PDLAQR1: An improved version of the ScaLAPACK routine PDLAHQR

By

Meiyue Shao

UMINF-11/22

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE
SE-901 87 UMEÅ
Sweden

`PDLAQR1`: An improved version of the ScaLAPACK routine `PDLAHQR`[*]

Meiyue Shao
Department of Computing Science and HPC2N
Umeå University
myshao@cs.umu.se

April 27, 2012

Abstract

`PDLAQR1` is a modified version of ScaLAPACK (version 1.8.0) routine `PDLAHQR`. In this note we summarize the difference between these routines. Some anomalies in the old routine are identified and fixed. We also implement some new features such as aggressive early deflation to improve the performance. The new routine is both faster and more reliable than the old one.

1 Introduction

The ScaLAPACK [2] routine `PDLAHQR` implements a parallel pipelined QR algorithm [7]. To our knowledge it is the only publicly available parallel software of the QR algorithm on distributed memory systems. Recently a novel parallel QR algorithm [5] has been developed, which is more than a magnitude faster compared to `PDLAHQR` for sufficiently large problems. In the routine `PDHSEQR` for the novel parallel QR algorithm, some eigensolvers for relatively small problems, such as LAPACK's [1] `DLAHQR`/`DLAQR4` and ScaLAPACK's `PDLAHQR`, are called in the aggressive early deflation (AED) [3, 5, 8] stage. Therefore we have decided to refine the ScaLAPACK routine `PDLAHQR` to make it more efficient and reliable. We present all major updates we have made to `PDLAHQR` in this report. These updates include some new features and bug fixes. Some algorithmic discussions for AED can been found in [8]. This note is a complement to [8] which focuses on further developments. The modified version of `PDLAHQR`, named `PDLAQR1`, is used in the software of the novel parallel eigensolver `PDHSEQR`[1].

[*]Technical Report UMINF-11.22, revised in April 2012.

[1]The software of the novel parallel QR algorithm is now available as a part of ScaLAPACK version 2.0.

2 New Features

2.1 Aggressive early deflation

Aggressive early deflation is an efficient deflation strategy for the QR algorithm and its variants. We have developed a new routine `PDLAQR2` to handle AED in `PDLAQR1`. Firstly the AED window is gathered to local memory on the processor who owns the first entry of the AED window. Then we call the LAPACK routine `DLAQR3` to deal with the local AED process. The output of `DLAQR3` is broadcasted to all other processors. The implementation is slightly different from the preliminary version presented in [8] so that the new implementation guarantees data consistency over processors even on a heterogeneous architecture. Finally, the corresponding off-diagonal blocks are updated by `DGEMM`/`PDGEMM`. After returning from `PDLAQR2`, we compute eigenvalues of a trailing submatrix of the same size as the AED window to obtain shifts for the next QR sweep. The size of work space in `PDLAQR1` is about 6MB larger compared to that in the original `PDLAHQR` so that it is large enough to hold all working copies of matrices (also for the purpose of conventional deflation, see Section 2.2). Further details about the implementation is available in [8].

2.2 Conventional deflation

Besides the implementation of AED, we have also enhanced the conventional deflation part in `PDLAHQR`. In the old routine, only 1×1 or 2×2 diagonal blocks can be deflated directly. Hence a pipelined QR sweep is needed even if the active block is as small as 3×3. Since `PDLAHQR` is usually much slower than LAPACK's `DLAHQR`/`DLAQR4` for small matrices, it is sensible to handle a larger active block locally. Also the corresponding off-diagonal blocks can be updated in a blocked manner.

The Implementation of this functionality is very similar to that for AED. This work is performed in a new routine `PDLAQR4`, which is almost identical to `PDLAQR2` except for the local calculation. Once a small active block ($\texttt{NH} \leq 385$) is detected, we copy it to local memory and call LAPACK's `DLAHQR`/`DLAQR4` to solve the small eigenvalue problem. Then the corresponding off-diagonal blocks are updated by `DGEMM`/`PDGEMM`. Now for matrices of size up to 385×385, `PDLAQR1` is almost as fast as `DLAHQR`/`DLAQR4`. For larger matrices, this strategy at least saves a lot of work during the final stages of the pipelined QR algorithm. Sometimes global convergence also helps to deflate some diagonal blocks. As we have reported in [8], the total execution time can be reduced by a non-negligible amount by adopting this strategy.

2.3 Restriction on 2×2 diagonal blocks

The quasi-upper triangular matrix computed by `PDLAHQR` is not in standard real Schur form; some 2×2 diagonal blocks can contain real eigenvalues. As we need to use `PDLAQR1` as an eigensolver inside the AED phase [5, 8], it is helpful to impose all 2×2 blocks to only have complex conjugate eigenvalues. Thus we need to modify the code snippet in the deflation stage. Once a 2×2 block is deflated, we apply the LAPACK routine `DLAQR1` to resolve this block into standard real Schur form. In this case a 2×2 block which contains two real eigenvalues is split into two 1×1 blocks. Then a PBLAS style routine `PDROT` is called to update the corresponding off-diagonal parts. For the newly implemented branch which handles a large deflated block, the quasi-upper triangular block is always in standard real Schur form, since `DLAHQR`/`DLAQR4`, which is called by `PDLAQR4`, always does the right job.

3 Fixed Anomalies

3.1 Wrong eigenvalues in `WR` and `WI`

In certain cases `PDLAHQR` may fail to read out the converged eigenvalues correctly. We have rewritten the code snippet for extracting eigenvalues from a 2×2 diagonal block in a simpler manner. For larger deflated windows, `DLAHQR`/`DLAQR4` always returns correct eigenvalues (see Section 2.2). Now the eigenvalues in `WR` and `WI` are correctly read out. Likely the bug has been removed by our updates.

3.2 Improper usage of `PDLACONSB`

Suppose we have found two consecutive small subdiagonal entries $h_{i,i-1}=\alpha$ and $h_{i+1,i}=\beta$ satisfying $\alpha\beta=O(\epsilon)$, where ϵ is the machine precision. It is possible to introduce the bulge from the i-th row instead of the top-left corner of H, i.e.,

$$H=\begin{bmatrix}\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times\\ &\alpha&\color{red}\times&\color{red}\times&\color{red}\times&\times&\times\\ &&\color{red}\beta&\color{red}\times&\color{red}\times&\times&\times\\ &&&\color{red}\times&\color{red}\times&\times&\times\\ &&&&\times&\times&\times\\ &&&&&\times&\times\end{bmatrix},$$

$$U^TH=\begin{bmatrix}\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times\\ &\times&\color{red}\times&\color{red}\times&\color{red}\times&\times&\times\\ &f&\color{red}\times&\color{red}\times&\color{red}\times&\times&\times\\ &f&\color{red}+&\color{red}\times&\color{red}\times&\times&\times\\ &&&&\times&\times&\times\\ &&&&&\times&\times\end{bmatrix},$$

$$U^T H U = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ & \times & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \times & \times \\ & f & {\color{red}\times} & {\color{red}\times} & {\color{red}\times} & \times & \times \\ & f & {\color{red}+} & {\color{red}\times} & {\color{red}\times} & \times & \times \\ & & + & + & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}.$$

The i-th column of U is parallel to

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ (h_{i,i} - \sigma_1 - \sigma_2)h_{i,i} + \sigma_1\sigma_2 + h_{i,i+1}\beta \\ (h_{i,i} + h_{i+1,i+1} - \sigma_1 - \sigma_2)\beta \\ h_{i+2,i+1}\beta \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad (1)$$

where σ_1 and σ_2 are the shifts. Very hopefully the 3×3 Householder vector has the form

$$\omega = \begin{bmatrix} \Theta(1) \\ \Theta(\beta) \\ O(\beta) \end{bmatrix}.$$

If so the two fill-ins (marked as f in the matrices above) are of size $O(\alpha\beta)$ and hence can be safely dropped. This technique is called *aggressive deflation* [9] but it is not essentially a direct deflation. It was firstly proposed by Francis in [4] for the purpose of saving computations. Another reason for using this technique was mentioned in [7], namely that consecutive small subdiagonal elements may cause some bulges deliver inaccurate information of the shifts.

`PDLACONSB` is the routine which looks for two consecutive small subdiagonal entries and tests whether a bulge can be introduced at such a place. If there are no suitable places in the middle of the active block, it returns the top-left corner as the start position for introducing the bulge. In `PDLAHQR`, `PDLACONSB` is called only once within a pipelined QR step, that is, only the first pair of shifts is tested but all other shifts are introduced at the same place without testing. (This does not exactly match the discussion in [7].) The "correctness" of this strategy is based on the following fact. Suppose $U = I - 2\omega\omega^T$ where

$$\omega = \begin{bmatrix} \Theta(1) \\ \Theta(\beta) \\ O(\beta) \\ 0 \end{bmatrix} \qquad (2)$$

and a bulge is introduced like

$$U^T \begin{bmatrix} \Theta(1) & \Theta(1) & \Theta(1) & \Theta(1) \\ \Theta(\beta) & \Theta(1) & \Theta(1) & \Theta(1) \\ & \Theta(1) & \Theta(1) & \Theta(1) \\ & & \Theta(1) & \Theta(1) \end{bmatrix} U = \begin{bmatrix} O(1) & O(1) & O(1) & O(1) \\ O(\beta) & \Theta(1) & \Theta(1) & \Theta(1) \\ O(\beta) & \Theta(1) & \Theta(1) & \Theta(1) \\ O(\beta) & O(\beta^2) & \Theta(1) & \Theta(1) \end{bmatrix},$$

the "two-consecutive-small-subdiagonal" structure can be preserved after the bulge is introduced and chased down. However this decision is too aggressive because the nonzero entries may not always be distributed in the same magnitude as we expected above. Most bulges are introduced without really checking the magnitude of fill-ins. The strategy is safe only for the first pair of shifts. When introducing another bulge from the same position, the Householder vector ω may not still keep the same structure as shown in (2) if some $O(1)$ entries in H become too small or the $O(\beta)$ entry becomes large during the pipelined QR sweep. It is also possible that some shifts can destroy the structure (2) of ω according to (1). Moreover, even if the "two-consecutive-small-subdiagonal" structure is always well preserved, the magnitude of dropped fill-ins can be several times larger than ϵ. Then the accuracy becomes a bit worse than what we normally expect. Thus it is numerically unreliable if the fill-ins are dropped without further checking. We have observed some inaccurate outputs caused by this strategy. For matrices of the class `hessrand` (random upper Hessenberg matrices whose nonzero entries are uniformly distributed in $[-1, 1]$), the relative residuals $\|Q^T AQ - T\|_F / \|A\|_F$ are typically around 10^{-11}; sometimes they can be as large as 10^{-5}. For the benchmark matrix AF23560, `PDLAHQR` also returns large residuals [5].

There are several possible ways to fix this problem, for example
(1) Skip `PDLACONSB` and always introduce bulges from the top-left corner of the active block;
(2) For every pair of shifts, call `PDLACONSB` to find a start position;
(3) Call `PDLACONSB` once but skip bulges which can not be safely introduced [7];
(4) Modify `PDLACONSB` so that it is able to find a common start position suitable for all bulges.
The first approach is the easiest but no work in the bulge-chasing phase can be saved. The second approach may cause different bulges have different start positions, which makes it complicated to chase bulges in parallel. The third approach contradicts a little to the original idea of introducing bulges from the middle—transmit correct shifts to the bottom; information of some shifts is lost. The fourth approach also seems complicated because it is as expensive as generating the first column of the shift polynomial $\prod_{k=1}^{\mathtt{NS}}(H - \sigma_k I)$. According to our experimental results, `PDLACONSB` most probably returns the top-left corner as the start position. For the cases when `PDLACONSB` does return a position in the middle of the active block, only less than 0.3% of the starting positions are far away from the top-left corner (with distance ≥ 50). Notice that `PDLACONSB` is not a cheap routine since it needs to perform some communications. So we usually save very little work in average by calling `PDLACONSB`. If a more conservative criterion (e.g., the second approach) is used, we expect more cost and

less benefit. On the other hand, tiny small subdiagonal entries rarely prevent implicit QR algorithm from convergence [10]. Even if forward instability really occurs, AED can still help deflating other eigenvalues which are transmitted accurately. Thus we decided to adopt the first approach, i.e., simply skipping `PDLACONSB`. With this modification, the relative residuals for the problematic matrices have been decreased to the scale $10^{-13} \sim 10^{-14}$.

4 Some Other Remarks

`PDLAQR1` inherits most restrictions in `PDLAHQR` (e.g., `NB` ≥ 6, `DESCA` $=$ `DESCZ`) except for that 2×2 diagonal blocks are now in standard Schur form. We tried to remove the restriction that $A(1,1)$ needs to lie on processor $(0,0)$ by modifying some calls to `NUMROC` and `INFORG1L` in `PDLAQR1`, `PDLACP3` and `PDLASMSUB`. But the restriction has not been totally removed yet. We tolerate this problem because usually we need to copy the submatrix to work space before it is passed to `PDLAQR1` (see [6] for details). As for the maximum number of shifts, we do not plan to increase the limit since `PDLAQR1` is only designed to solve medium-size problems. For large-scale problems, a more efficient solver `PDHSEQR` is preferred. As another modification, we add the recently removed workspace query functionality back to make this routine have the same calling convention with other ScaLAPACK routines[2].

In `PDLAHQR`/`PDLAQR1`, a lot of computations are performed by `DLAREF` and `DGEMM`. These computational kernels can be parallelized with multiple threads. This extension allows the code to work on modern hybrid distributed memory systems.

References

[1] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK User's Guide, 3rd Edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[2] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. W. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK User's Guide.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[3] K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. part II: Aggressive early deflation. *SIAM Journal on Matrix Analysis and Applications*, 23(4):948–973, 2002.

[2]In ScaLAPACK version 2.0.1, `PDLAHQR` is not replaced by `PDLAQR1` for downward compatibility since the latter one requires more work space. Patches are applied to `PDLAHQR` to fix the anomalies mentioned in Section 3.

[4] J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation — Part 2. *the Computer Journal*, 4(4):332–345, 1962.

[5] R. Granat, B. Kågström, and D. Kressner. A novel parallel QR algorithm for hybrid distributed memory HPC systems. *SIAM Journal on Scientific Computing*, 32(4):2345–2378, 2010.

[6] R. Granat, B. Kågström, D. Kressner, and M. Shao. Parallel library software for the multishift QR algorithm with aggressive early deflation. Technical report, Department of Computing Science and HPC2N, (in preparation).

[7] G. Henry, D. S. Watkins, and J. J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. *SIAM Journal on Scientific Computing*, 24(1):284–311, 2002.

[8] Bo Kågström, Daniel Kressner, and Meiyue Shao. On aggressive early deflation in parallel variants of the QR algorithm. In Kristján Jónasson, editor, *Applied Parallel and Scientific Computing (PARA 2010)*, volume 7133 of *Lecture Notes in Computer Science*, pages 1–10, Berlin Heidelberg, 2012. Springer-Verlag.

[9] G. W. Stewart. *Matrix Algorithms. Volume II: Eigensystems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[10] D. S. Watkins. Forward stability and transmission of shifts in the QR algorithm. *SIAM Journal on Matrix Analysis and Applications*, 16(2):469–487, 1995.