**Introduction**

The generalized eigenvalue problem for a matrix pair \((A, B)\) consists of computing scalars \(\lambda\) s.t.

\[ \det(A - \lambda B) = 0 \]

The standard eigenvalue problem is obtained by explicitly forming \(AB^{-1}\), a step which ought to be avoided as it may be ill-conditioned or even singular (as in many applications from mechanics or electrical engineering).

The QZ algorithm (Moler & Stewart, 1973) is the method of choice for computing all the eigenvalues of \((A, B)\) in a numerically reliable fashion. It consists of reducing \((A, B)\) to quasi-upper triangular form in two steps by orthogonal similarity transformations:

\[ (A, B) \rightarrow (H, T) \rightarrow (S, T) \]

Blocked and parallel algorithms for the first step, reduction to Hessenberg-triangular form, are described in (Doolan & Kågström, 1999). This paper focuses on the second step, applying QZ iterations to bring \(H\) closer to quasi-triangular form while retaining the triangular structure of \(T\). After this step, the eigenvalues can be read off from the block diagonals of \(S\) and \(T\).

To obtain a considerably improved serial and a first parallel, distributed memory implementation of the QZ algorithm, we combine:

- blocked multishift QZ iterations;
- aggressive early deflation;
- ScalAPACK techniques.

**Multishift QZ iterations**

- One doubleshift \((m=2)\) QZ iterations requires \(O(n^2)\) flops.
- On average two such QZ iterations are necessary to deflate one eigenvalue.
- Suffers from high memory access/computation ratio and poor memory reference pattern.

QZ iterations have poor performance on serial machines and are hard to parallelize.

One possibility: increase \(m\), the number of shifts, leading to larger bulges and admitting use of level 2/3 BLAS. \(\triangleright\) Shift blurring in finite-precision arithmetic \(\Rightarrow\) loss of convergence. \(\triangleright\)

Much safer to introduce a tightly coupled chain of tiny bulges:

\[ \text{Only the red areas are updated during the introduction, the blue areas can be updated afterwards using level 3 BLAS matrix-matrix multiplication.} \]

**Advanced deflation strategies**

Classic deflation strategies consider subdiagonal entries in \(H\); the aggressive early deflation strategy (Bunch & Reinsch, 2002; Bunch & Kågström, 2005) takes a small submatrix pair in the bottom right corner, computes its Schur form and investigates the spike for small entries.

**Parallel QZ**

In a DMM environment, matrix blocks are scattered evenly among the participating processors in a 2D processor grid. Message passing is required for communicating shifts and transformations in the QZ iterations.

For example, when updating border elements in the multishift QZ algorithm, the bulges for introduction and bulge chase are redesigned for the parallel case, conceptually illustrated below:

- **Bulge Introduction**
  1. Processor \(p\) holding \((H_{m+1}, T, n_m, n, a, b)\) computes the shifts and sends them to the processor owning the matrix block where the shifts should be introduced.
  2. First left transformation \(Q_0\) is calculated using the shift and data from \(H\) and \(T\) and broadcasted along current processor row(s).
  3. All processors update their portion of \(H\) and \(T\) with respect to \(Q_0 \rightarrow \text{fill-in} \).
  4. Processor owning the data to calculate first right transformation \(Z_{n_m}\), used to eliminate fill-in \(T\), broadcasts \(Z_{n_m}\) along current processor column(s).
  5. All processors update their portion of \(H\) and \(T\) with respect to \(Z_{n_m} \rightarrow \text{fill-in} \). Update is done from left to right.

- **Bulge Chasing**
  - These basic algorithms need some redesign in order to gain high performance due to the fact the only a few processors are active during the introduction and chase phases. Our current implementation makes use of:
    - tightly coupled bulge chasing with a tiny number of shifts per bulge;
    - accumulation of transformations in the bulge chasing window (resulting in matrix-matrix multiply update operations performed in parallel);
    - aggressive early deflation.

**Support**

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

Financial support has been provided by the Swedish Research Council under grant VR-621-2001-3284, by the Swedish Foundation for Strategic Research under grant A3-G128, and by the DFG Emmy Noether fellowship KR 2950-1-1.

---

**References**


---

**Conclusions & future work**

The new multishift QZ algorithm with aggressive early deflation significantly outperforms the current LAPACK implementation. On a wide range of large matrices from the Matrix Market and other benchmark collections, the average execution time was reduced to 14% (see Kågström & Kressner, 2005). A production code will soon be available and is planned to be part of the next release of LAPACK.

The parallel QZ algorithm is work in progress. Preliminary numerical experiments indicate promising results but further tuning and testing is necessary to attain optimal performance.

---

**Acknowledgments**

The authors thank the anonymous referees for useful comments and suggestions.