H2020-FETHPC-2014: GA 671633

NLAFET Working Note 2

PDHGEQZ User Guide

Björn Adlerborn, Bo Kågström, and Daniel Kressner

May, 2016

Document information

This preprint report is also published as Report UMINF 15.12 at the Department of Computing Science, Umeå University, Sweden.

Acknowledgements

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under the grant agreement number 671633.

PDHGEQZ User Guide*

Björn Adlerborn† Bo Kågström† Daniel Kressner‡

Abstract

Given a general matrix pair (A, B) with real entries, we provide software routines for computing a generalized Schur decomposition (S, T). The real and complex conjugate pairs of eigenvalues appear as 1×1 and 2×2 blocks, respectively, along the diagonals of (S, T) and can be reordered in any order. Typically, this functionality is used to compute orthogonal bases for a pair of deflating subspaces corresponding to a selected set of eigenvalues. The routines are written in Fortran 90 and targets distributed memory machines.

1 Introduction

`PDHGEQZ` is a parallel ScaLAPACK-style [6] package of routines for solving nonsymmetric real generalized eigenvalue problems. The package is written in Fortran 90 and targets distributed memory HPC systems. Using the, small and tightly coupled bulge, multishift QZ algorithm with aggressive early deflation, it computes the generalized Schur decomposition $(S, T_2) = (Q^T H Z, Q^T T_1 Z)$ of an upper Hessenberg matrix H, upper triangular matrix T_1, where $(H, T_1) \in \mathbb{R}^{n \times n}$, such that Q and Z are orthogonal, S is quasi-upper triangular, and T_2 is upper triangular. This document concerns the usage of `PDHGEQZ` and is a supplement to the article [3]. For the description of the algorithm and implementation, we refer to [3] and the references therein (especially, [2, 7, 11, 12, 13]). We have also included a parallel version of the initial Hessenberg-triangular reduction [1], as well as a routine for parallel reordering of eigenvalues of a matrix pair in generalized real Schur form [9]. Moreover, a preliminary version of a serial multishift QZ algorithm with aggressive early deflation is included, see [11], which is used internally within `PDHGEQZ` in favour of the LAPACK [4] routine `DHGEQZ`.

2 Installation

The routines have successfully been tested on both Windows and Linux-like machines. In the following, a Linux-like installation is assumed. There are no pre-built libraries available, so the user must download and build the package before using the software.

*NLAFET Working Note 2. *Report UMINF* 15.12, Dept. of Computing Science, Umeå University, SE-901 87 Umeå, Sweden.

This research was conducted using the resources of the High Performance Computing Center North (HPC2N). Partial support has been received from the European Unions Horizon 2020 research and innovation programme under the NLAFET grant agreement No 671633, the Swedish Research Council (VR) under grant A0581501, and by eSSENCE, a strategic collaborative e-Science programme funded by the Swedish Government via VR.

†Department of Computing Science and HPC2N, Umeå University, SE-90187 Umeå, Sweden (`adler@cs.umu.se, bokg@cs.umu.se`)

‡SB–MATHICSE–ANCHP, EPF Lausanne, Station 8, CH-1015 Lausanne, Switzerland (`daniel.kressner@epfl.ch`)

2.1 Prerequisites

To build the library the following is required:

- Fortran 90/95 compiler, or later.
- MPI library, for example, OpenMPI or Intel-MPI.
- BLAS library, for example, OpenBLAS or ACML.
- The LAPACK library, version 3.4.0 or later.
- The ScaLAPACK library, version 2.0.1 or later where BLACS and PBLAS are included.

2.2 Building the package

2.2.1 Download location

The latest version of `PDHGEQZ` can be obtained via the package homepage[1] in a `tar.gz` format with the name `pdhgeqz_latest`.

2.2.2 Structure of the package

After downloading and moved to a proper location, the package is expanded by issuing

```
tar xvfz pdhgeqz_latest.tar.gz
```

The unpacked package is structured in the following way:

- `examples/` This folder contains three different simple drivers:
  - `EXRAND1.f`: Generates a Hessenberg-Triangular problem, *Hessrand1* [3], and computes its generalized real Schur form.
  - `EXRAND2.f`: Generates a random matrix pair (A, B) where the entries are uniformly distributed in the interval $[0, 1]$. The problem is reduced to Hessenberg-triangular form using a QR factorisation of B, producing an upper triangular T, followed by calling `PDGGHRD` which further reduces the (A, T) pair to the desired Hessenberg-triangular form. Once there, we apply our parallel QZ algorithm to compute the generalized real Schur form and eigenvalues.
  - `EXRAND3.f`: This example reads two matrices A and B from two files using the Matrix Market file format [5]. The matrices A and B are stored in the files `mhd4800a.mtx` and `mhd4800b.mtx` respectively, and can be downloaded, with other benchmarks, at the Matrix Market homepage[2]. A and B are treated as dense, although they are sparse ($< 1\%$ nonzero entries), and first reduced to Hessenberg-triangular form before the parallel QZ algorithm is applied, as described for the example `EXRAND2.f`. After the generalized real Schur form is obtained, a reordering of the eigenvalues is performed such that all eigenvalues within the unit circle are moved to the top of the matrix pair, using the routine `PDTGORD`.

[1] PDHGEQZ homepage: https://archive.cs.umu.se/software/pdhgeqz/
[2] http://math.nist.gov/MatrixMarket

- `make_inc/` This folder contains various templates of `make.inc` files for different compiler setups.
- `Makefile` This is the default Makefile for building the `PDHGEQZ` package, which generally does not need to be modified. This Makefile will call the Makefiles stored in subdirectories. After a successful build, the library will be stored in the root folder as the file `libpdhgeqz.a`.
- `make.inc` File included by the Makefile. Should be modified to match the compiler setup.
- `readme` A shorter version of this Users' Guide. Intended as a quick installation quide.
- `src/` This folder contains all source files, in Fortran format, related to the `PDHGEQZ` software, stored in four different subdirectories:
  - `ab2ht/` Source files for the parallel Hessenberg-triangular reduction.
  - `kkqz/` Source files for the sequential QZ solver with multishift and aggressive early deflation.
  - `reorder/` Source files for the parallel generalized eigenvalue reordering.
  - `pdhgehz/` Main source files for the parallel QZ solver with multishifts and aggressive early deflation.
- `testing/` This folder contains test routines.
- `tools/` This folder contains various auxiliary routines, used for generation of matrices, input/output, etc.
- `userguide.pdf` A copy of this User Guide.

2.2.3 Compiling and building the `PDHGEQZ` package

By issuing the command

`make all` - this is the same as issuing `make`,

the library `libpdhgeqz.a` should be compiled, linked, and stored in the root folder. The three examples found in the `examples/` folder as well as the test routine stored in the `testing/` folder will also be built by this command. If only the library is desired, instead issue the command

```
make lib
```

which will compile, link and store the library in the root folder.

During `make all` some quick tests will be performed and hopefully the following will be displayed on the console:

```
% 5 tests out of 5 passed.
```

The script file `runmpi.sh` in the `testing/` folder might need some attention, depending upon what MPI installation the user has. The default MPI execution is `mpirun`.

2.3 Running tests

The three examples provided contain small problems, i.e. $n < 6000$, and should execute rather quickly. The problem size is governed by the parameter `N`. Change this, in file `EXRAND1.f` or `EXRAND2.f`, and rebuild, to run a different sized problem. The example problems can be executed serially, or in parallel with some MPI tool, for example `mpirun` or `mpiexec`.

3 Using the PDHGEQZ library

3.1 ScaLAPACK data layout convention

We follow the convention of ScaLAPACK for the distributed data storage of matrices. Suppose that $P = P_r \cdot P_c$ parallel processors are arranged in a $P_r \times P_c$ rectangular grid. The entries of a matrix are then distributed over the grid using a 2-dimensional block-cyclic mapping with block size n_b in both row and column dimensions. In principle, ScaLAPACK allows for different block sizes for the row and column dimensions but to avoid too many special cases in the code, square blocks are assumed, i.e. $m_b = n_b$.

3.2 Software hierarchy

Figure 1 shows an overview of the main routines related to our parallel QZ software, which are based upon routines provided by ScaLAPACK, BLACS, PBLAS, LAPACK, BLAS and MPI. One-directed arrows indicate that one routine calls other routines. For example, `PDHGEQZ1` is called by `PDHGEQZ`, `PDHGEQZ0` and `PDHGEQZ3` and calls four routines: `PDHGEQZ2`, `PDHGEQZ4`, `PDHGEQZ5`, and `PDHGEQZ7`. `PDHGEQZ3` will call `PDHGEQZ0` instead of `PDHGEQZ1` recursively if the AED window is large, i.e. $n_{\mathrm{AED}} > 6000$, indicated with the double directed arrow between `PDHGEQZ0` and `PDHGEQZ3`. To keep the stack from growing uncontrollably, due to static allocations, the level of recursive calls are limited; this version only supports one level of recursion. Main entry routine is `PDHGEQZ`, see Figure 2 for an interface description. Depending upon problem size n, `PDHGEQZ` calls `PDHGEQZ0` or `PDHGEQZ1` to perform the reduction; see section 4 and parameter n_{min1}. Table 1 gives a brief description of the routines displayed in Figure 1.

3.3 Calling sequence

The main purpose is to compute the generalised real Schur decomposition of a matrix pair in (upper)Hessenberg-triangular form using the routine `PDHGEQZ`. As a pre-processing step the general square matrix pair must be transformed to Hessenberg-triangular form using the supplied routine `PDGGHRD`. `PDGGHRD` used in conjunction with `PDHGEQZ` are exemplified in the routines found in the folder `examples/`. As a post-processing step, the supplied routine `PDTGORD` can be used to reorder the eigenvalues returned by `PDHGEQZ`. Below follows call sequences for these routines. As previously stated, both `PDGGHRD` and `PDTGORD` are preliminary.

3.3.1 PDHGEQZ

The interface for `PDHGEQZ` is similar to the existing (serial) LAPACK routine `DHGEQZ`, see Figure 2. The main difference between `PDHGEQZ` and `DHGEQZ` is the use of descriptors to define

[3] See Table 2 for `PDHGEQZ` parameters.

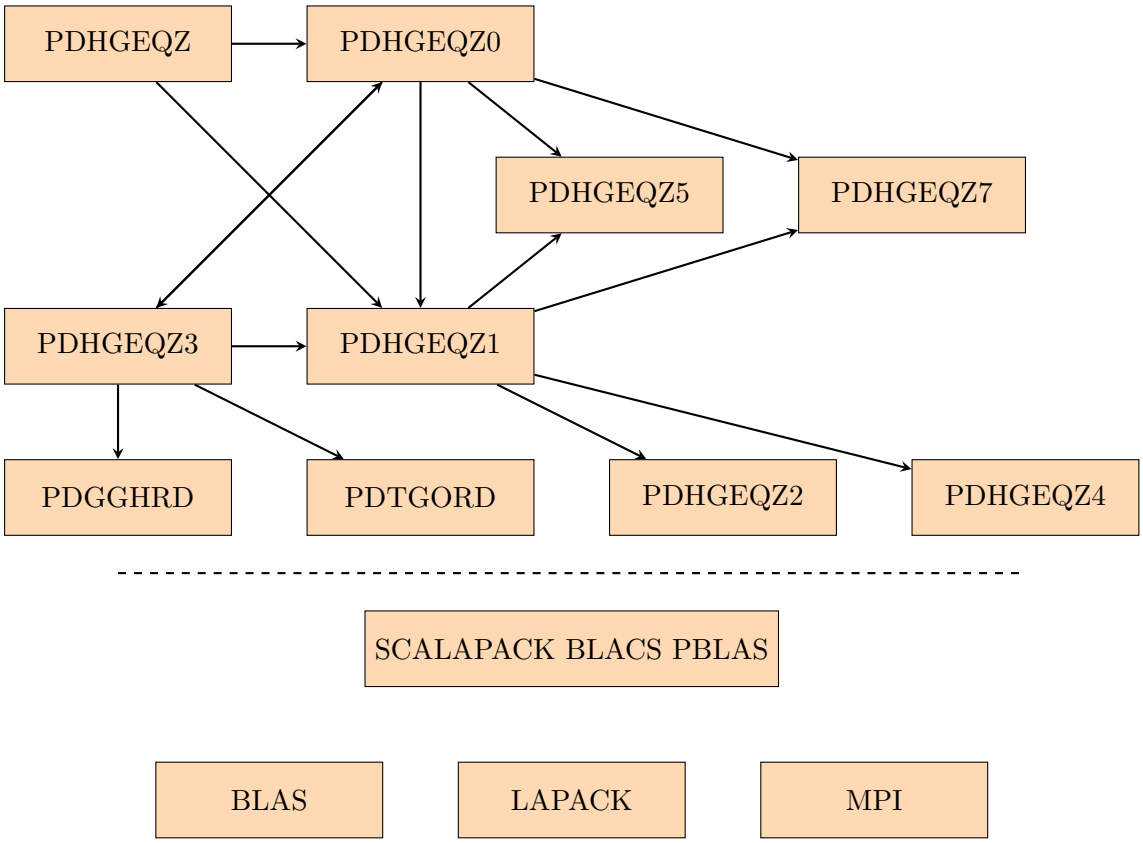

Figure 1: Software hierarchy for `PDGEHQZ`.

partitioning and the globally distributed matrices across the $P_r \times P_c$ process grid, instead of leading dimensions, and that `PDHGEQZ` requires an integer workspace.

Below follows a list and description of the arguments:

- `JOB` (global input) `CHARACTER*1`.
  `= 'E'`: compute only the eigenvalues represented by `ALPHAR`, `ALPHAI`, and `BETA`. The generalized Schur form of (H,T) will not be computed.
  `= 'S'`: compute the generalized Schur form of (H,T) as well as the eigenvalues represented by `ALPHAR`, `ALPHAI`, and `BETA`.

- `COMPQ` (global input) `CHARACTER*1`.
  `= 'N'`: Left generalized Schur vectors (i.e., Q) are not computed.
  `= 'I'`: Q is initialized to the unit matrix, and the orthogonal matrix Q is returned.
  `= 'V'`: Q must contain an orthogonal matrix $Q1$ on entry, and the product $Q1 \cdot Q$ is returned.

- `COMPZ` (global input) `CHARACTER*1`.
  `= 'N'`: Right generalized Schur vectors (i.e., Z) are not computed.
  `= 'I'`: Z is initialized to the unit matrix, and the orthogonal matrix Z is returned.
  `= 'V'`: Z must contain an orthogonal matrix $Z1$ on entry, and the product $Z1 \cdot Z$ is returned.

Table 1: PDHGEQZ routines.

Routine	Description
PDHGEQZ	Main entry routine. Calls PDHGEQZ0 or PDHGEQZ1 depending upon problem size.
PDHGEQZ0	Multishift Parallel QZ with AED. Tuned for larger problems and might do recursive calls when doing AED. Calls PDHGEQZ1 when problem size is split into smaller subproblems.
PDHGEQZ1	Multishift Parallel QZ with sequentially performed AED. Tuned for smaller problems.
PDHGEQZ2	Performs sequential AED, while off-diagonal blocks, outside the AED window, are updated in parallel.
PDHGEQZ3	Performs parallel AED and shift computation.
PDHGEQZ4	Performs sequential multishift QZ with AED, calls KKQZ for this purpose. Called by PDHGEQZ1 when a subproblem, of order less than $n_{\min 3}{}^3$, has been identified. Off-diagonal blocks, outside the subproblem, are updated in parallel.
PDHGEQZ5	Parallel multishift QZ iteration based on chains of tightly coupled bulges.
PDHGEQZ7	Parallel identification and deflation of infinite eigenvalues.
PDTGORD	Reorders a cluster of eigenvalues to the top of the matrix pair (S, T) in generalized real Schur form.
PDGGHRD	Parallel Hessenberg-triangular reduction, used by PDHGEQZ3 to restore the matrix pair to Hessenberg-triangular form.

- N (global input) INTEGER
  The order of the $N \times N$ matrices H, T, Q, and Z.

- ILO, IHI
  It is assumed that H and T is already in upper triangular form in rows and columns $(1 : \texttt{ILO}-1)$ and $(\texttt{IHI}+1 : \texttt{N})$. $1 \leq \texttt{ILO} \leq \texttt{IHI} \leq \texttt{N}$, if $\texttt{N} > 0$; $\texttt{ILO} = 1$ and $\texttt{IHI} = 0$, if $\texttt{N} = 0$.

- H (global input/output) DOUBLE PRECISION array of dimension (DESCH(9),*).
  DESCH (global and local input) INTEGER array of dimension 9.
  H and DESCH define the distributed matrix H.
  On entry, H contains the upper Hessenberg matrix H. On exit, if JOB = 'S', H is quasi-upper triangular in rows and columns (ILO : IHI), with 1×1 and 2×2 blocks on the diagonal where the 2×2 blocks correspond to complex conjugated pairs of eigenvalues. If JOB='E', H is unspecified on exit.

- T (global input/output) DOUBLE PRECISION array of dimension (DESCT(9),*).
  DESCT (global and local input) INTEGER array of dimension 9.
  T and DESCT define the distributed matrix T.
  On entry, T contains the upper triangular matrix T. If JOB='E', T is unspecified on exit, otherwise T is on exit overwritten by another upper triangular matrix $Q^T \cdot T \cdot Z$.

- ALPHAR (global output) DOUBLE PRECISION array, dimension N
  ALPHAI (global output) DOUBLE PRECISION array, dimension N
  BETA (global output) DOUBLE PRECISION array, dimension N
  On exit, (ALPHAR(j) + ALPHAI(j)*i)/BETA(j), $j = 1, ..., N$, will be the generalized eigenvalues. ALPHAR(j) + ALPHAI(j)*i and BETA(j), $j = 1, ..., N$ are the diagonals of the complex Schur form (H,T) that would result if the 2-by-2 diagonal blocks of the real generalized Schur form of (H,T) were further reduced to triangular form using complex unitary transformations. If ALPHAI(j) is zero, then the j-th eigenvalue is real; if positive, then the j-th and (j+1)-st eigenvalues are a complex conjugate pair, with ALPHAI(j+1) negative.

```
      SUBROUTINE PDHGEQZ( JOB, COMPQ, COMPZ,
     $       N, ILO, IHI, H, DESCH, T, DESCT,
     $       ALPHAR, ALPHAI, BETA, Q, DESCQ, Z, DESCZ,
     $       WORK, LWORK, IWORK, LIWORK, INFO )

*      ..
*      .. Scalar Arguments ..
*      ..
       CHARACTER          COMPQ, COMPZ, JOB
       INTEGER            IHI, ILO, INFO, N
       INTEGER            LWORK, LIWORK
*      ..
*      .. Array Arguments ..
*      ..
       DOUBLE PRECISION   H(*), T(*), Q(*), Z(*)
       DOUBLE PRECISION   WORK(*), ALPHAI(*), ALPHAR(*), BETA(*)
       INTEGER            IWORK(*), DESCH(9), DESCT(9), DESCQ(9), DESCZ(9)
```

Figure 2: Interface for `PDGEHQZ`

- `Q` (global input/output) `DOUBLE PRECISION` array of dimension `(DESCQ(9),*)`.
  `DESCQ` (global and local input) `INTEGER` array of dimension 9. `Q` and `DESCQ` define the distributed matrix Q.
  If `COMPQ = 'N'`, `Q` is not referenced.
  If `COMPQ = 'I'`, `Q` is initialized to the unit matrix, and on exit it contains the orthogonal matrix Q, where Q^T is the product of the transformations which are applied to the left hand side of `H` and `T`.
  If `COMPQ = 'V'`, on entry, `Q` must contain an orthogonal matrix $Q1$, and on exit this is overwritten by $Q1 \cdot Q$, where Q^T is the product of the transformations which are applied to the left hand side of `H` and `T`.

- `Z` (global input/output) `DOUBLE PRECISION` array of dimension `(DESCZ(9),*)`.
  `DESCZ` (global and local input) `INTEGER` array of dimension 9. `Z` and `DESCZ` define the distributed matrix Z.
  If `COMPZ = 'N'`, `Z` is not referenced.
  If `COMPZ = 'I'`, `Z` is initialized to the unit matrix, and on exit it contains the orthogonal matrix Z, where Z is the product of the transformations which are applied to the right hand side of `H` and `T`.
  If `COMPZ = 'V'`, on entry, `Z` must contain an orthogonal matrix $Z1$, and on exit this is overwritten by $Z1 \cdot Z$, where Z is the product of the transformations which are applied to the right hand side of `H` and `T`.

- `WORK` (local workspace) `DOUBLE PRECISION` array of dimension `LWORK`.
  `LWORK` (global input) `INTEGER`.
  If `LWORK = -1`, then a workspace query is assumed and required workspace is returned in `WORK(1)` and no further computation is performed.

- `IWORK` (local workspace) `INTEGER` array of dimension `LIWORK`.
  `LIWORK` (global input) `INTEGER`.
  If `LIWORK = -1`, then a workspace query is assumed and required workspace is returned

in IWORK(1) and no further computation is performed.

- INFO (global output) INTEGER
  = 0, successful exit.
  < 0, if INFO = -i, the i-th argument had an illegal value.

3.3.2 PDGGHRD

The interface for PDGGHRD is similar to the existing (serial) LAPACK routine DGGHRD, see Figure 3. The main difference between PDGGHRD and DGGHRD is the use of descriptors to define

```
      SUBROUTINE PDGGHRD( COMPQ, COMPZ,
     $      N, ILO, IHI, A, DESCB, B, DESCB,
     $      Q, DESCQ, Z, DESCZ,
     $      WORK, LWORK, INFO )

*     ..
*     .. Scalar Arguments ..
*     ..
      CHARACTER          COMPQ, COMPZ
      INTEGER            IHI, ILO, INFO, N, LWORK
*     ..
*     .. Array Arguments ..
*     ..
      DOUBLE PRECISION   A(*), B(*), Q(*), Z(*)
      DOUBLE PRECISION   WORK(*)
      INTEGER            DESCA(9), DESCB(9), DESCQ(9), DESCZ(9)
```

Figure 3: Interface for PDGGHRD

partitioning and the globally distributed matrices across the $P_r \times P_c$ process grid, instead of leading dimensions.

Below follows a list and description of the arguments:

- COMPQ (global input) CHARACTER*1.
  = 'N': Do not compute Q.
  = 'I': Q is initialized to the unit matrix, and the orthogonal matrix Q is returned.
  = 'V': Q must contain an orthogonal matrix $Q1$ on entry, and the product $Q1 \cdot Q$ is returned.

- COMPZ (global input) CHARACTER*1.
  = 'N': Do not compute Z.
  = 'I': Z is initialized to the unit matrix, and the orthogonal matrix Z is returned.
  = 'V': Z must contain an orthogonal matrix $Z1$ on entry, and the product $Z1 \cdot Z$ is returned.

- N (global input) INTEGER
  The order of the $N \times N$ matrices A, B, Q, and Z.

- ILO, IHI (global input) INTEGER
  It is assumed that A and B is already in upper triangular form in rows and columns $(1 : \texttt{ILO}-1)$ and $(\texttt{IHI}+1 : \texttt{N})$. $1 \leq \texttt{ILO} \leq \texttt{IHI} \leq \texttt{N}$, if N$> 0$; ILO $= 1$ and IHI $= 0$, if N $= 0$.

- A (global input/output) DOUBLE PRECISION array of dimension (DESCA(9),*).
  DESCA (global and local input) INTEGER array of dimension 9.
  A and DESCA define the distributed matrix A.
  On entry, the square general matrix A to be reduced. On exit, the upper triangle and the first subdiagonal of A are overwritten with the upper Hessenberg matrix $H = Q \cdot A \cdot Z$, and the remaining elements are set to zero.

- B (global input/output) DOUBLE PRECISION array of dimension (DESCB(9),*).
  DESCB (global and local input) INTEGER array of dimension 9.
  B and DESCB define the distributed matrix B.
  On entry, the square upper triangular matrix B. On exit, overwritten by the upper triangular matrix $T = Q \cdot B \cdot Z$. The elements below the diagonal are set to zero.

- Q (global input/output) DOUBLE PRECISION array of dimension (DESCQ(9),*).
  DESCQ (global and local input) INTEGER array of dimension 9. Q and DESCQ define the distributed matrix Q^T.
  If COMPQ = 'N', Q is not referenced.
  If COMPQ = 'I', Q is initialized to the unit matrix, and on exit it contains the orthogonal matrix Q^T, where Q is the product of the transformations which are applied to the left hand side of A and B.
  If COMPQ = 'V', on entry, Q must contain an orthogonal matrix $Q1$, and on exit this is overwritten by $Q1 \cdot Q^T$, where Q is the product of the transformations which are applied to the left hand side of A and B.

- Z (global input/output) DOUBLE PRECISION array of dimension (DESCZ(9),*).
  DESCZ (global and local input) INTEGER array of dimension 9. Z and DESCZ define the distributed matrix Z.
  If COMPZ = 'N', Z is not referenced.
  If COMPZ = 'I', Z is initialized to the unit matrix, and on exit it contains the orthogonal matrix Z, where Z is the product of the transformations which are applied to the right hand side of A and B.
  If COMPZ = 'V', on entry, Z must contain an orthogonal matrix $Z1$, and on exit this is overwritten by $Z1 \cdot Z$, where Z is the product of the transformations which are applied to the right hand side of A and B.

- WORK (local workspace) DOUBLE PRECISION array of dimension LWORK.
  LWORK (global input) INTEGER.
  If LWORK = -1, then a workspace query is assumed and required workspace is returned in WORK(1) and no further computation is performed.

- INFO (global output) INTEGER
  = 0, successful exit.
  $<$ 0, if INFO = -i, the i-th argument had an illegal value.

3.3.3 PDTGORD

The interface for PDTGORD is displayed in Figure 4.

Below follows a list and description of the arguments:

```
      SUBROUTINE PDTGORD( WANTQ, WANTZ,
     $       SEL, PARA, N,
     $       S, DESCS, T, DESCT,
     $       Q, DESCQ, Z, DESCZ,
     $       ALPHAR, ALPHAI, BETA,
     $       M, DWORK, LDWORK,
     $       IWORK, LIWORK, INFO )

*       ..
*       .. Scalar Arguments ..
*       ..
        LOGICAL              WANTQ, WANTZ
        INTEGER              INFO, LIWORK, LDWORK, M, N
*       ..
*       .. Array Arguments ..
*       ..
        INTEGER              SEL( * ), IWORK( * )
        INTEGER              PARA( 6 ), DESCS( 9 ), DESCT( 9 ), DESCQ( 9 ), DESCZ( 9 )
        DOUBLE PRECISION     S(*), T(*), Q(*), Z(*)
        DOUBLE PRECISION     DWORK(*), BETA( * ), ALPHAI( * ), ALPHAR( * )
```

Figure 4: Interface for `PDTGORD`

- `WANTQ` (global input) `LOGICAL`.
  `= .TRUE.`, Update $\mathtt{Q}^T$, with all transformations applied from left hand side to `S` and `T`.
  `= .FALSE.`, `Q` is not referenced.

- `WANTZ` (global input) `LOGICAL`.
  `= .TRUE.`, Update `Z`, with all transformations applied from right hand side to `S` and `T`.
  `= .FALSE.`, `Z` is not referenced.

- `SEL` (global input/output) INTEGER array, dimension `N`.
  `SEL` specifies the eigenvalues in the selected cluster. To select a real eigenvalue $w(j)$, `SEL(j)` must be set to 1. To select a complex conjugate pair of eigenvalues $w(j)$ and $w(j+1)$, corresponding to a 2-by-2 diagonal block, must be set to 1; On output, `SEL` is updated to reflect the performed reordering.

- `PARA` (global input) `INTEGER` array of dimension 6
  `PARA(1)` = maximum number of concurrent computational windows allowed in the algorithm. `0 < PARA(1)` $\leq min(P_\mathrm{r}, P_\mathrm{c})$ must hold.
  `PARA(2)` = number of eigenvalues in each computational window. `0 < PARA(2) ≤ PARA(3)` must hold.
  `PARA(3)` = size of computational window. `PARA(2) ≤ PARA(3)` $\leq n_\mathrm{b}$ must hold.
  `PARA(4)` = minimal percentage of flops required for performing matrix-matrix multiplications instead of pipelined orthogonal transformations. `0 ≤ PARA(4) ≤ 100` must hold.
  `PARA(5)` = width of block column slabs for row-wise application of pipelined orthogonal transformations in their factorized form. `0 < PARA(5)` $\leq n_\mathrm{b}$ must hold.
  `PARA(6)` = the maximum number of eigenvalues moved together over a process border; in practice, this will be approximately half of the cross border window size. `0 < PARA(6) ≤ PARA(2)` must hold.

- N (global input) INTEGER.
  The order of the $N \times N$ matrices S, T, Q, and Z.

- S (global input/output) DOUBLE PRECISION array of dimension (DESCS(9),*).
  DESCS (global and local input) INTEGER array of dimension 9.
  S and DESCS define the distributed matrix S. On entry, the global distributed upper quasi-triangular matrix S, in Schur form. On exit, S is overwritten by the reordered matrix S, again in Schur form, with the selected eigenvalues in the globally leading diagonal blocks of (S, T).

- T (global input/output) DOUBLE PRECISION array of dimension (DESCT(9),*).
  DESCT (global and local input) INTEGER array of dimension 9.
  T and DESCT define the distributed matrix T. On entry, the global distributed upper triangular matrix T. On exit, T is overwritten by the reordered matrix T, again in upper triangular form, with the selected eigenvalues in the globally leading diagonal blocks of (S, T).

- Q (global input/output) DOUBLE PRECISION array of dimension (DESCQ(9),*).
  DESCQ (global and local input) INTEGER array of dimension 9.
  Q and DESCQ define the distributed matrix Q. On entry, if WANTQ = .TRUE., the global distributed matrix Q of left generalized Schur vectors. On exit, WANTQ = .TRUE., Q has been postmultiplied by the global orthogonal transformation matrix, applied from the left, which reorders the matrix pair (S, T); the leading M columns of Q form left orthonormal bases for the specified deflating subspaces. If WANTQ = .FALSE., Q is not referenced.

- Z (global input/output) DOUBLE PRECISION array of dimension (DESCZ(9),*).
  DESCZ (global and local input) INTEGER array of dimension 9.
  Z and DESCZ define the distributed matrix Z. On entry, if WANTZ = .TRUE., the global distributed matrix Z of generalized right Schur vectors. On exit, WANTZ = .TRUE., Z has been postmultiplied by the global orthogonal transformation matrix, applied from the right, which reorders (S, T); the leading M columns of Z form right orthonormal bases for the specified deflating subspaces. If WANTZ = .FALSE., Z is not referenced.

- ALPHAR (global output) DOUBLE PRECISION array, dimension N
  ALPHAI (global output) DOUBLE PRECISION array, dimension N
  BETA (global output) DOUBLE PRECISION array, dimension N
  On exit, (ALPHAR(j) + ALPHAI(j)*i)/BETA(j), $j = 1, ..., N$, will be the generalized eigenvalues. ALPHAR(j) + ALPHAI(j)*i and BETA(j), $j = 1, ..., N$ are the diagonals of the complex Schur form (S,T) that would result if the 2-by-2 diagonal blocks of the real generalized Schur form of (S,T) were further reduced to triangular form using complex unitary transformations. If ALPHAI(j) is zero, then the j-th eigenvalue is real; if positive, then the j-th and (j+1)-st eigenvalues are a complex conjugate pair, with ALPHAI(j+1) negative.

- M (global output) INTEGER.
  The dimension of the specified pair of left and right eigenspaces (deflating subspaces). 0 $\leq$ M $\leq$ N. If M = 0, then no eigenvalues have been reordered.

- DWORK (local workspace) DOUBLE PRECISION array of dimension LDWORK.
  LDWORK (global input) INTEGER.
  If LDWORK = -1, then a workspace query is assumed and required workspace is returned in DWORK(1) and no further computation is performed.

- IWORK (local workspace) INTEGER array of dimension LIWORK.
  LIWORK (global input) INTEGER.
  If LIWORK = -1, then a workspace query is assumed and required workspace is returned in IWORK(1) and no further computation is performed.

- INFO (global output) INTEGER.
  = 0, successful exit.
  $<$ 0, if INFO = -i, the i-th argument had an illegal value. If the i-th argument is an array and the j-entry had an illegal value, then INFO = -(i*1000+j).

4 Parameters related to PDHGEQZ and subroutines

PDHGEQZ offers tuning of machine dependent parameters for experienced users. However, there is always a default value provided, and these settings should provide reasonable performance for machines similar to those we have performed our performance-runs on, see [3].

Many of these parameters are dependent upon the value of n_{b}, which the end user chooses a value for when setting up descriptors for the distributed matrices. For a suitable value of n_{b} correlated to the problem size n, see [3] Section 3.2. The parameters are stored in the files piparmq.f and pilanvx.f, found in the folder src/pdhgeqz/. PILAENVX(ISPEC = 50...56) and PILAENVX(ISPEC = 80...85) are parameters related to PDHGEQZ, listed and described in Table 2.

5 Terms of usage

The PDHGEQZ library is freely available for academic (non-commercial) use, and is provided on an "as is" basis. Any use of the PDHGEQZ library should be acknowledged by citing paper [3] and this User Guide.

6 Conclusions and future work

We have presented the high performance software package PDHGEQZ. The latest version of the package along with updated information and documentation will always be available for download from the PDHGEQZ website. We welcome bug-reports, comments and suggestions from users.

Acknowledgments

The authors are grateful to Lars Karlsson, and Meiyue Shao for helpful discussions on parallel QZ algorithms. We thank Åke Sandgren and the rest of the group at the High Performance Computing Center North (HPC2N) for providing computational resources and valuable support during test and performance runs.

Table 2: Tunable parameters for PDHGEQZ

ISPEC	Name	Description	Default value
50	n_{min1}	Threshold for when to choose PDHGEQZ0 instead of PDHGEQZ1; matrices of order less than this value are reduced by PDHGEQZ1.	6000
51	n_{AED}	Size of the aggressive early deflation window.	see [3] Section 3.2
52	NIBBLE	Threshold for when to skip repeated AED and instead do a multishift QZ sweep.	see [10] Section 3.4
53	n_{shift}	The number of simultaneous shifts in a multishift QZ iteration (in PDHGEQZ0).	see [3]§ Section 3.2
54	n_{min2}	When current problem size is less than this value, PDHGEQZ0 calls PDHGEQZ1 to perform the remaining reduction.	201
55	n_{min3}	When current problem size is less than this value PDHGEQZ1 stops executing and instead performs the remaining reduction serially.	201
56	P_{AED}	Number of processes to use when performing parallel AED.	see [3] Section 3.2
80	NUMWIN	Maximum number of concurrent computational windows (for parallel reordering only).	$min(p_{\text{r}}, p_{\text{c}}, n/n_{\text{b}})$
81	WINEIG	Number of eigenvalues/bulges in each window (for parallel reordering only).	$n_{\text{b}}/2$
82	WINSIZE	Computational window size (for parallel reordering only).	n_{b}
83	MMULT	Minimal percentage of flops required for performing matrix-matrix multiplications instead of pipelined orthogonal transformations (for parallel reordering only).	0. Throughout our tests, the matrix-matrix multiplications were faster than or equivalent to pipelined transformations. This is probably caused by the small problem sizes we are running on, i.e. the size of the AED window. For larger n and thereby larger n_{AED}, the value for MMULT might need fine tuning. However, a value of 0 will always work and will probably give good enough performance.
84	NCB	Width of block column slabs for row-wise application of pipelined orthogonal transformations in their factorized form (for parallel reordering only).	min(n_{b}, 32)
85	WNEICR	The maximum number of eigenvalues moved together over a process border (for parallel reordering only).	same as WINEIG

References

[1] B. Adlerborn, L. Karlsson, and B. Kågström. Distributed One-Stage Hessenberg-Triangular Reduction with Wavefront Scheduling. *Report UMINF* 16.10, Dept. of Computing Science, Umeå University, Sweden, 2016.

[2] B. Adlerborn, B. Kågström, and D. Kressner. Parallel variants of the multishift QZ algorithm with advanced deflation techniques. In B. Kågström, E. Elmroth, J. Dongarra, and J. Waśniewski, editors, *Applied Parallel Computing, PARA 2006*, LNCS 4699, pages 117–126. Springer Berlin Heidelberg, 2006.

[3] B. Adlerborn, B. Kågström, and D. Kressner. A Parallel QZ Algorithm for distributed memory HPC-systems. *SIAM J. Sci. Comput.*, 36(5):C480–C503, 2014.

[4] 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.

[5] Z. Bai, D. Day, J. W. Demmel, and J. J. Dongarra. 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`.

[6] 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 Users' Guide*. SIAM, Philadelphia, PA, 1997.

[7] 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.

[8] R. Granat and B. Kågström. Parallel solvers for Sylvester-type matrix equations with applications in condition estimation, Part II. *ACM Trans. Math. Software*, 37(3), 2007.

[9] R. Granat, B. Kågström, and D. Kressner. Parallel eigenvalue reordering in real Schur forms. *Concurrency and Computation: Practice and Experience*, 21(9):1225–1250, 2009.

[10] R. Granat, B. Kågström, D. Kressner, and M. Shao. Parallel library software for the multishift QR algorithm with aggressive early deflation. *ACM Trans. Math. Software*, 41(4), 2015.

[11] B. Kågström and D. Kressner. Multishift variants of the QZ algorithm with aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 29(1):199–227, 2006.

[12] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973.

[13] R. C. Ward. The combination shift QZ algorithm. *SIAM J. Numer. Anal.*, 12(6):835–853, 1975.

A List and description of supplied drivers and auxiliary routines.

Beside the routines briefly described in Table 1, we list the other provided routines in Tables 3–7.

Table 3: Routines not listed in Figure 1 but related to, and called by, the main routines of PDHGEQZ.

Routine	Description
PDHGEQZ8	Called by PDHGEQZ7 to perform parallel chase and deflation of infinite eigenvalues at the top left corner of (H, T).
PDHGEQZ9	Called by PDHGEQZ7 to perform parallel chase and deflation of infinite eigenvalues at the bottom right corner of (H, T).
PDHGEQZA	Called by PDHGEQZ5 to create bulges within a diagonal block of (H, T), followed by updates in parallel of off-diagonal elements.
PDHGEQZB	Called by PDHGEQZ5 to chase bulges within a diagonal block of (H, T), followed by updates in parallel of off-diagonal elements.
PDHGEQZ6	Called by PDHGEQZ2, PDHGEQZ4, PDHGEQZ8, PDHGEQZ9, PDHGEQZA, and PDHGEQZB to update off-diagonal entries in parallel.
PDLACP4	Called by PDHGEQZ2, PDHGEQZ4, PDHGEQZ5, PDHGEQZ8 and PDHGEQZ9 to copy a global diagonal block of (H, T) to a local workspace copy, or vice versa.
PDROT	Performs a planar rotation, in parallel.
DHGEQZ5	Serial chase of bulges within a diagonal block of (H, T)
DHGEQZ7	Serial chase of infinite eigenvalues, along the diagonal of T, up or down.

Table 4: Routines related to parallel reordering. These routines are also part of the SCASY library, see [8] and SCASY homepage http://www8.cs.umu.se/~granat/scasy.html, although slightly modified here

.

Routine	Description
PDTGSEN	Reorders a cluster of eigenvalues to the top of the matrix pair (S, T) in real generalized Schur form. Also provides functionality to compute condition number estimate for eigenvalues and eigenspaces.
BDLAGPP	Computes a transformation matrix Q resulting from performed swaps of diagonal blocks.
BDTGEXC	Moves a diagonal block from one position to another.
BDTGEX2	Swaps two adjacent diagonal blocks.

Table 5: Routines related to the parallel Hessenberg-triangular reduction.

Routine	Description
PDGGHRD	Parallel reduction of a matrix pair to Hessenberg-triangular form - main routine.
UPDATEANDREDUCECOLUMN	Applies previous row updates and reduces a column.
UPDATEANDREDUCECOLUMN_ROOT	Applies previous row updates and reduces a column. This version uses a single core to perform the updates and reduction and is called internally by UPDATEANDREDUCECOLUMN.
KRNLUPDATEANDREDUCECOLUMN	Applies previous row updates and reduces a column - kernel version.
SLIVERROWUPDATE	Applies row updates.
KRNLROWUPDATE	Applies row updates - kernel version.
SLIVERHESSCOLUMNUPDATE	Reduces a matrix in Hessenberg form to triangular form.
KRNLCOLUMNANNIHILATE	Annihilates sub diagonal entries - kernel version.
SLIVERCOLUMNUPDATE	Applies column updates.
KRNLCOLUMNUPDATE	Applies column updates - kernel version.
ACCUMULATEROWROTATIONS	Accumulates row rotations into transformation matrices.
KRNLACCUMULATEROWROTATIONS	Accumulates row rotations into transformation matrices - kernel version.
ACCUMULATECOLUMNROTATIONS	Accumulates column rotations into transformation matrices.
KRNLACCUMULATECOLUMNROTATIONS	Accumulates column rotations into transformation matrices - kernel version.
BLOCKSLIVERROWUPDATE	Applies accumulated transformation on block rows.
BLOCKSLIVERCOLUMNUPDATE	Applies accumulated transformations on block columns.
DUOBLOCKSLIVERCOLUMNUPDATE	Applies accumulated transformations on block columns. This routine takes two matrices as input and updates them at the same time.
GRN2LRN	Computes a local range from a global range.

Table 6: Routines related to the serial multishift QZ, with aggressive early deflation, KKQZ.

Routine	Description
KKQZ	Serial multishift QZ with aggressive early deflation - main routine.
KKQZCONF	Sets up parameters for the KKQZ routine. This routine needs to be called before any call to PDHGEQZ.
INVHSE	Computes an "inverted" Householder reflector.
QZDACK	Serial and blocked single/double shift QZ, based on [7].
QZDACKIT	Performs a blocked single/double QZ iteration, based on [7].
QZEARLY	Performs aggressive early deflation.
QZFCOL	Forms a multiple of the first column of the shift polynomial for the generalized eigenvalue problem.
QZINF	Identifies and deflates infinite eigenvalues.
QZLAP	Serial single/double shift QZ, based on the LAPACK routine DHGEQZ.
QZLAPIT	Performs a single/double QZ iteration, based on the LAPACK routine DHGEQZ.

Table 7: Routines supplied in the tools/ folder.

Routine	Description
PDMATGEN2	Generates a matrix with entries choosen from a uniform distribtution $\in [0, 1]$.
PDLAPRNT	Prints a distributed matrix.
PQZHELPS	Contains routines for calculating residuals, and misc. short helper routines.
MMIO	Contains misc. routines for reading and writing Matrix Market files.
PDRDMMM	Reads a Matrix Market matrix file and stores the content in a globally distributed matrix.