PEPACK: A Software Package for computing the Numerical Solution of palindromic and even eigenvalue problems using the Pencil Laub Trick *

Lisa Poppe†, Christian Schröder†, Ina Thies†

August 28, 2009

Abstract

We describe a software package for computing the numerical solution of palindromic and even eigenvalue problems using the Pencil Laub Trick. The package contains Fortran 77 routines which reduce real and complex palindromic and even pairs of matrices to anti-triangular form. We give explicit descriptions how to use the package including an example.

1 Purpose

Given a matrix pair (A, B) that is either palindromic (i.e., $B = A^\star$) or even (i.e., $A = A^\star$ and $B = -B^\star$) we intend to compute a unitary matrix U such that $U^\star AU$ and $U^\star BU$ are anti-triangular, i.e.,

$$U^\star AU = \begin{smallmatrix} \diagup & \diagup \\ \diagup & \diagup \end{smallmatrix}, \quad U^\star BU = \begin{smallmatrix} \diagup & \diagup \\ \diagup & \diagup \end{smallmatrix}. \tag{1.1}$$

Here A and B are real or complex square matrices and $A^\star$ denotes either the transpose A^T or the conjugate tranpose $A^H = \overline{A}^T$ of A. Thus we distinguish three cases:

i) $A, B, U \in \mathbb{R}^{n\times n}$, $A^\star := A^T$, $U^TU = I$ or equivalently $U^HU = I$;

ii) $A, B, U \in \mathbb{C}^{n\times n}$, $A^\star := A^T$, $U^HU = I$; and

iii) $A, B, U \in \mathbb{C}^{n\times n}$, $A^\star := A^H$, $U^HU = I$.

The report is organized as follows. Section 2 shortly introduces palindromic and even eigenvalue problems. We summarize how a matrix U can be constructed such that U reduces A and B simultaneously to anti-triangular form without destroying the palindromic or even structure of the pair (A, B). In Section 3 we explain some numerical details of the algorithms that we have discussed in Section 2. The algorithms were implemented in Fortran 77. Section 4 provides a documentation of our routines. In the Section 5 we present an example to demonstrate the use of our routines. There is also an Matlab Interface for the routines which is described in Section 3.

2 Mathematical Background

The algorithm is designed for palindromic and even eigenvalue problems that are defined as follows. The *palindromic eigenvalue problem* is given by the equation

$$Ax = \lambda A^\star x, \tag{2.1}$$

*Partially supported by *Deutsche Forschungsgemeinschaft* through project ME 790/16-1

†Institut für Mathematik, MA 4-5, Technische Universität Berlin, Str. des 17. Juni 136, 10623 Berlin, Germany. {poppe,schroed,thies}@math.tu-berlin.de

where A is a square matrix.
The *even eigenvalue problem* is of the form

$$Ax = \lambda Bx, \text{ with } A = A^\star,\ B = -B^\star, \tag{2.2}$$

where A is a square symmetric or Hermitian matrix and B is a square skew-symmetric or skew-Hermitian matrix, respectively.
As we have mentioned in Section 1, (2.1) and (2.2) each represent three different cases of eigenvalue problems, depending on whether $A^\star$ denotes A^T or A^H. These cases have similar, though not identical properties. Where possible we will treat them in a unified way.
The structure in the coefficient matrices of (2.1) and (2.2) results in a structure in the spectrum which we will explain in the next two subsections for both the palindromic and the even case. We also introduce the algorithms that lead to the anti-triangular form.
The name "Laub trick" goes back to an analogous method for Hamiltonian matrices introduced by Laub in [1].

2.1 Palindromic pairs

(Conjugate-) transposing the palindromic problem (2.1) yields $x^\star A = \frac{1}{\lambda^\star} x^\star A^\star$. So, if λ is an eigenvalue and x an associated eigenvector, then $\frac{1}{\lambda^\star}$ is also an eigenvalue with $x^\star$ as left eigenvector where $\lambda^\star$ is either $\bar{\lambda}$ (if $\star = H$) or just λ itself (otherwise). For a zero eigenvalue the corresponding eigenvalue will be an infinite eigenvalue.
The eigenvectors corresponding to λ and $\frac{1}{\lambda^\star}$ are guaranteed to be linearly independent, provided that the pairing is nontrivial, i.e., that $\lambda \neq \frac{1}{\lambda^\star}$. This condition is violated for the so-called $\star$-exceptional eigenvalues satisfying $\lambda^\star\lambda = 1$ which is the case for ± 1 (if $\star = T$) or every value on the unit circle (if $\star = H$). A major difference between the two complex palindromic problems is that there are only finitely many T-exceptional eigenvalues, while there is a whole continuum of H-exceptional eigenvalues. The exceptional eigenvalues of real palindromic problems are also given by the whole unit circle, because it cannot be guaranteed that the deflating subspaces for the pairs $(\lambda, \bar{\lambda})$ and $(1/\lambda, 1/\bar{\lambda})$ are linearly independent.
Reduction to anti-triangular form is not always possible, but as the following theorem shows reduction to block-anti-triangular form can always be achieved.

Theorem 2.1 ([2, 3]) *Let $A \in \mathbb{F}^{n\times n}$ with $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ be such that $(A, A^\star)$ is a regular pair*[1]*. Then there exists a matrix $U \in \mathbb{F}^{n\times n}$ with $U^H U = I$ such that*

$$U^\star A U = R = \begin{array}{c} \\ n_1 \\ \\ n_k \\ \\ n_1 \end{array} \begin{array}{c} \begin{array}{cccccc} n_1 & & n_k & & n_1 \end{array} \\ \begin{bmatrix} 0 & \cdots & 0 & & R_{1,2k-1} \\ \vdots & \iddots & & \iddots & \vdots \\ 0 & & R_{k,k} & & \vdots \\ & \iddots & & & \vdots \\ R_{2k-1,1} & \cdots & \cdots & \cdots & R_{2k-1,2k-1} \end{bmatrix} \end{array} \tag{2.3}$$

where

- *$(R_{k,k}, R_{k,k}^\star)$ has only $\star$-exceptional eigenvalues;*
- *$n_1, n_2, \ldots, n_{k-1}$ are all 1 if $\mathbb{F} = \mathbb{C}$ and are $\in \{1, 2\}$ if $\mathbb{F} = \mathbb{R}$;*
- *the spectrum of $(A, A^\star)$ is given by the union of the spectra of $(R_{k+i,k-i}, R_{k-i,k+i}^\star)$ for $i = -(k-1), \ldots, (k-1)$. If $(R_{k+i,k-i}, R_{k-i,k+i}^\star)$ is a 2×2 block and $i \neq 0$, then the eigenvalues of this block are given by a complex conjugate pair.*

[1] A matrix pair (A, B) is called *regular* if there is a number $\alpha \in \mathbb{C}$ such that $A - \alpha B$ is nonsingular.

The matrix R in (2.3) is called extended palindromic Schur form of A. The second theorem relates the generalized Schur form to the extended palindromic Schur form.

Theorem 2.2 ([2, 3]) *Let $A \in \mathbb{F}^{n\times n}$ with $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ be such that $(A, A^\star)$ is a regular pair. Let $Q = [Q_1, Q_2], Z = [Z_1, Z_2] \in \mathbb{F}^{n\times n}$ with $Q_1, Z_1 \in \mathbb{F}^{n\times n_1}$ be unitary matrices such that*

$$Q^H A Z = S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix},\ Q^H A^\star Z = T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}, \tag{2.4}$$

with $S_{11}, T_{11} \in \mathbb{F}^{n_1\times n_1}$, is a partial generalized (real) Schurform of $(A, A^\star)$ where the spectrum of (S_{11}, T_{11}) is $\star$-reciprocal free[2]. Let F denote the flip matrix

$$F = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}. \tag{2.5}$$

Then there exists a matrix $V \in \mathbb{F}^{n\times(n-2n_1)}$ such that $U = \left[Z_1, V, Q_1^{\star H} F\right]$ is unitary and

$$U^\star A U = \begin{bmatrix} & & R_{13} \\ & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{2.6}$$

with $R_{13} = T_{11}^\star F,\ R_{31} = F S_{11} \in \mathbb{F}^{n_1\times n_1}$.

Assume that Q, Z are such that $S = Q^H A Q, T = Q^H A^\star Q$ are in (real) generalized Schur form. Assume further that the eigenvalues are ordered such that the leading eigenvalues (along the diagonals of S, T) form a $\star$-reciprocal free set of maximal size n_1. We discuss the eigenvalue ordering in more detail in Section 3. Then Theorem 2.2 can be invoked resulting in (2.6) to reduce to the form (2.3).
These considerations lead to the following algorithm.

Algorithm 2.3 Palindromic Laub trick
Input: $A \in \mathbb{F}^{n,n}$
Output: unitary U, $R = U^\star A U$ in extended palindromic Schur form (2.3)
1: compute a (real) generalized Schur form of $(A, A^\star)$, i.e., $Q^H A Z = S,\ Q^H A^\star Z = T$
2: reorder the (real) generalized Schur form such that the leading eigenvalues are $\star$-reciprocal free
3: $U \leftarrow [z_1, z_2, \ldots, z_{\lceil n/2\rceil}, (q^H_{\lfloor n/2\rfloor})^\star, \ldots, (q_2^H)^\star, (q_1^H)^\star]$
4: $R \leftarrow U^\star A U$

2.2 Even pairs

For even problems, (conjugate) transposing (2.2) yields $x^\star A = -\lambda x^\star B$. Hence, the eigenvalues of an even problem occur in pairs $(\lambda, -\lambda^\star)$. Exceptional eigenvalues of $\star$-even problems are such that $\lambda = -\lambda^\star$, which are 0 and ∞ in the T-case and the whole imaginary axis including ∞ in the real and the H-case.
Analogous to the palindromic case we now state results providing the existence and a construction of the wanted matrix U.

[2] A subset $\mathcal{S} \subset \mathbb{C} \cup \{\infty\}$ is said to be *$\star$-reciprocal free* if $\lambda \in \mathcal{S}$ implies that $1/\lambda^\star \notin \mathcal{S}$. Here we define $1/0 = \infty$ and $1/\infty = 0$.

Theorem 2.4 ([3]) *Let $A, B \in \mathbb{F}^{n\times n}$ with $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ be such that $A = A^\star, B = -B^\star$ and (A, B) is a regular pair. Then there exists a matrix $U \in \mathbb{F}^{n\times n}$ with $U^H U = I$ such that*

$$U^\star AU = R = \begin{array}{c} \\ n_1 \\ n_k \\ \\ n_1 \end{array} \begin{array}{c} \begin{array}{cccccc} n_1 & & n_k & & n_1 \end{array} \\ \begin{bmatrix} 0 & \cdots & 0 & & R_{1,2k-1} \\ \vdots & \iddots & & \iddots & \vdots \\ 0 & & R_{k,k} & & \vdots \\ & \iddots & & & \vdots \\ R_{2k-1,1} & \cdots & \cdots & \cdots & R_{2k-1,2k-1} \end{bmatrix} \end{array},$$

$$U^\star BU = K = \begin{array}{c} \\ n_1 \\ n_k \\ \\ n_1 \end{array} \begin{array}{c} \begin{array}{cccccc} n_1 & & n_k & & n_1 \end{array} \\ \begin{bmatrix} 0 & \cdots & 0 & & K_{1,2k-1} \\ \vdots & \iddots & & \iddots & \vdots \\ 0 & & K_{k,k} & & \vdots \\ & \iddots & & & \vdots \\ K_{2k-1,1} & \cdots & \cdots & \cdots & K_{2k-1,2k-1} \end{bmatrix} \end{array}, \tag{2.7}$$

where $R_{ij} = R_{ji}^\star$, $K_{ij} = -K_{ji}^\star$ and

- *$(R_{k,k}, K_{k,k})$ have only $\star$-exceptional eigenvalues;*
- *$n_1, n_2, \ldots, n_{k-1}$ are all 1 if $\mathbb{F} = \mathbb{C}$ and are $\in \{1, 2\}$ if $\mathbb{F} = \mathbb{R}$;*
- *the spectrum of (A, B) is given by the union of the spectra of $(R_{k+i,k-i}, K_{k+i,k-i})$ for $i = -(k-1), \ldots, (k-1)$. If $(R_{k+i,k-i}, K_{k+i,k-i})$ is a 2×2 block and $i \neq 0$ then the eigenvalues of this block are given by a complex conjugate pair.*

The pair (R, K) as in (2.7) is called extended even Schur form of (A, B).

Theorem 2.5 ([3]) *Let $A, B \in \mathbb{F}^{n\times n}$ with $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ be such that $A = A^\star, B = -B^\star$ and (A, B) is a regular pair. Let $Q = [Q_1, Q_2], Z = [Z_1, Z_2] \in \mathbb{F}^{n\times n}$ with $Q_1, Z_1 \in \mathbb{F}^{n\times n_1}$ be unitary matrices such that*

$$Q^H AZ = S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix},\ Q^H BZ = T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \tag{2.8}$$

with $S_{11}, T_{11} \in \mathbb{F}^{n_1\times n_1}$ is a partial generalized (real) Schurform of (A, B) where the spectrum of (S_{11}, T_{11}) is $\star$-negative free[3]. Let F denote the flip matrix (2.5). Then there exists a matrix $V \in \mathbb{F}^{n\times(n-2n_1)}$ such that $U = \left[Z_1, V, Q_1^{\star H} F\right]$ is unitary and

$$U^\star AU = \begin{bmatrix} & & R_{13} \\ & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix},\ U^\star BU = \begin{bmatrix} & & K_{13} \\ & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix}, \tag{2.9}$$

with $R_{31} = FS_{11}, R_{13} = R_{31}^\star, K_{31} = FT_{11}, K_{13} = -K_{31}^\star \in \mathbb{F}^{n_1\times n_1}$.

Assume that Q, Z are such that $S = Q^H AQ, T = Q^H BQ$ are in (real) generalized Schur form. Similar to the palindromic case we assume further that the eigenvalues are ordered such that the leading eigenvalues (along the diagonals of S, T) form a $\star$-negative free set of maximal size n_1. We discuss the eigenvalue ordering in more detail in Section 3. Then Theorem 2.5 can be used for n_1 resulting in (2.9) to reduce to the form (2.7).
These considerations lead to the following algorithm.

[3]A subset $\mathcal{S} \subset \mathbb{C} \cup \{\infty\}$ is said to be *$\star$-negative free* if $\lambda \in \mathcal{S}$ implies that $-\lambda^\star \notin \mathcal{S}$.

Algorithm 2.6 Even Laub trick
Input: $A = A^\star \in \mathbb{F}^{n,n}$ and $B = -B^\star \in \mathbb{F}^{n,n}$
Output: unitary U, $R = U^\star AU$ and $K = U^\star BU$ in extended even Schur form (2.7)
1: compute a (real) generalized Schur form of (A, B), i.e., $Q^H AZ = S,\ Q^H BZ = T$
2: reorder the (real) generalized Schur form such that the leading eigenvalues are $\star$-negative free
3: $U \leftarrow [z_1, z_2, \ldots, z_{\lceil n/2 \rceil}, (q^H_{\lfloor n/2 \rfloor})^\star, \ldots, (q^H_2)^\star, (q^H_1)^\star]$
4: $R \leftarrow U^\star AU$, $K \leftarrow U^\star BU$

3 Numerical Realization

In this chapter we explain certain aspects of the algorithms in detail.

Reorthogonalization By Theorems 2.2 and 2.5 the matrix U obtained in line 3 of the Algorithms 2.3 and 2.6 is unitary, but in finite precision arithmetic or if $\star$-exceptional eigenvalues appear this may no longer be the case. Therefore in our routines we offer an option to reorthogonalize the matrix. This amounts to replacing line 3 in Algorithms 2.3 and 2.6 by
3.1: $[z_1, (q^H_1)^\star, z_2, (q^H_2)^\star, \ldots, z_{\lfloor n/2 \rfloor}, (q^H_{\lfloor n/2 \rfloor})^\star] \to \hat{Q}\hat{R}$ (QR factorization)
3.2: $U \leftarrow [\hat{q}_1, \hat{q}_3, \hat{q}_5, \ldots, \hat{q}_6, \hat{q}_4, \hat{q}_2]$.

Eigenvalue ordering in the palindromic case If an eigenvalue λ lies inside the unit circle then the paired eigenvalue $1/\lambda^\star$ is outside the unit circle and vice versa. Thus, it is straight forward to order the eigenvalues by their absolute value since then the leading ones are $\star$-reciprocal free. In our code we sort in ascending order. The $\star$-exceptional eigenvalues for the palindromic problem are 1 and -1 if $\star = T$ and the whole unit circle including ∞ if $\star = H$ and for real matrices. As a convenient side effect, the $\star$-exceptional eigenvalues are moved to the middle of the generalized Schur form by the above ordering scheme.

Eigenvalue ordering in the even case If an eigenvalue λ has positive real part then the paired eigenvalue $-\lambda^\star$ has negative real part and vice versa. As we mentioned in section 2.2 the $\star$-exceptional eigenvalues for even pencils are 0 and ∞ if $\star = T$ and the whole imaginary axis including ∞ if $\star = H$ and for real matrices. Unfortunately the straight forward approach of sorting by real part fails to classify eigenvalues of large magnitude (which may be perturbations of infinite eigenvalues) as being close to the imaginary axis. To bypass this problem we use the *Cayley transform*, which maps a value λ to $\frac{\lambda+1}{\lambda-1}$. The Cayley transform has the pleasing property to map the imaginary axis to the unit circle, $\{0, \infty\}$ to $\{1, -1\}$, the open left half plane to the open unit disc and the open right half plane to the outside of the unit circle. In other words the even $\star$-exceptional eigenvalues get mapped to the palindromic $\star$-exceptional eigenvalues. Therefore, we can use the same ordering scheme as before: we order by absolute value of the Cayley transformed eigenvalues.

Matrix storage In the palindromic case the input matrix A is overwritten by the matrix $R = U^\star AU$ on exit from the routine. The situation in the even case is more complex. Here only the upper triangular part of the symmetric (or Hermitian) matrix A is referenced. On exit it is overwritten by the upper triangular part of $R = U^\star AU$. The analogous statements apply to the lower triangular parts of the matrices B and $K = U^\star BU$.
The unreferenced elements remain untouched with one exception: It is possible to work with just one matrix by storing the upper triangular part of A and the lower triangular part of B in the same matrix and passing it twice to the routine.

Mixed precision routines We also provide mixed precision routines to offer a faster computation using less workspace. For real palindromic problems, for instance, this reduces the memory

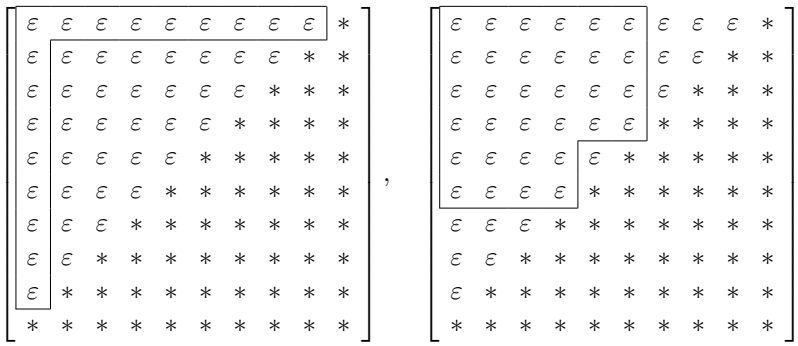

Figure 1: Area that is covered in $d(1)$ and $d(4)$ for $n = 10$.

requirements from $3n^2 + 11n + 16$ to $n^2 + 6n + 8$. Of course, then the results will be block-anti-triangular only with single precision accuracy.
The lines $1, 2$ of Algorithms 2.3 and 2.6 are carried out in single precision. In order to obtain a matrix U that is unitary to double precision we always reorthogonalize as described above. This reorthogonalization and line 4 of the two algorithms are performed in double precision.
The even mixed precision routines differ from the others in the following way. The products $U^\star AU$ and $U^\star BU$ are formed as matrix matrix multiplies only if the provided workspace is large enough. Otherwise these products are formed by a series of Householder updates. This saves up to $0.5n^2$ workspace.

Measuring the distance to anti-triangular form In exact arithmetic the output matrices R and K will be (block-) anti-triangular indeed. In finite arithmetic, however this is in general not the case. Thus we provide the following measurement of the distance to anti-triangular form:

$$d(i) = \|A(1 : i, 1 : n - i)\|_F^2 + \|A(i + 1 : n - i, 1 : i)\|_F^2 \text{ for } i = 1, \ldots, \tfrac{n}{2}.$$

The i-th element of the vector d contains the squared Frobenius norm of the Γ-shaped domain as illustrated in Figure 1. If a value $d(i)$ is neglectable then the problem decomposes into three independent problems of the sizes $i \times i$, $(n - 2i) \times (n - 2i)$ and $i \times i$.

Matlab interface We provide the two Matlab interfaces `pallaub.m` (for palindromic problems) and `skslaub.m` (for even problems). These interfaces offer the same functionality as the Fortran routines including support for the three variants ($\mathbb{F} = \mathbb{R}$; $\mathbb{F} = \mathbb{C}, \star = H$; and $\mathbb{F} = \mathbb{C}, \star = T$), reorthogonalization, and mixed precision. One of the convenient features of the Matlab interfaces is that it is automatically determined whether to use real or complex arithmetic. For example the two lines

```
A=rand(5);
[R,U]=pallaub(A);
```

return the extended palindromic Schur form of a random matrix.

routine	description: computes ...	sec.
DPALLAUB	real extended palindromic Schurform	4.1
DPALLAUB_MIX	real extended palindromic Schurform with mixed precision	4.1
ZPALLAUB	complex extended palindromic Schurform	4.3
ZPALLAUB_MIX	complex extended palindromic Schurform with mixed precision	4.3
DSKSLAUB	real extended even Schurform	4.5
DSKSLAUB_MIX	real extended even Schurform with mixed precision	4.5
ZSKSLAUB	complex extended even Schurform	4.7
ZSKSLAUB_MIX	complex extended even Schurform with mixed precision	4.7
DATRIERR	distance of a real matrix to anti-triangular form	4.2
ZATRIERR	distance of a complex matrix to anti-triangular form	4.4
DATRIERRSYM	distance of a real (skew-) sym. matrix to anti-triangular form	4.6
ZATRIERRSYM	distance of a complex (skew-) sym. matrix to anti-triangular form	4.8

Table 1: List of routines

4 Documentation

In this section we describe our implementations of the algorithms discussed so far. The programs are written in Fortran 77. (Note that some names of subroutines and variables are longer than six characters which is not valid in Fortran 77, but supported by every common compiler.) We mainly followed the SLICOT Implementation and Documentation Standards [4]. The computational routines are named by the following scheme `xyyyLAUB[_MIX]`, where `x` is `D` (double precision) or `Z` (double complex), `yyy` is `PAL` (palindromic) or `SKS` (symmetric/skew-symmetric, i.e., even). The optional suffix `_MIX` indicates the mixed precision routines. Moreover there are routines `xATRIERR[SYM]` to compute the distance from anti-triangularity and `xSKSUPDATE` to perform a Householder update of a (skew-) symmetric matrix. The individual routines are listed in Table 1 and are described in more detail in the following subsections.

4.1 DPALLAUB and DPALLAUB_MIX

```
      SUBROUTINE DPALLAUB (ORTH,N,A,LDA,U,LDU,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        ORTH
      INTEGER            N,LDA,LDU,LDWORK,INFO
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),U(LDU,N),DWORK(LDWORK)

      SUBROUTINE DPALLAUB_MIX(N,A,LDA,U,LDU,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDU,LDWORK,INFO
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),U(LDU,N),DWORK(LDWORK)
```

Purpose: Implements Algorithm 2.3.

ORTH – CHARACTER*1 (input).

Indicates whether the matrix U should be reorthogonalized before outputting.
ORTH = 'T','O','R': matrix U will be reorthogonalized
ORTH = 'F' : no reorthogonalization
`DPALLAUB_MIX` does not have this entry, because it reorthogonalizes at all times.

N – INTEGER (input).

The number of rows of the matrices A and U. N $\geq$ 0.

A – DOUBLE PRECISION array of DIMENSION (LDA,N) (input/output).

On entry, the matrix A.
On exit, the matrix R.

LDA – INTEGER (input).

The leading dimension of the array A.

U – DOUBLE PRECISION array of DIMENSION (LDU,N) (output).

The orthogonal matrix U.

LDU – INTEGER (input).

The leading dimension of the array U.

DWORK – DOUBLE PRECISION array of DIMENSION (LDWORK) (workspace/output).

On exit, if INFO = 0, DWORK(1) returns the optimal LDWORK.
If INFO > 0, DWORK(2) returns the original INFO value of the respective subroutine (see error indicator INFO below).
On exit, DWORK(3:(N+2)) returns the list of the diagonal block sizes of A.
DWORK((N+3):(N+N/2+2)) returns the error computed by `DATRIERR`. For further details about the error see subroutine `DATRIERR`.

LDWORK – INTEGER (input).

The dimension of the array DWORK.
for `DPALLAUB`: LDWORK $\geq \max(1\ ,\ 3N^2 + 11N + 16)$
for `DPALLAUB_MIX`: LDWORK $\geq \max(1\ ,\ N^2 + 6N + 8)$

INFO – INTEGER (output).

for `DPALLAUB`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `DPALLAUB` has an illegal value.
>0: if INFO = 1, error occured in `DGGES`,
= 2, error occured in `DTGEXC`,
= 3, error occured in `DGEQRF`,
= 4, error occured in `DORGQR`,
= 5, error occured in `DATRIERR`.

for `DPALLAUB_MIX`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `DPALLAUB_MIX` has an illegal value.
>0: if INFO = 1, error occured in `SGGES`,
= 2, error occured in `STGEXC`,
= 3, error occured in `DGEQRF`,
= 4, error occured in `DORGQR`,
= 5, error occured in `DATRIERR`.

4.2 DATRIERR

```
      SUBROUTINE DATRIERR(N,A,LDA,RES,LDRES,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDRES,INFO
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),RES((N/2-1)*LDRES+1)
```

Purpose: Computes the distance of a real square matrix A to anti-triangular form as described in Section 3 (see also Figure 1).

N – INTEGER (input).

The number of rows of the matrix A. N ≥ 0.

A – DOUBLE PRECISION array of DIMENSION (LDA,N) (input).

The matrix A.

LDA – INTEGER (input).

The leading dimension of the array A.

RES – DOUBLE PRECISION array of DIMENSION (RES((N/2-1)*LDRES+1)(output).

$RES(i) = \|A(1:i, 1:n-i)\|_F^2 + \|A(i+1:n-i, 1:i)\|_F^2 \quad \forall i = 1, \ldots, \frac{n}{2}$

LDRES – INTEGER (input).

The leading dimension (or increment) of the array RES.

INFO – INTEGER (output).

=0: successful exit;
<0: if INFO = -i, the i-th argument of DATRIERR has an illegal value.

4.3 ZPALLAUB and ZPALLAUB_MIX

```
SUBROUTINE ZPALLAUB (OP,ORTH,N,A,LDA,U,LDU,ZWORK,LZWORK,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        OP,ORTH
      INTEGER            N,LDA,LDU,LZWORK,LDWORK,INFO
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N),U(LDU,N),ZWORK(LZWORK)
      DOUBLE PRECISION   DWORK(LDWORK)

SUBROUTINE ZPALLAUB_MIX (OP,N,A,LDA,U,LDU,ZWORK,LZWORK,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        OP
      INTEGER            N,LDA,LDU,LZWORK,LDWORK,INFO
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N),U(LDU,N),ZWORK(LZWORK)
      DOUBLE PRECISION   DWORK(LDWORK)
```

Purpose: Implements Algorithm 2.3.

OP – CHARACTER*1 (input).

Indicates whether the Laub-Trick will be computed for a T-palindromic or H-palindromic pair of matrices:
= 'T': complex transpose (T-palindromic);
= 'H': complex conjugate transpose (H-palindromic).

ORTH – CHARACTER*1 (input).

Indicates whether the matrix U should be reorthogonalized before outputting.
ORTH = 'T','O','R': matrix U will be reorthogonalized
ORTH = 'F' : no reorthogonalization
ZPALLAUB_MIX does not have this entry, because it orthogonalizes at all times.

N – INTEGER (input).

The number of rows of the matrices A and U. N $\geq$ 0.

A – DOUBLE COMPLEX array of DIMENSION (LDA,N) (input/output).

On entry, the first matrix of the palindromic pair of matrices.
On exit, the by matrix U to anti-triangular form reduced matrix.

LDA – INTEGER (input).

The leading dimension of the array A.

U – DOUBLE COMPLEX array of DIMENSION (LDU,N) (output).

The matrix U that reduces A to anti-triangular form.

LDU – INTEGER (input).

The leading dimension of the array U.

ZWORK – DOUBLE COMPLEX array of DIMENSION (LZWORK) (workspace/output).

On exit, if INFO = 0, ZWORK(1) returns the optimal LZWORK.
If INFO > 0, ZWORK(2) returns the original INFO value of the respective subroutine (see error indicator INFO below).

LZWORK – INTEGER (input).

The dimension of the array ZWORK.
for `ZPALLAUB`: LZWORK $\geq \max(1\ ,\ 3N^2 + 4N)$
for `ZPALLAUB_MIX`: LZWORK $\geq \max(1\ ,\ N^2 + 2N + 8)$

DWORK – DOUBLE COMPLEX array of DIMENSION (LDWORK) (workspace/output).

On exit, DWORK(1:(N/2)) returns the error computed by `ZATRIERR`. For further details about the error see subroutine `ZATRIERR`.

LDWORK – INTEGER (input).

The dimension of the array DWORK.
LDWORK = 8N

INFO – INTEGER (output).

for `ZPALLAUB`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `ZPALLAUB` has an illegal value.
>0: if INFO = 1, error occured in `ZGGES`,
= 2, error occured in `ZTGEXC`,
= 3, error occured in `ZGEQRF`,
= 4, error occured in `ZUNGQR`,
= 5, error occured in `ZATRIERR`.

for `ZPALLAUB_MIX`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `ZPALLAUB_MIX` has an illegal value.
>0: if INFO = 1, error occured in `CGGES`,
= 2, error occured in `CTGEXC`,
= 3, error occured in `ZGEQRF`,
= 4, error occured in `ZUNGQR`,
= 5, error occured in `ZATRIERR`.

4.4 ZATRIERR

```
      SUBROUTINE ZATRIERR(N,A,LDA,RES,LDRES,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDRES,INFO
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N)
      DOUBLE PRECISION   RES((N/2-1)*LDRES+1)
```

Purpose: Computes the distance of a complex square matrix A to anti-triangular form as described in Section 3 (see also Figure 1).

N – INTEGER (input).

The number of rows of the matrix A. N $\geq$ 0.

A – DOUBLE COMPLEX array of DIMENSION (LDA,N) (input).

The matrix A.

LDA – INTEGER (input).

The leading dimension of the array A.

RES – DOUBLE PRECISION array of DIMENSION (RES((N/2-1)*LDRES+1)(output).

$RES(i) = \|A(1:i, 1:n-i)\|_F^2 + \|A(i+1:n-i, 1:i)\|_F^2 \quad \forall i = 1, \ldots, \frac{n}{2}$

LDRES – INTEGER (input).

The leading dimension (or increment) of the array RES.

INFO – INTEGER (output).

=0: successful exit;
<0: if INFO = -i, the i-th argument of ZATRIERR has an illegal value.

4.5 DSKSLAUB and DSKSLAUB_MIX

```
      SUBROUTINE DSKSLAUB (ORTH,N,A,LDA,B,LDB,U,LDU,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        ORTH
      INTEGER            N,LDA,LDB,LDU,LDWORK,INFO
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),B(LDB,N),U(LDU,N),DWORK(LDWORK)

      SUBROUTINE DSKSLAUB_MIX(N,A,LDA,B,LDB,U,LDU,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDB,LDU,LDWORK,INFO
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),B(LDB,N),U(LDU,N),DWORK(LDWORK)
```

Purpose: Implements Algorithm 2.6.

ORTH – CHARACTER*1 (input).

Indicates whether the matrix U should be reorthogonalized before outputting.
ORTH = 'T','O','R': matrix U will be reorthogonalized
ORTH = 'F' : no reorthogonalization
DSKSLAUB_MIX does not have this entry, because it orthogonalizes at all times.

N – INTEGER (input).

The number of rows of the matrices A,B and U. N $\geq$ 0.

A – DOUBLE PRECISION array of DIMENSION (LDA,N) (input/output).

On entry, the symmetric matrix.
We only change and use the upper triangle of the matrix.
On exit, the upper triangle of the by matrix U to anti-triangular form reduced matrix. The lower triangle still is the one of the input matrix A.

LDA – INTEGER (input).

The leading dimension of the array A.

B – DOUBLE PRECISION array of DIMENSION (LDB,N) (input/output).

On entry, the skew symmetric matrix.
We only change and use the lower triangle of the matrix.
On exit, the lower triangle of the by matrix U to anti-triangular form reduced matrix. The upper triangle still is the one of the input matrix B.

LDB – INTEGER (input).

The leading dimension of the array B.

U – DOUBLE PRECISION array of DIMENSION (LDU,N) (output).

The matrix U that reduces A and B to anti-triangular form.

LDU – INTEGER (input).

The leading dimension of the array U.

DWORK – DOUBLE PRECISION array of DIMENSION (LDWORK) (workspace/output).

On exit, if INFO = 0, DWORK(1) returns the optimal LDWORK.
If INFO >0, DWORK(2) returns the original INFO value of the respective subroutine (see error indicator INFO below).
On exit, DWORK(3:(N+2)) returns list of the diagonal block sizes of A.
DWORK((N+3):(N+N/2+2)) and DWORK((N+N/2+3):(2N+2)) return the errors computed by `DATRIERRSYM` for the to anti-triangular form reduced matrices A and B, respectively.
For further details about the computed error see `DATRIERRSYM`.

LDWORK – INTEGER (input).

The dimension of the array DWORK.
for `DSKSLAUB`: LDWORK $\geq \max(1\ ,\ 3N^2 + 11N + 16)$
for `DSKSLAUB_MIX`: LDWORK $\geq \max(1\ ,\ N^2 + 5N + 8\ ,\ N^2 + 6N)$
for optimal performance:
LDWORK $\geq \max(1\ ,\ 1.5(N^2 + N) + 0.5\ ,\ N^2 + 5N + 8\ ,\ N^2 + 6N)$

INFO – INTEGER (output).

for `DSKSLAUB`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `DSKSLAUB` has an illegal value.
>0: if INFO = 1, error occured in `DGGES`,
= 2, error occured in `DTGEXC`,
= 3, error occured in `DGEQRF`,

= 4, error occured in DORGQR,
= 5, error occured in DATRIERRSYM,
= 6, error occured in DATRIERRSYM.

for DSKSLAUB_MIX:
=0: successful exit;
<0: if INFO = -i, the i-th argument of DSKSLAUB_MIX has an illegal value.
>0: if INFO = 1, error occured in SGGES,
= 2, error occured in STGEXC,
= 3, error occured in DGEQRF,
= 4, error occured in DORGQR,
= 5, error occured in DATRIERRSYM,
= 6, error occured in DATRIERRSYM,
= 7, error occured in DSKSUPDATE,
= 8, error occured in DSKSUPDATE,
= 9, error occured in DORMQR,
=10, error occured in DORMQR.

4.6 DATRIERRSYM

```
      SUBROUTINE DATRIERRSYM(UPLO,SYMSK,N,A,LDA,RES,LDRES,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDRES,INFO
      CHARACTER*1        OP,UPLO,SYMSK
C     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA,N),RES((N/2-1)*LDRES+1)
```

Purpose: Computes the distance of a (skew-) symmetric real matrix A to anti-triangular form as described in Section 3 (see also Figure 1).

UPLO – CHARACTER*1 (input).

Indicates whether the array A contains the upper or lower triangular part of the (skew-) symmetric matrix A.
= 'U': upper triangle
= 'L': lower triangle

SYMSK – CHARACTER*1 (input).

Indicates whether the matrix A is symmetric or skew-symmetric.
= 'S': symmmetric
= 'K': skew-symmetric

N – INTEGER (input).

The number of rows of the matrix A. N ≥ 0.

A – DOUBLE PRECISION array of DIMENSION (LDA,N) (input).

The upper or lower triangular part of the (skew-) symmetric matrix A.

LDA – INTEGER (input).

The leading dimension of the array A.

RES – DOUBLE PRECISION array of DIMENSION (RES((N/2-1)*LDRES+1)(output).

$RES(i) = \|A(1:i, 1:n-i)\|_F^2 + \|A(i+1:n-i, 1:i)\|_F^2 \quad \forall i = 1, \ldots, \frac{n}{2}$

LDRES – INTEGER (input).

The leading dimension (or increment) of the array RES.

INFO – INTEGER (output).

=0: successful exit;
<0: if INFO = -i, the i-th argument of DATRIERRSYM has an illegal value.

4.7 ZSKSLAUB and ZSKSLAUB_MIX

```
      SUBROUTINE ZSKSLAUB(OP,ORTH,N,A,LDA,B,LDB,U,LDU,ZWORK,LZWORK,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        OP,ORTH
      INTEGER            N,LDA,LDB,LDU,LZWORK,LDWORK,INFO
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N),B(LDB,N),U(LDU,N),ZWORK(LZWORK)
      DOUBLE PRECISION   DWORK(LDWORK)

      SUBROUTINE ZSKSLAUB_MIX (OP,N,A,LDA,B,LDB,U,LDU,ZWORK,LZWORK,DWORK,LDWORK,INFO)
C     .. Scalar Arguments ..
      CHARACTER*1        OP
      INTEGER            N,LDA,LDB,LDU,LZWORK,LDWORK,INFO
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N),B(LDB,N),U(LDU,N),ZWORK(LZWORK)
      DOUBLE PRECISION   DWORK(LDWORK)
```

Purpose: Implements Algorithm 2.6.

OP – CHARACTER*1 (input).

Indicates whether the Laub-Trick will be computed for a T-palindromic or H-palindromic pair of matrices:
= 'T': complex transpose (T-palindromic);
= 'H': complex conjugate transpose (H-palindromic).

ORTH – CHARACTER*1 (input).

Indicates whether the matrix U should be reorthogonalized before outputting.
ORTH = 'T','O','R': matrix U will be reorthogonalized
ORTH = 'F' : no reorthogonalization
ZPALLAUB_MIX does not have this entry, because it orthogonalizes at all times.

N – INTEGER (input).

The number of rows of the matrices A and U. $N \geq 0$.

A – DOUBLE COMPLEX array of DIMENSION (LDA,N) (input/output).

On entry, the first matrix of the palindromic pair of matrices.
On exit, the by matrix U to anti-triangular form reduced matrix.

LDA – INTEGER (input).

The leading dimension of the array A.

B – DOUBLE COMPLEX array of DIMENSION (LDB,N) (input/output).

On entry, the skew symmetric matrix.
We only change and use the lower triangle of the matrix.
On exit, the lower triangle of the by matrix U to anti-triangular form reduced matrix. The upper triangle still is the one of the input matrix B.

LDB – INTEGER (input).

The leading dimension of the array B.

U – DOUBLE COMPLEX array of DIMENSION (LDU,N) (output).

The matrix U that reduces A and B to anti-triangular form.

LDU – INTEGER (input).

The leading dimension of the array U.

ZWORK – DOUBLE COMPLEX array of DIMENSION (LZWORK) (workspace/output).

On exit, if INFO = 0, ZWORK(1) returns the optimal LZWORK.
If INFO > 0, ZWORK(2) returns the original INFO value of the respective subroutine (see error indicator INFO below).

LZWORK – INTEGER (input).

The dimension of the array DWORK.
for `ZSKSLAUB`: LZWORK $\geq \max(1\ ,\ 3N^2 + 4N)$
for `ZSKSLAUB_MIX`: LZWORK $\geq \max(1\ ,\ N^2 + 4N)$
for optimal performance:
LZWORK $\geq \max(1\ ,\ N^2 + 0.5(N^2 + N + 1)\ ,\ N^2 + 4N\ ,\ N^2 + 6N)$

DWORK – DOUBLE COMPLEX array of DIMENSION (LDWORK) (workspace/output).

On exit, DWORK(1:(N/2)) returns the error computed by `ZATRIERRSYM` for U'*A*U, where A is the input matrix A. DWORK((N/2)+1 : N) returns the error computed by `ZATRIERRSYM` for U'*A'U, where B is the input matrix B. For further details see `ZATRIERRSYM`.

LDWORK – INTEGER (input).

The dimension of the array DWORK.
LDWORK = 8N

DWORK – DOUBLE PRECISION array of DIMENSION (LDWORK) (workspace/output).

On exit, DWORK(1:(N/2)) and DWORK((N/2+1):N) return the errors computed by `ZATRIERRSYM` for the to anti-triangular form reduced matrices A and B, respectively.
For further details about the computed error see `ZATRIERRSYM`.

INFO – INTEGER (output).

for `ZSKSLAUB`:
=0: successful exit;
<0: if INFO = -i, the i-th argument of `ZSKSLAUB` has an illegal value.
>0: if INFO = 1, error occured in `ZGGES`,
= 2, error occured in `ZTGEXC`,
= 3, error occured in `ZGEQRF`,
= 4, error occured in `ZUNGQR`,
= 5, error occured in `ZATRIERRSYM`,
= 6, error occured in `ZATRIERRSYM`.

for `ZSKSLAUB_MIX`:

=0: successful exit;
<0: if INFO = -i, the i-th argument of ZSKSLAUB_MIX has an illegal value.
>0: if INFO = 1, error occured in CGGES,
= 2, error occured in CTGEXC,
= 3, error occured in ZGEQRF,
= 4, error occured in ZUNGQR,
= 5, error occured in ZATRIERRSYM,
= 6, error occured in ZATRIERRSYM,
= 7, error occured in ZUNMQR,
= 8, error occured in ZUNMQR,
= 9, error occured in ZSKSUPDATE,
=10, error occured in ZSKSUPDATE.

4.8 ZATRIERRSYM

```
      SUBROUTINE ZATRIERRSYM(OP,UPLO,SYMSK,N,A,LDA,RES,LDRES,INFO)
C     .. Scalar Arguments ..
      INTEGER            N,LDA,LDRES,INFO
      CHARACTER*1        OP,UPLO,SYMSK
C     .. Array Arguments ..
      COMPLEX*16         A(LDA,N)
      DOUBLE PRECISION   RES((N/2-1)*LDRES+1)
```

Purpose: Computes the distance of a (skew-) symmetric or (skew-) Hermitian complex matrix A to anti-triangular form as described in Section 3 (see also Figure 1).

OP – CHARACTER*1 (input).

= 'T': complex transpose
= 'H': complex conjugate transpose

UPLO – CHARACTER*1 (input).

Indicates whether the array A contains the upper or lower triangular part of the (skew-) symmetric or (skew-) Hermitian matrix A.
= 'U': upper triangle
= 'L': lower triangle

SYMSK – CHARACTER*1 (input).

Indicates whether the matrix A is symmetric (or Hermitian) or skew-symmetric (or skew-Hermitian).
= 'S': symmmetric or Hermitian
= 'K': skew-symmetric or skew-Hermitian

N – INTEGER (input).

The number of rows of the matrix A. N $\geq$ 0.

A – DOUBLE COMPLEX array of DIMENSION (LDA,N) (input).

For this matrix the residual gets computed. See RES.

LDA – INTEGER (input).

The leading dimension of the array A.

RES – DOUBLE PRECISION array of DIMENSION (RES((N/2-1)*LDRES+1)(output).

$RES(i) = \|A(1:i, 1:n-i)\|_F^2 + \|A(i+1:n-i, 1:i)\|_F^2 \quad \forall i = 1, \ldots, \frac{n}{2}$

LDRES – INTEGER (input).

The leading dimension of the array RES.

INFO – INTEGER (output).

=0: successful exit;
<0: if INFO = -i, the i-th argument of `ZATRIERRSYM` has an illegal value.

5 Numerical Example

Here we demonstrate how to use our routines. We illustrate a successful application for both the palindromic and the even case.

5.1 Palindromic Case

In `SIMPLEEXAMPLEPAL.F` (Section 5.1.1) we apply the three routines `DPALLAUB`, `ZPALLAUB` and `ZPALLAUB_MIX` to the following matrix A.

$$A = \begin{bmatrix} 8 & 7 & 8 & 4 & 5 \\ 7 & 0 & 7 & 4 & 4 \\ 4 & 3 & 3 & 8 & 6 \\ 7 & 0 & 10 & 8 & 7 \\ 2 & 1 & 0 & 2 & 8 \end{bmatrix}.$$

We compare the results of `SIMPLEEXAMPLEPAL.F` at the end of Section 5.1.2.

5.1.1 Program Text

```
      PROGRAM SIMPLEEXAMPLEPAL
C
C     .. Local Scalars ..
      CHARACTER*1       ORTH
      INTEGER           N
      PARAMETER         (N=5)
      INTEGER           LDWORK,LZWORK,DZERO,INFO,I,J
      PARAMETER         (ORTH='T', LDWORK=3*N**2+11*N+16,
     $                  LZWORK=3*N**2+4*N, DZERO = 0.0E+0)
C     .. Local Arrays ..
      DOUBLE PRECISION A(N,N),A1(N,N),DU(N,N),DWORK(LDWORK),
     $                  BLOCKSIZEA1(N)
      DOUBLE COMPLEX    A2(N,N),A3(N,N),A4(N,N),ZU(N,N),ZWORK(LZWORK)
C     .. External Subroutines ..
      EXTERNAL          DPALLAUB,ZPALLAUB,ZPALLAUB_MIX
C     .. Executable Statements..
C
C
C     matrix A:
      A(1,1)=8.0
      A(2,1)=7.0
      A(3,1)=4.0
      A(4,1)=7.0
      A(5,1)=2.0
      A(1,2)=7.0
      A(2,2)=0.0
      A(3,2)=3.0
      A(4,2)=0.0
      A(5,2)=1.0
      A(1,3)=8.0
      A(2,3)=7.0
      A(3,3)=3.0
      A(4,3)=10.0
      A(5,3)=0.0
      A(1,4)=4.0
```

```
      A(2,4)=5.0
      A(3,4)=8.0
      A(4,4)=8.0
      A(5,4)=2.0
      A(1,5)=5.0
      A(2,5)=4.0
      A(3,5)=6.0
      A(4,5)=7.0
      A(5,5)=8.0
C
      DO I=1,N
         CALL DCOPY(N,A(1,I),1,A1(1,I),1)
      ENDDO
C
      DO I=1,N
         DO J=1,N
            A2(I,J)=CMPLX(A(I,J),DZERO)
         ENDDO
      ENDDO
C
      DO I=1,N
         CALL ZCOPY(N,A2(1,I),1,A3(1,I),1)
         CALL ZCOPY(N,A2(1,I),1,A4(1,I),1)
      ENDDO

C
C     Apply DPALLAUB
      CALL DPALLAUB (ORTH,N,A1,N,DU,N,DWORK,LDWORK,INFO)
C     save blocksize
      CALL DCOPY(N,DWORK(3),1,BLOCKSIZEA1(1),1)
C
C     Apply ZPALLAUB with OP='H'
      CALL ZPALLAUB ('H',ORTH,N,A2,N,ZU,N,ZWORK,LZWORK,DWORK,LDWORK,
     $               INFO)
C
C     Apply ZPALLAUB with OP='T'
      CALL ZPALLAUB ('T',ORTH,N,A3,N,ZU,N,ZWORK,LZWORK,DWORK,LDWORK,
     $               INFO)
C
C     Apply ZPALLAUB_MIX with OP='T'
      CALL ZPALLAUB_MIX ('T',N,A4,N,ZU,N,ZWORK,LZWORK,DWORK,LDWORK,
     $                   INFO)
C
C     Output
      OPEN(11,FILE = 'simpleexamplepal.OUT',STATUS = 'NEW')
C     Matrix A:
      WRITE(11,*) 'The matrix to be reduced: '
      WRITE(11,*)
      DO I=1,N
         WRITE(11,666) ((INT(A(I,J))),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying dpallaub:'
      WRITE(11,777) 'blocksize:',(INT(BLOCKSIZEA1(I)),I=1,N)
```

```
      WRITE(11,*)
C     Matrix A1:
      DO I=1,N
         WRITE(11,999) ((A1(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zpallaub with op=H:'
C     Matrix A2:
      DO I=1,N
         WRITE(11,888) ((A2(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zpallaub with op=T:'
C     Matrix A3:
      DO I=1,N
         WRITE(11,888) ((A3(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zpallaub_mix with op=T:'
C     Matrix A4:
      DO I=1,N
         WRITE(11,888) ((A4(I,J)),J=1,N)
      ENDDO
C
999   FORMAT (5(E9.2,2X))
888   FORMAT (5('(',E9.2,',',E9.2,')',2X))
777   FORMAT (A10,2X,5(I1,2X))
666   FORMAT (5(I2,2X))
C
C *** Last line of SIMPLEEXAMPLEPAL ***
      END
```

5.1.2 Results

The following is the content of the file `SIMPLEEXAMPLEPAL.OUT` (the output file of `SIMPLEEXAMPLEPAL.F`).

```
 The matrix to be reduced:

 8   7   8   4   5
 7   0   7   5   4
 4   3   3   8   6
 7   0  10   8   7
 2   1   0   2   8

 result of applying dpallaub:
blocksize:  2  1  2  0  0

-0.22E-14  -0.14E-14  -0.87E-17  -0.41E-14   0.15E+02
-0.49E-15   0.22E-14   0.16E-14   0.62E+01  -0.10E-13
 0.14E-14   0.27E-14   0.49E+01  -0.12E+00  -0.21E+01
-0.22E+01   0.12E+01  -0.96E+00   0.24E+01  -0.34E+01
 0.74E+01   0.27E+01  -0.66E+01   0.15E+01   0.20E+02

 result of applying zpallaub with op=H:
(-0.24E-14,-0.21E-14)  ( 0.15E-14, 0.23E-14)  (-0.25E-15, 0.77E-15)  (-0.23E-14,-0.57E-15)  ( 0.18E+01, 0.10E+02)
( 0.77E-15,-0.87E-17)  ( 0.11E-14, 0.45E-15)  (-0.12E-14,-0.25E-14)  (-0.94E+01,-0.12E+01)  (-0.85E+01, 0.37E+01)
( 0.71E-15, 0.54E-15)  (-0.22E-15, 0.12E-14)  ( 0.49E+01, 0.47E-15)  (-0.10E+01, 0.41E+00)  (-0.92E+00, 0.15E+01)
(-0.99E-15,-0.75E-15)  (-0.34E+01,-0.15E+01)  (-0.37E+01,-0.11E+01)  ( 0.63E+01, 0.17E+01)  ( 0.58E+01,-0.14E+01)
(-0.14E+01,-0.37E+01)  (-0.56E+01,-0.28E+01)  (-0.27E+01,-0.48E+01)  ( 0.72E+01, 0.46E+01)  ( 0.16E+02,-0.17E+01)
```

```
 result of applying zpallaub with op=T:
( 0.44E-14, 0.20E-14)  (-0.31E-14, 0.13E-14)  (-0.47E-16,-0.68E-15)  (-0.12E-14, 0.11E-14)  ( 0.18E+01,-0.10E+02)
(-0.27E-14,-0.99E-15)  ( 0.36E-14, 0.89E-14)  (-0.20E-14, 0.62E-14)  (-0.94E+01, 0.12E+01)  (-0.85E+01,-0.37E+01)
( 0.71E-15, 0.31E-15)  (-0.65E-15, 0.20E-14)  ( 0.47E+01, 0.14E+01)  (-0.98E+00,-0.49E+00)  (-0.79E+00,-0.16E+01)
(-0.48E-15,-0.68E-15)  (-0.34E+01,-0.15E+01)  (-0.36E+01,-0.14E+01)  ( 0.35E+01, 0.20E+01)  (-0.16E+01, 0.77E+01)
(-0.14E+01,-0.37E+01)  (-0.56E+01,-0.28E+01)  (-0.22E+01,-0.50E+01)  ( 0.30E+01, 0.95E+01)  (-0.32E+01, 0.15E+02)

 result of applying zpallaub_mix with op=T:
(-0.16E-05,-0.30E-06)  ( 0.63E-06, 0.18E-05)  ( 0.93E-06, 0.47E-06)  ( 0.11E-05,-0.10E-06)  ( 0.18E+01,-0.10E+02)
( 0.14E-05, 0.67E-06)  (-0.48E-06,-0.15E-05)  ( 0.14E-05,-0.99E-06)  (-0.94E+01, 0.12E+01)  (-0.43E+01, 0.81E+01)
(-0.10E-06,-0.22E-06)  ( 0.78E-06,-0.79E-07)  ( 0.49E+01, 0.82E+00)  (-0.44E+00,-0.10E+01)  (-0.17E+01,-0.51E+00)
( 0.12E-05, 0.39E-06)  (-0.34E+01,-0.15E+01)  (-0.19E+01,-0.34E+01)  (-0.17E+01, 0.36E+01)  (-0.16E+01, 0.77E+01)
(-0.14E+01,-0.37E+01)  (-0.33E+01, 0.54E+01)  (-0.52E+01,-0.18E+01)  ( 0.30E+01, 0.96E+01)  ( 0.14E+02, 0.44E+01)
```

We can observe that the output matrix of `DPALLAUB` is block-anti-triangular. The first block in the lower left corner is of size 2×2, followed by a 1×1 and another 2×2 block.
With `ZPALLAUB` (with OP='H' and OP='T') we get "unblocked" anti-triangular results with now complex entries. The output differs whether we use OP='H' or OP='T'.
Comparing the results of `ZPALLAUB` (with OP='T') and `ZPALLAUB_MIX` (with OP='T') we note that the entries that should be zero are on the order of magnitude of 10^{-15} for `ZPALLAUB` and 10^{-7} for `ZPALLAUB_MIX`. That is caused by the fact that we use mixed precision in the subroutine `ZPALLAUB_MIX`.

5.2 Even Case

In `SIMPLEEXAMPLEEVEN.F` (see 5.2.1) we apply `DSKSLAUB`, `ZSKSLAUB` and `ZSKSLAUB_MIX` to the following pair of matrices (A, B).

$$(A,B) = \left(\begin{bmatrix} 16 & 14 & 12 & 11 & 7 \\ 14 & 0 & 10 & 4 & 5 \\ 12 & 10 & 6 & 18 & 6 \\ 11 & 4 & 18 & 16 & 9 \\ 7 & 5 & 6 & 9 & 16 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 4 & -3 & 3 \\ 0 & 0 & 4 & 4 & 3 \\ -4 & -4 & 0 & -2 & 6 \\ 3 & -4 & 2 & 0 & 5 \\ -3 & -3 & -6 & -5 & 0 \end{bmatrix} \right).$$

We compare the results of `SIMPLEEXAMPLEEVEN.F` at the end of Section 5.2.2.

5.2.1 Program Text

```
      PROGRAM SIMPLEEXAMPLEEVEN
C
C     .. Local Scalars ..
      CHARACTER*1      ORTH
      INTEGER          N
      PARAMETER        (N=5)
      INTEGER          LDWORK,LZWORK,DZERO,INFO,I,J
      PARAMETER        (ORTH='T', LDWORK=3*N**2+11*N+16,
     $                 LZWORK=3*N**2+4*N, DZERO = 0.0E+0)
C     .. Local Arrays ..
      DOUBLE PRECISION A(N,N),B(N,N),A1(N,N),B1(N,N),DU(N,N),
     $                 DWORK(LDWORK),BLOCKSIZEA1(N)
      DOUBLE COMPLEX   A2(N,N),A3(N,N),A4(N,N),B2(N,N),B3(N,N),B4(N,N),
     $                 ZU(N,N),ZWORK(LZWORK)
C     .. External Subroutines ..
      EXTERNAL         DSKSLAUB,ZSKSLAUB,ZSKSLAUB_MIX
C     .. Executable Statements..
C
C
```

```
C     matrix A:
      A(1,1)=16.0
      A(2,1)=14.0
      A(3,1)=12.0
      A(4,1)=11.0
      A(5,1)=7.0
      A(1,2)=14.0
      A(2,2)=0.0
      A(3,2)=10.0
      A(4,2)=4.0
      A(5,2)=5.0
      A(1,3)=12.0
      A(2,3)=10.0
      A(3,3)=6.0
      A(4,3)=18.0
      A(5,3)=6.0
      A(1,4)=11.0
      A(2,4)=4.0
      A(3,4)=18.0
      A(4,4)=16.0
      A(5,4)=9.0
      A(1,5)=7.0
      A(2,5)=5.0
      A(3,5)=6.0
      A(4,5)=9.0
      A(5,5)=16.0
C     matrix B:
      B(1,1)=0.0
      B(2,1)=0.0
      B(3,1)=-4.0
      B(4,1)=3.0
      B(5,1)=-3.0
      B(1,2)=0.0
      B(2,2)=0.0
      B(3,2)=-4.0
      B(4,2)=-4.0
      B(5,2)=-3.0
      B(1,3)=4.0
      B(2,3)=4.0
      B(3,3)=0.0
      B(4,3)=2.0
      B(5,3)=-6.0
      B(1,4)=-3.0
      B(2,4)=4.0
      B(3,4)=-2.0
      B(4,4)=0.0
      B(5,4)=-5.0
      B(1,5)=3.0
      B(2,5)=3.0
      B(3,5)=6.0
      B(4,5)=5.0
      B(5,5)=0.0
C
      DO I=1,N
```

```
         CALL DCOPY(N,A(1,I),1,A1(1,I),1)
         CALL DCOPY(N,B(1,I),1,B1(1,I),1)
      ENDDO
C
      DO I=1,N
         DO J=1,N
            A2(I,J)=CMPLX(A(I,J),DZERO)
            B2(I,J)=CMPLX(B(I,J),DZERO)
         ENDDO
      ENDDO
C
      DO I=1,N
         CALL ZCOPY(N,A2(1,I),1,A3(1,I),1)
         CALL ZCOPY(N,B2(1,I),1,B3(1,I),1)
         CALL ZCOPY(N,A2(1,I),1,A4(1,I),1)
         CALL ZCOPY(N,B2(1,I),1,B4(1,I),1)
      ENDDO

C
C     Apply DSKSLAUB
      CALL DSKSLAUB (ORTH,N,A1,N,B1,N,DU,N,DWORK,LDWORK,INFO)
C     save blocksize
      CALL DCOPY(N,DWORK(3),1,BLOCKSIZEA1(1),1)
C
C     Apply ZSKSLAUB with OP='H'
      CALL ZSKSLAUB ('H',ORTH,N,A2,N,B2,N,ZU,N,ZWORK,LZWORK,DWORK,
     $               LDWORK,INFO)
C
C     Apply ZSKSLAUB with OP='T'
      CALL ZSKSLAUB ('T',ORTH,N,A3,N,B3,N,ZU,N,ZWORK,LZWORK,DWORK,
     $               LDWORK,INFO)
C
C     Apply ZSKSLAUB_MIX with OP='T'
      CALL ZSKSLAUB_MIX('T',N,A4,N,B4,N,ZU,N,ZWORK,LZWORK,DWORK,LDWORK,
     $                  INFO)
C
C     Output
      OPEN(11,FILE = 'simpleexampleeven.OUT',STATUS = 'NEW')
C     Matrix A:
      WRITE(11,*) 'The matrix pair (A,B) to be reduced: '
      WRITE(11,*)
      DO I=1,N
         WRITE(11,666) ((INT(A(I,J))),J=1,N), ((INT(B(I,J))),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying dskslaub:'
      WRITE(11,777) 'blocksize:',(INT(BLOCKSIZEA1(I)),I=1,N)
      WRITE(11,*)
C     Matrix A1:
      WRITE(11,*)'A:'
      DO I=1,N
         WRITE(11,999) ((A1(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
```

```
      WRITE(11,*)'B:'
      DO I=1,N
         WRITE(11,999) ((B1(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zskslaub with op=H:'
C     Matrix A2:
      WRITE(11,*)'A:'
      DO I=1,N
         WRITE(11,888) ((A2(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*)'B:'
      DO I=1,N
         WRITE(11,888) ((B2(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zskslaub with op=T:'
C     Matrix A3:
      WRITE(11,*)'A:'
      DO I=1,N
         WRITE(11,888) ((A3(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*)'B:'
      DO I=1,N
         WRITE(11,888) ((B3(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*) 'result of applying zskslaub_mix with op=T:'
C     Matrix A4:
      WRITE(11,*)'A:'
      DO I=1,N
         WRITE(11,888) ((A4(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
      WRITE(11,*)'B:'
      DO I=1,N
         WRITE(11,888) ((B4(I,J)),J=1,N)
      ENDDO
      WRITE(11,*)
C
999   FORMAT (5(E9.2,2X))
888   FORMAT (5('(',E9.2,',',E9.2,')',2X))
777   FORMAT (A10,2X,5(I1,2X))
666   FORMAT (5(I2,2X),10X,5(I2,2X))
C
C *** Last line of SIMPLEEXAMPLEEVEN ***
      END
```

5.2.2 Results

The following is the content of the file `SIMPLEEXAMPLEEVEN.OUT` (the output file of `SIMPLEEXAMPLEEVEN.F`).

```
 The matrix pair (A,B) to be reduced:
```

```
16  14  12  11   7            0   0   4  -3   3
14   0  10   4   5            0   0   4   4   3
12  10   6  18   6           -4  -4   0  -2   6
11   4  18  16   9            3  -4   2   0   5
 7   5   6   9  16           -3  -3  -6  -5   0

 result of applying dskslaub:
blocksize:  2  1  2  0  0

 A:
 0.14E-14   0.16E-14  -0.18E-14  -0.23E+01   0.20E+02
 0.14E+02   0.45E-14   0.83E-14   0.76E+01   0.14E+02
 0.12E+02   0.10E+02   0.92E+01  -0.13E+01  -0.29E+01
 0.11E+02   0.40E+01   0.18E+02   0.59E+01   0.30E+01
 0.70E+01   0.50E+01   0.60E+01   0.90E+01   0.39E+02

 B:
 0.00E+00   0.00E+00   0.40E+01  -0.30E+01   0.30E+01
-0.14E-14   0.00E+00   0.40E+01   0.40E+01   0.30E+01
 0.65E-15  -0.15E-14   0.00E+00  -0.20E+01   0.60E+01
-0.32E-15  -0.49E+01  -0.24E+01   0.00E+00   0.50E+01
-0.77E+01   0.74E-16  -0.49E+01   0.52E+01   0.00E+00

 result of applying zskslaub with op=H:
 A:
(-0.40E-14, 0.00E+00)  ( 0.52E-15,-0.45E-15)  (-0.77E-15,-0.22E-14)  (-0.40E-15, 0.30E-14)  ( 0.16E+02,-0.16E+01)
( 0.14E+02, 0.00E+00)  ( 0.10E-13, 0.00E+00)  ( 0.52E-14,-0.90E-15)  ( 0.33E+01,-0.11E+02)  ( 0.14E+02, 0.88E+01)
( 0.12E+02, 0.00E+00)  ( 0.10E+02, 0.00E+00)  ( 0.92E+01, 0.00E+00)  (-0.12E+01, 0.15E+01)  (-0.49E+00,-0.25E+01)
( 0.11E+02, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.18E+02, 0.00E+00)  ( 0.10E+02, 0.00E+00)  (-0.28E+01, 0.11E+02)
( 0.70E+01, 0.00E+00)  ( 0.50E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)  ( 0.90E+01, 0.00E+00)  ( 0.35E+02, 0.00E+00)

 B:
( 0.00E+00, 0.14E-14)  ( 0.00E+00, 0.00E+00)  ( 0.40E+01, 0.00E+00)  (-0.30E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
(-0.22E-14, 0.15E-14)  ( 0.00E+00, 0.40E-14)  ( 0.40E+01, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
(-0.16E-14,-0.25E-14)  (-0.16E-14, 0.91E-15)  ( 0.00E+00,-0.39E-15)  (-0.20E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)
( 0.80E-15,-0.20E-15)  ( 0.25E+00,-0.52E+01)  (-0.21E+01,-0.27E+01)  ( 0.00E+00,-0.25E+01)  ( 0.50E+01, 0.00E+00)
(-0.70E+01, 0.17E+01)  (-0.19E+01,-0.28E-01)  (-0.75E+00, 0.42E+01)  (-0.35E+01,-0.30E+01)  ( 0.00E+00, 0.25E+01)

 result of applying zskslaub with op=T:
 A:
( 0.29E-14,-0.63E-14)  ( 0.94E-15, 0.28E-14)  ( 0.92E-15, 0.23E-14)  ( 0.37E-15, 0.37E-16)  ( 0.16E+02, 0.16E+01)
( 0.14E+02, 0.00E+00)  ( 0.41E-14, 0.14E-13)  (-0.41E-14,-0.59E-15)  (-0.33E+01,-0.11E+02)  (-0.14E+02, 0.88E+01)
( 0.12E+02, 0.00E+00)  ( 0.10E+02, 0.00E+00)  ( 0.81E+01,-0.43E+01)  (-0.17E+01,-0.96E+00)  ( 0.47E+00, 0.25E+01)
( 0.11E+02, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.18E+02, 0.00E+00)  ( 0.26E+00, 0.73E+01)  ( 0.56E+01,-0.12E+02)
( 0.70E+01, 0.00E+00)  ( 0.50E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)  ( 0.90E+01, 0.00E+00)  (-0.32E+02,-0.12E+02)

 B:
(-0.55E-16,-0.15E-15)  ( 0.00E+00, 0.00E+00)  ( 0.40E+01, 0.00E+00)  (-0.30E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
(-0.19E-15, 0.11E-14)  (-0.14E-15,-0.42E-15)  ( 0.40E+01, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
(-0.25E-14,-0.19E-14)  (-0.10E-14, 0.29E-15)  ( 0.30E-16, 0.28E-17)  (-0.20E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)
( 0.19E-15,-0.13E-14)  (-0.25E+00, 0.52E+01)  (-0.29E+01,-0.18E+01)  (-0.13E-15,-0.30E-15)  ( 0.50E+01, 0.00E+00)
(-0.70E+01, 0.17E+01)  ( 0.19E+01, 0.28E-01)  ( 0.84E+00, 0.41E+01)  ( 0.51E+01,-0.12E+01)  ( 0.26E-15,-0.99E-16)

 result of applying zskslaub_mix with op=T:
 A:
( 0.64E-07,-0.11E-05)  ( 0.21E-05, 0.14E-05)  (-0.64E-06,-0.21E-07)  ( 0.78E-06,-0.20E-05)  ( 0.16E+02, 0.16E+01)
( 0.14E+02, 0.00E+00)  (-0.31E-06, 0.17E-05)  (-0.11E-05,-0.33E-06)  (-0.33E+01,-0.11E+02)  (-0.13E+02, 0.97E+01)
( 0.12E+02, 0.00E+00)  ( 0.10E+02, 0.00E+00)  ( 0.80E+01,-0.44E+01)  (-0.17E+01,-0.10E+01)  ( 0.57E+00, 0.25E+01)
( 0.11E+02, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.18E+02, 0.00E+00)  (-0.25E+00, 0.73E+01)  ( 0.56E+01,-0.12E+02)
( 0.70E+01, 0.00E+00)  ( 0.50E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)  ( 0.90E+01, 0.00E+00)  (-0.33E+02,-0.95E+01)

 B:
( 0.00E+00, 0.00E+00)  ( 0.00E+00, 0.00E+00)  ( 0.40E+01, 0.00E+00)  (-0.30E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
( 0.14E-06, 0.61E-06)  ( 0.00E+00, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.40E+01, 0.00E+00)  ( 0.30E+01, 0.00E+00)
(-0.64E-06, 0.10E-06)  ( 0.27E-06, 0.12E-06)  ( 0.00E+00, 0.00E+00)  (-0.20E+01, 0.00E+00)  ( 0.60E+01, 0.00E+00)
( 0.79E-07, 0.47E-06)  (-0.25E+00, 0.52E+01)  (-0.29E+01,-0.19E+01)  ( 0.00E+00, 0.00E+00)  ( 0.50E+01, 0.00E+00)
```

```
(-0.70E+01, 0.17E+01)  ( 0.19E+01,-0.11E+00)  ( 0.10E+01, 0.41E+01)  ( 0.51E+01,-0.12E+01)  ( 0.00E+00, 0.00E+00)
```

Here we can make analogous observations as in the palindromic case (see Section 5.1.2) but, moreover, we note the special structure of the output matrices. Instead of outputting the entire anti-triangular matrices we only change the upper triangle of A and the lower triangle of B. The other parts remain untouched.

References

[1] Alan J. Laub. A Schur method for solving algebraic Riccati equations. IEEE Trans. Automat. Control., vol. 24, 1979, no. 6, pp. 913–921. (See also *Proc. 1978 CDC (Jan. 1979)*, pp. 60-65)

[2] D. Steven Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. Numerical methods for palindromic eigenvalue problems: Computing the anti-triangular Schur form. Numerical Linear Algebra with Applications, Vol 16, pp.63–86, 2009.

[3] Christian Schröder. Palindromic and even eigenvalue problems - Analysis and Numerical Methods. Dissertation. TU Berlin, Str. des 17. Juni 136, D-10623 Berlin, Germany, 2008.

[4] SLICOT Implementation and Documentation Standards WGS Report 96-1, August 1996. `www.slicot.org`