MATLAB TOOLS FOR SOLVING PERIODIC EIGENVALUE PROBLEMS [1]

Robert Granat * **Bo Kågström** * **Daniel Kressner** *,[2]

* *Department of Computing Science and HPC2N, Umeå University, SE-90187 Umeå, Sweden.*
{granat,bokg,kressner}@cs.umu.se

Abstract: Software for computing eigenvalues and invariant subspaces of general matrix products is proposed. The implemented algorithms are based on orthogonal transformations of the original data and thus attain numerical backward stability, which enables good accuracy even for small eigenvalues. The prospective toolbox combines the efficiency and robustness of library-style Fortran subroutines based on state-of-the-art algorithms with the convenience of MATLAB interfaces. It will be demonstrated that this toolbox can be used to address a number of tasks in systems and control theory, such as the solution of periodic discrete-time algebraic Riccati equations.

Keywords: linear periodic system, matrix product, periodic eigenvalue problem, periodic Schur form, library software, MATLAB tools.

1. INTRODUCTION

Let us consider a *linear discrete-time system* of the form

$$\begin{aligned} E_k x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{aligned} \tag{1}$$

with state, input and output vectors x_k, u_k, and y_k, respectively. The coefficient matrices A_k, B_k, C_k, D_k, E_k are supposed to be of matching dimensions. Moreover, we also assume that (1) is *periodic* for some period $p \geq 1$, i.e., $A_{k+p} = A_k, B_{k+p} = B_k, C_{k+p} = C_k, D_{k+p} = D_k, E_{k+p} = E_k$ for all integers k. Computational tasks for such periodic systems can often be addressed by solving *periodic (or product) eigenvalue problems*, see (Varga and Van Dooren 2001) for an overview. For example, if all E_k are square and invertible then (1) is *asymptotically stable* if and only if all eigenvalues of the monodromy matrix

$$\Pi = E_p^{-1} A_p E_{p-1}^{-1} A_{p-1} \cdots E_1^{-1} A_1 \tag{2}$$

lie strictly inside the unit circle.

Theoretically, the eigenvalues of (2) can be computed by first forming the product Π explicitly and then applying any general-purpose eigenvalue solver. In finite-precision arithmetic, however, this approach may lead to potentially disastrous numerical inaccuracies. For example, if any of the matrices E_k is nearly singular then its inversion must be avoided whenever possible; a rule that has driven the development of the QZ algorithm for the case $p = 1$ (Moler and Stewart 1973). Even if all matrices E_k are well-conditioned, smaller eigenvalues of Π are severely affected by the lack of numerical backward stability in matrix multiplication (Higham 2002) and cannot be computed by such an approach.

[1] Supported by the Swedish Foundation for Strategic Research under the Frame Programme Grant A3 02:128.
[2] This author was additionally supported by a DFG Emmy Noether fellowship.

There are several alternatives that take the product structure of Π into account and avoid some or all of the numerical disadvantages mentioned above.

Lifting (Flamm 1991) is a well-known technique to transform a periodic system (1) into a time-invariant none. Applied to (2), this leads to a block cyclic matrix pencil

$$\mathcal{A} - \lambda\mathcal{E} = \begin{bmatrix} 0 & & & A_p \\ A_1 & \ddots & & \\ & \ddots & \ddots & \\ & & A_{p-1} & 0 \end{bmatrix} - \lambda \begin{bmatrix} E_p & & & \\ & E_1 & & \\ & & \ddots & \\ & & & E_{p-1} \end{bmatrix}. \quad (3)$$

The eigenvalues of $\mathcal{A} - \lambda\mathcal{E}$ are the p^{th} roots of the eigenvalues of Π. While this embedding enables the application of the standard QZ algorithm to attain good eigenvalue accuracy, the numerical identification of the p^{th} roots is problematic and the computational cost increases cubically with p.

Collapsing (Benner and Byers 2001, Van Dooren 1999) is a technique that reduces the long product (2) to the case $p = 1$ without inverting any of the matrices E_k. It can equivalently be viewed as an orthogonal equivalence transformation of a block cyclic matrix pencil closely related to (3). The computational cost of this approach increases linearly with p and numerical inaccuracies due to nearly singular E_k are avoided. On the other hand, collapsing does not lead to an overall numerically backward stable method for solving periodic eigenvalue problems. This has been demonstrated in (Granat *et al.* 2006*b*) for computing eigenvectors belonging to smaller eigenvalues.

The periodic QZ algorithm (Bojanczyk *et al.* 1992, Hench and Laub 1994) orthogonally transforms the matrices E_k and A_k into upper (quasi-)triangular form. It achieves numerical backward stability and has a computational cost that increases linearly with p. All publicly available implementations of the periodic QZ algorithm currently support only the case when each E_k is an identity matrix. In this case, the periodic QZ reduces to the periodic QR algorithm, see also (Kressner 2006).

Because of its numerical advantages, the implementations described in this paper are based on the periodic QZ algorithm and variants thereof. We are developing a Fortran library that admits non-trivial factors E_k; in fact, much more general matrix products are supported, see Section 2. The purpose of this paper is to describe prospective MATLAB functions that are based on MEX interfaces to these Fortran routines. In Section 3, we describe the functionality of the toolbox on a level that is suitable for developers and expert users. In Section 4, a MATLAB class `product` is introduced that allows the convenient use of these tools without knowledge of the internal algorithms or data structures.

Besides periodic systems there are a variety of other applications requiring the solution of product eigenvalue problems. For example, the structure-preserving solution of time-invariant linear-quadratic (LQ) optimal control problems gives rise to matrix products with up to five factors (Benner *et al.* 2002*a*).

2. NOTATION

In the scope of this paper, we consider product eigenvalue problems in their most general form:

$$A_p^{s_p} A_{p-1}^{s_{p-1}} \cdots A_1^{s_1}, \quad (4)$$

with the indices $s_1, \ldots, s_p \in \{1, -1\}$. The dimensions of $A_1, \ldots, A_p$ should match, i.e.,

$$A_k \in \begin{cases} \mathbb{R}^{n_{k+1} \times n_k} & \text{if } s_k = 1, \\ \mathbb{R}^{n_k \times n_{k+1}} & \text{if } s_k = -1, \end{cases}$$

where $n_{p+1} = n_1$ (to simplify the notation we will *identify* $p+1$ *with* 1 in the following). Additionally, we require the condition

$$\sum_{\substack{k=1 \\ s_k=1}}^{p} n_k + \sum_{\substack{k=1 \\ s_k=-1}}^{p} n_{k+1} = \sum_{\substack{k=1 \\ s_k=-1}}^{p} n_k + \sum_{\substack{k=1 \\ s_k=1}}^{p} n_{k+1}, \quad (5)$$

which ensures that the corresponding lifted pencil (3) is square.

The level of generality imposed by (4) allows to cover the other applications mentioned above that do not necessarily lead to a matrix product having the somewhat simpler form (2). It is worth mentioning that we admit matrices A_k corresponding to an index $s_k = -1$ to be singular or even rectangular, in which case the matrix product (4) should only be understood in a formal sense.

2.1 Periodic Schur decomposition

To define the eigenvalues associated with a formal matrix product, one could consider the canonical form of (4), see (Sergeichuk 2004) for an overview, or an appropriate generalization of the lifted representation (3). A more computationally oriented definition can be derived from the *periodic Schur decomposition*.

Theorem 1. There are orthogonal matrices $Q_1 \in \mathbb{R}^{n_1 \times n_1}, \ldots, Q_p \in \mathbb{R}^{n_p \times n_p}$ *such that each*

$$T_k := \begin{cases} Q_{k+1}^T A_k Q_k & \text{if } s_k = 1, \\ Q_k^T A_k Q_{k+1} & \text{if } s_k = -1. \end{cases} \quad (6)$$

takes the form

$$T_k = \begin{bmatrix} T_{11,k} & T_{12,k} & T_{13,k} \\ 0 & T_{22,k} & T_{23,k} \\ 0 & 0 & T_{33,k} \end{bmatrix}, \qquad (7)$$

where all $T_{22,k}$ are square and of the same order $n_c \le n$. Moreover, $T_{22,k}$ is upper quasi-triangular for one index $k = l$ and upper triangular for every other k. The formal matrix products $T_{11,p}^{s_p} \cdots T_{11,1}^{s_1}$ and $T_{33,p}^{s_p} \cdots T_{33,1}^{s_1}$ contain the dimension-induced infinite and zero eigenvalues, respectively.

A mathematically precise definition of dimension-induced infinite and zero eigenvalues is beyond the scope of this paper, see, e.g., (Sergeichuk 2004) for more details. The n_c eigenvalues of $T_{22,p}^{s_p} \cdots T_{22,1}^{s_1}$ are called *core eigenvalues* and can be easily computed by multiplying the diagonal elements of $T_{22,1}^{s_1}, \ldots, T_{22,p}^{s_p}$ explicitly.[3] Core infinite eigenvalues may exist if at least one $s_k = -1$ and $T_{22,k}$ is singular; otherwise all core eigenvalues are finite.

Let us illustrate the implication of Theorem 1 for the important special case when the inverted factors are square, i.e., $n_k = n_{k+1}$ whenever $s_k = -1$. Then the blocks $T_{11,k}$ in (7) are void, and

$$T_{33,p}^{s_p} \cdots T_{33,1}^{s_1} = 0 \in \mathbb{R}^{(n_1 - n_c) \times (n_1 - n_c)}$$

with $n_c = \min\{n_1, n_2, \ldots, n_p\}$. Note that n_c might be smaller than the minimum of all dimensions in the general case.

2.2 Deflating subspaces and eigenvectors

A sequence of subspaces $\mathcal{X}_1 \subseteq \mathbb{C}^{n_1}, \mathcal{X}_2 \subseteq \mathbb{C}^{n_2}, \ldots, \mathcal{X}_p \subseteq \mathbb{C}^{n_p}$ having the same dimension (e.g., $\dim(\mathcal{X}_k) = m$) is called a *periodic deflating subspace* if

$$\begin{aligned} A_k \mathcal{X}_k &\subset \mathcal{X}_{k+1}, \quad \text{if } s_k = 1, \\ A_k \mathcal{X}_{k+1} &\subset \mathcal{X}_k, \quad \text{if } s_k = -1. \end{aligned}$$

If $n_k = n_{k+1}$ whenever $s_k = -1$, the periodic Schur decomposition (6)–(7) is easily seen to imply that the first m columns of $Q_1, \ldots, Q_p$ span a periodic deflating subspace for every $m \le n_c$ such that the $(m+1, m)$ entry of $T_{22,l}$ is nonzero. It belongs to the core eigenvalues that reside in the leading m diagonal elements of $T_{22,1}^{s_1}, \ldots, T_{22,p}^{s_p}$. Periodic deflating subspaces belonging to other eigenvalues can be computed by reordering the periodic Schur decomposition (Granat *et al.* 2006*a*), see also Section 3.2.4.

[3] The numerical treatment of 2×2 blocks in the upper quasi-triangular matrix $T_{22,l}$ requires some attention (Van Loan 1973).

A *periodic eigenvector* can be defined as a sequence of nonzero vectors $x_1 \in \mathbb{C}^{n_1}, x_2 \in \mathbb{C}^{n_2}, \ldots, x_p \in \mathbb{C}^{n_p}$ that spans a periodic deflating subspace, with $\dim(\mathrm{span}(x_k)) = 1$ for all k. In the case $n_k = n_{k+1}$ whenever $s_k = -1$, an eigenvector can be computed by reordering the corresponding eigenvalue to the top-left corner of the product $T_{22,p}^{s_p} \cdots T_{22,1}^{s_1}$. Alternatively, it can be extracted from the null space of the correspondingly shifted block cyclic embedding, see also (Moravitz Martin and Van Loan 2002).

3. ALGORITHMS AND CORE FUNCTIONALITY

The computational approach in our toolbox is based on the periodic Schur decomposition (6)–(7).

3.1 Data structures

In MATLAB, we store matrices in a *cell array*. A cell array is a general purpose matrix. Each of the elements can contain data of a different type, size and dimension. For cell arrays, storage is allocated dynamically. In particular, for the matrices in (4), we may use a code snippet as

```
A = cell(1,p);
for k=1:p, A{k} = A(:,:,k);, end
```

to create the cell array `A` and store the matrices $A_1, A_2, \ldots, A_p$ in the array.

In contrast to ordinary three-dimensional arrays, this storage scheme does not waste a lot of memory for sequences with strong variation in the matrix dimensions or where the order of the matrices should be changed on runtime or the period p may be increased or decreased.

In our Fortran software, the data structure for storing the matrix sequences is an array of Fortran 90 *pointers* to ordinary two-dimensional data arrays which has the same storage saving benefits as the cell arrays.

The signatures $s_1, \ldots, s_p \in \{1, -1\}$ are provided in a one-dimensional array s of length p. Alternatively, the user can provide text strings `s = 'ones'` and `s = 'alter'` to denote the two commonly encountered situations $s_k \equiv 1$ and $s_k = (-1)^{k-1}$, respectively. By default, `s = 'ones'`.

3.2 Functions with arguments and results

In the following, we present MATLAB interfaces to the core functions in our toolbox. As usual, a function call provides input to the function by passing the data in an argument list. Data that

needs to be returned to the caller is passed back in a list of return values. Typically, the input and output data are cell arrays with signatures. Braces { ... } in the function definitions below identify optional arguments as well as result values. We remark that the transformed cell array `T` of an argument `A` has the same signature as `A`.

3.2.1. Reduction to a square product

`[{Q,}T] = per_square(A{,s})`

Deflates dimension-induced zero and infinite eigenvalues from a general matrix product (4). An orthogonal decomposition of the form (6)–(7) is computed with the square matrices $T_{22,1}, \ldots, T_{22,p}$ in unreduced form. The orthogonal transformation matrices and the transformed matrices are contained in the cell arrays `Q` and `T`, respectively.

3.2.2. Reduction to periodic HT form

`[{Q,}T] = per_hess(A{,s,l})`

Reduces a general matrix product (4) to periodic Hessenberg-triangular (HT) form. An orthogonal decomposition of the form (6)–(7) is computed, where $T_{22,l}$ is upper Hessenberg and $T_{22,k}$ is upper triangular for all $k \neq l$. By default, `l = 1`.

3.2.3. Reduction to periodic Schur form

`[{Q,}T] = per_schur(A{,s,l})`

Reduces a general matrix product (4) to periodic Schur form. An orthogonal decomposition of the form (6)–(7) is computed, where $T_{22,l}$ is upper quasi-triangular and $T_{22,k}$ is upper triangular for all $k \neq l$. By default, `l = 1`.

3.2.4. Reordering eigenvalues

`[{Q,}T] = per_ordqz({Q,}T{,s,l}, slct)`

Reorders eigenvalues so that a selected cluster of eigenvalues appears in the leading (top-left) diagonal blocks of $T_{22,1}, \ldots, T_{22,p}$ in the periodic Schur form `T`. This is done by orthogonal transformations (see below) and the associated periodic deflating subspace is spanned by the corresponding leading columns of `Q`. The logical vector `slct` specifies the selected cluster as they appear along the diagonals of the input $T_{22,1}, \ldots, T_{22,p}$. `slct` can also be a string variable that defines a region of the complex plane, e.g., `slct = 'udi'` corresponds to all eigenvalues inside the unit disk.

Periodic eigenvalue reordering is performed by generalizations of the LAPACK methodology (Bai and Demmel 1993, Kågström and Poromaa 1996) to general products of matrices of the form (4). First, the product is recasted on the exponent structure (2) by insertion of identity matrices. Then, we solve periodic matrix equations and apply sequences of orthogonal transformations to $T_{22,1}, \ldots, T_{22,p}$ (Granat and Kågström 2006, Granat *et al.* 2006*a*). One important step in these backward stable reordering methods is the computation of a solution to a *Blocked Almost Block Diagonal* (BABD) linear system to high accuracy using structured variants of Gaussian elimination with partial pivoting (GEPP) or QR factorization.

Currently we do not plan to support reordering of non-core eigenvalues since we are not aware of any applications calling for such an operation.

3.2.5. Computation of eigenvalues only

`e = per_eig(A{,s,'factor'})`

Returns a one-dimensional array `e` of length n_1 that contains the eigenvalues of a general matrix product (4). This contains all $n_c \leq n_1$ core eigenvalues and, possibly, dimension-induced zero and infinite eigenvalues. If the optional input argument `'factor'` is provided, the eigenvalues are returned in factorized form, i.e., an $n_1 \times p$ array `e` is returned whose row products give the eigenvalues (using `prod(e,2)`). This allows, e.g., computing the logarithms of eigenvalue magnitudes that would otherwise over- or underflow: `sum(log(abs(e)),2)`.

3.3 Complex matrices

If any of the matrices in the cell array `A` happens to be complex, the MATLAB function listed above return complex variants of the periodic decompositions: Q becomes unitary, the transpose in (6) becomes the complex conjugate transpose, and the quasi-triangular matrix $T_{l,22}$ becomes triangular.

3.4 Additional functionality

Additional functionality and tools will be provided, including the solution of various periodic Sylvester-type matrix equations (Varga and Van Dooren 2001, Granat *et al.* 2006*c*). The demand of such solvers appear in different control applications but are also useful for condition estimation and error bounds of computed quantities (eigenvalues, and periodic deflating subspaces). Relevant perturbation theory for product eigenvalue problems are presented in (Lin and Sun 2001, Benner *et al.* 2002*b*, Sun 2005).

4. THE MATRIX PRODUCT CLASS

MATLAB classes provide a convenient way to hide the technical complexity imposed by the generality of periodic eigenvalue problems. We have

designed a class `product` that contains the cell and signature arrays `A`, `s`, and provides several operations for creating and manipulating objects.

4.1 Construction of a `product` *object*

The simplest way to construct a `product` object is to type `p = product(A1)`, where `A1` is a two-dimensional array. This corresponds to a product of length 1: $\Pi = A_1$. Products of longer lengths can be created using the overloaded operations `*` and `inv`. For example,

```
p = inv(product(A2)) * product(A1)
```

corresponds to a (formal) product of length 2: $\Pi = A_2^{-1}A_1$. It is important to note that the inversion and multiplication are never carried out explicitly for product objects. In particular, this admits formal inverses of rectangular matrices.

4.2 Main computational functions

```
[{Q,} pT] = square({Q,} p)
[{Q,} pT] = hess({Q,} p)
[{Q,} pT] = schur({Q,} p)
[{Q,} pT] = ordqz({Q,} p, slct)
```

The functions listed above are `product` class equivalents of the MATLAB functions described in Sections 3.2.1–3.2.4. `Q` is a cell array containing the orthogonal transformation matrices and `T` is a `product` object containing the transformed product $T_p^{s_p}T_{p-1}^{s_{p-1}}\cdots T_1^{s_1}$. The function `e = eig(p{,'factor'})` returns the eigenvalues of the product corresponding to `p` as described in Section 3.2.5.

4.3 Utilities

In the following, we assume that `p` is a product object corresponding to the matrix product $A_p^{s_p}A_{p-1}^{s_{p-1}}\cdots A_1^{s_1}$.

`p'` transposes the product.

`p1 + p2` adds the coefficients of two `product` objects having the same dimensions.

`p * X` for a cell array `X` computes a `product` with new transformed coefficients A_kX_k (if $s_k = 1$) and X_kA_k (if $s_k = -1$).

`X * p` for a cell array `X` computes a `product` with new transformed coefficients $X_{k+1}A_k$ (if $s_k = 1$) and A_kX_{k+1} (if $s_k = -1$).

`lift(p)` returns a lifted, block cyclic representation of the form (3).

`norm(p, nrm)` returns a vector containing the norms of the coefficients. Which norm is controlled by `nrm`, e.g., `nrm = 'fro'` corresponds to the Frobenius norm.

`prod(p)` explicitly forms the matrix product if possible.

`dim(p)` returns a vector of length $p+1$ containing the matrix dimensions $n_1, n_2, ..., n_{p+1}$.

`signature(p)` returns a vector of length p containing the matrix signatures $s_1, s_2, ..., s_p$ with $s_k \in \{-1, +1\}$.

5. CASE STUDY: SOLUTION OF PERIODIC RICCATI EQUATIONS

The unique symmetric positive semidefinite solution $X_k = X_{k+p}$ of the *discrete-time periodic Riccati equation* (DPRE) (Bittanti *et al.* 1991, Hench and Laub 1994)

$$\begin{aligned} 0 = {} & C_k^TH_kC_k - E_{k-1}^TX_kE_{k-1} + A_k^TX_{k+1}A_k \\ & -A_k^TX_{k+1}B_k(N_k + B_k^TX_{k+1}B_k)^{-1}B_k^TX_{k+1}A_k, \end{aligned} \tag{8}$$

can be computed robustly in terms of an associated periodic deflating subspace problem. The DPRE (8) appears in the LQ optimal control problem associated with the system (1) where all E_k are nonsingular, and the weighting matrices of the cost functional are p-periodic, i.e., $H_{k+p} = H_k (= H_k^T \geq 0)$ and $N_{k+p} = N_k (= N_k^T > 0)$.

Similarly as for the case $E_k = I_n$ (Hench and Laub 1994), it can be shown that the $2n \times 2n$ matrix product

$$M_p^{-1}L_pM_{p-1}^{-1}L_{p-1}\cdots M_1^{-1}L_1 \tag{9}$$

with

$$L_k = \begin{bmatrix} A_k & 0 \\ -C_k^TH_kC_k & E_{k-1}^T \end{bmatrix}, M_k = \begin{bmatrix} E_{k-1}B_kN_k^{-1}B_k^T & \\ 0 & A_k^T \end{bmatrix}$$

has exactly n eigenvalues inside the unit disk under the reasonable assumption that (1) is *d-stabilizable* and *d-detectable*. By reordering the periodic Schur form of the matrix product (9) we can compute a periodic deflating subspace defined by the orthogonal matrices $U_k, V_k \in \mathbb{R}^{2n\times 2n}$ with $V_{k+p} = V_k$ such that

$$U_k^TL_kV_k = \begin{bmatrix} S_{11}^{(k)} & S_{12}^{(k)} \\ 0 & S_{22}^{(k)} \end{bmatrix}, \; U_k^TM_kV_{k+1} = \begin{bmatrix} T_{11}^{(k)} & T_{12}^{(k)} \\ 0 & T_{22}^{(k)} \end{bmatrix},$$

where the $n \times n$ matrix product

$$T_{11}^{(p)^{-1}}S_{11}^{(p)}\cdots T_{11}^{(2)^{-1}}S_{11}^{(2)}T_{11}^{(1)^{-1}}S_{11}^{(1)}$$

contains all eigenvalues inside the unit disk. If we partition

$$U_k = \begin{bmatrix} U_{11}^{(k)} & U_{12}^{(k)} \\ U_{21}^{(k)} & U_{22}^{(k)} \end{bmatrix}$$

with $U_{ij}^{(k)} \in \mathbb{R}^{n\times n}$, then

$$U_{21}^{(k)} \left[U_{11}^{(k)}\right]^{-1} = X_k E_{k-1},$$

from which X_k can be computed.

Using our MATLAB tools and assuming that we have defined cell arrays `L` and `M` corresponding to L_k and M_k above, the solution sequence X_k for $k = 1, \ldots, p$ can be computed as:

```
for k = 1:p,
  A{2*k-1} = L{k}; A{2*k} = M{k};
end
pA = product((-1).^[1:2*p], A);

[Q, pT] = schur(pA);
[Q, pT] = ordqz(Q, pT, 'udi');

X = cell(1,p);
for k = 1:p,
  X{k} = Q{2*k-1}(n+1:2*n,1:n) / ...
      Q{2*k-1}(1:n,1:n) / L{k}(1:n,1:n);
end
```

REFERENCES

Bai, Z. and J. W. Demmel (1993). On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.* **186**, 73–95.

Benner, P. and R. Byers (2001). Evaluating products of matrix pencils and collapsing matrix products. *Numerical Linear Algebra with Applications* **8**, 357–380.

Benner, P., R. Byers, V. Mehrmann and H. Xu (2002*a*). Numerical computation of deflating subspaces of skew-Hamiltonian/Hamiltonian pencils. *SIAM J. Matrix Anal. Appl.* **24**(1), 165–190.

Benner, P., V. Mehrmann and H. Xu (2002*b*). Perturbation analysis for the eigenvalue problem of a formal product of matrices. *BIT* **42**(1), 1–43.

Bittanti, S., P. Colaneri and G. De Nicolao (1991). The periodic Riccati equation. In: *The Riccati Equation* (S. Bittanti, A. J. Laub and J. C. Willems, Eds.). pp. 127–162. Springer-Verlag. Berlin, Heidelberg, Germany.

Bojanczyk, A., G. H. Golub and P. Van Dooren (1992). The periodic Schur decomposition; algorithm and applications. In: *Proc. SPIE Conference*. Vol. 1770. pp. 31–42.

Flamm, D. S. (1991). A new shift-invariant representation for periodic linear systems. *Systems Control Lett.* **17**(1), 9–14.

Granat, R. and B. Kågström (2006). Direct Eigenvalue Reordering in a Product of Matrices in Periodic Schur Form. *SIAM J. Matrix Anal. Appl.* **28**(1), 285–300.

Granat, R., B. Kågström and D. Kressner (2006*a*). Computing Periodic Deflating Subspaces Associated with a Specified Set of Eigenvalues. *BIT Numerical Mathematics* (submitted).

Granat, R., B. Kågström and D. Kressner (2006*b*). Reordering the Eigenvalues of a Periodic Matrix Pair with Applications in Control. In: *Proc. of 2006 IEEE Conference on Computer Aided Control Systems Design (CACSD)*. pp. 25–30. ISBN:0-7803-9797-5.

Granat, R., I. Jonsson and B. Kågström (2006*c*). Recursive Blocked Algorithms for Solving Periodic Triangular Sylvester-type Matrix Equations. In: *Proc. of PARA'06: State-of-the-art in Scientific and Parallel Computing*. LNCS, Springer (to appear).

Hench, J. J. and A. J. Laub (1994). Numerical solution of the discrete-time periodic Riccati equation. *IEEE Trans. Automat. Control* **39**(6), 1197–1210.

Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms*. second ed.. SIAM. Philadelphia, PA.

Kågström, B. and P. Poromaa (1996). Computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation: theory, algorithms and software. *Numer. Algorithms* **12**(3-4), 369–407.

Kressner, D. (2006). The periodic QR algorithm is a disguised QR algorithm. *Linear Algebra Appl.*, **417**(2-3):423–433.

Lin, W.-W. and J.-G. Sun (2001). Perturbation analysis for the eigenproblem of periodic matrix pairs. *Linear Algebra Appl.* **337**, 157–187.

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

Moravitz Martin, C. D. and Van Loan, C. F. (2002). Product triangular systems with shift. *SIAM J. Matrix Anal. Appl.* **24**, 292–301.

Sergeichuk, V. V. (2004). Computation of canonical matrices for chains and cycles of linear mappings. *Linear Algebra Appl.* **376**, 235–263.

Sun, J.-G. (2005). Perturbation bounds for subspaces associated with periodic eigenproblems. *Taiwanese J. of Mathematics* **9**(1), 17–38.

Van Dooren, P. (1999). Two point boundary value and periodic eigenvalue problems. In: *Proceedings IEEE CACSD conference*.

Van Loan, C. F. (1973). Generalized Singular Values with Algorithms and Applications. PhD thesis. The University of Michigan.

Varga, A. and P. Van Dooren (2001). Computational methods for periodic systems - an overview. In: *Proc. of IFAC Workshop on Periodic Control Systems, Como, Italy*. pp. 171–176.