Block variants of Hammarling's method for solving Lyapunov equations

Daniel Kressner*

January 18, 2007

Abstract

This paper is concerned with the efficient numerical solution of the Lyapunov equation $A^T X + XA = -C$ with a stable matrix A and a symmetric positive semidefinite matrix C of possibly small rank. We discuss the efficient implementation of Hammarling's method and propose among other algorithmic improvements a block variant, which is demonstrated to perform significantly better than existing implementations. An extension to the discrete-time Lyapunov equation $A^T XA - X = -C$ is also described.

1 Introduction

The matrix equation

$$A^T X + XA = -C, \qquad C = C^T, \tag{1}$$

with the coefficients $A, C \in \mathbb{R}^{n\times n}$ and the desired solution $X \in \mathbb{R}^{n\times n}$, is called the *(continuous-time) Lyapunov equation*. In the following, we assume that A is stable, i.e., all its eigenvalues have negative real part. This condition implies the unique existence and symmetry of X. Solving (1) is an important part of some numerical methods in control, most notably balanced truncation model reduction [2, 6].

Often, C is positive semidefinite and has the representation $C = B^T B$ with $B \in \mathbb{R}^{m\times n}$ and $m \leq n$. In this case, it is well known that X is also positive semidefinite and can be written in factorized form $X = U^T U$ as well. Hammarling's method [15] for solving (1) is an extension of the Bartels-Stewart method [4] and obtains the Cholesky factor U directly, without computing X first. This has several advantages, in particular increased numerical stability. Also, when (1) appears within a Krylov subspace or ADI method for solving Lyapunov equations then typically U and not X is needed, see, e.g., [22].

Hammarling's method consists of three steps. First, using orthogonal transformations, A is reduced to real Schur form. Then the reduced equation is solved. Finally, a Cholesky factor of X is obtained by a back transformation, applying the orthogonal transformations from the first step. In this paper, we mainly consider the second step and therefore focus on the solution of the reduced equation

$$A^T(U^T U) + (U^T U)A = -B^T B, \tag{2}$$

*Department of Computing Science and HPC2N, Umeå University, S-90187 Umeå, Sweden, kressner@cs.umu.se. The research of this author was supported by the DFG Emmy Noether fellowship KR 2950/1-1 and the Swedish Foundation for Strategic Research under grant A3 02:128.

where A is a block upper triangular matrix with 1×1 and 2×2 blocks on the diagonal. Hammarling's method essentially solves (2) by a forward substitution process, very similar to the forward substitution for solving lower triangular systems of equations [14]. Two modifications of this method were proposed by Sorensen and Zhou in [28]. First, a reformulation for Sylvester equations involving 2×2 diagonal blocks of A, corresponding to complex conjugate pairs of eigenvalues, is given. It is pointed out that this reformulation leads to a more efficient implementation, but it could also adversely affect the numerical stability properties. Second, a new technique for updating the right-hand side of (2) during the forward substitution is given. This reduces the number of floating point operations (flops) for $m < n$. Extensions of Hammarling's method to generalized Lyapunov equations can be found in [23, 29].

The purpose of this paper is to present a block variant of Hammarling's method, which – depending on the matrix size and computer architecture – can be much faster than the state-of-the-art implementation contained in SLICOT [5]. A general approach to derive block algorithms for linear algebra algorithms is to partition all coefficients into 2×2 blocks and solve the corresponding subsystems recursively. The coupling relation between these subsystems can often be represented by matrix-matrix multiplications, giving rise to highly efficient implementations almost entirely based on level 3 BLAS [13]. This methodology has been successfully used in [17, 25] to derive recursive algorithms for Lyapunov equations of the form (1) and other matrix equations. However, it will be seen that the direct application of such an approach to the factorized form (2) results in a numerically extremely unstable algorithm. We propose an alternative approach, which is still rich in level 3 BLAS operations but avoids the modification of numerically critical parts in Hammarling's method.

The rest of this paper is organized as follows. In Section 2, we present Hammarling's method in a slightly modified form, making it more suitable for $m < n$. Moreover, it is demonstrated that a straightforward recursive formulation has dissatisfying numerical properties, to put it mildly. In Section 3, the new block variant is described. Section 4 is an extension of these ideas to discrete-time Lyapunov equations. Section 5 is devoted to numerical experiments. In particular, it is shown how the combination of our block variant with recent improvements of the QR algorithm results in a more efficient solver for general, unreduced Lyapunov equations.

2 Hammarling's Method

To describe Hammarling's method, let us consider the reduced Lyapunov equation (2) and partition

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}, \tag{3}$$

where $A_{11} \in \mathbb{R}^{n_b \times n_b}, U_{11} \in \mathbb{R}^{n_b \times n_b}, B_1 \in \mathbb{R}^{m \times n_b}$, with $n_b = 1$ or $n_b = 2$, depending on whether the $(2,1)$ entry of A is zero. First, a QR decomposition is applied to the first n_b columns of B, yielding an $m \times m$ orthogonal matrix Q_1 such that

$$Q_1^T \begin{bmatrix} B_1 & B_2 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix} \tag{4}$$

Algorithm 1 Hammarling's method for continuous-time Lyapunov equations

Input: An $n \times n$ upper triangular matrix A and an $m \times n$ matrix B.
Output: An $n \times n$ upper triangular matrix U solving (2).

for $j \leftarrow 1, 2, \ldots, n$ **do**
  Let Q_j be a Householder reflector that maps the jth column of B to a multiple of e_1.
  $B(:, j:n) \leftarrow Q_j B(:, j:n)$, $U(j,j) \leftarrow B(1,j)/\sqrt{-2A(j,j)}$, $f_j \leftarrow B(1,j)/U(j,j)$.
  Determine right-hand side $r \leftarrow -B(1, j+1:n)^T f_j - A(j, j+1:n)^T U(j,j)$.
  Compute $U(j, j+1:n)^T \leftarrow (A(j+1:n, j+1:n)^T + A(j,j) I_{n-j})^{-1} r$ by solving a linear system.
  Update $B(1, j+1:n) \leftarrow B(1, j+1:n) - f_j U(j, j+1:n)$.
end for

with $B_{11} \in \mathbb{R}^{\min\{m,n_b\} \times n_b}$. Then it can be shown that (2) is equivalent to

$$\begin{aligned} A_{11}^T (U_{11}^T U_{11}) + (U_{11}^T U_{11}) A_{11} &= -B_{11}^T B_{11}, & (5) \\ A_{22}^T U_{12}^T + U_{12}^T (U_{11} A_{11} U_{11}^{-1}) &= -B_{12}^T F_1 - A_{12}^T U_{11}^T, & (6) \\ A_{22}^T (U_{22}^T U_{22}) + (U_{22}^T U_{22}) A_{22} &= -\tilde{B}_{22}^T \tilde{B}_{22}, & (7) \end{aligned}$$

where

$$\tilde{B}_{22} = \begin{bmatrix} B_{12} - F_1 U_{12} \\ B_{22} \end{bmatrix}, \quad F_1 = B_{11} U_{11}^{-1}.$$

This yields the following forward substitution process. For $n_b = 1$, the scalar equation (5) is trivially solved by setting $U_{11} = B_{11}/\sqrt{-2A_{11}}$. For $n_b = 2$, the solution of (5) can be reduced to the 1×1 case using the complex Schur decomposition of A_{11}, see [15] for more details. Having U_{11} determined, equation (6) becomes a Sylvester equation in U_{12}, which can be reformulated as an $n_b(n-n_b) \times n_b(n-n_b)$ upper triangular linear system using Kronecker products [14]. Note that (6) is not well-defined if U_{11} is singular. This issue poses no difficulty for $n_b = 1$ ($U_{11} = 0$ implies $U_{12} = 0$) and can also be addressed for $n_b = 2$, see [15, Sec. 6]. With U_{12} being determined, (7) becomes an $(n-n_b) \times (n-n_b)$ Lyapunov equation in factorized form, which can be solved by recursively applying the above procedure.

Algorithm 1 provides a MATLAB-like description of Hammarling's method, restricted to the case of an upper triangular matrix A for simplicity. It requires approximately $\frac{1}{3}n^3 + 2mn^2$ flops, making its cost for $m \ll n$ comparable to the multiplication of two triangular $n \times n$ matrices. In the case $m \ll n$, the bulk of the computational work in each loop consists of solving a $(n-j) \times (n-j)$ upper triangular linear system by backward substitution, which requires $(n-j)^2$ flops and access to $\mathcal{O}((n-j)^2)$ words of memory. This results in a rather high communication/computation ratio for the overall algorithm, leading to poor performance on computer architectures with a memory hierarchy, see also [12]. Another point to mention is that the matrices U and B are accessed row-wise which is disadvantageous when the matrices are stored column-wise, as it is usually done.

Remark 1 *Note that Algorithm 1 differs from the original description of Hammarling's method. In [15], it was proposed to reduce B to upper trapezoidal form prior to any other computation. The upper trapezoidal form of B is maintained throughout all subsequent computations using QR updating techniques as described, e.g., in [14]. This requires $3n^2m + \mathcal{O}(m^2n)$ flops for the orthogonal transformations of B alone. In contrast, the corresponding part in Algorithm 1 requires only $2n^2m + \mathcal{O}(m^2n)$ flops. Also for $m = \mathcal{O}(n)$, it can be shown that Algorithm 1 requires less flops. If, however, m is significantly larger than n, say $m > 3n/2$,*

then the initial QR decomposition becomes the dominating part of the computation in the original description. Since this part can be addressed by a block algorithm based on compact WY representations [14], we may expect better performance.

A direct way to enhance the poor performance of Algorithm 1 is to increase the value of n_b in the equations (5)–(7) from 1 or 2 to $n_b = \lfloor n/2 \rfloor$. Note that the value of n_b might need to be slightly adjusted to avoid cutting 2×2 blocks in A. Now, both Lyapunov equations (5) and (7) are roughly of order $n/2$ and can be solved recursively. Moreover, the size of the two matrix coefficients in the Sylvester equation (6) is balanced, which means that RECSY [17] can be invoked to solve this equation very efficiently. This idea eventually yields a recursive Hammarling method with most of the computational work consisting of large matrix-matrix multiplications. We can therefore expect performance improvements that are quite similar to the dramatic improvements reported in [17] for the recursive solution of Lyapunov equations in non-factorized form. However, the presence of U_{11}^{-1} in (6) and (7) raises doubts about the finite-precision behavior of such an approach. The singular values of solutions of Lyapunov equations often decay rapidly, in particular for right-hand sides of low rank [3, 24]. This implies that U_{11} may become very ill-conditioned as n_b increases and represents a potential source for numerical instability. To clarify this issue, we have applied a straightforward MATLAB implementation, denoted by `reclyapchol` below, to the matrices

$$A = \mathrm{diag}(-1, -2, -3, \ldots, -n), \qquad B = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix},$$

For Lyapunov equations of order at most 2 arising during the recursion, the SLICOT-based MATLAB control toolbox function `lyapchol` was used. In the following table, we report the Frobenius norms of the residuals,

$$\|A^T(U^TU) + (U^TU)A + B^TB\|_F,$$

for `lyapchol` (applied to the $n \times n$ Lyapunov equation) and `reclyapchol`.

n	4	8	16	32	64	128
`lyapchol`	10^{-16}	10^{-16}	10^{-15}	10^{-15}	10^{-14}	10^{-14}
`reclyapchol`	10^{-16}	10^{-15}	10^{-13}	10^{-5}	10^{26}	10^{168}

It can be seen that `reclyapchol` becomes extremely inaccurate as n increases. This demonstrates that a naive recursive formulation of Hammarling's method is an infeasible means to increase performance.

3 A Block Variant

In the following, we derive a block variant that enables the use of level 3 BLAS operations without abandoning numerical stability. For this purpose, let us assume that A is upper triangular and partition the matrices A, B and U in (2) as follows:

$$A = \begin{bmatrix} \tilde{A} & A_p & \star \\ 0 & A_d & \star \\ 0 & 0 & \star \end{bmatrix}, \quad B = \begin{bmatrix} \tilde{B} & B_p & \star \end{bmatrix}, \quad U = \begin{bmatrix} \tilde{U} & U_p & \star \\ 0 & U_d & \star \\ 0 & 0 & \star \end{bmatrix}, \tag{8}$$

where $\tilde{A}, \tilde{U} \in \mathbb{R}^{l \times l}$, $A_p, U_p \in \mathbb{R}^{l \times k}$, $A_d, U_d \in \mathbb{R}^{k \times k}$, $\tilde{B} \in \mathbb{R}^{m \times l}$, and $B_p \in \mathbb{R}^{m \times k}$, see also Figure 1. Assume that $\tilde{U}$ has been determined, e.g., by applying Algorithm 1 to $\tilde{A}$ and

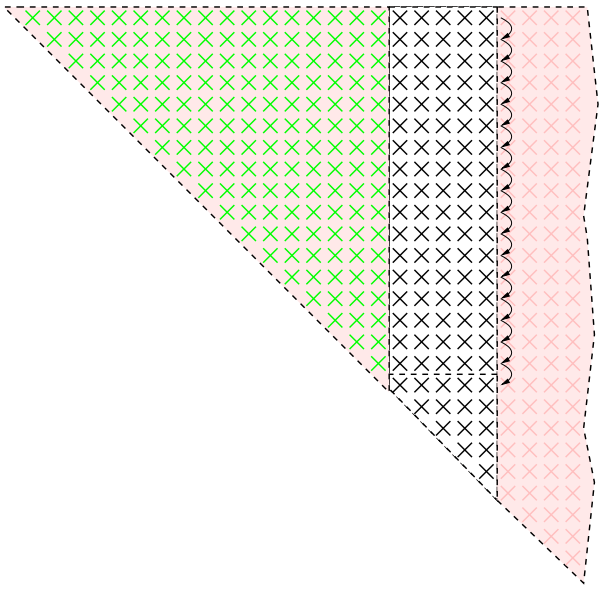

Figure 1: Illustration of the partitioning of U in (8). The arrows indicate data dependencies between rows within the block column panel and determine the order in which the rows must be processed.

$\tilde{B}$. Moreover, the quantities $Q_1, \ldots, Q_l$ and $f_1, \ldots, f_l$ produced during the algorithm are supposed to be kept in storage. Let us now investigate the arithmetic operations that have to be performed in order to obtain u_{1p}, the first row of the next block column panel U_p. First, we need to update B_p:

$$B_p \leftarrow Q_1 B_p.$$

To set up the linear system that determines u_{1p}, we note that

$$\left(\begin{bmatrix} \tilde{A}^T & 0 \\ A_p^T & A_d^T \end{bmatrix} + a_{11}\, I \right) \begin{bmatrix} \tilde{u}_1^T \\ u_{1p}^T \end{bmatrix} = \begin{bmatrix} \star \\ b_{1p}^T f_1 \end{bmatrix},$$

where $\tilde{u}_1$ denotes the solution that has been obtained in the first loop of Algorithm 1 applied to $\tilde{A}$ and $\tilde{B}$, and b_{1p} denotes the first row of B_p. This implies

$$(A_d^T + a_{11}\, I) u_{1p}^T = r_p - A_p^T \tilde{u}_1^T = b_{1p}^T f_1 - (e_1^T \tilde{U} A_p)^T.$$

Finally, b_{1p} needs to be updated again,

$$b_{1p} \leftarrow b_{1p} - f_1 u_{1p}.$$

Similarly, to compute the second row u_{2p} we first update

$$B_p \leftarrow Q_2 B_p,$$

solve the linear system

$$(A_d^T + a_{22}\, I) u_{2p}^T = b_{1p}^T f_2 - (e_2^T \tilde{U} A_p)^T.$$

Algorithm 2 Block algorithm for continuous-time Lyapunov equations

Input: An $n \times n$ upper triangular matrix A, an $m \times n$ matrix B, and a block size k.
Output: An $n \times n$ upper triangular matrix U solving (2).

$i \leftarrow 1$
while $i \leq n$ **do**
  $l \leftarrow \min\{n, i+k-1\}$.
  Compute diagonal block $U(i:l, i:l)$ by applying Algorithm 1 to $A(i:l, i:l)$ and $B(:, i:l)$.
  % The reflectors $Q_1, \ldots, Q_l$ and scalars $f_1, \ldots, f_l$ resulting from Algorithm 1 are kept in workspace.
  % Update next block column panel $l+1:l+k$.
  $p_f \leftarrow l+1, \qquad p_l \leftarrow \min\{n, l+k\}$.
  $U(1:l, p_f:p_l) \leftarrow U(1:l, 1:l)A(1:l, p_f:p_l)$
  for $j \leftarrow 1, 2, \ldots, l$ **do**
    $B(:, p_f:p_l) \leftarrow Q_j B(:, p_f:p_l)$.
    Determine right-hand side $r_p \leftarrow -B(1, p_f:p_l)^T f_j - U(j, p_f:p_l)^T$.
    Compute $U(j, p_f:p_l)^T \leftarrow (A(p_f:p_l, p_f:p_l)^T + A(j,j)I)^{-1} r_p$ by solving a linear system.
    Update $B(1, p_f:p_l) \leftarrow B(1, p_f:p_l) - f_j U(j, p_f:p_l)$.
  end for
  $i \leftarrow l+1$.
end while

and update again

$$b_{1p} \leftarrow b_{1p} - f_2 u_{2p}.$$

More general, the determination of the jth row u_{jp} requires the update of B_p and the solution of a linear system

$$(A_d^T + a_{jj}\, I) u_{jp}^T = b_{1p}^T f_j - (e_j^T \tilde{U} A_p)^T. \tag{9}$$

Because of the complicated dependencies in the updates of b_{1p} we see little hope to block the complete process. Note that for $m, k \ll l$ the computationally most expensive part is the determination of the jth row of $\tilde{U}A_p$ in (9), which does not depend on previously computed data within the panel. Our block variant attains its efficiency by precomputing $\tilde{U}A_p$ with a single matrix-matrix multiplication (by calling the level 3 BLAS `DTRMM`).

Once all $u_{1p}, \ldots, u_{lp}$ have been determined, we can compute U_d by applying Algorithm 1 to A_d and the updated B_p. Then we choose the columns $l+k+1, \ldots, l+2k$ as the next block column panel and repeat the process described above. This is summarized in Algorithm 2.

If $m \ll n$, the algorithm has the pleasing property to be approximately entirely based on level 3 BLAS, i.e, as $n/k \to \infty$ the portion of flops spent in calls to `DTRMM` approaches 100%. Still, good performance can only be attained if the block size k is chosen in a suitable manner. This point is emphasized, e.g., in [17, 18], where it is demonstrated that recursion automatically leads to nearly optimal block partitionings. Also, the numerical experiments in [17] clearly show the large impact of kernel solvers for tiny Lyapunov and Sylvester equations on the performance of the overall algorithm. Our implementation of Algorithm 2 is currently based on the very reliable kernel solvers provided in SLICOT. We expect further improvements from optimized super-scalar kernels that use explicit loop unrolling and other techniques assisting compiler optimization.

Note that during the execution of Algorithm 2, the orthogonal matrices $Q_1, \ldots, Q_n$ need to be stored. Fortunately, each Q_j can be represented by a Householder vector of length m and therefore fits precisely in the subsequently unused jth column of B. The scalars $f_1, \ldots, f_n$

and the diagonal elements of A (to maintain data locality) are stored in a separate work space array of length $\mathcal{O}(n)$.

As m increases, the orthogonal transformations of B become a non-negligible portion of the overall computational work. Although not based on level 3 BLAS the updates of B are only performed locally within stripes of k columns; we can therefore expect that performance will not be severely affected as m increases. However, in preliminary computational experiments we observed for $m \geq 2k$ that it is beneficial to initially reduce columns $i, \ldots, l$ of B to upper triangular form in each outer loop of Algorithm 2. This allows to rewrite the orthogonal transformation of $B(:, p_f : p_l)$ in the inner loop mainly in terms of compact WY representations of order k.

Finally, let us remark that Algorithm 2 can be extended without any difficulties to a matrix A having 2×2 blocks on the diagonal. The main difference is that two rows of the block column panel corresponding to such a 2×2 block are processed simultaneously requiring the solution of a Sylvester equation instead of the linear system (9).

4 Extension to the Discrete-Time Lyapunov Equation

Hammarling's method for computing the factorized solution of the reduced discrete-time Lyapunov equation (also called Stein equation)

$$A^T(U^TU)A - (U^TU) = -B^TB \tag{10}$$

is rather similar to the continuous-time case. Here, the d-stability of A (all eigenvalues are inside the unit disk) implies the existence of U. Again we consider a partitioning of the form (3) with $n_b \in \{1, 2\}$ and perform the QR decomposition (4) of B. Then (10) is equivalent to

$$\begin{aligned}
A_{11}^T(U_{11}^TU_{11})A_{11} - (U_{11}^TU_{11}) &= -B_{11}^TB_{11}, && (11)\\
A_{22}^TU_{12}^T(U_{11}A_{11}U_{11}^{-1}) - U_{12}^T &= -B_{12}^TF_1 - A_{12}^TU_{11}^T(U_{11}A_{11}U_{11}^{-1}), && (12)\\
A_{22}^T(U_{22}^TU_{22})A_{22} - (U_{22}^TU_{22}) &= -\tilde{B}_{22}^T\tilde{B}_{22}, && (13)
\end{aligned}$$

where

$$\tilde{B}_{22} = \begin{bmatrix} G_1^T(U_{12}A_{22} + U_{11}A_{12}) + H_1^TB_{12} \\ B_{22} \end{bmatrix}, \quad F_1 = B_{11}U_{11}^{-1}, \tag{14}$$

and G_1, H_1 are chosen such that

$$\begin{bmatrix} U_{11}A_{11}U_{11}^{-1} & G_1 \\ F_1 & H_1 \end{bmatrix}$$

is an orthogonal matrix. Note that (11) implies the orthogonality of the first n_b columns of this matrix and that for $n_b = 1$ we can simply set $G_1 = F_1$, $H_1 = -A_{11}$. For $n_b = 2$, this and related update formulas for $\tilde{B}_{22}$ are discussed in [16, 30]; its implementation in the SLICOT routine `SB03OT` is based on a 4×2 QR decomposition. Similarly as (5)–(7), the equations (11)–(13) can be solved recursively. This yields Algorithm 3, which is, again for simplicity only, restricted to an upper triangular matrix A and requires approximately $\frac{2}{3}n^3 + 2mn^2$ flops. Note the increase of flops in comparison to Algorithm 1. This increase is due to the need for computing $U(j, j : n)A(j : n, j + 1 : n)$ in the update of B and is actually avoidable; this

Algorithm 3 Hammarling's method for discrete-time Lyapunov equations

Input: An $n \times n$ upper triangular matrix A and an $m \times n$ matrix B.
Output: An $n \times n$ upper triangular matrix U solving (10).

for $j \leftarrow 1, 2, \ldots, n$ **do**
  Let Q_j be a Householder reflector that maps the jth column of B to a multiple of e_1.
  $B(:, j:n) \leftarrow Q_j B(:, j:n)$, $\quad U(j,j) \leftarrow B(1,j)/\sqrt{1-|A(j,j)|^2}$, $\quad f_j \leftarrow B(1,j)/U(j,j)$.
  Determine right-hand side $r \leftarrow -B(1, j+1:n)^T f_j - A(j, j+1:n)^T U(j,j) A(j, j)$.
  Compute $U(j, j+1:n)^T \leftarrow (A(j,j)A(j+1:n, j+1:n)^T - I_{n-j})^{-1} r$ by solving a linear system.
  Update $B(1, j+1:n) \leftarrow f_j U(j, j:n) A(j:n, j+1:n) - A(j,j) B(1, j+1:n)$.
end for

Algorithm 4 Block algorithm for discrete-time Lyapunov equations

Input: An $n \times n$ upper triangular matrix A, an $m \times n$ matrix B, and a block size k.
Output: An $n \times n$ upper triangular matrix U solving (10).

$i \leftarrow 1$
while $i \leq n$ **do**
  $l \leftarrow \min\{n, i+k-1\}$.
  Compute diagonal block $U(i:l, i:l)$ by applying Algorithm 3 to $A(i:l, i:l)$ and $B(:, i:l)$.
  % The reflectors $Q_1, \ldots, Q_l$ and scalars $f_1, \ldots, f_l$ resulting from Algorithm 3 are kept in workspace.
  % Update next block column panel $l+1:l+k$.
  $p_f \leftarrow l+1$, $\quad p_l \leftarrow \min\{n, l+k\}$.
  $U(1:l, p_f:p_l) \leftarrow U(1:l, 1:l) A(1:l, p_f:p_l)$
  for $j \leftarrow 1, 2, \ldots, l$ **do**
    $B(:, p_f:p_l) \leftarrow Q_j B(:, p_f:p_l)$.
    Save $w \leftarrow U(j, p_f:p_l)$
    Determine right-hand side $r_p \leftarrow -B(1, p_f:p_l)^T f_j - w^T A(j,j)$.
    Compute $U(j, p_f:p_l)^T \leftarrow (A(j,j)A(p_f:p_l, p_f:p_l)^T - I)^{-1} r_p$ by solving a linear system.
    Update $B(1, p_f:p_l) \leftarrow f_1 w + f_1 U(j, p_f:p_l) A(p_f:p_l, p_f:p_l) - A(j,j) B(1, p_f:p_l)$.
  end for
  $i \leftarrow l+1$.
end while

computation can be combined with the forward substitution process when solving the linear system. This technique, which is naturally included in our block variant, reduces the number of flops back to $\frac{1}{3}n^3 + 2mn^2$.

A block variant of Algorithm 3 can be derived along the lines of Section 3. We omit the details and only present the resulting algorithm. The remarks concerning the properties and implementation of Algorithm 2 apply likewise to Algorithm 4.

5 Numerical Experiments

To test the proposed block variants, we have implemented Algorithms 2 and 4 in a Fortran 77 routine called `LRLYAP`[1] and performed various numerical experiments on two machine platforms with different characteristics:

- Athlon MP2000+ with 64 kB instruction and 64 kB data L1 Cache, 256 kB of integrated

[1]The Fortran routine along with an example program and MATLAB interfaces are available from `http://www.cs.umu.se/~kressner/lyapunov.php`.

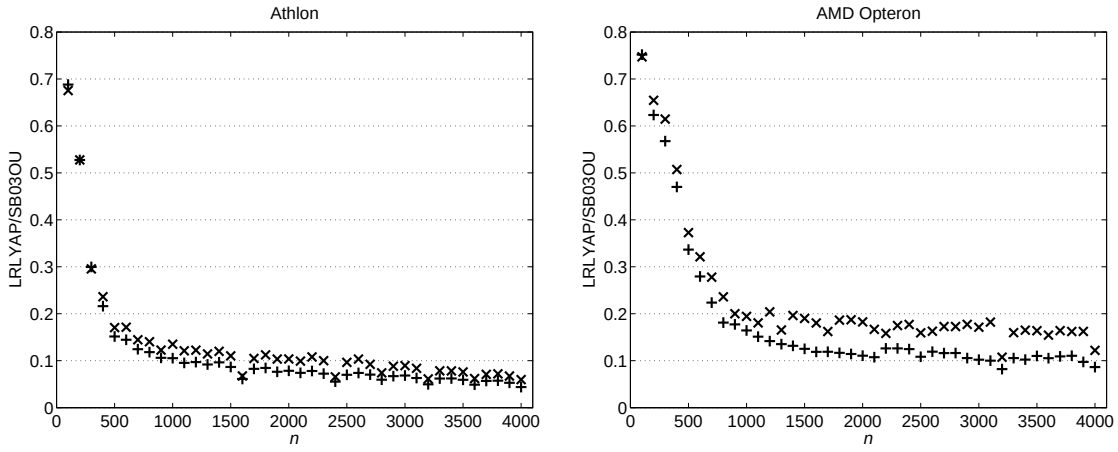

Figure 2: Ratios of execution times between new routine `LRLYAP` and SLICOT routine `SB03OU` for solving reduced, continuous-time ($\times$) and discrete-time Lyapunov ($+$) equations with $m = 100$.

L2 cache. Software used: Debian GNU/Linux 3.0, Portland F90 6.0, ATLAS BLAS 3.5.9.

- AMD Opteron 248 with 64 kB instruction and 64 kB data L1 Cache (2-way associative), 1024 kB unified L2 Cache (16-way associative). Software used: Debian GNU/Linux 3.1, Portland F90 6.0, Goto BLAS 0.94.

Several subroutines from LAPACK 3.0 [1] and SLICOT 4.0 [5] are called by `LRLYAP`. We have compared our routine with the SLICOT routine `SB03OU` for solving reduced Lyapunov equations, which is – to the best of our knowledge – the current state-of-the-art implementation of Hammarling's method. We refer to [27] for an evaluation of the matrix equation solvers contained in SLICOT.

All results reported are run on one processor in double precision real arithmetic ($\epsilon_{mach} \approx 2.2 \times 10^{-16}$).

5.1 Performance comparison to `SB03OU`

First, random upper triangular matrices $A \in \mathbb{R}^{n \times n}$ and random matrices $B \in \mathbb{R}^{m \times n}$ have been generated for $m = 100$ and $n \in \{100, 200, 300, \ldots, 4000\}$. Note that some care needs to be applied when generating A in order to avoid gradual under- or overflow in the solution, which would severely degrade the performance [21]. The block size k is chosen optimally from the set $\{8, 16, 24, \ldots, 120\}$. On both platforms it was observed that the performance was not very sensitive with respect to the choice of k; any choice between 48 and 100 resulted in an execution time that was at most 10% larger than the minimum. Figure 2 displays the ratio between the measured execution times of `SB03OU` and `LRLYAP`. The speedups range between a factor of 6 and 22 for $n \geq 1500$. It can be observed that the speedups are higher for the discrete-time Lyapunov equation, which was to be expected from the discussion in Section 4. Also, the speedups are more impressive on the Athlon MP2000+, which we attribute to the small L2 cache size. To provide a concrete example, let us consider the case $n = 4000$. On

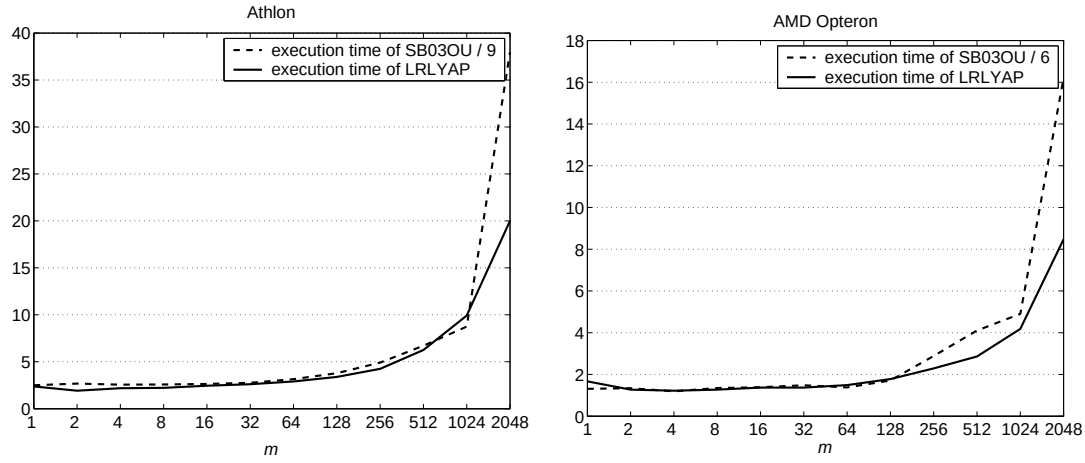

Figure 3: Execution times of SLICOT routine `SB03OU` and new routine `LRLYAP` for solving reduced, 2000×2000 continuous-time Lyapunov equations. Note that the numbers for `SB03OU` have been divided by 9 and 6, respectively.

Athlon, `SB03OU` requires 310 seconds (continuous-time case) and 452 seconds (discrete-time case); `LRLYAP` reduces these times to 18.5 and 19.8 seconds, respectively. On AMD Opteron, the corresponding figures are 82 and 126 seconds for `SB03OU`, but only 10.1 and 10.9 seconds for `LRLYAP`.

Next, we investigated the effect of increasing m on the performance. For this purpose, we kept $n = 2000$ fixed and varied m in $\{1, 2, 4, 8, \ldots, 2048\}$. The measured execution times are displayed in Figure 3. The speedup factor increases from 9.5 to 17 on Athlon and from 4.7 to 11.5 on AMD Opteron. Pleasantly, the sensitivity with respect to the choice of block size k remains modest as m increases; any choice between 48 and 88 resulted in an execution time that was at most 20% larger than the minimum.

5.2 Impact on the solution of unreduced Lyapunov equations

A solver for general, unreduced Lyapunov equations in factorized form consists of the following steps:

(i) Hessenberg reduction of A;

(ii) application of QR algorithm to reduce A further to real Schur form;

(iii) application of orthogonal matrix Q accumulated in Steps (i)+(ii) to right-hand side: $B \leftarrow BQ$;

(iv) solution of reduced Lyapunov equation;

(v) back transformation of solution: $U \leftarrow UQ^T$.

In this paper, we have only considered improvements to Step (iv). Recently, drastic improvements have been attained for the QR algorithm employed in Step (ii), using multishift and aggressive early deflation techniques [7, 8]. Also, Step (i) benefits from the changes proposed

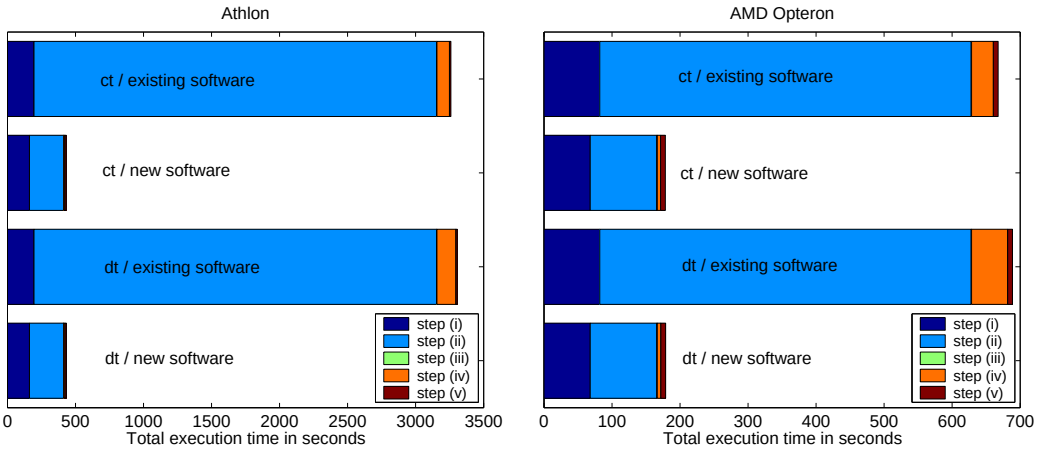

Figure 4: Total execution times for solving unreduced, 3000×3000 continuous-time (ct) and discrete-time (dt) Lyapunov equations with $m = 100$.

in [26]. Both improvements are included in LAPACK 3.1. Figure 4 reports the total execution time of existing software (LAPACK 3.0 and SLICOT) compared to newly developed software (LAPACK 3.1 and the routine `LRLYAP` described in this paper). It is interesting to note that without using `LRLYAP` the portion spent for solving the reduced equations would raise from $3\% \ldots 8\%$ to $18\% \ldots 32\%$. The use of `LRLYAP` brings these figures back to insignificant $2\% \ldots 3\%$. On the other hand, Figure 4 also quite clearly shows that the impovements of Step (ii) have a much larger impact on the overall performance than those of Step (iv). Of course, the impact of Step (iv) increases when multiple solutions of the same Lyapunov equation with different right-hand sides are required as, e.g., in balanced truncation model reduction [2, 6] or condition estimation [9].

5.3 Accuracy

To verify that the proposed block variants do not affect the numerical stability of Hammarling's method, we considered the benchmark collections CTDSX [19] and DTDSX [20], which contain various examples of linear time-invariant systems (A, B, C, D) with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$. Each of these systems gives rise to a pair of Lyapunov equations

$$A(U_c^T U_c) + (U_c^T U_c)A^T = -BB^T, \quad A^T(U_o^T U_o) + (U_o^T U_o)A = -C^T C, \tag{15}$$

in the continuous-time case (CTDSX) and

$$A(U_c^T U_c)A^T - (U_c^T U_c) = -BB^T, \quad A^T(U_o^T U_o)A - (U_o^T U_o) = -C^T C, \tag{16}$$

in the discrete-time case (DTDSX). We have applied `SB03OU` as well as `LRLYAP` to the solution of (15) and (16), after A has been balanced and reduced to Schur form, and B, C have been transformed accordingly. For the back transformed solutions, we have measured the Frobenius norms of the residuals

$$r_c = \|A(U_c^T U_c) + (U_c^T U_c)A^T + BB^T\|_F, \quad r_o = \|A^T(U_o^T U_o) + (U_o^T U_o)A + C^T C\|_F, \tag{17}$$

				SB03OU		LRLYAP		
Ex.	n	m	p	r_c	r_o	r_c	r_o	k
1.3	4	2	4	5×10^{-15}	2×10^{-13}	8×10^{-15}	2×10^{-13}	2
1.4	8	2	8	4×10^{-18}	1×10^{-14}	5×10^{-18}	9×10^{-15}	4
1.5	9	3	9	2×10^{-14}	7×10^{-14}	2×10^{-14}	6×10^{-14}	4
1.6	30	3	5	8×10^{-07}	3×10^{-08}	8×10^{-07}	3×10^{-08}	8
1.8	9	3	2	2×10^{-08}	2×10^{-03}	2×10^{-08}	1×10^{-03}	4
3.4	421	211	211	1×10^{-10}	$6\times10^{+14}$	1×10^{-10}	$3\times10^{+14}$	32
4.1	100	1	1	1×10^{-11}	1×10^{-15}	1×10^{-11}	1×10^{-15}	16
4.2	60	2	60	2×10^{-02}	$2\times10^{+02}$	1×10^{-01}	$8\times10^{+01}$	16

Table 1: Obtained residuals (17) for CTDSX examples.

				SB03OU		LRLYAP		
Ex.	n	m	p	r_c	r_o	r_c	r_o	k
1.7	4	2	4	1×10^{-14}	3×10^{-13}	1×10^{-14}	7×10^{-14}	2
1.8	4	4	4	3×10^{-10}	8×10^{-13}	4×10^{-10}	1×10^{-12}	2
1.9	5	2	5	1×10^{-16}	2×10^{-13}	1×10^{-16}	2×10^{-13}	2
1.10	6	2	2	0	2×10^{-15}	0	0	2
1.11	9	3	2	2×10^{-17}	8×10^{-14}	2×10^{-17}	8×10^{-14}	3
2.1	4	1	1	3×10^{-24}	0	2×10^{-24}	0	2
3.1	100	1	100	0	8×10^{-14}	0	9×10^{-14}	16

Table 2: Obtained residuals (18) for DTDSX examples.

and

$$r_c = \|A(U_c^T U_c)A^T - (U_c^T U_c) + BB^T\|_F \quad r_o = \|A^T(U_o^T U_o)A - (U_o^T U_o) + C^T C\|_F, \qquad (18)$$

respectively. We have only selected these examples from CTDSX and DTDSX for which the system matrix is stable and $n \geq 4$. The block size k has been chosen low enough to enable the block algorithm.

From Tables 1 and 2, it can be seen that the use of LRLYAP does not lead to significant changes in the residuals. Further tests with various types of random matrices confirm this observation. Moreover, it has been observed that the most notable differences in the residuals stem from the different ways of updating the right-hand side as explained in Remark 1.

6 Conclusions and Open Questions

Algorithmically improved, block variants of Hammarling's method have been developed, leading to implementations that significantly outperform existing software for solving Lyapunov equations without abandoning numerical stability.

We expect that the performance can be increased even further by developing highly efficient kernel solvers and numerically reliable ways of performing recursion. However, in view of the results in Section 5.2 such further improvements may only have a limited impact on the overall execution time required for solving an unreduced Lyapunov equation. Another direction of future research is to investigate whether the ideas from this paper lead to parallel

distributed memory algorithms competitive with existing parallel variants of Hammarling's method, see, e.g., [10, 11].

7 Acknowledgments

The computational experiments in this paper were performed using facilities of the High Performance Computing Center North (HPC2N) in Umeå. The author is grateful to Vasile Sima and the referees for pointing out several erratas in an earlier version of this paper.

References

[1] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[2] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, Philadelphia, PA, 2005.

[3] A. C. Antoulas, Sorensen D. C., and Zhou Y. On the decay rate of the Hankel singular values and related issues. *Sys. Control Lett.*, 46(5):323–342, 2002.

[4] R. H. Bartels and G. W. Stewart. Algorithm 432: The solution of the matrix equation $AX - BX = C$. *Communications of the ACM*, 8:820–826, 1972.

[5] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT—a subroutine library in systems and control theory. In *Applied and computational control, signals, and circuits, Vol. 1*, pages 499–539. Birkhäuser Boston, Boston, MA, 1999. See also `http://www.slicot.de`.

[6] P. Benner, V. Mehrmann, and D. C. Sorensen, editors. *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[7] K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. I. Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix Anal. Appl.*, 23(4):929–947, 2002.

[8] K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. II. Aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 23(4):948–973, 2002.

[9] R. Byers. A LINPACK-style condition estimator for the equation $AX - XB^T = C$. *IEEE Trans. Automat. Control*, 29(10):926–928, 1984.

[10] J. M. Claver and V. Hernández. Parallel wavefront algorithms solving Lyapunov equations for the Cholesky factor on message passing multiprocessors. *The Journal of Supercomputing*, 13:171–189, 1999.

[11] J. M. Claver, V. Hernández, and E. S. Quintana-Ortí. Parallel cyclic wavefront algorithms for solving semidefinite Lyapunov equations. In P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, and D. Ruiz, editors, *Euro-Par'99 - Parallel Processing:*

5th International Euro-Par Conference, Toulouse, France. Springer Berlin / Heidelberg, 1999.

[12] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[13] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[15] S. Hammarling. Numerical solution of the stable, nonnegative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2(3):303–323, 1982.

[16] S. Hammarling. Numerical solution of the discrete-time, convergent, nonnegative definite Lyapunov equation. *Systems Control Lett.*, 17(2):137–139, 1991.

[17] I. Jonsson and B. Kågström. Recursive blocked algorithm for solving triangular systems. I. One-sided and coupled Sylvester-type matrix equations. *ACM Trans. Math. Software*, 28(4):392–415, 2002.

[18] I. Jonsson and B. Kågström. Recursive blocked algorithm for solving triangular systems. II. Two-sided and generalized Sylvester and Lyapunov matrix equations. *ACM Trans. Math. Software*, 28(4):416–435, 2002.

[19] D. Kressner, V. Mehrmann, and T. Penzl. CTDSX - a collection of benchmark examples for state-space realizations of continuous-time dynamical systems. SLICOT working note 1998-9, WGS, 1998.

[20] D. Kressner, V. Mehrmann, and T. Penzl. DTDSX - a collection of benchmark examples for state-space realizations of discrete-time dynamical systems. SLICOT working note 1998-10, WGS, 1998.

[21] O. Lawlor, H. Govind, I. Dooley, M. Breitenfeld, and L. Kale. Performance degradation in the presence of subnormal floating-point values. In *Proc. Workshop on Operating System Interference in High Performance Applications*, 2005.

[22] J.-R. Li and J. White. Low-rank solution of Lyapunov equations. *SIAM Rev.*, 46(4):693–713, 2004.

[23] T. Penzl. Numerical solution of generalized Lyapunov equations. *Adv. Comput. Math.*, 8(1-2):33–48, 1998.

[24] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, 40(2):139–144, 2000.

[25] E. S. Quintana-Ortí and R. A. van de Geijn. Formal derivation of algorithms: the triangular Sylvester equation. *ACM Trans. Math. Software*, 29(2):218–243, 2003.

[26] G. Quintana-Ortí and R. A. van de Geijn. Improving the performance of reduction to Hessenberg form. *ACM Trans. Math. Software*, 32(2), 2006.

[27] M. Slowik, P. Benner, and V. Sima. Evaluation of the linear matrix equation solvers in SLICOT, 2006. To appear in *Journal of Numerical Analysis, Industrial and Applied Mathematics*.

[28] D. C. Sorensen and Y. Zhou. Direct methods for matrix Sylvester and Lyapunov equations. *J. Appl. Math.*, 6:277–303, 2003.

[29] T. Stykel. Numerical solution and perturbation theory for generalized Lyapunov equations. *Linear Algebra Appl.*, 349:155–185, 2002.

[30] A. Varga. A note on Hammarling's algorithm for the discrete Lyapunov equation. *Systems Control Lett.*, 15(3):273–275, 1990.