Numerical Solution of Generalized Lyapunov Equations

Thilo Penzl*

March 13, 1996

Abstract

Two efficient methods for solving generalized Lyapunov equations and their implementations in FORTRAN 77 are presented. The first one is a generalization of the Bartels–Stewart method and the second is an extension of Hammarling's method to generalized Lyapunov equations. Our LAPACK based subroutines are implemented in a quite flexible way. They can handle the transposed equations and provide scaling to avoid overflow in the solution. Moreover, the Bartels–Stewart subroutine offers the optional estimation of the separation and the reciprocal condition number. A brief description of both algorithms is given. The performance of the software is demonstrated by numerical experiments.

Key Words: Mathematical software, generalized Lyapunov equation, generalized Stein equation, condition estimation.
AMS(MOS) subject classifications: 65F05, 65F15, 93B40, 93B51

1 Basic Properties of Generalized Lyapunov Equations

Lyapunov equations play an essential role in control theory. In the past few years its generalizations, the generalized continuous–time Lyapunov equation (GCLE)

$$A^T XE + E^T XA = -Y \tag{1}$$

and the generalized discrete–time Lyapunov equation (GDLE) or generalized Stein equation

$$A^T XA - E^T XE = -Y, \tag{2}$$

have received a lot of interest. A, E, and Y are given real $n \times n$ matrices. The right hand side Y is symmetric and so is the solution matrix X if the equation has a unique solution.

We can consider the GCLE and the GDLE as special cases of the generalized Sylvester equation

$$R^T XS + U^T XV = -Y, \tag{3}$$

where in general, X and Y are $n \times m$ matrices. Since equation (3) is linear in the entries of X, it can be written as a system of linear equations. To this end, the entries of X are usually arranged in a vector by stacking the columns of $X = (x_{ij})_{n\times m}$. This is done by the mapping *vec*, which is defined as

$$\mathrm{vec}(X) = (x_{11}, \ldots, x_{n1}, x_{12}, \ldots, x_{n2}, \ldots, x_{1m}, \ldots, x_{nm})^T.$$

It can easily be shown that (3) is equivalent to

$$\left(S^T \otimes R^T + V^T \otimes U^T\right) \mathrm{vec}(X) = -\mathrm{vec}(Y), \tag{4}$$

where $\otimes$ denotes the Kronecker product of two matrices, e.g. [13]. The order of this system is nm. Therefore, it is not practicable to find X by solving the corresponding system of linear equations unless n and m are very small.

*Technische Universität Chemnitz–Zwickau, Fakultät für Mathematik, 09107 Chemnitz, FRG, E–mail: tpenzl@mathematik.tu-chemnitz.de. Supported by *Deutsche Forschungsgemeinschaft*, research grant *Me 790/7–1 Singuläre Steuerungsprobleme*.

The solvability of (3) depends on the generalized eigenstructure of the matrix pairs (R, U) and (V, S). A matrix pencil $\alpha R - \beta U$ is called *regular* iff there exists a pair of complex numbers (α', β') such that $\alpha' R - \beta' U$ is nonsingular. If $\alpha R x = \beta U x$ holds for a vector $x \neq 0$, the pair $(\alpha, \beta) \neq (0, 0)$ is a *generalized* eigenvalue. Two generalized eigenvalues (α, β) and (γ, δ) are considered to be equal iff $\alpha\delta = \beta\gamma$. The set of all generalized eigenvalues of the pencil $\alpha R - \beta U$ is designated by $\sigma(R, U)$. The following theorem, e.g. [3], gives necessary and sufficient conditions for the existence and uniqueness of the solution of (3).

Theorem 1 *The matrix equation (3) has a unique solution if and only if*

1. $\alpha R - \beta U$ *and* $\alpha V - \beta S$ *are regular pencils and*
2. $\sigma(R, U) \cap \sigma(-V, S) = \emptyset$.

Let (α_i, β_i) denote the *generalized* eigenvalues of the matrix pencil $\alpha A - \beta E$. For simplicity, we switch to the more conventional form of a matrix pencil $A - \lambda E$, whose eigenvalues are given by $\lambda_i = \beta_i / \alpha_i$ with $\lambda_i = \infty$ when $\alpha_i = 0$. Applying the above theorem to (1) and (2) gives the following corollary.

Corollary 1 *Let* $A - \lambda E$ *be a regular pencil. Then*

1. *the GCLE (1) has a unique solution if and only if all eigenvalues of* $A - \lambda E$ *are finite and* $\lambda_i + \lambda_j \neq 0$ *for any two eigenvalues* λ_i *and* λ_j *of* $A - \lambda E$,
2. *the GDLE (2) has a unique solution if and only if* $\lambda_i \lambda_j \neq 1$ *for any two eigenvalues* λ_i *and* λ_j *of* $A - \lambda E$ *(under the convention* $0 \cdot \infty = 1$*).*

As a consequence, singularity of one of the matrices A and E implies singularity of the GCLE (1). If both A and E are singular, the GDLE (2) is singular too. Thus, for an investigation of the unique solution of a generalized Lyapunov equation we may assume one of the matrices A and E to be nonsingular. Since A and E play a symmetric role (up to a sign in the discrete–time case), we expect E to be invertible. Under this assumption equations (1) and (2) are equivalent to the Lyapunov equations

$$(AE^{-1})^T X + X(AE^{-1}) = -E^{-T} Y E^{-1} \tag{5}$$

and

$$(AE^{-1})^T X (AE^{-1}) - X = -E^{-T} Y E^{-1}, \tag{6}$$

respectively, and the classical result about the positive definite solution of the stable Lyapunov equation [14] remains valid for the generalized equations.

Theorem 2 *Let* E *be nonsingular and* Y *be positive definite (semidefinite).*

1. *If* $Re(\lambda_i) < 0$ *for all eigenvalues* λ_i *of* $A - \lambda E$, *then the solution matrix* X *of the GCLE (1) is positive definite (semidefinite).*
2. *If* $|\lambda_i| < 1$ *for all eigenvalues* λ_i *of* $A - \lambda E$, *then the solution matrix* X *of the GDLE (2) is positive definite (semidefinite).*

There may be a definite solution in the discrete–time case even if E is singular. If $|\lambda_i| < 1$ for all eigenvalues λ_i of $E - \lambda A$, then the solution matrix X is negative definite (semidefinite). But this, of course, means nothing but reversing the roles of A and E.

Finally, it should be stressed that the reduction of a generalized equation (1) or (2) to a standard Lyapunov equation (5) or (6), respectively, is mainly of theoretical interest. If E is possibly ill–conditioned, this is not a numerically feasible approach to solve the generalized equation numerically. In the sequel we present two methods which solve generalized Lyapunov equations without inverting E.

2 Algorithms for Generalized Lyapunov Equations

2.1 A Generalization of the Bartels–Stewart Method

The algorithm described in this section is an extension of the Bartels–Stewart method [2] to generalized Lyapunov equations (see also [3], [6], [12]). We restrict ourself to the GCLE. The derivation of the algorithm for the GDLE is straightforward.

First the pencil $A - \lambda E$ is reduced to real generalized Schur form $A_s - \lambda E_s$ by means of orthogonal matrices Q and Z (QZ–algorithm [8]):

$$\begin{aligned} A_s &= Q^T A Z, & (7)\\ E_s &= Q^T E Z & (8) \end{aligned}$$

such that A_s is upper quasitriangular and E_s is upper triangular. Defining

$$\begin{aligned} Y_s &= Z^T Y Z,\\ X_s &= Q X Q^T & (9) \end{aligned}$$

leads to the reduced Lyapunov equation

$$A_s^T X_s E_s + E_s^T X_s A_s = -Y_s, \tag{10}$$

which is equivalent to (1).
Let A_s, E_s, Y_s, and X_s be partitioned into p by p blocks with respect to the subdiagonal structure of the upper quasi-triangular matrix A_s. Especially, the main diagonal blocks are 1×1 or 2×2 matrices according to real eigenvalues or conjugate complex eigenvalue pairs of the pencil $A_s - \lambda E_s$, respectively.

$$A_s = \begin{pmatrix} A_{11} & \cdots & A_{1p} \\ & \ddots & \vdots \\ O & & A_{pp} \end{pmatrix}, \quad E_s = \begin{pmatrix} E_{11} & \cdots & E_{1p} \\ & \ddots & \vdots \\ O & & E_{pp} \end{pmatrix},$$

$$Y_s = \begin{pmatrix} Y_{11} & \cdots & Y_{1p} \\ \vdots & \ddots & \vdots \\ Y_{p1} & \cdots & Y_{pp} \end{pmatrix}, \quad X_s = \begin{pmatrix} X_{11} & \cdots & X_{1p} \\ \vdots & \ddots & \vdots \\ X_{p1} & \cdots & E_{pp} \end{pmatrix}.$$

Now (10) can be solved by a block forward substitution, which is more complicated to derive than that utilized in the Bartels–Stewart method for the standard Lyapunov equation. Since X_s is symmetric, only $p(p+1)/2$ Sylvester equations

$$A_{kk}^T X_{kl} E_{ll} + E_{kk}^T X_{kl} A_{ll} = -\hat{Y}_{kl} \tag{11}$$

of order at most 2×2 with right hand side matrices

$$\hat{Y}_{kl} = Y_{kl} + \sum_{\substack{i=1,j=1\\(i,j)\neq(k,l)}}^{k,l} \left(A_{ik}^T X_{ij} E_{jl} + E_{ik}^T X_{ij} A_{jl} \right) \tag{12}$$

need to be solved. According to (4) the solution X_{ij} is found by solving the corresponding system of linear equations

$$\left(E_{ll}^T \otimes A_{kk}^T + A_{ll}^T \otimes E_{kk}^T \right) \operatorname{vec}(X_{kl}) = -\operatorname{vec}(\hat{Y}_{kl}).$$

We determine the blocks in the upper triangle of X_s in a row–wise order, i.e., we compute successively X_{11}, ..., X_{1p}, X_{22}, ..., X_{2p}, ..., X_{pp}. Computing the matrices $\hat{Y}_{kl}$ by (12) would result in an overall complexity $O(n^4)$. This can be avoided by a somewhat complicated updating technique based upon the following expansion of equation (12)

$$\begin{aligned} \hat{Y}_{kl} &= Y_{kl} + \sum_{i=1}^{k} \left(A_{ik}^T \left(\sum_{j=1}^{l-1} X_{ij} E_{jl} \right) + E_{ik}^T \left(\sum_{j=1}^{l-1} X_{ij} A_{jl} \right) \right) \\ &\quad + \sum_{i=1}^{k-1} \left(A_{ik}^T X_{il} E_{ll} + E_{ik}^T X_{il} A_{ll} \right). \end{aligned}$$

We determine $\hat{Y}_{kl}$ in $2k-1$ sweeps

$$\begin{aligned}
Y_{kl}^{(0)} &= Y_{kl}, \\
Y_{kl}^{(2i-1)} &= Y_{kl}^{(2i-2)} + A_{ik}^T \left(\sum_{j=1}^{l-1} X_{ij} E_{jl} \right) + E_{ik}^T \left(\sum_{j=1}^{l-1} X_{ij} A_{jl} \right), \quad i = 1, \ldots, k, \\
Y_{kl}^{(2i)} &= Y_{kl}^{(2i-1)} + A_{ik}^T X_{il} E_{ll} + E_{ik}^T X_{il} A_{ll}, \quad i = 1, \ldots, k-1, \\
\hat{Y}_{kl} &= Y_{kl}^{(2k-1)}.
\end{aligned}$$

Our implementation of the algorithm computes $Y_{kl}^{(2i-1)}$ $(k \geq i)$ right before solving (11) for X_{il}. After X_{il} is known, $Y_{kl}^{(2i)}$ $(k > i)$ can be computed. What decreases the complexity to $O(n^3)$ is that $Y_{i,l}^{(2i-1)}$, ..., $Y_{l,l}^{(2i-1)}$ (the elements below the main diagonal are not of interest) can be updated simultaneously, where we have to compute the terms $\sum_{j=1}^{l-1} X_{ij} E_{jl}$ and $\sum_{j=1}^{l-1} X_{ij} A_{jl}$ only once. Note that the blocks of the strict lower triangle of X_s appearing in each of both sums are given by symmetry. Moreover, it is apparent that the updating technique described above can be realized by operating on the array Y without the need for further workspace. Finally, the solution matrix X is obtained from X_s by (9).

Lyapunov equations whose coefficient matrices A and E are replaced by its transposed matrices can be solved in a similar fashion by block *backward* substitution.

2.2 Estimation of the Separation and the Condition Number

The software for the Bartels–Stewart method provides optional estimates for both the separation and the condition number of the Lyapunov operator. Especially the latter is important for estimating the accuracy of the computed solution.

In the continuous–time case the separation is defined as

$$\mathrm{sep}_c = \mathrm{sep}_c(A, E) = \min_{\|X\|_F = 1} \left\| A^T X E + E^T X A \right\|_F .$$

Due to the orthogonal invariance of the Frobenius norm, this quantity is not altered by the transformations (7) and (8). Therefore, the estimate can be obtained from the reduced equation (10) which lowers the computational cost significantly. According to (4),

$$\mathrm{sep}_c = \frac{1}{\left\| K_c^{-1} \right\|_2},$$

if $K_c = E_s^T \otimes A_s^T + A_s^T \otimes E_s^T$ is invertible. The quantity $\left\| K_c^{-1} \right\|$ is estimated by an algorithm due to Higham [11] based on Hager's method [9]. Actually, this method yields an estimate for the 1–norm. This is a quite good approximation of $\left\| K_c^{-1} \right\|_2$, since it deviates from $\left\| K_c^{-1} \right\|_1$ by no more than a factor n. The algorithm, which is available as routine DLACON in LAPACK [1], requires the solution of a few (say 4 or 5) generalized Lyapunov equations $A_s^T X_s E_s + E_s^T X_s A_s = -Y_s$ or $A_s X_s E_s^T + E_s X_s A_s^T = -Y_s$.

Finally, an estimate for the condition number of the generalized Lyapunov operator, which is represented by the corresponding matrix K_c, is provided by

$$\mathrm{cond}(K_c) \approx \frac{2 \|A_s\|_F \|E_s\|_F}{\mathrm{sep}_c}.$$

Again, note that $\|A\|_F = \|A_s\|_F$ and $\|E\|_F = \|A_s\|_F$.

The separation of the discrete–time Lyapunov operator is

$$\mathrm{sep}_d = \mathrm{sep}_d(A, E) = \min_{\|X\|_F = 1} \left\| A^T X A - E^T X E \right\|_F = \frac{1}{\left\| K_d^{-1} \right\|_2}$$

with $K_d = A_s^T \otimes A_s^T - E_s^T \otimes E_s^T$. After approximating sep_d by the reciprocal estimate for $\left\| K_d^{-1} \right\|_1$ an estimate for the condition number is gained from

$$\mathrm{cond}(K_d) \approx \frac{\|A_s\|_F^2 + \|E_s\|_F^2}{\mathrm{sep}_d}.$$

2.3 A Generalization of Hammarling's Method

The method due to Hammarling is an alternative to the Bartels–Stewart method in case the Lyapunov equation to be solved is stable and its right hand side is semidefinite. In [10] Hammarling suggested that his method can be extended to generalized Lyapunov equations. We will present such a generalization in the sequel.

We assume the pencil $A - \lambda E$ to be stable, i.e., its eigenvalues must lie in the open left half plane in the continuous–time case or inside the unit circle in the discrete–time case. If these conditions are met, the solutions X of the GCLE

$$A^T X E + E^T X A = -B^T B \tag{13}$$

and the GDLE

$$A^T X A - E^T X E = -B^T B \tag{14}$$

are positive semidefinite (see Theorem 2). In general, B is a real $m \times n$ matrix, while A, E, and X are real square matrices of order n. The Cholesky factor U of the solution $X = U^T U$ can be found without first computing X and without forming the matrix product $B^T B$. We focus on the continuous–time equation (13) and give a note about the differences in the discrete–time case.

Similar to the Bartels–Stewart method, the algorithm consists of three parts. First the equation is transformed to a reduced form by means of the orthogonal matrices resulting from the generalized Schur factorization. After solving the reduced equation, the solution is retrieved by back transformation.

Applying the QZ–algorithm to the pencil $A - \lambda E$ yields orthogonal matrices Q and Z such that $A_s = Q^T A Z$ is upper quasitriangular and $E_s = Q^T E Z$ is upper triangular. This leads to the reduced equation

$$A_s^T U_s^T U_s E_s + E_s^T U_s^T U_s A_s = -B_s^T B_s, \tag{15}$$

where the $n \times n$ upper triangular matrix B_s on the right hand side is formed as follows. If $m \geq n$, the matrix B_s is obtained from the rectangular QR–factorization

$$BZ = Q_B \begin{pmatrix} B_s \\ 0 \end{pmatrix},$$

where Q_B is an orthogonal matrix of order m. Otherwise, we partition BZ as

$$BZ = \begin{pmatrix} \hat{B}_1 & \hat{B}_2 \end{pmatrix},$$

where $\hat{B}_1$ is an $m \times m$ matrix with the QR–factorization

$$\hat{B}_1 = Q_B \hat{B}_3.$$

Thus, B_s is given by

$$B_s = \begin{pmatrix} \hat{B}_3 & Q_B^T \hat{B}_2 \\ 0 & 0 \end{pmatrix}.$$

To solve the reduced equation (15), we partition the involved matrices as

$$A_s = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}, \qquad E_s = \begin{pmatrix} E_{11} & E_{12} \\ 0 & E_{22} \end{pmatrix},$$

$$B_s = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}, \qquad U_s = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}.$$

The upper left blocks are $p \times p$ matrices ($p = 1, 2$), where $p = 2$ iff the pencil $A_{11} - \lambda E_{11}$ has a pair of complex conjugate eigenvalues.

Provided that U_{11} is nonsingular, the above partitioning leads to the following formulas

$$\begin{aligned}
A_{11}^T U_{11}^T U_{11} E_{11} + E_{11}^T U_{11}^T U_{11} A_{11} &= -B_{11}^T B_{11}, && (16)\\
A_{22}^T U_{12}^T + E_{22}^T U_{12}^T M_1 &= -B_{12}^T M_2 - A_{12}^T U_{11}^T - E_{12}^T U_{11}^T M_1, && (17)\\
A_{22}^T U_{22}^T U_{22} E_{22} + E_{22}^T U_{22}^T U_{22} A_{22} &= -B_{22}^T B_{22} - B_{12}^T B_{12} \\
&\quad -(A_{12}^T U_{11}^T + A_{22}^T U_{12}^T)(U_{11} E_{12} + U_{12} E_{22}) \\
&\quad -(E_{12}^T U_{11}^T + E_{22}^T U_{12}^T)(U_{11} A_{12} + U_{12} A_{22}) \\
&= -B_{22}^T B_{22} - y y^T && (18)
\end{aligned}$$

with the auxiliary matrices

$$\begin{aligned} M_1 &= U_{11} A_{11} E_{11}^{-1} U_{11}^{-1}, & (19)\\ M_2 &= B_{11} E_{11}^{-1} U_{11}^{-1}, & (20)\\ y &= B_{12}^T - (E_{12}^T U_{11}^T + E_{22}^T U_{12}^T) M_2^T. \end{aligned}$$

As mentioned in Section 1 the matrix E is nonsingular so that the inverse of E_{11} exists. Moreover, it can be proved that U_{11} is nonsingular as well if $B_{11} \neq 0$.

The above equations enable the entries of U_s to be determined recursively. The block U_{11} is gained from the stable Lyapunov equation (16). Afterwards, the matrices M_1 and M_2 are determined. Solving (16) for the case $p = 2$ is described later in an extra paragraph.

After computing U_{11}, M_1, and M_2, the generalized Sylvester equation (17) is solved for U_{12}^T. If U_{11} is nonsingular, the existence of a unique solution U_{12}^T is guaranteed by Theorem 1.

It remains to find the Cholesky factor $\tilde{B}_{22}$ of the right hand side matrix of equation (18)

$$B_{22}^T B_{22} + y y^T = \tilde{B}_{22}^T \tilde{B}_{22}.$$

The upper triangular matrix $\tilde{B}_{22}$ is cheaply obtained from the QR–factorization of

$$\begin{pmatrix} B_{22} \\ y^T \end{pmatrix} = Q_{\tilde{B}} \begin{pmatrix} \tilde{B}_{22} \\ 0 \end{pmatrix},$$

where $Q_{\tilde{B}}$ is an orthogonal matrix and 0 is the zero matrix which consists of p rows. The resulting equation

$$A_{22}^T U_{22}^T U_{22} E_{22} + E_{22}^T U_{22}^T U_{22} A_{22} = -B_{22}^T B_{22}$$

has the same structure as (15) but its size is lowered by p. Hence, all entries of U_s can be determined recursively.

After the reduced equation (15) is solved, we find the upper triangular solution matrix U of equation (13) by the QR–factorization of

$$U_s Q^T = Q_U U$$

with the orthogonal matrix Q_U. Of course, the latter need not be accumulated.

The next part of this section is addressed to the problems caused by the non–real eigenvalues of the pencil $A_{11} - \lambda E_{11}$ when $p = 2$. The procedure for solving the 2×2 Lyapunov equation (16) is similar to that described above for the equation of order n. For an explanation of the algorithm we make use of complex computation. Nevertheless, in our implementation complex operations are emulated by real ones. First the pencil $A_{11} - \lambda E_{11}$ is reduced to Schur form by means of unitary matrices $\hat{Q}$ and $\hat{Z}$ such that

$$\begin{aligned} \hat{Q}^H A_{11} \hat{Z} = \hat{A}_{11} &= \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}, \\ \hat{Q}^H E_{11} \hat{Z} = \hat{E}_{11} &= \begin{pmatrix} e_{11} & e_{12} \\ 0 & e_{22} \end{pmatrix}. \end{aligned}$$

Let $B_{11}\hat{Z} = \hat{Q}_B \hat{B}_{11}$ be the QR–factorization of $B_{11}\hat{Z}$ with an unitary matrix $\hat{Q}_B$, for which the entries on the main diagonal of the upper triangular matrix $\hat{B}_{11}$ are real and non–negative. Now the reduced form of (16) can be written as

$$\hat{A}_{11}^H \hat{U}_{11}^H \hat{U}_{11} \hat{E}_{11} + \hat{E}_{11}^H \hat{U}_{11}^H \hat{U}_{11} \hat{A}_{11} = -\hat{B}_{11}^H \hat{B}_{11}.$$

After partitioning $\hat{B}_{11}$ and $\hat{U}_{11}$ as follows

$$\hat{B}_{11} = \begin{pmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{pmatrix}, \qquad \hat{U}_{11} = \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix},$$

we find the entries of $\hat{U}_{11}$ from

$$\begin{aligned} \delta_1 &= \sqrt{-\bar{a}_{11} e_{11} - \bar{e}_{11} a_{11}}, \\ u_{11} &= \frac{b_{11}}{\delta_1}, \end{aligned}$$

$$\begin{aligned} u_{12} &= -\frac{b_{12}\delta_1 + (\bar{a}_{11}e_{12} + a_{12}\bar{e}_{11})u_{11}}{a_{22}\bar{e}_{11} + e_{22}\bar{a}_{11}}, \\ \delta_2 &= \sqrt{-\bar{a}_{22}e_{22} - \bar{e}_{22}a_{22}}, \\ y &= \bar{b}_{12} - \frac{\delta_1}{\bar{e}_{11}}(\bar{e}_{12}u_{11} + \bar{e}_{22}\bar{u}_{12}), \\ u_{22} &= \frac{1}{\delta_2}\sqrt{b_{22}^2 + |y|^2}. \end{aligned}$$

Finally, the upper triangular solution U_{11} of (16) is obtained by the QR–factorization

$$\hat{U}_{11}\hat{Q} = Q_U U_{11},$$

where Q_U is a unitary matrix.

It should be stressed that if $p = 2$ care is needed when computing the auxiliary matrices M_1 and M_2 since U_{11} may be ill conditioned. This is the case when the pair $(E_{11}^{-T}A_{11}^T, E_{11}^{-T}B_{11}^T)$ is near to an uncontrollable one. For a further discussion of this issue we refer to Section 6 in [10]. In our implementation we followed the procedure proposed there.

For the discrete–time equation (14) most of the algorithm is similar to that for the continuous–time equation. An essential difference is the solution of the reduced equation

$$A_s^T U_s^T U_s A_s - E_s^T U_s^T U_s E_s = -B_s^T B_s. \tag{21}$$

Here the recursion is based on the formulas

$$\begin{aligned} A_{11}^T U_{11}^T U_{11} A_{11} - E_{11}^T U_{11}^T U_{11} E_{11} &= -B_{11}^T B_{11} \\ A_{22}^T U_{12}^T M_1 - E_{22}^T U_{12}^T &= -B_{12}^T M_2 + E_{12}^T U_{11}^T - A_{12}^T U_{11}^T M_1 \\ A_{22}^T U_{22}^T U_{22} A_{22} - E_{22}^T U_{22}^T U_{22} E_{22} &= -B_{22}^T B_{22} - yy^T, \end{aligned} \tag{22}$$

where M_1 and M_2 are defined as in (19) and (20), respectively. The matrix y is given by

$$y = \begin{pmatrix} B_{12}^T & A_{12}^T U_{11}^T + A_{22}^T U_{12}^T \end{pmatrix} C,$$

where C is a matrix which fulfils

$$M_3 = I_{2p} - \begin{pmatrix} M_2 \\ M_1 \end{pmatrix} \cdot \begin{pmatrix} M_2 \\ M_1 \end{pmatrix}^T = CC^T.$$

From (19), (20), and (22) we obtain $M_2^T M_2 + M_1^T M_1 = I_p$ which leads to $M_3^2 = M_3$. Hence, the symmetric matrix M_3 is the orthogonal projector onto $\mathrm{span}((M_2^T \;\; M_1^T)^T)^\perp$. Thus, the $2p \times 2p$ matrix M_3 is actually positive semidefinite and has rank p. Consequently, the factor C is a $2p \times p$ matrix and the matrix y consists of only p columns.

3 Software Implementation

Our routines for solving the generalized Lyapunov equation possess some new features that should be mentioned here. Both Hammarling's method and the Bartels–Stewart method are implemented in a way that enables the transposed equations to be solved without transposing the involved matrices explicitly. Furthermore, our routines can benefit from Schur factorizations of the pencil $A - \lambda E$, which have been computed prior to calling the routine. To keep storage requirements low, the input right hand side and the output solution share the same array. This, of course, results in the loss of the right hand side matrix. An output parameter for scaling of the solution is provided to guard against overflow.

For the Bartels–Stewart method the symmetric right hand side matrix Y may be supplied as upper or lower triangle. In any case the full solution matrix is returned. Moreover, an optional estimation of the separation and the reciprocal condition number is provided. Of course, when solving the generalized Lyapunov equation via Hammarling's algorithm the condition estimator in the Bartels–Stewart routine can be utilized to detect ill–conditioning in the equation.

The number of flops required by the routines is given by the following table. It strongly depends on whether the generalized Schur factorization of the pencil $A - \lambda E$ is supplied (FACT = .TRUE.) or not, when calling one of both main routines. The flop estimate $66n^3$ for the QZ–algorithm, which delivers this factorization, is taken from [8]. We split up the flop count for the Bartels–Stewart method into the three possible cases, where the solution (JOB='X'), the separation (JOB='S'), or both quantities (JOB='B') are to be computed. Note that we count a single floating point arithmetic operation as one flop. The quantity c is an integer number of modest size (say 4 or 5).

Method		FACT=.TRUE.	FACT=.FALSE.
Bartels–Stewart	JOB='B'	$(26+8c)/3 \cdot n^3$	$(224+8c)/3 \cdot n^3$
	JOB='S'	$8c/3 \cdot n^3$	$(198+8c)/3 \cdot n^3$
	JOB='X'	$26/3 \cdot n^3$	$224/3 \cdot n^3$
Hammarling	$m \le n$	$(13n^3+6mn^2$ $+6m^2n-2m^3)/3$	$(211n^3+6mn^2$ $+6m^2n-2m^3)/3$
	$m > n$	$(11n^3+12mn^2)/3$	$(209n^3+12mn^2)/3$

Number of flops required by the routines.

Our implementation of the Bartels–Stewart algorithm is backward stable if the eigenvalues of the pencil $A - \lambda E$ are real. Otherwise, linear systems of order at most 4 are involved into the computation. These systems are solved by Gauss elimination with complete pivoting. The loss of stability in such eliminations is rarely encountered in practice. To our knowledge, there are no backward stability results for Hammarling's method for the standard Lyapunov equation.

The source code is implemented in FORTRAN 77 and it meets the programming standards of the Working Group on Software [15]. It makes use of BLAS–routines and routines available in LAPACK 2.0 [1]. Although BLAS–routines of level 3 are used the algorithms are basically of level 1 and 2. The type COMPLEX is not used. Complex computation is avoided or emulated by real operations.

The remainder of this section gives a brief survey of the subroutine organization. An interface specification of both main routines is enclosed in the appendix.

Bartels–Stewart method

DGLP Main routine. To be called by the user.

DGLPRC Solves the reduced continuous–time equation (10).

DGLPRD Solves the reduced discrete–time equation.

DGELUF LU–factorization of a square matrix with complete pivoting.

DGELUS Back substitution for linear systems whose coefficient matrix has been factorized by **DGELUF**. Provides scaling.

DMTRA Transposes a matrix.

DZTAZ Computes Z^TAZ or ZAZ^T for a symmetric matrix A.

Hammarling's method

DGLPHM Main routine. To be called by the user.

DGHRC Computes the Cholesky factor of the solution of the reduced continuous–time equation (15).

DGHRD Computes the Cholesky factor of the solution of the reduced discrete–time equation (21).

DGHNX2 Solves the matrix equation $A^TXC + E^TXD = Y$ or $AXC^T + EXD^T = Y$ for the $n \times m$ matrix X $(m = 1, 2)$. Provides scaling.

DGH2X2 Hammarling's method for the 2×2 Lyapunov equation in case the matrix pencil has complex conjugate eigenvalues. Provides scaling.

DCXGIV Computes parameters for the complex Givens rotation.

DGELUF LU–factorization of a square matrix with complete pivoting.

DGELUS Back substitution for linear systems whose coefficient matrix has been factorized by **DGELUF**. Provides scaling.

Note that the subroutines DGELUF and DGELUS are contained in LAPACK 2.1 under the names DGETC2 and DGESC2, respectively.

4 Numerical Experiments

In this section we demonstrate the performance of our software applied to two sample problems. Both depend on a scale parameter t which affects the condition number of the equation. We compare the results obtained by the Bartels–Stewart subroutine DGLP and the Hammarling subroutine DGLPHM with those of the established subroutines SYLGC and SYLGD [7]. The tests were carried out on a HP Apollo series 700 workstation under IEEE double precision and machine precision $\epsilon \approx 2.22 \cdot 10^{-16}$.

Example 1 [7]: The matrices A and E are defined as

$$\begin{aligned} A &= (2^{-t}-1)I_n + \operatorname{diag}(1,2,3,\ldots,n) + U_n^T \\ E &= I_n + 2^{-t}U_n \end{aligned}$$

in the continuous–time case and

$$\begin{aligned} A &= 2^{-t}I_n + \operatorname{diag}(1,2,3,\ldots,n) + U_n^T \\ E &= I_n + 2^{-t}U_n \end{aligned}$$

in the discrete–time case, where I_n is the $n \times n$ identity matrix and U_n is an $n \times n$ matrix with unit entries below the diagonal and all other entries zero. These systems become increasingly ill–conditioned as the parameter t increases. In all cases Y is defined as the $n \times n$ matrix that gives a true solution matrix X of all unit entries.

We generated the above problems for a medium size ($n = 100$) and various values of the parameter t. The following table shows the relative errors $\|\hat{X} - X\|_F / \|X\|_F$, where X is the known true solution and $\hat{X}$ is the computed solution affected by roundoff. Since the pencil $A - \lambda E$ is in general not stable, the Hammarling subroutine is not applied to this example.

	Cont.–time Eq.		Discr.–time Eq.	
t	DGLP	SYLGC	DGLP	SYLGD
0	7.478E–13	2.304E–12	1.267E–13	2.274E–13
10	4.042E–12	4.248E–12	1.304E–12	1.010E–11
20	1.113E–08	1.940E–09	2.172E–09	3.995E–09
30	9.136E–07	4.947E–06	7.732E–06	1.501E–06
40	1.460E–03	4.042E–03	7.613E–03	–

Example 1. $n = 100$. Relative error.

For $t = 40$ the subroutine SYLGD returned an error flag.

For problems of a smaller size ($n = 10$) we compare the estimates for the separation SEP and the reciprocal condition number RCOND with the 'true' values computed applying the singular value decomposition (SVD) to the corresponding Kronecker product matrix. Note that SYLGC and SYLGD do not return estimates for the separation.

	Cont.–time Eq.		Discr.–time Eq.	
t	DGLP	true	DGLP	true
0	3.685E–01	1.481E–01	2.492E+00	1.149E+00
10	1.952E–03	4.879E–04	3.909E–03	9.767E–04
20	1.907E–06	4.768E–07	3.815E–06	9.537E–07
30	1.863E–09	4.657E–10	3.725E–09	9.313E–10
40	1.818E–12	4.547E–13	3.637E–12	9.095E–13

Example 1. $n = 10$. Estimates for the separation.

	Cont.–time Eq.			Discr.–time Eq.		
t	DGLP	SYLGC	true	DGLP	SYLGD	true
0	1.198E–03	1.083E–03	3.813E–03	4.119E–03	7.104E–03	1.987E–02
10	1.699E–05	1.597E–05	4.537E–05	8.882E–06	2.852E–06	1.375E–05
20	1.660E–08	1.589E–08	4.441E–08	8.670E–09	2.817E–09	1.339E–08
30	1.621E–11	1.551E–11	4.337E–11	8.467E–12	2.751E–12	1.308E–11
40	1.582E–14	1.513E–14	4.231E–14	8.266E–15	2.689E–15	1.286E–14

Example 1. $n = 10$. Estimates for the reciprocal condition number.

Example 2: This example is generated in a way that enables setting the eigenvalues of the pencil $A - \lambda E$, which is advantageous for two reasons. In contrast to the previous example it is guaranteed that the pencil has both real and conjugate complex eigenvalues. On the other hand stable pencils can easily be constructed to include the Hammarling subroutine into the comparison. If $n = 3q$ we choose the $n \times n$ matrices A and E as

$$\begin{aligned} A &= V_n \operatorname{diag}(A_1, \ldots, A_q) W_n, \qquad A_i = \begin{pmatrix} s_i & 0 & 0 \\ 0 & t_i & t_i \\ 0 & -t_i & t_i \end{pmatrix} \\ E &= V_n W_n, \end{aligned}$$

where V_n and W_n are $n \times n$ matrices containing only zeroes and ones. The unit entries of V_n are on and below the anti–diagonal, those of W_n are on and below the diagonal. The parameters s_i and t_i determining the eigenvalues of the pencil are chosen as $s_i = t_i = t^i$ in the continuous–time case and $s_i = 1 - 1/t^i$, $t_i = -\sqrt{2} s_i / 2$ in the discrete–time case. The semidefinite right hand side is formed as the normal matrix

$$Y = B^T B, \qquad B = (1, 2, \ldots, n).$$

The drawback of this example is that the true solution is not known. Therefore, the accuracy of the computed solution $\hat{X}$ is measured by the relative residual defined as $\|A^T \hat{X} E + E^T \hat{X} A + Y\|_F / \|Y\|_F$ in the continuous–time case and $\|A^T \hat{X} A - E^T \hat{X} E + Y\|_F / \|Y\|_F$ in the discrete–time case. But this, of course, is a worse criterion compared to the relative error in case the equation is ill–conditioned. The following results were obtained for problems of size $n = 99$.

	Cont.–time Eq.			Discr.–time Eq.		
t	DGLP	DGLPHM	SYLGC	DGLP	DGLPHM	SYLGD
1.0	2.982E–13	6.564E–14	3.681E–14	1.716E–13	1.720E–13	5.755E–15
1.2	1.661E–13	1.028E–13	7.749E–14	1.850E–11	1.844E–11	4.412E–12
1.4	8.829E–12	3.285E–11	3.960E–12	2.857E–09	2.252E–09	9.921E–10
1.6	3.985E–10	4.047E–10	2.423E–10	3.328E–05	1.400E–07	4.732E–08
1.8	6.686E–09	5.559E–09	9.351E–09	–	–	–

Example 2. $n = 99$. Relative residual.

For $t = 1.8$ in the discrete–time case each of the subroutines returned an error flag.

Finally, the CPU–times obtained by means of the LAPACK–subroutine SECOND are compared for the parameters $n = 99$ and $t = 1.2$. They are split up into the times required for the three main stages of the computation, the transformation of the equation to the reduced form (T), the solution of the reduced equation (RE), and the back transformation of the solution (BT).

	Cont.–time Eq.			Discr.–time Eq.		
	DGLP	DGLPHM	SYLGC	DGLP	DGLPHM	SYLGD
T	9.71	9.33	58.61	10.30	9.63	67.64
RE	1.68	1.66	3.90	1.97	1.96	3.81
BT	0.42	0.24	4.83	0.45	0.23	4.84
Total	11.81	11.23	67.34	12.72	11.82	76.29

Example 2. $n = 99$. $t = 1.2$. CPU–times in seconds.

Concerning the accuracy of the computed solutions for both examples the involved subroutines do not differ significantly. The estimates of the condition number obtained by DGLP and SYLGC/SYLGD are of comparable size as well. A distinct merit of our subroutines are the shorter CPU–times. This is mainly caused by the fact that SYLGC/SYLGD make use of several older LINPACK [4] and EISPACK [5] subroutines. Especially, the QZ–subroutine DGEGS from LAPACK invoked by our subroutines is often faster than the modified EISPACK subroutines QZHESG, QZITG, and QZVALG utilized by SYLGC/SYLGD by a factor 6.

5 Acknowledgements

I wish to thank Andras Varga for providing his subroutines DMTRA and DZTAZ and the subroutines DGELUF and DGELUS written by Bo Kågström and Peter Poromaa.

A Subroutine Description

A.1 Bartels–Stewart Method

A.1.1 Purpose

To solve for X either the generalized continuous–time Lyapunov equation

$$\mathrm{op}(A)^T\, X\, \mathrm{op}(E) \;+\; \mathrm{op}(E)^T\, X\, \mathrm{op}(A) \;=\; -\mathit{scale}\, Y \tag{23}$$

or the generalized discrete–time Lyapunov equation

$$\mathrm{op}(A)^T\, X\, \mathrm{op}(A) \;-\; \mathrm{op}(E)^T\, X\, \mathrm{op}(E) \;=\; -\mathit{scale}\, Y \tag{24}$$

where $\mathrm{op}(M)$ is either M or M^T for $M = A, E$ and the right hand side Y is symmetric. A, E, Y, and the solution X are N by N matrices. *scale* is an output scale factor, set to avoid overflow in X.

An estimation of the separation and the reciprocal condition number of the Lyapunov operator is provided.

A.1.2 Specification

```
      SUBROUTINE DGLP( JOB, DISCR, FACT, TRANS, N, A, LDA, E, LDE,
     *                 UPPER, X, LDX, SCALE, Q, LDQ, Z, LDZ, IWORK,
     *                 RWORK, LRWORK, SEP, RCOND, IERR )
```

A.1.3 Argument List

Arguments In

N – INTEGER.

The order of the matrix A.

$N \geq 0$.

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

If FACT = .TRUE., then the leading N by N upper Hessenberg part of this array must contain the generalized Schur factor A_s of the matrix A. A_s must be an upper quasitriangular matrix. The elements below the upper Hessenberg part of the array A are not referenced.

If FACT = .FALSE., then the leading N by N part of this array must contain the matrix A.

Note: this array is overwritten if FACT = .FALSE..

LDA – INTEGER.

The leading dimension of the array A as declared in the calling program.

$\mathrm{LDA} \geq \mathrm{N}$.

E – DOUBLE PRECISION array of DIMENSION (LDE,N).

If FACT = .TRUE., then the leading N by N upper triangular part of this array must contain the generalized Schur factor E_s of the matrix E. The elements below the upper triangular part of the array E are not referenced.

If FACT = .FALSE., then the leading N by N part of this array must contain the coefficient matrix E of the equation.

Note: this array is overwritten if FACT = .FALSE..

LDE – INTEGER.

The leading dimension of the array E as declared in the calling program.

$\mathrm{LDE} \geq \mathrm{N}$.

X – DOUBLE PRECISION array of DIMENSION (LDX,N).

If JOB = 'B' or 'X', then the leading N by N part of this array must contain the right hand side matrix Y of the equation. On entry, either the lower or the upper triangular part of this array is referenced (see mode parameter UPPER).

If JOB = 'S', X is not referenced.

Note: this array is overwritten if JOB = 'B' or 'X'.

LDX – INTEGER.

The leading dimension of the array X as declared in the calling program.

LDX $\geq$ N.

Q – DOUBLE PRECISION array of DIMENSION (LDQ,N).

If FACT = .TRUE., then the leading N by N part of this array must contain the orthogonal matrix Q from the generalized Schur factorization.

If FACT = .FALSE., Q need not be set on entry.

LDQ – INTEGER.

The leading dimension of the array Q as declared in the calling program.

LDQ $\geq$ N.

Z – DOUBLE PRECISION array of DIMENSION (LDZ,N).

If FACT = .TRUE., then the leading N by N part of this array must contain the orthogonal matrix Z from the generalized Schur factorization.

If FACT = .FALSE., Z need not be set on entry.

LDZ – INTEGER.

The leading dimension of the array Z as declared in the calling program.

LDZ $\geq$ N.

Arguments Out

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N by N part of this array contains the generalized Schur factor A_s of the matrix A. (A_s is an upper quasitriangular matrix.)

E – DOUBLE PRECISION array of DIMENSION (LDE,N).

The leading N by N part of this array contains the generalized Schur factor E_s of the matrix E. (E_s is an upper triangular matrix.)

X – DOUBLE PRECISION array of DIMENSION (LDX,N).

If JOB = 'B' or 'X', then the leading N by N part of this array contains the solution matrix X of the equation.

If JOB = 'S', X has not been referenced.

SCALE – DOUBLE PRECISION.

The scale factor set to avoid overflow in X ($0 <$ SCALE ≤ 1).

Q – DOUBLE PRECISION array of DIMENSION (LDQ,N).

The leading N by N part of this array contains the orthogonal matrix Q from the generalized Schur factorization.

Z – DOUBLE PRECISION array of DIMENSION (LDZ,N).

The leading N by N part of this array contains the orthogonal matrix Z from the generalized Schur factorization.

SEP – DOUBLE PRECISION.

An estimate for the separation of the Lyapunov operator.

RCOND – DOUBLE PRECISION.

An estimate for the reciprocal condition number of the Lyapunov operator.

Work Space

IWORK – INTEGER array at least of DIMENSION (N**2).

If JOB = 'X', this array is not referenced.

RWORK – DOUBLE PRECISION array at least of DIMENSION (LRWORK).

On exit, if IERR = 0, RWORK(1) contains the optimal workspace.

LRWORK – INTEGER.

The dimension of the array RWORK. The following table contains the minimal work space requirements depending on the choice of JOB and FACT.

JOB	FACT	LRWORK
'X'	.TRUE.	N
'X'	.FALSE.	7N
'B', 'S'	.TRUE.	$2N^2$
'B', 'S'	.FALSE.	$\max(2N^2, 7N)$

Note: For good performance, LRWORK must generally be larger.

Tolerances

None.

Mode Parameters

JOB – CHARACTER*1.

Specifies if the solution is to be computed and if the separation (along with the reciprocal condition number) is to be estimated.

JOB = 'X', (Compute the solution only);
JOB = 'S', (Estimate the separation only);
JOB = 'B', (Compute the solution and estimate the separation).

DISCR – LOGICAL.

Specifies which type of equation is to be solved.

DISCR = .FALSE., (Continuous–time equation (23));
DISCR = .TRUE., (Discrete–time equation (24)).

FACT – LOGICAL.

Specifies whether the generalized real Schur factorization of the pencil $A - \lambda E$ is supplied on entry or not.

FACT = .FALSE., (The generalized real Schur factorization is not supplied);
FACT = .TRUE., (The generalized real Schur factorization is supplied).

TRANS – LOGICAL.

Specifies whether the transposed equation is to be solved or not.

TRANS = .FALSE., (op(A)=A, op(E)=E);
TRANS = .TRUE., (op(A)=A^T, op(E)=E^T).

UPPER – LOGICAL.

Specifies whether the lower or the upper triangle of the array X is referenced.

UPPER = .FALSE., (Only the lower triangle is referenced);
UPPER = .TRUE., (Only the upper triangle is referenced).

Warning Indicator

None.

Error Indicator

IERR – INTEGER.

Unless the routine detects an error (see next section), IERR contains 0 on exit.

A.1.4 Warnings and Errors detected by the Routine

IERR = 1:

On entry,	N < 0,
or	LDA $<$ N,
or	LDE $<$ N,
or	LDX $<$ N,
or	LDQ $<$ N,
or	LDZ $<$ N,
or	JOB $\notin$ {'B', 'b', 'S', 's', 'X', 'x'}.

IERR = 2:

LRWORK too small.

IERR = 3:

FACT = .TRUE. and the matrix contained in the upper Hessenberg part of the array A is not in upper quasitriangular form.

IERR = 4:

FACT = .FALSE. and the pencil $A - \lambda E$ cannot be reduced to generalized Schur form. LAPACK routine DGEGS has failed to converge.

IERR = 5:

DISCR = .TRUE. and the pencil $A - \lambda E$ has a pair of reciprocal eigenvalues. That is, $\lambda_i = 1/\lambda_j$ for some i and j, where λ_i and λ_j are eigenvalues of $A - \lambda E$. Hence, equation (24) is singular.

IERR = 6:

DISCR = .FALSE. and the pencil $A - \lambda E$ has a degenerate pair of eigenvalues. That is, $\lambda_i = -\lambda_j$ for some i and j, where λ_i and λ_j are eigenvalues of $A - \lambda E$. Hence, equation (23) is singular.

A.2 Hammarling's Method

A.2.1 Purpose

To compute the Cholesky factor U of the matrix $X = \mathrm{op}(U)^T \mathrm{op}(U)$, which is the solution of either the generalized c–stable continuous–time Lyapunov equation

$$\mathrm{op}(A)^T\, X\, \mathrm{op}(E) \;+\; \mathrm{op}(E)^T\, X\, \mathrm{op}(A) \;=\; -scale^2\, \mathrm{op}(B)^T\, \mathrm{op}(B) \tag{25}$$

or the generalized d–stable discrete–time Lyapunov equation

$$\mathrm{op}(A)^T\, X\, \mathrm{op}(A) \;-\; \mathrm{op}(E)^T\, X\, \mathrm{op}(E) \;=\; -scale^2\, \mathrm{op}(B)^T\, op(B) \tag{26}$$

without first finding X and without the need to form the matrix $\mathrm{op}(B)^T \mathrm{op}(B)$.

op(K) is either K or K^T for $K = A, B, E, U$. A and E are N by N matrices, op(B) is an M by N matrix. The resulting matrix U is an N by N upper triangular matrix with non–negative entries on its main diagonal. *scale* is an output scale factor set to avoid overflow in U.

A.2.2 Specification

```
      SUBROUTINE DGLPHM( DISCR, FACT, TRANS, N, M, A, LDA, E, LDE, B,
     *                   LDB, SCALE, Q, LDQ, Z, LDZ, RWORK, LRWORK,
     *                   IERR )
```

A.2.3 Argument List

Arguments In

N – INTEGER.

The order of the matrix A.

N $\geq$ 0.

M – INTEGER.

The number of rows in the matrix op(B).

M $\geq$ 1.

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

If FACT = .TRUE., then the leading N by N upper Hessenberg part of this array must contain the generalized Schur factor A_s of the matrix A. A_s must be an upper quasitriangular matrix. The elements below the upper Hessenberg part of the array A are not referenced.

If FACT = .FALSE., then the leading N by N part of this array must contain the matrix A.

Note: this array is overwritten if FACT = .FALSE..

LDA – INTEGER.

The leading dimension of the array A as declared in the calling program.

LDA $\geq$ N.

E – DOUBLE PRECISION array of DIMENSION (LDE,N).

If FACT = .TRUE., then the leading N by N upper triangular part of this array must contain the generalized Schur factor E_s of the matrix E. The elements below the upper triangular part of the array E are not referenced.

If FACT = .FALSE., then the leading N by N part of this array must contain the coefficient matrix E of the equation.

Note: this array is overwritten if FACT = .FALSE..

LDE – INTEGER.

The leading dimension of the array E as declared in the calling program.

LDE $\geq$ N.

B – DOUBLE PRECISION array of DIMENSION (LDB,N1).

If TRANS = .TRUE., the leading N by M part of this array must contain the matrix B and N1 $\geq$ MAX(M,N).

If TRANS = .FALSE., the leading M by N part of this array must contain the matrix B and N1 $\geq$ N.

Note: this array is overwritten.

LDB – INTEGER.

The leading dimension of the array B as declared in the calling program.

If TRANS = .TRUE., then LDB $\geq$ N.

If TRANS = .FALSE., then LDB $\geq$ MAX(M,N).

Q – DOUBLE PRECISION array of DIMENSION (LDQ,N).

If FACT = .TRUE., then the leading N by N part of this array must contain the orthogonal matrix Q from the generalized Schur factorization.

If FACT = .FALSE., Q need not be set on entry.

LDQ – INTEGER.

The leading dimension of the array Q as declared in the calling program.

LDQ $\geq$ N.

Z – DOUBLE PRECISION array of DIMENSION (LDZ,N).

If FACT = .TRUE., then the leading N by N part of this array must contain the orthogonal matrix Z from the generalized Schur factorization.

If FACT = .FALSE., Z need not be set on entry.

LDZ – INTEGER.

The leading dimension of the array Z as declared in the calling program.

LDZ $\geq$ N.

Arguments Out

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N by N part of this array contains the generalized Schur factor A_s of the matrix A. (A_s is an upper quasitriangular matrix.)

E – DOUBLE PRECISION array of DIMENSION (LDE,N).

The leading N by N part of this array contains the generalized Schur factor E_s of the matrix E. (E_s is an upper triangular matrix.)

B – DOUBLE PRECISION array of DIMENSION (LDB,N1).

The leading N by N part of this array contains the Cholesky factor U of the solution matrix X of the problem.

SCALE – DOUBLE PRECISION.

The scale factor set to avoid overflow in U (0 < SCALE $\leq$ 1).

Q – DOUBLE PRECISION array of DIMENSION (LDQ,N).

The leading N by N part of this array contains the orthogonal matrix Q from the generalized Schur factorization.

Z – DOUBLE PRECISION array of DIMENSION (LDZ,N).

The leading N by N part of this array contains the orthogonal matrix Z from the generalized Schur factorization.

Work Space

RWORK – DOUBLE PRECISION array at least of DIMENSION (LRWORK).

On exit, if IERR = 0, RWORK(1) contains the optimal workspace.

LRWORK – INTEGER.

The dimension of the array RWORK.

If FACT = .TRUE., then LRWORK $\geq$ MAX(6∗N-6,1).

If FACT = .FALSE., then LRWORK $\geq$ MAX(7∗N,1).

Note: For good performance, LRWORK must generally be larger.

Tolerances

None.

Mode Parameters

DISCR – LOGICAL.

Specifies which type of equation is to be solved.

DISCR = .FALSE., (Continuous–time equation (25));
DISCR = .TRUE., (Discrete–time equation (26)).

FACT – LOGICAL.

Specifies whether the generalized real Schur factorization of the pencil $A - \lambda E$ is supplied on entry or not.

FACT = .FALSE.,	(The generalized real Schur factorization is not supplied);
FACT = .TRUE.,	(The generalized real Schur factorization is supplied).

TRANS – LOGICAL.

Specifies whether the transposed equation is to be solved or not.

TRANS = .FALSE.,	(op(K)=K, $K = A, B, E, U$);
TRANS = .TRUE.,	(op(K)=K^T, $K = A, B, E, U$).

Warning Indicator

None.

Error Indicator

IERR – INTEGER.

Unless the routine detects an error (see next section), IERR contains 0 on exit.

A.2.4 Warnings and Errors detected by the Routine

IERR = 1:

On entry,	N < 0,
or	M < 1,
or	LDA < N,
or	LDE < N,
or	LDB < N,
or	LDQ < N,
or	LDZ < N,
or	(TRANS = .FALSE. and LDB < M).

IERR = 2:

LRWORK too small.

IERR = 3:

FACT = .TRUE. and the matrix contained in the upper Hessenberg part of the array A is not in upper quasitriangular form.

IERR = 4:

FACT = .FALSE. and the pencil $A - \lambda E$ cannot be reduced to generalized Schur form. LAPACK routine DGEGS has failed to converge.

IERR = 5:

FACT = .TRUE. and there is a 2 by 2 block on the main diagonal of the pencil $A_s - \lambda E_s$ with real eigenvalues.

IERR = 6:

DISCR = .FALSE. and the pencil $A - \lambda E$ is not c–stable.

IERR = 7:

DISCR = .TRUE. and the pencil $A - \lambda E$ is not d–stable.

IERR = 8:

DISCR = .TRUE. and the LAPACK routine DSYEVX has failed to converge during the solution of the reduced equation. This error is unlikely to occur.

B Example Programs

Two sample programs are enclosed to demonstrate the usage of the routines DGLP and DGLPHM.

B.1 Bartels–Stewart Method

Example

To find the solution matrix X, the separation, and the reciprocal condition number of the equation

$$A^T X E + E^T X A = -Y,$$

where

$$A = \begin{pmatrix} 3.0 & 1.0 & 1.0 \\ 1.0 & 3.0 & 0.0 \\ 1.0 & 0.0 & 2.0 \end{pmatrix}, \quad E = \begin{pmatrix} 1.0 & 3.0 & 0.0 \\ 3.0 & 2.0 & 1.0 \\ 1.0 & 0.0 & 1.0 \end{pmatrix}, \text{ and } Y = \begin{pmatrix} 64.0 & 73.0 & 28.0 \\ 73.0 & 70.0 & 25.0 \\ 28.0 & 25.0 & 18.0 \end{pmatrix}.$$

Program Text

```
*     DGLP EXAMPLE PROGRAM TEXT
*
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5, NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=20)
      INTEGER          LDA, LDE, LDQ, LDX, LDZ
      PARAMETER        (LDA=NMAX, LDE=NMAX, LDQ=NMAX, LDX=NMAX,
     +                 LDZ=NMAX)
      INTEGER          LIWORK, LRWORK
      PARAMETER        (LIWORK=NMAX**2, LRWORK=MAX(2*NMAX**2,7*NMAX))
*     .. Local Scalars ..
      CHARACTER        JOB
      DOUBLE PRECISION RCOND, SCALE, SEP
      INTEGER          I, IERR, J, N
      LOGICAL          DISCR, FACT, TRANS, UPPER
*     .. Local Arrays ..
      INTEGER          IWORK(LIWORK)
      DOUBLE PRECISION A(LDA,NMAX), E(LDE,NMAX), Q(LDQ,NMAX),
     +                 RWORK(LRWORK), X(LDX,NMAX), Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL         DGLP
*     .. Executable Statements ..
*
      WRITE (NOUT,FMT=99999)
*     Skip the heading in the data file and read the data.
      READ (NIN,FMT='()')
      READ (NIN,FMT=*) N, JOB, DISCR, FACT, TRANS, UPPER
      IF (N.LE.0 .OR. N.GT.NMAX) THEN
         WRITE (NOUT,FMT=99993) N
      ELSE
         READ (NIN,FMT=*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,FMT=*) ((E(I,J),J=1,N),I=1,N)
         IF (FACT) THEN
            READ (NIN,FMT=*) ((Q(I,J),J=1,N),I=1,N)
            READ (NIN,FMT=*) ((Z(I,J),J=1,N),I=1,N)
         END IF
         READ (NIN,FMT=*) ((X(I,J),J=1,N),I=1,N)
*        Find the solution matrix X and the scalars RCOND and SEP.
```

```
         CALL DGLP(JOB,DISCR,FACT,TRANS,N,A,LDA,E,LDE,UPPER,X,LDX,SCALE,
     +             Q,LDQ,Z,LDZ,IWORK,RWORK,LRWORK,SEP,RCOND,IERR)
*
         IF (IERR.NE.0) THEN
            WRITE (NOUT,FMT=99998) IERR
         ELSE
            IF (JOB.EQ.'B'.OR.JOB.EQ.'S') THEN
               WRITE (NOUT,FMT=99997) SEP
               WRITE (NOUT,FMT=99996) RCOND
            END IF
            IF (JOB.EQ.'B'.OR.JOB.EQ.'X') THEN
               WRITE (NOUT,FMT=99995) SCALE
               DO 20 I = 1, N
                  WRITE (NOUT,FMT=99994) (X(I,J),J=1,N)
   20          CONTINUE
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (' DGLP EXAMPLE PROGRAM RESULTS',/1X)
99998 FORMAT (' IERR on exit from DGLP = ',I2)
99997 FORMAT (' SEP =    ',F8.4)
99996 FORMAT (' RCOND = ',F8.4)
99995 FORMAT (' SCALE = ',F8.4,//' The solution matrix X is ')
99994 FORMAT (20(1X,F8.4))
99993 FORMAT (/' N is out of range.',/' N = ',I5)
      END
```

Program Data

```
 DGLP EXAMPLE PROGRAM DATA
  3       B       F       F       F       T
  3.0     1.0     1.0
  1.0     3.0     0.0
  1.0     0.0     2.0
  1.0     3.0     0.0
  3.0     2.0     1.0
  1.0     0.0     1.0
 64.0    73.0    28.0
  0.0    70.0    25.0
  0.0     0.0    18.0
```

Program Results

```
 DGLP EXAMPLE PROGRAM RESULTS

 SEP =      .2867
 RCOND =    .0055
 SCALE =   1.0000

 The solution matrix X is
  -2.0000  -1.0000    .0000
  -1.0000  -3.0000  -1.0000
    .0000  -1.0000  -3.0000
```

B.2 Hammarling's Method

Example

To find the Cholesky factor U of the solution $X = U^T U$ of the equation

$$A^T X E + E^T X A = -B^T B,$$

where

$$A = \begin{pmatrix} -1.0 & 3.0 & -4.0 \\ 0.0 & 5.0 & -2.0 \\ -4.0 & 4.0 & 1.0 \end{pmatrix}, \; E = \begin{pmatrix} 2.0 & 1.0 & 3.0 \\ 2.0 & 0.0 & 1.0 \\ 4.0 & 5.0 & 1.0 \end{pmatrix}, \text{ and } B = \begin{pmatrix} 2.0 & -1.0 & 7.0 \end{pmatrix}.$$

Program Text

```
*     DGLPHM EXAMPLE PROGRAM TEXT
*
*     .. Parameters ..
      INTEGER           NIN, NOUT
      PARAMETER         (NIN=5, NOUT=6)
      INTEGER           NMAX
      PARAMETER         (NMAX=20)
      INTEGER           LDA, LDB, LDE, LDQ, LDZ
      PARAMETER         (LDA=NMAX, LDB=NMAX, LDE=NMAX, LDQ=NMAX,
     +                  LDZ=NMAX)
      INTEGER           LRWORK
      PARAMETER         (LRWORK=MAX(7*NMAX,6*NMAX-6,1))
*     .. Local Scalars ..
      DOUBLE PRECISION  SCALE
      INTEGER           I, IERR, J, N, M
      LOGICAL           DISCR, FACT, TRANS
*     .. Local Arrays ..
      DOUBLE PRECISION  A(LDA,NMAX), B(LDB,NMAX), E(LDE,NMAX),
     +                  Q(LDQ,NMAX), RWORK(LRWORK), Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL          DGLPHM
*     .. Executable Statements ..
*
      WRITE (NOUT,FMT=99999)
*     Skip the heading in the data file and read the data.
      READ (NIN,FMT='()')
      READ (NIN,FMT=*) N, M, DISCR, FACT, TRANS
      IF (N.LT.0 .OR. N.GT.NMAX) THEN
         WRITE (NOUT,FMT=99995) N
      ELSEIF (M.LT.1 .OR. M.GT.NMAX) THEN
         WRITE (NOUT,FMT=99994) M
      ELSE
         READ (NIN,FMT=*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,FMT=*) ((E(I,J),J=1,N),I=1,N)
         IF (FACT) THEN
            READ (NIN,FMT=*) ((Q(I,J),J=1,N),I=1,N)
            READ (NIN,FMT=*) ((Z(I,J),J=1,N),I=1,N)
         END IF
         IF (TRANS) THEN
            READ (NIN,FMT=*) ((B(I,J),J=1,M),I=1,N)
         ELSE
            READ (NIN,FMT=*) ((B(I,J),J=1,N),I=1,M)
         END IF
*        Find the Cholesky factor U of the solution matrix.
         CALL DGLPHM(DISCR,FACT,TRANS,N,M,A,LDA,E,LDE,B,LDB,SCALE,Q,LDQ,
     +       Z,LDZ,RWORK,LRWORK,IERR)
*
         IF (IERR.NE.0) THEN
            WRITE (NOUT,FMT=99998) IERR
         ELSE
            WRITE (NOUT,FMT=99997) SCALE
            DO 20 I = 1, N
               WRITE (NOUT,FMT=99996) (B(I,J),J=1,N)
   20       CONTINUE
         END IF
      END IF
      STOP
*
99999 FORMAT (' DGLPHM EXAMPLE PROGRAM RESULTS',/1X)
99998 FORMAT (' IERR on exit from DGLPHM = ',I2)
99997 FORMAT (' SCALE = ',F8.4,//' The Cholesky factor U of the solution
     + matrix is')
99996 FORMAT (20(1X,F8.4))
99995 FORMAT (/' N is out of range.',/' N = ',I5)
99994 FORMAT (/' M is out of range.',/' M = ',I5)
      END
```

Program Data

```
 DGLPHM EXAMPLE PROGRAM DATA
   3      1      F      F      F
  -1.0    3.0   -4.0
   0.0    5.0   -2.0
  -4.0    4.0    1.0
   2.0    1.0    3.0
   2.0    0.0    1.0
   4.0    5.0    1.0
   2.0   -1.0    7.0
```

Program Results

```
 DGLPHM EXAMPLE PROGRAM RESULTS

 SCALE =   1.0000

 The Cholesky factor U of the solution matrix is
   1.6003   -.4418   -.1523
    .0000    .6795   -.2499
    .0000    .0000    .2041
```

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson, *LAPACK User's Guide*, Philadelphia, PA, 1992.

[2] R.H. Bartels, G.W. Stewart, *Solution of the Equation $AX + XB = C$*, Comm. A.C.M., 15:820–826, 1972.

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

[4] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart, *LINPACK User's Guide*, Philadelphia, PA, 1979.

[5] B.S. Garbow, J.M. Boyle, J.J. Dongarra, C.B. Moler, *Matrix Eigensystem Routines – EISPACK Guide Extension*, Lecture Notes in Computer Science, Vol.51, Springer–Verlag, New York, 1977.

[6] J.D. Gardiner, A.J. Laub, J.J. Amato, C.B. Moler, *Solution of the Sylvester Matrix Equation $AXB^T + CXD^T = E$*, A.C.M. Trans. Math. Soft., 18:223–231, 1992.

[7] J.D. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, C.B. Moler, *A FORTRAN–77 Software Package for Solving the Sylvester Matrix Equation $AXB^T + CXD^T = E$*, A.C.M. Trans. Math. Soft., 18:232–238, 1992.

[8] G.H. Golub, C.F. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, Second Edition, 1989.

[9] W.W. Hager, *Condition Estimates*, SIAM J. Sci. Stat. Comput., 5:311–316, 1984.

[10] S.J. Hammarling, *Numerical Solution of the Stable, Non–negative Definite Lyapunov Equation*, IMA J. Num. Anal., 2:303–323, 1982.

[11] N.J. Higham, *FORTRAN Codes for Estimating the One–Norm of a Real or Complex Matrix, with Applications to Condition Estimation*, A.C.M. Trans. Math. Soft., 14:381-396, 1988.

[12] B. Kågström, L. Westin, *Generalized Schur Methods with Condition Estimators for Solving the Generalized Sylvester Equation*, IEEE Trans. Aut. Control, 34:745-751, 1989.

[13] P. Lancaster, M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd edition, 1985.

[14] J. Snyders, M. Zakai, *On Non–negative Solutions of the Equation $AD + DA^T = -C$*, SIAM J. Appl. Math., 18:704-714, 1970.

[15] Working Group on Software (WGS), *Implementation and Documentation Standards*, WGS–Report 90–1, Eindhoven/Oxford, 1990.