Recursive Blocked Algorithms for Solving Periodic Triangular Sylvester-Type Matrix Equations

Robert Granat, Isak Jonsson, and Bo Kågström

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

Abstract. Recently, recursive blocked algorithms for solving triangular one-sided and two-sided Sylvester-type equations were introduced by Jonsson and Kågström. This elegant yet simple technique enables an automatic variable blocking that has the potential of matching the memory hierarchies of today's HPC systems. The main parts of the computations are performed as level 3 general matrix multiply and add (GEMM) operations. We extend and apply the recursive blocking technique to solving periodic Sylvester-type matrix equations. Successive recursive splittings are performed on 3-dimensional arrays, where the third dimension represents the periodicity of a matrix equation.

Keywords: Sylvester-type matrix equations, periodic matrix equations, recursion, blocking, level 3 BLAS, superscalar.

1 Introduction

The standard Sylvester equation $AX - XB = C$ has a *periodic* counter-part

$$\begin{aligned} A_k X_k - X_{k+1} B_k &= C_{k+1}, \qquad k = 1, \ldots, p-1, \\ A_p X_p - X_1 B_p &= C_1, \end{aligned}$$

where p is the periodicity of the matrix sequences, such that $A_{k+p} = A_k, B_{k+p} = B_k$ and $C_{k+p} = C_k$ [16,19]. In this contribution, we focus on recursive blocked algorithms for solving triangular periodic matrix equations, i.e., the matrix sequences A_k and B_k for $k = 1, \ldots, p$ are assumed to be in *periodic real Schur form* (PRSF) [4,10]. This means that $p-1$ of the matrices in each sequence are upper triangular and one matrix in each sequence, say A_r and B_s, $1 \leq r, s \leq p$, is quasi-triangular. The products of conforming diagonal blocks of the matrix sequences A_k and B_k contain the eigenvalues of the matrix products $A_1 A_2 \cdots A_p$ and $B_1 B_2 \cdots B_p$, respectively, where the 1×1 and 2×2 blocks on the main block diagonal of A_r and B_s correspond to real and complex conjugate pairs of eigenvalues of the corresponding matrix products.

Triangular matrix equations appear naturally in estimating the condition numbers of matrix equations and different eigenspace computations (e.g., see [14,15]

B. Kågström et al. (Eds.): PARA 2006, LNCS 4699, pp. 531–539, 2007.

and [2,17]), including decoupling and stability analysis. Periodic Sylvester-type matrix equations also appear in the context of eigenvalue reordering for computation and condition estimation of periodic invariant (deflating) subspaces of a matrix (pair) sequence [7,8,9]. To solve a triangular matrix equation is also a major step in the classical Bartels-Stewart method [1], which is our base-point for solving general non-reduced periodic matrix equations.

2 Recursive Algorithms for Periodic Triangular Matrix Equations

Our work includes novel recursive blocked algorithms for solving the most common one-sided and two-sided triangular periodic Sylvester-type matrix equations. In Table 1, a summary of the periodic matrix equations considered is displayed.

The classification in one-sided and two-sided matrix equations was introduced in [11,12] and is implicit in the definition of a matrix equation. A periodic matrix equation is *one-sided* if it only includes terms where the solution is involved in matrix products of two matrices, e.g., $\mathrm{op}(A_k)X_k$ or $X_k\mathrm{op}(A_k)$, where $\mathrm{op}(A_k)$ can be A_k or A_k^T. Similarly, a periodic matrix equation is *two-sided* if it includes matrix products of three matrices, e.g., $A_kX_kB_k^T$, where X_k is the solution sequence. This distinction relates to how blocks (subarrays) of matrices of the solution sequence (e.g., X_k in PSYCT) are used in updates of the right hand side matrices (C_k in PSYCT) in the recursive blocked algorithms. For example, our algorithms for two-sided matrix equations require more space and flops, compared to similar algorithms for one-sided equations.

For the two-sided equations, we only display one periodic transpose variant. The other variants can be derived by moving the transpose to the left multiplying matrices and replacing the periodic dependence $k+1$ by k and vice versa. For example, a second variant of PSYDT is

$$\begin{aligned} {A_k}^T X_{k+1} B_k - X_k &= C_k, \qquad k = 1, \ldots, p-1, \\ {A_p}^T X_1 B_p - X_p &= C_p. \end{aligned}$$

For the generalized equations in Table 1 we assume that the involved periodic matrix pairs, namely (A_k, D_k) and (B_k, E_k) in PGCSY, (A_k, C_k) and (B_k, D_k) in PGSYL, and (A_k, E_k) in PGLYCT and PGLYDT, are in *generalized* periodic real Schur form (GPRSF) (see, e.g., [4,10] for details).

By using and reusing recursive templates, we can solve all matrix equations listed in Table 1 utilizing only a small set of subroutines. By this, we mean that, e.g., the PLYCT problem can be largely solved by the PSYCT routine. Therefore, the efforts of optimizing the implementation can be concentrated on a few core routines.

In the following, we discuss only a few of the periodic matrix equations of Table 1 in some more detail.

The periodic matrix sequences are stored as 3-dimensional arrays, where the third dimension is the periodicity p of the matrix equation. The successive recur-

Table 1. Considered one-sided (top) and two-sided (bottom) periodic Sylvester-type matrix equations. Here, p is the periodicity of each equation and $1 \leq k < p-1$.

Name	Mnemonic	Matrix equation
Periodic continuous-time standard Sylvester	PSYCT	$\begin{cases} A_k X_k - X_{k+1} B_k = C_k \\ A_p X_p - X_1 B_p = C_p \end{cases}$
Periodic continuous-time standard Lyapunov	PLYCT	$\begin{cases} A_k X_k + X_{k+1} {A_k}^T = C_k \\ A_p X_p + X_1 {A_p}^T = C_p \end{cases}$
Periodic generalized coupled Sylvester	PGCSY	$\begin{cases} A_k X_k - Y_k B_k = C_k \\ D_k X_{k+1} - Y_k E_k = F_k \\ A_p X_p - Y_p B_p = C_p \\ D_p X_1 - Y_p E_p = F_p \end{cases}$
Periodic discrete-time standard Sylvester	PSYDT	$\begin{cases} A_k X_k {B_k}^T - X_{k+1} = C_k \\ A_p X_p {B_p}^T - X_1 = C_p \end{cases}$
Periodic discrete-time standard Lyapunov	PLYDT	$\begin{cases} A_k X_k {A_k}^T - X_{k+1} = C_k \\ A_p X_p {A_p}^T - X_1 = C_p \end{cases}$
Periodic generalized Sylvester	PGSYL	$\begin{cases} A_k X_k {B_k}^T - C_k X_{k+1} {D_k}^T = E_k \\ A_p X_p {B_p}^T - C_p X_1 {D_p}^T = E_p \end{cases}$
Periodic continuous-time generalized Lyapunov	PGLYCT	$\begin{cases} A_k X_k {E_k}^T + E_k X_{k+1} {A_k}^T = C_k \\ A_p X_p {E_p}^T + E_p X_1 {A_p}^T = C_p \end{cases}$
Periodic discrete-time generalized Lyapunov	PGLYDT	$\begin{cases} A_k X_k {A_k}^T - E_k X_{k+1} {E_k}^T = C_k \\ A_p X_p {A_p}^T - E_p X_1 {E_p}^T = C_p \end{cases}$

sive splittings are performed on the 3-dimensional arrays explicitly, leading to new types of data locality issues, compared to our previous work with RECSY [11,12,13].

2.1 Periodic Recursive Sylvester Solvers

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

$$\begin{aligned} A_k X_k - X_{k+1} B_k &= C_{k+1}, \qquad k = 1, \ldots, p-1, \\ A_p X_p - X_1 B_p &= C_1, \end{aligned}$$

where the sequences A_k of size $M \times M$ and B_k of size $N \times N$ for $k = 1, \ldots, p$ are in PRSF form. The right hand sides C_k and the solution matrices X_k are of side $M \times N$. Depending on the dimensions M and N, we consider three ways of

recursive splitting. First, we consider splitting of A_k by rows and columns and C_k by rows only. The second alternative is to split B_k by rows and columns and C_k by columns only. The third alternative is to split all three matrices by rows and columns. No matter which alternative is chosen, the number of flops is the same. Performance may differ greatly, though. Our algorithm picks the alternative that keeps matrices as "squarish" as possible, i.e., $1/2 < M/N < 2$, which guarantees a good ratio between the number of flops and the number of elements referenced.

Next, we consider the *periodic generalized continuous-time Lyapunov* (PGLYCT) equation

$$\begin{aligned} A_k X_k {E_k}^T + E_k X_{k+1} {A_k}^T &= C_k, \qquad k = 1, \ldots, p-1, \\ A_p X_p {E_p}^T + E_p X_1 {A_p}^T &= C_p, \end{aligned}$$

where the periodic matrix pair sequence (A_k, E_k) is in generalized periodic real Schur form (GPRSF), and C_k and X_k (overwrites C_k) are symmetric $M \times M$. Because of symmetry, there is only one way to split the equation, resulting in two triangular PGLYCT equations and one *generalized periodic Sylvester* (PGSYL) equation, all of which can be solved recursively using the following template:

$$\begin{aligned} A_{11}^{(k)} X_{11}^{(k)} {E_{11}^{(k)}}^T + E_{11}^{(k)} X_{11}^{(k+1)} {A_{11}^{(k)}}^T &= C_{11}^{(k)} \\ &\quad - A_{12}^{(k)} X_{12}^{(k)} {E_{11}^{(k)}}^T - E_{12}^{(k)} X_{12}^{(k+1)^T} {A_{11}^{(k)}}^T \\ &\quad - A_{11}^{(k)} X_{12}^{(k)} {E_{12}^{(k)}}^T - E_{11}^{(k)} X_{12}^{(k+1)} {A_{12}^{(k)}}^T \\ &\quad - A_{12}^{(k)} X_{22}^{(k)} {E_{12}^{(k)}}^T - E_{12}^{(k)} X_{22}^{(k+1)} {A_{12}^{(k)}}^T, \\ A_{11}^{(k)} X_{12}^{(k)} {E_{22}^{(k)}}^T + E_{11}^{(k)} X_{12}^{(k+1)} {A_{22}^{(k)}}^T &= C_{12}^{(k)} - A_{12}^{(k)} X_{22}^{(k)} {E_{22}^{(k)}}^T - E_{12}^{(k)} X_{22}^{(k+1)} {A_{22}^{(k)}}^T, \\ A_{22}^{(k)} X_{22}^{(k)} {E_{22}^{(k)}}^T + E_{22}^{(k)} X_{22}^{(k+1)} {A_{22}^{(k)}}^T &= C_{22}^{(k)}. \end{aligned}$$

Assuming that we have algorithms for solving PGLYCT and PGSYL (kernel solvers are discussed in Section 2.2), we start by solving for the sequence $X_{22}^{(k)}$, $k = 1, \ldots, p$ from the third (last) equation. After updating $C_{12}^{(k)}$ in the PGSYL equation (second above) with respect to $X_{22}^{(k)}$, $k = 1, \ldots, p$, we can solve for the sequence $X_{12}^{(k)}$. Finally, after updating $C_{11}^{(k)}$ in the first PGLYCT equation with respect to $X_{12}^{(k)}$ and $X_{22}^{(k)}$ for $k = 1, \ldots, p$, we solve for the sequence $X_{11}^{(k)}$.

The recursive template is now applied repeatedly to the three periodic matrix equations above (divide phase) until the subproblems are small enough, when kernel solvers are used for solving the node-leaf problems of the recursive tree. In the conquer phase, the tree is traversed level by level, finally producing the complete matrices X_k, $k = 1, \ldots, p$, i.e., the solution of PGLYCT.

2.2 Kernel Solvers for Leaf Problems

In each step of the recursive blocking, the original periodic matrix equation is reduced to several subproblems involving smaller and smaller matrices, and a

great part of the computation emerges as standard matrix-matrix operations, such as general matrix multiply and add (GEMM) or triangular matrix multiply (TRMM) operations. At the end of the recursion tree, small instances of periodic matrix equations have to be solved. Each such matrix equation can be represented as a linear system $Zx = c$, where Z is a Kronecker product representation of the associated periodic Sylvester-type operator, and it belongs to the class of bordered almost block diagonal (BABD) matrices [6]. For example, the PSYCT matrix equation can be expressed as $Zx = c$, where the matrix Z of size $mnp \times mnp$ is

$$Z = \begin{bmatrix} B_p^T \otimes I_m & & & I_n \otimes A_p \\ I_n \otimes A_1 & B_1^T \otimes I_m & & \\ & \ddots & \ddots & \\ & & I_n \otimes A_{p-1} & B_{p-1}^T \otimes I_m \end{bmatrix},$$

and

$$x = [\mathrm{vec}(X_1), \mathrm{vec}(X_2), \cdots, \mathrm{vec}(X_p)]^T, \quad c = [\mathrm{vec}(C_1), \mathrm{vec}(C_2), \cdots, \mathrm{vec}(C_p)]^T.$$

In the algorithm, recursion proceeds down to problem sizes of 1×1 to 2×2. For these problems, a compact form of the matrix Z which utilizes the sparsity structure of the problem is computed and the problem is solved using Gaussian elimination with partial pivoting (GEPP). These solvers are based on the superscalar kernels that were developed for the RECSY library [13]. Moreover, the block diagonal of the matrix Z sometimes (see, e.g., PGCSY in [8]) has a certain structure that can be exploited by the GEPP procedure. The memory usage for Z is $O(m^2n^2p)$, and the number of operations required to solve the problem is $O(m^3n^3p)$. In case of an ill-conditioned matrix Z, the Gaussian elimination is aborted when bad pivot elements are detected, an error condition is signaled, and the solution process is restarted with a new problem from a higher level of the recursion tree. This larger problem is then solved using LU with complete pivoting (GECP) on a non-compact form of Z, which in turn results in a p times larger memory requirement, namely $O(m^2n^2p^2)$ storage. However, since m and n are small, typically 1 or 2, this is an effective procedure for typical sizes of the periodicity p. The extra workspace can either be provided by the user or dynamically allocated.

2.3 Storage Layout of Matrices

For storage of regular dense matrices, there are two major linear variants: row-major ("C-style") and column-major ("Fortran-style"). In addition, several recursive blocked storage schemas have proven to give substantial performance improvements, see [5] and further references therein.

For the periodic matrix sequences, there are six (3!) different linear variants. One advantage of having the periodic dimension as the minor (innermost) dimension is better locality of the Z matrix in the kernel solver. However, this

efficiently disables all use of standard level 3 BLAS. Therefore, we have the periodicity p as the outermost dimension. Column-major storage layout is used for each coefficient matrix (A_k, B_k, C_k etc.) in the periodic matrix sequences.

3 Sample Performance Results

The recursive blocked algorithms for the periodic Sylvester-type equations have been implemented in Fortran 90, using the facilities for recursive calls of subprograms, dynamic memory allocation and threads. In this section, we present sample performance results of implementations of solvers for one-sided and two-sided equations executing on an AMD Opteron processor-based system. The system has a dual AMD Opteron 2.2 GHz processor, with a 64 kB level 1 cache, a 1024 kB level 2 cache and 8 GB memory per node. Theoretical peak performance is 4.4 Gflops/s per processor. The peak performance of DGEMM and other level 3 BLAS routines used vary between 3.0–3.5 Gflops/s. All performance numbers presented are based on uniprocessor computations.

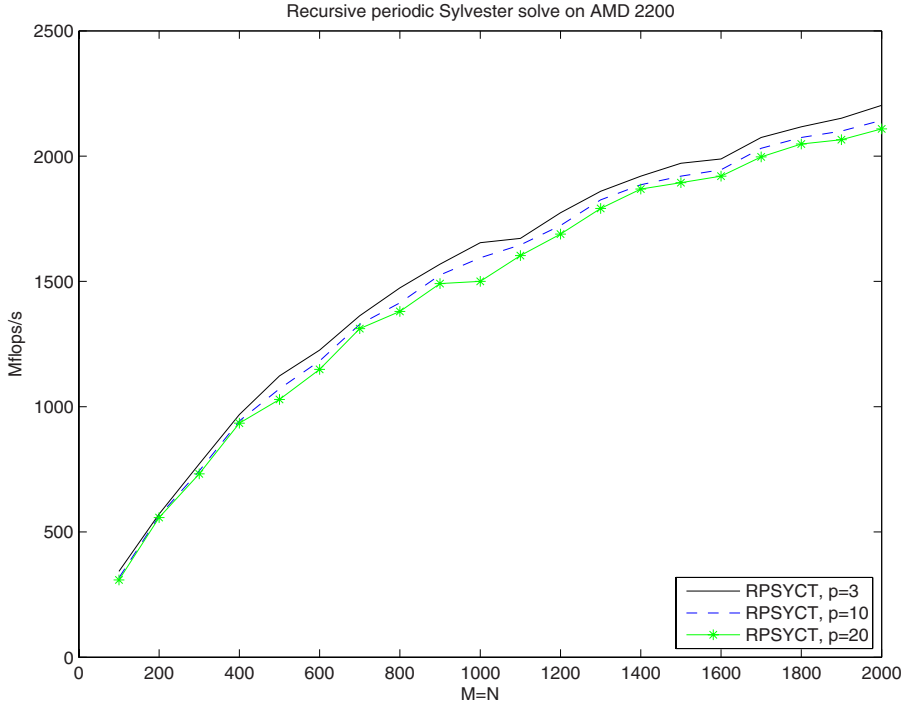

Fig. 1. Performance results for the one-sided PSYCT equation ($M = N$) on AMD Opteron. The three graphs correspond to the periodicity $p = 3$, 10, and 20.

3.1 Performance of the Recursive Blocked PSYCT Solver

In Figure 1, performance graphs for the implementation of the recursive blocked PSYCT algorithm are displayed. The problem size ($M = N$) ranges from 100 to 2000, and the periodicity $p = 3, 10$, and 20. For large enough problems the performance approaches 70% of the DGEMM performance, which is on a level with the recursive blocked SYCT solver in RECSY [11,13]. For an increasing periodicity p the performance decreases only marginally.

3.2 Performance of the Recursive Blocked PGSYL Solver

In Figure 2, we show performance graphs for our implementation of the recursive blocked algorithm for the two-sided PGSYL equation. The performance results are somewhat inferior to the PSYCT performance, but still on a level with the recursive blocked GSYL solver in RECSY [12,13]. We also see that the relative decrease in performance with respect to an increasing periodiciy p is larger than for the PSYCT solver. Reasons for this degradation include that the PGSYL kernel solver is more complex and somewhat less efficient, and the two-sided updates in the recursive blocked algorithm result in extra operations compared

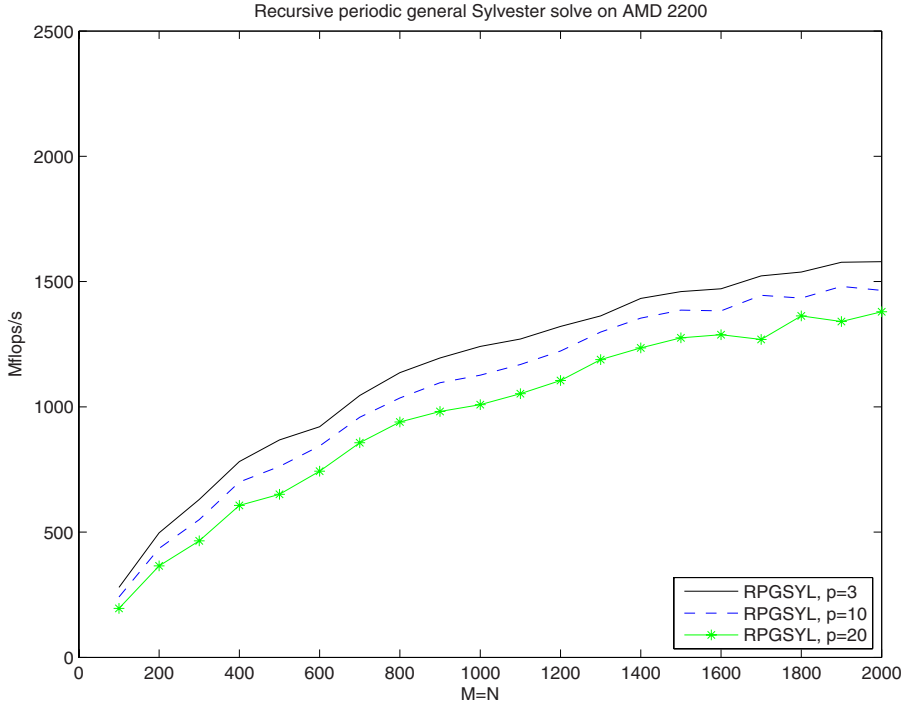

Fig. 2. Performance results for the two-sided PGSYL equation ($M = N$) on AMD Opteron. The three graphs correspond to the periodicity $p = 3$, 10, and 20.

to a true level 1 or level 2 algorithm. Such overhead never appear in the one-sided equations, but increases with p for two-sided equations. However, the use of efficient level 3 operations compensates for some of this computational overhead.

4 Conclusions and Future Work

We have presented novel recursive blocked algorithms for solving various periodic triangular matrix equations. Such equations stem from different applications with a periodic or seasonal behaviour, e.g., the study of periodic control systems [3], and discrete-time periodic (descriptor) systems [19] in particular.

Our recursive blocked algorithms are based on RECSY, an HPC library for the most common non-periodic matrix equations (see [11,12,13]). The performance results are on the level of RECSY, which confirm that the recursive blocking approach is an efficient way of solving periodic triangular Sylvester-type equations. The reason is three-fold: (*i*) recursion allows for good temporal locality; (*ii*) recursion enables the periodic matrix equations to be rewritten mainly as level 3 operations; (*iii*) novel superscalar kernel solvers deliver good performance for the small leaf-node problems. Our goal is to provide a complete periodic counterpart of the RECSY library. This study will also include alternative blocking techniques.

Acknowledgments

This research was conducted using the resources of the High Performance Computer 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. Bartels, R.H., Stewart, G.W.: Solution of the equation $AX + XB = C$. Comm. Assoc. Comput. Mach. 15, 820–826 (1972)
2. Benner, P., Mehrmann, V., Xu, H.: Perturbation analysis for the eigenvalue problem of a formal product of matrices. BIT 42(1), 1–43 (2002)
3. Bittanti, S., Colaneri, P. (eds.): Periodic Control Systems. In: Proceedings volume from the IFAC Workshop, August 27-28, 2001, Elsevier Science & Technology, Cernobbio-Como, Italy (2001)
4. Bojanczyk, A.W., Golub, G., Van Dooren, P.: The Periodic Schur Decomposition. Algorithms and Applications. In: Luk, F.T. (ed.) Proceedings SPIE Conference, vol. 1770, pp. 31–42 (1992)
5. Elmroth, E., Gustavson, F., Jonsson, I., Kågström, B.: Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Library Software. SIAM Review 46(1), 3–45 (2004)
6. Fairweather, G., Gladwell, I.: Algorithms for Almost Block Diagonal Linear Systems. SIAM Review 44(1), 49–58 (2004)

7. 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)
8. Granat, R., Kågström, B., Kressner, D.: 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 (2006) ISBN: 0-7803-9797-5
9. Granat, R., Kågström, B., Kressner, D.: Computing Periodic Deflating Subspaces Associated with a Specified Set of Eigenvalues. BIT Numerical Mathematics, December 2006 (submitted)
10. Hench, J.J., Laub, A.J.: Numerical solution of the discrete-time periodic Riccati equation. IEEE Trans. Automat. Control 39(6), 1197–1210 (1994)
11. Jonsson, I., Kågström, B.: Recursive blocked algorithms for solving triangular systems — Part I: One-sided and coupled Sylvester-type matrix equations. ACM Trans. Math. Softw. 28(4), 392–415 (2002)
12. Jonsson, I., Kågström, B.: Recursive blocked algorithms for solving triangular systems — Part II: Two-sided and generalized Sylvester and Lyapunov matrix equations. ACM Trans. Math. Softw. 28(4), 416–435 (2002)
13. Jonsson, I., Kågström, B.: RECSY — A High Performance Library for Sylvester-Type Matrix Equations. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 810–819. Springer, Heidelberg (2003)
14. 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, 78–103 (1996)
15. Kågström, B., Westin, L.: Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. IEEE Trans. Automat. Control 34(4), 745–751 (1989)
16. Sreedhar, J., Van Dooren, P.: A Schur approach for solving some matrix equations. In: Helmke, U., Menniken, R., Saurer, J. (eds.) Systems and Networks: Mathematical Theory and Applications, Mathematical Research, vol. 77, pp. 339–362 (1994)
17. Sun, J.-G.: Perturbation bounds for subspaces associated with periodic eigenproblems. Taiwanese Journal of Mathematics 9(1), 17–38 (2005)
18. Varga, A.: Periodic Lyapunov equations: some applications and new algorithms. Internat. J. Control 67(1), 69–87 (1997)
19. Varga, A., Van Dooren, P.: Computational methods for periodic systems. In: Prepr. IFAC Workshop on Periodic Control Systems, Como, Italy, pp. 177–182 (2001)