Parallel Algorithms and Condition Estimators for Standard and Generalized Triangular Sylvester-Type Matrix Equations

Robert Granat and Bo Kågström

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

Abstract. We discuss parallel algorithms for solving eight common standard and generalized triangular Sylvester-type matrix equation. Our parallel algorithms are based on explicit blocking, 2D block-cyclic data distribution of the matrices and wavefront-like traversal of the right hand side matrices while solving small-sized matrix equations at different nodes and updating the rest of the right hand side using level 3 operations. We apply the triangular solvers in condition estimation, developing parallel sep^{-1}-estimators. Some experimental results are presented.

Keywords: Sylvester-type matrix equations, triangular matrix equations, Bartels–Stewart's method, explicit blocking, condition estimation, level 3 BLAS, ScaLAPACK, 2D block-cyclic data distribution.

1 Introduction

We consider the following standard Sylvester-type matrix equations: the *continuous-time Sylvester equation* (SYCT)

$$AX - XB = C, \tag{1}$$

the discrete-time Sylvester equation (SYDT)

$$AXB^T - X = C, \tag{2}$$

the continuous-time Lyapunov equation (LYCT)

$$AX + XA^T = C, \tag{3}$$

and *the discrete-time Lyapunov equation* (LYDT)

$$AXA^T - X = C, \tag{4}$$

where A of size $m \times m$, B of size $n \times n$ and C of size $m \times n$ or $m \times m$ are general matrices with real entries.

B. Kågström et al. (Eds.): PARA 2006, LNCS 4699, pp. 127–136, 2007.

We also consider the following generalized Sylvester-type matrix equations: *the generalized coupled Sylvester equation* (GCSY)

$$(AX - YB,\ DX - YE) = (C,\ F), \tag{5}$$

where A and D of size $m \times m$, B and E of size $n \times n$ and C and F of size $m \times n$ are general matrices with real entries, *the generalized Sylvester equation* (GSYL)

$$AXB^T - CXD^T = E, \tag{6}$$

where A and C of size $m \times m$, B and D of size $n \times n$ and E of size $m \times n$ are general matrices with real entries, *the continuous-time generalized Lyapunov equation* (GLYCT)

$$AXE^T + EXA^T = C, \tag{7}$$

where A, E and C of size $m \times m$ are general matrices with real entries, and *the discrete-time generalized Lyapunov equation* (GLYDT)

$$AXA^T - EXE^T = C, \tag{8}$$

where A, E and C of size $m \times m$ are general matrices with real entries.

Solvability conditions for equations (1)-(8) can be formulated in terms of the standard or generalized eigenvalues of the involved matrices or regular matrix pairs, see, e.g., [15,16]. For (G)LYCT/(G)LYDT a symmetric right hand side C implies a symmetric solution X.

SYCT, LYCT and GCSY are called *one-sided* because the undetermined X (or X and Y) is multiplied by another matrix from one side only. SYDT, LYDT, GSYL, GLYCT and GLYDT are called *two-sided* [15,16].

In this contribution, we assume that all left hand side coefficient matrices or matrix pairs are (quasi-)triangular, i.e, in real or generalized Schur form (see, e.g., [5]). If this is not the case, we utilize Bartels–Stewart's method [2] for reducing the matrix equation to triangular form by orthogonal transformations:

1. Reduce the known left hand side matrices (or matrix pairs) of equations (1)-(8) to real (generalized) Schur form.
2. Update the right hand side matrix (or matrix pair) with respect to the Schur decompositions.
3. Solve the resulting triangular matrix equation.
4. Transform the computed solution matrix (or matrix pair) back to the original coordinate system.

Based on our previous work with parallel solvers for triangular matrix equations (see, e.g., [8,9,10]), we now focus on developing a complete set of parallel algorithms and library routines for solving the reduced matrix equations corresponding to equations (1)-(8). Besides being an integral part in solving general large scale matrix equations, these solvers are applied to condition estimation of the matrix equations themselves as well as for different subspace problems with applications in control theory. In this contribution, we present some of this work in progress. The final goal is a complete library of ScaLAPACK-style routines called SCASY for solving general as well as reduced (triangular) standard and generalized Sylvester-type equations (1)-(8).

2 Blocked Methods for Solving Reduced Matrix Equations

We focus on step 3 above for solving the reduced matrix equations. Assuming $m = n$, this is an $O(n^3)$ operation. We apply explicit blocking (see below) to reformulate each matrix equation problem into as much level 3 BLAS operations as possible. In the following, the (i, j)th block of a partitioned matrix, say X, is denoted X_{ij}.

Let mb and nb be block sizes used in an explicit block partitioning of the matrices A and B in SYCT, respectively. In turn, this imposes a similar block partitioning of C and X (which overwrites C). Then $D_a = \lceil m/mb \rceil$ and $D_b = \lceil n/nb \rceil$ are the number of diagonal blocks in A and B, respectively. Now, SYCT can be rewritten in block-partitioned form as

$$A_{ii}X_{ij} - X_{ij}B_{jj} = C_{ij} - \Big(\sum_{k=i+1}^{D_a} A_{ik}X_{kj} - \sum_{k=1}^{j-1} X_{ik}B_{kj} \Big), \tag{9}$$

for $i = 1, 2, \ldots, D_a$ and $j = 1, 2, \ldots, D_b$ [10].

For LYCT we use a similar approach: Partition A and C by rows and columns using a single block size mb and rewrite LYCT as

$$A_{ii}X_{ij} + X_{ij}A_{jj}^T = C_{ij} - \Big(\sum_{k=i+1}^{D_a} A_{ik}X_{kj} + \sum_{k=j+1}^{D_a} X_{ik}A_{jk}^T \Big), \tag{10}$$

reformulating our single LYCT problem into smaller SYCT ($i \neq j$) and LYCT ($i = j$) problems and level 3 updates in the right hand side C. Moreover, if C is symmetric, we rewrite (10) for the main diagonal blocks C_{ii} as

$$A_{ii}X_{ii} + X_{ii}A_{ii}^T = C_{ii} - \Big(\sum_{k=i+1}^{D_a} A_{ik}X_{ik}^T + X_{ik}A_{ik}^T \Big), \tag{11}$$

which defines a sum of SYR2K-operations, which are as fast as regular GEMM-operations when implemented as a GEMM-based level 3 BLAS [18,19].

We block the two-sided standard equations SYDT and LYDT similarly:

$$A_{ii}X_{ij}B_{jj}^T - X_{ij} = C_{ij} - \sum_{(k,l)=(i,j)}^{(D_a,D_b)} A_{ik}X_{kl}B_{jl}^T, \ (k,l) \neq (i,j) \tag{12}$$

and

$$A_{ii}X_{ij}A_{jj}^T - X_{ij} = C_{ij} - \sum_{(k,l)=(i,j)}^{(D_a,D_a)} A_{ik}X_{kl}A_{jl}^T, \ (k,l) \neq (i,j). \tag{13}$$

Notice that the blocking of LYDT decomposes the problem into several smaller SYDT ($i \neq j$) and LYDT ($i = j$) equations.

The same method of explicit blocking is applied to the generalized matrix equations (5)-(8).

All linear matrix equations considered can be rewritten as an equivalent large linear system of equations $Zx = y$, where Z is the Kronecker product representation of the corresponding Sylvester-type operator. For example, SYCT (1) corresponds to $Z_{\mathrm{SYCT}} = I_n \otimes A - B^T \otimes I_m$, $x = \mathrm{vec}(X)$, $y = \mathrm{vec}(C)$ (see also Section 4). These formulations are only efficient to use explicitly when solving small-sized problems in kernel solvers, see, e.g., LAPACK's `DLASY2` and `DTGSY2` for solving SYCT and GCSY and the kernels of the RECSY library [15,16,17].

3 Parallel Algorithms for Triangular Matrix Equations

The parallel algorithms for SYCT presented in [24,10,8,9] were based on the following basic ideas: Utilize explicit blocking and 2D block cyclic distribution of the matrices over a rectangular $P_r \times P_c$ process grid, following the ScaLAPACK conventions [3], and compute the solution by a wavefront-like traversal of the block diagonals of the right hand side matrix where several solutions of diagonal subsystems are computed in parallel, broadcasted along the corresponding block rows and columns, and used in level 3 updates of the rest of the right hand side. This is illustrated for SYCT in Figure 1. Notice that the solution X overwrites the right hand side C blockwise.

The algorithms are adapted to the symmetric LYCT by wavefront-like traversal of the *anti-diagonals* of the right hand side matrix while solving for the lower (or upper) triangular part of the solution. The situation is described in Figure 2. However, our solvers must be able to solve non-symmetric LYCT problems as well since symmetry cannot be assumed in condition estimation algorithms (see Section 4).

For two-sided standard matrix equations SYDT/LYDT the main difference from the SYCT/LYCT cases are the need for an extra buffer for storing *intermediate sums of matrix products* caused by a more complex data dependency (see equations (12)-(13)) which will, assuming $m = n$, cause any trivially blocked solver to use $O(n^4)$ flops. We illustrate with the following explicitly blocked SYDT system:

$$\begin{cases} A_{11}X_{11}B_{11}^T - X_{11} = C_{11} - A_{11}X_{12}B_{12}^T - A_{12}(X_{21}B_{11}^T + X_{22}B_{12}^T) \\ A_{11}X_{12}B_{22}^T - X_{12} = C_{12} - A_{12}X_{22}B_{22}^T \\ A_{22}X_{21}B_{11}^T - X_{21} = C_{21} - A_{22}X_{22}B_{12}^T \\ A_{22}X_{22}B_{22}^T - X_{22} = C_{22}. \end{cases} \tag{14}$$

From (14) we observe that by computing $X_{21}B_{11}^T + X_{22}B_{12}^T$ before multiplying with A_{12} and by computing $X_{22}B_{12}^T$ only once we avoid redundant computations. Consequently, for SYDT/LYDT we broadcast each subsolution X_{ij} in the process row corresponding to block row i and a sum of matrix products in the process column corresponding to block column j.

The generalized matrix equations are solved as follows: for GCSY the SYCT methodology is used except for the fact that we are now working with two

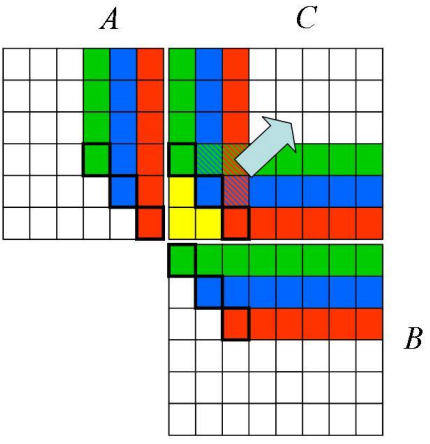

Fig. 1. The SYCT wavefront: standard, one-sided, non-symmetric. Yellow blocks correspond to already solved blocks, the blocks with bold borders correspond to the current position of the wavefront, blocks with the same color are used together in subsystems solves or GEMM-updates, stripe-colored blocks are involved in several rounds of GEMM-updates corresponding to the same block diagonal. The wavefront direction is indicated by the arrow. Each subsolution is broadcasted in the corresponding block row and column.

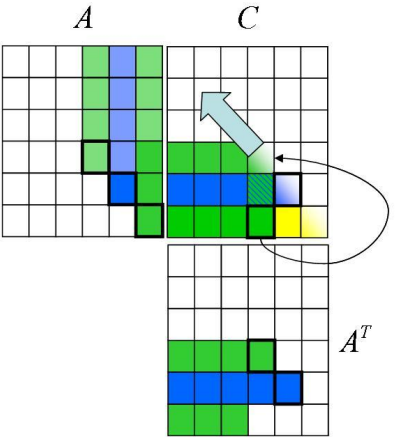

Fig. 2. The symmetric LYCT wavefront: standard, one-sided, symmetric. Each subsolution (i.e., a block of the solution matrix X) located outside the main block anti-diagonal is broadcasted in the corresponding block row and column and the block row corresponding to its transposed position.

equations at the same time. The methods of SYDT and LYDT are generalized for GSYL and GLYCT/GLYDT, respectively, in a similar fashion by using two extra buffers for storing intermediate sums of matrix products.

We remark that in a trivially blocked solver for the two-sided Lyapunov equations, we may reformulate the updates of the main block diagonal of C in terms of SYR2K-operation, as in the LYCT case. However, this is not possible when we use the intermediate sums of matrix products to reduce the complexity.

4 Condition Estimators for Triangular Matrix Equations

We utilize a general method [11,12,20] for estimating $\|A^{-1}\|_1$ for a square matrix A using reverse communication of $A^{-1}x$ and $A^{-T}x$, where $\|x\|_2 = 1$. In

particular, for SYCT this approach is based on linear system $Z_{\mathrm{SYCT}}x = y$ (see Section 1 and Table 1) which is used to compute a lower bound of the inverse of the *separation between the matrices A and B* [27]:

$$\mathrm{sep}(A,B) = \inf_{\|X\|_F=1} \|AX - XB\|_F = \sigma_{min}(Z_{\mathrm{SYCT}}) = \|Z_{\mathrm{SYCT}}^{-1}\|_2^{-1}. \qquad (15)$$

The quantity (15) is used frequently in perturbation theory and error bounds (see, e.g., [13]). The exact value can be computed at the cost $O(m^3n^3)$ flops by the SVD of Z_{SYCT} but its inverse can be estimated much cheaper by solving a few (normally around five) triangular SYCT equations to the cost $O(m^2n+mn^2)$ flops [20].

Table 1. The Kronecker product representations of $Z_\star$ and $Z_\star^T$ considered in condition estimation of the standard and generalized matrix equations (1)-(8)

Acronym (ACRO)	$Z_\star$	$Z_\star^T$
SYCT	$I_n \otimes A - B^T \otimes I_m$	$I_n \otimes A^T - B \otimes I_m$
SYDT	$B \otimes A - I_{m\cdot n}$	$B^T \otimes A^T - I_{m\cdot n}$
LYCT	$I_m \otimes A + A \otimes I_m$	$I_m \otimes A^T - A^T \otimes I_m$
LYDT	$A \otimes A - I_{m^2}$	$A^T \otimes A^T - I_{m^2}$
GCSY	$\begin{bmatrix} I_n \otimes A & -B^T \otimes I_m \\ I_n \otimes D & -E^T \otimes I_m \end{bmatrix}$	$\begin{bmatrix} I_n \otimes A^T & I_n \otimes D^T \\ -B \otimes I_m & -E \otimes I_m \end{bmatrix}$
GSYL	$B \otimes A - D \otimes C$	$B^T \otimes A^T - D^T \otimes C^T$
GLYCT	$A \otimes A - E \otimes E$	$A^T \otimes A^T - E^T \otimes E^T$
GLYDT	$E \otimes A + A \otimes E$	$E^T \otimes A^T + A^T \otimes E^T$

This estimation method is applied to all matrix equations by considering the corresponding Kronecker product representation of the associated Sylvester-type operator (see Table 1). However, notice that condition estimation of GCSY is not as straightforward as for the uncoupled equations, since transposing Z_{GCSY} is not just a matter of transposing all involved left hand side matrices (excluding the solution), but requires a different algorithm (see, e.g., [21]).

The condition estimator in [20] was based on the serial LAPACK-routine `DLACON` [1]. The parallel version we use is implemented in ScaLAPACK [3,26] as the auxiliary routine `PDLACON`.

In our parallel estimators, we compute P_c different estimates independently and concurrently, one for each process column by taking advantage of the fact that `PDLACON` requires a column vector distributed over a single process column as right hand side, and we form the global maximum by a scalar *all-to-all reduction* [7] in each process row (which is negligible in terms of execution time). The column vector y in each process column is constructed by performing an *all-to-all broadcast* [7] of the local pieces of the right hand side matrix or matrices in each process row, forming P_c different right hand side vectors. Altogether, we compute P_c different estimates (lower bounds of the associated sep^{-1}-function) and choose the largest value at the same cost in time as computing only one estimate.

5 Experimental Results

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 [23], LAPACK 3.0 [22], GOTO-BLAS r0.94 [6], ScaLAPACK 1.7.0 [26], BLACS 1.1patch3 [4] and RECSY 0.01alpha [25] (used as node solvers). All experiments are conducted in double precision arithmetic.

Table 2. Condition estimation of GSYL invoking `PGSYLCON` on *sarek* using the blocksize 64. All timings are in seconds. For this table, (A, C) and (B, D) are chosen as random upper triangular matrices with specified eigenvalues as $\lambda^{(i)}_{(A,C)} = i$ and $\lambda^{(i)}_{(B,D)} = -i$, respectively. The known solution X is a random matrix with uniform distribution in the interval $[-1, 1]$.

$m = n$	$P_r \times P_c$	Time	*iter*	*est*	R_a	R_r	E_a	E_r
1024	1×1	12.7	5	0.6E-03	0.2E-06	0.1E+01	0.1E-11	0.2E-14
1024	2×2	7.4	5	0.6E-03	0.1E-06	0.1E+01	0.1E-11	0.2E-14
1024	4×4	3.9	5	0.6E-03	0.1E-06	0.1E+01	0.1E-11	0.2E-14
1024	8×8	2.1	5	0.6E-03	0.1E-06	0.1E+01	0.1E-11	0.2E-14
2048	1×1	86.6	5	0.3E-03	0.1E-05	0.1E+01	0.3E-11	0.2E-14
2048	2×2	48.3	5	0.3E-03	0.1E-05	0.1E+01	0.2E-11	0.2E-14
2048	4×4	21.9	5	0.3E-03	0.1E-05	0.1E+01	0.3E-11	0.2E-14
2048	8×8	9.7	5	0.3E-03	0.1E-05	0.1E+01	0.2E-11	0.2E-14
4096	1×1	923.9	7	0.2E-03	0.1E-04	0.1E+01	0.7E-11	0.3E-14
4096	2×2	503.3	7	0.2E-03	0.1E-04	0.1E+01	0.6E-11	0.2E-14
4096	4×4	193.8	7	0.2E-03	0.1E-04	0.1E+01	0.6E-11	0.2E-14
4096	8×8	77.5	7	0.2E-03	0.1E-04	0.1E+01	0.8E-11	0.3E-14
8192	1×1	5302.4	5	0.8E-04	0.1E-03	0.1E+01	0.1E-10	0.3E-14
8192	2×2	2625.9	5	0.8E-04	0.1E-03	0.1E+01	0.1E-10	0.3E-14
8192	4×4	904.5	5	0.8E-04	0.1E-03	0.1E+01	0.1E-10	0.3E-14
8192	8×8	331.4	5	0.8E-04	0.1E-03	0.1E+01	0.1E-10	0.3E-14

In Table 2, we present performance results for the parallel GSYL condition estimator `PGSYLCON` solving well-conditioned problems using the corresponding parallel triangular GSYL solver `PTRGSYLD`. For this table, *iter* is the number of iterations and calls to the triangular solver `PTRGSYLD`, *est* is the lower bound estimate of sep^{-1}[GSYL], R_a, R_r, E_a and E_r correspond to the absolute and relative residual and error norms and are computed as follows:

$$R_a = \|E - A\tilde{X}B + C\tilde{X}D\|, \tag{16}$$

$$R_r = (\epsilon^{-1}_{\mathrm{mach}} R_a)/((\|A\|\|B\| + \|C\|\|D\|)\|\tilde{X}\| + \|E\|), \tag{17}$$

$$E_a = \|X - \tilde{X}\|, \tag{18}$$

$$E_r = E_a/\|X\|. \tag{19}$$

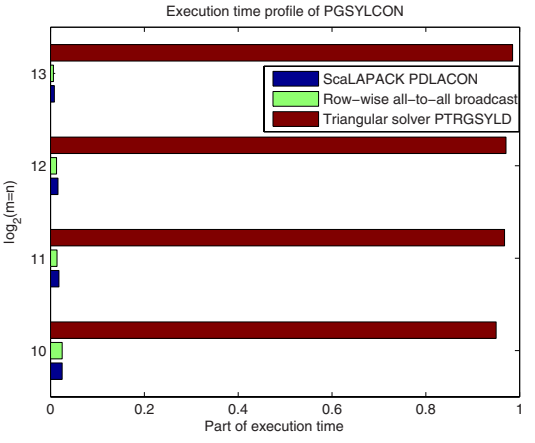

Fig. 3. Execution time profile of `PGSYLCON` on *sarek* using the blocksize 64. The results are typical for what we found using multiple (4×4) processors to solve the problem.

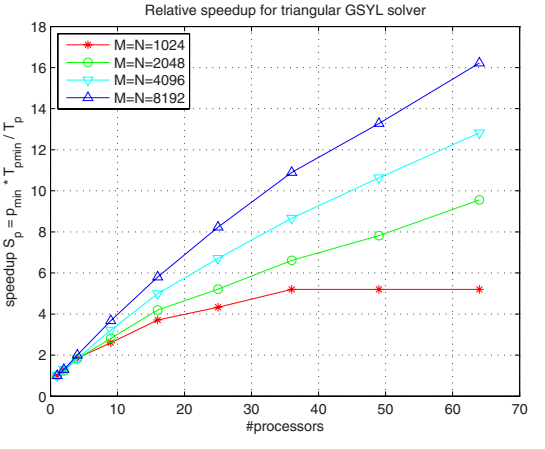

Fig. 4. Parallel speedup of `PTRGSYLD` on *sarek* using the blocksize 64

Here, $\epsilon_{\text{mach}} (\approx 2.2 \times 10^{-16})$ is the relative machine precision and X and $\tilde{X}$ are the known and the computed solutions, respectively. The relative residual norm is computed in the 1-norm and the absolute residual and the error norms are computed in the Frobenius norm, respectively. Ideally, the relative residual norm should be of $O(1)$ [21], which is fulfilled remarkably well for this set of test problems. The high absolute residual norm results emerge from the large norms $(O(n^{3/2}))$ of the known left hand side matrices. The stable value of *est* depends on that exactly the same problem is generated and solved for every value of $m = n$.

An execution time profile of PGSYLCON is presented in Figure 3. The major part of the work is spent in the triangular solver, which is called around five (5) times (see Table 2). The influences of PDLACON and the all-to-all broadcast of the right hand side in each process row on the total execution time are diminished as the problem size grows. This implies that any effort on improving the condition estimator should concentrate on the triangular solver.

A representative selection of parallel speedup results for the triangular GSYL solver is presented in Figure 4. The algorithms for the other equations (see Section 1) have similar good qualitative behavior.

Acknowledgements

This research was conducted using the resources of the High Performance Computing Center North (HPC2N) [14]. 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. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J.W., Dongarra, J.J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.C.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
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. BLACS - Basic Linear Algebra Communication Subprograms. See http://www.netlib.org/blacs/index.html
5. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
6. GOTO-BLAS - High-Performance BLAS by Kazushige Goto. See http://www.cs.utexas.edu/users/flame/goto/
7. Grama, A., Gupta, A., Karypsis, G., Kumar, V.: Introduction to Parallel Computing, 2nd edn. Addison-Wesley, Reading (2003)
8. Granat, R., Jonsson, I., Kågström, B.: Combining Explicit and Recursive Blocking for Solving Triangular Sylvester-Type Matrix Equations in Distributed Memory Platforms. In: Danelutto, M., Vanneschi, M., Laforenza, D. (eds.) Euro-Par 2004. LNCS, vol. 3149, pp. 742–750. Springer, Heidelberg (2004)
9. Granat, R., Kågström, B.: Evaluating Parallel Algorithms for Solving Sylvester-Type Matrix Equations: Direct Transformation-Based versus Iterative Matrix-Sign-Function-Based Methods. In: Dongarra, J.J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 719–729. Springer, Heidelberg (2006)
10. Granat, R., Kågström, B., Poromaa, P.: Parallel ScaLAPACK-style Algorithms for Solving Continuous-Time Sylvester Equations. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 800–809. Springer, Heidelberg (2003)

11. Hager, W.W.: Condition estimates. SIAM J. Sci. Statist. Comput. (3), 311–316 (1984)
12. 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)
13. Higham, N.J.: Perturbation theory and backward error for $AX - XB = C$. BIT 33(1), 124–136 (1993)
14. HPC2N - High Performance Computing Center North. See `http://www.hpc2n.umu.se`
15. 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)
16. Jonsson, I., Kågström, B.: Recursive blocked algorithms for solving triangular systems. II. Two-sided and generalized Sylvester and Lyapunov matrix equations. ACM Trans. Math. Software 28(4), 416–435 (2002)
17. 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)
18. Kågström, B., Ling, P., Van Loan, C.: GEMM-Based Level 3 BLAS: High-Performance Model Implementations and Performance Evaluation Benchmark. ACM Trans. Math. Software 24(3), 268–302 (1998)
19. Kågström, B., Ling, P., Van Loan, C.: Algorithm 784: GEMM-Based Level 3 BLAS: Portability and Optimization Issues. ACM Trans. Math. Software 24(3), 303–316 (1998)
20. 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)
21. Kågström, B., Poromaa, P.: 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 (1996)
22. LAPACK - Linear Algebra Package. See `http://www.netlib.org/lapack/`
23. MPI - Message Passing Interface. See `http://www-unix.mcs.anl.gov/mpi/`
24. Poromaa, P.: Parallel Algorithms for Triangular Sylvester Equations: Design, Scheduling and Scalability Issues. In: Kagström, B., Elmroth, E., Waśniewski, J., Dongarra, J.J. (eds.) PARA 1998. LNCS, vol. 1541, pp. 438–446. Springer, Heidelberg (1998)
25. RECSY - High Performance library for Sylvester-type matrix equations, See `http://www.cs.umu.se/research/parallel/recsy`
26. ScaLAPACK Users' Guide, see `http://www.netlib.org/scalapack/slug/`
27. Stewart, G.W., Sun, J.-G.: Matrix Perturbation Theory. Academic Press, New York (1990)