Enhancing Parallelism of Tile Bidiagonal Transformation on Multicore Architectures using Tree Reduction

Hatem Ltaief[1], Piotr Luszczek[2], and Jack Dongarra[2,3,4]

[1] KAUST Supercomputing Laboratory Thuwal, Saudi Arabia
[2] University of Tennessee, Knoxville, TN, USA
[3] Oak Ridge National Laboratory
[4] University of Manchester

Abstract. The objective of this paper is to enhance the parallelism of the tile bidiagonal transformation using tree reduction on multicore architectures. First introduced by Ltaief et. al [LAPACK Working Note #247, 2011], the bidiagonal transformation using tile algorithms with a two-stage approach has shown very promising results on square matrices. However, for tall and skinny matrices, the inherent problem of processing the panel in a domino-like fashion generates unnecessary sequential tasks. By using tree reduction, the panel is horizontally split, which creates another dimension of parallelism and engenders many concurrent tasks to be dynamically scheduled on the available cores. The results reported in this paper are very encouraging. The new tile bidiagonal transformation, targeting tall and skinny matrices, outperforms the state-of-the-art numerical linear algebra libraries LAPACK V3.2 and Intel MKL ver. 10.3 by up to 29-fold speedup and the standard two-stage PLASMA BRD by up to 20-fold speedup, on an eight socket hexa-core AMD Opteron multicore shared-memory system.

Keywords: Bidiagonal Transformation; Tree Reduction; High Performance Computing; Multicore Architecture; Dynamic Scheduling;

1 Introduction

This paper extends our prior work with one-sided factorizations and in particular, the tridiagonal reduction (TRD) [18] to the bidiagonal reduction (BRD) case, which presents more challenges due to its increased algorithmic complexity. BRD is an important first step when calculating the singular value decomposition (SVD). Two-stage reduction algorithms for two-sided factorizations are not new approaches but have recently enjoyed rekindled interest in the community. For instance, it has been used by Bischof et al. [6] for TRD (SBR toolbox) and Kågström et al. [16] in the context of Hessenberg and Triangular reductions for the generalized eigenvalue problem for dense matrices. The tile bidiagonal reduction for square matrices that was obtained in this way considerably outperforms the state-of-the-art open-source and commercial numerical libraries [17].

BRD for any rectangular dense matrix [11, 9, 22] is: $A = U\Sigma V^T$ with $A, \Sigma \in \mathbb{R}^{M\times N}$, $U \in \mathbb{R}^{M\times M}$, and $V \in \mathbb{R}^{N\times N}$. Following the decompositional approach to matrix computation [21], we transform the dense matrix A to an upper bidiagonal form B by applying

successive distinct orthogonal transformations [15] from the left (X) as well as from the right (Y): $B = X^T A Y \quad B,X,A,Y \in \mathbb{R}^{N \times N}$. This reduction step actually represents the most time consuming phase when computing the singular values. Our primary focus is the BRD portion of the computation which can easily consume over 99% of the time needed to obtain the singular values and roughly 75% if singular vectors are calculated [17].

The necessity of calculating SVDs emerges from various computational science areas, e.g., in statistics where it is directly related to the principal component analysis method [13, 14], in signal processing and pattern recognition as an essential filtering tool and in analysis control systems [19]. However, majority of the applications and especially data collected from large sensor systems involve rectangular matrices with the number of rows by far exceeding the number of columns [4, 10]. We refer to such matrices as *tall and skinny*. For such matrices, the bulge chasing procedure (see Section 3) is no longer the bottleneck as it is the case for square matrices [17]. It is the reduction to the band form that poses a challenge which we address in this paper.

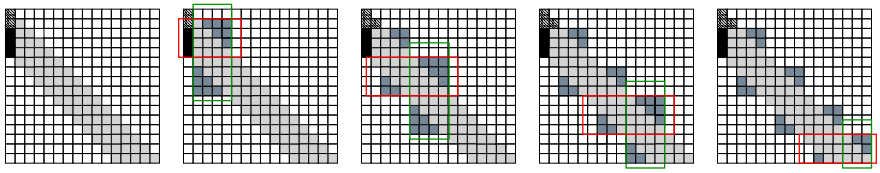
The remainder of this document is organized as follows: Sections 2 and 3 recalls the block BRD algorithm as implemented in LAPACK [1] as well as the two-stage BRD algorithm available in PLASMA [23] and explains their main deficiencies, especially in the context of tall and skinny matrices. Sections 4 gives a detailed overview of previous projects in this area and outlines the main contributions of the paper. Section 5 describes the implementation of the parallel two-stage tile BRD algorithm using a tree reduction for tall and skinny matrices. Section 6 presents the performance results. Finally, Section 7 summarizes the results of this paper and presents the ongoing work.

2 LAPACK Bidiagonal Transformation

LAPACK [1] implements a so called block variant of singular value decomposition algorithms. Block algorithms are characterized by two successive phases: a panel factorization and an update of the trailing submatrix. During the panel factorization, the orthogonal/unitary transformations are applied only within the panel one column at a time. As a result, the panel factorization is very rich in Level 2 BLAS operations. Once accumulated within the panel, the transformations are applied to the rest of the matrix (commonly called the trailing submatrix) in a blocking manner, which leads to an abundance of calls to Level 3 BLAS. While the update of the trailing submatrix is compute-bound and very efficient, the panel factorization is memory-bound and has mostly been a bottleneck for the majority of numerical linear algebra algorithms. Lastly, the parallelism within LAPACK occurs only at the level of the BLAS routines, which results in an expensive fork-join scheme of execution with synchronization around each call.

The use of tall-and-skinny matrices compounds the aforementioned inefficiencies. On one hand, the memory-bound panel factorization is now disproportionately expensive compared with the trailing matrix update. On the other hand, the fork-join parallelism does not benefit at all the execution of the panel factorization because only Level 2 BLAS may be used – memory-bound operations that only marginally benefit from parallelization. Clearly, the parallelism needs to be migrated from the BLAS level up to the factorization algorithm itself.

3 PLASMA Bidiagonal Transformation using Two-Stage Approach



(a) First column annihilation (b) Bulge creation (c) Chasing the bulge with the left and right transformations (d) Chasing the bulge further down (e) Chasing in the bottom right corner

Fig. 1. Execution breakdown of the bulge chasing procedure for a band bidiagonal matrix of size N=16 with NB=4.

In our last implementation described in [17], we fully utilize a two-stage approach – a technique that has recently proven its value as a viable solution for achieving high performance in the context of two-sided reductions [6, 16, 18]. The first stage consists of reducing the original matrix to a band form. The overhead of the Level 2 BLAS operations dramatically decreases and most of the computation is performed by the Level 3 BLAS, which makes this stage run closer to the theoretical peak of the machine. In fact, this stage has even enough computational load to benefit from offloading the work to GPU accelerators [5]. The second stage further reduces the band matrix to the corresponding compact form. A bulge chasing procedure, that uses orthogonal transformations annihilates the off-diagonal elements column-wise and eliminates the resulting fill-in elements that occur towards to the bottom right corner of the matrix. Figure 1 depicts the execution breakdown of chasing the first column (black elements) on a band bidiagonal matrix of size M=N= 16 and NB= 4.

The next section explains why this current implementation of the tile BRD using a two-stage approach is not appropriate for the case of tall and skinny matrices.

4 Related Work and Relevant Contributions

Numerical schemes based on tree reductions have been developed first for dense one-sided factorization algorithms [8]. It was done in the context of minimizing communication amount between the levels of the memory hierarchy as well as between remote parallel processors. Given the fact that such reduction schemes are numerically stable under a set of practical assumptions, we are able to apply similar schemes for the two-sided reductions.

Practical applications of these numerical reduction schemes on multicore architectures may indeed achieve very competitive results in terms of performance [12]. And the same apply equally to GPU-accelerated implementations [2] as well as the codes designed specifically for distributed memory clusters of multicore nodes [20].

To broaden the scope, Bouwmeester et al. [7] provide a taxonomy of QR variants based on, among others, the reduction algorithm for achieving unitary annihilation across the full matrix height. Accordingly, our implementation may be considered as the version named TT by the authors. And this includes both the left (QR) and the right (LQ) application of the Householder reflectors [15]. A more exhaustive study of various combinations of reduction algorithms is beyond the scope of this paper.

The communication optimality for eigenvalue and singular value decompositions (as well as other related decompositions) has been studied for the entire process of reduction to a condensed form [3]. Here, however, we only concern ourselves with the reduction to the bidiagonal form and focus primarily on the implementation aspects of a particular two-sided factorization algorithm.

Enhancing parallelism using tree reduction has already been performed in the context of one-sided factorizations for tall and skinny matrices, as mentioned earlier. However, we propose here the very first practical implementation that uses a tree reduction for BRD. We contribute the idea of splitting the sequential panel factorization step into independent subdomains, that can simultaneously be operated upon. Each subdomain computation proceeds locally in parallel. Once the local computation finishes, the reduction step is triggered using a binary tree, in which the contributions of neighbor pairwise subdomains are merged. With tile algorithms, the whole computation can be modeled as a directed acyclic graph (DAG), where nodes represent fine-grained tasks and edges correspond to data dependencies. Thanks to the dynamic scheduling framework QUARK [24], the different fine-grained tasks are processed as soon as their data dependencies are satisfied. Therefore, unnecessary synchronization points are completely removed between the steps of the local computations within each subdomain. The cores that are no longer active in the merging step do not have to wait until the end of the merging step before proceeding with the next panel. The whole computation then proceeds seamlessly by following a *producer-consumer* model.

5 Tile Bidiagonal Transformation using Tree Reduction

5.1 Methodology

In the context of tall and skinny matrices, the first stage of the standard two-stage tile BRD is not suitable anymore. Indeed, when the number of rows is substantially larger than the number of columns, i.e. $M \gg N$, the first stage becomes now the bottleneck because of the panel being processed sequentially. The goal of the new implementation of this two-stage BRD is to horizontally split the matrix into subdomains, allowing independent computational tasks to concurrently execute. A similar algorithm has been used to improve the QR factorization of tall and skinny matrices in [12]. The second stage, which reduces the band matrix to the bidiagonal form, only operates on the top matrix of size $N \times N$ and is negligible compared to the overall execution time. Therefore, the authors will primarily focus on optimizing the first stage.

5.2 Description of the Computational Kernels

This section is only intended to make the paper self-contained as the description of the computational kernels have already been done in previous author's research pa-

pers [12, 18, 17]. There are ten kernels overall, i.e. four kernels for the QR factorizations, four kernels for the LQ factorizations and two kernels in order to process the merging step. *CORE_DGEQRT* and *CORE_DGELQT* perform the QR/LQ factorization of a diagonal tile, respectively. It produces an upper (QR) or lower (LQ) triangular matrix. The upper triangular matrix is called reference tile because it will be eventually used to annihilate the subsequent tiles located below, on the same panel. *CORE_DTSQRT* and *CORE_DTSLQT* compute the QR/LQ factorization of a matrix built by coupling the upper/lower triangular matrices produced by *CORE_DGEQRT* (reference tile) and *CORE_DGELQT* with a tile located below (QR) or to the right of the diagonal (LQ), respectively. The transformations accumulated during the panel computation (characterized by the four kernels described above) are then applied to the trailing submatrix with *CORE_DORMQR* and *CORE_DORMLQ* using the Householder reflectors computed by *CORE_DGEQRT* and *CORE_DGELQT* and with *CORE_DTSMQR* and *CORE_DTSMLQ* using the Householder reflectors computed by *CORE_DTSQRT* and *CORE_DTSLQT*, respectively. The last two kernels, which perform the merging steps for tall and skinny matrices are *CORE_DTTQRT* and *CORE_DTTMQR*.

5.3 DAG Analysis

The dynamic runtime system QUARK [24] has the capability to generate DAGs of execution on the fly, which are critical in order to understand the performance numbers reported in this paper. For the next three figures, the yellow and blue nodes correspond to the tasks of the QR and LQ factorizations, respectively. The red nodes represent the tasks involved during the tree reduction, i.e. the merging step. The matrix size is defined to 10×2 in terms of number of row and column tiles. Figure 2 shows the DAG of the standard two-stage PLASMA BRD (first stage only). The bottleneck of sequentially computing the panel clearly appears. Here, the first stage would take 22 steps to achieve the desired band form. Figure 3 highlights the DAG of the two-stage PLASMA BRD using tree reduction with two subdomains. The two distinct entry points are identified, which allows the panel to proceed in parallel. Once computed, the merging phase (red nodes) can be initiated as soon as the data dependencies are satisfied. Here, the number of steps to obtain the band form has been significantly reduced to 15 steps. Finally, Figure 4 pictures the DAG of the two-stage PLASMA BRD using tree reduction with eight subdomains. The eight distinct entry points are clearly distinguished. The DAG is now more populated with red nodes due to the high number of merging steps. The number of steps to obtain the band form has been further reduced to 13 steps, while the number of concurrent tasks has drastically increased.

6 Experimental Results

6.1 Environment Setting

We have performed our tests on a shared memory machine with the largest number of cores we could access. It is composed of eight AMD Opteron^{TM} processors labelled 8439 SE. Each of the processors contains six processing cores each clocked at 2.8 GHz.

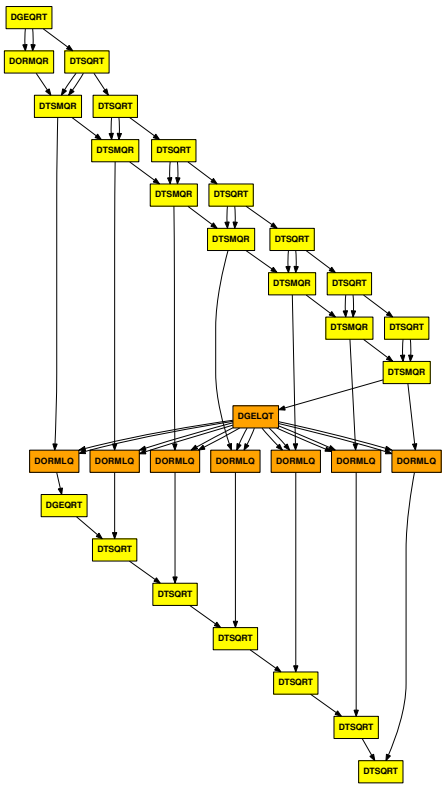


Fig. 2. DAG of the standard two-stage PLASMA BRD (first stage only) on a matrix with MT= 8 and NT= 2 tiles.

The total number of cores is evenly spread among two physical boards. The theoretical peak for this machine for double precision floating-point operations is 537.6 Gflop/s (11.2 Gflop/s per core). And the total available memory is 128 GB which is spread among 8 NUMA nodes. On the software side, we used Intel Math Kernel Library MKL version 10.3 with an appropriate threading setting to force single-thread execution. The blocking parameters we used in our tests were NB of 144 (the external tile blocking) and IB of 48 (the internal tile blocking) for our double precisions runs. All experiments have been conducted on all 48 cores to stress not only the asymptotic performance but also scalability with the largest core count we could access.

6.2 Performance Comparisons

In the figures presented in this section, we refer to the standard two-stage tile BRD as *PLASMA* and to the optimized two-stage tile BRD for tall and skinny matrices using tree reduction as *PLASMA TR*. Figure 5 shows the performance of PLASMA TR with M = 57600 and N = 2880 (both sizes are fixed) and a varying number of subdomains. When the number of subdomain is one, the implementation of *PLASMA TR* is in

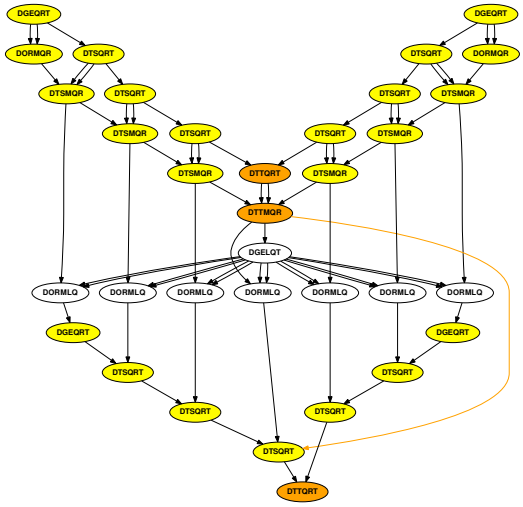


Fig. 3. DAG of the two-stage PLASMA BRD using tree reduction on a matrix with MT= 8 and NT= 2 tiles using two subdomains.

fact equivalent to the one of *PLASMA* and this is why they report the same performance number. However, when the number of subdomains increases, *PLASMA TR* rapidly outperforms *PLASMA*. Noteworthy to mention the very low rates of execution of LAPACK and MKL. This has been noticed for square matrices [17] and it is even worse for tall and skinny matrices. Figure 6 shows the performance of *PLASMA TR* with $M = 57600$ (fixed) and a varying number of column tiles. The subdomain sizes giving the best performance have been selected for *PLASMA TR*. When the the matrix has only a small number of column tiles (i.e., skinny), this is where our implementation performs the best compared to the three other libraries. *PLASMA TR* achieves up to 20-fold speedup and up to 29-fold speedup compared to *PLASMA* (with $M = 57600$ and $N = 3 \times 144 = 432$) and LAPACK and MKL (with $M = 57600$ and $N = 15 \times 144 = 2160$), respectively. The clear advantage over LAPACK stems from exposing parallelism of reduction of tall matrix panels and the good locality and plentiful parallelism of the two-stage approach. As the number of column tiles or the matrix width increases, the performance of *PLASMA* implementation starts catching up *PLASMA TR*, since the matrix becomes square.

7 Conclusions and Future Work

In this paper, we presented a new parallel implementation of the tile two-stage BRD algorithm suitable for tall and skinny matrices on shared-memory multicore architectures. Our implementation is far superior to any functionally equivalent code that we are aware of. In fact, it outperforms LAPACK and Intel MKL nearly 29-fold and PLASMA – 20-fold for matrices it was designed for. Our ongoing and future work focuses on automatic selection of domains in the tree reduction stage as well as optimal interleaving strategies for QR and LQ application of the orthogonal reflectors.

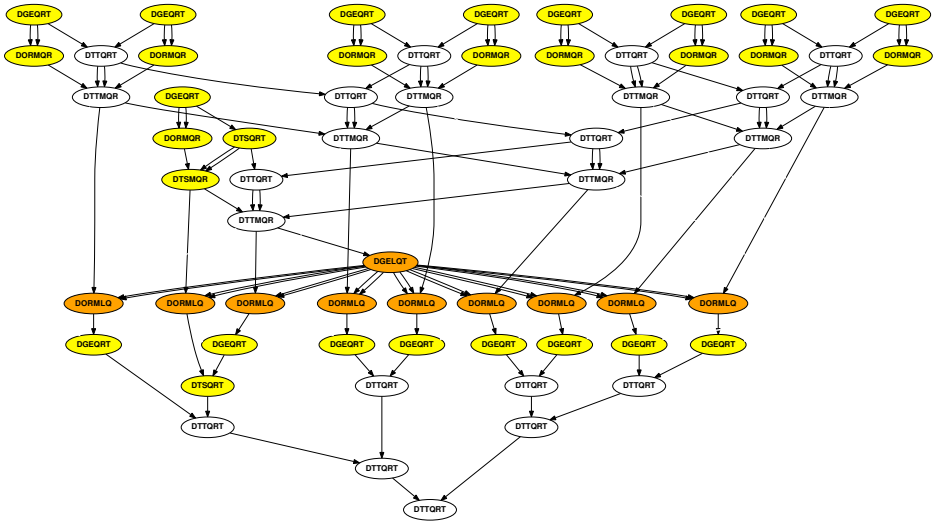


Fig. 4. DAG of the two-stage PLASMA BRD using tree reduction on a matrix with MT= 8 and NT= 2 tiles using eight subdomains.

References

1. E. Anderson, Z. Bai, C. Bischof, Suzan L. Blackford, James W. Demmel, Jack J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and Danny C. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 3rd edition, 1999.
2. Michael Anderson, Grey Ballard, James Demmel, and Kurt Keutzer. Communication-avoiding QR decomposition for GPUs. In *Proceedings of IPDPS 2011*, Anchorage, AK USA, 2011. ACM. Also available as Technical Report UCB/EECS-2010-131, Feb 18, 2011 and LAWN 240.
3. Grey Ballard, James Demmel, and Ioana Dumitriu. Minimizing communication for eigen-problems and the singular value decomposition, 2010. arXiv:1011.3077.
4. G. Balmino, S. Bruinsma, and J-C. Marty. Numerical simulation of the gravity field recovery from goce mission data. In *Proceedings of the Second International GOCE User Workshop "GOCE, The Geoid and Oceanography"*, ESA-ESRIN, Frascati, Italy, March 8-10 2004.
5. P. Bientinesi, F. Igual, D. Kressner, and E. Quintana-Orti. Reduction to Condensed Forms for Symmetric Eigenvalue Problems on Multi-core Architectures. *Parallel Processing and Applied Mathemetics*, pages 387–395, 2010.
6. Christian H. Bischof, Bruno Lang, and Xiaobai Sun. Algorithm 807: The SBR Toolbox—software for successive band reduction. *ACM Trans. Math. Softw.*, 26(4):602–616, 2000.
7. Henricus Bouwmeester, Mathias Jacquelin, Julien Langou, and Yves Robert. Tiled qr factorization algorithms. Technical Report RR-7601, INRIA, 2011.
8. James W. Demmel, Laura Grigori, Mark F. Hoemmen, and Julien Langou. Communication-optimal parallel and sequential qr and lu factorizations. Technical Report 204, LAPACK Working Note, August 2008.
9. G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

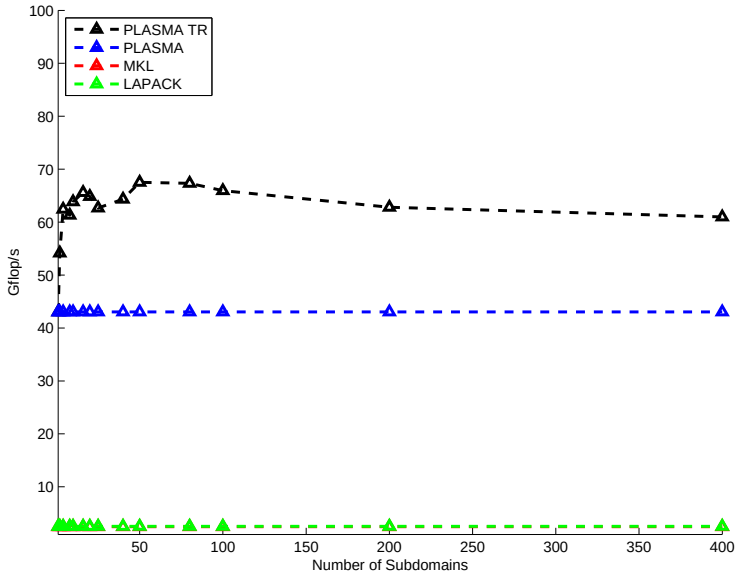


Fig. 5. Performance of *PLASMA TR* with M = 57600 and N = 2880 (fixed) and a varying number of subdomains on 48 AMD Opteron cores.

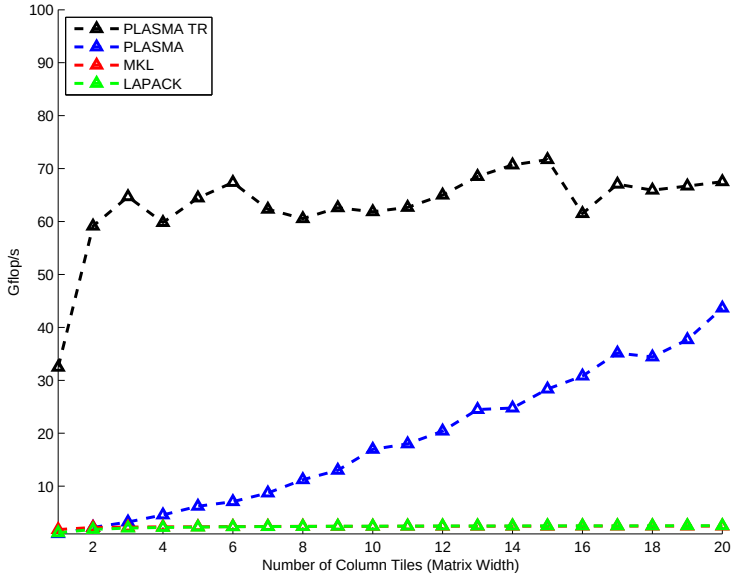


Fig. 6. Performance of *PLASMA TR* with M = 57600 (fixed) and a varying number of column tiles on 48 AMD Opteron cores.

10. Gene H. Golub, P. Manneback, and Ph. L. Toint. A comparison between some direct and iterative methods for certain large scale geodetic least squares problems. *SIAM J. Scientific Computing*, 7(3):799–816, 1986.
11. Gene H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14:403–420, 1970.
12. Bilel Hadri, Hatem Ltaief, Emmanuel Agullo, and Jack Dongarra. Tile QR factorization with parallel panel processing for multicore architectures. In *IPDPS*, pages 1–10. IEEE, 2010.
13. H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych.*, 24:417–441, 498–520, 1933.
14. H. Hotelling. Simplified calculation of principal components. *Psychometrica*, 1:27–35, 1935.
15. Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM (JACM)*, 5(4), October 1958. DOI 10.1145/320941.320947.
16. Bo Kågström, Daniel Kressner, Enrique Quintana-Orti, and Gregorio Quintana-Orti. Blocked Algorithms for the Reduction to Hessenberg-Triangular Form Revisited. *BIT Numerical Mathematics*, 48:563–584, 2008.
17. Hatem Ltaief, Piotr Luszczek, and Jack Dongarra. High performance bidiagonal reduction using tile algorithms on homogeneous multicore architectures. Technical report, 2011. LAPACK Working Note 247.
18. Piotr Luszczek, Hatem Ltaief, and Jack Dongarra. Two-stage tridiagonal reduction for dense symmetric matrices using tile algorithms on multicore architectures. In *Proceedings of IPDPS 2011*, Anchorage, AK USA, 2011. ACM.
19. Bruce C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, AC-26(1), February 1981.
20. Fengguang Song, Hatem Ltaief, Bilel Hadri, and Jack Dongarra. Scalable tile communication-avoiding qr factorization on multicore cluster systems. In *Proceedings of SC'10*, New Orleans, Louisiana, November 2010. ACM. Also available as Technical Report UT-CS-10-653 Mar 4, 2011 and LAWN 241.
21. G. W. Stewart. The decompositional approach to matrix computation. *Computing in Science & Engineering*, 2(1):50–59, Jan/Feb 2000. ISSN: 1521-9615; DOI 10.1109/5992.814658.
22. L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997. http://www.siam.org/books/OT50/Index.htm.
23. University of Tennessee. *PLASMA Users' Guide, Parallel Linear Algebra Software for Multicore Archtectures, Version 2.2*, November 2009.
24. Asim YarKhan, Jakub Kurzak, and Jack Dongarra. Quark users' guide: Queueing and runtime for kernels. Technical Report ICL-UT-11-02, University of Tennessee, Innovative Computing Laboratory, 2011.