Performance evaluation of the shifted Cholesky QR algorithm for ill-conditioned matrices

Takeshi Fukaya
Hokkaido University
fukaya@iic.hokudai.ac.jp

Ramaseshan Kannan
Arup UK

Yuji Nakatsukasa
National Institute of Informatics

Yusaku Yamamoto
The University of Electro-Communications

Yuka Yanagisawa
Waseda University

ABSTRACT

The Cholesky QR algorithm, which computes the QR factorization of a matrix, is a simple yet efficient algorithm as it is rich in matrix-matrix operations. It is therefore better suited for high-performance computing than other factorization methods that depend on matrix-vector approaches. However it suffers from numerical instability. In a recent work, this instability has been remedied by repeating Cholesky QR twice; this approach is called the CholeskyQR2 algorithm. ChokeskyQR2, however, is still prone to numerical breakdown when applying to ill-conditioned matrices with condition number roughly larger than 10^8. To overcome this limitation, we introduce a shifting technique to Cholesky QR and use it as a preconditioning step before CholeskyQR2. The key idea is that Cholesky QR with shift reduces the condition number of the input matrix. We call the resulting algorithm shiftedCholeskyQR3, which is still simple and does not need extended precision arithmetic. In this poster, we present the results of our performance evaluation of shiftedCholeskyQR3. We demonstrate that shiftedCholeskyQR3 accurately computes the QR factorization of ill-conditioned matrices and that it outperforms other conventional algorithms in execution time.

1 INTRODUCTION

Let $A \in \mathbb{R}^{m\times n}$ ($m \geq n$, especially $m \gg n$). We consider computing its QR factorization $A = QR$, with $Q \in \mathbb{R}^{m\times n}$, $Q^TQ = I_n$ and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. We focus on the Cholesky QR algorithm [9], which computes the QR factorization via the Cholesky factorization of the Gram matrix $A^\top A$ (Algorithm 1) [5, Thm. 5.2.3]. Cholesky QR has nice properties from the HPC viewpoint: most of its computation can be done by Level-3 BLAS routines (`dsyrk`/`dgemm` and `dtrsm`), and it can be regarded as a communication-avoiding algorithm [2]. However, unfortunately its numerical instability prevents one from using it in practical situations. In recent work [4, 11, 12], it was shown that this instability can be remedied by simply repeating Cholesky QR twice, which is called the CholeskyQR2 algorithm (Algorithm 2). Despite these improvements one faces the issue of breakdown when applying CholeskyQR2 on an ill-conditioned matrix where the condition number of the matrix $\kappa_2(A)$ is roughly larger than 10^8 when using double precision arithmetic. A mixed-precision approach was proposed for solving this problem [13], but in order to obtain sufficient performance, highly tuned double-double precision routines are needed, which are rarely available in current standard computational environments.

Algorithm 1 Cholesky QR

Function: $[Q,\ R] = \mathrm{CholQR}(A)$
1: $W := A^\top A$
2: $R := \mathrm{chol}(W)$ // Cholesky factorization
3: $Q := AR^{-1}$

Algorithm 2 CholeskyQR2

Function: $[Q,\ R] = \mathrm{CholQR2}(A)$
1: $[A', R_1] := \mathrm{CholQR}(A)$
2: $[Q, R_2] := \mathrm{CholQR}(A')$
3: $R := R_2R_1$

In this work, we overcome the instability in CholeskyQR2 using only double precision operations. We introduce a shift technique to Cholesky QR and use it as a preconditioning step before CholeskyQR2; we call the resulting algorithm *shiftedCholeskyQR3*. Details on the derivation and stability analysis is described in [3]; here we briefly describe it and report its performance. The performance evaluation presented here shows promising results that shiftedCholeskyQR3 computes accurate QR factorizations for ill-conditioned matrices and that it outperforms other conventional algorithms in a modern multi-core environment.

2 THE SHIFTED CHOLESKY QR ALGORITHM

When applying CholeskyQR2 to an ill-conditioned (roughly $\kappa_2(A) \gtrsim 10^8$ in double precision arithmetic), it usually fails to compute the QR factorization due to the breakdown of the Cholesky factorization in the first Cholesky QR algorithm. One reason is that the computed Gram matrix is not positive definite, and the other reason is that rounding errors arise in the computation of the Cholesky factorization.

Our idea is motivated by the notion that Cholesky QR and CholeskyQR2 are classified into *triangular orthogonalization* type algorithms [10, Lecture 10]. Namely, if we find an upper triangular matrix $\tilde{R}$ that satisfies $\kappa_2(A\tilde{R}) \lesssim 10^8$, then we can compute the QR factorization of $A\tilde{R}$ using CholeskyQR2, and then obtain the QR factorization

$$A\tilde{R} = QR \quad \Rightarrow \quad A = Q(R\tilde{R}^{-1}).$$

Algorithm 3 shifted Cholesky QR

Function: $[Q, R] = \mathrm{sCholQR}(A, s)$ // $s > 0$
1: $W := A^\top A$
2: $R := \mathrm{chol}(W + sI)$ // Cholesky factorization
3: $Q := AR^{-1}$

Algorithm 4 shiftedCholeskyQR3

Function: $[Q, R] = \mathrm{sCholQR3}(A, s)$ // $s > 0$
1: $[A', R_1] := \mathrm{sCholQR}(A, s)$
2: $[Q, R_2] := \mathrm{CholQR2}(A')$
3: $R := R_2 R_1$

The question is hence how to obtain a good $\tilde{R}$. To answer this, we use the fact that if the smallest eigenvalue of a symmetric positive definite matrix is sufficiently large, its Cholesky factorization numerically succeeds [8, Thm. 2.3]. In light of this, we introduce a small shift to the computed Gram matrix and let $\tilde{R}$ be the inverse of its Cholesky factor. One can show [3] under natural assumptions that $\kappa_2(A\tilde{R}) \ll \kappa_2(A)$ if s is sufficiently small. We call this process *shifted Cholesky QR* (Algorithm 3) and consider using it as a preconditioning step before CholeskyQR2; we also refer to as *shiftedCholeskyQR3* the overall algorithm, described in Algorithm 4. It is worth repeating that no extended precision arithmetic is necessary for shiftedCholeskyQR3.

3 PERFORMANCE EVALUATION

We evaluated the performance of the shiftedCholeskyQR3 algorithm and other competitive algorithms on the Laurel 2 supercomputer system at ACCMS, Kyoto University, whose computational nodes have two 18-core Intel Xeon E5-2695v4 (2.1GHz) processors and 128GB memory. The code is written in Fortran90 using LAPACK and BLAS routines, and complied by Intel Fortran complier (`ifort` ver. 17.0.6) with options "`-O3`, `-mcmodel=medium`, `-shared-intel`, `-qopenmp`, `-ipo`, `-xHost`". We used the Intel MKL library (ver. 2017.0.6, with "`-mkl=parallel`").

Test matrices are generated by forming $A := U\Sigma V$, where U and V are random $m \times n$ and $n \times n$ orthogonal matrix, respectively, and $\Sigma = \mathrm{diag}(1, \sigma^{1/n-1}, \ldots, \sigma^{n-2/n-1}, \sigma)$ where $0 < \sigma < 1$, so $\kappa_2(A) = 1/\sigma$. In the following evaluation, we set the shift parameter $s = \sqrt{m}\|A\|_F^2 \times 10^{-16}$ for shiftedCholeskyQR3, as introduced in [3].

Table 1 compares the accuracy of each method as measured by the loss of orthogonality and residual of the computed Q and R factors. Here `dgeqrf` and `dgeqr` are Householder QR routines provided in LAPACK. We use them in combination with routines `dorgqr` and `dgemqr` for forming the explicit Q factors. It is worth noting that `dgeqr` is a novel routine that appropriately uses the TSQR algorithm [2]. From the table, we see that shiftedCholeskyQR3 successfully computes Q and R as accurately as Householder QR methods, whereas CholeskyQR2 fails due to the breakdown in the Cholesky factorization of the Gram matrix.

Table 1: Accuracy: $m = 100000$, $n = 64$, $\kappa_2(A) = 1.0 \times 10^{14}$.

Method	$\|Q^\top Q - I_n\|_F/\sqrt{n}$	$\|QR - A\|_F/\|A\|_F$
scholqr3	2.19×10^{-16}	4.20×10^{-16}
cholqr2	failed	
dgeqrf + dorgqr	3.00×10^{-16}	5.56×10^{-16}
dgeqr + dgemqr	2.75×10^{-16}	7.60×10^{-16}
cgs	3.82×10^{0}	2.72×10^{-16}
mgs	6.96×10^{-4}	2.72×10^{-16}
cgs2	5.16×10^{-16}	2.77×10^{-16}

Table 2: Execution time (in sec.) on a node of Laurel 2: $m = 100000$.

Method	$n = 32$	$n = 64$	$n = 128$
scholqr3 (syrk)	2.39×10^{-3}	6.70×10^{-3}	2.47×10^{-2}
scholqr3 (gemm)	2.58×10^{-3}	8.17×10^{-3}	3.06×10^{-2}
dgeqrf + dorgqr	6.13×10^{-2}	8.43×10^{-2}	1.19×10^{-1}
dgeqr + dgemqr	4.38×10^{-3}	1.30×10^{-2}	4.20×10^{-2}
cgs2	9.41×10^{-3}	2.24×10^{-2}	1.31×10^{-1}
block-cgs2	1.52×10^{-2} (bl = 16)	2.82×10^{-2} (bl = 32)	7.13×10^{-2} (bl = 32)

We next give the results of execution time measured by using a node of Laurel 2 (using 36 threads) in Table 2. Here we evaluate two implementations of shiftedCholeskyQR3; using `dsyrk` or `dgemm` for computing $A^\top A$. We also compare them with the block version of cgs2 [1]; the size of block is also described in the table. The table clearly shows that shiftedCholeskyQR3 outperforms other methods for a wide range of n; the syrk version is slightly faster than the gemm version in this environment.

4 CONCLUSION

We proposed a shift technique for avoiding the numerical breakdown of CholeskyQR2 when applied to an ill-conditioned matrix. Our performance results indicate that shiftedCholeskyQR3, which uses shifted Cholesky QR as a preconditioning step before CholeskyQR2, computes the QR factorization of ill-conditioned matrices robustly and efficiently, only using double precision arithmetic. It is also demonstrated that shiftedCholeskyQR3 can outperform other conventional algorithms in a modern multi-core environment.

Detailed stability analysis of shiftedCholeskyQR3 is given in the preprint [3], including shift strategies to avoid Cholesky breakdown. There we also describe and analyze an extension of shiftedCholeskyQR3 to an oblique inner product space [6, 7].

REFERENCES

[1] Jesse L. Barlow and Alicja Smoktunowicz. 2013. Reorthogonalized block classical Gram–Schmidt. *Numer. Math.* 123, 3 (01 Mar 2013), 395–423.

[2] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. 2012. Communication-optimal Parallel and Sequential QR and LU Factorizations. *SIAM J. Sci. Comp.* 34, 1 (2012), 206–239.

[3] Takeshi Fukaya, Ramaseshan Kannan, Yuji Nakatsukasa, Yusaku Yamamoto, and Yuka Yanagisawa. 2018. Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices. (2018). arXiv:1809.11085.

[4] Takeshi Fukaya, Yuji Nakatsukasa, Yuka Yanagisawa, and Yusaku Yamamoto. 2014. CholeskyQR2: a simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system. In *Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. IEEE Press, 31–38.

[5] Gene H. Golub and Charles F. Van Loan. 2012. *Matrix Computations* (4th ed.). Johns Hopkins University Press.

[6] Ramaseshan Kannan. 2013. Efficient sparse matrix multiple-vector multiplication using a bitmapped format. In *20th IEEE International Conference on High Performance Computing (HiPC'13)*. 286–294. https://doi.org/10.1109/HiPC.2013.6799135

[7] Ramaseshan Kannan. 2014. *Numerical Linear Algebra problems in Structural Analysis*. Ph.D. Dissertation. School of Mathematics, The University of Manchester.

[8] Siegfried M. Rump and Takeshi Ogita. 2007. Super-fast validated solution of linear systems. *J. Comput. Appl. Math.* 199, 2 (2007), 199–206.

[9] Andreas Stathopoulos and Kesheng Wu. 2002. A Block Orthogonalization Procedure With Constant Synchronization Requirements. *SIAM J. Sci. Comp.* 23 (2002), 2165–2182.

[10] Lloyd N. Trefethen and III David Bau. 1997. *Numerical Liner Algebra*. SIAM, Philadelphia.

[11] Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. 2015. Roundoff error analysis of the CholeskyQR2 algorihm. *Analysis Electron. Trans. Numer. Anal.* 44 (2015), 306–326.

[12] Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. 2016. Roundoff error analysis of the CholeskyQR2 algorithm in an oblique inner product. *JSIAM Letters* 8 (2016), 5–8.

[13] Ichitaro Yamazaki, Stanimire Tomov, and Jack Dongarra. 2015. Mixed-Precision Cholesky QR Factorization and Its Case Studies on Multicore CPU with Multiple GPUs. *SIAM J. Sci. Comp.* 37, 3 (2015), C307–C330.