Combining Explicit and Recursive Blocking for Solving Triangular Sylvester-Type Matrix Equations on Distributed Memory Platforms

EuroPar 2004 contribution

Robert Granat, Isak Jonsson & Bo Kågström,

Umeå University and HPC2N, Sweden

Outline

- Triangular Sylvester-type matrix equations
- ScaLAPACK-style algorithms
- RECSY – recursive blocked algorithms and HPC library
- Experimental results
- Conclusions and summary
- Future work
- References

Triangular Sylvester-type matrix equations (1)

Arise in many different applications in science and engineering

- Condition estimation of eigenvalue problems
- Control theory

Name	Matrix Equation
Standard Sylvester (CT)	$AX - XB = C$
Standard Lyapunov (CT)	$AX + XA^T = C$
Standard Sylvester (DT)	$AXB^T - X = C$
Standard Lyapunov (DT)	$AXA^T - X = C$
Generalized Coupled Sylvester	$(AX - YB, DX - YE) = (C, F)$
Generalized Sylvester	$AXB^T - CXD^T = E$
Generalized Lyapunov (CT)	$AXE^T + EXA^T = C$
Generalized Lyapunov (DT)	$AXA^T - EXE^T = C$

Consider the continuous-time Sylvester equation (SYCT):

$$AX - XB = C, \quad A \in R^{m\times m}, B \in R^{n\times n}, C \in R^{m\times n}$$

A and *B* in real Schur form

Solution is unique iff $\lambda(A) \cap \lambda(B) = \varnothing$

Triangular Sylvester-type matrix equations (2)

- SYCT equivalent to linear system of equations:

$$\begin{cases} Z_{SYCT}\, vec(X) = vec(C), \\ Z_{SYCT} = I_n \otimes A - B^T \otimes I_m \end{cases}$$

- Explicit Kronecker product representation used in kernel solvers, e.g., in blocked algorithms:

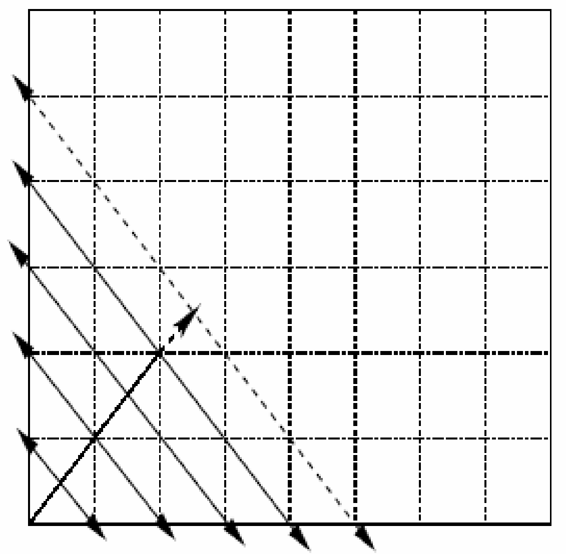
$$A_{ii}X_{ij} - X_{ij}B_{jj} = C_{ij} - \left(\sum_{k=i+1}^{D_a} A_{ik}X_{kj} - \sum_{k=1}^{j-1} X_{ik}B_{kj} \right)$$

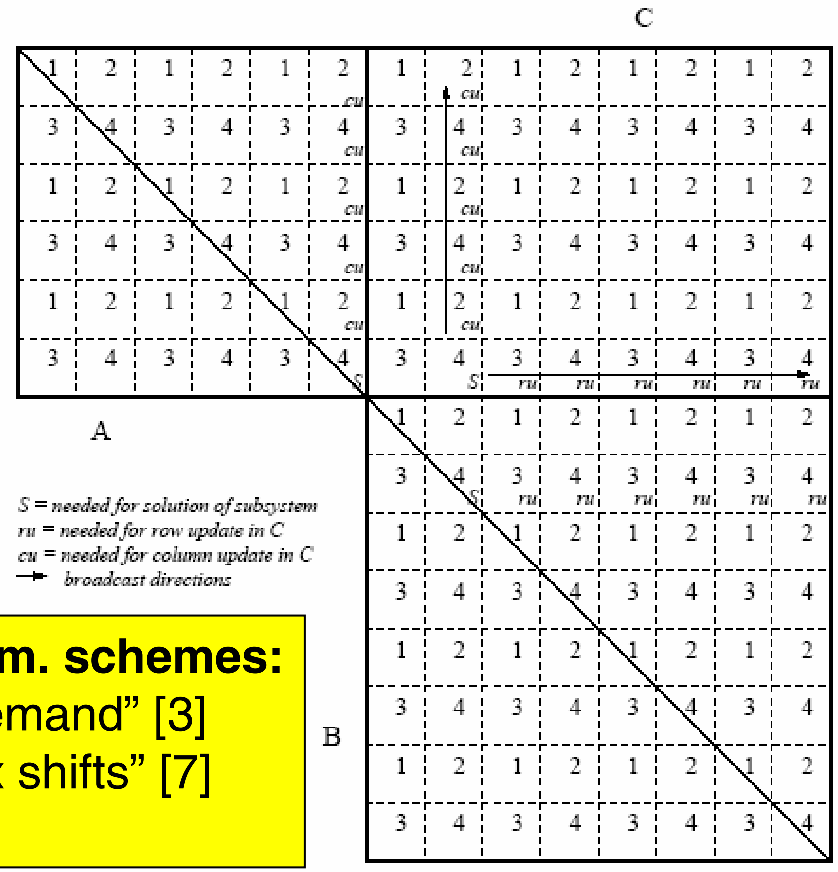
```
for j=1, D_b
    for i=D_a, 1, -1
        {Solve the (i, j)th subsystem using a kernel solver}
        A_ii X_ij − X_ij B_jj = C_ij
        for k=1, i − 1
            {Update block column j of C}
            C_kj = C_kj − A_ki X_ij
        end
        for k=j + 1, D_b
            {Update block row i of C}
            C_ik = C_ik + X_ij B_jk
        end
    end
end
```

With appropriate blocking, cost is dominated by GEMM-updates of right hand side

ScaLAPACK-style algorithms (1)

- Rectangular process grid
- 2D block cyclic mapping
- Wave-front matrix traversal
- LAPACK's DTRSYL kernel solver (ess. Level-2)





Two comm. schemes:

1. "On demand" [3]
2. "Matrix shifts" [7]

ScaLAPACK-style algorithms (2)

- Communication operations (”On demand”):
  1. Implicit redistribution [3] (splitted 2x2 diagonal blocks)
     - Minor impact on total execution time
  2. Solving subsystem *(i,j)*
     - Minor impact on total execution time
  3. Broadcast of subsolution *(i,j)* in block row *i* and block column *j*
     - Bottleneck – causes idle processes waiting for BC to start
  4. GEMM-updates of right hand side
     - Bottleneck – causes idle processes
- ”Matrix shifts” approach combines (2) and (4) into one single (expensive) operation but suffers from broadcast bottleneck
- ”OD” allows single transposes in equation, whereas ”MS” does not

- This contribution attempts to minimize (3) by means of a faster kernel node solver thus improving the total execution time – and creating ScaLAPACK-style hybrid algorithms

RECSY – recursive blocked algorithms (1)

- HPC library which solves 42 sign- and transpose variants of 8 Sylvester-type matrix equations
  - Recursive blocking approach
  - Automatic variable-sized blocking matches memory hierarchies of today´s modern computers
  - Significant speedup compared to standard library routines
    - For example SYCT solver RECSYCT: 10-folded speedup compared to LAPACK's DTRSYL
- Based on work by Jonsson-Kågström [4,5,6]
- Library was presented at EuroPar 2003.
- Documentation, download and installation instructions at http://www.cs.umu.se/research/parallel/recsy

RECSY – recursive blocked algorithms (2)

- Recursive blocking makes RECSY rich in GEMM-operations => high performance (if good GEMM impl. is available)
- Keep subsystems ”squarish” by testing dimensions in each step of the recursion. At leaf node apply superscalar kernel.
- For example n/2 < m < 2n:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} - \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$\Leftrightarrow$$

$$\begin{aligned} A_{11}X_{11} - X_{11}B_{11} &= C_{11} - A_{12}X_{21}, \\ A_{11}X_{12} - X_{12}B_{22} &= C_{12} - A_{12}X_{22} + X_{11}B_{12}, \\ A_{22}X_{21} - X_{21}B_{11} &= C_{21}, \\ A_{22}X_{22} - X_{22}B_{22} &= C_{22} + X_{21}B_{12}. \end{aligned}$$

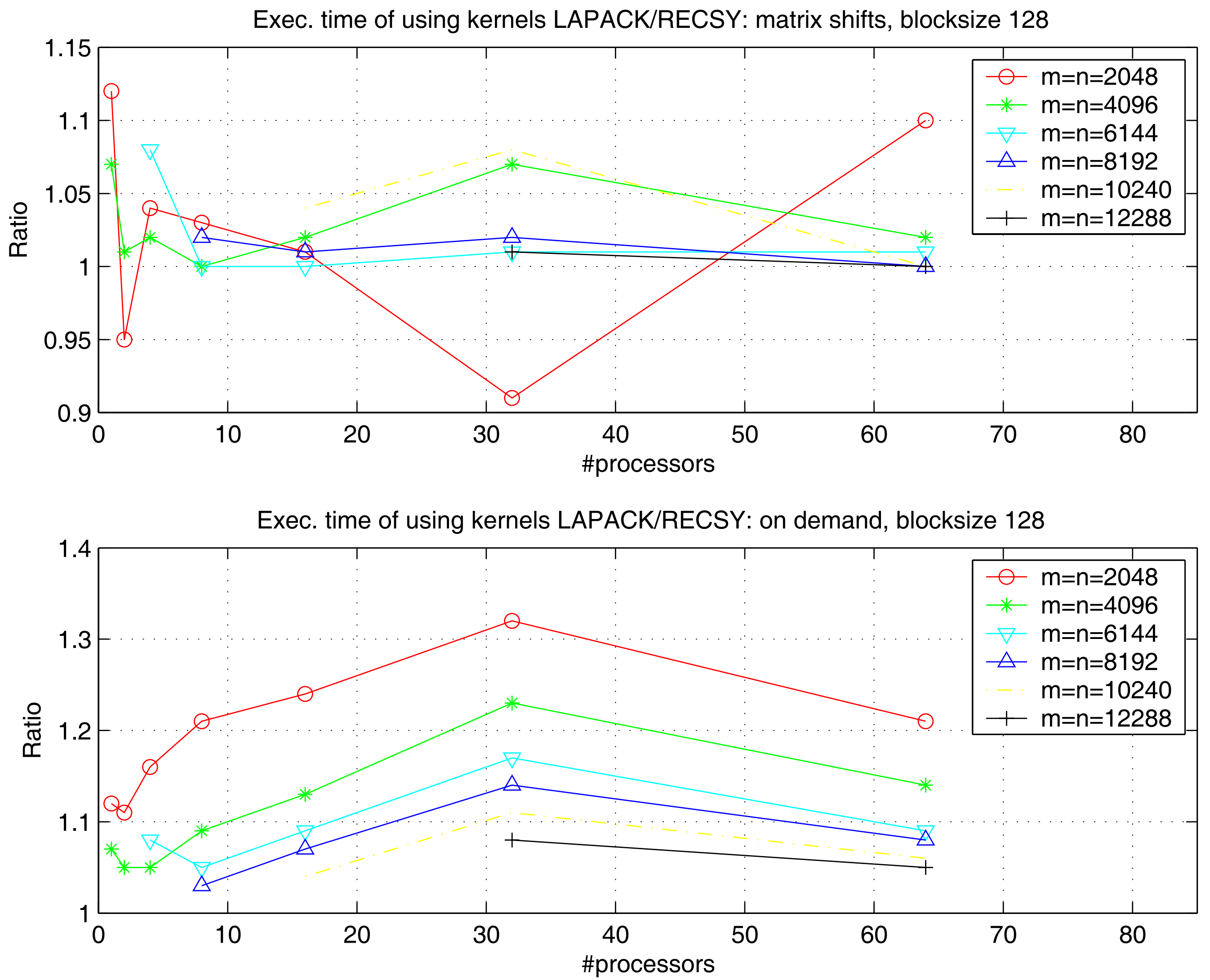
RECSY template – applied recursively to all subsystems

1. Solve for X_21 (recursively)
2. Update C_11 and C_22
3. Solve for X_11 and X_22 (recursively)
4. Update C_12
5. Solve for X_12 (recursively)

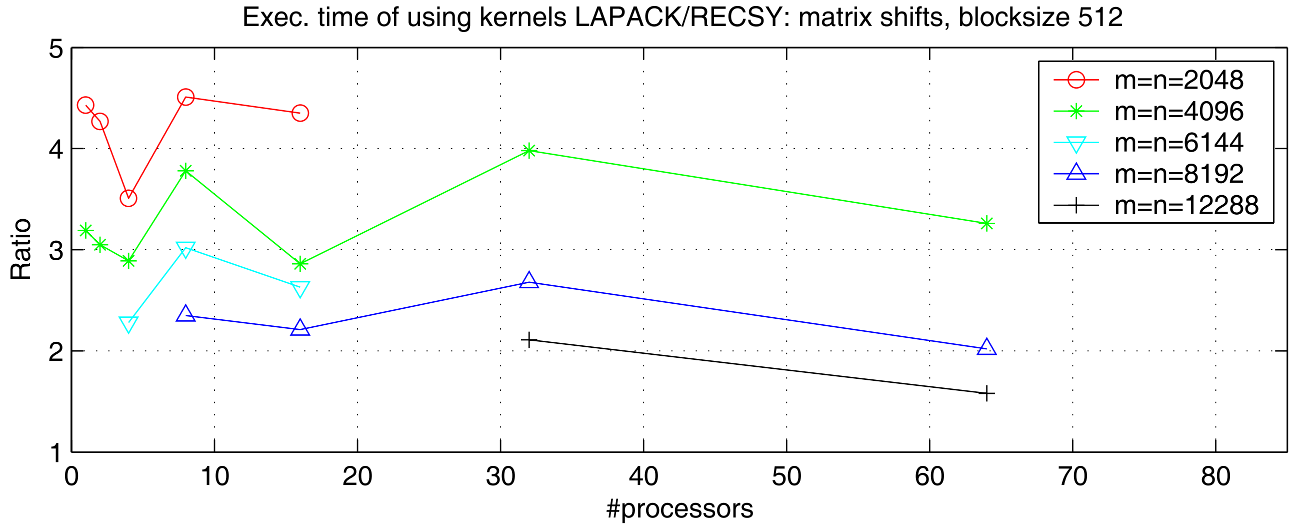
Experimental results (1)

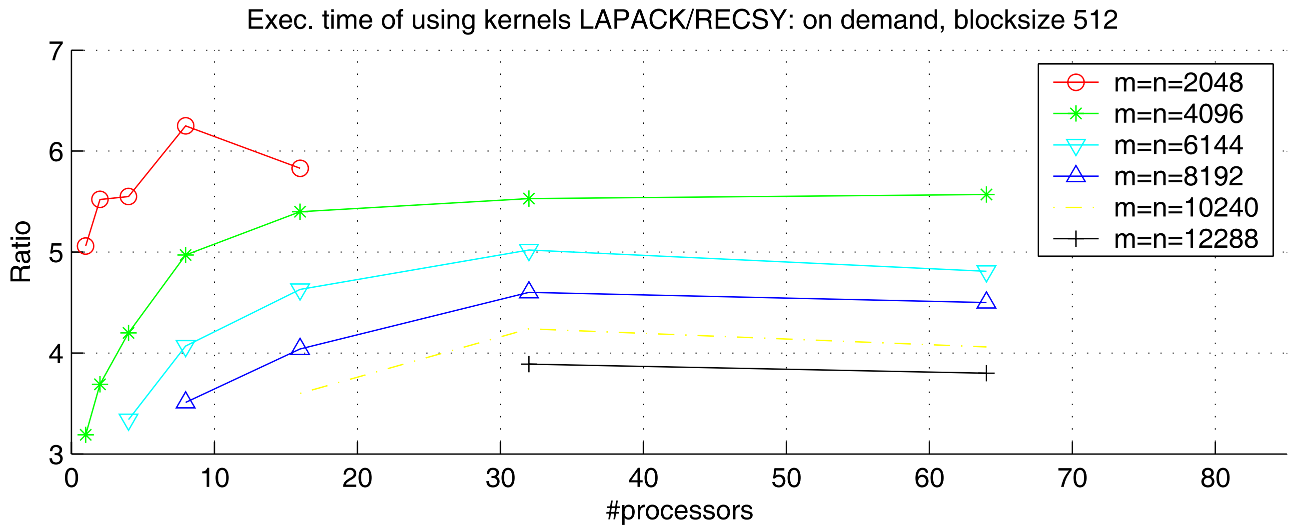
- Comparing ScaLAPACK-style SYCT-solving algorithm with LAPACK or RECSY node solvers
- Target computer system: HPC2N Linux Super Cluster
  - 120 dual 1.667MHz nodes with 1GB RAM
  - 667 Mbyte/sec peak network bandwidth
  - Quite unbalanced
    - Time(flop) / (Memory bandwith)^(-1) ratio: 0.31
    - Time(flop) / (Network bandwidth)^(-1) ratio: 0.10
  - Efficient use of memory hierarchy and network necessary for good performance
- We present ratios of the parallel execution time with the node solvers from LAPACK and RECSY, respectively, for both communication schemes going from normal blocksize (128) to large (512)
  - Ratio > 1.0 represents speedup when going from LAPACK to RECSY

Experimental results (2)

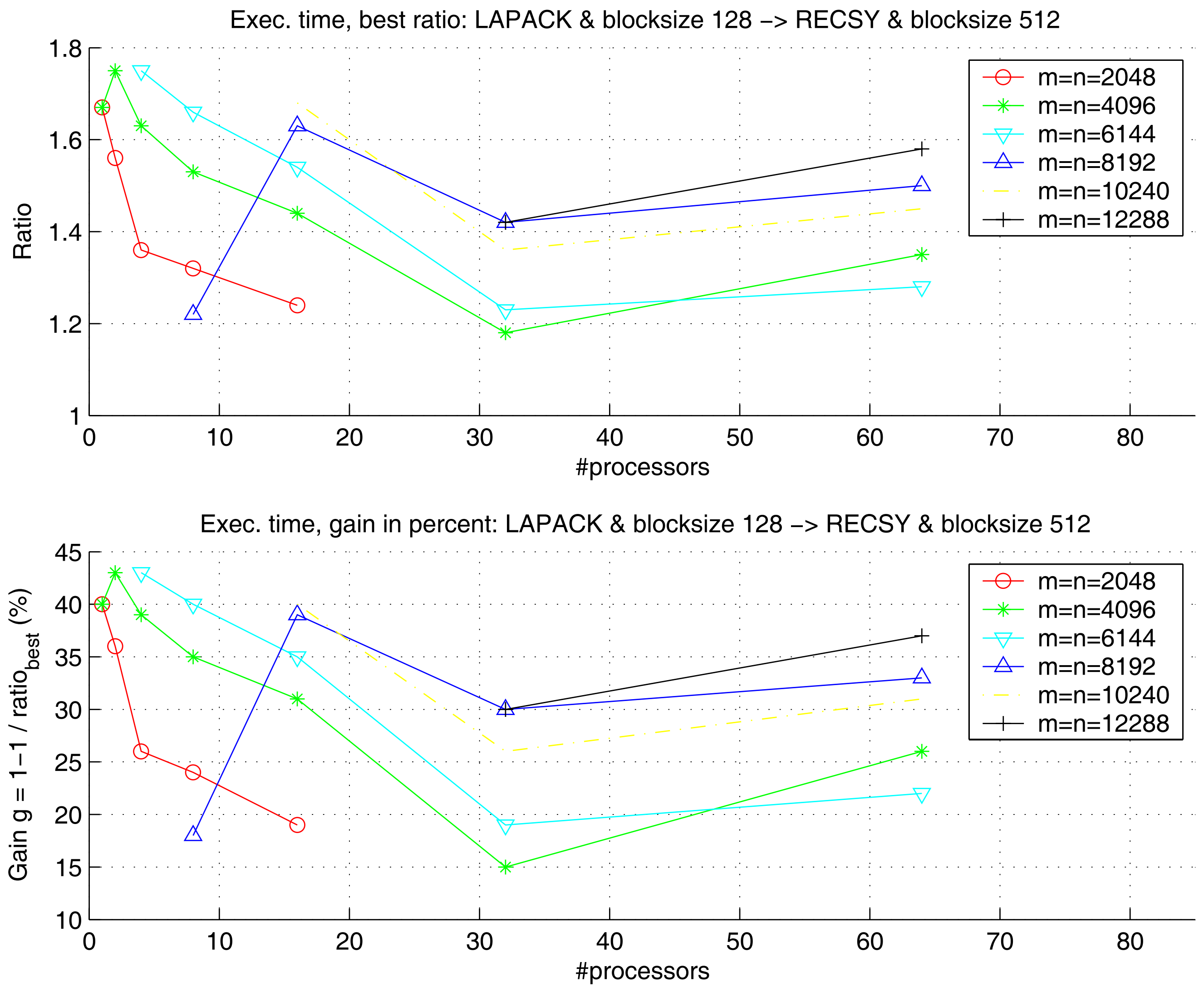


Experimental results (3)





Experimental results (4)



Conclusions and summary

- RECSY solver improves overall parallel performance by 15-43% for the results presented (Notice the misprint in contribution abstract!)
  - Allows larger blocks in the parallel algorithm
    - Close(r) to GEMM-performance for subsystem solves
    - Decreases synchronization cost for both ”On demand” and ”Matrix shifts”
    - But affects scalability (harder to hide comm. during comp.)
- Improves comparision with iterative methods [2] on unbalanced systems
- With RECSY both communications schemes perform equally good
- With LAPACK:
  - ”On demand” best for small blocks
  - ”Matrix shifts” best for larger blocks
- RECSY's LAPACK/SLICOT wrappers allow the user to choose kernel node solver without modifying source code

Future Work

- New implementations for all transpose and sign-variants of the non-generalized standard matrix equations
- User may choose communication scheme
- Future software package *SCASY* will also contain generalized solvers and parallel condition estimators
- By linking with RECSY, hybrid algorithms come for free

$op(A)X \pm Xop(B) = C$	SYCT	√
$op(A)X + Xop(A^T) = C$	LYCT	√
$op(A)Xop(B) \pm X = C$	SYDT	√
$op(A)Xop(A^T) - X = C$	LYDT	√
$\begin{cases} op(A)X \pm Yop(B) = C, \\ op(D)X \pm Yop(E) = F \end{cases}$	GCSY	
$op(A)Xop(B) \pm op(D)Xop(E) = C$	GSYL	
$op(A)Xop(A^T) - op(E)Xop(E^T) = C$	GLYCT	
$op(A)X(E^T) + op(E)Xop(A^T) = C$	GLYDT	

References

[1] R.H. Bartels and G.W. Stewart. Algorithm 432: Solution of the Equation AX+XB = C, *Comm. ACM*, 15(9):820-826, 1972.

[2] Robert Granat and Bo Kågström. Evaluating Parallel Algorithms for Solving Sylvester-Type Matrix Equations: Direct Transformation-Based versus Iterative Matrix-Sign-Function-Based Methods. To appear in *PARA'04 State-of-the-Art in Scientific Computing Conference Proceedings*, *LCNS*, Springer Verlag, 2004.

[3] R. Granat, B. Kågström, P. Poromaa. Parallel ScaLAPACK-style Algorithms for Solving Continuous-Time Sylvester Equations. In H. Kosch et al. (eds), Euro-Par 2003 Parallel Processing. *Lecture Notes in Computer Science*, Vol. 2790, pp. 800-809, 2003.

[4] I. Jonsson and B. Kågström, Recursive Blocked Algorithms for Solving Triangular Matrix Equations - Part I: One-Sided and Coupled Sylvester-Type Equations, *ACM Trans. Math. Software*, Vol. 28, No. 4, pp 393-415, 2002.

[5] I. Jonsson and B. Kågström, Recursive Blocked Algorithms for Solving Triangular Matrix Equations - Part II:Two-Sided and Generalized Sylvester and Lyapunov Equations, *ACM Trans. Math. Software*, Vol. 28, No. 4, pp 416-435, 2002.

[6] I. Jonsson and B. Kågström, RECSY – A High Performance Library for Solving Sylvester-Type Matrix Equations, In H. Kosch et al. (editors), *Euro-Par 2003 Parallel Processing*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2790, pp. 810-819, 2003.

[7] P. Poromaa. Parallel Algorithms for Triangular Sylvester Equations: Design, Scheduling and Scalability Issues. In Kågström et al. (eds), *Applied Parallel Computing. Large Scale Scientific and Industrial Problems, Lecture Notes in Computer Science*, Vol. 1541, pp. 438-446, Springer-Verlag, 1998.