Parallel Variants of the Multishift QZ Algorithm with Advanced Deflation Techniques*

Björn Adlerborn, Bo Kågström, and Daniel Kressner

Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden.
{adler,bokg,kressner}cs.umu.se

Abstract. The QZ algorithm reduces a regular matrix pair (H, T) in Hessenberg-triangular form to a generalized Schur form (S, T), which can be used to address the generalized eigenvalue problem. A novel parallel implementation of a small-bulge multishift QZ algorithm is presented. The algorithm chases chains of small bulges instead of only one bulge in each QZ iteration and makes use of delayed updates, which makes it more amenable to a parallel setting. In addition, advanced deflation strategies are used, specifically the so called aggressive early deflation. Recent progress will be reported including parallel performance results.

1 Introduction

In this paper, we consider parallel variants of the multishift QZ algorithm with aggressive early deflation discussed in [2]. All variants of the QZ algorithm [9] for computing the eigenvalues of a regular matrix pair $(A, B) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$ rely on computing orthogonal matrices Q and Z such that $(S, T) = (Q^T AZ, Q^T BZ)$ is in real generalized Schur form. The parallel reduction from (A, B) to (S, T) proceeds in three stages [6]:

$$(A, B) \overset{\text{Stage 1}}{\longrightarrow} (H_r, T) \overset{\text{Stage 2}}{\longrightarrow} (H, T) \overset{\text{Stage 3}}{\longrightarrow} (S, T).$$

Stage 1 reduces (A, B) to block upper Hessenberg-triangular form (H_r, T) using mainly level 3 (matrix-matrix) operations. In the second stage, all but one of the r subdiagonals of H_r are set to zero using Givens rotations, leading to (H, T) in Hessenberg-triangular form. The third stage computes the generalized Schur form (S, T) by applying QZ iterations to (H, T).

Parallel distributed memory (DM) algorithms and implementations for the first two stages have been presented in [1]. In this contribution, we consider the third stage, QZ iterations. Our new parallel variants are based on the LAPACK [3] implementation of the QZ algorithm as well as the blocked variants described in [6, 8]. In order to gain better performance and scalability, we employ the following extensions:

* This research was conducted using the resources of the High Performance Computing Center North (HPC2N).

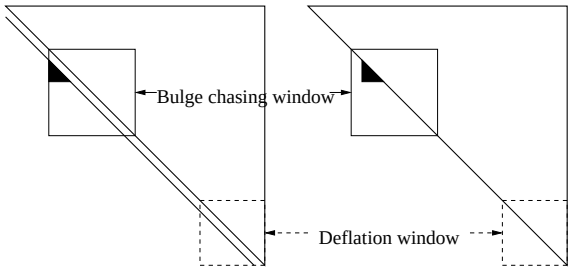

Fig. 1. Illustration of a multishift QZ step with aggressive early deflation.

- Use of several small bulges introduced and chased down the diagonals of H and T in a blocked manner.
- Accumulation of orthogonal transformations in order to apply them in a blocked manner, leading to delayed updates.
- Use of the aggressive early deflation technique which drastically reduces the number of QZ iterations needed. This method is described in detail in [5, 8].

2 Parallel QZ algorithm

Given a Hessenberg-triangular matrix pair (H, T), a parallel QZ iteration is divided into three major operations, which are implemented in separate routines: (1) deflation check; (2) bulge introduction; (3) bulge chasing. In the following, we give a brief description of these operations.

2.1 Parallel QZ Step – Deflation check

The deflation check routine searches and tests for deflated eigenvalues at the bottom right corners of H and T using the aggressive early deflation technique, see [5, 8]. This routine also returns the shifts, calculated eigenvalues from a bottom right submatrix pair of H and T within the current deflation window, see Figure 1, needed to start up a new bulge introduction and chase iteration.

The deflation check is performed by all processors and therefore communication is required before all processors have the required data. The output of the deflation check is, beside the deflated window, two orthogonal matrices which contain the accumulated equivalence transformations. If deflation was successful, these transformations are applied to the right and above the deflation window. The update is performed in parallel using general matrix multiply and add (GEMM) operations. Some nearest neighbor communication is required to be able to perform the multiplications. The subsequent QZ iterations are restricted to the deflated submatrix pair, also called the *active submatrix pair*.

2.2 Parallel QZ Step – Bulge introduction

The introduction of bulges takes place at the top left corner of the active submatrix pair. The number of bulges that can be created depends on how many

shifts were returned from the last deflation check. At most #shifts/2 bulges are created using information from the top left corner of (H, T) to compute the first column of the shift polynomial.

After a bulge has been introduced it has to be chased down some steps in order to give room for a new bulge. If $N < n$ bugles are to be introduced the first bulge is chased $N \cdot (n_H + 1)$ positions, where n_H is the size of the Householder transformation, the second $(N-1) \cdot (n_H + 1)$ positions and so forth ($n_H = 3$ in general). The chasing consists of applying Householder transformations to (H, T) from the left and right. We limit the update of (H, T) to a window of size $NB \times NB$. The orthogonal updates are also applied to two matrices U and V, initially set to the identity matrix. This way we can introduce all bulges and after that update the remaining parts of (H, T) by using GEMM operations with U and V to complete the calculation of the corresponding equivalence transformation $(Q^T HZ, Q^T TZ)$.

The window of size $NB \times NB$ is held by all processors. Communication is therefore required to send the data to all the processors. The subsequent update is performed in parallel where every processor updates its corresponding portion of (H, T). The communication in the update part is limited to nearest neighbor processors, interchanging matrix border elements (row and column data) to be able to perform the GEMM operations independently in parallel.

2.3 Parallel QZ Step – Bulge chasing

The introduced bulges are repeatedly moved together within a bulge chasing window, see Figure 1, of size $NB \times NB$. The movement begins by moving the first introduced bulge until the bottom of the bulge chasing window. This is then repeated by moving each bulge the same number of steps. As in the introduction phase the bulge movement arises from applying pairs of (left and right) Householder transformations. Moreover, the update of (H, T) is again limited to the window of size $NB \times NB$ and the update of the remaining parts is performed afterwards in parallel as described in Section 2.2.

3 Performance results

A preliminary Fortran implementation of the described algorithms has been developed based on BLACS and ScaLAPACK [4]. In the following, we report some preliminary results of experiments performed on a Linux cluster consisting of 190 HP DL145 nodes, with dual AMD Opteron 248 (2.2GHz) and 8 GB memory per node, connected in a Myrinet 2000 high speed interconnect. The AMD Opteron 248 has a 64 kB instruction and 64 kB data L1 Cache (2-way associative) and a 1024 kB unified L2 Cache (16-way associative). Figure 2 gives a brief but representative impression of the obtained timings. It can be seen from Figure 2 that the new parallel variant of the QZ algorithm is significantly faster than the ScaLAPACK implementation of the QR algorithm [7]. (Note that the traditional QZ algorithm takes roughly *twice* the computational effort

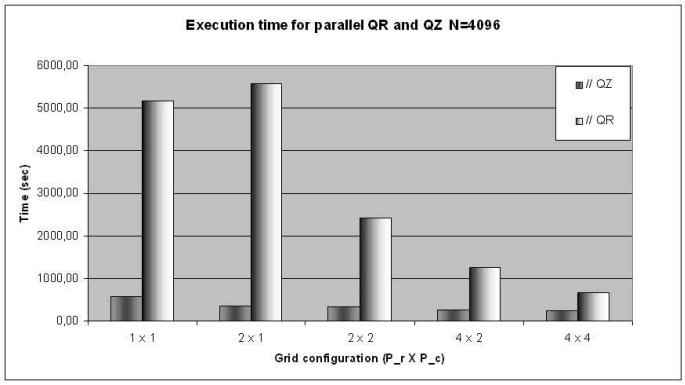

Fig. 2. Execution times for ScaLAPACK's QR for a 4096×4096 random Hessenberg matrix and parallel QZ for a 4096×4096 random Hessenberg-triangular matrix pair.

of the QR.) This effect can be contributed to the use of blocking techniques and aggressive early deflation. However, it can also be seen that the scalability of the parallel QZ is improvable; this issue will be subject to further investigation.

References

1. B. Adlerborn, K. Dackland, and B. Kågström. Parallel and blocked algorithms for reduction of a regular matrix pair to Hessenberg-triangular and generalized Schur forms. In J. Fagerholm et al., editor, *PARA 2002*, LNCS 2367, pp 319–328, 2002.
2. B. Adlerborn, D. Kressner, and B. Kågström. The multishift QZ algorithm with aggressive early deflation. In *PARA 2006*, 2006.
3. E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.
4. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.
5. K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. II. Aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 23(4):948–973, 2002.
6. K. Dackland and B. Kågström. Blocked algorithms and software for reduction of a regular matrix pair to generalized Schur form. *ACM Trans. Math. Software*, 25(4):425–454, 1999.
7. G. Henry, D. S. Watkins, and J. J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. *SIAM J. Sci. Comput.*, 24(1):284–311, 2002.
8. B. Kågström and D. Kressner. Multishift variants of the QZ algorithm with aggressive early deflation. Report UMINF-05.11, Dept. Computing Science, Umeå Univ., Umeå, Sweden, 2005.
9. C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973.