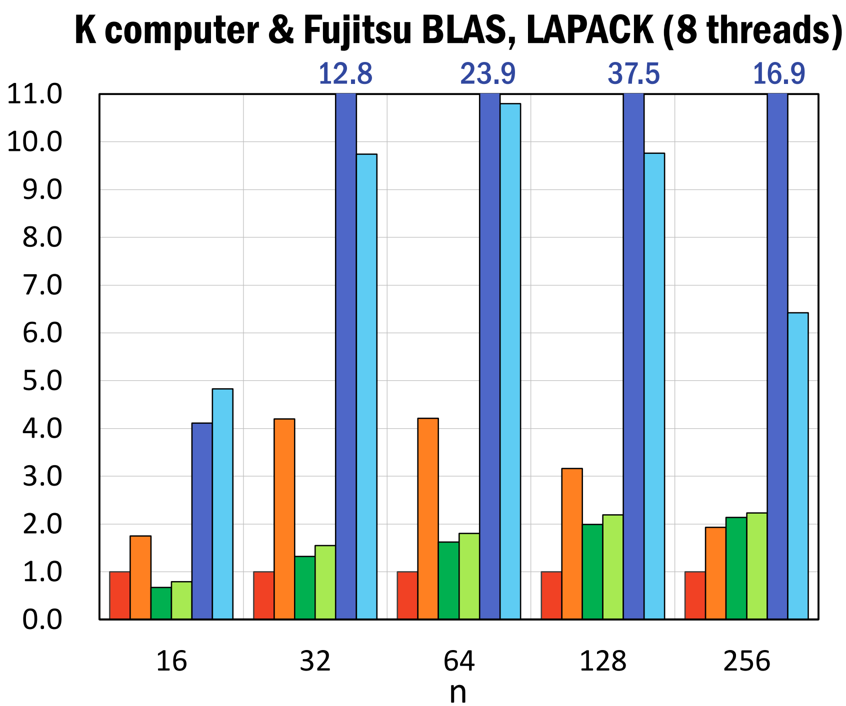
- **performance gap of local computational kernels:** matrix-matrix mult. is much higher performance than general QR fact.
- **difference of reduction operation**: cost for structured QR fact. becomes a serious bottleneck in TSQR.

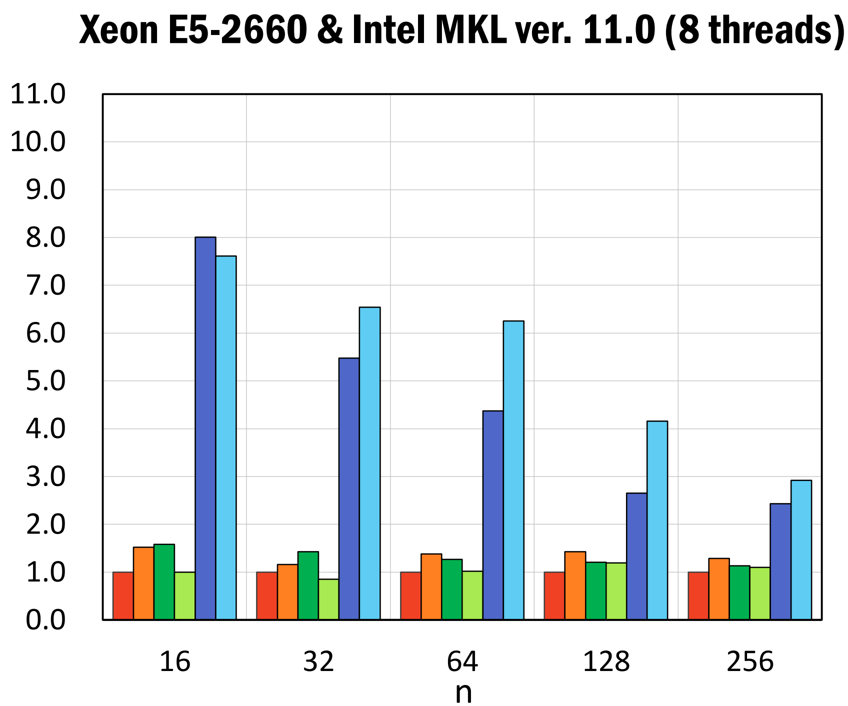
Although CholeskyQR2 requires more comm. cost, it has big advantages.

Local kernel performance

comparing relative unit time for one floating-point operation to DGEMM





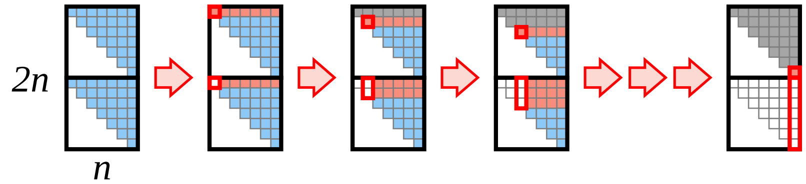


Routines in CholeskyQR2 achieve higher performance than those in TSQR.

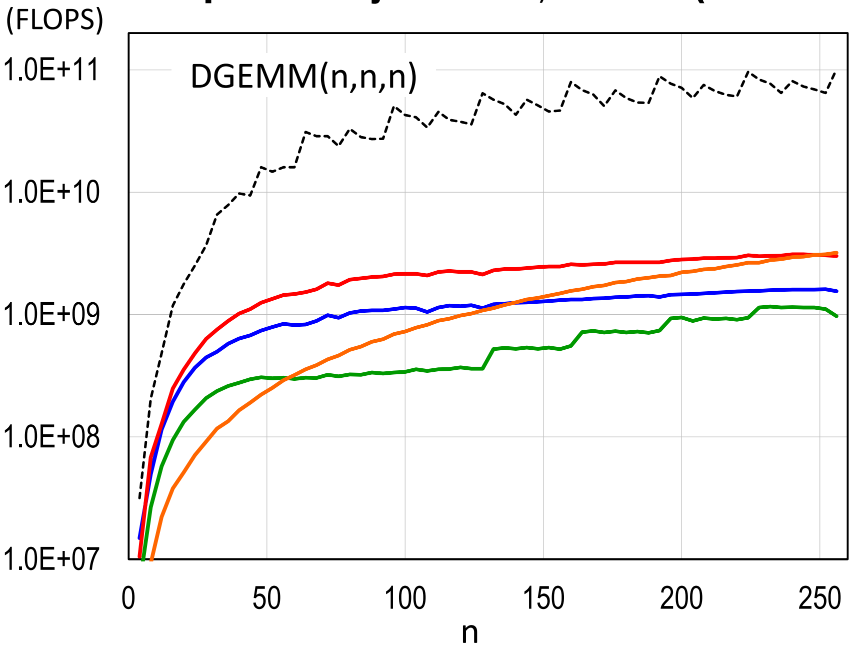
Performance of structured QR fact.

- ours (simple loop blocking)
- naive
- DTPQRT (in LAPACK)
- DGEQRF (in LAPACK, for general QR fact.)

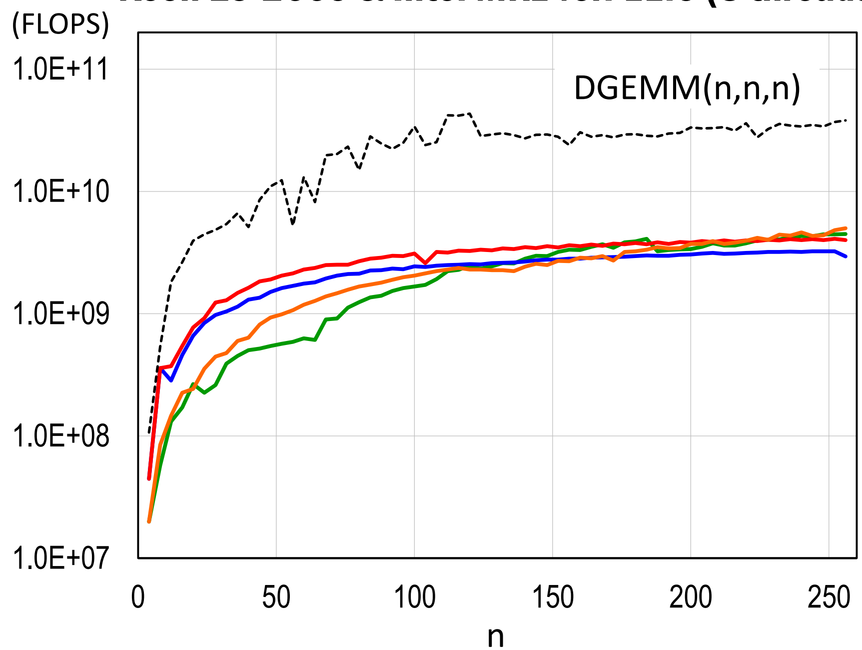
exploiting the triangular structure => reducing #flops



K computer & Fujitsu BLAS, LAPACK (8 threads)



Xeon E5-2660 & Intel MKL ver. 11.0 (8 threads)



(note: FLOPS is calculated assuming that #flops of structured QR fact. is $2n^3/3$.)

High performance implementation of structured QR fact. is difficult.

Conclusion

Conclusion

◆Summary

CholeskyQR2: repeating the Cholesky QR factorization twice

- is as stable as Householder QR (& TSQR) until $cond(A) \lesssim 10^8$.
- is still faster than TSQR (as far as on the K computer).
- can be applied to related algorithms (e.g. block Gram-Schmidt and block Householder QR by replacing panel factorization).
- is practical due to its simplicity of implementation.

◆Future work

- evaluation for much more ill-conditioned matrices
  - ✓ with double-double precision [I. Yamazaki, et al., 2014]
  - ✓ with more repeat with shift [Y. Yanagisawa, et al., 2014]
- evaluation for not tall-skinny matrices (& 2D data distribution)

For more details

- T. Fukaya, et al., CholeskyQR2: a simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system, ScalA'14, 2014.
- Y. Yamamoto, et al., Roundoff error analysis of the CholeskyQR2 algorithm , ETNA, Vol. 44, pp. 306-326, 2015.

Thank you very much