# Parallel Variants of the Multishift QZ Algorithm with Advanced Deflation Techniques[⋆]

Björn Adlerborn, Bo Kågström, and Daniel Kressner

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

**Abstract.** The QZ algorithm reduces a regular matrix pair to generalized Schur form, which can be used to address the generalized eigenvalue problem. This paper summarizes recent work on improving the performance of the QZ algorithm on serial machines and work in progress on a novel parallel implementation. In both cases, the QZ iterations are based on chasing chains of tiny bulges. This allows to formulate the majority of the computation in terms of matrix-matrix multiplications, resulting in natural parallelism and better performance on modern computing systems with memory hierarchies. In addition, advanced deflation strategies are used, specifically the so called aggressive early deflation, leading to a considerable convergence acceleration and consequently to a reduction of floating point operations and computing time.

## 1 Introduction

The QZ algorithm is the most widely used method for computing all $n$ eigenvalues $\lambda$ of a regular matrix pair $(A, B)$ with $A, B \in \mathbb{R}^{n \times n}$, which satisfy

$$\det(A - \lambda B) = 0.$$

The QZ algorithm was developed by Moler and Stewart in [19] and relies on computing orthogonal matrices $Q$ and $Z$ such that $(S, T) = (Q^T A Z, Q^T B Z)$ is in real generalized Schur form, i.e., $S$ is quasi-upper triangular with $1 \times 1$ and $2 \times 2$ blocks on the diagonal, while $T$ is upper triangular. This equivalence transformation preserves the eigenvalues of $(A, B)$, which then can be easily extracted from the block diagonals of $S$ and $T$. The LAPACK [2] implementation of the QZ algorithm is mainly based on [19], with some improvements proposed in [13,22,24]. It consists of the following subroutines:

DGGBAL performs an optional preliminary balancing step [23] aiming to improve the accuracy of subsequently computed eigenvalues.

---

DGGHRD  reduces a general matrix pair $(A, B)$ to Hessenberg-triangular form, i.e.,
it computes in a finite number of steps orthogonal matrices $Q_1$ and $Z_1$ such
that $H = Q_1^T A Z_1$ is upper Hessenberg while $T = Q_1^T B Z_1$ is upper triangular.

DHGEQZ  reduces $(H, T)$ further, by applying single- and double-shift QZ itera-
tions combined with deflations, to real generalized Schur form.

DTGSEN and DTGEVC post-process the output of DHGEQZ to compute selected
eigenvectors and deflating subspaces [12] of $(A, B)$.

Additionally, there are a number of support and driver routines for solving
generalized eigenvalue problems. The focus of the improvements described in
the following are the QZ iterations and deflations implemented in DHGEQZ, see
also [1,11]. Improvements to DGGBAL, DGGHRD and DTGSEN, which are also consid-
ered for inclusion in the next LAPACK release, can be found in [8,15,16,18,21].

The rest of this paper is organized as follows. Section 2 is concerned with
techniques intented to decrease the execution time of the QZ algorithm on serial
machines: chains of tightly coupled tiny bulges [5,11,17] and aggressive early
deflation [6,11]. In Section 3, it is shown how these techniques can be employed
to derive a parallel variant of the QZ algorithm. Numerical experiments, reported
in Section 4, illustrate the performance of Fortran implementations based on the
ideas presented in this paper.

## 2   Improvements to the Serial QZ Algorithm

Let us consider a regular matrix pair $(H, T)$ in Hessenberg-triangular form. By a
preliminary deflation of all infinite eigenvalues [19], we may assume without loss
of generality that $T$ is nonsingular. In the following, we only describe the two
improvements of the QZ algorithm proposed in [11] that make our implemen-
tation perform so well in comparison with existing implementations. However,
it should be emphasized that for a careful re-implementation of LAPACK's QZ
algorithm one also needs to reinvestigate several somewhat detailed but never-
theless important issues, such as the use of ad hoc shifts to avoid convergence
failures and the optimal use of the pipelined QZ iterations described in [8] for
addressing medium-sized subproblems.

### 2.1   Multishift QZ Iterations

The traditional implicit double-shift QZ iteration [19] starts with computing the
vector

$$v = (HT^{-1} - \sigma_1 I)(HT^{-1} - \sigma_2 I), \tag{1}$$

where $I$ denotes the $n \times n$ identity matrix and $\sigma_1, \sigma_2 \in \mathbb{C}$ are suitably chosen
shifts. Next an orthogonal matrix $Q$ (e.g., a Householder reflector [9]) is com-
puted such that $Q^T v$ is mapped to a scalar multiple of the first unit vector $e_1$.
This transformation is applied from the left to $H$ and $T$:

$$H \leftarrow Q^T H, \qquad T \leftarrow Q^T T.$$
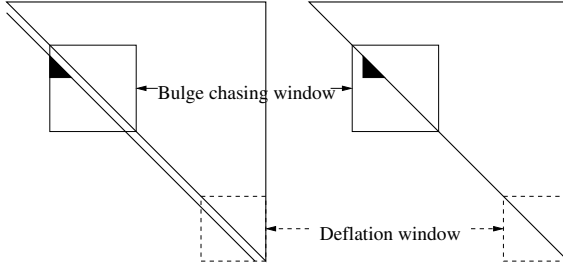
**Fig. 1.** Illustration of a multishift QZ step with aggressive early deflation

The Hessenberg-triangular structure of the updated matrix pair is destroyed in the first three rows and the rest of the implicit QZ iteration consists of reducing it back to Hessenberg-triangular form without touching the first row of $H$ or $T$. Due to the special structure of $(H, T)$, this process requires $O(n^2)$ flops and can be seen as chasing a pair of $3 \times 3$ bulges along the subdiagonals of $H$ and $T$ down to the bottom right corner, see [19,24]. If the shifts are chosen to be the eigenvalues of the $2 \times 2$ lower bottom submatrix pair of $(H, T)$ then typically the $(n - 1, n - 2)$ subdiagonal entry of $H$ converges to zero. Such a subdiagonal entry is explicitly set to zero if it satisfies

$$|h_{j+1,j}| \leq \mathbf{u}(|h_{jj}| + |h_{j+1,j+1}|), \tag{2}$$

where $\mathbf{u}$ denotes the unit roundoff. This criterion not only ensures numerical backward stability but may also yield high relative accuracy in the eigenvalues for graded matrix pairs, see [11] for more details. Afterwards, the QZ iterations are continued on the deflated lower-dimensional generalized eigenvalue problems.

The described QZ iteration performs $O(n^2)$ flops while accessing $O(n^2)$ memory. This poor computation/communication ratio limits the effectiveness of the QZ algorithm for larger matrices. An idea which increases the ratio without affecting the convergence of QZ iterations, has been extrapolated in [11] from existing techniques for the QR algorithm, see, e.g., [5]. Instead of only one bulge pair corresponding to one double shift, a tightly coupled chain of bulge pairs corresponding to several double shifts is introduced and simultaneously chased. This allows the use of level 3 BLAS without a significant increase of flops in the overall QZ algorithm.

The implementation of such a multishift QZ iteration is illustrated in Figure 1. In the beginning of a chasing step, the bulge chain resides in the top left corner of the bulge chasing window. Each bulge is subsequently chased downwards until the complete bulge chain arrives at the bottom right corner of the window. During this process only the window parts of $H$ and $T$ are updated. All resulting orthogonal transformations are accumulated and applied in terms of matrix-matrix multiplications (GEMM) to the rest of the matrix pair.

## 2.2  Aggressive Early Deflation

Another ingredient, which may drastically lower the number of iterations needed
by the QZ algorithm, is aggressive early deflation introduced in [6] and extended
in [1,11]. Additionally to the classic deflation criterion (2), the following strategy
is implemented. First, $H$ and $T$ are partitioned

$$(H, T) = \left( \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{bmatrix}, \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ 0 & T_{22} & T_{23} \\ 0 & 0 & T_{33} \end{bmatrix} \right),$$

such that $H_{32} \in \mathbb{R}^{m \times 1}$ and $H_{33}, T_{33} \in \mathbb{R}^{m \times m}$ (typical choices of $m$ are between
40 and 240). Then the matrix pair $(H_{33}, T_{33})$, which corresponds to the deflation
window illustrated in Figure 1, is reduced to real generalized Schur form. By
applying the corresponding left orthogonal transformation to $H_{32}$, a spike is
introduced in $H$. If the trailing $d \le m$ spike elements can be safely set to zero
(see [6] for various criteria) then the bottom right $d \times d$ submatrix pair can
be deflated. Otherwise, the Schur form of $(H_{33}, T_{33})$ is reordered to move other,
untested eigenvalues to its bottom right corner, see [11] for more implementation
details.

## 3   A Parallel QZ Algorithm

Parallel distributed memory (DM) algorithms and implementations for reducing
a matrix pair to Hessenberg-triangular form have been presented in [1,7]. In this
contribution, we consider the remaining part of the QZ algorithm, QZ iterations.
Our new parallel variants are based on the LAPACK [2] implementation of the
QZ algorithm as well as the blocked serial variants described in [8,11]. In order
to gain better performance and scalability, we employ the following extensions:

- 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 described in Section 2.2.

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.

### 3.1   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 [6,11]. 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 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*.

## 3.2   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 $(Q^T H Z, Q^T T Z)$.

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.

## 3.3   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 3.2.

# 4  Numerical Experiments

In the following, we briefly report on numerical experiments performed on a Linux cluster Sarek at HPC2N, 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).

## 4.1  Serial Results

First, we tested a random $2000 \times 2000$ matrix pair reduced to Hessenberg-triangular form. LAPACK's DHGEQZ requires 270 seconds, while the multishift QZ algorithm described in Section 2.1 with 60 simultaneous shifts requires 180 seconds (on machines with smaller L2 cache this reduction was observed to be even more significant). Applying aggressive early deflation with deflation window size 200 reduced the execution time further, to remarkable 28 seconds. This significant reduction of execution time carries over to other, practically more relevant examples. Table 1 contains a list of benchmark examples from [11] with order $n \geq 1900$.

**Table 1.** Selected set of matrix pairs from the Matrix Market collection [3], the Oberwolfach model reduction benchmark collection [14], and corner singularities computations [20]

| # Name | $n$ | Brief description | Source |
|---|---|---|---|
| 1 **HEAT** | 1900 | Heat conduction through a beam | [14] |
| 2 **BCSST26** | 1922 | Seismic analysis, nuclear power station | [3] |
| 3 **BEAM** | 1992 | Linear beam with damping | [14] |
| 4 **BCSST13** | 2003 | Fluid flow | [3] |
| 5 **CIRC90** | 2166 | Circular cone, opening angle 90 degrees | [20] |
| 6 **FICH1227** | 2454 | Fichera corner, Dirichlet boundary conditions | [20] |
| 7 **BCSST23** | 3134 | Part of a 3D globally triangularized building | [3] |
| 8 **MHD3200** | 3200 | Alfven spectra in magnetohydrodynamics | [3] |
| 9 **BCSST24** | 3562 | Calgary Olympic Saddledome arena | [3] |
| 10 **BCSST21** | 3600 | Clamped square plate | [3] |

We compared the performance of three Fortran implementations of the QZ algorithm: DHGEQZ (LAPACK), KDHGEQZ (pipelined QZ iterations [8]), MULTIQZ (multishift QZ iterations + aggressive early deflation). From Figure 2, which shows the execution time ratios DHGEQZ/KDHGEQZ and DHGEQZ/MULTIQZ, it can be observed that MULTIQZ is 2–12 times faster than LAPACK. It should be noted that aggressive early deflation can also be used to decrease the execution times of DHGEQZ and KDHGEQZ, see [11] for more details.
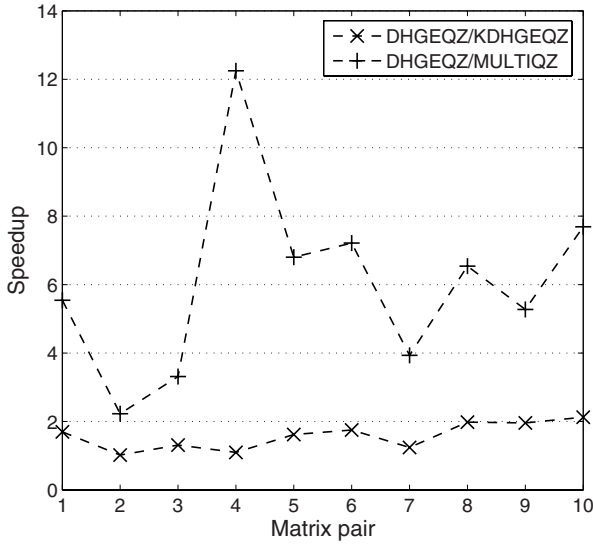
**Fig. 2.** Performance comparison on Sarek for of three serial implementations of the QZ algorithm. The test matrices used are from Table 1.
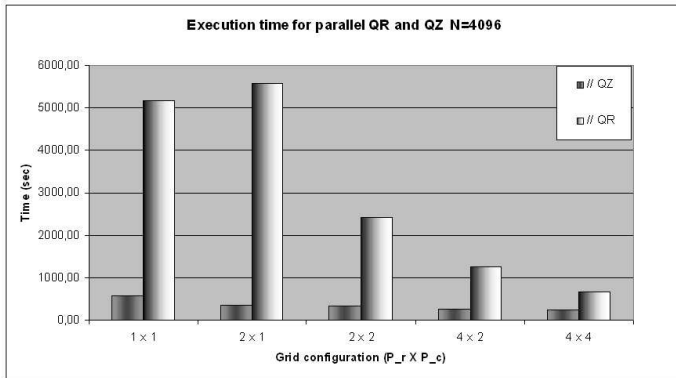


**Fig. 3.** Execution times for ScaLAPACK's QR for a 4096 × 4096 random Hessenberg matrix and parallel QZ for a 4096 × 4096 random Hessenberg-triangular matrix pair

### 4.2   Parallel Results

A preliminary Fortran implementation of the described parallel variant of the QZ algorithm has been developed based on BLACS and ScaLAPACK [4].

Figure 3 gives a brief but representative impression of the obtained timings. It can be seen from Figure 3 that the new parallel variant of the QZ algorithm is significantly faster than the ScaLAPACK implementation of the QR algorithm [10].

(Note that the traditional QZ algorithm takes roughly *twice* the computational effort of the QR.) This effect can be contributed to the use of blocking techniques and aggressive early deflation. In Table 2, we display performance results on Sarek for 1 up to 16 processors of the three stages in reducing a regular matrix pair to generalized Schur form. Stage 1 reduces a regular $(A, B)$ to block upper Hessenberg-triangular form $(H_r, T)$ using mainly level 3 (matrix-matrix) operations [1,8]. In Stage 2, 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 [1,8]. Finally, Stage 3 computes the generalized Schur form $(S, T)$ by applying our parallel QZ implementation. The overall speedup for the complete reduction to generalized Schur form is over $\sqrt{p}$, where $p$ is the number of processors used. From the performance results presented it can be seen that the scalability of the parallel QZ (Stage 3) is improvable; this issue will be subject to further investigation.

**Table 2.** Sample performance results on Sarek for 1 up to 16 processors of the three stages in reducing a regular matrix pair to generalized Schur form

| Configuration | | | | Stage 1 | Stage 2 | Stage 3 | Total | |
|---|---|---|---|---|---|---|---|---|
| $N$ | $P_r$ | $P_c$ | $NB$ | *Time* | *Time* | *Time* | *Time* | $S_P$ |
| 1024 | 1 | 1 | 160 | 5.8 | 19.5 | 13.3 | 38.5 | 1.0 |
| 1024 | 2 | 1 | 160 | 3.3 | 12.4 | 11.5 | 27.2 | 1.4 |
| 1024 | 2 | 2 | 160 | 2.8 | 7.8 | 11.8 | 22.3 | 1.7 |
| 2048 | 1 | 1 | 160 | 55.1 | 188.3 | 67.8 | 311.1 | 1.0 |
| 2048 | 2 | 2 | 160 | 21.2 | 64.6 | 62.5 | 148.3 | 2.1 |
| 2048 | 4 | 2 | 160 | 13.8 | 37.1 | 49.9 | 100.8 | 3.1 |
| 2048 | 4 | 4 | 160 | 14.0 | 28.6 | 43.5 | 86.1 | 3.6 |
| 2048 | 8 | 2 | 160 | 11.9 | 28.5 | 43.3 | 83.7 | 3.7 |
| 4096 | 1 | 1 | 160 | 551.3 | 1735.4 | 494.1 | 2780.7 | 1.0 |
| 4096 | 2 | 2 | 160 | 166.9 | 475.5 | 365.9 | 1008.3 | 2.8 |
| 4096 | 4 | 2 | 160 | 99.3 | 294.7 | 303.2 | 679.2 | 4.1 |
| 4096 | 4 | 4 | 160 | 97.6 | 245.6 | 248.2 | 591.4 | *4.7* |
| 4096 | 8 | 2 | 160 | 78.9 | 204.9 | 231.4 | 515.2 | *5.4* |

# References

1. Adlerborn, B., Dackland, K., Kågström, B.: Parallel and blocked algorithms for reduction of a regular matrix pair to Hessenberg-triangular and generalized Schur forms. In: Fagerholm, J., Haataja, J., Järvinen, J., Lyly, M., Råback, P., Savolainen, V. (eds.) PARA 2002. LNCS, vol. 2367, pp. 319–328. Springer, Heidelberg (2002)
2. Anderson, E., Bai, Z., Bischof, C.H., Blackford, S., Demmel, J.W., Dongarra, J.J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.C.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
3. Bai, Z., Day, D., Demmel, J.W., Dongarra, J.J.: A test matrix collection for non-Hermitian eigenvalue problems (release 1.0). Technical Report CS-97-355, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, (March 1997), Also available online from `http://math.nist.gov/MatrixMarket`

4. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J.W., Dhillon, I., Dongarra, J.J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. SIAM, Philadelphia (1997)
5. Braman, K., Byers, R., Mathias, R.: The multishift $QR$ algorithm. I. Maintaining well-focused shifts and level 3 performance. SIAM J. Matrix Anal. Appl. 23(4), 929–947 (2002)
6. Braman, K., Byers, R., Mathias, R.: The multishift $QR$ algorithm. II. Aggressive early deflation. SIAM J. Matrix Anal. Appl. 23(4), 948–973 (2002)
7. Dackland, K., Kågström, B.: A ScaLAPACK-style algorithm for reducing a regular matrix pair to block Hessenberg-triangular form. In: Kagström, B., Elmroth, E., Waśniewski, J., Dongarra, J.J. (eds.) PARA 1998. LNCS, vol. 1541, pp. 95–103. Springer, Heidelberg (1998)
8. Dackland, K., Kågström, B.: Blocked algorithms and software for reduction of a regular matrix pair to generalized Schur form. ACM Trans. Math. Software 25(4), 425–454 (1999)
9. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
10. Henry, G., Watkins, D.S., Dongarra, J.J.: A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. SIAM J. Sci. Comput. 24(1), 284–311 (2002)
11. Kågström, B., Kressner, D.: Multishift variants of the QZ algorithm with aggressive early deflation. Report UMINF-05.11, Department of Computing Science, Umeå University, Umeå, Sweden, 2005, SIAM J. Matrix Anal. Appl. (to appear)
12. Kågström, B., Poromaa, P.: Computing eigenspaces with specified eigenvalues of a regular matrix pair $(A, B)$ and condition estimation: theory, algorithms and software. Numer. Algorithms 12(3-4), 369–407 (1996)
13. Kaufman, L.: Some thoughts on the $QZ$ algorithm for solving the generalized eigenvalue problem. ACM Trans. Math. Software 3(1), 65–75 (1977)
14. Korvink, J.G., Evgenii, B.R.: Oberwolfach benchmark collection. In: Benner, P., Mehrmann, V., Sorensen, D.C. (eds.) Dimension Reduction of Large-Scale Systems. Lecture Notes in Computational Science and Engineering, vol. 45, pp. 311–316. Springer, Heidelberg (2005)
15. Kressner, D.: Numerical Methods and Software for General and Structured Eigenvalue Problems. PhD thesis, TU Berlin, Institut für Mathematik, Berlin, Germany (2004)
16. Kressner, D.: Block algorithms for reordering standard and generalized Schur forms, 2005, ACM Trans. Math. Software (to appear)
17. Lang, B.: Effiziente Orthogonaltransformationen bei der Eigen- und Singulärwertzerlegung. Habilitationsschrift (1997)
18. Lemonnier, D., Van Dooren, P.: Balancing regular matrix pencils, 2004 SIAM J. Matrix Anal. Appl. (to appear)
19. Moler, C.B., Stewart, G.W.: An algorithm for generalized matrix eigenvalue problems. SIAM J. Numer. Anal. 10, 241–256 (1973)
20. Pester, C.: CoCoS – computation of corner singularities. Preprint SFB393/05-03, Technische Universität Chemnitz (2005), See
http://www.tu-chemnitz.de/sfb393/
21. Quintana-Ortí, G., Quintana-Ortí, E.S.: An efficient algorithm for computing the Hessenberg-triangular form. Technical report ICC 2006-05-01, Universidad Jaime I, Castellón, Spain (2006)

22. Ward, R.C.: The combination shift QZ algorithm. SIAM J. Numer. Anal. 12(6), 835–853 (1975)
23. Ward, R.C.: Balancing the generalized eigenvalue problem. SIAM J. Sci. Statist. Comput. 2(2), 141–152 (1981)
24. Watkins, D.S., Elsner, L.: Theory of decomposition and bulge-chasing algorithms for the generalized eigenvalue problem. SIAM J. Matrix Anal. Appl. 15, 943–967 (1994)