SLICOT Working Note 2003-1

FORTRAN 77 ROUTINES FOR μ-SYNTHESIS AND H_∞ DESIGN [1]

Asparuh Markovski[2], **P. Petkov**[3], **D.-W. Gu**[4] and **M. Konstantinov**[5]

January 13, 2004

[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, Kasteelpark Arenberg 10, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN2003-1.ps.Z*

[2]Department of Automatics, Technical University of Sofia, 1756 Sofia, Bulgaria. Email:agm@bgmountains.bglink.net

[3]Department of Automatics, Technical University of Sofia, 1756 Sofia, Bulgaria.

[4]Department of Engineering, University of Leicester, Leicester LE1 7RH, U.K.,Email: dag@le.ac.uk

[5]University of Architecture & Civil Engineering, 1 Hr. Smirnenski Blv., 1421 Sofia, Bulgaria.

Abstract

A set of Fortran 77 subroutines aimed to perform μ-synthesis procedure via DK iterations or H_∞ design alone is presented. The software is intended for linear, time-invariant, continuous-time systems, but it handles also discrete-time systems via bilinear transformation. The methods for μ-synthesis and H_∞ design implemented in the routines are briefly described. The subroutines make use of LAPACK and BLAS libraries and can be easily implemented from MATLAB by a mex-file. The subroutines are included in the SLICOT library.

1 Introduction

In this report we present a set of Fortran 77 subroutines aiming at performing the μ-synthesis procedure via DK iterations. In one step of the μ-synthesis procedure, the H_∞ design is required. Hence the H_∞ design routines are included as well which would let users perform the H_∞ design alone. The software is intended for linear, time-invariant, continuous-time systems, but it handles also discrete-time systems via bilinear transformation. The methods for the μ-synthesis implemented in the routines are described. The subroutines make use of LAPACK [AndBBD95] and BLAS [LawHKK79], [DonD88], [DonD90] libraries and can be easily implemented from MATLAB[1] by a mex-file. The subroutines have been included in the SLICOT library.

The report is organised as follows. In Section 2 we present the main steps in the μ-synthesis of continuous-time systems. In Section 3 we give detailed description of the algorithms and subroutines for μ-synthesis. In Section 4 a sixth-order example is given to illustrate the implementation of the routines. In the Appendix we include the source code of the M-files intended for μ-synthesis.

The following notation is used in the paper. $\mathcal{R}(\mathcal{C})$ denotes the field of real(complex) numbers; $\mathcal{R}(\mathcal{C})^{m\times n}$ – the space of real (complex) $m \times n$ matrices $A = [a_{ij}]$; A^T – the transposed matrix A; $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ – the maximum and minimum singular values of A; $\|G(s)\|_\infty$ – the $\mathcal{H}_\infty$ norm of a stable transfer function matrix $G(s)$ and I_n – the unit $n \times n$ matrix.

2 Formulae for μ- synthesis of linear, time-invariant, continuous-time systems in the state-space

The standard structure of the μ-controller design and steps in the design procedure are detailed in the following sub-sections.

2.1 Main set-up for μ-synthesis

For the technique of the μ- synthesis to be applied, the design problem is needed to be represented in a standard configuration as Figure 1, which is based on the conventional usage of the Linear-Fractional Transformations (LFT).

The main idea is to separate the known, nominal dynamic model ($P(s)$) and the uncertain dynamics ($\Delta_{pert}(s)$) sub-blocks. The design goal is to decide a stabilising controller $K(s)$ which minimises the infinity norm of the transfer function from the "disturbances" d to the "error"signals e. Both the stabilisation and minimisation should be achieved by taking into consideration of the perturbation ($\Delta_{pert}(s)$). The perturbation ($\Delta_{pert}(s)$) has a certain structure. It is usually (block-)diagonal rather than a full matrix.

Here P comprises the nominal part of the model and, moreover, it usually includes the weighting functions for the uncertainty and performance. The next form of P reflects its structure as a transfer function between the signal sets $[z, d, u]^T$ and $[w, e, y]^T$.

$$P(s) \quad = \quad \begin{bmatrix} P_{11}(s) & P_{12}(s) & P_{13}(s) \\ P_{21}(s) & P_{22}(s) & P_{23}(s) \\ P_{31}(s) & P_{32}(s) & P_{33}(s) \end{bmatrix} \tag{2.1}$$

[1]MATLAB is a trademark of MathWorks, Inc.

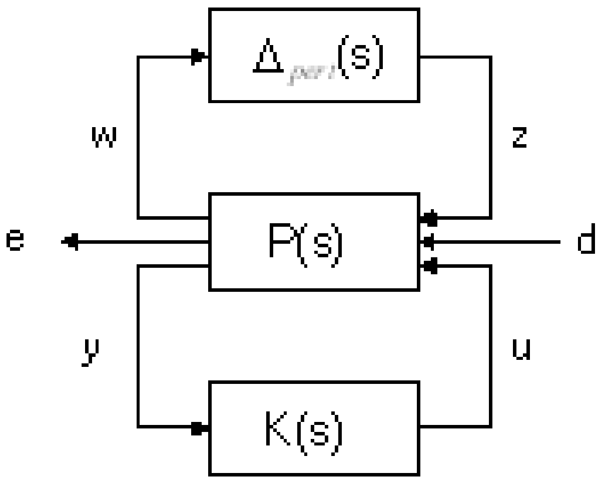

Figure 1: Main set-up for μ- synthesis

Δ_{pert} is the uncertainty matrix, representing the dynamic uncertainties of the model. It has a certain structure,

$$\mathbf{\Delta_{pert}} = \{diag[\Delta_1, \cdots, \Delta_p, \delta_1 I_{r_1}, \cdots, \delta_s I_{r_s}] : \Delta_j \in \mathcal{C}^{m_j \times m_j}, \delta_i \in \mathcal{C}\} \tag{2.2}$$

where $\sum_{j=1}^{p} m_j + \sum_{i=1}^{s} r_i = n$ with n is the dimension of the block Δ_{pert}. Δ_{pert} is norm-bounded, i.e. it obeys the inequality

$$\max_{\omega} \sigma_{max}[\Delta_{pert}(j\omega)] \leq 1 \tag{2.3}$$

Here the signals, accessible to the controller K, are denoted by y, and the control signal produced by K denoted by u.

Using LFT-s, the closed-loop system can be presented in the form

$$z = F_U(F_L(P,K), \Delta_{pert})w = F_L(F_U(P, \Delta_{pert}), K)w \tag{2.4}$$

For $M = F_L(P,K)(j\omega)$ the *structured singular value* μ in respect to the uncertainty structure $\mathbf{\Delta_{pert}}$ is defined by

$$\mu^{-1}_{\mathbf{\Delta_{pert}}}(M) := \min_{\Delta \in \mathbf{\Delta_{pert}}} \{\sigma_{\max}(\Delta) : \ det(I - M\Delta) = 0\} \tag{2.5}$$

If there is no $\Delta \in \mathbf{\Delta_{pert}}$ such that $det(I - M\Delta) = 0$, then $\mu_{\mathbf{\Delta_{pert}}}(M) := 0$.

The maximum value of μ over the frequency range $[0, \infty]$ is related to the robust stability and robust performance of the uncertain closed-loop system [ZhoD1].

The condition for robust performance of the closed-loop system is

$$\max_{\omega} \mu_{\mathbf{\Delta}}(M) < 1 \tag{2.6}$$

where $\mu_{\mathbf{\Delta}}(.)$ is the structured singular value with respect to the extended uncertainty matrix $\Delta \in \mathbf{\Delta}$ which includes a unity-norm bounded, performance block Δ_F. Thus,

$$\Delta \equiv \begin{bmatrix} \Delta_{pert} & 0 \\ 0 & \Delta_F \end{bmatrix} : \Delta_F \in C^{nd \times nc}, \tag{2.7}$$

Δ_F is a fictitious perturbation from the signal e to d, representing the performance requirements.

Hence, the goal of the μ-synthesis problem can be formulated like that: find a stabilising controller K, which minimises the maximal value of $\mu_\Delta(.)$ (over a range of given frequencies), i.e.,

$$\min_K \max_\omega \mu_{\mathbf{\Delta}}(F_L(P,K)(j\omega)) \tag{2.8}$$

Up to now, this problem cannot yet be solved in a straight way. All the known methods, including those using the Linear Matrix Inequalities (LMI), are based upon some searching procedure. The following result is exploited in the method with DK iterations [ZhoD1], which is based on the characteristics of the μ-norm:

$$\rho(M) \le \mu_{\mathbf{\Delta}}(M)) \le \sigma_{\max}(M)$$

where $M \equiv (F_L(P,K))$, and $\rho(M)$ is the spectral radius of M. To estimate $\mu_{\mathbf{\Delta}}(M)$ the following transformations of M can be used, which do not change $\mu_{\mathbf{\Delta}}(M)$, but do change $\rho(M)$ and $\sigma_{\max}(M)$:

$$\max_\Delta \rho(\Delta M) = \mu_{\mathbf{\Delta}}(M) \le \inf_D (DMD^{-1}) \tag{2.9}$$

where the equality $\Delta D{=}D\Delta$ is in force for the scaling system D. The structure of D is thus as follows:

$$D = diag(d_1 I_{r_1} ... d_n I_{r_s}, D_1 ... D_p, I_F) \tag{2.10}$$

Here the blocks $d_1 I_{m_1} ... d_p I_{m_p}$ correspond to the full (complex) blocks $\Delta_1 ... \Delta_p$, $D_1 ... D_s$ correspond to the multiple (for $r_i > 1$) or alone (for $r_i{=}1$) scalar blocks $\delta_1 I_{r_1} ... \delta_p I_{r_p}$ in the uncertainty structure. I_F corresponds to the fictitious block Δ_F which is responsible for the performance.

Therefore, the problem for the synthesis of a μ-optimal controller can be reformulated in the following way:

$$\min_{K_s} \max_\omega \min_{D_\omega} \sigma_{\max}(D_\omega M D_\omega^{-1}) \tag{2.11}$$

where K_s stands for *stabilising controller.* The fact that multiplying by an orthonormal matrix does not change the maximal singular value gives the opportunity of trade-off to be made by the scaling matrix D and its corresponding $\hat{D} = UD$, where U is an orthonormal matrix. Hence, there is a freedom in determining the phase of D_ω, which allows it to be fitted with a real-rational, stable, minimum phase system $D(s)$. Finally, changing the maximal singular value with its H_∞ norm, the problem for μ optimal synthesis takes the shape

$$\min_{Ks} \min_{D(s)} \left[D(s) F_L(P,K) D(s)^{-1} \right]_\infty \tag{2.12}$$

This formulation allows the problem for μ-optimal synthesis to be solved via the method of a two-step optimisation, also known as DK iterations:

A. K step. For a fixed $D(s)$ the problem of finding H_∞ optimal controller K_s is solved ($D(s) = I$ can be appropriate for the first step. See Step D below for following steps.):

$$K_s : \min_{Ks} \|F_L(P_e, K)\|_\infty \tag{2.13}$$

B. $\mu_\Delta(M)$ is estimated. Continue if the μ-values are not satisfactory yet.

C. D step. For a fixed K, solve the following convex optimization problem for D at each frequency over the selected frequency range,

$$D(j\omega) = arg \inf_{D \in \mathbf{D}} \sigma_{\max}[DF_l(P_e, K) D^{-1}(j\omega)]$$

The obtained D_ω are fitted with real-rational, stable and minimum phase $D(s)$.

D. The system is then extended by the D-s:

$$P_e(s)=diag(D,I)P_e(s)diag(D^{-1},I) \tag{2.14}$$

E. Go to the step A with the newly extended system. The iterations stops if a better controller in the sense of the μ- norm of the closed-loop system cannot be found after certain number of iterations.

One of the main drawbacks of the method is the need of processing too high order systems.

Such a procedure can be constructed for discrete-time systems as well. But in the developed package here, the discrete-time systems are treated via transforming the original model into continuous-time by bilinear transformation, and the controller obtained is discretised to be used in the original discrete-time context. Our experience in practice shows that this approach works in most cases, although in some designs a pure, discrete-time synthesis could give better results.

The lack of global convergence is another drawback of the method based on DK iterations.

2.2 The K step; H_∞ synthesis

The standard set-up for H_∞ synthesis problem is shown in Figure 2.

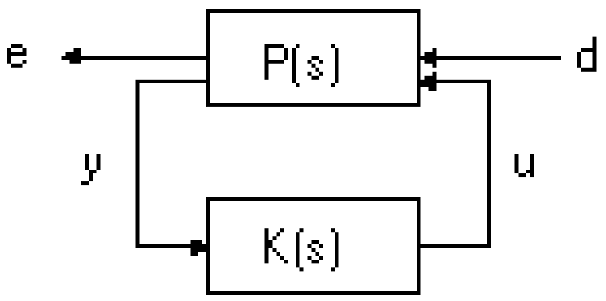

Figure 2: Set-up for H_∞ synthesis

The goal of the suboptimal problem is to find such a stabilising controller K, so the next inequality holds:

$$\min_{K_s} [F_L(P,K)]_\infty < \gamma \tag{2.15}$$

for a given positive number γ. To find the optimal controller, one can iteratively decrease γ, until it reaches its minimal admissible value, as it is shown below in the text.

According to the figure, the model of the object can be presented in the following way:

$$P(s) = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \tag{2.16}$$

Let us first concentrate on the $D_{22} = 0$ case. Actually, this condition holds for almost every real system in the practice; if it is not in force (as, for example, in the μ- synthesis due to the performance weighting functions wrapped in the model), the general solution of the problem for H_∞ synthesis can be easily constructed based on that case.

To apply the original approach based on solving two Riccati equations, known also as DGKF approach, the necessary and sufficient conditions for the existence of the H_∞ *suboptimal* controller [GloD1] must hold:

A1. (A, B_2) is stabilisable and (C_2, A) is detectable;

A2. $D_{12} = \begin{bmatrix} 0 \\ I_{m2} \end{bmatrix}$ *and* $D_{21} = \begin{bmatrix} 0 & I_{p2} \end{bmatrix}$;

A3. $\begin{bmatrix} A - j\omega I & B_2 \\ C_1 & D_{12} \end{bmatrix}$ *has full column rank for all* ω;

A4. $\begin{bmatrix} A - j\omega I & B_1 \\ C_2 & D_{21} \end{bmatrix}$ *has full row rank for all* ω.

The two matrix algebraic Riccati equations (X and Y equations) which the solving of the H_∞ synthesis problem is based on have the following structure:

$$\begin{array}{c} E_1^T X + XE_1 \text{ - } XW_1X + Q_1 = 0 \\ E_2Y + Y_2ET \text{ - } YW_2Y + Q_2 = 0 \end{array} \tag{2.17}$$

which correspond to the Hamiltonian matrices $H \equiv \begin{bmatrix} E_1 & -W_1 \\ -Q_1 & -E_1^T \end{bmatrix}$, $J \equiv \begin{bmatrix} E_2^T & -W_2 \\ -Q_2 & -E_2 \end{bmatrix}$, where W_i=W_i^T and $Q_i = Q_i^T$. Then the stabilising solutions X and Y of (2.17), if exist, are symmetrical matrices and are denoted by

$$X \equiv Ric \begin{bmatrix} E_1 & -W_1 \\ -Q_1 & -E_1^T \end{bmatrix} \quad , \quad Y \equiv Ric \begin{bmatrix} E_2^T & -W_2 \\ -Q_2 & -E_2 \end{bmatrix} \tag{2.18}$$

The following notions are introduced for the H_∞ synthesis problem:

$$R_n \equiv D_{1*}TD_{1*} - \begin{bmatrix} \gamma^2 \mathrm{I}_{m1} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.19}$$

$$\tilde{R}_n \equiv D_{*1}TD_{*1} - \begin{bmatrix} \gamma^2 \mathrm{I}_{p1} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.20}$$

where $D_{1*} = \begin{bmatrix} D_{11} & D_{12} \end{bmatrix}$ and $D_{*1} = \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix}$. The two Hamiltonian matrices, corresponding to the X and Y Riccati equations, are defined as follows:

$$H \equiv \begin{bmatrix} A & 0 \\ -C_1^T C_1 & -A^T \end{bmatrix} - \begin{bmatrix} B \\ -C_1^T D_{1*} \end{bmatrix} R_n^{-1} \begin{bmatrix} D_{1*}^T C_1 & B^T \end{bmatrix} \tag{2.21}$$

$$J \equiv \begin{bmatrix} A^T & 0 \\ -B_1 B_1^T & -A \end{bmatrix} - \begin{bmatrix} C^T \\ -B_1 D_{*1}^T \end{bmatrix} \tilde{R}_n - 1 \begin{bmatrix} D_{*1}^T B_1^T & C \end{bmatrix} \tag{2.22}$$

Based on X=$Ric(H)$, Y=$Ric(J)$ the matrices F (*state feedback*) and H (*output injection*) are constructed, which will be used in the controller structure:

$$F := -R^{-1}(D_{1*}^T C_1 + B^T X) =: \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} =: \begin{bmatrix} F_{11} \\ F_{12} \\ F_2 \end{bmatrix} \tag{2.23}$$

$$L := -(B_1 D_{*1}^T + YC^T)\tilde{R}_n^{-1} =: \left[\begin{array}{cc} L_1 & L_2 \end{array}\right] =: \left[\begin{array}{ccc} L_{11} & L_{12} & L_2 \end{array}\right] \tag{2.24}$$

Here F_1, F_2, F_{11}, F_{12} have m_1, m_2, $m_1 - p_2$, p_2 rows, respectively, and L_1, L_2, L_{11}, L_{12} - p_1, p_2, $p_1 - m_2$, m_2 columns.

The matrix D_{11} is divided in the following way:

$$D_{11} = \left[\begin{array}{cc} D_{1111} & D_{1112} \\ D_{1121} & D_{1122} \end{array}\right] \tag{2.25}$$

where D_{1122} has m_2 rows and p_2 columns. Then the solution of the H_∞ synthesis problem is given by the next theorem:

Theorem [GloD1]: If the conditions A1 - A4 hold for P(s), then: a) The stabilising controller K(s): $\|F_L(P,K)\|_\infty < \gamma$ *exists if and only if*

(i) $\gamma > \max(\sigma_{max}[D_{1111}, D_{1112}], \sigma_{max}[D_{1111}^T, D_{1121}^T])$, *and*

(ii) the stabilising solutions X=Ric(H) , Y=Ric(J) exist and the following inequality holds

$$\rho(XY) < \gamma^2;$$

(b) If the conditions in (a) are in force, then all rational, stabilising controllers K(s): $\|F_L(P,K)\|_\infty < \gamma$ *are given by the formula*

$$K(s) = F_L(M(s), \Phi(s))$$

where for the rational $\Phi(s)$ *the inequality* $\|\Phi(\mathrm{s})\|_\infty < \gamma$ *holds, and M(s) has the following realisation:*

$$M(s) = \left[\begin{array}{ccc} \hat{A} & \hat{B}_1 & \hat{B}_2 \\ \hat{C}_1 & \hat{D}_{11} & \hat{D}_{12} \\ \hat{C}_2 & \hat{D}_{21} & \hat{D}_{22} \end{array}\right]$$

$$\hat{D}_{11} = -D_{1121} D_{1111}^T (\gamma^2 I - D_{1111} D_{1111}^T)^{-1} D_{1112} - D_{1122}$$

For the matrices $\hat{D}_{11} \in R^{m_2 \times m_2}$ *and* $\hat{D}_{21} \in R^{p_2 \times p_2}$ *(which are Cholesky factors) the following equalities are in force*

$$\hat{D}_{12} \hat{D}_{12}^T = I - D_{1121}(\gamma^2 I - D_{1111}^T D_{1111})^{-1} D_{1121}^T$$

$$\hat{D}_{21}^T \hat{D}_{21} = I - D_{1112}^T (\gamma^2 I - D_{1111} D_{1111}^T)^{-1} D_{1121}$$

The other system matrices are determined by the formulas

$$\hat{B}_2 = Z(B_2 + L_{12})\hat{D}_{12}$$

$$\hat{C}_2 = -\hat{D}_{21}(C_2 + F_{12})$$

$$\hat{B}_1 = -ZL_2 + \hat{B}_2 \hat{D}_{12}^{-1} \hat{D}_{11} = -ZL_2 + Z(B_2 + L_{12})\hat{D}_{11}$$

$$\hat{C}_1 = F_2 + \hat{D}_{11} \hat{D}_{21}^{-1} \hat{C}_2 = F_2 - \hat{D}_{11}(C_2 + F_{12})$$

$$\hat{A} = A + BF + \hat{B}_1 \hat{D}_{21} \hat{C}_2 = A + BF - \hat{B}_1(C_2 + F_{12})$$

where

$$Z = (I - \gamma^{-2} YX)^{-1} \quad .$$

If $\Phi(s) = 0$, the suboptimal controller is called *central* and has the realisation

$$K(s) = \left[\begin{array}{cc} \hat{A} & \hat{B}_1 \\ \hat{C}_1 & \hat{D}_{11} \end{array} \right] \tag{2.26}$$

The formulas above are valid if, according to the condition A2, the matrices D_{12} and D_{21} are already in *normalised* form, i.e., the system $P(s)$ is *normalised.* The normalisation can be made in the following way [GuP1]:

Let the matrices D_{12} and D_{21} have full column and row rank, correspondingly. Then the orthonormal matrices U_{12}, V_{12}, U_{21}, V_{21} exist (which can be found via singular value decomposition method, for example), so that

$$U_{12}D_{12}V_{12}^T = \left[\begin{array}{c} 0 \\ \Sigma_{12} \end{array} \right] \tag{2.27}$$

$$U_{21}D_{21}V_{21}^T = \left[\begin{array}{cc} 0 & \Sigma_{21} \end{array} \right] \tag{2.28}$$

where Σ_{12} and Σ_{21} are nonsingular. Additionally, it is in force

$$U_{12}D_{12}V_{12}^T\Sigma_{12}^{-1} = \left[\begin{array}{c} 0 \\ I \end{array} \right] \tag{2.29}$$

$$\Sigma_{21}^{-1}U_{21}D_{21}V_{21}^T = \left[\begin{array}{cc} 0 & I \end{array} \right] \tag{2.30}$$

That way, the right parts of the equations above are in transformed into normalised form. If $p_1 > m_2$ and $p_2 < m_1$, then the matrices U_{12} and V_{21} can be divided as follows:

$$U_{12} = \left[\begin{array}{c} U_{121} \\ U_{122} \end{array} \right] \tag{2.31}$$

$$V_{21} = \left[\begin{array}{c} V_{211} \\ V_{212} \end{array} \right] \tag{2.32}$$

with $U_{121} \in \mathcal{R}^{(p_1-m_2)\times p_1}, U_{122} \in \mathcal{R}^{m_2\times p_1}, V_{211} \in \mathcal{R}^{(m_1-p_2)\times m_1}, V_{212} \in \mathcal{R}^{p_2\times m_1}$.

Then the normalisation formulas for P(s) take the following form:

$$\bar{B}_2 = B_2V_{12}^T\Sigma_{12}^{-1} \tag{2.33}$$

$$\bar{C}_1 = U_{12}C_1 \tag{2.34}$$

$$\bar{C}_2 = \Sigma_{21}^{-1}U_{21}C_2 \tag{2.35}$$

$$\bar{D}_{11} = U_{12}D_{11}V_{21}^T = \left[\begin{array}{cc} U_{121}D_{11}V_{211}^T & U_{121}D_{11}V_{212}^T \\ U_{122}D_{11}V_{211}^T & U_{122}D_{11}V_{212}^T \end{array} \right] \tag{2.36}$$

$$\bar{D}_{12} = U_{12}D_{12}V_{12}^T\Sigma_{12}^{-1} = \left[\begin{array}{c} 0 \\ I \end{array} \right] \tag{2.37}$$

$$\bar{D}_{21} = \Sigma_{21}^{-1}U_{21}D_{21}V_{21}^T = \left[\begin{array}{cc} 0 & I \end{array} \right] \tag{2.38}$$

The transformations above do not change the μ norm of the system due to the orthonormality of the matrices V_{21} and U_{12}. The controller for the original system description can be obtained by the formula

$$K(s) = V_{12}^T\Sigma_{12}^{-1}\bar{K}(s)\Sigma_{21}^{-1}U_{21} \tag{2.39}$$

Finally, let us take a look at the $D_{22} = 0$ case. If there is already synthesised a controller for the simplified system with $D_{22} = 0$, which has the state-space realisation

$$K(s) = \left[\begin{array}{cc} A_K & B_K \\ C_K & D_K \end{array} \right] \tag{2.40}$$

then the controller for the full system is given by the formula

$$\tilde{K}(s) = K(s)(I + D_{22}K(s))^{-1} \tag{2.41}$$

and its state-space realisation is

$$\tilde{K}(s) = \left[\begin{array}{ll} A_K - B_K D_{22}(I + D_K D_{22})^{-1} C_K & B_K(I + D_{22}D_K)^{-1} \\ (I + D_K D_{22})^{-1} C_K & D_K(I + D_{22}D_K)^{-1} \end{array} \right] \tag{2.42}$$

2.3 The D step

The case with complex uncertainty blocks is examined, namely

$$\Delta = \left[\begin{array}{ccccc} \Delta_1 & 0 & . & 0 & 0 \\ 0 & \Delta_2 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & \Delta_N & 0 \\ 0 & 0 & . & 0 & \Delta_F \end{array} \right] \tag{2.43}$$

where Δ_i are matrices with dimensions $r_i \times c_i$, which corresponds to a scaling system D of the form

$$D(s) = \left[\begin{array}{lllll} d_1(s)I_1 & 0 & . & 0 & 0 \\ 0 & d_2(s)I_2 & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & d_N(s)I_N & 0 \\ 0 & 0 & . & 0 & I_F \end{array} \right] \tag{2.44}$$

Then the problem of fitting $D(\omega)$ (refferred as D_ω) with $D(s)$ can be divided into subproblems of fitting every $d_i(\omega)$ with its corresponding $d_i(s)$. As it was shown above, the changes in the phase of $d_i(\omega)$ did not make influence on $\sigma_{max}(D_\omega F_L(P, K)(j\omega)D_\omega^{-1})$, so, for every positive $d_i(\omega)$, where the set of the frequencies is limited, a proper, stable and minimal phase transfer function $d_i(s)$ can be found.

The problem can be solved into the following steps:

a) for the gain $d_i(\omega)$ a phase $\theta_i(\omega)$ is found so that the function

$$d_i(\omega) := d_i(\omega)e^{j\theta_i(\omega)} \tag{2.45}$$

corresponds to a stable, minimal phase system;

b) for $d_i(\omega)$ the coefficients of the proper transfer function $d_i(s)$ are found using the Least Squares Method (with respect to its coefficients);

c) a state-space description in a phase-coordinate form is constructed for the di(s) obtained;

d) now it is necessary to check whether this state-space description still corresponds to a stable, minimal-phase system, because in the process of the applying Least Squares Method in step b) these properties can be lost. For this purpose the system poles p and zeros z are estimated and if some of them have positive real parts, they are exchanged correspondingly with $-p$ and $-z$. For the set of poles and zeros obtained with negative real parts the transfer function $d_i'(s)$ is constructed; then, using the theorem of Volovich for the observable structures, a minimal state-space realisation for $d_i'(s)$ is constructed. This state-space description is recalled in a way so it has the same gain as the original one;

e) the state-space descriptions, which correspond to all $d_i(s)$, are placed into $D(s)$, which the matrices I_l or I_r are docked to finally, corresponding to the number of the inputs or outputs of the controller (depending on which is greater).

***

For finding the minimal phase of the given frequency function $d_i(\omega)$ the features of the logarithm of the discrete Fourier (or z-) transform can be exploited, using the so-called *complex cepstrum* [BogH1]. An idea of how to make this is given in [OppS1] as well as in [BalD1] and it is applied in the μ-toolbox for MATLAB:

The rational z-transform corresponding to a signal $x(n)$, in general has the form

$$X(z) = \frac{Az^r \prod\limits_{k=1}^{m_i} (1 - a_k z^{-1}) \prod\limits_{k=1}^{m_0} (1 - b_k z)}{\prod\limits_{k=1}^{p_i} (1 - c_k z^{-1}) \prod\limits_{k=1}^{p_0} (1 - d_k z)} \tag{2.46}$$

where $|a_k|, |b_k|, |c_k|, |d_k|$ are less then 1, so that, on the one hand, the multipliers $(1 - a_k z^{-1})$ and $(1 - c_k z^{-1})$ correspond to zeros and poles inside the unit circle, and on the other hand $(1 - b_k z)$ and $(1 - d_k z)$ correspond to zeros and poles outside it.

The length of the signal realisation here is limited, i.e., $X(z)$ does not have denominator; moreover, it can be accepted without lost of generality that there is no pure delay, i.e., $r = 0$. Then $X(z)$ gets the following simplified form:

$$X(z) = A \prod_{k=1}^{mi} (1 - a_k z^{-1}) \prod_{k=1}^{m0} (1 - b_k z) \tag{2.47}$$

For $X(z)$ to be minimal phase, the coefficients $b_k = 0$ is a must.

Then $ln(X(z))$ is obtained:

$$\hat{X}(z) \equiv \ln(X(z)) = \ln(A) + \sum_{k=1}^{m_i} \ln(1 - a_k z^{-1}) + \sum_{k=1}^{m_0} \ln(1 - b_k z) \tag{2.48}$$

In order to find the inverse z-transform of the term above, one can exploit their representing in series in powers of z and z^{-1}:

$$\ln(1 - a_k z^{-1}) = -\sum_{n=1}^{\infty} \frac{a_k^n}{n} z^{-n}, \quad |z| > |a_k| \tag{2.49}$$

$$\ln(1 - b_k z) = -\sum_{n=1}^{\infty} \frac{a_k^n}{n} z^{n}, \quad \left|z^{-1}\right| > |b_k| \tag{2.50}$$

Making use of the formulas for inverse z-transform of these terms, it is obtained for the logarithm of $x(k)$

$$\hat{x}(n) \equiv \ln(x(n)) = \begin{cases} \ln(A), n = 0 \\ -\sum\limits_{k=1}^{m_i} \frac{a_k n}{n}, n > 0 \\ -\sum\limits_{k=1}^{m_0} \frac{b_k n}{n}, n < 0 \end{cases} \tag{2.51}$$

The conclusion is - to get minimal phase $ln(X(z))$, namely $bk = 0$, one must have $ln(x(n))$ causal, i.e., $\hat{x}(n)$=0 for n<0.

The z-transform of a causal series is uniformly determined by the real part of theirs Fourier transforms, so, to find $\hat{x}(n)$, it is enough to be estimated

$$\hat{X}_R(e^{j\omega}) = \ln\left|X(e^{j\omega})\right| \tag{2.52}$$

The inverse discrete Fourier transform of $\hat{X}_R(e^{j\omega})$ is equivalent to the even part of $\hat{x}(n)$, here denoted by $c(n)$:

$$c(n) = \frac{\hat{x}(n) + \hat{x}(-n)}{2} \tag{2.53}$$

Since $\hat{x}(n)$=0 for n <0, the following dependence is valid:

$$\hat{x}(n) = c(n)u_+(n) \text{ for } u_+(n) = \begin{cases} 0, n < 0 \\ 1, n = 0 \\ 2, n > 0 \end{cases}$$

It is clear now from the exposed above how such a $qi(\omega)$ can be constructed, which corresponds to a minimal phase $d_i(\omega)$:

- estimate ln(.) of the data given for $|d_i(\omega)|$, ($2N$ points):

$$ld_i(\omega){=}\ln|d_i(\omega)| \tag{2.54}$$

- find the inverse discrete Fourier transform of the obtained, applying the Fast Fourier Transform (FFT) algorithm (N points):

$$ld_i(n) = F^{-1}\{ld_i(\omega)\} \tag{2.55}$$

- the value of the result for $n = 0$ has to be taken with a half of the weight in comparison to these for $1 < n \leq N/2$:

$$ld_i(0) = ld_i(0)/2 \tag{2.56}$$

- return to the frequency domain using the FFT algorithm ($N/2$ points):

$$ld_i(\omega) = F\{ld_i(n)\} \tag{2.57}$$

- finally, the antilogarithm of the result is estimated:

$$d_i(\omega) = e^{ld_i(\omega)} \tag{2.58}$$

***

In order to perform the b) step, i.e., to fit $d_i(\omega)$ with $d_i(s)$, the coefficients of $d_i(s)$ must be close to the frequency data in a least square sense, according to the Bode integral formula. The following optimization problem has to be solved: Find the vector X with the $2n + 1$ coefficients of $d_i(s)$ from the equation

$$B_x = A_x X \tag{2.59}$$

where

$$A_x = \begin{bmatrix} Re(a_0, A_B) \\ Im(a_0, A_B) \end{bmatrix} \tag{2.60}$$

$$B_x = \begin{bmatrix} Re(B_p) \\ Im(B_p) \end{bmatrix} \tag{2.61}$$

$$a_0 = \begin{bmatrix} v(1)^n & . & v(1)^2 & v(1) & 1 \\ v(2)^n & . & v(2)^2 & v(2) & 1 \\ . & . & . & . & . \\ v(N_\omega)^n & . & v(N_\omega)^2 & v(N_\omega) & 1 \end{bmatrix} \quad , \quad v(k) = e^{j\omega(k)} \tag{2.62}$$

$$A_B = \begin{bmatrix} d_{i\omega}(1)a_0(1,2) & . & d_{i\omega}(1)a_0(1,n+1) \\ . & . & . \\ d_{i\omega}(N_\omega)a_0(1,2) & . & d_{i\omega}(N_\omega)a_0(1,n+1) \end{bmatrix} \tag{2.63}$$

$$B_p = \begin{bmatrix} v(1)^n d_{i\omega}(1) \\ . \\ v(N_\omega)^n d_i\omega(N_\omega) \end{bmatrix} \tag{2.64}$$

***

The approach, described above, works in the discrete-time domain. If the data are continuous-time, they can be transformed into discrete-time at the beginning of b) step making use of the scaled bilinear transformation

$$\omega(k) := a\cos\left(\frac{1-\omega^2(k)/p}{1+\omega^2(k)/p}\right), \quad p = \sqrt{\omega^2(0)+\omega^2(N/2)} \tag{2.65}$$

In this case a discrete-to-continuous transformation must be performed for the state-space description obtained at the end of step c).

Additionally, linear interpolation of the data around the unit circle to improve the numerical performance of the algorithm for finding the minimal phase is made in step b).

A similar algorithm for performing the D step is realised in μ-toolbox for MATLAB.

3 Description and program schemes of the routines for making μ-synthesis via DK iterations

3.1 The program for μ-synthesis `SB10DK.f`

• ***Purpose.*** `SB10DK.f` has the form of a gateway function for MATLAB for the convenience of the user; thus, it allows easy work and interpretation of the results obtained by it. It completes the whole procedure of DK iterations: for the given as an input state-space description, uncertainty structure and the frequency vector it estimates μ optimal controller. It can also compute H_∞ controller only, optimal or suboptimal. The synthesis procedure is steered by an user-supplied values for some tolerances: one for controlling the accuracy of the K step (how close γ has got to the γ_{opt}) and one for the D step (how D is close to D_ω in the sense of frequency response), as well as for choosing a method for getting close to γ_{opt} (by bisection, by scanning, or by bisection and then scanning). An information for the condition numbers of the Riccati equations solved in the K step is taken out; outputted are also the reached $\bar{\mu}(\omega)$ of the closed-loop system and the condition numbers of the transformations, which make the initial system matching the A1-A4 conditions at every K step.

For discrete-time systems the problem is solved via transformation to the continuous-time of the initial data, and at the final the controller obtained is discretised. The repeated uncertainty blocks are treated as single; the practice shows that the result does not become much worse due to such a limitation.

• ***Remarks***. The K and D steps are performed by the new routines `SB10AD.f` and `SB10MD.f`, respectively. For the discrete-to-continuous transformation of the original system and discretising the obtained controller (if needed) the SLICOT subroutine `AB04MD.f` is employed, which uses scaled bilinear transformation [SafG1].

• ***Additional routines:***

`slimju.m` (M-file for easier calling sb10dk in the case of μ controller)
`slihinf.m` (M-file for easier calling sb10dk in the case of H_∞ controller)
`test_slimju.m` (test program for MATLAB)
`sb10dk.mat` (data file for `test_sb10dk.m`)

• ***Program scheme:***

1. If (discrete-time data) : $P \xrightarrow[\alpha=1,\beta=1]{\substack{discr \rightarrow cont. \\ bilinear\ transform}} P$, $\omega(k) \leftarrow \cos\left(\frac{1-\cos\omega(k)}{1+\cos\omega(k)+tol}\right)$,

where $P \equiv \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

2.1. Solve $(K, \gamma_{min}) : \min\limits_K [F_L(P,K), \gamma]_\infty$, $\max(\bar{\mu}) \leftarrow \gamma_{min}$, where $K \equiv \begin{bmatrix} A_K & B_K \\ C_K & D_K \end{bmatrix}$

2.2. Estimate $M = F_L(P,K)$, where $M \equiv \begin{bmatrix} A_C & B_C \\ C_C & D_C \end{bmatrix}$

3.1. Solve $\mu_\Delta(M)$, D_ω

3.2. $D_\omega \xrightarrow{\substack{complex\ cepstrum \\ Bode\ integral\ formula}} D_L, D_R$, where $D_L \equiv \begin{bmatrix} A_{DL} & B_{DL} \\ C_{DL} & D_{DL} \end{bmatrix}$, $D_R \equiv \begin{bmatrix} A_{DR} & B_{DR} \\ C_{DR} & D_{DR} \end{bmatrix}$

4. If $\max\limits_\omega(\bar{\mu_i}) \geq \max\limits_\omega(\bar{\mu_{i-1}})$: go to 11

5. $K_{best} \leftarrow K$

6. Solve $D_L P$:

6.1. $A_E \leftarrow \begin{bmatrix} A_{DL} & B_{DL}C \\ 0 & A \end{bmatrix}$

6.2. $B_E \leftarrow \begin{bmatrix} B_{DL}D \\ B \end{bmatrix}$

6.3. $C_E \leftarrow \begin{bmatrix} C_{DL} & D_{DL}C \end{bmatrix}$

6.4. $D_E \leftarrow [D_{DL}D]$

7. Solve $D_{Ri} \equiv D_R^{-1}$:

7.1. LU factorisation of D_{DR}

7.2. $D_{DRi} \leftarrow D_{DR}^{-1}$

7.3. $B_{DRi} \leftarrow -B_{DR}D_{DRi}$

7.4. $A_{DRi} \leftarrow A_{DR} + B_{DRi}C_{DR}$

7.5. $C_{DRi} \leftarrow D_{DRi}C_{DR}$

8. Solve $P_E \equiv D_L P D_R^{-1}$, where $P_E \equiv \begin{bmatrix} A_E & B_E \\ C_E & D_E \end{bmatrix}$:

8.1. $A_E \leftarrow \begin{bmatrix} A_E & B_E C_{DRi} \\ 0 & A_{DRi} \end{bmatrix}$

8.2. $B_E \leftarrow \begin{bmatrix} B_E D_{DRi} \\ B_{DRi} \end{bmatrix}$

8.3. $C_E \leftarrow \begin{bmatrix} C_E & D_E C_{DRi} \end{bmatrix}$

8.4. $D_E \leftarrow [D_E D_{DRi}]$

9.1. $(K, \gamma_{min,i}) : \min\limits_{K} [F_L(P_E, K), \gamma_{min,i-1}]_\infty$

9.2. $M = F_L(P_E, K)$

10. If (K exists) go to 3

11. If(discr.) $P \quad \overset{\substack{cont. \rightarrow discr.. \\ bilinear\ transform \\ \alpha = 1, \beta = 1}}{\longrightarrow} \quad P$ •

3.2 The computational subroutine for H_∞ synthesis `SB10AD.f`

•***Purpose.***

The computational subroutine `SB10AD.f` performs the K step, i.e., for a given continuous-time state-space description it estimates an optimal H_∞ controller, utilising the Glover&Doyle 1988 formulas (modified according to [PetG1] for increasing the numerical stability); there are three options for approaching γ_{opt} by bisection, by scanning, or combined (bisection, then scanning) and the procedure is tolerance-controllable. The closed-loop system is also estimated, if exist, and a stability test for it is performed. An information about the condition numbers of the corresponding Riccati equations and normalisation transformations is also outputted, as well as for the gopt achieved.

As an option, a suboptimal controller only can be produced.

•***Remarks.*** The next SLICOT subroutines are used: `SB10PD.f` (for normalisation of the original system using orthonormal transformations), `SB10QD.f` (for estimating the F and L matrices of the suboptimal controller for the current step and the condition numbers of the Riccati equations utilising a scaled Schur method), `SB10RD.f` (for constructing the suboptimal controller state-space realisation), `SB10LD.f` (for obtaining the closed-loop system). A program error is found in the current version of `SB10LD.f` in the development process, which is fixed and a new version of it is released to SLICOT.

•***Additional routines:***

`SB10AG.f` (gateway function for MATLAB)

•***Program scheme:***

1. $P(s) \overset{normalize}{\longrightarrow} P(s) \equiv \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & 0 \end{bmatrix}$:

1.1. $B_2 \leftarrow B_2 V_{12}^T \Sigma_{12}^{-1}$

1.2. $C_1 \leftarrow U_{12} C_1$

1.3. $C_2 \leftarrow \Sigma_{21}^{-1} U_{21} C_2$

1.4. $D_{11} \leftarrow U_{12} D_{11} V_{21}^T = \begin{bmatrix} U_{121} D_{11} V_{211}^T & U_{121} D_{11} V_{212}^T \\ U_{122} D_{11} V_{211}^T & U_{122} D_{11} V_{212}^T \end{bmatrix}$

1.5. $D_{12} \leftarrow U_{12} D_{12} V_{12}^T \Sigma_{12}^{-1} = \begin{bmatrix} 0 \\ I \end{bmatrix}$

1.6. $D_{21} \leftarrow \Sigma_{21}^{-1} U_{21} D_{21} V_{21}^T = \left[\begin{array}{cc} 0 & I \end{array}\right]$
2. $\gamma_{min} = \max(\sigma_{max}[D_{1111}, D_{1112}], \sigma_{max}[D_{1111}^T, D_{1121}^T])$
3. If $\gamma < \gamma_{min}$: Error exit
4. Start of iterations: $\gamma_{step} = \gamma - \gamma_{min}$, $\gamma_{opt} = \gamma$, $\gamma_{max} = \gamma$
5.1. $D_{1*} = \left[\begin{array}{cc} D_{11} & D_{12} \end{array}\right]$
5.2. $D_{*1} = \left[\begin{array}{c} D_{11} \\ D_{21} \end{array}\right]$
5.3. $R_n = {D_{1*}}^T D_{1*} - \left[\begin{array}{cc} \gamma^2 I_{m1} & 0 \\ 0 & 0 \end{array}\right]$
5.4. $\tilde{R}_n = D_{*1}^T D_{*1} - \left[\begin{array}{cc} \gamma^2 I_{p1} & 0 \\ 0 & 0 \end{array}\right]$
5.5. $H = \left[\begin{array}{cc} A & 0 \\ -C_1^T C_1 & -A^T \end{array}\right] - \left[\begin{array}{c} B \\ -C_1^T D_{1*} \end{array}\right] R_{n-1} \left[\begin{array}{cc} D_{1*}^T C_1 & B^T \end{array}\right]$
5.6. $J = \left[\begin{array}{cc} A^T & 0 \\ -B_1 B_1^T & -A \end{array}\right] - \left[\begin{array}{c} C^T \\ -B_1 D_{*1}^T \end{array}\right] \tilde{R}_{n-1} \left[\begin{array}{cc} D_{*1}^T B_1^T & C \end{array}\right]$
5.7. $X = Ric(H)$, $Y = Ric(J)$
6. Construct $\hat{K}(s) \equiv \left[\begin{array}{cc} \hat{A} & \hat{B}_1 \\ \hat{C}_1 & \hat{D}_{11} \end{array}\right]$:
6.1. $\hat{D}_{11} = -D_{1121} D_{1111}^T (\gamma^2 I - D_{1111} D_{1111}^T)^{-1} D_{1112} - D_{1122}$
6.2. $Z = (I - \gamma^{-2} YX)^{-1}$
6.3. $\hat{B}_1 = -Z L_2 + Z(B_2 + L_{12}) \hat{D}_{11}$
6.4. $\hat{C}_1 = F_2 - \hat{D}_{11}(C_2 + F_{12})$
6.5. $\hat{A} = A + BF - \hat{B}_1 (C_2 + F_{12})$
7. $K(s) = V_{12}^T \Sigma_{12}^{-1} \hat{K}(s) \Sigma_{21}^{-1} U_{21} \equiv \left[\begin{array}{cc} A_K & B_K \\ C_K & D_K \end{array}\right]$
8. $K(s) \leftarrow \left[\begin{array}{cc} A_K - B_K D_{22}(I + D_K D_{22})^{-1} C_K & B_K (I + D_{22} D_K)^{-1} \\ (I + D_K D_{22})^{-1} C_K & D_K (I + D_{22} D_K)^{-1} \end{array}\right]$
9. $M = F_L(P, K)$
10.1. If *Bisection*:
If $\max(Re(eig(M))) < 0$: $\gamma_{opt} = \gamma$, $\gamma = \gamma - \gamma_{step}$
else $\gamma = \min(\gamma + \gamma_{step}, \gamma_{max})$
10.2. If *Scanning*: $\gamma = \gamma - \max(0.1, \gamma_{tol})$
If $\max(Re(eig(M))) < 0$: $\gamma_{opt} = \gamma$
11. If *Bisection+Scanning* & $2\gamma_{step} < \gamma_{tol}$: Continue as *Scanning,* $\gamma = \gamma_{opt}$
12.1. If *Suboptimal regulator only desired*: goto 13
12.2. If *Bisection* & $2\gamma_{step} \geq \gamma_{tol}$ | *Scanning* & $\gamma > 0$: goto 4
13. Final step: $\gamma = \gamma_{opt}$
14.1. $D_{1*} = \left[\begin{array}{cc} D_{11} & D_{12} \end{array}\right]$
14.2. $D_{*1} = \left[\begin{array}{c} D_{11} \\ D_{21} \end{array}\right]$
14.3. $R_n = {D_{1*}}^T D_{1*} - \left[\begin{array}{cc} \gamma^2 I_{m1} & 0 \\ 0 & 0 \end{array}\right]$

14.4. $\tilde{R}_n = D_{*1}^T D_{*1} - \begin{bmatrix} \gamma^2 I_{p1} & 0 \\ 0 & 0 \end{bmatrix}$

14.5. $H = \begin{bmatrix} A & 0 \\ -C_1^T C_1 & -A^T \end{bmatrix} - \begin{bmatrix} B \\ -C_1^T D_{1*} \end{bmatrix} R_{n-1} \begin{bmatrix} D_{1*}^T C_1 & B^T \end{bmatrix}$

14.6. $J = \begin{bmatrix} A^T & 0 \\ -B_1 B_1^T & -A \end{bmatrix} - \begin{bmatrix} C^T \\ -B_1 D_{*1}^T \end{bmatrix} \tilde{R}_{n-1} \begin{bmatrix} D_{*1}^T B_1^T & C \end{bmatrix}$

14.7. $X = Ric(H)$, $Y = Ric(J)$

15. Construct $\hat{K}(s) \equiv \begin{bmatrix} \hat{A} & \hat{B}_1 \\ \hat{C}_1 & \hat{D}_{11} \end{bmatrix}$:

15.1. $\hat{D}_{11} = -D_{1121} D_{1111}^T (\gamma^2 I - D_{1111} D_{1111}^T)^{-1} D_{1112} - D_{1122}$

15.2. $Z = (I - \gamma^{-2} Y X)^{-1}$

15.3. $\hat{B}_1 = -Z L_2 + Z(B_2 + L_{12})\hat{D}_{11}$

15.4. $\hat{C}_1 = F_2 - \hat{D}_{11}(C_2 + F_{12})$

15.5. $\hat{A} = A + BF - \hat{B}_1(C_2 + F_{12})$

16. $K(s) = V_{12}^T \Sigma_{12}^{-1} \hat{K}(s) \Sigma_{21}^{-1} U_{21} \equiv \begin{bmatrix} A_K & B_K \\ C_K & D_K \end{bmatrix}$

17. $K(s) \leftarrow \begin{bmatrix} A_K - B_K D_{22}(I + D_K D_{22})^{-1} C_K & B_K(I + D_{22} D_K)^{-1} \\ (I + D_K D_{22})^{-1} C_K & D_K(I + D_{22} D_K)^{-1} \end{bmatrix}$

18. $M = F_L(P, K)$ •

3.3 The computational subroutine for H_∞ synthesis SB10BD.f

•***Purpose***. The computational subroutine SB10BD.f performs the K step, i.e., for a given continuous-time state-space description it estimates an optimal H_∞ controller, utilising the Glover&Doyle 1988 formulas (modified according to [PetG1] for increasing the numerical stability); there are three options for approaching γ_{opt} - by bisection, by scanning, or combined (bisection, then scanning) and the procedure is tolerance-controllable. The closed-loop system is also estimated, if exist, and a stability test for it is performed. An information about the condition numbers of the corresponding Riccati equations and normalisation transformations is also outputted, as well as for the gopt achieved.

As an option, a suboptimal controller only can be produced.

The program scheme is nearly the same as of SB10AD.f, with the exception that the closed-loop system is not estimated at each step, nor its stability checked, but the conditions $\rho(XY) < \gamma^2$, $\min(eig(X)) \geq 0$, $\min(eig(Y)) \geq 0$ are tested.

•***Remarks.*** The next SLICOT subroutines are used: SB10PD.f (for normalisation of the original system using orthonormal transformations), SB10QD.f (for estimating the F and L matrices of the suboptimal controller for the current step and the condition numbers of the Riccati equations utilising a scaled Schur method), SB10RD.f (for constructing the suboptimal controller state-space realisation).

•***Additional programs:***

SB10BG.f (gateway function for MATLAB); SB10BG.f estimates also the closed-loop system, so from an user's point of view SB10BG.f is equivalent to the SB10AG.f.

3.4 The subroutine for D step `SB10MD.f`

•***Purpose.*** The computational subroutine `SB10MD.f` performs the D step for continious-time systems, namely, for a given state-space description (of the closed-loop system, produced, for example, by `SB10AD.f`), uncertainty structure and a frequency vector it estimates the frequency response and than D_ω; then $D(s)$ is estimated using block-by-block fitting procedure with the corresponding blocks of D_ω, and the procedure is controlled by a user-supplied tolerance and maximal block order. As an option, the user can demand m(w) only to be estimated, without $D(s)$.

•***Remarks.*** The fitting procedure for the separate blocks the new subroutine `SB10YD.f` is employed. The SLICOT procedures `AB13MD.f` (for obtaining the estimation of the μ- norm and D_ω) and `TB05AD.f` (for estimating the frequency responses).

•***Additional programs:***

`SB10MG.f` (gateway function for MATLAB)

•***Program scheme:***

1. For $M(s) = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$: Estimate $W(j\omega) = D + C(j\omega I - A)^{-1}B$:

1.1. $W(j\omega) \leftarrow (j\omega I - A)^{-1}$

1.2. $W(j\omega) \leftarrow (j\omega I - A)^{-1}B$

1.3. $W(j\omega) \leftarrow C(j\omega I - A)^{-1}B$

1.4. $W(j\omega) \leftarrow C(j\omega I - A)^{-1}B + D$

2. Estimate $\mu_\Delta(M)$, D_ω and exit if desired.

3. $i = 1$ (block counter)

4. $n = 1$ (order of the block i)

5. $d_i(\omega) \overset{order\ n}{\longrightarrow} d_i(s) \equiv \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

6. Estimate $w(j\omega) = d + c(j\omega I - a)^{-1}b$:

6.1. $w(j\omega) \leftarrow (j\omega I - a)^{-1}$

6.2. $w(j\omega) \leftarrow (j\omega I - a)^{-1}b$

6.3. $w(j\omega) \leftarrow c(j\omega I - a)^{-1}b$

6.4. $w(j\omega) \leftarrow c(j\omega I - a)^{-1}b + d$

7.1. $D_{errmean} = \sum\limits_{k=1}^{N_\omega} \frac{d_i(\omega k) - \sqrt{Re(w(j\omega k))^2 + Re(w(j\omega k))^2}}{d_i(\omega k)}$

7.2. $D_{errmax} = \max\limits_k D_{errmean}$

8. If $\frac{D_{errmax}+D_{errmean}}{2} \leq D_{errtol} \quad | \quad n = \max ord \quad | \quad n = N_\omega - 1$: goto 9

else: $n = n + 1$, goto 10

9. Copy a, b, c, d to A_D, B_D, C_D, D_D at the appropriate place, $D(s) \equiv \begin{bmatrix} A_D & B_D \\ C_D & D_D \end{bmatrix}$

10. If $i < N$: $i = i + 1$, goto 4

11. Add I_F , $F = \max(m_2, np_2)$, to $D(s) = \begin{bmatrix} d_1(s) & 0 & . & 0 \\ 0 & d_2(s) & . & 0 \\ . & . & . & . \\ 0 & 0 & . & d_N(s) \end{bmatrix}$:

$$D(s) = \begin{bmatrix} d_1(s) & 0 & . & 0 & 0 \\ 0 & d_2(s) & . & 0 & 0 \\ . & . & . & . & . \\ 0 & 0 & . & d_N(s) & 0 \\ 0 & 0 & . & 0 & I_F \end{bmatrix} .$$

3.5 The computational subroutine for fitting `SB10YD.f`

•***Purpose.*** The computational subroutine `SB10YD.f` fits d_ω with $d(s)$ of order n, described in the state-space for the continious-time or discrete-time cases. As an option it can be demanded the $d(s)$ obtained to be stable and minimal-phase (as it is needed in the D step).

•***Remarks.*** For transforming the $d(s)$ obtained into a stable, minimal phase description, the new program `SB10ZP.f` is called. The SLICOT procedures `DG01MD.f` (for makeing FFT) and `AB04MD.f` (for discrete-to-continious transformation) are also used.

•***Additional programs:***

`SB10YG.f` (gateway function for MATLAB)

•***Program scheme:***

1. If (continious data): bilinear transformation

$$\omega'(k) = a\cos\left(\frac{1-\omega^2(k)/p}{1+\omega^2(k)/p}\right) \quad , \quad k = 1 : N_\omega, \quad p = \sqrt{\omega^2(0)+\omega^2(N_\omega)}$$

else $\omega'(k) = \omega(k)$

2. Linear interpolation:

2.1. $d_\omega(k) \leftarrow \frac{\ln\left|d_\omega(k)\right|}{\ln(10)}$

2.2. $l'_\omega(k) = \frac{(k-1)\pi}{N_\omega}$

2.3. If $l'_\omega(k) < \omega'(1) : ld_\omega(k) = d_\omega(1)$, $k = 1 : N$

else if $l'_\omega(k) \geq \omega'(N_\omega) : ld(k) = d(N_\omega)$, $k = 1 : N$

else : $d_\omega(k) = 0$, $k = 1 : N$

$$2.4. \quad \left.\begin{array}{l} p_1 = \left\lceil \frac{N_\omega l'_\omega(k-1)}{\pi} + 1 \right., p_2 = \left\lceil \frac{N_\omega l'_\omega(k)}{\pi} + 1 \right. \\ \left.\begin{array}{l} r = \frac{l'_\omega(p)-\omega'(k-1)}{\omega'(k)-\omega'(k-1)} \\ ld_\omega(p) = (1-r).d\omega(k-1) + r.d\omega(k) \end{array}\right\} p = p_1 : p_2 \end{array}\right\} k = 2 : N\omega$$

3. $ld_\omega(k) \leftarrow e^{ld\omega(k)\ln(10)}$

$$4.\ \text{Duplicate data:} \quad \left.\begin{array}{l} \omega' \leftarrow \begin{bmatrix} l'_\omega(k) \\ 2\pi - l'_\omega(N_\omega - k) \end{bmatrix} \\ ld_\omega \leftarrow \begin{bmatrix} ld_\omega(k) \\ ld_\omega(N_\omega - k) \end{bmatrix} \end{array}\right\} k = 1 : N$$

5. Complex cepstrum method:

5.1. $ld_\omega(k)$=ln $|d_\omega(k)|$, $k = 1 : 2N$

5.2. $ld_n = F^{-1}\{ld_\omega\}$, FFT, N points

5.3. $ld_n(0) \leftarrow ld_n(0)/2$

5.4. $\hat{x} = e^{F\{ld_n\}}$, FFT, $N/2$ points

6. Interpolate back to the original frequencies:

6.1. $start = 1$, $stop = N_\omega$
6.2. If $\omega'(k) \leq \omega(1)$: $d_\omega(k) = \hat{x}(1)$, $start++$
else if $\omega'(k) \geq \omega(N_\omega/2)$: $d_\omega(k) = \hat{x}(N_\omega/2)$, $stop--$
else : $d_\omega(k) = 0$

6.3. $$\left.\begin{array}{r} p = \min k : \omega(p) \geq \omega'(k) \\ r = \frac{\omega'(k)-\omega(p-1)}{\omega(p)-\omega(p-1)} \\ d_\omega(k) = r.\hat{x}(p) + (1-r)\hat{x}(p-1) \end{array}\right\} k = start : stop$$

7. If (continious data) : $v(k) = e^{\omega'(k)}$, $k = 1 : N_\omega$
else : $v(k) = e^{\omega(k)}$, $k = 1 : N_\omega$
8. ***CASE*** $n > 0$**:**

8.1. Construct $a_0 = \begin{bmatrix} v(1)^n & . & v(1)^2 & v(1) & 1 \\ v(2)^n & . & v(2)^2 & v(2) & 1 \\ . & . & . & . & . \\ v(N_\omega)^n & . & v(N_\omega)^2 & v(N_\omega) & 1 \end{bmatrix}$

8.2. Construct $B_p = \begin{bmatrix} v(1)^n d_\omega(1) \\ . \\ v(N_\omega)^n d_\omega(N_\omega) \end{bmatrix}$

8.3. Construct $A_B = \begin{bmatrix} d_\omega(1)a_0(1,2) & . & d_\omega(1)a_0(1,n+1) \\ . & . & . \\ d_\omega(N_\omega)a_0(1,2) & . & d_\omega(N_\omega)a_0(1,n+1) \end{bmatrix}$

8.4. Construct $A_x = \begin{bmatrix} Re(a_0, A_B) \\ Im(a_0, A_B) \end{bmatrix}$

8.5. Construct $B_x = \begin{bmatrix} Re(B_p) \\ Im(B_p) \end{bmatrix}$

8.6. Solve X: $Bx = AxX$

8.7. Construct $d(s) \equiv \begin{bmatrix} A & B \\ C & D \end{bmatrix}$:

8.7.1. $A = \begin{bmatrix} -X(n+1) & 1 & 0 & . & 0 & 0 \\ 0 & 0 & 1 & . & 0 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & 1 & 0 \\ 0 & 0 & 0 & . & 0 & 1 \\ -X(2n+1) & 0 & 0 & . & 0 & 0 \end{bmatrix}$

8.7.2. $B = \begin{bmatrix} X(2) \\ . \\ X(n+1) \end{bmatrix} - \begin{bmatrix} X(n+2) \\ . \\ X(2n+1) \end{bmatrix} X(1)$

8.7.3. $C = \begin{bmatrix} 1 & 0 & . & 0 \end{bmatrix}$
8.7.5. $D = X(1)$

8.8. If (continious-time data): $d(s) \xrightarrow[\alpha=1,\beta=p]{\substack{discr. \rightarrow cont. \\ bilinear\ transform}} d(s)$
8.9. Constrain $d(s)$ to be stable, minimal phase if desired:
8.9.1. $P_p = poles(d(s))$, $Z_z = zeros(d(s))$
8.9.2. If $Re(P_p(j)) > 0$: $P_p(j) = -P_p(j)$, $j = 1 : n$

8.9.3. If $Re(Z_z(j)) > 0 : \; Z_z(j) = -Z_z(j), \; j = 1 : n$
9. ***CASE*** $n = 0$***:***

9.1. Construct $a_x = \left[\begin{array}{c} 1 \\ . \\ 1 \\ 0 \\ . \\ 0 \end{array}\right] \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} N_\omega \\ \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} N_\omega \end{array}$,

$b_x = \left[\begin{array}{c} Re(d_\omega) \\ Im(d_\omega) \end{array}\right]$
9.2. Solve $b_x = a_x x$
9.3. Construct $d(s) = x$ •

3.6 The `SB10ZP.f` auxuiliary subroutine

•***Purpose***. The role of the subroutine `SB10ZP.f` is to find the poles and the zeros of the given as an input state-space SISO system and to exchange these with positive real parts with their negatives. For handling discrete-time systems they are first transformed into continious-time, and the result obtained is discretized.

It is an auxiliary program to aid `SB10YD.f` in its work.

•***Remarks.*** The following SLICOT subroutines are used: `AB04MD.f` (for an eventual discrete-to-continious transformation and back), `AB07ND.f` (for obtaining an inverse system), `MC01PD.f` (for the transformation state-space -> transfer function), `TD04AD.f` (for estimating the minimal realization in the state-space).

•***Program scheme:***

1. If (discrete-time data) : $P \xrightarrow[]{\substack{discr \rightarrow cont. \\ bilinear\ transform \\ \alpha = 1, \beta = 1}} P$, where $P \equiv \left[\begin{array}{cc} A & B \\ C & D \end{array}\right]$
2. Scaling factors:
2.1. $Scal_D = D$
2.2. $Scal_C = \sqrt{|Scal_D|}$
2.3. If $Scal_D < 0 : \; Scal_B = -Scal_C$
else : $Scal_B = Scal_C$
3. Solve $P_p = eig(A)$
4. $P_i = P^{-1}$, where $P \equiv \left[\begin{array}{cc} A_i & B_i \\ C_i & D_i \end{array}\right]$
5. Solve $Z_z = eig(A_i)$
6.1. If $Re(P_p(j)) > 0 : \; P_p(j) = -P_p(j), \; j = 1 : n$
6.2. If $Re(Z_z(j)) > 0 : \; Z_z(j) = -Z_z(j), \; j = 1 : n$
7.1. Determine $P(s) = (s - P_p(1)).(s - P_p(2))..(s - P_p(n))$
7.2. Determine $Q(s) = (s - Z_z(1)).(s - Z_z(2))..(s - Z_z(m)$
8. Rearrange coefficients of $P(s)$, $Q(s)$ in powers of s^{-1}
9. $P \xleftarrow{minimal\ realisation} T(s)$, where $P \equiv \left[\begin{array}{cc} A & B \\ C & D \end{array}\right]$

10.1. $B = B.Scal_B$
10.2. $C = C.Scal_C$
10.3. $D = Scal_D$

11. If (discrete-time data): $P \xrightarrow{\substack{cont. \to discr. \\ bilinear\ transform \\ \alpha = 1, \beta = 1}} P$ •

4 A test example

Some tests are made with the new software, comparing its work with that of MATLAB. They involve system descriptions of order up to 28 and make an evidence that the new software behave well, and in some cases better in the terms of features of the controller estimated than the existing software in MATLAB; as for the speed of the work, estimating the controller is from twice faster (for μ controller) to 7-8 times faster (for H_∞ controller only). A drawback is the order of the controller, which is higher than MATLAB's due to the worse criterium of estimating how precise the D scaling system obtained is.

Here a test example is given which demonstrates H_∞ and μ- synthesis with `SB10DK.f`. Although the `SB10DK.f`'s mexfile could be called directly, it is more convenient the files `slihinf.m` and `slimju.m` to be used instead. For more information, read the help included in their code.

4.1 Data

```
a =

 [-1.0,  0.0,  4.0,  5.0, -3.0, -2.0;

  -2.0,  4.0, -7.0, -2.0,  0.0,  3.0;

  -6.0,  9.0, -5.0,  0.0,  2.0, -1.0;

  -8.0,  4.0,  7.0, -1.0, -3.0,  0.0;

   2.0,  5.0,  8.0, -9.0,  1.0, -4.0;

   3.0, -5.0,  8.0,  0.0,  2.0, -6.0]


b =

 [-3.0, -4.0  -2.0,  1.0,  0.0;

   2.0,  0.0,  1.0, -5.0,  2.0;

  -5.0, -7.0,  0.0,  7.0, -2.0;

   4.0, -6.0,  1.0,  1.0, -2.0;

  -3.0,  9.0, -8.0,  0.0,  5.0;

   1.0, -2.0,  3.0, -6.0, -2.0]


c =
 [ 1.0, -1.0,  2.0, -4.0,  0.0, -3.0;
  -3.0,  0.0,  5.0, -1.0,  1.0,  1.0;
  -7.0,  5.0,  0.0, -8.0,  2.0, -2.0;
   9.0, -3.0,  4.0,  0.0,  3.0,  7.0;
   0.0,  1.0, -2.0,  1.0, -6.0, -2.0]

d =
 [ 1.0, -2.0, -3.0,  0.0,  0.0;
   0.0,  4.0,  0.0,  1.0,  0.0;
   5.0, -3.0, -4.0,  0.0,  1.0;
   0.0,  1.0,  0.0,  1.0, -3.0;
   0.0,  0.0,  1.0,  7.0,  1.0];

ncon = 2;             %the number of control inputs
nmeas = 2;            %the number of measurements
gamma = 100;          %the initial value of gamma
sys = pck(a,b,c,d)    %to get the system representation in the manner of mu-toolbox
```

4.2 H_∞ synthesis

```
job=1; discfl=0       %bisection method, continuous-time system

[sysk,gmin,rcond] = slihinf(job,discfl,sys,ncon,nmeas,gamma)

sysk:
ak =
 [-6.3e+02,  3.5e+03,  2.8e+03,  1.4e+03,  1.7e+03,  1.5e+03;
  -2.2e+02,  1.2e+03,  9.9e+02,  4.8e+02,  5.9e+02,  5.3e+02;
  -7.0e+02,  3.8e+03,  3.1e+03,  1.5e+03,  1.8e+03,  1.6e+03;
  -2.9e+02,  1.5e+03,  1.2e+03,  5.7e+02,  7.2e+02,  6.2e+02;
  -9.3e+02,  5.2e+03,  4.3e+03,  2.1e+03,  2.6e+03,  2.2e+03;
   2.3e+02, -1.3e+03, -1.1e+03, -5.3e+02, -6.4e+02, -5.7e+02]

bk =
 [-2.2e-01, -1.1e-01;
  -8.5e-01, -6.5e-01;
   8.2e-01,  5.8e-01;
   8.4e-02,  1.1e-02;
  -5.6e-01, -2.5e-01;
   6.8e-03, -7.6e-01]

ck =
 [-2.0e+00, 9.3e+00, 7.7e+00, 3.9e+00, 5.2e+00, 4.0e+00;
  -3.4e+01, 2.0e+02, 1.7e+02, 8.0e+01, 9.9e+01, 8.8e+01]

dk =
 [ 5.5e-02, 1.3e-01;
  -3.2e-01, 3.3e-0]

gmin = 10.1847
rcond =
[1.0000;
1.0000;
0.0104;
0.0001]
```

4.3 μ- synthesis

```
job=1; discfl=0              %bisection method, continuous-time system
w=logspace(-1,3,20); w=w'    %the frequency vector
nblock=[1;1;1]               %the number of uncertainty blocks
itype=[2;2;2]                %the type of uncertainties
maxord=3                     %the max. order of d subblocks
qutol=0.1                    %the tolerance for the accuracy of D step

[sysk,mu,rcond] = slimju(job,discfl,sys,ncon,nmeas,gamma,w,nblock,itype,maxord,qutol)

sysk:
```

```
ak =

 [ 9.2e+01,  3.2e+01,  1.1e+02,  7.6e+02, -9.5e+01, -1.6e+02,  1.9e-04,  2.5e+03,
  -2.2e+04, -1.6e+04, -7.5e+03, -8.6e+03, -8.4e+03, -9.9e+03,  4.5e+03,  1.7e+01,
   5.1e+03,  1.7e+03, -7.1e+02, -6.1e-04;
  -6.0e+01, -1.7e+01, -7.0e+01, -4.1e+02,  5.1e+01,  8.4e+01, -1.0e-04, -1.3e+03,
   1.2e+04,  8.9e+03,  4.0e+03,  4.6e+03,  4.5e+03,  5.4e+03, -2.5e+03,  1.9e+01,
  -2.7e+03, -8.9e+02,  3.4e+02,  3.2e-04;
   3.7e+00,  9.0e+01, -2.3e+02, -3.1e+02,  3.1e+01,  8.6e+01, -6.3e-05, -1.6e+03,
   8.3e+03,  6.3e+03,  2.5e+03,  3.6e+03,  3.6e+03,  3.5e+03, -1.7e+03,  1.5e+02,
  -2.1e+03, -5.1e+02,  3.9e+02,  2.4e-04;
  -5.7e+01, -2.0e+01, -6.4e+01, -4.6e+02,  5.8e+01,  1.2e+02, -1.1e-04, -1.5e+03,
   1.3e+04,  1.0e+04,  4.5e+03,  5.2e+03,  5.1e+03,  6.1e+03, -2.8e+03, -7.8e+00,
  -3.1e+03, -1.0e+03,  4.4e+02,  3.7e-04;
   4.5e+01,  1.6e+01,  4.6e+01,  3.6e+02, -4.6e+01, -8.4e+01,  9.1e-05,  1.2e+03,
  -1.1e+04, -8.0e+03, -3.6e+03, -4.2e+03, -4.0e+03, -4.8e+03,  2.2e+03, -2.2e+01,
   2.5e+03,  8.0e+02, -2.8e+02, -2.9e-04;
   1.4e+02,  4.6e+01,  1.6e+02,  1.1e+03, -3.7e+01, -3.9e+02,  2.7e-04,  3.9e+03,
  -3.2e+04, -2.4e+04, -1.1e+04, -1.2e+04, -1.2e+04, -1.4e+04,  6.5e+03,  5.3e+01,
   7.3e+03,  2.4e+03, -1.3e+03, -8.7e-04;
   1.1e-07,  3.7e-08,  1.2e-07,  8.8e-07, -1.1e-07, -1.9e-07, -1.0e+01,  2.8e-06,
  -2.5e-05, -1.9e-05, -8.8e-06, -9.8e-06, -9.6e-06, -1.1e-05,  5.3e-06,  1.2e-07,
   5.8e-06,  1.9e-06, -8.7e-07, -7.1e-13;
  -1.4e+01, -4.8e+00, -1.5e+01, -1.1e+02,  1.4e+01,  2.3e+01, -2.8e-05, -3.7e+02,
   3.3e+03,  2.5e+03,  1.1e+03,  1.3e+03,  1.2e+03,  1.5e+03, -6.8e+02, -1.8e+00,
  -7.6e+02, -2.5e+02,  9.5e+01,  9.0e-05;
  -5.1e+00, -1.8e+00, -6.1e+00, -4.3e+01,  5.3e+00,  9.2e+00, -1.1e-05, -1.4e+02,
   1.2e+03,  9.3e+02,  4.2e+02,  4.9e+02,  4.8e+02,  5.6e+02, -2.6e+02,  2.4e+00,
  -2.9e+02, -9.3e+01,  3.9e+01,  3.4e-05;
  -1.6e+01, -5.4e+00, -1.7e+01, -1.3e+02,  1.6e+01,  2.6e+01, -3.1e-05, -4.3e+02,
   3.7e+03,  2.8e+03,  1.3e+03,  1.4e+03,  1.4e+03,  1.7e+03, -7.6e+02, -4.8e+00,
  -8.5e+02, -2.8e+02,  1.1e+02,  1.0e-04;
  -1.4e+01, -4.9e+00, -1.5e+01, -1.2e+02,  1.5e+01,  2.4e+01, -2.9e-05, -4.0e+02,
   3.4e+03,  2.5e+03,  1.1e+03,  1.3e+03,  1.3e+03,  1.5e+03, -7.0e+02,  8.9e+00,
  -7.8e+02, -2.5e+02,  9.3e+01,  9.2e-05;
  -4.5e+00, -1.2e+00, -4.8e+00, -2.7e+01,  3.6e+00,  5.9e+00, -7.0e-06, -5.9e+01,
   7.8e+02,  6.0e+02,  2.9e+02,  3.0e+02,  3.1e+02,  3.6e+02, -1.6e+02, -1.5e+01,
  -1.8e+02, -6.2e+01,  4.1e+01,  2.2e-05;
   1.0e+00,  2.3e-01,  9.5e-01,  4.7e+00, -6.7e-01, -9.0e-01,  1.3e-06,  9.7e+00,
  -1.4e+02, -9.9e+01, -5.9e+01, -5.5e+01, -6.4e+01, -6.7e+01,  2.9e+01,  5.0e+00,
   3.1e+01,  1.2e+01, -8.5e+00, -4.0e-06;
   4.5e-01,  2.3e-01,  8.4e-01,  5.8e+00, -6.8e-01, -1.0e+00,  1.3e-06,  2.2e+01,
  -1.7e+02, -1.2e+02, -4.8e+01, -6.8e+01, -6.3e+01, -7.3e+01,  3.4e+01, -7.7e-01,
   3.9e+01,  1.2e+01, -2.5e+00, -4.2e-06;
  -1.1e+01, -3.6e+00, -1.2e+01, -8.4e+01,  1.1e+01,  1.7e+01, -2.1e-05, -2.7e+02,
   2.4e+03,  1.8e+03,  8.4e+02,  9.4e+02,  9.2e+02,  1.1e+03, -5.0e+02, -2.5e+01,
  -5.6e+02, -1.8e+02,  7.8e+01,  6.7e-05;
   4.3e+01,  1.5e+01,  4.8e+01,  3.5e+02, -4.3e+01, -7.3e+01,  8.6e-05,  1.1e+03,
  -1.0e+04, -7.5e+03, -3.4e+03, -3.9e+03, -3.8e+03, -4.5e+03,  2.2e+03, -8.5e+01,
   2.3e+03,  7.6e+02, -3.2e+02, -2.8e-04;
   1.1e+01,  3.9e+00,  1.3e+01,  9.2e+01, -1.1e+01, -2.0e+01,  2.3e-05,  3.0e+02,
  -2.6e+03, -2.0e+03, -9.0e+02, -1.0e+03, -1.0e+03, -1.2e+03,  5.5e+02,  1.2e+00,
   6.1e+02,  2.0e+02, -7.2e+01, -7.3e-05;
  -1.2e+01, -4.0e+00, -1.2e+01, -9.5e+01,  1.2e+01,  1.9e+01, -2.3e-05, -3.1e+02,
   2.7e+03,  2.1e+03,  9.1e+02,  1.1e+03,  1.0e+03,  1.2e+03, -5.7e+02,  5.0e+00,
  -6.4e+02, -2.1e+02,  6.5e+01,  7.4e-05;
  -3.1e+01, -1.0e+01, -3.6e+01, -2.5e+02,  3.1e+01,  5.5e+01, -6.1e-05, -8.2e+02,
   7.1e+03,  5.4e+03,  2.4e+03,  2.8e+03,  2.7e+03,  3.2e+03, -1.5e+03, -9.1e+00,
  -1.6e+03, -4.4e+02,  1.5e+02,  2.0e-04;
  -6.5e-09, -1.8e-09, -5.2e-09, -4.1e-08,  5.4e-09,  7.5e-09, -1.1e-14, -1.2e-07,
   1.2e-06,  9.0e-07,  4.3e-07,  4.6e-07,  4.6e-07,  5.6e-07, -2.5e-07, -6.3e-09,
  -2.7e-07, -9.5e-08,  3.5e-08, -1.0e+01]

bk =
 [ 6.3e-01,  2.2e+00;
  -2.1e+00, -1.6e+00;
  -1.9e+01,  8.7e+00;
   6.9e-02, -4.5e-01;
   2.5e+00,  2.1e+00,
  -6.2e+00, -9.2e+00;
  -7.7e-09,  3.3e-09;
  -2.1e-01, -3.9e-01;
  -1.0e+00, -6.8e-01;
   7.8e-01,  3.0e-01;
  -4.5e-03, -3.4e-01;
  -6.9e-01, -3.6e-02;
   4.8e-02, -8.0e-01;
   3.1e-01, -3.1e-03;
   2.5e-01, -4.2e-01;
  -6.3e-01,  1.3e+00;
   7.0e-02,  2.8e-01;
  -5.5e-01, -3.7e-01;
   3.3e-01, -3.6e-01;
   4.2e-10, -5.5e-10;

ck =
 [ 2.4e-03,  5.3e-03,  1.4e-02,  1.5e-01, -1.6e-02, -3.7e-02,  3.2e-08,  2.7e-01,
  -4.1e+00, -3.1e+00, -1.2e+00, -1.1e+00, -1.6e+00, -1.8e+00,  8.7e-01, -1.2e-01,
   1.0e+00,  2.7e-01, -1.1e-01, -1.1e-07;
  -4.3e-01, -1.4e-01, -4.6e-01, -3.4e+00,  4.2e-01,  7.1e-01, -8.3e-07, -8.1e+00,
   9.6e+01,  7.4e+01,  3.3e+01,  3.9e+01,  3.9e+01,  4.4e+01, -2.0e+01, -5.7e-02,
  -2.2e+01, -7.4e+00,  3.3e+00,  2.7e-06]

dk =
 [ 5.5e-02, 1.3e-01;
  -3.2e-01, 3.3e-02]

mu =
 [8.7719;
  8.8651;
  8.9864;
  9.0942;
  9.1437;
  9.1765;
  9.2921;
  9.3955;
  9.1340;
  8.7096;
  8.8793;
  9.1135;
  9.3620;
  8.8837;
  8.8738;
  9.3385;
  9.2929;
  9.1111;
  8.9183;
  8.6801]

rcond =
 [1.0000;
  1.0000;
  0.0104;
  0.0001;
  0.6006;
  0.6006;
  0.0008;
  0.0001]
```

5 Bibliography

[AndBBD95] E. Anderson, Z. Bai, C.H. Bischof, J.M. Demmel, J.J. Dongarra, J.J. DuCroz, A. Greenbaum, S.J. Hammarling, A. McKenney, S. Ostrouchov and D.C. Sorensen (1992). *LAPACK Users's Guide.* SIAM, Philadelphia.

[BalD1] Balas G., J. Doyle, K. Glover, A. Packard, R. Smith (1998). *μ-Analysis and Synthesis Toolbox - User's Guide*, The Mathworks, Inc., MUSYN, Inc.

[BogH1] Bogert B. P., M. J. R. Healy, J. W. Tukey (1963). *The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe-cracking*, Proc. Symp.

Time-Series Analysis, M. Rosenblatt, New York, J. Wiley&Sons, Inc., New York, pp. 209-243.

[DonD88] J.J. Dongarra, J. DuCroz, S. Hammarling and R.J. Hanson (1988). An extended set of Fortran basic linear algebra subprograms. *ACM Trans. Math. Software*, vol. 14, pp. 1–17.

[DonD90] J.J. Dongarra, J. Du Croz, I. Duff and S. Hammarling (1990). A set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Software*, vol. 16, pp. 1–1.

[GloD1] Glover K., J. Doyle (1988). *State-space formulae for all stabilizing controllers that satisfy an H_∞ norm bound and relations to risk sensitivity*, Systems and Control Letters, vol. 11, pp. 167-172.

[GuP1] Gu D.-W., P. Petkov, M. Konstantinov (1999). *An introduction to H_∞ optimisation designs*, NICONET report, No 99-7.

[LawHKK79] C. Lawson R. Hanson, D. Kinkaid, F. Krogh (1979). Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software*, vol. 5, pp. 308–323.

[OppS1] Oppenheim A., R. Schafer (1975). *Digital Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey.

[PetG1] Petkov P., D. Gu, M. Konstantinov (1998). *Fortran 77 routines for H_∞ and H_2 design of continuous-time linear control systems*, NICONET Report 98-14.

[SafG1] Safonov M., K. Goh, J. Ly (1994). *Control system synthesis via bilinear matrix inequalities*, Proc. ACC, Baltimore, 1994.

[ZhoD1] Zhou K., J. Doyle, K. Glover (1996). *Robust and Optimal Control*, Prentice Hall, Englewood Cliffs, New Jersey 07632.

Appendix. M-files for calling `SB10DK.f` mex-function

`slihinf.m`

```
function [sysk,gmin,rcond] = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol,actol)
%
%  This program computes the H_inf optimal controller
%  given the model in a state space.
%  The discrete-time systems are handled via bilinear transformation
%  to continuous-time and then the controller obtained is discretised.
%  It employes the SB10DK.f mex-function.
%--------------------------------------------------------------------
% Usage:
%
% sysk = slihinf(job,discfl,sys,ncon,nmeas,gmax)
% sysk = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol)
% sysk = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol,actol)
%
% [sysk,gmin] = slihinf(job,discfl,sys,ncon,nmeas,gmax)
% [sysk,gmin] = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol)
% [sysk,gmin] = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol,actol)
%
% [sysk,gmin,rcond] = slihinf(job,discfl,sys,ncon,nmeas,gmax)
% [sysk,gmin,rcond] = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol)
% [sysk,gmin,rcond] = slihinf(job,discfl,sys,ncon,nmeas,gmax,gtol,actol)
%--------------------------------------------------------------------
% Input parameters:
%  job     Indicates the strategy for reducing the gamma value:
%          = 1: Use bisection method for decreasing gamma from gmax
%               to gmin until the closed-loop system leaves
%               stability.
%          = 2: Scan from gmax to 0 trying to find the minimal gamma
%               for which the closed-loop system retains stability.
%          = 3: First bisection, then scanning.
%          = 4: Suboptimal H_inf controller only.
%  discfl  Indicates the type of the system, as follows:
%          = 0: continuous-time system;
%          = 1: discrete-time system.
%  sys     contains the matrices A,B,C,D of the system in a packed
%          form.
%  ncon    the number of control inputs (M2). NP-NMEAS >= NCON >= 0.
%  nmeas   the number of measurements (NP2). M-NCON >= NMEAS >= 0.
%  gmax    The initial value of gamma on input. It is assumed that
%          gmax is sufficiently large so that the controller is
%          admissible. gmax >= 0.
%  gtol    (optional) tolerance used for controlling the accuracy
%          of gmin and its distance to the estimated minimal possible
%          value of gamma.
%          If gtol <= 0, then sqrt(eps) is used, where eps
%          is the relative machine precision.
%          The default is gtol = 0.01
%  actol   (optional) upper bound for the poles of the closed-loop
%          system used for determining if it is stable.
%          actol <= 0 for stable systems.
%          The default is actol=0
%
% Output parameters:
%  sysk    contains the matrices Ak,Bk,Ck,Dk of the controller in a
%          packed form.
%  gmin    the minimal gamma found.
%  rcond   rcond(1) contains the reciprocal condition number of the
%                   control transformation matrix;
%          rcond(2) contains the reciprocal condition number of the
%                   measurement transformation matrix;
%          rcond(3) contains an estimate of the reciprocal condition
%                   number of the X-Riccati equation;
%          rcond(4) contains an estimate of the reciprocal condition
%                   number of the Y-Riccati equation.
% --------------------------------------------------------------------
% Contributor:
%   A. Markovski, Technical University of Sofia, October 2003
%---------------------------------------------------------------------
%
if (nargin<6 | nargin>8 | nargout<1 | nargout>3)
    error('Usage: [sysk,{gmin,{rcond}}] = ...
        slihinf(job,discfl,sys,ncon,nmeas,gmax,{gtol,{actol}})')
end
    [a,b,c,d] = unpck(sys);
    job = job + 3;
%
if (nargin==6)
    [ak,bk,ck,dk,gmin,rcond] = sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax);
elseif (nargin==7)
    [ak,bk,ck,dk,gmin,rcond] = sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,gtol);
elseif (nargin==8)
    [ak,bk,ck,dk,gmin,rcond] = sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,gtol,actol);
end
```

```
%
sysk = pck(ak,bk,ck,dk);
```

slimju.m

```
function [sysk,mju,rcond] = ...
    slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol,actol)
%
%  This program computes the mu optimal controller
%  given the model in a state space. It also outputs the mu norm
%  of the closed loop system.
%  The discrete-time systems are handled via bilinear transformation
%  to continuous-time and then the controller obtained is discretised.
%  It employes the SB10DK.f mex-function.
%-------------------------------------------------------------------------------
% Usage:
%
% sysk = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord)
% sysk = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol)
% sysk = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol)
% sysk = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol,actol)
%
% [sysk,mju] = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord)
% [sysk,mju] = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol)
% [sysk,mju] = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol)
% [sysk,mju] = ...
%    slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol,actol)
%
% [sysk,mju,rcond] = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord)
% [sysk,mju,rcond] = slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol)
% [sysk,mju,rcond] = ...
%    slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol)
% [sysk,mju,rcond] = ...
%    slimju(job,discfl,sys,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol,actol)
```

```
%------------------------------------------------------------------------------------------
% Input parameters:
%  job     Indicates the strategy for reducing the gamma value:
%          = 1: Use bisection method for decreasing gamma from gmax
%               to gmin until the closed-loop system leaves
%               stability.
%          = 2: Scan from gmax to 0 trying to find the minimal gamma
%               for which the closed-loop system retains stability.
%          = 3: First bisection, then scanning.
%  discfl  Indicates the type of the system, as follows:
%          = 0: continuous-time system;
%          = 1: discrete-time system.
%  sys     contains the matrices A,B,C,D of the system in a packed form.
%  ncon    the number of control inputs (M2). NP-NMEAS >= NCON >= 0.
%  nmeas   the number of measurements (NP2). M-NCON >= NMEAS >= 0.
%  gmax    The initial value of gamma on input. It is assumed that
%          gmax is sufficiently large so that the controller is
%          admissible. gmax >= 0.
%  w       the vector with the frequencies.
%          They must be nonnegative, in increasing order, and
%          for discrete-time systems between 0 and pi.
%  nblock  the vector with the block structure of the uncertainty.
%          nblock(i) is the size of each block.
%  itype   the vector of the same length as NBLOCK indicating
%          the type of each block.
%          itype(i) = 1 indicates that the corresponding block is a
%          real block. THIS OPTION IS NOT SUPPORTED NOW.
%          itype(i) = 2 indicates that the corresponding block is a
%          complex block. THIS IS THE ONLY ALLOWED VALUE NOW!
%  maxord  the MAX order of EACH block in the D-fitting procedure.
%          1<=ORD
%  qutol   (optional) the acceptable mean relative error between
%          the D(jw) and the frequency responce of the estimated block
%          [ADi,BDi;CDi,DDi]. When it is reached, the result is
%           taken as good enough.
%           The default is qutol = 2
%  gtol     (optional) tolerance used for controlling the accuracy
%           of gmin and its distance to the estimated minimal possible
%           value of gamma.
%           If gtol <= 0, then sqrt(eps) is used, where eps
%           is the relative machine precision.
%           The default is gtol = 0.01
%  actol    (optional) upper bound for the poles of the closed-loop
%           system used for determining if it is stable.
%           actol <= 0 for stable systems.
%           The default is actol = 0
%
% Output parameters:
%  sysk     contains the matrices Ak,Bk,Ck,Dk of the controller in a packed form.
%  mju      the vector with the estimated upper bound of the structured
%           singular value for each w for the closed-loop system.
%  rcond    for each successful j-th K step:
%           rcond(j) contains the reciprocal condition number of the
%                    control transformation matrix;
%           rcond(j+1) contains the reciprocal condition number of the
%                    measurement transformation matrix;
%           rcond(j+2) contains an estimate of the reciprocal condition
%                    number of the X-Riccati equation;
%           rcond(j+3) contains an estimate of the reciprocal condition
%                    number of the Y-Riccati equation.
% ----------------------------------------------------------------------
% Contributor:
%   A. Markovski, Technical University of Sofia, October 2003
%-----------------------------------------------------------------------
%
if (nargin<10 | nargin>13 | nargout<1 | nargout>3)
    error('Usage: [sysk,{mju,{rcond}}] = ...
    slimju(job,discfl,sys,ncon,nmeas,gmax,w,BK,T2,maxord,{qutol,{gtol,{actol}}})')
```

```
end
    [a,b,c,d] = unpck(sys);
%
if (nargin==10)
    [ak,bk,ck,dk,mju,rcond] = ...
    sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,w,nblock,itype,maxord);
elseif (nargin==11)
    [ak,bk,ck,dk,mju,rcond] = ...
    sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol);
elseif (nargin==12)
    [ak,bk,ck,dk,mju,rcond] = ...
    sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol);
elseif(nargin==13)
    [ak,bk,ck,dk,mju,rcond] = ...
    sb10dk(job,discfl,a,b,c,d,ncon,nmeas,gmax,w,nblock,itype,maxord,qutol,gtol,actol);
end
%
sysk = pck(ak,bk,ck,dk);
```