# Recursive Blocked Algorithms for Solving Triangular Systems—Part II: Two-Sided and Generalized Sylvester and Lyapunov Matrix Equations

ISAK JONSSON and BO KÅGSTRÖM
Umeå University

We continue our study of high-performance algorithms for solving triangular matrix equations. They appear naturally in different condition estimation problems for matrix equations and various eigenspace computations, and as reduced systems in standard algorithms. Building on our successful recursive approach applied to one-sided matrix equations (Part I), we now present novel recursive blocked algorithms for two-sided matrix equations, which include matrix product terms such as $AXB^T$. Examples are the discrete-time standard and generalized Sylvester and Lyapunov equations. The means for achieving high performance is the recursive variable blocking, which has the potential of matching the memory hierarchies of today's high-performance computing systems, and level-3 computations which mainly are performed as GEMM operations. Different implementation issues are discussed, including the design of efficient new algorithms for two-sided matrix products. We present uniprocessor and SMP parallel performance results of recursive blocked algorithms and routines in the state-of-the-art SLICOT library. Although our recursive algorithms with optimized kernels for the two-sided matrix equations perform more operations, the performance improvements are remarkable, including 10-fold speedups or more, compared to standard algorithms.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*conditioning, linear systems*; G.4 [**Mathematical Software**]: Algorithm design and analysis, efficiency, parallel and vector implementations, portability, reliability and robustness

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Matrix equations, standard discrete-time Sylvester and Lyapunov, generalized Sylvester and Lyapunov, recursion, automatic blocking, superscalar, GEMM-based, level-3 BLAS, SMP parallelization, LAPACK, SLICOT

This research was conducted using the resources of the High Performance Computing Center North (HPC2N) and PDC-Parallelldatorcentrum at KTH, Stockholm. Financial support was provided by the Swedish Research Council under grants TFR 98-604 and VR 621-2001-3284.

Authors' address: Department of Computing Science and HPC2N, Umeå University, SE-901 87 Umeå, Sweden; email: {isak,bokg}@cs.umu.se.

## 1. INTRODUCTION

We continue the presentation of recursive blocked algorithms for solving various types of triangular matrix equations. Our goal is to design efficient algorithms for today's HPC systems with multilevel memory hierarchies. The hierarchical recursive blocking promotes good data locality and combined with our highly optimized superscalar kernels, we obtain a notable boost in performance compared to existing algorithms as implemented in state-of-the-art libraries [Anderson et al. 1999; SLICOT 2001]. In Part I [Jonsson and Kågström 2002], we introduced and discussed new recursive blocked algorithms for *one-sided* Sylvester-type equations, including the continuous-time standard Sylvester and Lyapunov equations, and a generalized coupled Sylvester equation.

In this contribution (Part II), we introduce and discuss new recursive blocked algorithms for *two-sided* matrix equations, which include matrix product terms of type $\text{op}(A)X\,\text{op}(B)$, where $\text{op}(Y)$ can be $Y$ or its transpose $Y^T$. Examples include discrete-time standard and generalized Sylvester and Lyapunov equations. The standard methods for solving two-sided matrix equations are also based on the Bartels–Stewart method [Bartels and Stewart 1972]. As in Part I, we focus on the solution of the two-sided triangular counterparts, which typically are obtained after an initial transformation of matrices (or regular matrix pairs) to Schur (or generalized Schur) form [Anderson et al. 1999; Dackland and Kågström 1999].

Before we go into any further details, we outline the contents of the rest of the article. In Section 2, we introduce Sep-functions associated with the two-sided matrix equations and reestablish the relationship between the solution of triangular matrix equations and condition estimation. Section 3 introduces our recursive blocked algorithms for two-sided matrix equations, including the generalized Sylvester equation (Section 3.1), the discrete-time Sylvester equation (Section 3.2), the generalized and standard discrete-time Lyapunov equations (Sections 3.3 and 3.4), and finally the generalized continuous-time Lyapunov equation (Section 3.5). In Section 4, we revisit our discussion about implementation issues, now focusing on the design of optimized two-sided matrix product kernels. Sample performance results of our recursive blocked algorithms are presented and discussed in Section 5. Finally, we give some concluding remarks in Section 6.

## 2. CONDITION ESTIMATION OF MATRIX EQUATIONS REVISITED

The two-sided matrix equations can also be written as a linear system of equations $Zx = c$, where $Z$ is a *Kronecker product matrix representation* of the associated matrix equation operator. The solution $x$ and the right-hand side $c$ are represented in $\text{vec}(\cdot)$ notation, where $\text{vec}(X)$ denotes a column vector with the columns of $X$ stacked on top of each other.

We introduce the following $Z$-matrices.

$$\begin{aligned}
Z_{\text{SYDT}} &= Z_{AXB^T-X} = B \otimes A - I_n \otimes I_m,\\
Z_{\text{LYDT}} &= Z_{AXA^T-X} = A \otimes A - I_n \otimes I_n,\\
Z_{\text{GSYL}} &= Z_{AXB^T-CXD^T} = B \otimes A - D \otimes C,
\end{aligned}$$

$$\begin{aligned} Z_{\mathrm{GLYCT}} &= Z_{AXE^T+EXA^T} = E \otimes A + A \otimes E, \\ Z_{\mathrm{GLYDT}} &= Z_{AXA^T-EXE^T} = A \otimes A - E \otimes E, \end{aligned}$$

where from top to bottom they represent the matrix operators of the discrete-time Sylvester and Lyapunov equations, the generalized (discrete/continuous-time) Sylvester equation, and the generalized continuous-time and discrete-time Lyapunov equations.

Similar techniques to those described in the Part I article [Jonsson and Kågström 2002] can be used for estimating the conditioning of two-sided matrix equations. This leads to solving triangular two-sided matrix equations for estimating the Sep-function:

$$\mathrm{Sep}[\cdot] = \|Z^{-1}\|_2^{-1} = \sigma_{\min}(Z),$$

where $Z$ is any of the Kronecker product matrices above. The Sep-functions associated with the two-sided matrix equations considered are:

$$\begin{aligned} \mathrm{Sep[SYDT]} &= \inf_{\|X\|_F=1} \|AXB^T - X\|_F &&= \sigma_{\min}(Z_{\mathrm{SYDT}}), \\ \mathrm{Sep[LYDT]} &= \inf_{\|X\|_F=1} \|AXA^T - X\|_F &&= \sigma_{\min}(Z_{\mathrm{LYDT}}), \\ \mathrm{Sep[GSYL]} &= \inf_{\|X\|_F=1} \|AXB^T - CXD^T\|_F &&= \sigma_{\min}(Z_{\mathrm{GSYL}}), \\ \mathrm{Sep[GLYCT]} &= \inf_{\|X\|_F=1} \|AXE^T - EX(-A^T)\|_F &&= \sigma_{\min}(Z_{\mathrm{GLYCT}}), \\ \mathrm{Sep[GLYDT]} &= \inf_{\|X\|_F=1} \|AXA^T - EXE^T\|_F &&= \sigma_{\min}(Z_{\mathrm{GLYDT}}). \end{aligned}$$

One-norm $\mathrm{Sep}^{-1}$-estimators based on LAPACK techniques [Hager 1984; Higham 1988; Anderson et al. 1999] are implemented in the SLICOT [2001] library. Each estimator involves solving several (around five) triangular matrix equations. Therefore, it is important that these triangular matrix equation problems can be solved efficiently on today's memory tiered systems.

The underlying perturbation theory for standard and generalized matrix equations is presented in Higham [1993] and Kågström [1994], respectively.

## 3. RECURSIVE BLOCKED ALGORITHMS FOR TWO-SIDED MATRIX EQUATIONS

There is a quite extensive literature on the solution of matrix equations, and we refer to the selection of fundamental papers by Bartels and Stewart [1972], Golub et al. [1979], Hammarling [1982], Chu [1987], Kågström and Westin [1989], Gardiner et al. [1992a], Kågström and Poromaa [1996], and Penzl [1998] that all present reliable algorithms. Our recursive approach also applies to triangular two-sided matrix equations, which is the topic of this section.

Some of the matrix equations considered can be seen as special cases of other formulations. However, this is mainly of theoretical interest, since either these equivalences include matrix inversion (when transforming a generalized matrix equation to a standard counterpart), or the matrix equations have symmetry structure that we want to utilize in the algorithms.

As in Part I, we define recursive splittings for each matrix equation which in turn lead to a few similar subproblems to be solved. These splittings are recursively applied to all "half-sized" triangular matrix equations. We terminate the recursion when the new problem sizes ($M$ and/or $N$) are smaller than a certain block size, *blks*, which is chosen such that at least a few submatrices involved in the current matrix equation fit in the first-level cache memory. For the solution of the small-sized problems, we apply our new high-performance kernels (see Part I).

We present Matlab-style templates for two of the recursive blocked solvers. We remark that all updates with respect to the solution of subproblems in the recursion are GEMM-rich operations of the type

$$C \leftarrow \beta C + \alpha \mathrm{op}(A) X \,\mathrm{op}(B)^T,$$

where $\alpha$ and $\beta$ are real scalars, and op(Y) is $Y$ or $Y^T$. $A$ and/or $B$ can be dense or triangular. This is due to the "two-sidedness" of the matrix equations. We refer to Section 4.1 for a discussion of the design of a high-performance implementation of this operation. In the algorithm descriptions, we use the function $[C] = \mathbf{axb}(A, X, B)$, which implements the matrix-product operation $C = C + AXB$. Other functions, including level-3 BLAS operations are introduced the first time they are used in the algorithm descriptions. See Part I for BLAS references.

## 3.1 Recursive Triangular Generalized Sylvester Solvers

Consider the real *generalized Sylvester* (GSYL) matrix equation

$$AXB^T - CXD^T = E, \tag{1}$$

where $(A, C)$ of size $M \times M$ and $(B, D)$ of size $N \times N$ are in generalized Schur form; that is, $A$ and $B$ are upper quasitriangular and $C$, $B$ are upper triangular. The right-hand side $E$ and the solution $X$ are of size $M \times N$ and, typically, the solution overwrites the right-hand side ($E \leftarrow X$). The GSYL equation (1) has a *unique solution* if and only if $A - \lambda C$ and $D - \lambda B$ are regular and have disjoint spectra [Chu 1987], or equivalently Sep[GSYL] $\neq 0$.

We see that (1) is a generalization of the continuous-time Sylvester equation ($B = I_N$ and $C = I_M$), as well as the discrete-time Sylvester equation ($C = I_M$ and $D = I_N$). We consider three alternatives for doing a *recursive splitting*.

**Case 1** ($1 \leq N \leq M/2$). We split $(A, C)$ by rows and columns, and $E$ by rows only:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} B^T - \begin{bmatrix} C_{11} & C_{12} \\ & C_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} D^T = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix},$$

or equivalently

$$\begin{aligned} A_{11}X_1B^T - C_{11}X_1D^T &= E_1 - A_{12}X_2B^T + C_{12}X_2D^T, \\ A_{22}X_2B^T - C_{22}X_2D^T &= E_2. \end{aligned}$$

The original problem is split in two triangular generalized Sylvester equations. First, we solve for $X_2$ and after updating $E_1$ with respect to $X_2$, we can solve for $X_1$.

**Case 2** ($1 \leq M \leq N/2$). We split $(B, D)$ by rows and columns, and $E$ by columns only:

$$A\,[X_1\ X_2] \begin{bmatrix} B_{11}^T & \\ B_{12}^T & B_{22}^T \end{bmatrix} - C\,[X_1\ X_2] \begin{bmatrix} D_{11}^T & \\ D_{12}^T & D_{22}^T \end{bmatrix} = [E_1\ E_2],$$

or equivalently

$$\begin{aligned} AX_1B_{11}^T - CX_1D_{11}^T &= E_1 - AX_2B_{12}^T + CX_2D_{12}^T, \\ AX_2B_{22}^T - CX_2D_{22}^T &= E_2. \end{aligned}$$

As in Case 1, we first solve for $X_2$ and then after updating the right-hand side of the first equation, solve for $X_1$.

**Case 3** ($N/2 < M < 2N$). We split $(A, C)$, $(B, D)$, and $E$ by rows and columns:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_{11}^T & \\ B_{12}^T & B_{22}^T \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ & C_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} D_{11}^T & \\ D_{12}^T & D_{22}^T \end{bmatrix}$$
$$= \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}.$$

This recursive splitting results in the following four triangular generalized Sylvester equations:

$$\begin{aligned} A_{11}X_{11}B_{11}^T - C_{11}X_{11}D_{11}^T &= E_{11} - A_{12}X_{21}B_{11}^T - (A_{11}X_{12} + A_{12}X_{22})B_{12}^T \\ &\qquad + C_{12}X_{21}D_{11}^T + (C_{11}X_{12} + C_{12}X_{22})D_{12}^T, \\ A_{11}X_{12}B_{22}^T - C_{11}X_{12}D_{22}^T &= E_{12} - A_{12}X_{22}B_{22}^T + C_{12}X_{22}D_{22}^T, \\ A_{22}X_{21}B_{11}^T - C_{22}X_{21}D_{11}^T &= E_{21} - A_{22}X_{22}B_{12}^T + C_{22}X_{22}D_{12}^T, \\ A_{22}X_{22}B_{22}^T - C_{22}X_{22}D_{22}^T &= E_{22}. \end{aligned}$$

We start by solving for $X_{22}$ in the fourth equation. After updating $E_{12}$ and $E_{21}$ with respect to $X_{22}$, we can solve for $X_{12}$ and $X_{21}$. Both updates and the triangular generalized Sylvester solves are independent operations and can be executed concurrently. Finally, after updating $E_{11}$ with respect to $X_{12}$, $X_{21}$, and $X_{22}$, we solve for $X_{11}$. Some of the updates of $E_{11}$ can be combined in larger GEMM operations. We obtain

$$\begin{aligned} E_{11} = E_{11} \;-\; & A_{12}X_{21}B_{11}^T + C_{12}X_{21}D_{11}^T - \left([A_{11}\ A_{12}] \begin{bmatrix} X_{12} \\ X_{22} \end{bmatrix}\right) B_{12}^T \\ &+ \left([C_{11}C_{12}] \begin{bmatrix} X_{12} \\ X_{22} \end{bmatrix}\right) D_{12}^T. \end{aligned}$$

Alternatively, the update of $E_{11}$ can be performed as

$$\begin{aligned} E_{11} = E_{11} \;-\; & A_{11}X_{12}B_{12}^T + C_{11}X_{12}D_{12}^T - A_{12}([X_{21}\ X_{22}][B_{11}\ B_{12}])^T \\ &+ C_{12}\left([X_{21}\ X_{22}] \begin{bmatrix} D_{11} \\ D_{12} \end{bmatrix}\right)^T. \end{aligned}$$

```
Algorithm 1: rtrgsyl
Input:  $(A, C)$ $(M \times M)$ and $(B, D)$ $(N \times N)$ in upper generalized Schur form. $E$ $(M \times N)$ dense
        matrix. $blks$, block size that specifies when to switch to a standard algorithm for solving
        small-sized triangular generalized Sylvester equations.
Output: $X$ $(M \times N)$, the solution of $AXB^T - CXD^T = E$. $X$ is allowed to overwrite $E$.

function $[X] = \mathbf{rtrgsyl}(A, B, C, D, E, uplo, blks)$
if $1 \leq M, N \leq blks$ then
    $X = \mathbf{trgsyl}(A, B, C, D, E, uplo)$;
else switch $uplo$
case 1
    if $1 \leq N \leq M/2$   % Case 1: Split $(A, C)$ (by rows and columns), $E$ (by rows only)
        $X_2 = \mathbf{rtrgsyl}(A_{22}, B, C_{22}, D, E_2, uplo, blks)$;
        $E_1 = \mathbf{axb}(-A_{12}, X_2, B^T, E_1)$;  $E_1 = \mathbf{axb}(C_{12}, X_2, D^T, E_1)$;
        $X_1 = \mathbf{rtrgsyl}(A_{11}, B, C_{11}, D, E_1, uplo, blks)$;
        $X = [X_1; X_2]$;
    elseif $1 \leq M \leq N/2$   % Case 2: Split $(B, D)$ (by rows and columns), $E$ (by columns only)
        $X_2 = \mathbf{rtrgsyl}(A, B_{22}, C, D_{22}, E_2, uplo, blks)$;
        $E_1 = \mathbf{axb}(-A, X_2, B_{12}^T, E_1)$;  $E_1 = \mathbf{axb}(C, X_2, D_{12}^T, E_1)$;
        $X_1 = \mathbf{rtrgsyl}(A, B_{11}, C, D_{11}, E_1, uplo, blks)$;
        $X = [X_1, X_2]$;
    else  % $M, N > blks$   Case 3: Split $(A, C)$, $(B, D)$ and $E$ (all by rows and columns)
        $X_{22} = \mathbf{rtrgsyl}(A_{22}, B_{22}, C_{22}, D_{22}, E_{22}, uplo, blks)$;
        $E_{12} = \mathbf{axb}(-A_{12}, X_{22}, B_{22}^T, E_{12})$;  $E_{12} = \mathbf{axb}(C_{12}, X_{22}, D_{22}^T, E_{12})$;
        $E_{21} = \mathbf{axb}(-A_{22}, X_{22}, B_{12}^T, E_{21})$;  $E_{21} = \mathbf{axb}(C_{22}, X_{22}, D_{12}^T, E_{21})$;
        $X_{12} = \mathbf{rtrgsyl}(A_{11}, B_{22}, C_{11}, D_{22}, E_{12}, uplo, blks)$;
        $X_{21} = \mathbf{rtrgsyl}(A_{22}, B_{11}, C_{22}, D_{11}, E_{21}, uplo, blks)$;
        if $M < N$ then
            $E_{11} = \mathbf{axb}(-A_{11}, X_{12}, B_{12}^T, E_{11})$;  $E_{11} = \mathbf{axb}(C_{11}, X_{12}, D_{12}^T, E_{11})$;
            $E_{11} = \mathbf{axb}(-A_{12}, [X_{21}\,X_{22}], [B_{11}\,B_{12}]^T, E_{11})$;
            $E_{11} = \mathbf{axb}(C_{12}, [X_{21}\,X_{22}], [D_{11}\,D_{12}]^T, E_{11})$;
        else
            $E_{11} = \mathbf{axb}(-A_{12}, X_{21}, B_{11}^T, E_{11})$;  $E_{11} = \mathbf{axb}(C_{12}, X_{21}, D_{11}^T, E_{11})$;
            $E_{11} = \mathbf{axb}(-[A_{11}\,A_{12}], \left[\, X_{12}^T \;\; X_{22}^T \,\right]^T, B_{12}^T, E_{11})$;
            $E_{11} = \mathbf{axb}([C_{11}\,C_{12}], \left[\, X_{12}^T \;\; X_{22}^T \,\right]^T, D_{12}^T, E_{11})$;
        end
        $X_{11} = \mathbf{rtrgsyl}(A_{11}, B_{11}, C_{11}, D_{11}, E_{11}, uplo, blks)$;
        $X = [X_{11}, X_{12};\ X_{21}, X_{22}]$;
    end
case 2   % Code for $uplo = 2$.
case 3   % Code for $uplo = 3$.
case 4   % Code for $uplo = 4$.
end
```

Algorithm 1. Recursive blocked algorithm for solving the triangular generalized Sylvester equation.

In the discussion above, we have assumed that $A$, $B$, $C$, and $D$ are upper triangular (or quasitriangular). However, it is straightforward to derive recursive splittings where each of the matrices can be in either upper or lower Schur form.

In Algorithm 1, a Matlab-style function $[X] = \mathbf{rtrgsyl}(A, B, C, D, E, uplo, blks)$ for our recursive blocked solver is presented. The algorithm also shows an alternative way to combine the matrix multiply operations into large GEMM operations, and the choice of which one to use is dependent on the sizes of $M$ and $N$. The input *uplo* signals the triangular structure of the matrix pairs $(A, B)$ and $(C, D)$. The function $[X] = \textbf{trgsyl}(A, B, C, D, E)$ implements an algorithm for solving triangular generalized Sylvester kernel problems.

## 3.2 Recursive Triangular Discrete-Time Sylvester Solvers

By setting $C = D = I$ in real generalized Sylvester matrix equation (1), the real *discrete-time Sylvester* (SYDT) matrix equation is obtained:

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

The SYDT equation (2) has a *unique solution* if and only if $A$ and $B^{-1}$ (or $A^{-1}$ and $B$) have disjoint spectra; that is, $\lambda_i(A)\lambda_j(B) \neq 1$ for all $i$ and $j$, or equivalently Sep[SYDT] $\neq 0$.

The algorithm for solving the SYDT equation, named **rtrsydt**, is similar to Algorithm 1. For example, the recursive splitting of both dimensions, Case 3, results in the following four triangular discrete-time Sylvester equations.

$$\begin{aligned}
A_{11}X_{11}B_{11}^T - X_{11} &= C_{11} - A_{12}\left(X_{21}B_{11}^T + X_{22}B_{12}^T\right) - A_{11}X_{12}B_{12}^T,\\
A_{11}X_{12}B_{22}^T - X_{12} &= C_{12} - A_{12}X_{22}B_{22}^T,\\
A_{22}X_{21}B_{11}^T - X_{21} &= C_{21} - A_{22}X_{22}B_{12}^T,\\
A_{22}X_{22}B_{22}^T - X_{22} &= C_{22}.
\end{aligned}$$

We start by solving for $X_{22}$ in the fourth equation. After updating $C_{12}$ and $C_{21}$ with respect to $X_{22}$, we can solve for $X_{12}$ and $X_{21}$. Both updates and the triangular Sylvester solves are independent operations and can be executed concurrently. Finally, after updating $C_{11}$ with respect to $X_{12}$, $X_{21}$, and $X_{22}$, we solve for $X_{11}$. Also, two of the matrix multiply operations can be combined in one large GEMM operation, as for the GSYL equation. The other two cases are similar.

## 3.3 Recursive Triangular Generalized Discrete-Time Lyapunov Solvers

Consider the real *generalized discrete-time Lyapunov* (GLYDT) matrix equation

$$AXA^T - EXE^T = C, \tag{3}$$

where $A$ and $E$ of size $N \times N$ are upper quasitriangular and upper triangular, respectively. In other words, $(A, E)$ is in generalized Schur form. If $C$ is symmetric, then $X$ is symmetric as well. This is a special case of GSYL with $A = B$ and $C = D$ in (1), and we want to make use of the symmetry properties.

The GLYDT equation (3) has a *unique symmetric solution* if and only if $C = C^T$, and the eigenvalues $\lambda_i$ of $A - \lambda E$ satisfy $\lambda_i\lambda_j \neq 1$ for all $i$ and $j$ (with the convention that $0 \cdot \infty = 1$), or equivalently Sep[GLYDT] $\neq 0$. If $C$ is (semi)definite and $|\lambda_i(A - \lambda E)| < 1$ for all $i$, then a unique (semi)definite solution exists [Penzl 1998].

Since all matrices are square, we only have one way of doing a *recursive splitting*. We split $A$, $E$, and $C$ by rows and columns:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} A_{11}^T & \\ A_{12}^T & A_{22}^T \end{bmatrix} - \begin{bmatrix} E_{11} & E_{12} \\ & E_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} E_{11}^T & \\ E_{12}^T & E_{22}^T \end{bmatrix}$$
$$= \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

Since $X_{21} = X_{12}^T$, the recursive splitting results in three triangular generalized discrete-time Lyapunov equations:

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

We start by solving for $X_{22}$ in the third equation. After updating $C_{12}$ with respect to $X_{22}$, we can solve for $X_{12}$. Finally, after updating $C_{11}$ with respect to $X_{12}$ and $X_{22}$, we solve for $X_{11}$.

Four of the two-sided matrix product updates of $C_{11}$ can be expressed as two SYR2K operations:

$$\begin{aligned} C_{11} &= C_{11} - (A_{11}X_{12})A_{12}^T - A_{12}(A_{11}X_{12})^T, \quad \text{and} \\ C_{11} &= C_{11} + (E_{11}X_{12})E_{12}^T + E_{12}(E_{11}X_{12})^T, \end{aligned}$$

where $A_{11}X_{12}$ and $E_{11}X_{12}$ are TRMM operations. The GEMM-rich updates

$$C_{11} = C_{11} - A_{12}X_{22}A_{12}^T \quad \text{and} \quad C_{11} = C_{11} + E_{12}X_{22}E_{12}^T$$

are performed by the SLICOT MB01RD subroutine; see Section 4.1.

In Algorithm 2, a Matlab-style function $[X] =$ **rtrglydt**$(A, E, C, blks)$ implementing our recursive blocked solver is presented. The function $[X] =$ **trglydt**$(A, E, C)$ implements an algorithm for solving triangular generalized discrete-time Lyapunov kernel problems. For solving the triangular generalized Sylvester equations that appear, we make use of the recursive algorithm **rtrgsyl**, described in Section 3.1.

## 3.4 Recursive Triangular Discrete-Time Lyapunov Solvers

The real *discrete-time Lyapunov* (LYDT) or *Stein* matrix equation is obtained by setting $E = I_N$ in (3). We get the equation

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

where $A$ is upper triangular or upper quasitriangular, that is, in real Schur form. This equation is also the result of setting $A = B$ in (2). If $C$ is symmetric, then $X$ is symmetric as well, and the Stein matrix equation corresponds to the discrete-time standard algebraic Lyapunov equation. The LYDT equation (4) has a *unique solution* if and only if $\lambda_i\lambda_j \neq 1$ for all $i$ and $j$, or equivalently

**Algorithm 2: rtrglydt**

*Input:* $(A, E)$ $(N \times N)$ in upper generalized Schur form. $C$ $(N \times N)$ dense matrix. *blks*, block size that specifies when to switch to a standard algorithm for solving small-sized triangular generalized Lyapunov equations.

*Output:* $X$ $(N \times N)$, the solution of $AXA^T - EXE^T = C$. $X$ is allowed to overwrite $C$, and is symmetric if $C = C^T$ on entry.

```
function $[X] = \textbf{rtrglydt}(A, E, C, blks)$
if $1 \le N \le blks$ then
    $X = \textbf{trglydt}(A, E, C)$;
elseif $C$ is symmetric
     % Split $(A, E)$, and $C$ (all by rows and columns)
    $X_{22} = \textbf{rtrglydt}(A_{22}, E_{22}, C_{22}, blks)$;
    $C_{12} = \textbf{axb}(-A_{12}, X_{22}, A_{22}^T, C_{12})$; $C_{12} = \textbf{axb}(E_{12}, X_{22}, E_{22}^T, C_{12})$;
    $X_{12} = \textbf{rtrgsyl}(A_{11}, A_{22}, E_{11}, E_{22}, C_{12}, blks)$; $X_{21} = X_{12}^T$;
    $C_{11} = \textbf{syr2k}(\textbf{trmm}(A_{11}, X_{12}), -A_{12}, C_{11})$;
    $C_{11} = \textbf{syr2k}(\textbf{trmm}(E_{11}, X_{12}), E_{12}, C_{11})$;
    $C_{11} = \textbf{axb}(-A_{12}, X_{22}, A_{12}^T, C_{11})$; $C_{11} = \textbf{axb}(E_{12}, X_{22}, E_{12}^T, C_{11})$;
    $X_{11} = \textbf{rtrglydt}(A_{11}, E_{11}, C_{11}, blks)$;
    $X = [X_{11}, X_{12};\ X_{21}, X_{22}]$;
else
    $X = \textbf{rtrgsyl}(A, A, E, E, C, blks)$;
end
```

Algorithm 2. Recursive blocked algorithm for solving the triangular generalized discrete-time Lyapunov equation.

Sep[LYDT] $\neq 0$. If $C$ is (semi)definite and $|\lambda_i| < 1$ for all $i$, then a unique (semi)definite solution exists [Hammarling 1982].

The template for the recursive splitting follows the one for the GLYDT equation. If $C = C^T$, then $X_{21} = X_{12}^T$ and the recursive splitting leads to three triangular matrix equations:

$$\begin{aligned} A_{11}X_{11}A_{11}^T - X_{11} &= C_{11} - A_{12}X_{12}^TA_{11}^T - A_{11}X_{12}A_{12}^T - A_{12}X_{22}A_{12}^T, \\ A_{11}X_{12}A_{22}^T - X_{12} &= C_{12} - A_{12}X_{22}A_{22}^T, \\ A_{22}X_{22}A_{22}^T - X_{22} &= C_{22}. \end{aligned}$$

The first and the third are standard Stein-type equations, while the second is a triangular discrete-time Sylvester equation. The recursive blocked algorithm is named **rtrlydt**.

## 3.5 Recursive Triangular Generalized Continuous-Time Lyapunov Solvers

Consider the real *generalized continuous-time Lyapunov* (GLYCT) matrix equation

$$AXE^T + EXA^T = C, \tag{5}$$

where $A$ and $E$ of size $N \times N$ are upper quasitriangular and upper triangular, respectively. If $C$ is symmetric, then $X$ is symmetric as well. We treat this case as a special case of the generalized Sylvester equation with $A = C$, $B = D$, and changing the sign in the GSYL equation (1). The right-hand side $C$ and the solution $X$ are of size $N \times N$, and typically, the solution overwrites the right-hand

Table I. Complexity of Standard Algorithms Measured in Flops

| Matrix Equation | Overall Cost in Flops |
|---|---|
| SYDT (2) | $2M^2N + MN^2$ $(M \le N)$ |
| | $M^2N + 2MN^2$ $(M > N)$ |
| LYDT (4) | $\frac{4}{3}N^3$ |
| GSYL (1) | $4M^2N + 2MN^2$ $(M \le N)$ |
| | $2M^2N + 4MN^2$ $(M > N)$ |
| GLYCT (5) | $\frac{8}{3}N^3$ |
| GLYDT (3) | $\frac{8}{3}N^3$ |

side ($C \leftarrow X$). By choosing $E = I_N$ in (5), we get the standard continuous-time Lyapunov equation, which is covered in Jonsson and Kågström [2001].

The GLYCT equation (5) has a *unique symmetric solution* if and only if $C = C^T$, and all eigenvalues $\lambda_i$ of $A - \lambda E$ are finite with $\lambda_i + \lambda_j \neq 0$ for all $i$ and $j$, or equivalently Sep[GLYCT] $\neq 0$. If $C$ is (semi)definite and $\text{Re}\lambda_i < 0$ for all $i$, then a unique (semi)definite solution exists [Penzl 1998].

Since $X$ is symmetric, the recursive splitting results in two triangular GLYCT equations and one GSYL equation:

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

We start by solving for $X_{22}$ in the third equation. After updating $C_{12}$ with respect to $X_{22}$, we can solve for $X_{12}$. Finally, after updating $C_{11}$ with respect to $X_{12}$ and $X_{22}$, we solve for $X_{11}$. We remark that the update of $C_{11}$ includes two SYR2K operations, namely,

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

where $A_{11}X_{12}$ and $E_{11}X_{12}$ are TRMM operations and $A_{12}X_{22}$ is a GEMM operation. The recursive blocked algorithm is called **rtrglyct**.

## 3.6 Number of Operations and Execution Order

In Table I, we summarize the complexity measured in *floating point operations* (flops) of the standard elementwise explicit algorithms for solving two-sided triangular matrix equations (e.g., see Bartels and Stewart [1972], Chu [1987], Gardiner et al. [1992a], and Penzl [1998]). They are all based on backward or forward substitutions with one or several right-hand sides. We remark that the difference in flops between the two extreme cases (i.e., when all diagonal blocks of the matrices in upper Schur form are of size $1 \times 1$ or $2 \times 2$, respectively) only show up in the lower-order terms.

Table II. Complexity of Recursive Blocked Algorithms Measured in Flops

| Matrix Equation | Overall Cost in Flops | Flop Ratio ($M = N$) |
|---|---|---|
| SYDT (2) | $\frac{6}{4}M^2N + 2MN^2 \ (M \leq N)$ | 1.1667 |
| | $2M^2N + \frac{6}{4}MN^2 \ (M > N)$ | |
| LYDT (4) | $\frac{25}{12}N^3$ | 1.5625 |
| GSYL (1) | $3M^2N + 4MN^2 \ (M \leq N)$ | 1.1667 |
| | $4M^2N + 3MN^2 \ (M > N)$ | |
| GLYCT (5) | $\frac{21}{6}N^3$ | 1.3125 |
| GLYDT (3) | $\frac{25}{6}N^3$ | 1.5625 |

Assuming $2N > M > N/2$, the flopcounts of the recursive blocked algorithms can be expressed in terms of the following recurrence formulas.

$$F_{\text{SYDT}}(M,N) = 4F_{\text{SYDT}}(M/2,N/2) + M^2N + \frac{1}{2}MN^2, \tag{6}$$

$$F_{\text{LYDT}}(N) = 2F_{\text{LYDT}}(N/2) + F_{\text{SYDT}}(N/2,N/2) + \frac{7}{8}N^3, \tag{7}$$

$$F_{\text{GSYL}}(M,N) = 4F_{\text{GSYL}}(M/2,N/2) + 2M^2N + \frac{3}{2}MN^2, \tag{8}$$

$$F_{\text{GLYCT}}(N) = 2F_{\text{GLYCT}}(N/2) + F_{\text{GSYL}}(N/2,N/2) + \frac{5}{4}N^3, \tag{9}$$

$$F_{\text{GLYDT}}(N) = 2F_{\text{GLYDT}}(N/2) + F_{\text{GSYL}}(N/2,N/2) + \frac{7}{4}N^3. \tag{10}$$

These expressions correspond to the most general case when the recursive splitting is by rows and by columns for all input matrices. We have collected the cost for all two-sided matrix multiplications (including the SYR2K operations) of each recursive algorithm in the last term(s) of the recurrences. The flopcounts take any triangular structure of the matrices in each two-sided matrix product into account (see Section 4.1). Ignoring the lower-order terms, it is straightforward to derive the expressions in Table II from Equations (6) through (10), respectively.

Our recursive blocked algorithms require both extra workspace and more flops compared to the standard elementwise algorithms. The extra workspace is needed in the evaluation of the two-sided matrix multiplications (see Section 4.1). In the third column of Table II, the ratios between the flopcounts for the recursive blocked algorithms and the standard algorithms are listed. Despite the quite large flops penalties of the recursive blocked algorithms, they outperform the standard algorithms for large enough problems (see measured performance results in Section 5). This fact is mainly due to the difference in their data reference patterns, that is, the order in which they access data and how many times the data are moved in the memory hierarchy of the target computer system. The cost of redundant memory transfers can be devastating to the algorithm performance.

## 4. DESIGN AND IMPLEMENTATION ISSUES

In Part I [Jonsson and Kågström 2002], we describe the three levels of triangular matrix equation solvers used in our implementations. Here, we recapitulate

```
Algorithm 3: axb
Input:  $A\ (M \times P)$, $X\ (P \times Q)$, $B\ (Q \times N)$ and $C\ (M \times N)$.
Output: $C\ (M \times N)$, the result of the matrix product $C = C + AXB$.

function $[C]$ = axb($A, X, B, C$)
if $A, B$ dense then
    $C$ = gemm($A$, gemm($X, B, \emptyset$), $C$);
elseif $A$ triangular, $B$ dense
    $C$ = gemm(trmm($A, X$), $B, C$);
elseif $A$ dense, $B$ triangular
    $C$ = gemm($A$, trmm($X, B$), $C$);
elseif $A$ triangular, $B$ triangular
    $C$ = gemm($\emptyset, \emptyset$, trmm($A$, trmm($X, B$)));
end
```

Algorithm 3. Algorithm for computing a GEMM-rich matrix product.

the properties of these solvers. At the user level there are the recursive blocked **rtr*** *solvers*. Each of them calls a **tr*** *block* or *subsystem solver* when the current problem sizes ($M$ and/or $N$) from a recursive splitting are smaller than a certain block size, *blks*. Finally, each of the block solvers calls a *superscalar kernel* for solving matrix equations with $M, N \leq 4$.

We also refer the reader to Part I for discussions on the impact and design of the subsystem solvers and superscalar kernels, block sizes, BLAS implementations, and parallelization issues. In this section, the impact of the two-sidedness of the matrix equations on the different routines is discussed.

### 4.1 Design of Optimized Two-Sided Matrix Product Kernels

The update of the right-hand side of the equations is done by the **axb** operation, that is, an optimized kernel which performs the two-sided matrix product operation

$$C \leftarrow \beta C + \alpha \mathrm{op}(A) X \mathrm{op}(B)^T,$$

where $\alpha$ and $\beta$ are real scalars, and op($Y$) is $Y$ or $Y^T$. $A$ and/or $B$ can be dense or triangular. Besides, one or several of the three matrices can be symmetric. Unfortunately, these operations need extra workspace, which can be substantial (typically the size of the current right-hand side).

For an outline version that implements the generic operation $C = C + AXB$ see Algorithm 3. Depending on the sizes of $A$, $X$, and $B$, we should perform the matrix product as $(\mathrm{op}(A)X)\mathrm{op}(B)$ or as $\mathrm{op}(A)(X\,\mathrm{op}(B))$. Here, it is important that the BLAS implementation provide proper blocking for the cases which lead to bad memory access patterns. If matrices are stored in column-major order, the order $(A^T X)B$ might give a significantly better access pattern than $A^T(XB)$. Also, $A$ and/or $B$ might be quasitrapezoidal and/or quasitriangular. Examples of both cases are found in **rtrgsyl**, Algorithm 1. We also want to make use of symmetry properties; for example, see **rtrglydt**, Algorithm 2, where $C \leftarrow C - AXA^T$, $X = X^T$, is computed.

In our implementation, the costs of computing $(\mathrm{op}(A)X)\mathrm{op}(B)$ ($flops_{1A} + flops_{1B}$) and $\mathrm{op}(A)(X\,\mathrm{op}(B))$ ($flops_{2A} + flops_{2B}$) are compared. The operation that requires the least amount of flops is then chosen. Small trapezoidal matrices are treated as dense matrices, as the performance gain from handling the triangular and the rectangular parts separately is penalized by the extra function call and matrix setup. Subroutine setup costs are minimized by the use of our superscalar GEMM routine for problems that fit into the level-1 cache, that is, problems solved by the **tr*** subsystem solvers. In the algorithm, this is controlled by the parameter *blks*, the same parameter used in the recursive algorithms to switch from the recursive blocked algorithms to the subsystem solvers. In the implementation, fix-up code for handling any $2\times 2$ diagonal blocks of the quasitriangular matrices is implemented using level-1 BLAS routines.

For the symmetric case, $\tilde{C} \leftarrow C - AXA^T$, $X = X^T$, and $C = C^T$, the SLICOT MB01RD subroutine is used [SLICOT 2001]. It reduces the complexity from $2M^3$ to $(3/2)M^3$, by calculating the upper triangular part of the symmetric matrix $\mathrm{triu}(\tilde{C}) = V + \mathrm{triu}(B) + \mathrm{stril}(B)^T + \mathrm{diag}(V) + \mathrm{diag}(\mathrm{triu}(B))$, where $B = ATA^T$, $T = \mathrm{triu}(X) - \frac{1}{2}\mathrm{diag}(X)$, and $V = \mathrm{triu}(C) - \frac{1}{2}\mathrm{diag}(C)$, and triu, stril, and diag denote the upper triangular, the strictly lower triangular, and the diagonal parts, respectively. The MB01RD routine is implemented using level-3 BLAS routines and requires no workspace.

However, for some problems, the complexity can be lowered even more, at the cost of extra workspace. In **rtrglydt**, Algorithm 2, the temporary result $W = A_{12}X_{22}$ in the operation $C_{12} = \mathbf{axb}(-A_{12}X_{22}A_{22}^T)$ can be reused in the operation $C_{11} = \mathbf{axb}(-A_{12}X_{22}A_{22}^T)$, and analogously for the $E$ matrix. Although this triples the workspace needed, it lowers the overall complexity of the algorithm from $(25/6)N^3$ to $(23/6)N^3$.

## 5. PERFORMANCE RESULTS OF RECURSIVE BLOCKED ALGORITHMS

The recursive blocked algorithms for solving two-sided triangular matrix equations have been implemented in Fortran 90, using the facilities for recursive calls of subprograms, dynamic memory allocation, and threads. In this section, sample performance results of these implementations executing on IBM Power, MIPS, and Intel Pentium processor-based systems are presented. We have selected a few processors representing different vendors and different architecture characteristics and memory hierarchies. The details of the processors, memory subsystems, and operating systems are all described in Part I [Jonsson and Kågström 2002]. The only difference from Part I is the Power3 machine, which is running at 375 MHz, giving each processor a theoretical peak rate of 1500 Mflops/s. The performance results (measured in Mflops/s) are computed using the flopcounts presented in Table I of Section 3.6. The results are displayed in graphs and tables, where the performances of different variants of the recursive blocked implementations are visualized together with existing routines in the state-of-the-art SLICOT library [SLICOT 2001]. Moreover, the relative performances (measured as speedup) between different algorithms including sequential as well as parallel implementations are presented in tables. Most of the results presented should be quite self-explanatory. Therefore, our

Table III. Performance Results for Triangular Generalized Sylvester Equation: IBM Power3 (Left) and Intel Pentium III (Right)[a]

| | IBM Power3 | | | Intel Pentium III | | |
|---|---|---|---|---|---|---|
| | Time (sec) | Speedup | | Time (sec) | Speedup | |
| $M = N$ | A | B/A | C/A | A | B/A | C/A |
| 100 | 1.30e − 02 | 0.96 | 1.16 | 4.30e − 02 | 0.98 | 1.26 |
| 250 | 1.31e − 01 | 0.48 | 1.26 | 4.49e − 01 | 0.96 | 1.34 |
| 500 | 8.91e − 01 | 0.67 | 1.50 | 3.07e + 00 | 0.96 | 1.38 |
| 1000 | 6.42e + 00 | 1.05 | 1.91 | 2.18e + 01 | 1.05 | 1.51 |
| 1500 | 2.13e + 01 | 1.34 | 2.09 | 7.10e + 01 | 1.09 | 1.54 |
| 2000 | 4.77e + 01 | 1.48 | 2.16 | 1.60e + 02 | 1.14 | 1.59 |

A – **rtrgsyl**
B – A + linking with SMP BLAS
C – B + utilizing explicit // in the recursion tree

[a]Labels A–C represent different algorithms and implementations.

discussion is restricted to significant or unexpected differences between the implementations executing on different computing platforms. For additional performance results see Jonsson and Kågström [2001]. The accuracy of the results computed by our recursive blocked algorithms are overall very good and similar to the accuracy obtained by the corresponding SLICOT routines. For benchmark problems see Kressner et al. [1999a,b] and SLICOT [2001].

## 5.1 Triangular Generalized Sylvester Equation

In Table III, performance results for different algorithms and implementations, executing on IBM Power3 and Intel Pentium III processor-based systems, for solving the triangular generalized Sylvester equation are displayed. We have not found any library or public software for solving triangular generalized Sylvester equations, so the results presented here are for our recursive blocked algorithms only.

## 5.2 Triangular Discrete-Time Sylvester Equation

In Figure 1, we show performance graphs for different algorithms and implementations, executing on IBM Power3 and Intel Pentium III processor-based systems, for solving triangular discrete-time Sylvester equations.

The SLICOT SB04PY implements an explicit Bartels–Stewart solver and is mainly a level-2 routine, which explains its poor performance behavior. Our recursive blocked **rtrsydt** shows between a 2-fold and a 118-fold speedup with respect to SLICOT SB04PY and an additional speedup up to 2.14 on a four-processor Power3 node for large enough problems.

## 5.3 Triangular Discrete-Time Lyapunov Equation

In Figure 2, performance results for the triangular discrete-time Lyapunov equation are presented, now using IBM PowerPC 604e and SGI MIPS R10000 processor-based systems. We compare our recursive blocked **rtrlydt** algorithm with the SLICOT SB03MX routine, which mainly is a level-2 implementation

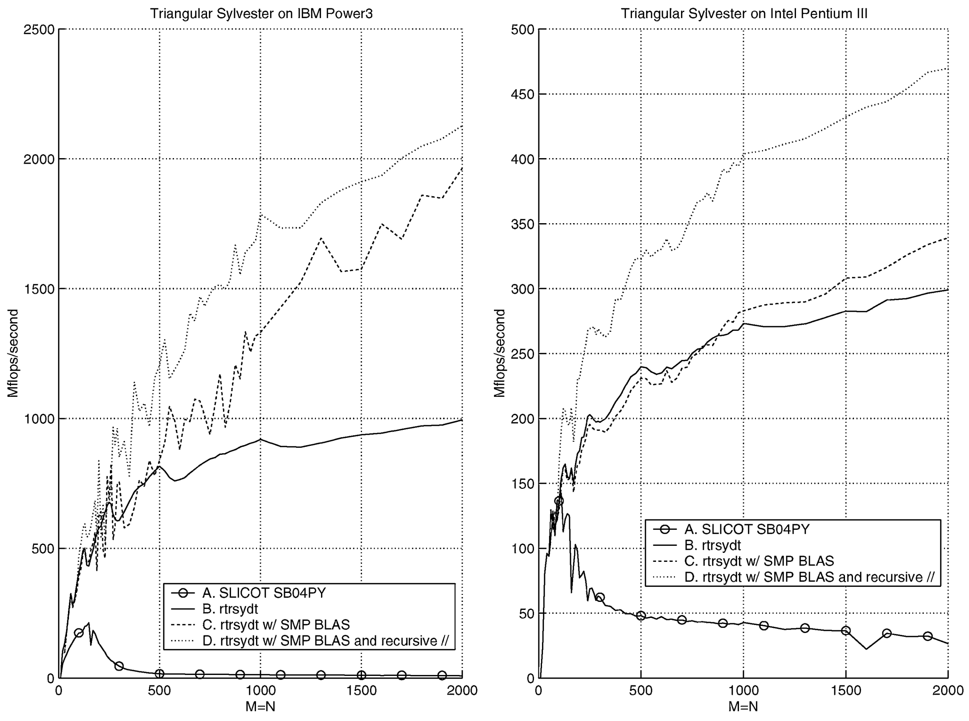
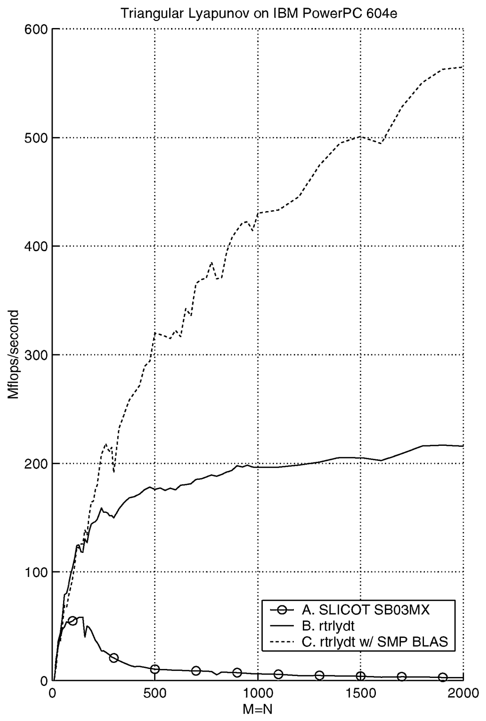


Fig. 1. Performance results for the triangular discrete-time Sylvester equation ($M = N$): IBM Power3 (left) and Intel Pentium III (right).

as well. As expected, the relative behavior between the different algorithms and implementations follows qualitatively that of the discrete-time Sylvester equation. We remark that the speedup of **rtrlydt** with respect to the SLICOT SB03MX is between 1.9 and 76 and an additional speedup of 2.6 on a four-processor IBM PowerPC 604e node for large enough problems. The results for the MIPS R10000 show between a 1.5-fold to over 28-fold speedup. Due to lack of support for nested multithreading with OpenMP and incompatibilities between pthreads and efficient SMP BLAS, we do not show any parallel performance results for the MIPS processor.

## 5.4 Relative Performance of the Recursive Blocked Solvers

We have also investigated the relative performance between the recursive blocked algorithms of the two-sided triangular matrix equations. In Table IV (upper part), the relative performance of the solvers with respect to **rtrsydt** is displayed. All timings are from using one IBM Power3 processor. In the lower part, the ratios of the dominating terms of the flops counts in Table II are shown, and we see that the time and flops ratios agree nicely. We remark that the flops ratio of **rtrglydt** corresponds to the implemented algorithm, which has a lower dominating factor than recorded in Table II (see Section 4.1).



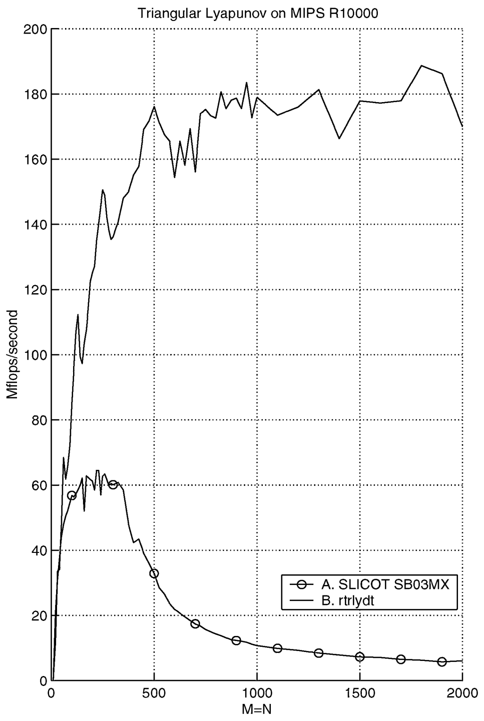


Fig. 2. Performance results for the triangular discrete-time Lyapunov equation: IBM PowerPC 604e (left) and SGI Onyx2 MIPS R10000 (right).

Table IV. Relative Performance Results for Recursive Blocked Solvers

| | Time | Time Relative to **rtrsydt** | | | |
|---|---|---|---|---|---|
| $N$ | **rtrsydt** | **rtrlydt** | **rtrgsyl** | **rtrglydt** | **rtrglyct** |
| 100 | 0.0075 | 0.68 | 1.77 | 1.22 | 0.99 |
| 250 | 0.0700 | 0.60 | 1.87 | 1.11 | 0.97 |
| 500 | 0.4607 | 0.60 | 1.93 | 1.09 | 0.99 |
| 1000 | 3.2667 | 0.60 | 1.96 | 1.10 | 1.00 |
| 1500 | 10.8120 | 0.60 | 1.97 | 1.09 | 1.00 |
| 2000 | 24.1414 | 0.61 | 1.98 | 1.10 | 1.01 |

| | Flops | Flops Relative to **rtrsydt** | | | |
|---|---|---|---|---|---|
| $N$ | **rtrsydt** | **rtrlydt** | **rtrgsyl** | **rtrglydt** | **rtrglyct** |
| | $3.5N^3$ | 0.60 | 2.0 | 1.10 (1.19) | 1.0 |

## 5.5 Impact on Solving Unreduced Two-Sided Generalized Matrix Equations

As for the one-sided matrix equations, we have investigated the impact of the choice of triangular matrix equation solver on the time to solve unreduced two-sided matrix equations. Although the transformation of an unreduced matrix equation to a triangular counterpart and the backtransformation of the solution are normally operations at least as costly (measured in flops) as the triangular solve, the impact of using our recursive triangular solvers can be substantial.

Table V. (a) Total Execution Times for Solving an Unreduced Generalized Discrete-Time Lyapunov Equation (Upper Part) and Computing an Estimate of Sep[GLYDT] (Lower Part) for Two Different Triangular Generalized Lyapunov Solvers. Here, the Reduction to Generalized Schur Form of $(A, E)$ is Performed Using the LAPACK Routines DGGHRD and DHGEQZ. (b) Corresponding Results Using Blocked Algorithms Described in Dackland and Kågström [1999] for the Initial Condensing Operation on $(A, E)$

| (a) | SG03AD Using SG03AX | | SG03AD Using **rtrglydt** | | | |
|---|---|---|---|---|---|---|
| $N$ | Total Time | Solver Part (%) | Total Time | Solver Part (%) | Speedup | Job |
| 50 | 0.0277 | 49.9 | 0.0185 | 20.1 | 1.50 | $X$ |
| 100 | 0.180 | 51.2 | 0.0967 | 9.0 | 1.86 | $X$ |
| 250 | 2.89 | 46.8 | 1.62 | 4.7 | 1.79 | $X$ |
| 500 | 59.0 | 42.3 | 34.5 | 1.5 | 1.71 | $X$ |
| 750 | 303.4 | 42.0 | 177.5 | 0.9 | 1.71 | $X$ |
| 1000 | 646.6 | 44.6 | 361.8 | 1.0 | 1.79 | $X$ |
| 50 | 0.117 | 87.6 | 0.0263 | 45.6 | 4.44 | $X$+Sep |
| 100 | 0.709 | 87.3 | 0.152 | 40.6 | 4.68 | $X$+Sep |
| 250 | 9.98 | 84.5 | 2.08 | 25.4 | 4.81 | $X$+Sep |
| 500 | 178.6 | 80.9 | 37.8 | 9.4 | 4.73 | $X$+Sep |
| 750 | 924.1 | 80.9 | 184.4 | 4.5 | 5.01 | $X$+Sep |
| 1000 | 2076.6 | 82.7 | 391.8 | 8.4 | 5.30 | $X$+Sep |

| (b) | SG03AD Using SG03AX | | SG03AD Using **rtrglydt** | | | |
|---|---|---|---|---|---|---|
| $N$ | Total Time | Solver Part (%) | Total Time | Solver Part (%) | Speedup | Job |
| 50 | 0.0306 | 44.7 | 0.019 | 11.2 | 1.61 | $X$ |
| 100 | 0.213 | 43.2 | 0.129 | 6.5 | 1.65 | $X$ |
| 250 | 3.18 | 42.5 | 1.90 | 3.9 | 1.67 | $X$ |
| 500 | 41.8 | 59.7 | 17.3 | 2.9 | 2.41 | $X$ |
| 750 | 187.1 | 68.2 | 61.1 | 2.7 | 3.06 | $X$ |
| 1000 | 428.9 | 67.2 | 144.1 | 2.5 | 2.98 | $X$ |
| 50 | 0.120 | 85.4 | 0.0285 | 39.1 | 4.19 | $X$+Sep |
| 100 | 0.741 | 83.5 | 0.166 | 26.2 | 4.47 | $X$+Sep |
| 250 | 10.3 | 82.1 | 2.69 | 31.0 | 3.83 | $X$+Sep |
| 500 | 161.3 | 89.5 | 20.0 | 15.3 | 8.06 | $X$+Sep |
| 750 | 807.7 | 92.6 | 67.9 | 12.1 | 11.89 | $X$+Sep |
| 1000 | 1857.4 | 92.4 | 174.0 | 18.9 | 10.68 | $X$+Sep |

For illustration, we first use the SLICOT routine SG03AD which solves unreduced generalized discrete-time Lyapunov equations. SG03AD also has an option to compute an estimate of the separation Sep[GLYDT] (see Section 2).

In Table V(a), timings for the SG03AD routine are displayed for problem sizes ranging from 50 to 1000 using two different triangular matrix equation solvers. These solvers are SG03AX provided in SLICOT, which implements a variant of the Bartels–Stewart method calling BLAS [Penzl 1998], and our recursive blocked **rtrglydt** algorithm. In the second column, the total times for solving an unreduced system with SG03AX as the triangular solver are displayed. This includes the time for the generalized Schur factorization and backtransformation of the solution. In the fourth and fifth columns, similar results are displayed when SG03AX is replaced by the **rtrglydt** routine. We see up to 90% speedup for the problem sizes considered. The numbers in the lower part of Table V(a) also include the time for computing a 1-norm-based estimate for Sep[GLYDT]. The condition estimation process includes repeated calls (typically five) to the generalized triangular solver, and as expected we see a four- to fivefold speedup when using our recursive solver.

In Table V(b), the corresponding results are displayed when we have replaced the LAPACK routines DGGHRD and DHGEQZ by Dackland–Kågström's blocked Hessenberg-triangular reduction and $QZ$ algorithms [Dackland and Kågström 1999] for transforming the regular pair $(A, E)$ to generalized Schur form. For large enough problems, this gives another factor of two speedup.

We have also compared the routine SYLG [Gardiner et al. 1992a,b] which implements a variant of the Hessenberg–Schur method [Golub et al. 1979] for solving unreduced generalized Sylvester equations (1). Also for this matrix equation, we see a substantial impact in using our recursive blocked algorithm (almost a factor 3 speedup for $N = 1000$). For detailed results we refer to Jonsson and Kågström [2001]. We remark that we only see small benefits (or no benefits at all) in using our recursive algorithm for small problem sizes or when $M = 10N$. This result is not due to the properties of the recursive solvers, but rather to the properties of the Hessenberg–Schur method, where only one of the matrices of $A$ and $D$ is reduced to Schur form, and the other is reduced to Hessenberg form. The Hessenberg–Schur algorithm is best suited for cases when $M \gg N$ or $M \ll N$, when only the smaller of the matrix pairs is reduced to upper quasitriangular Schur form.

## 6. SUMMARY AND SOME CONCLUSIONS

The performance results verify that our recursive approach is also very efficient for solving two-sided triangular matrix equations on today's hierarchical memory computer systems. Despite the quite large flops penalties of the recursive blocked algorithms they outperform the standard algorithms for large enough problems. For example, solving discrete-time Sylvester and Lyapunov equations with coefficient matrices of size $2000 \times 2000$ takes approximately one hour using current routines in the SLICOT [2001] library, and the solution times of our recursive blocked algorithms are less than one minute for the same problems. This is partly due to the difference in the data reference patterns of the algorithms. Our recursive blocked algorithms automatically match the memory hierarchy of a target machine and provide good data locality. As described in the Part I paper on one-sided matrix equations, we develop new high-performance superscalar kernels for solving the remaining small-sized triangular matrix equations and lightweight GEMM operations, which implies that a larger part of the total execution time is spent in high-performance GEMM operations. Also for the two-sided matrix equations we terminate the recursion with $blks = 4$ without degrading performance, which in turn means that the implementations are architecture-independent. Moreover, we have developed optimized two-sided matrix product kernels that take any symmetry properties as well as any triangular or trapezoidal structure of the matrices into account. In order to maximize the performance of these two-sided matrix product operations (e.g., $AXB^T$), we make use of optimized level-3 BLAS, which by default require some temporary storage. Altogether, this leads to simpler and much faster algorithms for solving reduced as well as unreduced two-sided and generalized Sylvester and Lyapunov matrix equations and different associated condition estimation problems. Our intention is to make these algorithms available in the SLICOT [2001] library. For additional references see Part I [Jonsson and Kågström 2002].

The concept of recursion is very old. Our invention is new recursive blocked algorithms for efficient solution of triangular one-sided (Part I) and two-sided (Part II) matrix equations including new efficient matrix equation kernel solvers. Our work extends earlier work on level-3 BLAS and matrix factorizations.

Furthermore, we want to emphasize that developing a novel set of algorithms for an interesting class of problems is a merit itself. However, in our opinion the real merit comes first when it is proved in practice that the software implementations give a substantial performance boost without sacrificing accuracy for not too ill-conditioned problems. Indeed, both these merits are attained by our work presented in the Part I and II articles. Together they cover all common linear matrix equations. Moreover, since the triangular matrix equations considered also appear as frequent subproblems in solving Riccati-type matrix equations, we foresee a great impact of our work in control theory applications.

## ACKNOWLEDGMENTS

We thank Fred Gustavson and the colleagues in the Umeå HPC and Parallel Computing Research Group for stimulating and fruitful discussions. Finally, we thank Sven Hammarling and the referees for constructive comments on an earlier version of this manuscript.

## REFERENCES

Anderson, E., Bai, Z., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. 1999. *LAPACK Users' Guide*, third ed. SIAM, Philadelphia.

Bartels, R. H. and Stewart, G. W. 1972. Algorithm 432: Solution of the equation $AX + XB = C$, *Commun. ACM 15*, 9, 820–826.

Chu, K.-W. E. 1987. The solution of the matrix equation $AXB - CXD = Y$ and $(YA - DZ, YC - BZ) = (E, F)$. *Linear Algebra Appl. 93*, 93–105.

Dackland, K. and Kågström, B. 1999. Blocked algorithms and software for reduction of a regular matrix pair to generalized Schur form. *ACM Trans. Math. Softw. 25*, 4, 425–454.

Gardiner, J. D., Laub, A. J., Amato, J. J., and Moler, C. B. 1992a. Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Softw. 18*, 223–231.

Gardiner, J. D., Wette, M. R., Laub, A. J., Amato, J. J., and Moler, C. B. 1992b. A Fortran 77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Softw. 18*, 232–238.

Golub, G., Nash, S., and Van Loan, C. 1979. A Hessenberg–Schur method for the matrix problem $AX + XB = C$. *IEEE Trans. Autom. Contr. AC-24*, 6, 909–913.

Hager, W. W. 1984. Condition estimates. *SIAM J. Sci. Stat. Comp. 5*, 311–316.

Hammarling, S. J. 1982. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal. 2*, 303–323.

Higham, N. J. 1988. Fortran codes for estimating the one-norm of a real or complex matrix with applications to condition estimation. *ACM Trans. Math. Softw. 14*, 381–396.

Higham, N. J. 1993. Perturbation theory and backward error for $AX - XB = C$. *BIT 33*, 124–136.

Jonsson, I. and Kågström, B. 2001. Recursive blocked algorithms for solving triangular matrix equations—Part II: Two-sided and generalized Sylvester and Lyapunov equations. *SLICOT*

*Working Note* 2001-5. Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden.

JONSSON, I. AND KÅGSTRÖM, B. 2002. Recursive blocked algorithms for solving triangular systems—Part I: One-sided and coupled Sylvester-type matrix equations. *ACM Trans. Math. Softw*. *28*, 4, (Dec.).

KÅGSTRÖM, B. 1994. A perturbation analysis of the generalized Sylvester equation $(AR - LB, DR - LE) = (C, F)$. *SIAM J. Matrix Anal. Appl. 15*, 4, 1045–1060.

KÅGSTRÖM, B. AND POROMAA, P. 1996. LAPACK–style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs. *ACM Trans. Math. Softw. 22*, 1, 78–103.

KÅGSTRÖM, B. AND WESTIN, L. 1989. Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. *IEEE Trans. Autom. Contr. 34*, 4, 745–751.

KRESSNER, D., MEHRMANN, V., AND PENZL, T. 1999a. CTLEX—A collection of benchmark examples for continuous-time Lyapunov equations. *SLICOT Working Note* 1999–6.

KRESSNER, D., MEHRMANN, V., AND PENZL, T. 1999b. DTLEX—A collection of benchmark examples for discrete-time Lyapunov equations. *SLICOT Working Note* 1999–7.

PENZL, T. 1998. Numerical solution of generalized Lyapunov equations, *Adv. Comp. Math. 8*, 33–48.

SLICOT 2001. Library and the numerics in control network (NICONET) Web site: www.win.tue.nl/niconet/index.html.

Received October 2001; revised June 2002; accepted July 2002