Parallel Algorithms for Triangular Periodic Sylvester-Type Matrix Equations

Per Andersson, Robert Granat, Isak Jonsson, and Bo Kågström

Department of Computing Science and HPC2N, Umeå University,
SE-901 87 Umeå, Sweden
{c02apr,granat,isak,bokg}@cs.umu.se

Abstract. We present parallel algorithms for triangular periodic Sylvester-type matrix equations, conceptually being the third step of a periodic Bartels–Stewart-like solution method for general periodic Sylvester-type matrix equations based on variants of the periodic Schur decomposition. The presented algorithms are designed and implemented in the framework of the recently developed HPC library SCASY and are based on explicit blocking, 2-dimensional block cyclic data distribution and a wavefront-like traversal of the right hand side matrices. High performance is obtained by rich usage of level 3 BLAS operations. It is also demonstrated how several important key concepts of SCASY regarding communications and the treatment of quasi-triangular coefficient matrices are generalized to the periodic case. Some experimental results from a distributed memory Linux cluster demonstrate are also presented.

Keywords: Periodic Sylvester-type matrix equations, Bartels–Stewart's method, explicit blocking, level-3 BLAS, ScaLAPACK, condition estimation, RECSY, SCASY.

1 Introduction

Consider the periodic continuous-time Sylvester (PSYCT) matrix equation

$$A_k X_k - X_{k+1} B_k = C_k,\ \ k = 0, 1, \ldots, P-1, \tag{1}$$

where $A_k \in \mathbb{R}^{m\times m}$, $B_k \in \mathbb{R}^{n\times n}$ and $C_k, X_k \in \mathbb{R}^{m\times n}$ are P-cyclic general matrices with real entries. A P-cyclic matrix is characterized by that it repeats itself in a sequence of matrices every Pth time, e.g., $A_P = A_0$, $A_{P+1} = A_1$, etc. Matrix equations of the form (1) have applications in, e.g., computation and condition estimation of *periodic invariant subspaces* of square matrix products of the form

$$\mathcal{A}_{P-1} \cdots \mathcal{A}_1 \mathcal{A}_0 \in \mathbb{R}^{l\times l}, \tag{2}$$

and in periodic systems design and analysis (see, e.g., [28] and the references therein). Matrix products of the form (2) are conceptually studied via the *periodic real Schur form* (PRSF): there exists an orthogonal P-cyclic matrix sequence $\mathcal{Z}_k \in \mathbb{R}^{l\times l}$ such that the sequence

$$\mathcal{Z}_{k+1}^T \mathcal{A}_k \mathcal{Z}_k = \mathcal{T}_k,\ \ k = 0, 1, \ldots, P-1, \tag{3}$$

consists of $P-1$ upper triangular matrices and one upper quasi-triangular matrix. The products of conforming 1×1 and 2×2 diagonal blocks of the matrix sequence $\mathcal{T}_k$ contain the real and complex conjugate pairs of eigenvalues of the matrix product (2). Similar to the standard case ($P = 1$, e.g., see [6]), the PRSF is computed by means of a reduction to periodic Hessenberg form followed by applying a periodic QR algorithm to the resulting sequence [4,15]. The PRSF is an important tool in several applications, including solving periodic Sylvester-type and Riccati matrix equations, see, e.g., [15,27,28].

Periodic matrix equations of the forms (1) is a special case of the periodic Sylvester-like (PSLE) equation

$$\begin{cases} A_k X_k - X_{k+1} B_k = C_k, & \text{for} \quad s_k = 1, \\ A_k X_{k+1} - X_k B_k = C_k, & \text{for} \quad s_k = -1, \end{cases} \tag{4}$$

arising in computing periodic eigenspaces of matrix products of the form

$$\mathcal{A}_{P-1}^{s_{P-1}} \cdots \mathcal{A}_1^{s_1} \mathcal{A}_0^{s_0}, \tag{5}$$

where $s_k \in \{0, 1\}$, i.e., matrix products with arbitrary order of the ± 1 exponents, see [13] for details. Another special case of (4) is the periodic continuous-time *generalized coupled* Sylvester (PGCSY) equation

$$\begin{cases} A_k X_k - Y_k B_k = C_k, \\ D_k X_{k+1} - Y_k E_k = F_k, \end{cases} \tag{6}$$

which is considered when computing periodic deflating subspaces of matrix products of the form

$$\mathcal{E}_{P-1}^{-1} \mathcal{A}_{P-1} \cdots \mathcal{E}_1^{-1} \mathcal{A}_1 \mathcal{E}_0^{-1} \mathcal{A}_0, \tag{7}$$

which are conducted via variants of the *generalized* periodic Schur decomposition [4,15]. We refer to [12] for details.

Equation (1) has a unique solution if the matrix products $A_{P-1} \cdots A_1 A_0$ and $B_{P-1} \cdots B_1 B_0$ have no eigenvalues in common. We solve it via periodic variants of Bartels–Stewart's method [2]:

1. Transform the matrix sequences A_k and B_k to PRSFs:
$$T_A^{(k)} = Q_{k+1}^T A_k Q_k, \tag{8}$$
$$T_B^{(k)} = U_{k+1}^T B_k U_k, \tag{9}$$
where Q_k and U_k are P-cyclic orthogonal matrices and $T_A^{(k)}$ and $T_B^{(k)}$ are the periodic real Schur forms, $k = 0, 1, \ldots, P-1$, with quasi-triangular factors $T_A^{(r)}$ and $T_B^{(s)}$, $0 \le r, s \le P-1$.
2. Update the matrix sequence C_k with respect to the two periodic Schur decompositions:
$$\tilde{C}_k = Q_{k+1}^T C_k U_k, \; k = 0, 1, \ldots, P-1. \tag{10}$$

3. Solve the reduced triangular periodic matrix equation:

$$T_A^{(k)}\tilde{X}_k - \tilde{X}_{k+1}T_B^{(k)} = \tilde{C}_k,\ k = 0, 1, \ldots, P-1. \tag{11}$$

4. Transform the sequence $\tilde{X}_k$ back to the original coordinate system:

$$X_k = Q_{k+1}\tilde{X}_k U_k^T,\ k = 0, 1, \ldots, P-1. \tag{12}$$

In step 1, reliable and efficient software for computing the periodic Schur decomposition should be used. Several attempts of implementing such software have been conducted (see, e.g., [15,21,23] and the PEP software library [13]) and the state-of-the-art implementation is implemented in the SLICOT routines `MBVH03` (periodic Hessenberg reduction) and `MBWS03` (periodic QR iterations). To the best of our knowledge, no parallel implementation exists today.

Steps 2 and 4 above are performed as two series of P *two-sided* matrix-matrix multiplication updates by P pairs of GEMM-operations [22].

In the rest of the paper, we focus on step 3. The reduced triangular problem (11) can be solved via a linear system representation of the corresponding periodic Sylvester operator (see, e.g., [8]):

$$Z_{\mathrm{PSYCT}}\tilde{x} = \tilde{c}, \tag{13}$$

where

$$Z_{\mathrm{PSYCT}} = \begin{bmatrix} -T_B^{(P-1)^T} \otimes I_m & & & I_n \otimes T_A^{(P-1)} \\ I_n \otimes T_A^{(0)} & -T_B^{(0)^T} \otimes I_m & & \\ & \ddots & \ddots & \\ & & I_n \otimes T_A^{(P-2)} & -T_B^{(P-2)^T} \otimes I_m \end{bmatrix} \tag{14}$$

and

$$\tilde{x} = \begin{bmatrix} \mathrm{vec}(\tilde{X}_0) \\ \mathrm{vec}(\tilde{X}_1) \\ \cdots \\ \mathrm{vec}(\tilde{X}_{P-1}) \end{bmatrix}, \quad \tilde{c} = \begin{bmatrix} \mathrm{vec}(\tilde{C}^{(P-1)}) \\ \mathrm{vec}(\tilde{C}^{(0)}) \\ \cdots \\ \mathrm{vec}(\tilde{C}^{(P-2)}) \end{bmatrix}. \tag{15}$$

Only the nonzero blocks of Z_{PSYCT} are displayed explicitly in (14) and by exploiting this structure the system (13) can be solved at the cost of $O(P(m^2n + mn^2))$ flops by using Gaussian elimination with partial pivoting [8] or structured variants of QR factorization [12,13]; the latter method avoids excessive pivot growth for ill-conditioned problems [5]. By storing only the block main diagonal, the block sub-diagonal and the rightmost block column vector, the storage requirement for Z_{PSYCT} can be kept at $3Pm^2n^2$. Linear systems with this kind of sparsity structure, *Bordered Almost Block Diagonal* (BABD) systems have been studied extensively [5].

The conditioning of (1) is essentially guided by the Sep-function

$$\mathrm{Sep}[\mathrm{PSYCT}] = \inf_{\|x\|_2=1} \|Z_{\mathrm{PSYCT}}x\|_2 = \|Z_{\mathrm{PSYCT}}^{-1}\|_2^{-1} = \sigma_{\min}(Z_{\mathrm{PSYCT}}) \quad (16)$$

$$= \inf_{(\sum_{k=0}^{K-1}\|X_k\|_F^2)^{1/2}=1} (\sum_{k=0}^{K-1} \|A_kX_k - X_{k+1}B_k\|_F^2)^{1/2}.$$

The quantity sep[PSYCT] can be estimated at the cost of solving a few PSYCTs by exploiting the estimation technique for the 1-norm of the inverse of a matrix [14,16,19,20] which was conducted in SCASY for the non-periodic case [10,11].

The Kronecker product representation (13) is only effective to use when m and n are very small, e.g., in kernel solvers for (sub)matrices of dimensions 1–2. Typically, for large-scale triangular matrix equations of the form (11), *recursive matrix blocking* and/or *iterative matrix blocking* of several layers is applied to reformulate the majority of the computational work into level 3 operations.

In [10], parallel algorithms for *non-periodic* Sylvester-type matrix equations were presented, introducing the software library SCASY [11,26]. In this paper, we show how the key concepts of SCASY can be generalized to cover even periodic Sylvester-type matrix equations of the forms (1) and (11). The rest of the paper is organized as follows. In Section 2, we present parallel wavefront algorithms for solving the triangular reduced problem (11). In Section 3, we discuss some implementation issues and before giving a summary and listing some future work in Section 5, we present some real experimental results in Section 4.

2 Parallel Algorithms for Periodic Triangular Matrix Equations

We assume that the cyclic matrix sequences A_k and B_k, $k = 0, 1, \ldots, P-1$, are already in periodic Schur form with quasi-triangular factors A_r and B_s. If A_k and B_k are partitioned by square $m_b \times m_b$ and $n_b \times n_b$ blocks, respectively, we can rewrite (1) in block partitioned form

$$A_{ii}^{(k)}X_{ij}^{(k)} - X_{ij}^{(k+1)}B_{jj}^{(k)} = C_{ij}^{(k)} - (\sum_{k=i+1}^{D_A} A_{ik}^{(k)}X_{kj}^{(k)} - \sum_{k=1}^{j-1} X_{ik}^{(k+1)}B_{kj}^{(k)}), \quad (17)$$

where $D_A = \lceil m/m_b \rceil$. Summation (17) can be implemented as a serial blocked algorithm using a couple of nested loops, see, e.g., [19,10] and the algorithms in [13] for matrix equations of the form (4). For high performance and portability, level 3 BLAS (mostly GEMM operations) should be utilized for the periodic right hand side updates.

Starting at the South-West corner of X_k, Equation (17) reveals that all subsolutions $X_{ij}^{(k)}$ located on the same block (sub- or super)diagonal, are independent and can be computed in parallel. Moreover, all subsequent updates are internally independent and can be performed in parallel.

Our parallel algorithms adapt to the ScaLAPACK (see, e.g., [3]) conventions of a distributed memory (DM) environment, as follows:

– The parallel processes are organized into a rectangular $P_r \times P_c$ mesh labeled from $(0, 0)$ to $(P_r - 1, P_c - 1)$ according to their specific position indices in the mesh.
– The matrices are distributed over the mesh using 2-dimensional (2D) block cyclic mapping with the block sizes m_b and n_b in the row and column dimensions, respectively.

Here we assume that the sequences A_k, B_k and C_k are internally aligned. Then, along the lines of the algorithms in SCASY, we formulate a parallel wavefront algorithm for PSYCT in Algorithm 1.

Algorithm 1. Parallel algorithm for PSYCT.

```
Input:   Matrix sequences $A_k$, $B_k$ and $C_k$. $A_k$ and $B_k$ in PRSF. Block sizes $m_b$ and $n_b$.
         Process grid configuration $P_r, P_c$.
Output:  Solution matrix sequence $X_k$ (which overwrites $C_k$).

for $k = 1$, # block diagonals in $C_k$ do
  % Solve subsystems on current $P$ block diagonals of $C_{0:P-1}$ in parallel
  if ($mynode$ holds $C_{ij}^{(0:P-1)}$) then
    if ($mynode$ does not hold $A_{ii}^{(0:P-1)}$ and/or $B_{jj}^{(0:P-1)}$) then
      Communicate for $A_{ii}^{(0:P-1)}$ and/or $B_{jj}^{(0:P-1)}$
    end if
    Solve subsystem $A_{ii}^{(k)} X_{ij}^{(k)} - X_{ij}^{(k+1)} B_{ij}^{(k)} = C_{ij}^{(k)},\ k = 0, 1, \ldots, P-1$
    Broadcast $X_{ij}^{(0:P-1)}$ to processors holding blocks in block row $i$ or block column $j$ of $C_{0:P-1}$
  else if ($mynode$ needs $X_{ij}^{(0:P-1)}$) then
    Receive $X_{ij}^{(0:P-1)}$
  end if
  if ($mynode$ needs block in $A_{0:P-1}$ for updates in block column $j$ of $C_{0:P-1}$) then
    Communicate for requested block in $A_{0:P-1}$
  end if
  Update block column $j$ of $C_{0:P-1}$ in parallel
  if ($mynode$ needs block in $B_{0:P-1}$ for updates in block row $i$) then
    Communicate for requested block in $B_{0:P-1}$
  end if
  Update block row $i$ of $C_{0:P-1}$ in parallel
end for
```

Notice that the *on-demand* communication scheme from SCASY [10] is generalized to the periodic case by communication of subsequences of the involved matrices. It is also possible to generalize the *matrix block shifts* communication strategy (see [10] and the references therein) to the periodic case.

3 Implementation Issues

Node solvers. As node solvers, we use both the multi-layer blocked periodic solvers from [9] and the recursive blocked periodic solvers from [8] which were developed in the framework of RECSY [17,18]. Both variants are rich in level

3 BLAS operations and apply blocking of each local subsystem to reduce its matrix dimensions down to a level where a kernel solver based on the Kronecker product representation (13) can be efficiently utilized.

Periodic implicit redistribution. To remove 2×2 blocks in A_r and B_s being shared by several blocks (and processors) in the explicit blocking, we generalize the concept of performing an *implicit redistribution* from SCASY [11]. In principle, the implicit redistribution rearranges the data distribution such that all data elements in the quasi-triangular factors are included but the distribution still conforms with ScaLAPACK conventions (see [11] for details). Three main routines from SCASY are used: `PDEXTCHK`, `PDIMPRED` and `PDBCKRD`. The periodic implicit redistribution works as follows:

– `PDEXTCHK` is used for searching the main diagonals of A_r and B_s for any 2×2 blocks shared by several data layout blocks. The routine returns *redistribution* information which is broadcasted to all processors. `PDEXTCHK` is only called two times, once for A_r and once for B_s, regardless of the length of the period P.
– `PDIMPRED` exchanges data between the processors via message passing to build up local arrays of extra elements which are used locally in constructing and decomposing "correct" subsequences from A_k, B_k and C_k (and X_k) on the nodes before invoking local node solvers or performing local GEMM updates. `PDIMPRED` is called P times, once for each triplet $(A_k, B_k, C_k),\ k = 0, 1, \ldots, P-1$, using the information from `PDEXTCHK`.
– `PDBCKRD` is called right before returning from the corresponding triangular solver and sends back the redistributed parts of a solution matrix sequence to their original owner processes such that the solution matrix sequence is correctly distributed over the process mesh on output. `PDBCKRD` is called P times, once for each right hand side matrix C_k.

4 Experimental Results

We have implemented Algorithm 1 as the routine `PTRPSYCTD` in Fortran 77 following the ScaLAPACK coding style. Our target machine is the 64-bit Opteron Linux Cluster *sarek* with 192 dual AMD Opteron nodes (2.2 GHz), 8GB RAM per node and a Myrinet-2000 high-performance interconnect with 250 MB/sec bandwidth. All experiments where conducted using the Portland Group's `pgf77 1.2.5 64-bit` compiler, the compiler flag `-fast` and the following software: MPICH-GM 1.5.2 [24], LAPACK 3.0 [22], GOTO-BLAS r0.94 [7], ScaLAPACK 1.7.0 and BLACS 1.1patch3 [3], SCASY 0.10beta and RECSY 0.01alpha [25]. All experiments are conducted in double precision arithmetic ($\epsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$).

Our test examples have the non-intersecting spectra: $\lambda(A_{P-1} \cdots A_1 A_0) = \{1, 2, \ldots, m\},\ \lambda(B_{P-1} \cdots B_1 B_0) = \{-1, -2, \ldots, -n\}$ and we use random right hand sides $C_{ij}^{(k)} \in [-1, 1],\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n,\ k = 0, 1, \ldots, P-1$.

We present representative results for PSYCT using up to 8×8 processor meshes in Table 1, including following quantities:

– The periodicity and the dimensions of the PSYCT equation: P, m and n.
– Parallel execution time: T_p, in seconds, where $p = P_r \times P_c$ is the number of utilized processor nodes.
– Parallel speedup: $S_p = T_{p_{\min}}/T_p$, where p is the number of utilized processors and $p_{\min}$ is the smallest number of processors for which the data structures of the current problem instance can be stored in the $p_{\min}$ chunks of main memory of the target computer.
– Mflops-rate: Mflops $= P(m^2n + mn^2)10^{-6}T_p^{-1}$
– Relative residual (Frobenius) norm:

$$r = \max_k \frac{\epsilon_{\text{mach}}^{-1}\|C_k - A_k\tilde{X}_k - \tilde{X}_{k+1}B_k\|}{\|A_k\|\|\tilde{X}_k\| + \|B_k\|\|\tilde{X}_{k+1}\| + \|C_k\|},$$

where $\tilde{X}_k,\ i = 0, 1, \ldots, P-1$, is the computed solution sequence. This residual norm should be of $O(1)$ for a reliable solution sequence $\tilde{X}_k$ [20], regardless of the conditioning of the underlying problem.

A few remarks regarding Table 1 are in order:

– The parallel execution time increases roughly linearly with the periodicity P, as illustrated in Figure 1.
– The Mflops-rate sometimes decreases with an increasing periodicity P for fixed values of m and n. This can be partly explained by the fact that an increased period leads to new data locality issues, since the blocks involved in the different operations (subsystem solves and GEMM-updates) are located

Table 1. Results of `PTRPSYCTD` on *sarek* using the block sizes $m_b = n_b = 64$

P	m	n	$P_r \times P_c$	T_p	Mflops	S_p	r
2	3000	3000	1×1	63.1	1712	1.0	0.3
2	3000	3000	2×2	29.9	3583	2.1	0.3
2	3000	3000	4×4	14.8	6993	4.3	0.3
2	3000	3000	8×8	7.54	14524	8.4	0.3
4	3000	3000	1×1	139	1556	1.0	0.4
4	3000	3000	2×2	66.4	3253	2.1	0.4
4	3000	3000	4×4	31.9	6771	4.4	0.4
4	3000	3000	8×8	15.8	13704	8.8	0.4
8	3000	3000	2×2	142	3038	1.0	0.3
8	3000	3000	4×4	67.2	6406	2.1	0.3
8	3000	3000	8×8	32.0	13505	4.4	0.3
2	6000	6000	2×2	195	3658	1.0	0.2
2	6000	6000	4×4	90.1	7898	2.2	0.2
2	6000	6000	8×8	40.4	17624	4.8	0.2
4	6000	6000	2×2	415	3432	1.0	0.4
4	6000	6000	4×4	208	6856	2.0	0.4
4	6000	6000	8×8	87.4	16278	4.7	0.4
8	6000	6000	4×4	429	8047	1.0	0.2
8	6000	6000	8×8	167	20736	2.6	0.2

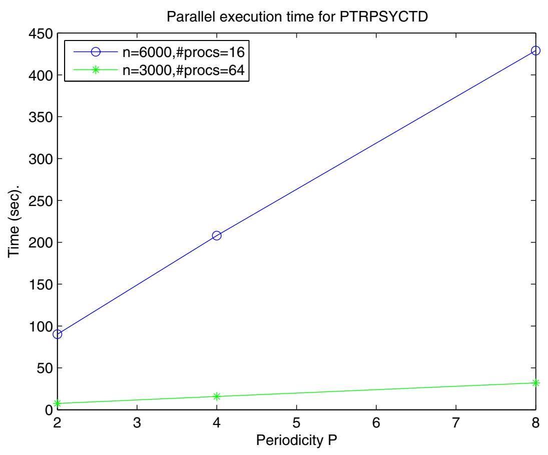

Fig. 1. Execution time results for `PTRPSYCTD` in relation to the periodicity P solving the PSYCT equation with $m = n = 3000, 6000$ on *sarek* using the block sizes $m_b = n_b = 64$

at far distance from each other, a problem that is amplified with an increasing period (see [8] for a similar observation).

– A main limitation of the possibility of achieving parallel speedup is the limited amount of physical memory on the target machine. For large periods, and when m and n are large enough for motivating the use of a distributed memory parallel computer, there is often not sufficient space to store all P matrices in the main memory unless one uses a huge number of processors.

The execution time information is also displayed in Figure 2. A general observation is that for all but the last results for $n = 6000$ and $p = 8$, an increase of the number of processors by a factor 4 cuts down the parallel execution time

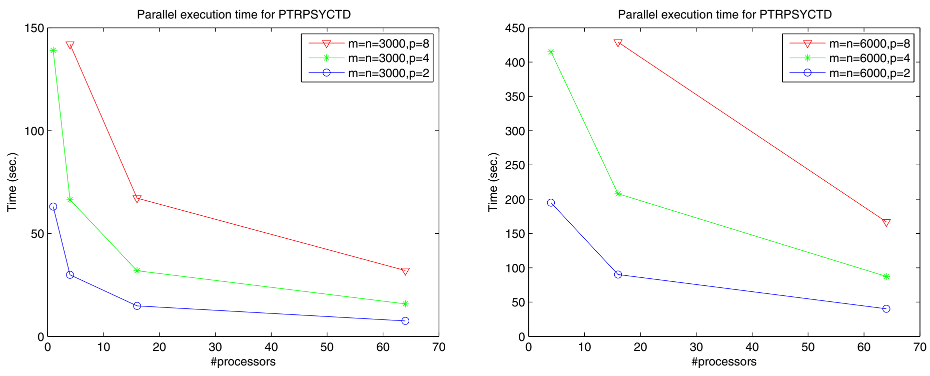

Fig. 2. Execution time results for `PTRPSYCTD` solving the PSYCT equation with $m = n = 3000, 6000$ on *sarek* using the block sizes $m_b = n_b = 64$

by roughly a factor 2. This is consistent with earlier observations, see, e.g., the performance model of the non-periodic ($P = 1$) SCASY implementations in [10].

5 Summary and Future Work

The work presented in this contribution was based on some preliminary results presented in [1] and can be further generalized to other periodic matrix equations (see, e.g., [8] for a list). In this context, there is a need for developing parallel versions of the periodic QR and QZ algorithms; for serial variants see, e.g., [4,15,21,23] and the references therein.

Acknowledgements

This research was conducted using the resources of the High Performance Computing Center North (HPC2N). Financial support has been provided by the *Swedish Research Council* under grant VR 621-2001-3284 and by the *Swedish Foundation for Strategic Research* under grant A3 02:128.

References

1. Andersson, P.: Parallella algoritmer för periodiska ekvationer av Sylvester-typ. Master's thesis, Tech. report UMNAD 715/07, Department of Computing Science, Umeå University (in Swedish) (2007)
2. Bartels, R.H., Stewart, G.W.: Algorithm 432: The Solution of the Matrix Equation $AX - BX = C$. Communications of the ACM 8, 820–826 (1972)
3. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J.W., Dhillon, I., Dongarra, J.J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. SIAM, Philadelphia (1997)
4. Bojanczyk, A., Golub, G.H., Van Dooren, P.: The periodic Schur decomposition; algorithm and applications. In: Proc. SPIE Conference, vol. 1770, pp. 31–42 (1992)
5. Fairweather, G., Gladwell, I.: Algorithms for Almost Block Diagonal Linear Systems. SIAM Review 44(1), 49–58 (2004)
6. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
7. GOTO-BLAS - High-Performance BLAS by Kazushige Goto, `http://www.cs.utexas.edu/users/flame/goto/`
8. Granat, R., Jonsson, I., Kågström, B.: Recursive Blocked Algorithms for Solving Periodic Triangular Sylvester-type Matrix Equations. In: Kågström, B., Elmroth, E., Dongarra, J., Waśniewski, J. (eds.) PARA 2006. LNCS, vol. 4699, pp. 531–539. Springer, Heidelberg (2007)
9. Granat, R., Kågström, B.: Direct eigenvalue reordering in a product of matrices in periodic schur form. SIAM J. Matrix Anal. Appl. 28(1), 285–300 (2006)
10. Granat, R., Kågström, B.: Parallel Solvers for Sylvester-type Matrix Equations with Applications in Condition Estimation, Part I: Theory and Algorithms, Tech. report UMINF-07.15. ACM TOMS (submitted 2007)

11. Granat, R., Kågström, B.: Parallel Solvers for Sylvester-type Matrix Equations with Applications in Condition Estimation, Part II: the SCASY Software, Tech. report UMINF-07.16. ACM TOMS (submitted 2007)
12. Granat, R., Kågström, B., Kressner, D.: Computing Periodic Deflating Subspaces Associated with a Specified Set of Eigenvalues. BIT Numerical Mathematics 47(4), 763–791 (2007)
13. Granat, R., Kågström, B., Kressner, D.: MATLAB tools for Solving Periodic Eigenvalue Problems. In: Proceedings of 3rd IFAC Workshop PSYCO 2007, Saint Petersburg, Russia (2007)
14. Hager, W.W.: Condition estimates. SIAM J. Sci. Statist. Comput. 3, 311–316 (1984)
15. Hench, J.J., Laub, A.J.: Numerical solution of the discrete-time periodic Riccati equation. IEEE Trans. Automat. Control 39(6), 1197–1210 (1994)
16. Higham, N.J.: Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. ACM Trans. of Math. Software 14(4), 381–396 (1988)
17. Jonsson, I., Kågström, B.: Recursive blocked algorithms for solving triangular systems. I. One-sided and coupled Sylvester-type matrix equations. ACM Trans. Math. Software 28(4), 392–415 (2002)
18. Jonsson, I., Kågström, B.: RECSY - A High Performance Library for Solving Sylvester-Type Matrix Equations. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 810–819. Springer, Heidelberg (2003)
19. Kågström, B., Poromaa, P.: Distributed and shared memory block algorithms for the triangular Sylvester equation with sep^{-1} estimators. SIAM J. Matrix Anal. Appl. 13(1), 90–101 (1992)
20. Kågström, B., Poromaa, P.: LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs. ACM Trans. Math. Software 22(1), 78–103 (1996)
21. Kressner, D.: An efficient and reliable implementation of the periodic QZ algorithm. In: IFAC Workshop on Periodic Control Systems (2001)
22. LAPACK - Linear Algebra Package, `http://www.netlib.org/lapack/`
23. Lust, K.: Improved numerical Floquet multipliers. Internat. J. Bifur. Chaos Appl. Sci. Engrg. 11(9), 2389–2410 (2001)
24. MPI - Message Passing Interface, `http://www-unix.mcs.anl.gov/mpi/`
25. RECSY - High Performance library for Sylvester-type matrix equations, `http://www.cs.umu.se/research/parallel/recsy`
26. SCASY - ScaLAPACK-style solvers for Sylvester-type matrix equations, `http://www.cs.umu.se/granat/scasy.html`
27. Varga, A.: Periodic Lyapunov equations: some applications and new algorithms. Internat. J. Control 67(1), 69–87 (1997)
28. Varga, A., Van Dooren, P.: Computational methods for periodic systems - an overview. In: Proc. of IFAC Workshop on Periodic Control Systems, Como, Italy, pp. 171–176 (2001)