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

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

November 1999

[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/nic1999-15.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) and its implementation in the software package *SLICOT*. The developed routines are tested in a design example, and are included as appendices.

Key Words: Robust control systems design, $\mathcal{H}_\infty$ loop shaping design procedure, Normalised coprime factor perturbation, SLICOT

1 Introduction

Since early 1980s, the $\mathcal{H}_\infty$ optimisation approach has been developed into a powerful yet flexible robust control systems theory and design method, espcially for linear systems. Several formulations of cost function are applicable in the robust controller design. For instances, the weighted S/KS and S/T design methods. The optimisation of S/KS, where S is the sensitivity function and K the controller to be designed, would achieve the nominal performance in terms of tracking or output disturbance rejection and robustly stabilise the system against additive model perturbations. On the other hand, the mixed sensitivity optimisation of S/T, where T is the complementary sensitivity function, would achieve robust stability against multiplicative model perturbations in addition to the nominal performance. Both of them are useful robust controller design methods, but the model perturbation representations are limited by the condition on the number of right-half complex plane poles. Also, there may exist undesirable pole-zero cancellations between the nominal model and the $\mathcal{H}_\infty$ controllers. In this report, an alternative way to represent the model uncertainty is introduced. The uncertainty is described by the perturbations directly on the coprime factors of the nominal model. The $\mathcal{H}_\infty$ robust stabilisation against such perturbations and the consequently developed design method, the $\mathcal{H}_\infty$ loop shaping design procedure (LSDP), would remove the restrictions on the number of right-half plane poles and produce no pole-zero cancellations between the nominal model and controller designed. This method does not require an iterative procedure to obtain an optimal solution and thus raises the computational efficiency. Furthermore, the $\mathcal{H}_\infty$ LSDP inherits classical loop shaping design ideas so that practising control engineers would feel more comfortable to use it.

The readership of this report is the SLICOT users. The introductions on the robust control against normalised coprime factor perturbations and the $\mathcal{H}_\infty$ loop shaping design procedure will be brief. Interested readers should consult the reference [1, 2]. The report will, however, introduce in detail the routines developed in the SLICOT package to implement the $\mathcal{H}_\infty$ loop shaping design procedure and will illustrate the usage of the routines with a design example.

2 Robust stabilisation against normalised coprime factor perturbations

In this section, the main results on normalised coprime factorization and related robust stabilisation given by McFarlane and Glover [1, 2] would be summarised.

2.1 Left coprime factorization

Matrices $(\tilde{M},\ \tilde{N}) \in \mathcal{H}_\infty^+$, where $\mathcal{H}_\infty^+$ denotes the space of functions with no poles in the closed right half complex plane, constitute a left coprime factorization of a given plant model G if and only if

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

(ii) the plant model is given by

$$G = \tilde{M}^{-1}\tilde{N} \tag{2.1}$$

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

$$\tilde{M}\tilde{V} \ + \ \tilde{N}\tilde{U} \ = \ I \tag{2.2}$$

A left coprime factorization of a plant model G as defined in (2.1) is *normalized* if and only if

$$\tilde{N}\tilde{N}^- \ + \ \tilde{M}\tilde{M}^- \ = \ I, \qquad \forall s \tag{2.3}$$

where $\tilde{N}^-(s) = \tilde{N}^T(-s)$, etc.

For a minimal realization of $G(s)$,

$$\begin{aligned} G(s) &= D + C(sI - A)^{-1}B && (2.4)\\ &\stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] && (2.5)\end{aligned}$$

A state-space construction for the normalized left coprime factorization can be obtained in terms of the solution to the generalized filter algebraic Riccati equation

$$\begin{aligned}(A - BD^TR^{-1}C)Z + Z(A - BD^TR^{-1}C)^T &- ZC^TR^{-1}CZ \\ &+ B(I - D^TR^{-1}D)B^T = 0 \qquad (2.6)\end{aligned}$$

where $R := I + DD^T$ and $Z \geq 0$ is the unique stabilizing solution. If $H = -(ZC^T + BD^T)R^{-1}$, then

$$\left[\begin{array}{cc} \tilde{N} & \tilde{M} \end{array}\right] \stackrel{s}{=} \left[\begin{array}{c|cc} A + HC & B + HD & H \\ \hline R^{-1/2}C & R^{-1/2}D & R^{-1/2} \end{array}\right] \tag{2.7}$$

is a normalizaed left coprime factorization of G such that $G = \tilde{M}^{-1}\tilde{N}$.

The normalized right coprime factorization can be defined dually and a state-space representation can be similarly obtained in terms of the solution to the generalized control algebraic Riccati equation [1],

$$\begin{aligned}(A-BS^{-1}D^TC)^TX+X(A-BS^{-1}D^TC) &- XBS^{-1}B^TX \\ &+ C^T(I-DS^{-1}D^T)C=0 \end{aligned} \tag{2.8}$$

where $S := I+D^TD$ and $X \geq 0$ is the unique stabilizing solution.

A perturbed plant transfer function can be described by

$$G_\Delta = (\tilde{M}+\Delta_{\tilde{M}})^{-1}(\tilde{N}+\Delta_{\tilde{N}})$$

where $(\Delta_{\tilde{M}}, \Delta_{\tilde{N}})$ are stable, unknown transfer functions which represent the uncertainty (perturbation) in the nominal plant model. The design objective of robust stabilisation is to stabilize not only the nominal model G, but the family of perturbed plants defined by

$$\mathcal{G}_\epsilon = \{(\tilde{M}+\Delta_{\tilde{M}})^{-1}(\tilde{N}+\Delta_{\tilde{N}}) : \|[\Delta_{\tilde{M}}, \Delta_{\tilde{N}}]\|_\infty < \epsilon\}$$

where $\epsilon > 0$ is the stability margin. Using a feedback controller K as shown schematically in Fig. 1, the feedback system $(\tilde{M}, \tilde{N}, K, \epsilon)$ is said to be robustly stable if and only if (G, K) is internally stable and

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

To maximise robust stability of the closed-loop system given in Fig. 1, one must minimise

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

The lowest achievable value of γ for all stabilizing controllers K is

$$\gamma_o = \inf_{K\ stabilizing} \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I-GK)^{-1}\tilde{M}^{-1} \right\|_\infty \tag{2.9}$$

and is given in [1] by

$$\gamma_o = (1 - \|[\tilde{N} \quad \tilde{M}]\|_H^2)^{-1/2} \tag{2.10}$$

where $\|\cdot\|_H$ denotes the Hankel norm. From [1]

$$\|[\tilde{N} \quad \tilde{M}]\|_H^2 = \lambda_{max}(ZX(I+ZX)^{-1}) \tag{2.11}$$

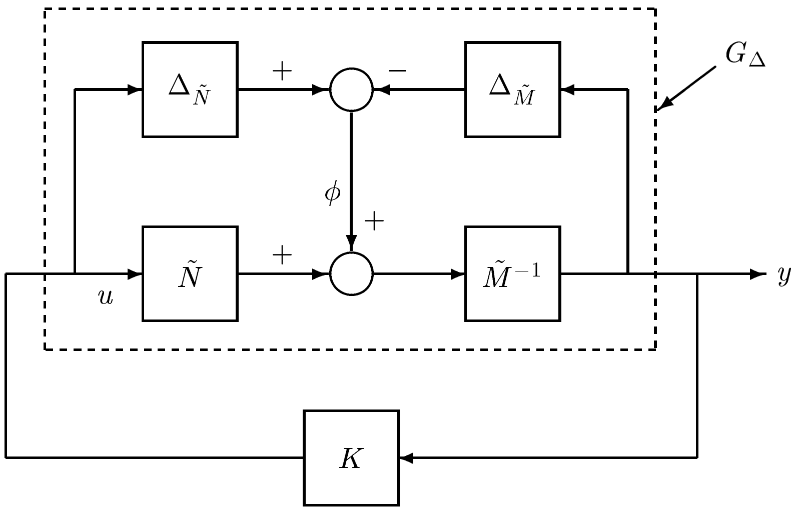

Figure 1: Robust stabilisation with regard to coprime factor uncertainty.

where $\lambda_{max}(\cdot)$ represents the maximum eigenvalue, hence from (2.10),

$$\gamma_o = (1 + \lambda_{max}(ZX))^{1/2} \tag{2.12}$$

From [1], all controllers optimising γ are given by $K = UV^{-1}$, where U and V are stable and are right coprime factorizations of K, and where

$$\left\| \begin{bmatrix} -\tilde{N}^* \\ \tilde{M}^* \end{bmatrix} + \begin{bmatrix} U \\ V \end{bmatrix} \right\|_\infty = \|[\tilde{N} \quad \tilde{M}]\|_H$$

This is a Hankel approximation problem and can be solved using an algorithm developed by Glover [3].

A controller which achieves a $\gamma > \gamma_0$ is given in [1] by

$$K \stackrel{s}{=} \left[\begin{array}{c|c} A + BF + \gamma^2 (L^T)^{-1} Z C^T (C + DF) & \gamma^2 (L^T)^{-1} Z C^T \\ \hline B^T X & -D^T \end{array}\right] \tag{2.13}$$

where $F = -S^{-1}(D^T C + B^T X)$ and $L = (1 - \gamma^2)I + XZ$.

However, if $\gamma = \gamma_o$, then $L = XZ - \lambda_{max}(XZ)I$ which is singular, and thus (2.13) cannot be implemented. This problem can be resolved using the descriptor system [5, 6]. A controller which achieves a $\gamma \geq \gamma_o$ can be given in the descriptor form by

$$K \stackrel{s}{=} \left[\begin{array}{c|c} -L^T s + L^T (A + BF) + \gamma^2 Z C^T (C + DF) & \gamma^2 Z C^T \\ \hline B^T X & -D^T \end{array}\right] \tag{2.14}$$

2.2 A loop shaping design procedure

In the classical control systems design with single-input-single-output (SISO) systems, it is a well known and practically effective technique to use a compensator to alter the frequency response (the Bode diagram) of the open-loop transfer function so that the unity feedback system would achieve stability, good performance and certain robustness. A direct extension of the above method into multivariable systems is difficult not only because of multiinput and multioutput are involved but also due to the loss of phase information which makes it impossible to predict the stability of the closed-loop system formed by the unity feedback. However, based on the robust stabilization against perturbations on normalised coprime factorizations, a design method, known as the $\mathcal{H}_\infty$ loop shaping design procedure (LSDP), has been developed [1, 2]. The LSDP method augments the plant with appropriately chosen weights so that the frequency response of the open loop system (the weighted plant) is reshaped in order to meet the closed-loop performance requirements.

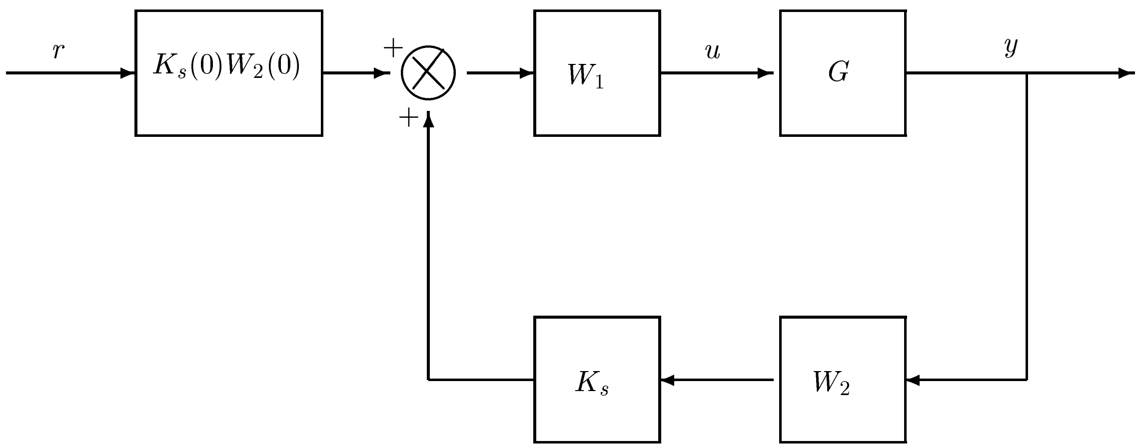

Figure 2: One-degree-of-freedom LSDP configuration.

This loop shaping can be done by the following design procedure.

(i) Using a pre-compensator, W_1, and/or a post-compensator, W_2, as depicted in Fig. 2, the singular values of the nominal system G are modified to give a desired loop shape. Usually, the least singular value of the weighted system should be made big over the low frequency range to achieve good performance such as the tracking, the largest singular value small over high frequency range to deal with unmodelled dynamics, and the bandwidth affects the system response speed while the slope of the singular values near the bandwidth frequency should not be too steep.

The nominal system and weighting functions W_1 and W_2 are combined to form the shaped system, G_s, where $G_s = W_2 G W_1$. It is assumed that W_1 and W_2 are such that G_s contains no hidden unstable modes.

(ii) A feedback controller, K_s, is synthesized which robustly stabilizes the normalized left coprime factorization of G_s, with a stability margin ϵ. It can be shown [2] that if ϵ is not less than 0.2, the frequency response of $K_s W_2 G W_1$ will be similar to that of $W_2 G W_1$. Of course, on the other hand, if the achievable ϵ is too big, that would probably indicate an over-designed case in respect of the robustness, which means that the performance of the system may possibly be improved by using a larger γ in computing K_s.

(iii) The final feedback controller, K_{final}, is then constructed by combining the $\mathcal{H}_\infty$ controller K_s, with the weighting functions W_1 and W_2 such that

$$K_{final} = W_1 K_s W_2.$$

For a tracking problem, the reference signal is generally fed between K_s and W_1, so that the closed loop transfer function between the reference r and the plant output y becomes

$$Y(s) = [I - G(s)K_{final}(s)]^{-1} G(s) W_1(s) K_s(0) W_2(0) R(s)$$

where the reference r is connected through a gain $K_s(0)W_2(0)$ where

$$K_s(0)W_2(0) \;\; = \;\; \lim_{s \to 0} K_s(s) W_2(s)$$

The above design procedure can be developed further into a two-degree-of-freedom (2-DOF) scheme as shown in Fig. 3.

The philosophy of the 2-DOF scheme is to use the feedback controller $K_2(s)$ to meet the requirements of internal and robust stability, disturbance rejection, measurement noise attenuation, and sensitivity minimisation. The pre-compensator K_1 is applied to the reference signal, which optimizes the response of the overall system to the command input such that the output of the system would be 'near'to that of a chosen ideal system M_o. The feedforward compensator K_1 depends on design objectives and can be synthesized together with the feedback controller in a single step via the $\mathcal{H}_\infty$ LSDP [4].

3 Numerical algorithms and software

The loop shaping design of a positive feedback controller for a continuous-time system is accomplished by the algorithm *LSDP* as listed below.

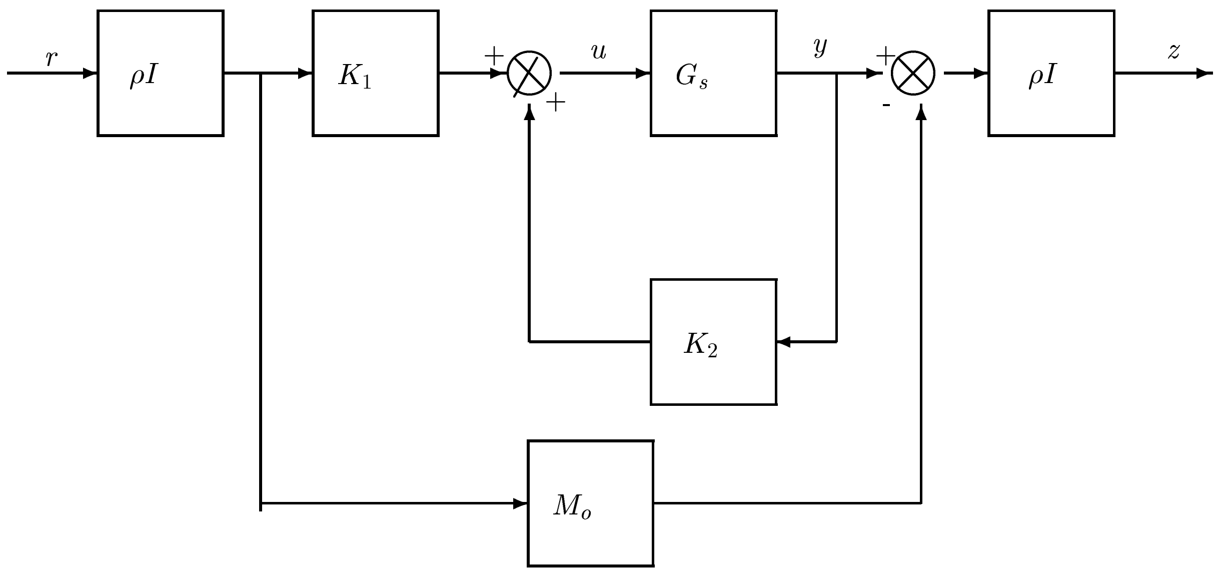

Figure 3: Two-degree-of-freedom design configuration.

For given matrices $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times n}$, $D \in R^{p\times m}$ in the state-space description of the shaped system, algorithm *LSDP* computes the matrices A_k, B_k, C_k, D_k of the feedback controller K. The conversion from a descriptor state-space system into a regular state-space form is done by the algorithm *DES2SS*.

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

Compute $R = I_p + DD^T$

Compute $S = I_m + D^TD$

Compute $A_X = A - BS^{-1}D^TC$

Compute $C_X = C^TR^{-1}C$

Compute $D_X = BS^{-1}B^T$

Solve $A_X^TX + XA_X + C_X - XD_XX = 0$

Set $A_Z = A_X,\ \ C_Z = D_X,\ \ D_Z = C_X$

Solve $A_ZZ + ZA_Z^T + C_Z - ZD_ZZ = 0$

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

Compute $F_1 = -(S^{-1}D^TC + S^{-1}B^TX)$

Compute $A_c = A + BF_1$

Compute $W_1 = (1 - \gamma_o^2)I_n + XZ$

Compute $B_{cp} = \gamma_o^2 ZC^T$

Compute $C_{cp} = B^TX$

Compute $A_{cp} = W_1^T A_c + \gamma_o^2 ZC^T(C + DF_1)$

Compute $D_{cp} = -D^T$

Compute the controller matrices A_k, B_k, C_k, D_k from matrices $A_{cp}, B_{cp}, C_{cp}, D_{cp}$ by using the algorithm *DES2SS*

Check the stability of the closed-loop state matrix

$$\begin{bmatrix} A + B(I_m - D_kD)^{-1}D_kC & B(I_m - D_kD)^{-1}C_k \\ B_k(I_p - DD_k)^{-1}C & A_k + B_k(I_p - DD_k)^{-1}DC_k \end{bmatrix}$$

■

The algorithm *LSDP* is implemented by the double precision Fortran 77 subroutine SB10ID. The subroutine produces the suboptimal controller matrices along with estimates of the condition numbers of the X- and Z-Riccati equations which are to be solved. These condition numbers give information about the accuracy of the computations in obtaining the controller. The matrix Riccati equations are solved by the subroutine SB02RD from SLICOT which produces also estimates of the condition numbers of these equations. Listing of the subroutine SB10ID is given in the Appendix.

The conversion of the descriptor state-space system

$$\begin{aligned} E\dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

into the regular state-space system

$$\begin{aligned} \dot{x} &= A_dx + B_du \\ y &= C_dx + D_du \end{aligned}$$

is done by the algorithm *DES2SS*. For given matrices $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times n}$, $D \in R^{p\times m}$, $E \in R^{n\times n}$, this algorithm produces the order $nsys$ of the regular state-space system and the matrices $A_d \in R^{nsys \times nsys}$, $B_d \in R^{nsys \times m}$, $C_d \in R^{p \times nsys}$, $D_d \in R^{p \times m}$ of this system. In the description of the algorithm we denote by the superscript $^+$ the pseudoinverse of the corresponding matrix.

Algorithm *DES2SS: Converting descriptor state-space system into regular state-space form*

Compute the SVD of E : $E = U_E S_E V_E^T$

Determine the rank n_1 of E (the nonzero singular values of E)

If $n_1 > 0$, then

Transform $E \leftarrow U_E^T E V_E = S$, $A \leftarrow U_E^T A V_E$, $B \leftarrow U_E^T B$, $C \leftarrow C V_E$

Set $A := \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ where $A_{11} \in R^{n_1 \times n_1}$

Set $B := \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$ where $B_1 \in R^{n_1 \times m}$

Set $C := \begin{bmatrix} C_1 & C_2 \end{bmatrix}$ where $C_1 \in R^{p \times n_1}$

Set $k = n - n_1$ (the dimension of the null-space of E)

If $k > 0$ then

Compute the SVD of A_{22} : $A_{22} = U_A S_A V_A^T$

Transform $A_{12} \leftarrow A_{12} V_A$

Transform $A_{21} \leftarrow U_A^T A_{21}$

Transform $B_2 \leftarrow U_A^T B_2$

Transform $C_2 \leftarrow C_2 V_A$

Compute $A_d = A_{11} - A_{12} S_A^+ A_{21}$

Compute $B_d = B_1 - A_{12} S_A^+ B2$

Compute $C_d = C_1 - C_2 S_A^+ A_{21}$

Compute $D_d = D - C_2 S_A^+ B_2$

Else

Set $A_d = A_1$, $B_d = B_1$, $C_d = C_1$, $D_d = D$

End if

Compute $S_2 = S_E(1 : n_1, 1 : n_1)^{\frac{1}{2}}$

Transform $A_d \leftarrow S_2 A_d S_2, \quad B_d \leftarrow S_2 B_d, \quad C_d \leftarrow C_d S_2$

Set $nsys = n_1$

Else

Set $A_d = -10^{15} I_n, \quad B_d = 0_{n\times m}, \quad C_d = 0_{p\times n}$

Set $nsys = n$

End if ■

The algorithm *DES2SS* is implemented by the double precision Fortran 77 subroutine SB10JD. Listing of this subroutine is given in the Appendix.

4 An illustrative example

Consider, as an example, the LSDP to be applied to the IFAC 93 benchmark example problem [7]. The nominal transfer function $G(s)$ of the plant is given by

$$G(s) = \frac{K(-T_2 s + 1)\omega_0^2}{(s^2 + 2\xi\omega_0 s + \omega_0^2)(T_1 s + 1)}$$

where $T_1 = 5, \; T_2 = 0.4, \; \omega_0 = 5, \; \xi = 0.3, \; K = 1.$

The frequensy response characteristics of the unweighted plant is given in Fig. 4.

There are 3 stress levels representing the parameter variations in the benchmark problem. In this exercise, only the stress level 2 is to be considered, which is given in the following table.

δT_1	δT_2	$\delta \omega_0$	$\delta \xi$	δK
± 0.30	± 0.10	± 2.50	± 0.15	± 0.15

The complete transfer function $G_t(s)$ of the plant which includes the neglected dynamics is given by

$$G(s) = \frac{K(-T_2 s + 1)\omega_0^2}{(s^2 + 2\xi\omega_0 s + \omega_0^2)(T_1 s + 1)} \frac{\omega_\delta^2}{(s^2 + 2\xi_\delta\omega_\delta + \omega_\delta^2)(T_1^\delta s + 1)(T_2^\delta s + 1)}$$

where $T_1^\delta = \frac{1}{8}, \; T_2^\delta = \frac{1}{12}, \omega_\delta = 15, \xi_\delta = 0.6.$

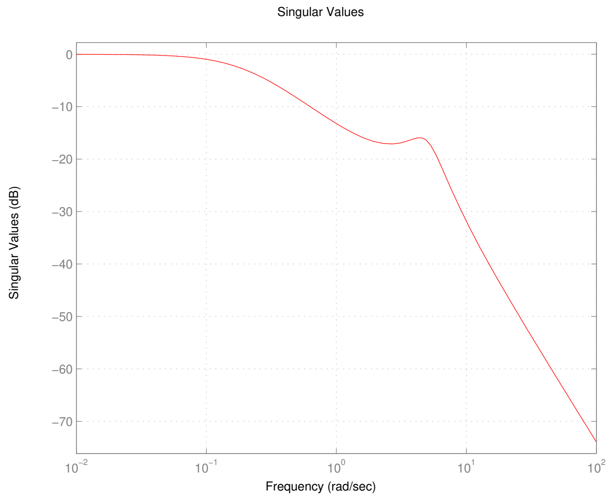

Figure 4: Frequensy response characteristics of the plant

The designed controller should make the closed-loop system stable. In response to a unit step signal, the output of the full order plant must remain between -1.5 and $+1.5$ at all time and between -1.2 and $+1.2$ most of the time. The system should achieve zero steady-state tracking error, fast settling time and under/overshoot below 0.2 most of the time. All these design specifications should be satisfied over the parameter variation ranges.

In this design, the weighting functions are chosen as [7]

$$W_1(s) = 1.941 + \frac{1.461}{s}, \quad W_2 = 1.$$

The frequency response characteristics of the shaped plant is shown in Fig. 5.

Using the subroutines SB10ID and SB10JD, described in the previous section, a controller

$$K(s) = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]$$

was obtained where (up to four digits)

$$A_k = \begin{bmatrix} -1.6474 & -0.5568 & 1.8710 \\ 4.5365 & -2.6773 & -5.0618 \\ -2.9723 & 5.2186 & -0.3896 \end{bmatrix},$$

$$B_k = \begin{bmatrix} 3.6218 \\ -2.4951 \\ -1.5443 \end{bmatrix},$$

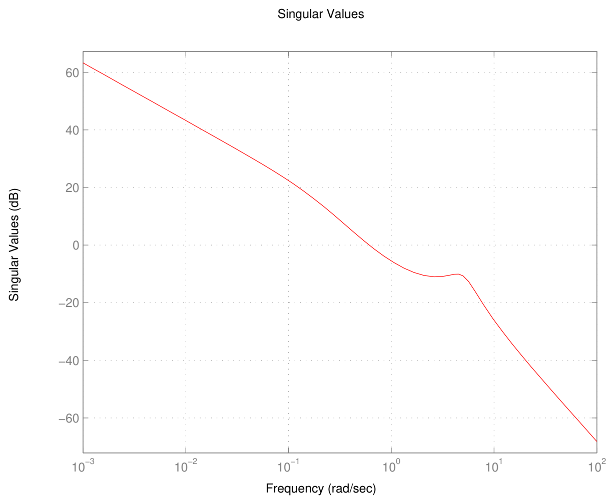

Figure 5: Frequensy response of the shaped plant

$$C_k = \begin{bmatrix} 0.5545 & 0.1520 & -0.5840 \end{bmatrix},$$

$$D_k = -1.7610.$$

The transfer function of the controller is

$$K(s) = \frac{-1.761s^3 - 5.771s^2 - 45.460s - 14.306}{s^3 + 4.714s^2 + 40.598s + 25.192}$$

It is noticed that the order of the controller is 3, reduced by one from the order of the augmented (shaped) plant. The reason is that the optimal value of γ, γ_o in (2.10), is used in the construction of the controller. In this design, $\gamma_o = 2.0251$. In practice, the complete controller would include the weight W_1, hence a 4^{th} order controller. The condition numbers of solutions to the two Riccati equations are both reasonable, which gives the confidence with the designed controller.

The closed-loop time responses of the perturbed, full order plant, obtained for the representative values of δT_1, δT_2, $\delta\omega_0$, $\delta\xi$, δK, are shown in Fig. 6. It can be seen that the design specifications have been met.

The optimal γ_o is used to compute the controller. That would better test the numerical property of the subroutines. In most designs, however, a sub-optimal controller would be considered. Therefore, a larger value of γ is used instead. For instance, $\gamma = 4$ in this example. The

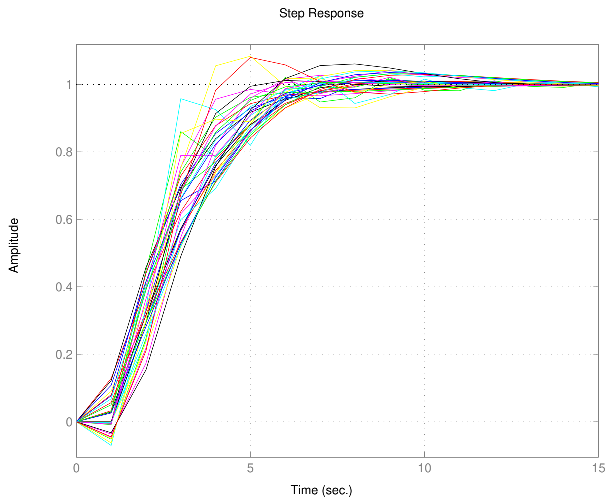

Figure 6: Transient responses for the perturbed plant: γ_o case

corresponding controller is of order 4, and has a state-space realization as the following.

$$A_K = \begin{bmatrix} -1.7067 & 0.2247 & 1.3214 & -1.6059 \\ -4.5099 & -2.9729 & 4.2691 & -2.8759 \\ -3.3219 & -5.1050 & -0.0219 & -0.7801 \\ 0.7408 & -0.4013 & -1.1365 & -0.3007 \end{bmatrix}$$

$$B_K = \begin{bmatrix} -2.5473 \\ 0.7725 \\ 0.9386 \\ 4.6836 \end{bmatrix}$$

$$C_K = \begin{bmatrix} -0.1933 & -0.0202 & 0.0305 & -0.3831 \end{bmatrix}$$

$$D_K = 0$$

$$K(s) = \frac{-1.289s^3 - 4.285s^2 - 33.773s - 10.032}{s^4 + 5.002s^3 + 32.936s^2 + 55.855s + 26.629}$$

5 Conclusions

The $\mathcal{H}_\infty$ loop shaping design procedure and its implementation in *SLICOT* has been summarised in this report. In our design experience in applying $\mathcal{H}_\infty$ and its related design methods, the LSDP has shown to be a more effective and more successful method. Hence, it is included in the Robust control Toolbox of *SLICOT*. With the improved numerical routines, we hope the $\mathcal{H}_\infty$ LSDP would be adopted by practising control engineers in their control systems design and provide better and more reliable results.

References

[1] D.C. MaFarlane and K. Glover. *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*. In Lecture Notes in Control and Information Sciences, **vol.136**. Springer-Verlag, Berlin, 1990.

[2] D.C. MaFarlane and K. Glover. "A loop shaping design procedure using $\mathcal{H}_\infty$ synthesis." *IEEE Trans. Automat. Contr.*, **vol.37**, pp. 749-769, 1992

[3] K. Glover. "All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds." *International J. of Control*, **vol.39**, pp. 1115-1193, 1984.

[4] D.J. Hoyle, R.A. Hyde and D.J.N. Limebeer. "An $\mathcal{H}_\infty$ approach to two degree of freedom design." *Proceedings of the 30th IEEE Conference on Decision and Control*, pp. 1579-1580, Brighton. U.K., 1991.

[5] M.G. Safonov, R.Y. Chiang and D.J.N. Limebeer. "Hankel model reduction without balancing – a descriptor approach." *Proceedings of the 26th IEEE Conference on Decision and Control*, pp. 112-117, Los Angeles, California, 1987.

[6] M.G. Safonov, D.J.N. Limebeer and R.Y. Chiang. "Simplifying the $\mathcal{H}_\infty$ theory via loop-shifting, matrix- pencil and descriptor concepts." *International J. of Control*, **vol.50**, pp. 2467-2488, 1989.

[7] J.F. Whidborne, G. Murad, D.-W. Gu and I. Postlethwaite. "Robust control of an unknown plant – the IFAC 93 benchmark." *Int. J. Control*, **v-61, No.3**, pp. 589 - 640, 1995.

Appendix. Fortran 77 subroutines for Loop Shaping Design of continuous-time systems

SB10ID - Driver subroutine for computation of the positive feedback controller

SB10JD - Subroutine to convert a descriptor state-space system into regular state space form

```
      SUBROUTINE SB10ID( N, M, NP, A, LDA, B, LDB, C, LDC, D, LDD,
     $                   NK, AK, LDAK, BK, LDBK, CK, LDCK, DK, LDDK,
     $                   RCOND, IWORK, WORK, LWORK, BWORK, INFO )
C
C     RELEASE 3.0, WGS COPYRIGHT 1999.
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 | D |
C
C     in the McFarlane/Glover 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     D       (input) DOUBLE PRECISION array, dimension (LDD,M)
C             The NP-by-M system matrix D of the shaped plant.
C
C     LDD     INTEGER
C             The leading dimension of the array D. LDD >= max(1,NP).
C
C     NK      (output) INTEGER
C             The order of the positive feedback controller. NK <= N.
C
C     AK      (output) DOUBLE PRECISION array, dimension (LDAK,N)
C             The NK-by-NK 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 NK-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-NK 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 X-Riccati equation
C             RCOND(2) contains an estimate of the reciprocal condition
C                      number of the Z-Riccati equation
C
C     Workspace
C
C     IWORK   INTEGER array, dimension max(2*N,N*N)
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 >= 12*N*N + M*M + NP*NP + 3*M*N + M*NP + 3*N*NP +
C                      4*N + 5 + max(M,4*N*N+8*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 X-Riccati equation is not solved successfuly
C           = 2: The Z-Riccati equation is not solved successfuly
C           = 3: The iteration to compute eigenvalues or singular
C                values failed to converge
C           = 4: The matrix Ip - D*Dk is singular
C           = 5: The matrix Im - Dk*D is singular
C           = 6: The closed-loop system is unstable
C
C     METHOD
C
C     The routine implements the formulas, given in [1].
C
C     REFERENCES
C
C     [1] D. McFarlane and K. Glover. A loop shaping design procedure
C         using H_infinity synthesis. IEEE Trans. Automat. Control,
C         vol. AC-37, no. 6, pp. 759-769, 1992.
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 (see the
C     output parameter RCOND).
C
C     CONTRIBUTORS
C
C     P.Hr. Petkov, D.W. Gu and M.M. Konstantinov, October 1999
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, LDD, LDAK, LDBK, LDCK,
     $                   LDDK, LWORK, M, N, NK, NP
C     ..
C     .. Array Arguments ..
      INTEGER            IWORK( * )
      LOGICAL            BWORK( * )
      DOUBLE PRECISION   A( LDA, * ), B( LDB, * ), C( LDC, * ),
     $                   D( LDD, * ), AK( LDAK, * ), BK( LDBK, * ),
     $                   CK( LDCK, * ), DK( LDDK, * ), RCOND( 2 ),
     $                   WORK( * )
C     ..
C     .. Local Scalars ..
      CHARACTER*1        HINV
      LOGICAL            SELECT
      INTEGER            I1, I2, I3, I4, I5, I6, I7, I8, I9, I10, I11,
     $                   I12, I13, I14, INFO2, IWRK, LWA, LWAMAX,
     $                   MINWRK, N2, NS, SDIM
      DOUBLE PRECISION   SEP, FERR, GAMMA
C     ..
C     .. External Subroutines ..
      EXTERNAL           DGEMM, DGEES, DGETRF, DGETRI, DLACPY, DLASET,
     $                   DPOTRF, DPOTRS, DSYRK, SB10JD, 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( LDD.LT.MAX( 1, NP ) ) THEN
         INFO = -11
      ELSE IF( LDAK.LT.MAX( 1, N ) ) THEN
         INFO = -14
      ELSE IF( LDBK.LT.MAX( 1, N ) ) THEN
         INFO = -16
      ELSE IF( LDCK.LT.MAX( 1, M ) ) THEN
         INFO = -18
      ELSE IF( LDDK.LT.MAX( 1, M ) ) THEN
         INFO = -20
      END IF
C
C     Compute workspace.
C
      MINWRK = 12*N*N + M*M + NP*NP + 3*M*N + M*NP + 3*N*NP + 4*N + 5 +
     $         MAX( M, 4*N*N+8*N )
      IF( LWORK.LT.MINWRK ) THEN
         INFO = -24
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'SB10ID', -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
C
      I1 = N*N
      I2 = I1 + N*N
```

```
      I3 = I2 + M*N
      I4 = I3 + M*N
      I5 = I4 + M*M
      I6 = I5 + NP*NP
      I7 = I6 + NP*N
      I8 = I7 + N*N
      I9 = I8 + N*N
      I10 = I9 + N*N
      I11 = I10 + N*N
      I12 = I11 + N*N
      I13 = I12 + 2*N
      I14 = I13 + 2*N
      IWRK = I14 + 4*N*N
      LWAMAX = 0
C
C     Compute D'*C .
C
      CALL DGEMM( 'T', 'N', M, N, NP, ONE, D, LDD, C, LDC, ZERO,
     $            WORK( I2+1 ), M )
C
C     Compute S = Im + D'*D .
C
      CALL DLASET( 'U', M, M, ZERO, ONE, WORK( I4+1 ), M )
      CALL DSYRK( 'U', 'T', M, NP, ONE, D, LDD, ONE, WORK( I4+1 ), M )
C
C     Factorize S .
C
      CALL DPOTRF( 'U', M, WORK( I4+1 ), M, INFO2 )
C
C             -1
C     Compute S  D'*C
C
      CALL DPOTRS( 'U', M, N, WORK( I4+1 ), M, WORK( I2+1 ), M, INFO2 )
C
C             -1
C     Compute S  B'
C
      DO 20 I = 1, M
```

```
         DO 10 J = 1, N
            WORK( I3+(J-1)*M+I ) = B( J, I )
   10    CONTINUE
   20 CONTINUE
      CALL DPOTRS( 'U', M, N, WORK( I4+1 ), M, WORK( I3+1 ), M, INFO2 )
C
C     Compute R = Ip + D*D' .
C
      CALL DLASET( 'U', NP, NP, ZERO, ONE, WORK( I5+1 ), NP )
      CALL DSYRK( 'U', 'N', NP, M, ONE, D, LDD, ONE, WORK( I5+1 ), NP )
C
C     Factorize R .
C
      CALL DPOTRF( 'U', NP, WORK( I5+1 ), NP, INFO2 )
C
C               -1
C     Compute R  C .
C
      CALL DLACPY( 'F', NP, N, C, LDC, WORK( I6+1 ), NP )
      CALL DPOTRS( 'U', NP, N, WORK( I5+1 ), NP, WORK( I6+1 ), NP,
     $             INFO2 )
C
C                          -1
C     Compute Ar = A - B*S  D'*C .
C
      CALL DLACPY( 'F', N, N, A, LDA, WORK( I7+1 ), N )
      CALL DGEMM( 'N', 'N', N, N, M, -ONE, B, LDB, WORK( I2+1 ), M, ONE,
     $            WORK( I7+1 ), N )
C
C                       -1
C     Compute Cr = C'*R  *C
C
      CALL DGEMM( 'T', 'N', N, N, NP, ONE, C, LDC, WORK( I6+1 ), NP,
     $            ZERO, WORK( I8+1 ), N )
C
C                      -1
C     Compute Dr = B*S  B'
C
```

```
      CALL DGEMM( 'N', 'N', N, N, M, ONE, B, LDB, WORK( I3+1 ), M, ZERO,
     $            WORK( I9+1 ), N )
C
C     Solution of the Riccati equation Ar'*X + X*Ar + Cr - X*Dr*X = 0
C
      N2 = 2*N
      CALL SB02RD( 'A', 'C', HINV, 'N', 'U', 'G', 'S', 'N', 'O', N,
     $             WORK( I7+1 ), N, WORK( I10+1 ), N, WORK( I11+1 ), N,
     $             WORK( I9+1 ), N, WORK( I8+1 ), N, WORK, N, SEP,
     $             RCOND( 1 ), FERR, WORK( I12+1 ), WORK( I13+1 ),
     $             WORK( I14+1 ), N2, 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     Solution of the Riccati equation Ar*Z + Z*Ar' + Dr - Z*Cr*Z = 0
C
      CALL SB02RD( 'A', 'C', HINV, 'T', 'U', 'G', 'S', 'N', 'O', N,
     $             WORK( I7+1 ), N, WORK( I10+1 ), N, WORK( I11+1 ), N,
     $             WORK( I8+1 ), N, WORK( I9+1 ), N, WORK( I1+1 ), N,
     $             SEP, RCOND( 2 ), FERR, WORK( I12+1 ), WORK( I13+1 ),
     $             WORK( I14+1 ), N2, 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                       -1        -1
C     Compute F1 = -( S  D'*C + S  B'*X )
C
      CALL DGEMM( 'N', 'N', M, N, N, -ONE, WORK( I3+1 ), M, WORK, N,
```

```
     $            -ONE, WORK( I2+1 ), M )
C
C     Compute gamma
C
      CALL DGEMM( 'N', 'N', N, N, N, ONE, WORK, N, WORK( I1+1 ), N,
     $            ZERO, WORK( I7+1 ), N )
      CALL DGEES( 'N', 'N', SELECT, N, WORK( I7+1 ), N, SDIM,
     $            WORK( I12+1 ), WORK( I13+1 ), WORK( IWRK+1 ), N,
     $            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 )
      GAMMA = ZERO
      DO 30 I = 1, N
         GAMMA = MAX( GAMMA, WORK( I12+I ) )
   30 CONTINUE
      GAMMA = SQRT( ONE + GAMMA )
C
C     Workspace usage
C
      I4 = I3 + N*N
      I5 = I4 + N*N
      I6 = I5 + N*NP
      I7 = I6 + NP*N
      I8 = I7 + N*N
      IWRK = I8 + M*NP
C
C     Compute Ac = A + B*F1
C
      CALL DLACPY( 'F', N, N, A, LDA, WORK( I3+1 ), N )
      CALL DGEMM( 'N', 'N', N, N, M, ONE, B, LDB, WORK( I2+1 ), M,
     $            ONE, WORK( I3+1 ), N )
C
C     Compute W1' = (1-gamma^2)*In + Z*X
C
```

```
      CALL DLASET( 'F', N, N, ZERO, ONE-GAMMA*GAMMA, WORK( I4+1 ), N )
      CALL DGEMM( 'N', 'N', N, N, N, ONE, WORK( I1+1 ), N, WORK, N,
     $            ONE, WORK( I4+1 ), N )
C
C     Compute Bcp = gamma^2*Z*C'
C
      CALL DGEMM( 'N', 'T', N, NP, N, GAMMA*GAMMA, WORK( I1+1 ), N, C,
     $            LDC, ZERO, WORK( I5+1 ), N )
C
C     Compute C + D*F1
C
      CALL DLACPY( 'F', NP, N, C, LDC, WORK( I6+1 ), NP )
      CALL DGEMM( 'N', 'N', NP, N, M, ONE, D, LDD, WORK( I2+1 ), M,
     $            ONE, WORK( I6+1 ), NP )
C
C     Compute Acp = W1'*Ac + gamma^2*Z*C'*(C+D*F1)
C
      CALL DGEMM( 'N', 'N', N, N, N, ONE, WORK( I4+1 ), N, WORK( I3+1 ),
     $            N, ZERO, WORK( I7+1 ), N )
      CALL DGEMM( 'N', 'N', N, N, NP, ONE, WORK( I5+1 ), N,
     $            WORK( I6+1 ), NP, ONE, WORK( I7+1 ), N )
C
C     Compute Ccp = B'*X
C
      CALL DGEMM( 'T', 'N', M, N, N, ONE, B, LDB, WORK, N, ZERO,
     $             WORK( I2+1 ), M )
C
C     Set Dcp = -D'
C
      DO 50 I = 1, M
         DO 40 J = 1, NP
            WORK( I8+(J-1)*M+I ) = -D( J, I )
   40    CONTINUE
   50 CONTINUE
C
C     Reduce the generalized state-space description to regular one
C
      CALL SB10JD( N, NP, M, WORK( I7+1 ), N, WORK( I5+1 ), N,
```

```
     $             WORK( I2+1 ), M, WORK( I8+1 ), M, WORK( I4+1 ), N,
     $             NK, AK, LDAK, BK, LDBK, CK, LDCK, DK, LDDK,
     $             WORK( IWRK+1 ), LWORK-IWRK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 3
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Workspace usage
C
      I2 = NP*NP
      I3 = I2 + NK*NP
      I4 = I3 + M*M
      I5 = I4 + N*M
      I6 = I5 + NP*NK
      I7 = I6 + M*N
      I8 = I7 + ( N + NK )*( N + NK )
      I9 = I8 + N + NK
      IWRK = I9 + N + NK
C
C     Compute Ip - D*Dk
C
      CALL DLASET( 'Full', NP, NP, ZERO, ONE, WORK, NP )
      CALL DGEMM( 'N', 'N', NP, NP, M, -ONE, D, LDD, DK, LDDK, ONE,
     $             WORK, NP )
C
C                          -1
C     Compute Bk*(Ip-D*Dk)
C
      CALL DGETRF( NP, NP, WORK, NP, IWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 4
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
```

```
      CALL DGETRI( NP, WORK, NP, IWORK, WORK( IWRK+1 ), LWORK-IWRK,
     $             INFO2 )
      CALL DGEMM( 'N', 'N', NK, NP, NP, ONE, BK, LDBK, WORK, NP, ZERO,
     $             WORK( I2+1 ), NK )
C
C     Compute Im - Dk*D
C
      CALL DLASET( 'Full', M, M, ZERO, ONE, WORK( I3+1 ), M )
      CALL DGEMM( 'N', 'N', M, M, NP, -ONE, DK, LDDK, D, LDD, ONE,
     $             WORK( I3+1 ), M )
C
C                          -1
C     Compute B*(Im-Dk*D)
C
      CALL DGETRF( M, M, WORK( I3+1 ), M, IWORK, INFO2 )
      IF( INFO2.NE.0 ) THEN
         INFO = 5
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
      LWAMAX = MAX( LWA, LWAMAX )
      CALL DGETRI( M, WORK( I3+1 ), M, IWORK, WORK( IWRK+1 ),
     $             LWORK-IWRK, INFO2 )
      CALL DGEMM( 'N', 'N', N, M, M, ONE, B, LDB, WORK( I3+1 ), M, ZERO,
     $             WORK( I4+1 ), N )
C
C     Compute D*Ck
C
      CALL DGEMM( 'N', 'N', NP, NK, M, ONE, D, LDD, CK, LDCK, ZERO,
     $             WORK( I5+1 ), NP )
C
C     Compute Dk*C
C
      CALL DGEMM( 'N', 'N', M, N, NP, ONE, DK, LDDK, C, LDC, ZERO,
     $             WORK( I6+1 ), M )
C
C     Compute the closed-loop state matrix
C
```

```
      CALL DLACPY( 'F', N, N, A, LDA, WORK( I7+1 ), N+NK )
      CALL DGEMM( 'N', 'N', N, N, M, ONE, WORK( I4+1 ), N, WORK( I6+1 ),
     $            M, ONE, WORK( I7+1 ), N+NK )
      CALL DGEMM( 'N', 'N', N, NK, M, ONE, WORK( I4+1 ), N, CK, LDCK,
     $            ZERO, WORK( I7+(N+NK)*N+1 ), N+NK )
      CALL DGEMM( 'N', 'N', NK, N, NP, ONE, WORK( I2+1 ), NK, C, LDC,
     $            ZERO, WORK( I7+N+1 ), N+NK )
      CALL DLACPY( 'F', NK, NK, AK, LDAK, WORK( I7+(N+NK)*N+N+1 ),
     $             N+NK )
      CALL DGEMM( 'N', 'N', NK, NK, NP, ONE, WORK( I2+1 ), NK,
     $            WORK( I5+1 ), NP, ONE, WORK( I7+(N+NK)*N+N+1 ), N+NK )
C
C     Compute the closed-loop poles
C
      CALL DGEES( 'N', 'N', SELECT, N+NK, WORK( I7+1 ), N+NK, SDIM,
     $            WORK( I8+1 ), WORK( I9+1 ), WORK( IWRK+1 ), N,
     $            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     Check the stability of the closed-loop system
C
      NS = 0
      DO 60 I = 1, N+NK
         IF( WORK( I8+I ).GE.ZERO ) NS = NS + 1
   60 CONTINUE
      IF( NS.GT.0 ) THEN
         INFO = 6
         RETURN
      END IF
C
      LWA = 11*N*N + M*M + NP*NP + 2*M*N + M*NP + 2*N*NP + 4*N + LWAMAX
      WORK( 1 ) = DBLE( LWA )
      RETURN
```

```
C *** Last line of SB10ID ***
      END
```

```
      SUBROUTINE SB10JD( N, M, NP, A, LDA, B, LDB, C, LDC, D, LDD, E,
     $                   LDE, NSYS, AD, LDAD, BD, LDBD, CD, LDCD, DD,
     $                   LDDD, WORK, LWORK, INFO )
C
C     RELEASE 3.0, WGS COPYRIGHT 1999.
C
C     PURPOSE
C
C     To convert the descriptor state-space system
C
C     Edx/dt = Ax + Bu
C          y = Cx + Du
C
C     into regular state-space form
C
C      dx/dt = Adx + Bdu
C          y = Cdx + Ddu .
C
C     ARGUMENTS
C
C     Input/Output Parameters
C
C     N       (input) INTEGER
C             The order of the descriptor system. 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/output) DOUBLE PRECISION array, dimension (LDA,N)
C             The N-by-N state matrix A of the descriptor system.
C
C     LDA     INTEGER
C             The leading dimension of the array A. LDA >= max(1,N).
C
C     B       (input/output) DOUBLE PRECISION array, dimension (LDB,M)
```

```
C             The N-by-M input matrix B of the descriptor system.
C
C     LDB     INTEGER
C             The leading dimension of the array B. LDB >= max(1,N).
C
C     C       (input/output) DOUBLE PRECISION array, dimension (LDC,N)
C             The NP-by-N output matrix C of the descriptor system.
C
C     LDC     INTEGER
C             The leading dimension of the array C. LDC >= max(1,NP).
C
C     D       (input/output) DOUBLE PRECISION array, dimension (LDD,M)
C             The NP-by-M matrix D of the descriptor system.
C
C     LDD     INTEGER
C             The leading dimension of the array D. LDD >= max(1,NP).
C
C     E       (input/output) DOUBLE PRECISION array, dimension (LDE,N)
C             The N-by-N matrix E of the descriptor system.
C
C     LDE     INTEGER
C             The leading dimension of the array E. LDE >= max(1,N).
C
C     NSYS    (output) INTEGER
C             The order of the converted state-space system.
C
C     AD      (output) DOUBLE PRECISION array, dimension (LDAD,N)
C             The NSYS-by-NSYS state matrix AD of the converted system.
C
C     LDAD    INTEGER
C             The leading dimension of the array AD. LDAD >= max(1,N).
C
C     BD      (output) DOUBLE PRECISION array, dimension (LDBD,M)
C             The NSYS-by-M input matrix BD of the converted system.
C
C     LDBD    INTEGER
C             The leading dimension of the array BD. LDBD >= max(1,N).
C
```

```
C     CD      (output) DOUBLE PRECISION array, dimension (LDCD,NSYS)
C             The NP-by-NSYS output matrix CD of the converted system.
C
C     LDCD    INTEGER
C             The leading dimension of the array CD. LDCD >= max(1,NP).
C
C     DD      (output) DOUBLE PRECISION array, dimension (LDDD,M)
C             The NP-by-M matrix DD of the converted system.
C
C     LDDD    INTEGER
C             The leading dimension of the array DD. LDDD >= max(1,NP).
C
C     Workspace
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 >= 7*N*N + 2*M*N + 2*NP*N + N + max(4*N,5*N-4).
C             For good performance, LWORK must generally be larger.
C
C     Error Indicator
C
C     INFO    INTEGER
C             = 0: successful exit
C             < 0: if INFO = -i, the i-th argument had an illegal
C                  value
C             = 1: the iteration for computing singular value
C                  decomposition did not converge
C
C     METHOD
C
C     The routine performs the transformations described in [1].
C
C     REFERENCES
C
C     [1] R.Y. Chiang and M.G. Safonov. Robust Control Toolbox User's,
```

```
C          Guide, The MathWorks Inc., Natick, Mass., 1992.
C
C      CONTRIBUTORS
C
C      P.Hr. Petkov, D.W. Gu and M.M. Konstantinov, October 1999
C
C      KEYWORDS
C
C      Descriptor systems, state-space models
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, LDAD, LDB, LDBD, LDC, LDCD, LDD,
      $                   LDDD, LWORK, M, N, NSYS, NP
C      ..
C      .. Array Arguments ..
       DOUBLE PRECISION   A( LDA, * ), B( LDB, * ), C( LDC, * ),
      $                   D( LDD, * ), E( LDE, * ), AD( LDAD, * ),
      $                   BD( LDBD, * ), CD( LDCD, * ), DD( LDDD, * ),
      $                   WORK( * )
C      ..
C      .. Local Scalars ..
       INTEGER            I, IA, IA12, IA21, IA22, IB, IB2, IC, IC2,
      $                   INFO2, IS, ISA, IU, IV, IWRK, J, K, LWA,
      $                   LWAMAX, MINWRK, NS1
       DOUBLE PRECISION   EPS, SCALE, TOL
C      ..
C      .. External Subroutines ..
       EXTERNAL           DGEMM, DGESVD, DLACPY, DLASET, DSCAL, XERBLA
C      ..
C      .. Intrinsic Functions ..
       INTRINSIC          DBLE, INT, MAX
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( LDD.LT.MAX( 1, NP ) ) THEN
         INFO = -11
      ELSE IF( LDE.LT.MAX( 1, N ) ) THEN
         INFO = -13
      ELSE IF( LDAD.LT.MAX( 1, N ) ) THEN
         INFO = -16
      ELSE IF( LDBD.LT.MAX( 1, N ) ) THEN
         INFO = -18
      ELSE IF( LDCD.LT.MAX( 1, NP ) ) THEN
         INFO = -20
      ELSE IF( LDDD.LT.MAX( 1, NP ) ) THEN
         INFO = -22
      END IF
C
C     Set tol.
C
      EPS = DLAMCH( 'Epsilon' )
      TOL = SQRT( EPS )
C
C     Compute workspace.
C
```

```
      MINWRK = 7*N*N + 2*M*N + 2*NP*N + N + MAX( 4*N, 5*N-4 )
      IF( LWORK.LT.MINWRK ) THEN
         INFO = -24
      END IF
      IF( INFO.NE.0 ) THEN
         CALL XERBLA( 'SB10JD', -INFO )
         RETURN
      END IF
C
C     Quick return if possible.
C
      IF( N.EQ.0 ) THEN
         CALL DLACPY( 'Full', NP, M, D, LDD, DD, LDDD )
         RETURN
      END IF
C
C     Workspace usage.
C
      IA = N*N
      IB = IA + N*N
      IC = IB + N*M
      IS = IC + NP*N
      IU = IS + N
      IV = IU + N*N
      IWRK = IV + N*N
      LWAMAX = 0
C
C     Compute the SVD of E.
C
      CALL DLACPY( 'Full', N, N, E, LDE, WORK, N )
      CALL DGESVD( 'S', 'S', N, N, WORK, N, WORK( IS+1 ), WORK( IU+1 ),
     $             N, WORK( IV+1 ), N, WORK( IWRK+1 ), LWORK-IWRK,
     $             INFO2)
      IF( INFO2.NE.0 ) THEN
         INFO = 1
         RETURN
      END IF
      LWA = INT( WORK( IWRK+1 ) )
```

```
      LWAMAX = MAX( LWA, LWAMAX )
C
C     Determine the rank of E.
C
      NS1 = 0
      DO 10 I = 1, N
         IF( WORK( IS+I ).GT.TOL ) NS1 = NS1 + 1
   10 CONTINUE
      IF( NS1.GT.0 ) THEN
C
C        Transform E.
C
         CALL DGEMM( 'T', 'N', N, N, N, ONE, WORK( IU+1 ), N, E, LDE,
     $               ZERO, WORK( IWRK+1 ), N )
         CALL DGEMM( 'N', 'T', N, N, N, ONE, WORK( IWRK+1 ), N,
     $               WORK( IV+1 ), N, ZERO, WORK, N )
C
C        Transform A.
C
         CALL DGEMM( 'T', 'N', N, N, N, ONE, WORK( IU+1 ), N, A, LDA,
     $               ZERO, WORK( IWRK+1 ), N )
         CALL DGEMM( 'N', 'T', N, N, N, ONE, WORK( IWRK+1 ), N,
     $               WORK( IV+1 ), N, ZERO, WORK( IA+1 ), N )
C
C        Transform B.
C
         CALL DGEMM( 'T', 'N', N, M, N, ONE, WORK( IU+1 ), N, B, LDB,
     $               ZERO, WORK( IB+1 ), N )
C
C        Transform C.
C
         CALL DGEMM( 'N', 'T', NP, N, N, ONE, C, LDC, WORK( IV+1 ), N,
     $               ZERO, WORK( IC+1 ), NP )
C
         K = N - NS1
         IF( K.GT.0 ) THEN
            IA12 = IS + N
            IA21 = IA12 + NS1*K
```

```
         IA22 = IA21 + K*NS1
         IB2 = IA22 + K*K
         IC2 = IB2 + K*M
         IU = IC2 + NP*K
         ISA = IU + K*K
         IV = ISA + K
         IWRK = IV + K*K
C
C        Compute the SVD of A22.
C
         CALL DLACPY( 'Full', K, K, WORK( IA+N*NS1+NS1+1 ), N,
     $                WORK( IA22+1 ), K )
         CALL DGESVD( 'S', 'S', K, K, WORK( IA22+1 ), K,
     $                WORK( ISA+1 ), WORK( IU+1 ), K, WORK( IV+1 ),
     $                K, WORK( IWRK+1 ), LWORK-IWRK, INFO2)
         IF( INFO2.NE.0 ) THEN
            INFO = 1
            RETURN
         END IF
         LWA = INT( WORK( IWRK+1 ) )
         LWAMAX = MAX( LWA, LWAMAX )
C
C        Compute the transformed A12.
C
         CALL DGEMM( 'N', 'T', NS1, K, K, ONE, WORK( IA+N*NS1+1 ), N,
     $               WORK( IV+1 ), K, ZERO, WORK( IA12+1 ), NS1 )
C
C        Compute the transformed A21.
C
         CALL DGEMM( 'T', 'N', K, NS1, K, ONE, WORK( IU+1 ), K,
     $               WORK( IA+NS1+1 ), N, ZERO, WORK( IA21+1 ), K )
C
C        Compute BB2.
C
         CALL DGEMM( 'T', 'N', K, M, K, ONE, WORK( IU+1 ), K,
     $               WORK( IB+NS1+1 ), N, ZERO, WORK( IB2+1 ), K )
C
C        Compute CC2.
```

```
C
            CALL DGEMM( 'N', 'T', NP, K, K, ONE, WORK( IC+NP*NS1+1 ),
     $                  NP, WORK( IV+1 ), K, ZERO, WORK( IC2+1 ), NP )
C
C           Compute A12*pinv(A22) and CC2*pinv(A22).
C
            DO 20 J = 1, K
               SCALE = ZERO
               IF( WORK( ISA+J ).GT.TOL ) SCALE = ONE/WORK( ISA+J )
               CALL DSCAL( NS1, SCALE, WORK( IA12+(J-1)*NS1+1 ), 1 )
               CALL DSCAL( NP, SCALE, WORK( IC2+(J-1)*NP+1 ), 1 )
   20       CONTINUE
C
C           Compute AD.
C
            CALL DLACPY( 'Full', NS1, NS1, WORK( IA+1 ), N, AD, LDAD )
            CALL DGEMM( 'N', 'N', NS1, NS1, K, -ONE, WORK( IA12+1 ),
     $                  NS1, WORK( IA21+1 ), K, ONE, AD, LDAD )
C
C           Compute BD.
C
            CALL DLACPY( 'Full', NS1, M, WORK( IB+1 ), N, BD, LDBD )
            CALL DGEMM( 'N', 'N', NS1, M, K, -ONE, WORK( IA12+1 ), NS1,
     $                  WORK( IB2+1 ), K, ONE, BD, LDBD )
C
C           Compute CD.
C
            CALL DLACPY( 'Full', NP, NS1, WORK( IC+1 ), NP, CD, LDCD )
            CALL DGEMM( 'N', 'N', NP, NS1, K, -ONE, WORK( IC2+1 ), NP,
     $                  WORK( IA21+1 ), K, ONE, CD, LDCD )
C
C           Compute DD.
C
            CALL DLACPY( 'Full', NP, M, D, LDD, DD, LDDD )
            CALL DGEMM( 'N', 'N', NP, M, K, -ONE, WORK( IC2+1 ), NP,
     $                  WORK( IB2+1 ), K, ONE, DD, LDDD )
         ELSE
            CALL DLACPY( 'Full', NS1, NS1, WORK( IA+1 ), N, AD, LDAD )
```

```
            CALL DLACPY( 'Full', NS1, M, WORK( IB+1 ), N, BD, LDBD )
            CALL DLACPY( 'Full', NP, NS1, WORK( IC+1 ), NP, CD, LDCD )
            CALL DLACPY( 'Full', NP, M, D, LDD, DD, LDDD )
         END IF
         DO 30 I = 1, NS1
            SCALE = ONE/SQRT( WORK( IS+I ) )
            CALL DSCAL( NS1, SCALE, AD( I, 1 ), LDAD )
            CALL DSCAL( M, SCALE, BD( I, 1 ), LDBD )
   30    CONTINUE
         DO 40 J = 1, NS1
            SCALE = ONE/SQRT( WORK( IS+J ) )
            CALL DSCAL( NS1, SCALE, AD( 1, J ), 1 )
            CALL DSCAL( NP, SCALE, CD( 1, J ), 1 )
   40    CONTINUE
         NSYS = NS1
      ELSE
         CALL DLASET( 'F', N, N, ZERO, -ONE/EPS, AD, LDAD )
         CALL DLASET( 'F', N, M, ZERO, ZERO, BD, LDBD )
         CALL DLASET( 'F', NP, N, ZERO, ZERO, CD, LDCD )
         NSYS = N
      END IF
      LWA = 7*N*N + 2*M*N + 2*NP*N + N + LWAMAX
      WORK( 1 ) = DBLE( LWA )
      RETURN
C *** End of SB10JD ***
      END
```