On Discrete $\mathcal{H}_\infty$ Loop Shaping Design Procedure Routines [1]

Da-Wei Gu[2], Petko Hr. Petkov[3] and Mihail M. Konstantinov[4]

November 2000

[1]This paper presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET (contract number BRRT-CT97-5040) and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN2000-6.ps.Z*
[2]Control Systems Research, Department of Engineering, University of Leicester, Leicester LE1 7RH, U.K.
[3]Department of Automatics, Technical University of Sofia, 1756 Sofia, Bulgaria
[4]University of Architecture & Civil Engineering, 1 Hr. Smirnenski Blv., 1421 Sofia, Bulgaria

Abstract

This report briefly introduces the $\mathcal{H}_\infty$ loop shaping design procedure (LSDP) in the discrete-time case as well as its implementation in the software package *SLICOT*. Solution formulae are presented with the exposure of a relationship between the solutions to the three discrete-time, algebraic Riccati equations (DARE) required in the construction of an LSDP controller. These SLICOT routines also produce estimates of the condition numbers of the DARE solutions, which reveals the accuracy and reliability of the computational results. The developed routines are tested in a design example, and are included as appendices.

Key Words: Robust control systems design, Discrete-time $\mathcal{H}_\infty$ loop shaping design procedure, Normalized coprime factor perturbation, SLICOT routines

1 Introduction

This report complements the Niconet report No. 1999-15 [1] as the $\mathcal{H}_\infty$ loop shaping design procedure (LSDP) in the discrete-time case is discussed here. Interested readers are referred to [1], and the references quoted therein, for introductions on and basic structures of the LSDP as well as all the necessary mathematical ingredients. The present report will concentrate on the formulae in the discrete-time case. Furthermore, a theorem will be introduced which reveals a relation between the solutions to the three discrete-time, algebraic Riccati equations appearing in the solution procedure. This results implies that only two such equations need be solved instead of three, which obviously is of great advantages in terms of computational effort and time in real design and consequently increases the computational reliability of the resultant controller designed.

The report will also introduce in detail the routines developed in the SLICOT package to implement the $\mathcal{H}_\infty$ loop shaping design procedure in the discrete-time case and will illustrate the usage of such routines in a design example.

2 Normalized coprime factorization of discrete-time plant

Let $G(z)$ be a minimal realization, discrete-time model of a plant,

$$\begin{aligned} G(z) &= D + C(zI - A)^{-1}B \qquad (2.1)\\ &\stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \end{aligned}$$

with $A : n \times n$, $B : n \times m$, $C : p \times n$, and $D : p \times m$.

Matrices $(\tilde{M}(z),\ \tilde{N}(z)) \in \mathcal{H}_\infty^+$, where $\mathcal{H}_\infty^+$ denotes the space of functions with all poles in the open unit disc of the complex plane, constitute a left coprime factorization of $G(z)$ if and only if

(i) $\tilde{M}$ is square, and $\det(\tilde{M}) \neq 0$.

(ii) the plant model is given by

$$G = \tilde{M}^{-1}\tilde{N} \qquad (2.2)$$

(iii) There exists $(\tilde{V},\ \tilde{U}) \in \mathcal{H}_\infty^+$ such that

$$\tilde{M}\tilde{V} + \tilde{N}\tilde{U} = I_p \qquad (2.3)$$

A left coprime factorization of G as defined in (2.2) is *normalized* if and only if

$$\tilde{N}(z)\tilde{N}^T(\frac{1}{z}) + \tilde{M}(z)\tilde{M}^T(\frac{1}{z}) = I_p. \qquad (2.4)$$

The concept of right coprime factorization and normalized right coprime factorization can be introduced dually. However, the work presented in the report will follow the (normalized) left coprime factorization, although all results concerning the (normalized) right coprime factorization can be derived similarly.

State-space constructions for the normalized coprime factorizations can be obtained in terms of the solutions to the following two discrete algebraic Riccati equations (DAREs),

$$\Phi^T P \Phi - P - \Phi^T P B Z_1 {Z_1}^T B^T P \Phi + C^T {R_1}^{-1} C = 0 \tag{2.5}$$

and

$$\Phi Q \Phi^T - Q - \Phi Q C^T Z_2^T Z_2 C Q \Phi^T + B {R_2}^{-1} B^T = 0 \tag{2.6}$$

where $R_1 = I_p + DD^T$, $R_2 = I_m + D^T D$, $\Phi = A - B{R_2}^{-1} D^T C$, $Z_1 {Z_1}^T = (R_2 + B^T P B)^{-1}$, ${Z_2}^T Z_2 = (R_1 + CQC^T)^{-1}$. And, $P \geq 0$ and $Q \geq 0$ are the unique stabilizing solutions, respectively.

Without loss of generality, we may assume that both Z_1 and Z_2 are square matrices, and $Z_1 = {Z_1}^T$, $Z_2 = {Z_2}^T$.

Further, define $H = -(AQC^T + BD^T){Z_2}^T Z_2$, and $F = -Z_1 {Z_1}^T (B^T P A + D^T C)$ then

$$\left[\begin{array}{cc} \tilde{N} & \tilde{M} \end{array} \right] \stackrel{s}{=} \left[\begin{array}{c|cc} A + HC & B + HD & H \\ \hline Z_2 C & Z_2 D & Z_2 \end{array} \right] \tag{2.7}$$

and

$$\left[\begin{array}{c} N \\ M \end{array} \right] \stackrel{s}{=} \left[\begin{array}{c|c} A + BF & BZ_1 \\ \hline C + DF & DZ_1 \\ F & Z_1 \end{array} \right] \tag{2.8}$$

are the normalizaed left, and right, coprime factorizations of G, correspondingly.

3 Robust controller formulae

Same as in the continuous-time case, the discrete-time $\mathcal{H}_\infty$ loop shaping design procedure is based on the construction of a robust stabilizing controller against the perturbations on the coprime factors, as depicted in Figure 1.

In the practical design using the $\mathcal{H}_\infty$ LSDP, the $(\tilde{M}, \tilde{N})$ in Figure 1 is the normalized left coprime facorization of an augmented system, the normal model with pre- and/or post- loop shaping weighting functions.

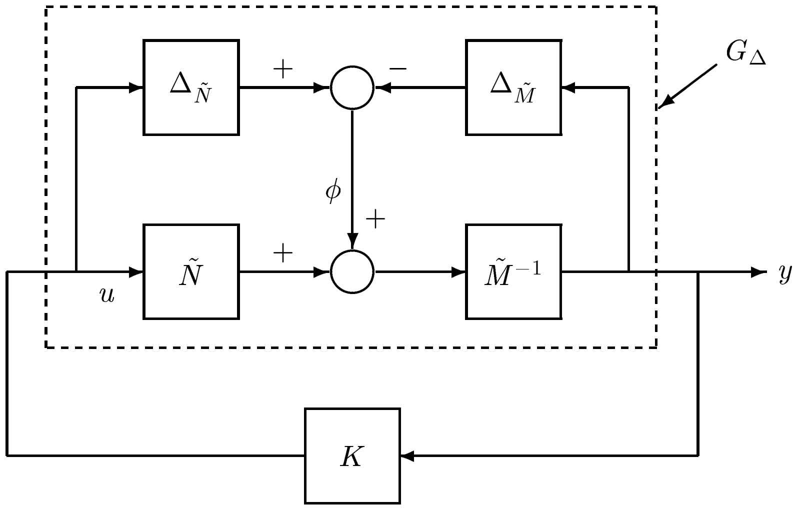

Figure 1: Robust stabilization with regard to coprime factor uncertainty.

To maximize the robust stability margin of the closed-loop system given in Fig. 1, one must minimize

$$\gamma := \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} \tilde{M}^{-1} \right\|_\infty$$

Thus, the lowest achievable value of γ for all stabilizing controllers K is

$$\gamma_o = \inf_{K \ stabilizing} \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} \tilde{M}^{-1} \right\|_\infty \tag{3.1}$$

and is given in by

$$\gamma_o = (1 + \lambda_{max}(QP))^{1/2} \tag{3.2}$$

where Q and P are the solutions to (2.6) and (2.5), respectively.

For a given $\gamma > \gamma_o$, a sub-optimal $\mathcal{H}_\infty$ LSDP controller K is that K internally stabilizes the nominal system in Fig. 1 and achieves

$$\left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} \tilde{M}^{-1} \right\|_\infty < \gamma$$

The synthesis of a sub-optimal $\mathcal{H}_\infty$ LSDP controller can be recast as a standard $\mathcal{H}_\infty$ sub-optimal control problem, which has been discussed in the Niconet report 1999-5 [2]. The generalized (interconnected) system in this case is

$$
\begin{aligned}
P(z) &= \left[\begin{array}{cc} P_{11}(z) & P_{12}(z) \\ P_{21}(z) & P_{22}(z) \end{array}\right] \\
&= \left[\begin{array}{c|c} 0 & I_m \\ \tilde{M}^{-1} & G \\ \hline \tilde{M}^{-1} & G \end{array}\right] \\
&= \left[\begin{array}{c|cc} A & -HZ_2{}^{-1} & B \\ \hline 0 & 0 & I_m \\ C & Z_2{}^{-1} & D \\ C & Z_2{}^{-1} & D \end{array}\right]
\end{aligned} \tag{3.3}
$$

Following the solution procedure given in [2], one more DARE need be solved in order to compute the required controller. In general $\mathcal{H}_\infty$ sub-optimal problems, two algebraic Riccati equations are to be solved. Here, however, due to the structure of $P(z)$ in (3.3), it can be shown that the solution to one of the DARE is always zero. The third DARE is the following

$$
A^T X_\infty A - X_\infty - \tilde{F}^T (R + \left[\begin{array}{c} -Z_2{}^{-1} H^T \\ R_2{}^{-1/2} B^T \end{array}\right] X_\infty \left[\begin{array}{cc} -HZ_2{}^{-1} & BR_2{}^{-1/2} \end{array}\right]) \tilde{F} + C^T C = 0 \tag{3.4}
$$

where

$$
\tilde{F} = -(R + \left[\begin{array}{c} -Z_2{}^{-1} H^T \\ R_2{}^{-1/2} B^T \end{array}\right] X_\infty \left[\begin{array}{cc} -HZ_2{}^{-1} & BR_2{}^{-1/2} \end{array}\right])^{-1} (\left[\begin{array}{c} -Z_2{}^{-1} C \\ D^T R_1{}^{-1/2} C \end{array}\right] + \left[\begin{array}{c} -Z_2{}^{-1} H^T \\ R_2{}^{-1/2} B^T \end{array}\right] X_\infty A)
$$

and

$$
R = \left[\begin{array}{cc} Z_2{}^{-2} - \gamma^2 I_p & Z_2{}^{-1} R_1{}^{-1/2} D \\ D^T R_1{}^{-1/2} Z_2{}^{-1} & I_m \end{array}\right]
$$

Further, by defining $\tilde{F} = \left[\begin{array}{c} F_1 \\ F_2 \end{array}\right]$, where, $F_1 : p \times n$ and $F_2 : m \times n$, the sub-optimal $\mathcal{H}_\infty$ DLSDP controller K can be constructed as

$$
K(z) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right]
$$

where

$$\begin{aligned}
A_K &= \hat{A}_K - \hat{B}_K D(I + \hat{D}_K D)^{-1}\hat{C}_K \\
B_K &= \hat{B}_K(I + D\hat{D}_K)^{-1} \\
C_K &= (I + \hat{D}_K D)^{-1}\hat{C}_K \\
D_K &= \hat{D}_K(I + D\hat{D}_K)^{-1}
\end{aligned} \tag{3.5}$$

with

$$\begin{aligned}
\hat{D}_K &= -(R_2 + B^T X_\infty B)^{-1}(D^T - B^T X_\infty H) \\
\hat{B}_K &= -H + B\hat{D}_K \\
\hat{C}_K &= R_2^{-1/2} F_2 - \hat{D}_K(C + Z_2{}^{-1}F_1) \\
\hat{A}_K &= A + HC + B\hat{C}_K
\end{aligned} \tag{3.6}$$

4 The strictly proper case

It may be appropriate to say that most plants considered in the practical, discrete-time control systems design are strictly proper, i.e. $D = 0$. That is not only because that most plants in industrial studies are strictly proper, as in the continuous-time case, but also because the $\mathcal{H}_\infty$ controllers designed tend to be proper due to the "flatness" of the optimality sought in the synthesis. Hence, when the plant is just proper, it is possible to encounter the problem of algebraic loop in the implementation of the resultant controller.

When the plant under consideration is strictly proper, all the computations and formulae described above will be significantly simpler. The two DAREs (2.5) and (2.6) become

$$A^T PA - P - A^T PBZ_1 Z_1{}^T B^T PA + C^T C = 0 \tag{4.1}$$

and

$$AQA^T - Q - AQC^T Z_2^T Z_2 CQA^T + BB^T = 0 \tag{4.2}$$

where $Z_1 Z_1{}^T = (I_m + B^T PB)^{-1}$, $Z_2{}^T Z_2 = (I_p + CQC^T)^{-1}$.

The third DARE (3.4) is now the following

$$A^T X_\infty A - X_\infty - \tilde{F}^T (R + \begin{bmatrix} -Z_2{}^{-1}H^T \\ B^T \end{bmatrix} X_\infty \begin{bmatrix} -HZ_2{}^{-1} & B \end{bmatrix})\tilde{F} + C^T C = 0 \tag{4.3}$$

where

$$\tilde{F} = -(R + \begin{bmatrix} -Z_2^{-1}H^T \\ B^T \end{bmatrix} X_\infty \begin{bmatrix} -HZ_2^{-1} & B \end{bmatrix})^{-1}(\begin{bmatrix} -Z_2^{-1}C \\ 0 \end{bmatrix} + \begin{bmatrix} -Z_2^{-1}H^T \\ B^T \end{bmatrix} X_\infty A)$$

and

$$\begin{aligned} R &= \begin{bmatrix} Z_2^{-2} - \gamma^2 I_p & 0 \\ 0 & I_m \end{bmatrix} \\ H &= -AQC^T Z_2^T Z_2. \end{aligned}$$

Further, by defining $\tilde{F} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$, where, $F_1 : p \times n$ and $F_2 : m \times n$, the sub-optimal $\mathcal{H}_\infty$ DLSDP controller K in the case of a strictly proper G can be constructed as

$$K(z) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right]$$

where

$$\begin{aligned} D_K &= (I_m + B^T X_\infty B)^{-1} B^T X_\infty H \\ B_K &= -H + BD_K \\ C_K &= F_2 - D_K(C + Z_2^{-1}F_1) \\ A_K &= A + HC + BC_K \end{aligned} \tag{4.4}$$

5 On the three DARE solutions

As discussed above, the discrete-time $\mathcal{H}_\infty$ LSDP sub-optimal controller formulae require the solutions to the three discrete-time algebraic Riccati equations, (2.5), (2.6) and (3.4), or (4.1), (4.2) and (4.3) in the strictly proper case. In this section, we will reveal that there is a relation between those three solutions, namely the solution X_∞ to the third DARE can be calculated directly from the first two solutions P and Q. This fact is important and useful, especially in the numerical implementation of the DLSDP routines.

We start with a general DARE, hence the notations are not related to those defined earlier in the paper,

$$F^T XF - X - F^T XG_1(G_2 + G_1^T XG_1)^{-1}G_1^T XF + C^T C = 0 \tag{5.1}$$

where $F,\ H,\ X \in \mathcal{R}^{n\times n}$, $G_1 \in \mathcal{R}^{n\times m}$, $G_2 \in \mathcal{R}^{m\times m}$, and $G_2 = G_2^T > 0$. We assume that $(F,\ G_1)$ is a stabilizable pair and that $(F,\ C)$ a detectable pair. We also define $G = G_1 G_2^{-1} G_1^T$.

It is well known that, see [3] or others, solutions to DARE (5.1) are closely linked with a matrix pencil pair $(M,\ L)$, where

$$\begin{aligned} M &= \begin{bmatrix} F & 0 \\ -H & I \end{bmatrix} \\ L &= \begin{bmatrix} I & G \\ 0 & F^T \end{bmatrix} \end{aligned} \tag{5.2}$$

It also can be shown that if there exist $n \times n$ matrices S, U_1 and U_2, with U_1 invertible, such that

$$M \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = L \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} S \tag{5.3}$$

then, $X = U_2 U_1^{-1}$ is a solution to (5.1). Further, the matrix $F - G_1(G_2 + G_1^T X G_1)^{-1} G_1^T X F$ shares the same spectrum as S. Hence, if S is stable, i.e. all the eigenvalues are within the open unit disc, $F - G_1(G_2 + G_1^T X G_1)^{-1} G_1^T X F$ is also stable. Such an X is non-negative definite and unique, and is called the stabilizing solution to (5.1).

Under the above assumptions on (5.1), it was shown in [3] that none of the generalized eigenvalues of the pair $(M,\ L)$ lies on the unit circle, and if $\lambda \neq 0$ is a generalized eigenvalue of the pair, then $1/\lambda$ is also a generalized eigenvalue of the same multiplicity. In other words, the stable spectrum, consisting of n generalized eigenvalues lying in the open unit disc, is unique. Therefore, if there exists another triple $(V_1,\ V_2,\ T)$ satisfying (5.3), with V_1 being invertible and T stable, then there must exist an invertible R such that $T = R^{-1} S R$. Consequently,

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} R^{-1} \tag{5.4}$$

The solution of course remains the same, since $X = V_2 V_1^{-1} = (U_2 R)(U_1 R)^{-1} = U_2 U_1^{-1}$.

In our present study, we can accordingly define the three matrix pencils as

$$\begin{aligned} M_P &= \begin{bmatrix} \Phi & 0 \\ -C^T R_1^{-1} C & I \end{bmatrix} \\ L_P &= \begin{bmatrix} I & B R_2^{-1} B^T \\ 0 & \Phi^T \end{bmatrix} \end{aligned} \tag{5.5}$$

$$
M_Q = \begin{bmatrix} \Phi^T & 0 \\ -BR_2^{-1}B^T & I \end{bmatrix} \tag{5.6}
$$

$$
L_Q = \begin{bmatrix} I & C^T R_1^{-1} C \\ 0 & \Phi \end{bmatrix}
$$

$$
M_X = \begin{bmatrix} A - \begin{bmatrix} -HZ_2^{-1} & BR_2^{-1/2} \end{bmatrix} R^{-1} \begin{bmatrix} Z_2^{-1}C \\ D^T R_1^{-1/2} C \end{bmatrix} & 0 \\ -C^T C + \begin{bmatrix} C^T Z_2^{-1} & C^T R_1^{-1/2} D \end{bmatrix} R^{-1} \begin{bmatrix} Z_2^{-1}C \\ D^T R_1^{-1/2} C \end{bmatrix} & I \end{bmatrix} \tag{5.7}
$$

$$
L_X = \begin{bmatrix} I & \begin{bmatrix} -HZ_2^{-1} & BR_2^{-1/2} \end{bmatrix} R^{-1} \begin{bmatrix} -Z_2^{-1}H^T \\ R_2^{-1/2}B^T \end{bmatrix} \\ 0 & A^T - \begin{bmatrix} C^T Z_2^{-1} & C^T R_1^{-1/2} D \end{bmatrix} R^{-1} \begin{bmatrix} -Z_2^{-1}H^T \\ R_2^{-1/2}B^T \end{bmatrix} \end{bmatrix} \tag{5.8}
$$

With all the above properties of the DAREs and the notations, we are ready to prove the following theorem.

Theorem 1 *Let P, Q and X_∞ be the stabilizing solutions to the DAREs (2.5), (2.6) and (3.4), (or, (4.1), (4.2) and (4.3) when G is strictly proper), respectively, the following identity holds*

$$
\begin{aligned} X_\infty &= P\left[\left(1-\gamma^{-2}\right) I_n - \gamma^{-2} QP\right]^{-1} \\ &= \gamma^2 P \left[\gamma^2 I_n - (I_n + QP)\right]^{-1} \end{aligned} \tag{5.9}
$$

■

Proof: (See Appendix A.)

Similar work has been reported in [4], where the relationship between three discrete-time algebraic Riccati equations arising in the general $\mathcal{H}_\infty$ sub-optimal design was discussed. The result revealed here is explicitly concerning the discrete-time loop shaping design procedure, with three different DAREs and a slightly different conclusion.

6 Numerical algorithm and software

The loop shaping design of a positive feedback controller for a discrete-time system is accomplished by the algorithm DLSDP as listed below. For the sake of simplicity, Only the algorithm for the strictly proper case is included. The algorithm and code for the general case is available in the SLICOT library.

For given matrices $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times n}$ in the state-space description of the shaped system, algorithm *DLSDP* computes the matrices A_k, B_k, C_k, D_k of the positive feedback controller K.

Algorithm *DLSDP: Computation of positive feedback controller for Loop Shaping Design of a discrete-time system*

Compute $C_r = C^T C$

Compute $D_r = BB^T$

Solve $A^T P(I_n + D_r P)^{-1} A - P + C_r = 0$

Solve $AQ(I_n + C_r Q)^{-1} A^T - Q + D_r = 0$

Compute $\gamma_o = \sqrt{1 + \max\lambda_i(QP)}$

Compute $I_p + CQC^T$

Compute the eigensystem decomposition $UDU^T = I_p + CQC^T$

Compute $(I_p + CQC^T)^{-1} = UD^{-1}U^T$

Compute $Z_2 = UD^{-1/2}U^T$

Compute $Z_2^{-1} = UD^{1/2}U^T$

Compute $H = -AQC^T(I_p + CQC^T)^{-1}$

Compute $R_x = \begin{bmatrix} Z_2^{-2} - \gamma^2 I_p & 0 \\ 0 & I_m \end{bmatrix}$

Compute $B_x = [-HZ_2^{-1} \;\; B]$

Compute $S_x = [C^T Z_2^{-1} \;\; 0_{n\times m}]$

Solve $X = A^T XA + C_x - (S_x + A^T XB_x)(R_x + B_x XB)^{-1}(S_x^T + B_X^T XA)$

Compute $F = -(R_x + B_x^T XB_x)^{-1}(S_x^T + B_x^T XA)$

Set $F_1 = F(1:p,:)$, $F_2 = F(p+1:p+m,:)$

Compute $D_k = (I_m + B^T XB)^{-1} B^T XH$

Compute $B_k = -H + BD_k$

Compute $C_k = F_2 - D_K Z_2^{-1}(F_1 + Z_2 C)$

Compute $A_k = A + HC + BC_k$

Check the stability of the closed-loop state matrix

$$\begin{bmatrix} A + BD_kC & BC_k \\ B_kC & A_k \end{bmatrix}$$

The algorithm *DLSDP* is implemented by the double precision Fortran 77 subroutine SB10KD. The subroutine produces the (sub)optimal controller matrices along with estimates of the condition numbers of the systems of linear equations from which the P-, Q- and X-Riccati equations which are to be solved. These condition numbers gives an impression about the accuracy of the computations in obtaining the controller. The matrix Riccati equations are solved by the subroutine SB02OD from SLICOT. Listing of the subroutine SB10KD is given in the Appendix B.

7 A numerical example

Consider a sixth-order two-input two-output system with matrices

$$A = \begin{bmatrix} 0.2 & 0.0 & 0.3 & 0.0 & -0.3 & -0.1 \\ -0.3 & 0.2 & -0.4 & -0.3 & 0.0 & 0.0 \\ -0.1 & 0.1 & -0.1 & 0.0 & 0.0 & -0.3 \\ 0.1 & 0.0 & 0.0 & -0.1 & -0.1 & 0.0 \\ 0.0 & 0.3 & 0.6 & 0.2 & 0.1 & -0.4 \\ 0.2 & -0.4 & 0.0 & 0.0 & 0.2 & -0.2 \end{bmatrix},$$

$$B = \begin{bmatrix} -1.0 & -2.0 \\ 1.0 & 3.0 \\ -3.0 & -4.0 \\ 1.0 & -2.0 \\ 0.0 & 1.0 \\ 1.0 & 5.0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1.0 & -1.0 & 2.0 & -2.0 & 0.0 & -3.0 \\ -3.0 & 0.0 & 1.0 & -1.0 & 1.0 & -1.0 \end{bmatrix}.$$

Using the subroutine SB10KD one obtains for $\gamma = 1.1\gamma_o$ the following suboptimal controller matrices (up to four decimal digits)

$$A_k = \begin{bmatrix} 0.0337 & 0.0222 & 0.0858 & 0.1264 & -0.1872 & 0.1547 \\ 0.4457 & 0.0668 & -0.2255 & -0.3204 & -0.4548 & -0.0691 \\ -0.2419 & -0.2506 & -0.0982 & -0.1321 & -0.0130 & -0.0838 \\ -0.4402 & 0.3654 & -0.0335 & -0.2444 & 0.6366 & -0.6469 \\ -0.3623 & 0.3854 & 0.4162 & 0.4502 & 0.0065 & 0.1261 \\ -0.0121 & -0.4377 & 0.0604 & 0.2265 & -0.3389 & 0.4542 \end{bmatrix},$$

$$B_k = \begin{bmatrix} 0.0931 & -0.0269 \\ -0.0872 & 0.1599 \\ 0.0956 & -0.1469 \\ -0.1728 & 0.0129 \\ 0.2022 & -0.1154 \\ 0.2419 & -0.1737 \end{bmatrix},$$

$$C_k = \begin{bmatrix} -0.3677 & 0.2188 & 0.0403 & -0.0854 & 0.3564 & -0.3535 \\ 0.1624 & -0.0708 & 0.0058 & 0.0606 & -0.2163 & 0.1802 \end{bmatrix},$$

$$D_k = \begin{bmatrix} -0.0857 & -0.0246 \\ 0.0460 & 0.0074 \end{bmatrix}.$$

8 Conclusions

The discrete-time $\mathcal{H}_\infty$ loop shaping design procedure and its implementation in *SLICOT* has been summarised in this report. The solution formulae included those for the general case as well as those for the case of a strictly proper plant model. A result on the relationship between the three discrete-time algebraic Riccati equation solutions required was presented, which would be useful in making the computation more efficient and more reliable. The report also described the implementation of the synthesis algorithm, with an illustrative example.

References

[1] D.-W. Gu, P.Hr. Petkov and M.M. Konstantinov. *$\mathcal{H}_\infty$ Loop Shaping Design Procedure Routines in SLICOT*. **NICONET** Report 1999-15, http://www.win.tue.nl/wgs/niconet.html, November, 1999.

[2] P.Hr. Petkov, D.-W. Gu and M.M. Konstantinov. *Fortran 77 Routines for $\mathcal{H}_\infty$ and $\mathcal{H}_2$ Design of Linear Discrete-Time Control Systems.* **NICONET** Report 1999-5, http://www.win.tue.nl/wgs/niconet.html, May, 1999.

[3] T. Pappas, A.J. Laub and N.R. Sandell, JR. "On the Numerical Solution of the Discrete-Time Algebraic Riccati Equation."*IEEE Trans. Automat. Contr.*, **vol.AC-25**, pp. 631-641, 1980.

[4] D.J. Walker. "Relationship between three discrete-time $\mathcal{H}_\infty$ algebraic Riccati equation solutions."*International J. of Control*, **vol.52**, **No.4**, pp. 801-809, 1990.

Appendix A. The proof of Theorem 1

Assume that $P = P_2 P_1^{-1}$ and $X_\infty = X_2 X_1^{-1}$, satisfying

$$M_P \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = L_P \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} S_P \tag{8.1}$$

and

$$M_X \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = L_X \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} S_X \tag{8.2}$$

with S_P and S_X being stable.

The main idea in the proof is to find an invertible matrix W such that

$$M_X \begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}Q \\ 0 & I \end{bmatrix} = W M_P \tag{8.3}$$

and

$$L_X \begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}Q \\ 0 & I \end{bmatrix} = W L_P \tag{8.4}$$

If such a W exists, then by pre-multiplying W on the both side of (8.1), we have

$$M_X \begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}Q \\ 0 & I \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = L_X \begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}Q \\ 0 & I \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} S_P \tag{8.5}$$

By comparing (8.5) with (8.2), and from the properties of the stabilizing solution to a DARE discussed in Section 5, it can be deducted that

$$\begin{aligned} X_\infty &= \left(\begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}\right)\left(\begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}Q \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}\right)^{-1} \\ &= P_2 \left[-\gamma^{-2}(1-\gamma^2)P_1 - \gamma^{-2}QP_2\right]^{-1} \\ &= P_2 P_1^{-1} \left[(1-\gamma^{-2})I_n - \gamma^{-2}QP_2P_1^{-1}\right]^{-1} \\ &= P\left[(1-\gamma^{-2})I_n - \gamma^{-2}QP\right]^{-1} \\ &= \gamma^2 P \left[\gamma^2 I_n - (I_n + QP)\right]^{-1} \end{aligned} \tag{8.6}$$

It can be shown that

$$W = \begin{bmatrix} -\gamma^{-2}(1-\gamma^2)I_n & -\gamma^{-2}(1-\gamma^2)(A - BD^T R_1^{-1} C)Q\left[(1-\gamma^2)I_n + C^T R_1^{-1} CQ\right]^{-1} \\ 0 & (1-\gamma^2)\left[(1-\gamma^2)I_n + C^T R_1^{-1} CQ\right]^{-1} \end{bmatrix} \tag{8.7}$$

satisfies (8.3) and (8.4), by matrix manipulations and noticing that

$$R^{-1} = \begin{bmatrix} I_p & 0 \\ -D^T R_1^{-1/2} Z_2^{-1} & I_m \end{bmatrix} \begin{bmatrix} (Z_2^{-1} R_1^{-1} Z_2^{-1} - \gamma^2 I_p & 0 \\ 0 & I_m \end{bmatrix} \begin{bmatrix} I_p & -Z_2^{-1} R_1^{-1/2} D \\ 0 & I_m \end{bmatrix}$$

In the strictly proper case, W is simply

$$W = \begin{bmatrix} -\gamma^{-2}(1-\gamma^2) I_n & -\gamma^{-2}(1-\gamma^2) AQ \left[(1-\gamma^2) I_n + C^T C Q\right]^{-1} \\ 0 & (1-\gamma^2)\left[(1-\gamma^2) I_n + C^T C Q\right]^{-1} \end{bmatrix} \tag{8.8}$$

Appendix B. A Fortran 77 subroutine for Loop Shaping Design of discrete-time systems

```
      SUBROUTINE SB10KD( N, M, NP, A, LDA, B, LDB, C, LDC, FACTOR,
     $                   AK, LDAK, BK, LDBK, CK, LDCK, DK, LDDK,
     $                   RCOND, IWORK, WORK, LWORK, BWORK, INFO )
C
C     RELEASE 3.0, WGS COPYRIGHT 2000.
C
C     PURPOSE
C
C     To compute the matrices of the positive feedback controller
C
C              | AK | BK |
C          K = |----|----|
C              | CK | DK |
C
C     for the shaped plant
C
C              | A | B |
C          G = |---|---|
C              | C | 0 |
C
C     in the Discrete-Time Loop Shaping Design Procedure.
C
C     ARGUMENTS
C
C     Input/Output Parameters
C
C     N       (input) INTEGER
C             The order of the plant. N >= 0.
C
C     M       (input) INTEGER
C             The column size of the matrix B. M >= 0.
C
C     NP      (input) INTEGER
C             The row size of the matrix C. NP >= 0.
```

```
C
C     A       (input) DOUBLE PRECISION array, dimension (LDA,N)
C             The N-by-N system state matrix A of the shaped plant.
C
C     LDA     INTEGER
C             The leading dimension of the array A. LDA >= max(1,N).
C
C     B       (input) DOUBLE PRECISION array, dimension (LDB,M)
C             The N-by-M system input matrix B of the shaped plant.
C
C     LDB     INTEGER
C             The leading dimension of the array B. LDB >= max(1,N).
C
C     C       (input) DOUBLE PRECISION array, dimension (LDC,N)
C             The NP-by-N system output matrix C of the shaped plant.
C
C     LDC     INTEGER
C             The leading dimension of the array C. LDC >= max(1,NP).
C
C     FACTOR  (input) DOUBLE PRECISION
C             = 1 implies that an optimal controller is required
C             > 1 implies that a suboptimal controller is required
C                 achieving a performance FACTOR less than optimal.
C
C     AK      (output) DOUBLE PRECISION array, dimension (LDAK,N)
C             The N-by-N controller state matrix AK.
C
C     LDAK    INTEGER
C             The leading dimension of the array AK. LDAK >= max(1,N).
C
C     BK      (output) DOUBLE PRECISION array, dimension (LDBK,NP)
C             The N-by-NP controller input matrix BK.
C
C     LDBK    INTEGER
C             The leading dimension of the array BK. LDBK >= max(1,N).
C
C     CK      (output) DOUBLE PRECISION array, dimension (LDCK,N)
C             The M-by-N controller output matrix C.
```

```
C
C     LDCK    INTEGER
C             The leading dimension of the array CK.
C             LDCK >= max(1,M).
C
C     DK      (output) DOUBLE PRECISION array, dimension (LDDK,NP)
C             The M-by-NP controller matrix DK.
C
C     LDDK    INTEGER
C             The leading dimension of the array DK.
C             LDDK >= max(1,M).
C
C     RCOND  (output) DOUBLE PRECISION array, dimension (2)
C             RCOND(1) contains an estimate of the reciprocal condition
C                      number of the linear system of equations from
C                      which the solution of the P-Riccati equation is
C                      obtained
C             RCOND(2) contains an estimate of the reciprocal condition
C                      number of the linear system of equations from
C                      which the solution of the Q-Riccati equation is
C                      obtained
C             RCOND(3) contains an estimate of the reciprocal condition
C                      number of the linear system of equations from
C                      which the solution of the X-Riccati equation is
C                      obtained
C             RCOND(4) contains an estimate of the reciprocal condition
C                      number of the matrix Rx + Bx'*X*Bx
C
C     Workspace
C
C     IWORK   INTEGER array, dimension 2*max(N,NP+M)
C
C     WORK    DOUBLE PRECISION array, dimension (LWORK)
C             On exit, if INFO = 0, WORK(1) contains the optimal LWORK.
C
C     LWORK   INTEGER
C             The dimension of the array WORK.
C             LWORK >= 17*N*N + M*M + 8*NP*NP + 12*M*N + 2*M*NP + 12*N*NP +
```

```
C                    6*N + NP + max(14*N+23,16*N).
C            For good performance, LWORK must generally be larger.
C
C     BWORK   LOGICAL array, dimension (2*N)
C
C     Error Indicator
C
C     INFO    (output) INTEGER
C             = 0: successful exit
C             < 0: if INFO = -i, the i-th argument had an illegal value
C             = 1: The P-Riccati equation is not solved successfuly
C             = 2: The Q-Riccati equation is not solved successfuly
C             = 3: The X-Riccati equation is not solved successfuly
C             = 4: The iteration to compute eigenvalues failed to converge
C             = 5: The matrix Rx + Bx'*X*Bx is singular
C             = 6: The closed-loop system is unstable
C
C     METHOD
C
C     The routine implements the method, presented in [1].
C
C     REFERENCES
C
C     [1] D.-W. Gu, P.H. Petkov, M.M. Konstantinov. On discrete
C         H_infinity loop shaping design procedure routines. NICONET
C         Report 2000-xx, http://www.win.tue.nl/wgs/niconet.html,
C         December 2000.
C
C     NUMERICAL ASPECTS
C
C     The accuracy of the results depends on the conditioning of the
C     two Riccati equations solved in the controller design. For
C     better conditioning it is advised to take FACTOR > 1.
C
C     CONTRIBUTORS
C
C     P.Hr. Petkov, D.W. Gu and M.M. Konstantinov, October 2000
C
```

```
C     KEYWORDS
C
C     Loop-shaping design, Robust control, H_infinity control
C
C     ******************************************************************
C
C     .. Parameters ..
      DOUBLE PRECISION   ZERO, ONE
      PARAMETER          ( ZERO = 0.0D+0, ONE = 1.0D+0 )
C     ..
C     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LDB, LDC, LDAK, LDBK, LDCK, LDDK,
     $                   LWORK, M, N, NK, NP
      DOUBLE PRECISION   FACTOR
C     ..
C     .. Array Arguments ..
      INTEGER            IWORK( * )
      LOGICAL            BWORK( * )
      DOUBLE PRECISION   A( LDA, * ), B( LDB, * ), C( LDC, * ),
     $                   AK( LDAK, * ), BK( LDBK, * ), CK( LDCK, * ),
     $                   DK( LDDK, * ), RCOND( 3 ), WORK( * )
C     ..
C     .. Local Scalars ..
      INTEGER            I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11,
     $                   I12, I13, I14, I15, I16, I17, I18, I19, I20,
     $                   I21, I22, I23, I24, I25, I26, I27, I28, INFO2,
     $                   IWRK, LWA, LWAMAX, MINWRK, N2, NS, SDIM
      DOUBLE PRECISION   GAMMA, RNORM
C     ..
C     .. External Functions ..
      LOGICAL SELECT
      DOUBLE PRECISION DLANSY
      EXTERNAL DLANSY, SELECT
C     ..
C     .. External Subroutines ..
      EXTERNAL           DGEMM, DGEES, DLACPY, DLASET, DPOTRF, DPOTRS,
     $                   DSYCON, DSYTRF, DSYTRS, SB02OD, XERBLA
C     ..
```

```
C     .. Intrinsic Functions ..
      INTRINSIC          MAX, SQRT
C     ..
C     .. Executable Statements ..
C
C     Decode and Test input parameters.
C
      INFO = 0
      IF( N.LT.0 ) THEN
         INFO = -1
      ELSE IF( M.LT.0 ) THEN
         INFO = -2
      ELSE IF( NP.LT.0 ) THEN
         INFO = -3
      ELSE IF( LDA.LT.MAX( 1, N ) ) THEN
         INFO = -5
      ELSE IF( LDB.LT.MAX( 1, N ) ) THEN
         INFO = -7
      ELSE IF( LDC.LT.MAX( 1, NP ) ) THEN
         INFO = -9
      ELSE IF( FACTOR.LT.ONE ) THEN
         INFO = -10
      ELSE IF( LDAK.LT.MAX( 1, N ) ) THEN
         INFO = -12
      ELSE IF( LDBK.LT.MAX( 1, N ) ) THEN
         INFO = -14
      ELSE IF( LDCK.LT.MAX( 1, M ) ) THEN
         INFO = -16
      ELSE IF( LDDK.LT.MAX( 1, M ) ) THEN
         INFO = -18
      END IF
C
C     Compute workspace.
C
      MINWRK = 17*N*N + M*M + 8*NP*NP + 12*M*N + 2*M*NP + 12*N*NP +
     $         6*N + NP + MAX( 14*N+23, 16*N )
      IF( LWORK.LT.MINWRK ) THEN
         INFO = -22
```

```
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'SB10KD', -INFO )
         RETURN
      END IF
C
C     Quick return if possible.
C
      IF( N.EQ.0 .OR. M.EQ.0 .OR. NP.EQ.0 ) RETURN
C
C     Workspace usage.
C
      I1 = N*N
      I2 = I1 + N*N
      I3 = I2 + N*N
      I4 = I3 + N*N
      I5 = I4 + 2*N
      I6 = I5 + 2*N
      I7 = I6 + 2*N
      I8 = I7 + 4*N*N
      I9 = I8 + 4*N*N
      I10 = I9 + 4*N*N
      IWRK = I10 + N*N
      LWAMAX = 0
C
C     Compute Cr = C'*C .
C
      CALL DGEMM( 'T', 'N', N, N, NP, ONE, C, LDC, C, LDC, ZERO,
     $            WORK( I2+1 ), N )
C
C     Compute Dr = B*B' .
C
      CALL DGEMM( 'N', 'T', N, N, M, ONE, B, LDB, B, LDB, ZERO,
     $            WORK( I3+1 ), N )
C                                                     -1
C     Solution of the Riccati equation A'*P*(In + Dr*P)  A - P +
C                                              Cr = 0
      N2 = 2*N
```

```
      CALL SB02OD( 'D', 'G', 'N', 'U', 'Z', 'S', N, M, NP, A, LDA,
     $             WORK( I3+1 ), N, WORK( I2+1 ), N, WORK, M, WORK,
     $             N, RCOND( 1 ), WORK, N, WORK( I4+1 ), WORK( I5+1 ),
     $             WORK( I6+1 ), WORK( I7+1 ), N2, WORK( I8+1 ), N2,
     $             WORK( I9+1 ), N2, -ONE, IWORK, WORK( IWRK+1 ),
     $             LWORK-IWRK, BWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 1
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Transpose A .
C
      DO 40 J = 1, N
         DO 30 I = 1, N
            WORK( I10+I+(J-1)*N ) = A( J,I )
   30    CONTINUE
   40 CONTINUE
C                                                            -1
C     Solution of the Riccati equation A*Q*(In + Cr*Q)  *A' - Q +
C                                       Dr = 0
      CALL SB02OD( 'D', 'G', 'N', 'U', 'Z', 'S', N, M, NP,
     $             WORK( I10+1 ), N, WORK( I2+1 ), N, WORK( I3+1 ), N,
     $             WORK, M, WORK, N, RCOND( 2 ), WORK( I1+1 ), N,
     $             WORK( I4+1 ), WORK( I5+1 ), WORK( I6+1 ),
     $             WORK( I7+1 ), N2, WORK( I8+1 ), N2, WORK( I9+1 ), N2,
     $             -ONE, IWORK, WORK( IWRK+1 ), LWORK-IWRK, BWORK,
     $             INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 2
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Compute gamma.
```

```
C
      CALL DGEMM( 'N', 'N', N, N, N, ONE, WORK( I1+1 ), N, WORK, N,
     $            ZERO, WORK( I10+1 ), N )
      CALL DGEES( 'N', 'N', SELECT, N, WORK( I10+1 ), N, SDIM,
     $            WORK( I6+1 ), WORK( I7+1 ), WORK( IWRK+1 ), N,
     $            WORK( IWRK+1 ), LWORK-IWRK, BWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 4
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
      GAMMA = ZERO
      DO 50 I = 1, N
         GAMMA = MAX( GAMMA, WORK( I6+I ) )
   50 CONTINUE
      GAMMA = SQRT( ONE + GAMMA )
      GAMMA = FACTOR*GAMMA
C
C     Workspace usage.
C
      I3 = I2 + N*NP
      I4 = I3 + NP*NP
      I5 = I4 + NP*NP
      I6 = I5 + NP*NP
      I7 = I6 + NP
      I8 = I7 + NP*NP
      I9 = I8 + NP*NP
      I10 = I9 + NP*NP
      I11 = I10 + N*NP
      I12 = I11 + N*NP
      I13 = I12 + ( NP+M )*( NP+M )
      I14 = I13 + N*( NP+M )
      I15 = I14 + N*( NP+M )
      I16 = I15 + N*N
      I17 = I16 + 2*N
      I18 = I17 + 2*N
      I19 = I18 + 2*N
```

```
      I20 = I19 + ( 2*N+NP+M )*( 2*N+NP+M )
      I21 = I20 + ( 2*N+NP+M )*2*N
      I22 = I21 + ( 2*N+NP+M )*2*N
      I23 = I22 + ( NP+M )*N
      I24 = I23 + ( NP+M )*( NP+M )
      I25 = I24 + ( NP+M )*N
      I26 = I25 + M*M
      I27 = I26 + M*N
      I28 = I27 + M*NP
      IWRK = I22 + NP*N
C
C     Compute Q*C' .
C
      CALL DGEMM( 'N', 'T', N, NP, N, ONE, WORK( I1+1 ), N, C, LDC,
     $            ZERO, WORK( I2+1 ), N )
C
C     Compute Ip + C*Q*C' .
C
      CALL DLASET( 'Full', NP, NP, ZERO, ONE, WORK( I3+1 ), NP )
      CALL DGEMM( 'N', 'N', NP, NP, N, ONE, C, LDC, WORK( I2+1 ), N,
     $            ONE, WORK( I3+1 ), NP )
C
C     Compute the eigenvalues and eigenvectors of Ip + C'*Q*C
C
      CALL DLACPY( 'U', NP, NP, WORK( I3+1 ), NP, WORK( I5+1 ), NP )
      CALL DSYEV( 'V', 'U', NP, WORK( I5+1 ), NP, WORK( I6+1 ),
     $           WORK( IWRK+1 ), LWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 4
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C                              -1
C     Compute ( Ip + C'*Q*C )
C
      DO 70 J = 1, NP
         DO 60 I = 1, NP
```

```
            WORK( I9+I+(J-1)*NP ) = WORK( I5+J+(I-1)*NP ) / WORK( I6+I )
   60    CONTINUE
   70 CONTINUE
      CALL DGEMM( 'N', 'N', NP, NP, NP, ONE, WORK( I5+1 ), NP,
     $            WORK( I9+1 ), NP, ZERO, WORK( I4+1 ), NP )
C
C     Compute Z2
C
      DO 90 J = 1, NP
         DO 80 I = 1, NP
            WORK( I9+I+(J-1)*NP ) = WORK( I5+J+(I-1)*NP ) /
     $                              SQRT( WORK( I6+I ) )
   80    CONTINUE
   90 CONTINUE
      CALL DGEMM( 'N', 'N', NP, NP, NP, ONE, WORK( I5+1 ), NP,
     $            WORK( I9+1 ), NP, ZERO, WORK( I7+1 ), NP )
C               -1
C     Compute Z2
C
      DO 110 J = 1, NP
         DO 100 I = 1, NP
            WORK( I9+I+(J-1)*NP ) = WORK( I5+J+(I-1)*NP )*
     $                              SQRT( WORK( I6+I ) )
  100    CONTINUE
  110 CONTINUE
      CALL DGEMM( 'N', 'N', NP, NP, NP, ONE, WORK( I5+1 ), NP,
     $            WORK( I9+1 ), NP, ZERO, WORK( I8+1 ), NP )
C
C     Compute A*Q*C' .
C
      CALL DGEMM( 'N', 'N', N, NP, N, ONE, A, LDA,
     $            WORK( I2+1 ), N, ZERO, WORK( I10+1 ), N )
C                                         -1
C     Compute H = -A*Q*C'*( Ip + C*Q*C' )   .
C
      CALL DGEMM( 'N', 'N', N, NP, NP, -ONE, WORK( I10+1 ), N,
     $            WORK( I4+1 ), NP, ZERO, WORK( I11+1 ), N )
C
```

```
C     Compute Rx .
C
      CALL DLASET( 'F', NP+M, NP+M, ZERO, ONE, WORK( I12+1 ), NP+M )
      DO 130 J = 1, NP
         DO 120 I = 1, NP
            IF( I.EQ.J ) THEN
               WORK( I12+I+(I-1)*(NP+M) ) = WORK( I3+I+(I-1)*NP ) -
     $                                      GAMMA*GAMMA
            ELSE
               WORK( I12+I+(J-1)*(NP+M) ) = WORK( I3+I+(J-1)*NP )
            END IF
  120    CONTINUE
  130 CONTINUE
C
C     Compute Bx .
C
      CALL DGEMM( 'N', 'N', N, NP, NP, -ONE, WORK( I11+1 ), N,
     $           WORK( I8+1 ), NP, ZERO, WORK( I13+1 ), N )
      DO 150 J = 1, M
         DO 140 I = 1, N
            WORK( I13+N*NP+I+(J-1)*N ) = B( I, J )
  140    CONTINUE
  150 CONTINUE
C
C     Compute Sx .
C
      CALL DGEMM( 'T', 'N', N, NP, NP, ONE, C, LDC,
     $           WORK( I8+1 ), NP, ZERO, WORK( I14+1 ), N )
      CALL DLASET( 'F', N, M, ZERO, ZERO, WORK( I14+N*NP+1 ), N )
C
C     Solve the Riccati equation
C                                                         -1
C       X = A'*X*A + Cx - (Sx + A'*X*Bx)*(Rx + Bx'*X*B )  *(Sx'+Bx'*X*A)
C
      CALL SB02OD( 'D', 'B', 'C', 'U', 'N', 'S', N, NP+M, NP, A, LDA,
     $            WORK( I13+1 ), N, C, LDC, WORK( I12+1 ), NP+M,
     $            WORK( I14+1 ), N, RCOND( 3 ), WORK( I15+1 ), N,
     $            WORK( I16+1 ), WORK( I17+1 ), WORK( I18+1 ),
```

```
     $             WORK( I19+1 ), 2*N+NP+M, WORK( I20+1 ), 2*N+NP+M,
     $             WORK( I21+1 ), 2*N+NP+M, -ONE, IWORK, WORK( IWRK+1 ),
     $             LWORK-IWRK, BWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 3
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Compute Bx'*X .
C
      CALL DGEMM( 'T', 'N', NP+M, N, N, ONE, WORK( I13+1 ), N,
     $            WORK( I15+1 ), N, ZERO, WORK( I22+1 ), NP+M )
C
C     Compute Rx + Bx'*X*Bx .
C
      CALL DLACPY( 'F', NP+M, NP+M, WORK( I12+1 ), NP+M, WORK( I23+1 ),
     $              NP+M )
      CALL DGEMM( 'N', 'N', NP+M, NP+M, N, ONE, WORK( I22+1 ), NP+M,
     $             WORK( I13+1 ), N, ONE, WORK( I23+1 ), NP+M )
C
C     Compute -( Sx' + Bx'*X*A ) .
C
      DO 170 J = 1, N
         DO 160 I = 1, NP+M
            WORK( I24+I+(J-1)*(NP+M) ) = WORK( I14+J+(I-1)*N )
  160    CONTINUE
  170 CONTINUE
      CALL DGEMM( 'N', 'N', NP+M, N, N, -ONE, WORK( I22+1 ), NP+M,
     $             A, LDA, -ONE, WORK( I24+1 ), NP+M )
C
C     Factorize Rx + Bx'*X*Bx .
C
      RNORM = DLANSY( '1', 'U', NP+M, WORK( I23+1 ), NP+M,
     $                WORK( IWRK+1 ) )
      CALL DSYTRF( 'U', NP+M, WORK( I23+1 ), NP+M, IWORK,
     $             WORK( IWRK+1 ), LWORK-IWRK, INFO2 )
```

```
      IF( INFO2.NE.0 ) THEN
         INFO = 5
         RETURN
      END IF
      CALL DSYCON( 'U', NP+M, WORK( I23+1 ), NP+M, IWORK, RNORM,
     $             RCOND( 4 ), WORK( IWRK+1 ), IWORK( NP+M+1), INFO2 )
C                                     -1
C     Compute F = -( Rx + Bx'*X*Bx )  ( Sx' + Bx'*X*A ) .
C
      CALL DSYTRS( 'U', NP+M, N, WORK( I23+1 ), NP+M, IWORK,
     $             WORK( I24+1 ), NP+M, INFO2 )
C
C     Compute B'*X .
C
      CALL DGEMM( 'T', 'N', M, N, N, ONE, B, LDB, WORK( I15+1 ), N,
     $             ZERO, WORK( I26+1 ), M )
C
C     Compute Im + B'*X*B .
C
      CALL DLASET( 'Full', M, M, ZERO, ONE, WORK( I25+1 ), M )
      CALL DGEMM( 'N', 'N', M, M, N, ONE, WORK( I26+1 ), M, B, LDB, ONE,
     $             WORK( I25+1 ), M )
C
C     Factorize Im + B'*X*B .
C
      CALL DPOTRF( 'U', M, WORK( I25+1 ), M, INFO2 )
C                              -1
C     Compute ( Im + B'*X*B )  B'*X .
C
      CALL DPOTRS( 'U', M, N, WORK( I25+1 ), M, WORK( I26+1 ),
     $             M, INFO2 )
C                                   -1
C     Compute Dk = ( Im + B'*X*B )  B'*X*H .
C
      CALL DGEMM( 'N', 'N', M, NP, N, ONE, WORK( I26+1 ), M,
     $             WORK( I11+1 ), N, ZERO, DK, LDDK )
C
C     Compute Bk = -H + B*Dk .
```

```
C
      CALL DLACPY( 'F', N, NP, WORK( I11+1 ), N, BK, LDBK )
      CALL DGEMM( 'N', 'N', N, NP, M, ONE, B, LDB, DK, LDDK, -ONE,
     $            BK, LDBK )
C                  -1
C     Compute Dk*Z2  .
C
      CALL DGEMM( 'N', 'N', M, NP, NP, ONE, DK, LDDK,
     $            WORK( I8+1 ), NP, ZERO, WORK( I27+1 ), M )
C
C     Compute F1 + Z2*C .
C
      CALL DLACPY( 'F', NP, N, WORK( I24+1 ), NP+M, WORK( I28+1 ), NP )
      CALL DGEMM( 'N', 'N', NP, N, NP, ONE, WORK( I7+1 ), NP, C, LDC,
     $            ONE, WORK( I28+1 ), NP )
C                             -1
C     Compute Ck = F2 - Dk*Z2  *( F1 + Z2*C ) .
C
      CALL DLACPY( 'F', M, N, WORK( I24+NP+1 ), NP+M, CK, LDCK )
      CALL DGEMM( 'N', 'N', M, N, NP, -ONE, WORK( I27+1 ), M,
     $            WORK( I28+1 ), NP, ONE, CK, LDCK )
C
C     Compute Ak = A + H*C + B*Ck .
C
      CALL DLACPY( 'F', N, N, A, LDA, AK, LDAK )
      CALL DGEMM( 'N', 'N', N, N, NP, ONE, WORK( I11+1 ), N,
     $            C, LDC, ONE, AK, LDAK )
      CALL DGEMM( 'N', 'N', N, N, M, ONE, B, LDB,
     $            CK, LDCK, ONE, AK, LDAK )
C
C     Workspace usage.
C
      I1 = M*N
      I2 = I1 + 4*N*N
      I3 = I2 + 2*N
      IWRK = I3 + 2*N
C
C     Compute Dk*C .
```

```
C
      CALL DGEMM( 'N', 'N', M, N, NP, ONE, DK, LDDK, C, LDC, ZERO,
     $             WORK, M )
C
C     Compute the closed-loop state matrix.
C
      CALL DLACPY( 'F', N, N, A, LDA, WORK( I1+1 ), 2*N )
      CALL DGEMM( 'N', 'N', N, N, M, -ONE, B, LDB, WORK,
     $            M, ONE, WORK( I1+1 ), 2*N )
      CALL DGEMM( 'N', 'N', N, N, M, ONE, B, LDB, CK, LDCK,
     $            ZERO, WORK( I1+2*N*N+1 ), 2*N )
      CALL DGEMM( 'N', 'N', N, N, NP, -ONE, BK, LDBK, C, LDC,
     $            ZERO, WORK( I1+N+1 ), 2*N )
      CALL DLACPY( 'F', N, N, AK, LDAK, WORK( I1+2*N*N+N+1 ),
     $             2*N )
C
C     Compute the closed-loop poles.
C
      CALL DGEES( 'N', 'N', SELECT, 2*N, WORK( I1+1 ), 2*N, SDIM,
     $            WORK( I2+1 ), WORK( I3+1 ), WORK( IWRK+1 ), N,
     $            WORK( IWRK+1 ), LWORK-IWRK, BWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 4
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Check the stability of the closed-loop system.
C
      NS = 0
      DO 180 I = 1, 2*N
         IF( WORK( I2+I )*WORK( I2+I )+
     $       WORK( I3+I )*WORK( I3+I ).GT.ONE ) NS = NS + 1
  180 CONTINUE
      IF( NS.GT.0 ) THEN
         INFO = 6
         RETURN
```

```
      END IF
C
      LWA = 17*N*N + M*M + 8*NP*NP + 12*M*N + 2*M*NP + 12*N*NP + 6*N +
     $      NP + LWAMAX
      WORK( 1 ) = DBLE( LWA )
      RETURN
C *** Last line of SB10KD ***
      END
```