Frequency-Domain Robust Controller Design

A Toolbox for MATLAB

User Manual

Version 1.0

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Automatic Control Laboratory
EPFL, Switzerland
May 2012

Contents

Chapter 1

Introduction

The Frequency-Domain Robust Controller Design Toolbox is a tool for designing robust linearly parameterized controllers in the Nyquist diagram. It can be used to design linearly parameterized controllers of any order for parametric models or nonparametric models obtained for example by the identification toolbox of MATLAB. The robust controllers are designed in terms of H_∞ performance or classical robustness margins such as the gain and phase margin, for single/multi-model, SISO/MIMO systems. The toolbox also supports designing gain-scheduled controllers.

In all of design cases, linear or convex optimization problems are solved. For linear and quadratic optimization the well-known `linprog` or `quadprog` (depending on the problem) commands of the Optimization toolbox of MATLAB are used. While convex optimization problems are formulated with YALMIP and can be solved with all available solvers (e.g. `sdpt3` is used as a default solver) . Many commands of the Control toolbox of MATLAB are used as well.

In this manual as well as describing theoretical bases of the optimization problems in a terse manner, our main attempt is to provide a quite comprehensive set of examples to exhibit wide functionality, yet user-friendliness of this tool. The theoretical bases used in this toolbox have been published in [4, 5, 3, 2]. The development of the toolbox was started by the works of Vinicius de Oliveira and continued by Mehdi Sadeghpoor during an internship in Automatic Control Laboratory [6].

In this chapter, the theoretical bases of the design methods used in the toolbox are presented. The description of the optimization problems that are solved in the toolbox for different desired control performance (GPhC, loop shaping, and H_∞), is the main subject of this section. The extension of these methods to MIMO systems and gain-scheduled controller design, is described. In the sequel, first the class of controllers and models are presented

and then different control performances that can be considered by the toolbox are given.

1.1 Class of models and controllers

1.1.1 Models

Models can be parametric or nonparametric. The design method, in fact, needs the frequency response of the plant model in a finite number of frequencies which can be obtained directly from data by spectral analysis (for example by the identification toolbox of Matlab) or computed from a parametric model. Therefore, high order models with pure time delay and non minimum phase zeros can be considered with no approximation. Thus, we define the set of models: $M = \{G_i(j\omega),\ i = 1, \ldots, m\}$ where $G_i(j\omega)$ can be a scalar or a matrix function representing the frequency response of a SISO or MIMO model, respectively, over a vector of frequency points $\omega_i = [\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{N_i}}]$. N_i is large enough to give a good approximation of the frequency response of the system G_i. The methods are described on a SISO system. It will also be explained how these methods can be applied in designing multivariable decoupling controllers for MIMO systems.

1.1.2 Linearly Parameterize Controllers

The toolbox designs linearly parameterized controllers. A linearly parameterized controller has the following form:

$$\rho^T \phi(s) = [\rho_1 \ \ \rho_2 \ \ \ldots \ \ \rho_n] \times [\phi_1(s) \ \ \phi_2(s) \ \ \ldots \ \ \phi_n(s)]^T \tag{1.1}$$

where ρ_1, ρ_2, ..., ρ_n are controller parameters and $\phi_1(s)$, $\phi_2(s)$, ..., $\phi_n(s)$ are basis transfer functions. These transfer functions must be stable, i.e. with no right half plane poles.

PID, PI, PD:

The proportional-integral-derivative (PID) controller is a familiar case of these types of controllers with 3 parameters $[\rho_1 \ \ \rho_2 \ \ \rho_3] = [k_p \ \ k_i \ \ k_d]$ and the vector of basis transfer functions:

$$\phi(s) = [1 \ \ \frac{1}{s} \ \ \frac{s}{1+\tau s}]^T \tag{1.2}$$

for continuous-time systems and:

$$\phi(z) = [1 \ \ \frac{z}{z-1} \ \ \frac{z-1}{z}]^T \tag{1.3}$$

for discrete-time systems, where τ is the time constant of the derivative part in the continuous-time one. PI and PD are special cases of the above equations. Besides these common linearly parameterized controllers, this toolbox supports the following higher order types, too:

Laguerre basis functions:

$$\phi_1(s) = 1, \quad \phi_i(s) = \frac{\sqrt{2\xi}(s-\xi)^{i-2}}{(s+\xi)^{i-1}} \quad i = 2, 3, \ldots, n+1 \tag{1.4}$$

for continuous-systems and:

$$\phi_1(z) = 1, \quad \phi_i(z) = \frac{\sqrt{1-a^2}}{z-a}\left(\frac{1-az}{z-a}\right)^{i-2} \quad i = 2, 3, \ldots, n+1 \tag{1.5}$$

for discrete-time systems, where n is the order of the controller, $\xi > 0$, and $-1 < a < 1$.

Generalized orthonormal basis functions:

$$\phi_1(s) = 1, \quad \phi_i(s) = \frac{\sqrt{2R_e(\xi_{i-1})}}{s+\xi_{i-1}} \prod_{k=1}^{i-2} \frac{s-\bar{\xi_k}}{s+\xi_k} \quad i = 2, 3, \ldots, n+1 \tag{1.6}$$

for continuous-time systems and

$$\phi_1(z) = 1, \quad \phi_i(z) = \frac{\sqrt{1-|\xi_{i-1}|^2}}{z-\xi_{i-1}} \prod_{k=1}^{i-2} \frac{1-\bar{\xi_k}z}{z-\xi_k} \quad i = 2, 3, \ldots, n+1 \tag{1.7}$$

for discrete-time systems, where $\xi_1, \ldots, \xi_n$ are complex numbers, R_e denotes the real part of a complex number, and $\bar{\xi_k}$ is the complex conjugate of ξ_k.

As stated earlier, one can define one's own vector of basis functions, ϕ, of any desired order as well as the above listed types.

1.1.3 Smith Predictor Controllers

A Smith Predictor controller has the form shown in Fig. 1.1, where $P_i(s)$ is the plant model, $P_n(s)$ is the nominal plant model, $G_n(s)$ is the nominal, delay-free plant model and $C(s)$ is the controller. The open-loop transfer function $L_i(s) = [P_i(s) + H(s)]\, C(s)$, where $H(s) = G_n(s) - P_n(s)$. $C(s)$ is a linearly parameterized controller, as described above.

The Smith Predictor structure can be used for stable SISO and MIMO systems with time delays. For unstable SISO systems, $H(s)$ should be modified such that it is stable in order to compute a stabilizing controller.

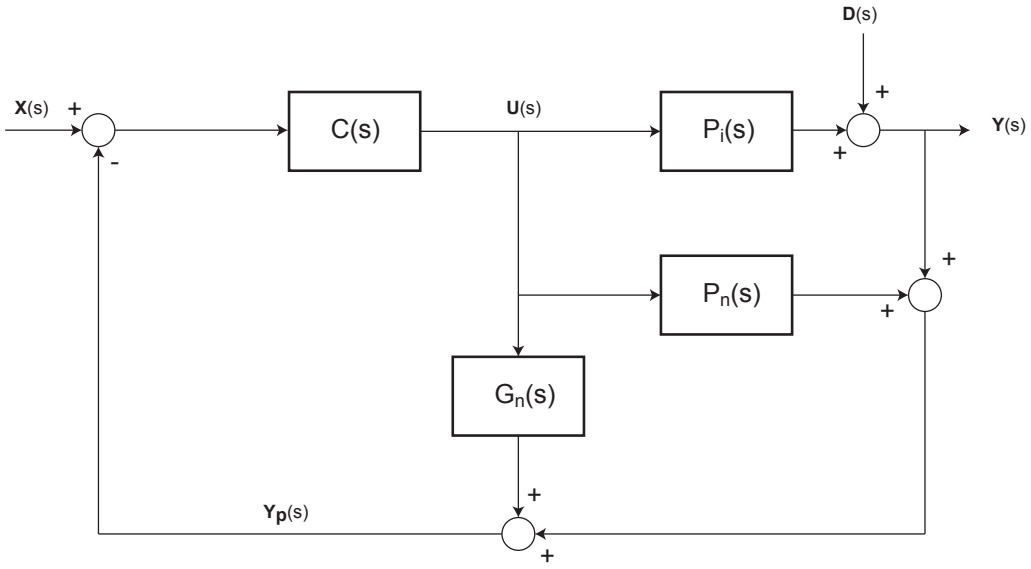


Figure 1.1: Smith Predictor structure

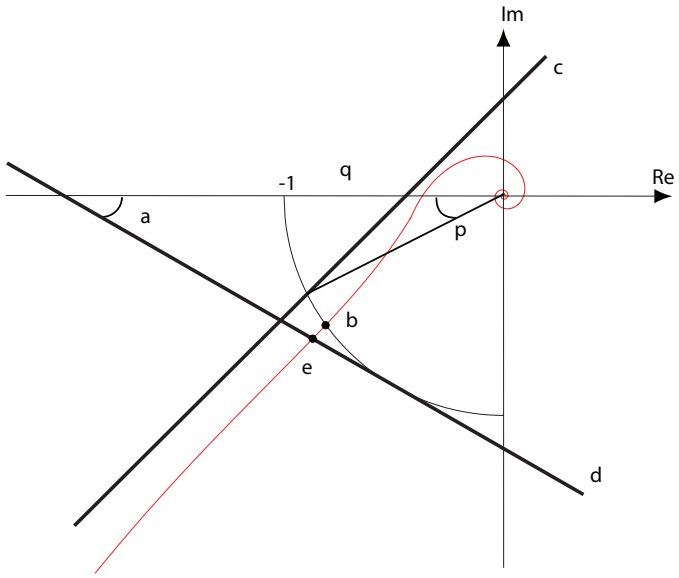


Figure 1.2: GPhC specifications converted to linear constraints in Nyquist diagram

1.2 GPhC controller

Gain margin, phase margin and crossover frequency (GPhC) are typical performance specifications for PID controller design in industry. We use these specifications for SISO stable systems if the number of integrators in the open-loop transfer function is less than or equal to 2. Specifying the gain and phase margin defines a straight line in the Nyquist diagram (see d_1 in Fig. 1.2). Now, if the Nyquist curve of the open loop system lies in the right side of d_1 the desired values for the gain margin g_m and phase margin ϕ_m will be assured. This can be represented by a set of linear constraints thanks to the linear parameterization of the controller. Now, consider another straight line d_2 which is tangent to the middle of the unit circle in the sector created by d_1 and the imaginary axis. If we call ω_x the frequency at which the Nyquist curve intersects d_2, a crossover frequency greater than or equal to ω_x can be achieved by satisfying a set of linear constraints. In fact, for frequencies greater than ω_x the Nyquist curve should lie below d_1 and above d_2 while for frequencies less than ω_x it should lie below d_2.

Let us define the set of all points in the complex plane on the line d by $f(x+iy, d) = 0$. Assume that $f(x+iy, d) < 0$ represents the half plane that excludes the critical point. Then, to find optimal controller parameters, an optimization problem like the following is used:

$$\max_{\rho} g$$

subject to:

$$\begin{aligned} & f(\rho^T \phi(j\omega_{ik}) G_i(j\omega_{ik}), d_1) < 0 \quad \text{for} \quad \omega_{ik} > \omega_x, \\ & f(\rho^T \phi(j\omega_{ik}) G_i(j\omega_{ik}), d_2) > 0 \quad \text{for} \quad \omega_{ik} > \omega_x, \\ & f(\rho^T \phi(j\omega_{ik}) G_i(j\omega_{ik}), d_2) < 0 \quad \text{for} \quad \omega_{ik} \leqslant \omega_x, \\ & \text{for} \quad k = 1, \ldots, N_i \ \text{and} \ i = 1, \ldots, m. \end{aligned} \tag{1.8}$$

where, the objective function for minimization, g, can be one of the two following cases:

- When one wants the open-loop of the system to be close to a desired open-loop, L_d. In this case, g would be the quadratic criterion below:

$$g = \sum_{i=1}^{m} \sum_{k=1}^{N_i} |L_i(j\omega_{ik}, \rho) - L_d(j\omega_{ik})|^2 \tag{1.9}$$

  where $L_i(j\omega_{ik}) = \rho^T \phi(j\omega_{ik}) G_i(j\omega_{ik})$.

- When the control objective is to optimize the load disturbance rejection of the closed-loop. This is, in general, achieved by maximizing the controller gain at low frequencies. For example, for a PID controller it corresponds to maximizing the coefficient of the integral part, i.e. $g = k_i$.

It should be noted that for all controller types including self-defined ones, one can specify an L_d to minimize the criterion (1.9). But, optimizing load disturbance rejection is considered only in PID, PI, PD, and Laguerre controllers.

In many control problems a constraint on the controller gain at high frequencies can help reducing the large pick values of the control input. This can be achieved by considering a bound on the real and the imaginary part of the controller, $\rho^T \phi(j\omega_i)$, at frequencies greater than ω_h:

$$\begin{aligned} & -K_u < Re(\rho^T \phi(j\omega_{ik})) < K_u \quad \text{for} \quad \omega_{ik} > \omega_h \\ & -K_u < Im(\rho^T \phi(j\omega_{ik})) < K_u \quad \text{for} \quad \omega_{ik} > \omega_h \end{aligned} \tag{1.10}$$

where Re and Im denote, respectively, real and imaginary parts of a complex value. These linear constraints will be included in the optimization problem (1.8) if specified by user.

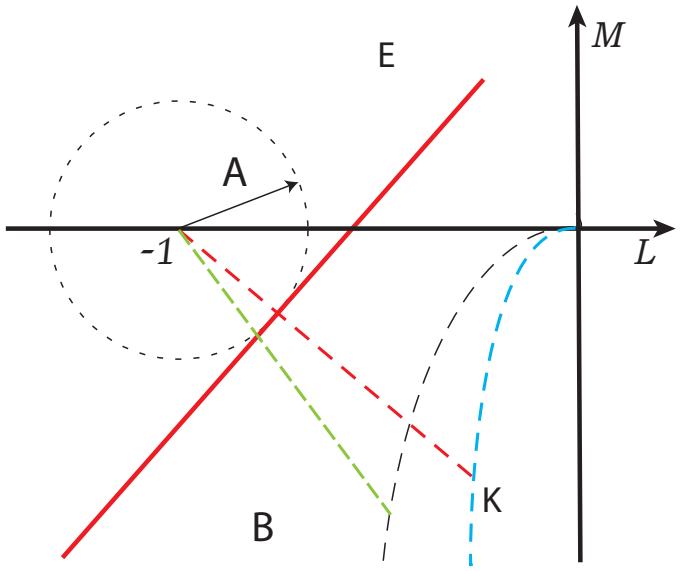


Figure 1.3: Loop shaping in Nyquist diagram by quadratic programming

1.3 Loop shaping controller

The performance specification can be defined by a desired open loop transfer function (or nonparametric frequency response data), $L_d(j\omega)$. It can be computed if a desired reference model M is available: $L_d = M/1 - M$. Typically for stable systems $L_d(s) = \omega_c/s$ would work well. Then a controller can be designed by minimizing the quadratic criterion of (1.9).

The modulus margin, the shortest distance between the Nyquist curve and the critical point, which is a better robustness indicator than the classical gain and phase margins, is considered in the loop shaping controller design method. For example, a modulus margin M_m of 0.5 is met if the Nyquist curve does not intersect a circle of radius 0.5 centered at the critical point. This can be achieved if the Nyquist diagram is at the side of d, a straight line tangent to the modulus margin circle, that excludes the critical point. This constraint is linear but conservative. The conservatism can be reduced if the slop of this line changes with frequency. A good choice is a line $d(M_m, L_d(j\omega_k))$ orthogonal to the line that connects the critical point and $L_d(j\omega_k)$ and tangent to the modulus margin circle (see Fig. 1.3). Thus the controller is designed solving the following quadratic optimization problem:

$$\min_{\rho} \sum_{i=1}^{m} \sum_{k=1}^{N_i} |L_i(j\omega_{ik}, \rho) - L_d(j\omega_{ik})|^2$$

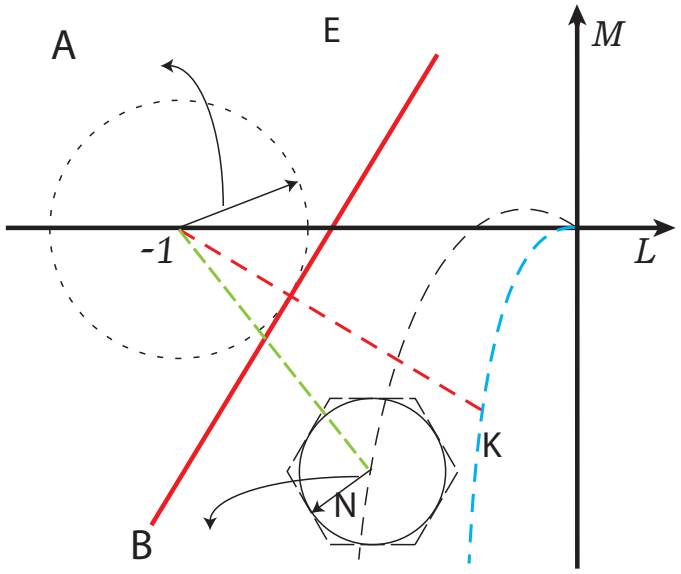


Figure 1.4: Expression of the robust performance condition as linear or convex constraints

subject to:

$$f(\rho^T \phi(j\omega_{ik}) G_i(j\omega_{ik}), d(M_m, L_d(j\omega_{ik}))) < 0 \qquad (1.11)$$
$$\text{for} \quad k = 1, \ldots, N_i \ \text{ and } \ i = 1, \ldots, m.$$

This approach can be applied to unstable systems if L_d contains the same number of unstable poles as well as the poles on the imaginary axis of $L_i(s)$ (see [3] for details). The toolbox will give a warning if L_d does not contain the same number of unstable poles or the poles on the imaginary axis of $L_i(s)$, though the calculation is susceptible to rounding errors and should be verified manually.

1.4 H_∞ controller

Consider a SISO plant model with multiplicative unstructured uncertainty:

$$\tilde{G}(j\omega) = G(j\omega)[1 + W_2(j\omega)\Delta]$$

where $G(j\omega)$ is the plant nominal frequency function, $W_2(j\omega)$ is the uncertainty weighting frequency function, and Δ is a stable transfer function with $\|\Delta\|_\infty < 1$. In the Nyquist diagram the open loop frequency function will belong to a disk centered at $L(j\omega, \rho)$ with a radius of $|W_2(j\omega)L(j\omega, \rho)|$. This

disk can be approximated by a circumscribed polygon with $n_q > 2$ vertices, such that $L_x(j\omega, \rho) = K(j\omega, \rho)G_x(j\omega)$ for $x = 1, \ldots, n_q$, where

$$G_x(j\omega) = G(j\omega)\left[1 + \frac{|W_2(j\omega)|}{\cos(\pi/n_q)} e^{j2\pi x/n_q}\right] \tag{1.12}$$

Suppose that the nominal performance is defined as $\|W_1 S\|_\infty < 1$, where $S = (1+KG)^{-1}$ is the sensitivity function and W_1 is the performance weighting filter. This condition is satisfied if the Nyquist curve of the nominal model does not intersect the performance disk, a disk centered at the critical point with a radius of $|W_1(j\omega)|$. Therefore, the robust performance is achieved if there is no intersection between the uncertainty and performance disks [1] (see Fig. 1.4). This constraint can be linearized using a straight line $d(W_1(j\omega), L_d(j\omega))$ which is tangent to the performance disk and orthogonal to the line connecting the critical point and $L_d(j\omega)$ [3]. The robust performance is met if $L_x(j\omega, \rho)$ is at the side of $d(W_1(j\omega), L_d(j\omega))$ that excludes the critical point for all ω. This can be represented by the following set of linear constraints:

$$\begin{aligned} &f(\rho^T \phi(j\omega_k)G_x(j\omega_k), d(W_1(j\omega_k), L_d(j\omega_k))) < 0 \\ &\text{for} \quad k = 1, \ldots, N \quad \text{and} \quad x = 1, \ldots, X. \end{aligned} \tag{1.13}$$

Then, in (1.13), if we let $W_2 = 0$, it will be equivalent to the nominal performance condition:

$$\|W_1 S\|_\infty < 1 \tag{1.14}$$

and if we let $W_1 = 0$, the robust stability condition:

$$\|W_2 T\|_\infty < 1 \tag{1.15}$$

will be obtained where T is the complementary sensitivity function. Other than these two constraints, constraints on the weighted infinity norm of other closed loop sensitivity functions, after being linearized in a similar manner, can also be included in the optimization problem (see more details in [3]):

$$\|W_3 KS\|_\infty < 1 \quad \text{and} \quad \|W_4 GS\|_\infty < 1 \tag{1.16}$$

where W_3 and W_4 are corresponding weighting filters.

As for a loop shaping controller, L_d should be defined such that it satisfies the Nyquist criterion and contains the poles of $L_i(s)$ on the imaginary axis. The toolbox will give a warning if L_d does not contain the same number of unstable poles or the poles on the imaginary axis of $L_i(s)$, though the calculation is susceptible to rounding errors and should be verified manually.

As a result, two different optimization problems are considered in H_∞ controller design method:

1. Defining a desired open-loop L_d and minimizing the criterion (1.9) in the following optimization problem:

$$\min_{\rho} \sum_{i=1}^{m} \sum_{k=1}^{N_i} |L_i(j\omega_{ik}, \rho) - L_d(j\omega_{ik})|^2$$

subject to:

$$\|W_1 S\|_\infty < 1\,, \quad \|W_2 T\|_\infty < 1\,, \quad \|W_3 KS\|_\infty < 1\,, \quad \|W_4 GS\|_\infty < 1$$

2. Solving the following optimization problem:

$$\begin{aligned} &\min \gamma \\ &\text{subject to:} \\ &\|\lambda_1|W_1 S| + \lambda_2|W_2 T| + \lambda_3|W_3 KS| + \lambda_4|W_4 GS|\|_\infty < \gamma \end{aligned} \tag{1.17}$$

where $\lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]$ is a vector of positive coefficients determining the importance of each constraint. If $\lambda_i = 0$ and W_i is defined, then the constraint corresponding to W_i will also be considered. For example if $\lambda = [1\,0\,1\,0]$ and W_1, W_2 and W_3 are defined the following optimization problem is solved:

$$\min \gamma$$

subject to:

$$\begin{aligned} \||W_1 S| + |W_3 KS|\|_\infty < \gamma \\ \|W_2 T\|_\infty < 1 \end{aligned}$$

If $\lambda = [0\,0\,0\,0]$ an upper bound on all weighted sensitivity function will be minimize:

$$\min \gamma$$

subject to:

$$\|W_1 S\|_\infty < \gamma\,,\ \|W_2 T\|_\infty < \gamma\,,\ \|W_3 KS\|_\infty < \gamma\,,\ \|W_4 GS\|_\infty < \gamma$$

As stated before, all these constraints are used in their linearized form and γ is minimized by a bisection algorithm to reach the best performance possible in terms of the above constraints.

1.5 MIMO controller

The performance specifications for SISO systems can also be used for designing MIMO controllers if the open loop system is decoupled. The main idea is to design a MIMO decoupling controller such that the open-loop transfer matrix $L(j\omega)$ becomes diagonally dominant. For this reason a diagonal desired open loop transfer matrix $\mathbf{L}_d$ is considered and the following quadratic criterion is minimized:

$$J(\rho) = \sum_{i=1}^{m} \sum_{k=1}^{N_i} \|\mathbf{L}_i(j\omega_{ik}, \rho) - \mathbf{L}_d(j\omega_{ik})\|_F \tag{1.18}$$

where F stands for the Frobenius norm.

MIMO controllers presented by a matrix of transfer functions are considered where each element K_{ij} of the matrix should be linearly parameterized, i.e., $K_{ij} = \rho_{ij}^T \phi_{ij}$. The controller parameters are obtained by minimizing $J(\rho)$ under some constraints to meet the SISO specifications for each diagonal element.

In MIMO systems, besides the performance constraints, there are other constraints implying the stability of MIMO systems that should be considered. In fact, because the closed-loop system will not be completely diagonal, the stability of dominant loops will not guarantee the stability of the MIMO system. However, a stability condition can be obtained based on Gershgorin bands (see [2]):

$$\begin{aligned} &r_q(\omega_k, \rho)\big|[1 + L_{dq}(j\omega_k)]\big| - Re\big([1 + L_{dq}(-j\omega_k)][1 + L_{qq}(j\omega_k, \rho)]\big) < 0 \\ &\text{for} \quad q = 1, \ldots, n_o \quad \text{and} \quad k = 1, \ldots, N \end{aligned} \tag{1.19}$$

where

$$r_q(\omega, \rho) = \sum_{p=1, p\neq q}^{n_o} |L_{pq}(j\omega, \rho)|,$$

n_o is the number of the outputs of the system, and L_{dq} is the q^{th} diagonal element of L_d. This constraint is written for one model. It will be considered for all models when tackling multi-model systems.

In summary, in MIMO systems, since the method is based on decoupling, always the criterion (1.18) is minimized. Hence, for every diagonal element of the open loop matrix ($\mathbf{L} = \mathbf{GK}$), an L_d should be specified. ($\mathbf{L}_d$ is a diagonal matrix with these L_d's as its diagonal elements.) Also all the previously explained performance conditions (GPhC, loop shaping, and H_∞) will be applied on the diagonal elements of the open loop transfer matrix. The stability conditions of (1.19) will be added to the other performance constraints.

1.6 Gain-scheduled controller

All presented robust controller design methods for systems with multimodel uncertainty can be extended to designing gain-scheduled controllers. Suppose that each model G_i is associated to a value of a scheduling parameter vector θ, which is measured in real time. The controller parameters can be polynomial functions of θ and be computed by the optimization algorithm. Then, in all the previous constraints, we would place:

$$\rho = M\bar{\theta}_l$$

where

$$M = \begin{bmatrix} (\rho_{1,n_p})^T & \cdots & (\rho_{1,1})^T & (\rho_{1,0})^T \\ \vdots & \ddots & \vdots & \vdots \\ (\rho_{n_\rho,n_p})^T & \cdots & (\rho_{n_\rho,1})^T & (\rho_{n_\rho,0})^T \end{bmatrix} \tag{1.20}$$

$$\text{and} \quad \bar{\theta}_l = [\theta_l^{n_p} \ \ldots \ \theta_l \ \overrightarrow{1}]^T.$$

where n_p is the order of polynomials describing controller parameters and n_ρ is the dimension of ρ which is the same as the dimension of the vector of the basis functions ϕ.

For instance, for a PID controller ($n_\rho = 3$) with a scalar scheduling parameter and the vector ρ as a second order polynomial of θ ($n_p = 2$) we will have a parametrization like this:

$$\rho(\theta) = \begin{bmatrix} k_{p_2} & k_{p_1} & k_{p_0} \\ k_{i_2} & k_{i_1} & k_{i_0} \\ k_{d_2} & k_{d_1} & k_{d_0} \end{bmatrix} \begin{bmatrix} \theta^2 \\ \theta \\ 1 \end{bmatrix} \tag{1.21}$$

For more details about the gain-scheduled controller design see [5].

Chapter 2

Toolbox commands

The procedure of design comprises three steps. First the type (or structure) of the controller should be determined. Then the desired performance characteristics are specified, and finally a controller with the desired type and performance is designed. In the following comes a description of these three steps with corresponding commands.

2.1 Determining controller structure

The first step of design is determining the desired controller type. By defining the controller type in fact the vector of basis transfer functions ϕ is specified. In the following command the controller type and subsequently the vector ϕ are specified by the user.

```
phi = conphi (ConType , ConPar , CorD , F, ConStruc, ConOpt)
```

`ConType` is a string representing the desired controller type. It is not case sensitive when being defined. it can be:

`'PID'`	For PID controller
`'PD'`	For PD controller
`'PI'`	For PI controller
`'Laguerre'`	For Laguerre basis function
`'Generalized'`	For Generalized basis function
`'UD'`	For user defined structure

`ConPar` is a scalar or a vector of parameters for the chosen controller type. For continuous-time PID or PD controller `ConPar` can be the time constant of the derivative part (if it is not specified a default value

equal to $1.2/\omega_{\max}$ will be computed, where $\omega_{\max}$ is the maximum value of the vector ω).

For discrete time PI, PD and PID controllers, `ConPar` specifies the sampling period.

For continuous-time Laguerre basis function, `ConPar` is $[\xi \quad n]$ where ξ defines the parameter of Laguerre basis and n is its order. For discrete-time Laguerre basis function, `ConPar` is $[T_s \quad a \quad n]$ where T_s is the sampling period, a the parameter of Laguerre basis and n its order.

For continuous-time generalized basis function, `ConPar` is ξ, a n-th dimensional vector containing the parameters of the generalized basis function. For discrete-time generalized basis function, `ConPar` is $[T_s \quad \xi]$.

For user defined structure, `ConPar` is a column vector of stable transfer functions.

`CorD` is either `'s'` or `'z'` showing, respectively, that the controller is in continuous- or discrete-time. If not mentioned, the continuous-time case will be considered.

`F` is a transfer function by which the vector ϕ is multiplied. For example one can multiply a factor of integral $\frac{1}{s}$ to PID basis functions to get: $\phi = \frac{1}{s} \times [1 \ \frac{1}{s} \ \frac{s}{1+\tau s}]^T$.

`ConStruc` is either `'LP'` for a linearly parameterized controller (default) or `'SP'` for a Smith Predictor controller.

`ConOpt` for a Smith Predictor controller gives $H(s) = G_n(s) - P_n(s)$.

2.1.1 Example 1

A continuous-time PID controller with $\tau = 0.1$:

```
phi = conphi ('PID' , 0.1)

phi.phi

Transfer function from input to output...
 #1:  1

      1
 #2:  -
      s
```

```
          s
 #3:  ----------
      0.1 s + 1
```

2.1.2 Example 2

A discrete-time PI controller with sampling time of 0.05 seconds:

```
phi = conphi ('PI' , 0.05 , 'z')

phi.phi

Transfer function from input to output...
 #1:  1

        z
 #2:  -----
      z - 1

Sampling time: 0.05
```

2.1.3 Example 3

A continuous-time PID controller with $\tau = 0.1$ multiplied by an integrator:

```
s=tf('s');  F=1/s;
phi = conphi ('PID' , 0.1 , 's' , F)

phi.phi

Transfer function from input to output...
      1
 #1:  -
      s

       1
 #2:  ---
      s^2

           s
 #3:  ------------
      0.1 s^2 + s
```

2.1.4 Example 4

A continuous-time 3rd-order Laguerre basis with $\xi = 1$:

```
phi = conphi ('Laguerre',[1,3])

phi.phi

Transfer function from input to output...
 #1:  1

      1.414
 #2:  -----
      s + 1

      1.414 s - 1.414
 #3:  ---------------
       s^2 + 2 s + 1

      1.414 s^2 - 2.828 s + 1.414
 #4:  ---------------------------
         s^3 + 3 s^2 + 3 s + 1
```

2.1.5 Example 5

A continuous-time, user defined vector of basis functions:

```
s = tf ('s');
phi = conphi('ud',[1 ; 1/s ; s/(s^2+2*s+1)]);

phi.phi

Transfer function from input to output...
 #1:  1

      1
 #2:  -
      s

             s
 #3:  ---------------
      s^2 + 2 s + 1
```

2.1.6 Example 6

A Smith Predictor, PID controller structure

```
s = tf('s');
G = 1/((5*s+1)*(10*s+1));
P{1} = G*exp(-5*s);

H = G - P{1};

phi = conphi('PID',0.01,'s',[],'SP',H);
```

After defining our desired controller structure, we shall proceed to the next step: specifying performance characteristics.

2.2 Determining control performance

The desired control performance attributes of the system are determined by the following command:

```
per = conper (PerType , par , Ld)
```

`PerType` is a string specifying the desired performance of the system. It can be `'GPhC'`, `'LS'` or `'Hinf'`.

- `'GPhC'` stands for the case when one wants to determine desired values for gain margin, phase margin, and crossover frequency.
- `'LS'` stands for Loop Shaping controller. In this method the distance between $L = GK$ (open-loop transfer function) and L_d (the desired one) in one thousand frequencies linearly spaced between $\omega_{\min}$ and $\omega_{\max}$ is minimized. A lower bound on the Modulus margin (the inverse of the infinity norm of the sensitivity function) is also guaranteed.
- `'Hinf'` In this method the distance between $L = GK$ and L_d is minimized under some H_∞ constraints on the weighted closed-loop sensitivity functions.

`par` is a structure that contains all the data specified by the user in this command.

For the `'GPhC'` method `par` is a vector containing the lower bounds of gain margin g_m, phase margin φ_m, crossover frequency ω_c, and an upper

bound for the controller gain K_u which may be applied at frequencies higher than ω_h. The crossover frequency ω_c, K_u and ω_h are optional values. If they are not assigned, no lower bound for the crossover frequency and no upper bound for the controller gain in high frequencies is considered. If `Ld` is specified, the quadratic criterion (1.9) will be minimized; otherwise, the controller gain at low frequencies will be maximized. It should, however, be noted that maximizing controller gain at low frequencies will be done in PID, PI, PD, and Laguerre controllers. In `'generalized'` type and in `'user defined'` basis functions, an `Ld` should be specified unless no optimization is performed and only a feasible solution is given.

For the `'LS'` method, `par` is a vector containing modulus margin M_m, and (if desired) K_u and ω_h. In loop shaping, the objective is to force open loop to act like a desired open loop function. So a desired open loop function (L_d) should always be specified by the user.

For the `'Hinf'` method, `par` is a cell, W, containing up to four weighting filters W_1, W_2, W_3 and W_4. The following constraints are applied:

$$\|W_1 S\|_\infty < 1\,, \quad \|W_2 T\|_\infty < 1\,, \quad \|W_3 KS\|_\infty < 1\,, \quad \|W_4 GS\|_\infty < 1$$

where $S = (1 + GK)^{-1}$ and $T = 1 - S$ are respectively sensitivity and complementary sensitivity functions. W_i can be any LTI type model or frequency domain model (e.g. 'frd' model). The distance between $L = GK$ and L_d is minimized.

`Ld` is a desired open loop which can be a parametric transfer function or a nonparametric `frd` object containing frequency response data over a frequency vector. It should contain the poles on the stability boundary of the plant model and the controller. It should also satisfies the Nyquist stability criterion.

2.2.1 Examples

`GPhC` : A lower bound of 2 for gain margin and 60 degree for phase margin is considered in the following command:

```
performance=conper('GPhC', [2 , 60]);
```

For PID controllers as well as Laguerre basis functions the low frequency gain of the controller will be maximized by the linear programming approach. A lower bound of 3 rad/s for the crossover frequency can be added by:

```
performance=conper('GPhC', [2 , 60, 3]);
```

The hard constraint on the crossover frequency can be replaced by defining `Ld=3/s` and minimizing `F(L-Ld)` by the quadratic programming approach as follows:

```
performance=conper('GPhC', [2 , 60], 3/s);
```

(`F` will be defined later on). If the controller gain at high frequencies is too large, say greater than 20 such that the control input becomes saturated, it can be limited to, say 10, for frequencies greater than 30 rad/s by the following command:

```
performance=conper('GPhC', [2 , 60, 0, 10, 30], 3/s);
```

Suppose that a robust controller should be designed that guarantees a gain margin of 2 and a phase margin of 45 for two models G_1 and G_2. Assume also that the lower bound of the crossover frequency for the first model is 1 rad/s and for the second model is 5 rad/s. The performances can be defined as follows:

```
MMper{1}=conper('GPhC', [2 , 45, 1]);
MMper{2}=conper('GPhC', [2 , 45, 5]);
```

`LS` : A modulus margin of 0.6 and a desired open-loop transfer function of `Ld=10/s` can be defined as control performance by the following command:

```
LSperformance=conper('LS', 0.6, 10/s);
```

An upper bound of 30 for frequencies greater than 100 rad /s can be considered by:

```
LSperformance=conper('LS', [0.6 , 30 , 100], 10/s);
```

`Hinf` : Consider performance weighting filters W_1, multiplicative uncertainty filter W_2 and input sensitivity weighting filter W_3 in an H_∞ controller design problem then the following commands can be used:

```
W{1}=W1;W{2}=W2;W{3}=W3;
```

```
Ld=W{1}-1;
HinfPer=conper('Hinf', W, Ld);
```

Note that if we consider that the desired sensitivity function S_d is close to the inverse of W_1, a good choice for L_d is $W_1 - 1$ (because $S_d^{-1} = 1 + L_d$).

2.3 Controller Design

After gathering the required data from user, the controller is designed by the following command:

```
K = condes (G , phi , per , options)
```

`G` is a cell, i. e., `G{1}`, `G{2}`, ..., `G{m}` represent SISO or MIMO models G_1, G_2, ..., G_m. In case there is just one model, define it as `G{1}` or simply `G`. Nonparametric models should be defined as an `frd` or `idfrd` object (type `doc frd` to see details about creating or converting your model to an `frd` object).

`phi` is the output of the first command. The command `condes` designs a controller of the specified type to meet the performance criteria implied by `per`. For MIMO systems `phi` is an $n_i \times n_o$ cell where n_i is the number of inputs and n_o is the number of outputs of the system. For example `phi{p,q}` is the vector of basis functions ϕ_{pq} for the element of row `p` and column `q` of the controller matrix K. So each element of the controller matrix may have a different vector of basis functions, or different structure. They should be defined separately by the first command, for example in a loop. Should you specify one controller type or structure for all the elements of the controller matrix, just simply enter one `phi` (not a cell) in the `condes` command.

Example : Consider a 2×2 MIMO controller where the diagonal elements are Laguerre basis functions of order 3 with an integral action and the off diagonal elements are PI controllers, Then the following commands should be used:

```
s = tf ('s');
phi{1,1} = conphi('Laguerre',[1 3],'s',1/s);
phi{1,2} = conphi('PI');
phi{2,1} = conphi('PI');
phi{2,2} = conphi('Laguerre',[1 3],'s',1/s);
```

`per` is a cell. `per{1}`, `per{2}`, ..., `per{m}` contain the desired performance characteristics of, respectively, `G{1}`, `G{2}`, ..., `G{m}`. When you want to apply one performance criterion for all models, simply just enter `per` (you don't need to define a cell). For MIMO systems, `per` is a cell. `per{i}{q}` ($i = 1, \ldots, m$ and $q = 1, \ldots, n_o$) contains the performance characteristics of the q^{th} diagonal element of the open loop `L{i}` ($L_i = G_i \times K$), which should be defined by the command `conper`. In MIMO systems the objective function (1.18) will be minimized (the aim is to design a controller to decouple the system as much as possible). Hence `Ld` must be defined in `per` for every diagonal element of every open loop system. If your performance characteristics differ for the different diagonal elements of the open loop matrix, yet are the same for every model, you can simply define `per{1}`, `per{2}`, ..., `per{`n_o`}` where `per{q}` contains the performance characteristics of the q^{th} diagonal element of the open loop `L{i}` for $i = 1, \ldots, m$. If your desired performance characteristics are the same for all the diagonal elements in all of the models, you can simply enter one `per` (not a cell) in the `condes` command to be applied to all of them.

Example : Suppose that a robust 2×2 MIMO controller should be designed for three MIMO models G_1, G_2 and G_3. Assume that the desired open-loop transfer function for the first output is $2/s$ and for the second output $10/s$ with a modulus margin of 0.5 for all models then the performance is defined as follows:

```
MIMOper{1}=conper('LS', 0.5, 2/s);
MIMOper{2}=conper('LS', 0.5, 10/s);
```

Now suppose that the lower bound on the modulus margin for the first model is 0.3, for the second model is 0.4 and for the third one is 0.5. In this case the performance is defined by:

```
MIMOper{1}{1}=conper('LS', 0.3, 2/s);
MIMOper{1}{2}=conper('LS', 0.3, 10/s);
MIMOper{2}{1}=conper('LS', 0.4, 2/s);
MIMOper{2}{2}=conper('LS', 0.4, 10/s);
MIMOper{3}{1}=conper('LS', 0.5, 2/s);
MIMOper{3}{2}=conper('LS', 0.5, 10/s);
```

`options` is a structure whose fields can be set by a specific command that will be explained in the next subsection.

When determining an `Ld` in `per`, (which is compulsory in `'LS'` and `'Hinf'` cases but optional in `'GPhC'` case) the objective of design will be minimizing the difference between the system open loop and the desired open loop `Ld` which is a quadratic objective function in terms of controller parameters ρ. The constraints in all of the three cases are linear in terms of ρ. Hence we have a quadratic programming problem which can be solved by the solver `quadprog`.

In MIMO systems the Gershgorin stability conditions are applied on the controller design. These constraints are convex in terms of ρ, but can be linearized without conservatism for systems with two outputs. For systems with more than two outputs a convex optimization solver is required. The users should instal Yalmip package and a convex optimization solver (e.g. `sdpt3`). The use of Yalmip and convex optimization solvers increase the execution time but can be used if the users do not have the optimization toolbox of Matlab. However, regarding the fact that the Gershgorin bands constraints are sometimes conservative, one can skip applying them (see options section below). In this case, the standard solver `quadprog` will be used for solving the resulting quadratic optimization problem.

2.3.1 Examples

Controller design with `GPhC` performance for a single model system

Design a PID controller for the following first order model with delay:

$$G = \frac{e^{-s}}{(s+1)^3}$$

The objective is to ensure: Gain margin = 2, Phase margin = 60°, Crossover frequency = 0.08 rad/s.

```
s=tf('s');
G=exp(-s)/(s+1)^3;

phi=conphi('PID');
per=conper('GPhC',[2,60,.08]);

K=condes(G,phi,per)

174.0174 (s^2 - 0.09479s + 0.04465)
-----------------------------------
            s (s+83.33)
```

Controller design with GPhC performance for a multi-model system

Consider a system that has three different models in three operating points as follows:

$$G_1 = \frac{4e^{-3s}}{(10s+1)} \qquad G_2 = \frac{e^{-5s}}{(s^2+14s+7.5)} \qquad G_3 = \frac{2e^{-s}}{(20s+1)}$$

The objective is to design a robust PID controller to ensure a gain margin of 3, phase margin of 60 for all models with different desired crossover frequencies as follows: ω_{c_1}=0.2 rad/s (for model 1), ω_{c_2}=0.01 rad/s (for model 2), ω_{c_3}=0.07 rad/s (for model 3).

```
s=tf('s');
G{1}=exp(-3*s)*4/(10*s+1);
G{2}=exp(-5*s)/(s^2+14*s+7.5);
G{3}=exp(-s)*2/(20*s+1);

phi=conphi('PID',.05);

per{1}=conper('GPhC',[3,60,.2]);
per{2}=conper('GPhC',[3,60,.01]);
per{3}=conper('GPhC',[3,60,.07]);

K=condes(G,phi,per)

13.9441 (s+0.1735) (s+0.5121)
-----------------------------
          s (s+20)
```

Controller design with LS performance for a multivariable system

Consider a MIMO model given by:

$$G(s) = \begin{bmatrix} \frac{5e^{-3s}}{4s+1} & \frac{2.5e^{-5s}}{15s+1} \\ \frac{-4e^{-6s}}{20s+1} & \frac{e^{-4s}}{5s+1} \end{bmatrix}$$

The objective is to design a MIMO PI controller to achieve a modulus margin of 0.5 for the diagonal open-loop systems and a desired open-loop transfer function:

$$L_d(s) = \begin{bmatrix} \frac{1}{30s} & 0 \\ 0 & \frac{1}{30s} \end{bmatrix}$$

```
s=tf('s');
G=[5*exp(-3*s)/(4*s+1) 2.5*exp(-5*s)/(15*s+1); ...
   -4*exp(-6*s)/(20*s+1) exp(-4*s)/(5*s+1)];

phi=conphi('PI');

per=conper('LS',0.5,1/(30*s));

K=condes(G,phi,per)

Zero/pole/gain from input 1 to output...
      0.029928 (s+0.07842)
 #1:  ---------------------
                  s

      0.041268 (s+0.1954)
 #2:  ---------------------
                  s

Zero/pole/gain from input 2 to output...
      -0.007108 (s+0.7555)
 #1:  ---------------------
                  s

      0.15087 (s+0.07343)
 #2:  ---------------------
                  s
```

Controller design with `Hinf` performance

Consider the following system:

$$G(z) = \frac{z - 0.2}{z^3 - 1.2z^2 + 0.5z - 0.1}$$

with sampling period $Ts = 1s$. Compute a third-order discrete-time controller with integral action that satisfies $\|W_1 S\|_\infty < 1$ and has a bandwidth of 0.2 rad/s, where

$$W_1(z) = \frac{0.4902(z^2 - 1.0432z + 0.3263)}{z^2 - 1.282z + 0.282}$$

We proceed as follows:

```
Ts=1;
z=tf('z',Ts);
s=tf('s');
```

```
G=(z-0.2)/(z^3-1.2*z^2+0.5*z-0.1);
W{1}=0.4902*(z^2-1.0432*z+0.3263)/(z^2-1.282*z+0.282);

Ld=0.2/s;

phi=conphi('Laguerre',[Ts 0 3],'z',z/(z-1));

per=conper('Hinf',W,Ld);

K=condes(G,phi,per)

0.15968 (z+0.0614) (z^2 - 0.9455z + 0.2318)
-------------------------------------------
                z^2 (z-1)
```

2.4 Controller design options

The controller design options are defined by the following command:

```
options = condesopt ('param1',value1,'param2',value2,...)
```

`w` : In nonparametric models where the class of models are `frd`, every model has its own vector of frequency points and we do not need to assign frequency vectors in options. If you want to specify a frequency grid for each parametric model, define `w` as a cell such that `w{i}` is the vector of frequency points in which the frequency response $G_i(j\omega)$ is obtained. In other words, after you type:

```
options = condesopt ('w' , w)
```

`options.w` will be a cell with `options.w{1}`, `options.w{2}`, ... Should you use just one frequency vector for all models, simply define `w` as a vector (not a cell). If you do not specify `w`, a default frequency grid for each model will be created by the `bode` command.

`F` : This is a weighting filter for L-Ld. For almost all optimization problems an approximation of the two norm of F(L-Ld) is minimized. Its value is 1/(1+Ld) by default. F should be set to 1 if no filter is desired.

`nq` : is an integer greater than 2 representing the number of vertices of a polygon of least area that circumscribes the frequency domain model

uncertainty circle in the Nyquist plot. If it is empty, the circle will not be approximated by a polygon and therefore a convex constraint is defined and an SDP solver with YALMIP is used instead of `'linprog'` or `'quadprog'`.

`gamma` : This option is used only for `Hinf` performance, when γ is minimized in the optimization problem (1.17). Minimizing the infinity norm is performed by a bisection algorithm. For multimodel case the maximum of $\gamma(i)$ is minimized for all models. `'gamma'` is a vector containing `[`$g_{\min}$`,` $g_{\max}$`, tol]` where $g_{\min}$ and $g_{\max}$ are the minimum and maximum value of gamma and `tol` is a small positive number that indicates the tolerance of optimal γ.

`lambda` : This option is used together with `gamma` only for `Hinf` performance. It indicates the sum of which weighted sensitivity functions should be minimized.

`Gbands` : is a string that takes two values : `'on'` or `'off'`. Its default value is `'on'` meaning that the Gershgorin stability conditions are considered in the design of MIMO controller. If this option is turned to `'off'`, the optimization problem becomes linear (for systems of more than two outputs) which had less numerical problem and is much faster than the convex solvers. However, the closed-loop stability should be verified after optimization. This option can be changed to `'off'` by typing:

```
options = condesopt ('Gbands' , 'off')
```

`np` : This option is used only for gain scheduled controller design. `np` is a vector that indicates the degree of polynomials describing the gain-scheduled controller parameters. For example `np=[2 1]` indicates that we have two scheduling parameters. The first one is described by a second-order polynomial and the second one is linear. Its default value is `[]` which implies that the controller is not gain-scheduled.

`gs` : This option is used if a gain-scheduled controller should be designed. `gs` is a m by n matrix, where n is the number of scheduling parameters and m the number of operating points. The i-th row of `gs` contains the values of n scheduling parameters that corresponds to the i-th model G_i. The default value of `gs` is `[]`.

`yalmip` : is a string that can be set to `'on'` to activate the YALMIP interface. It should be activated when Gbands='on' for multivariable

controller design of more than 2 inputs. It will be automatically activated if YALMIP has already been installed and `nq` is empty. As the default solver SDPT3 is used but can be changed by 'solver' option, e.g. options=condesopt('yalmip','on','solver','sedumi').

`Solver options` : Besides, the options mentioned above, one can alter every option available for different solvers. In this case, one should be aware of which solver would be used in which optimization problem. For example, if one wants to design a PID controller for a SISO system without specifying a desired open loop (where optimization problem (1.8) is solved by `linprog`), one can change every option available for the solver `linprog` in this command. For example:

```
options = condesopt ('w', w1, 'MaxIter', 100, 'Display', ...
    'iter')
```

changes the values of `MaxIter` and `Display`, which are some of `linprog`'s options, to the specified values as well as assigning `w1` to `w`.

2.4.1 Examples

H_∞ controller design for an unstable system

Consider the family of plants described by the following multiplicative uncertainty model:

$$\tilde{G}(s) = \frac{(s+1)(s+10)}{(s+2)(s+4)(s-1)}[1 + W_2(s)\Delta(s)]$$

where

$$W_2(s) = 0.8\frac{1.1337s^2 + 6.8857s + 9}{(s+1)(s+10)}$$

The nominal performance is defined by $\|W_1\mathcal{S}\|_\infty < 1$ with :

$$W_1(s) = \frac{2}{(20s+1)^2}$$

Design a PID controller to optimize the robust performance

$$\||W_1S| + |W_2T|\|_\infty < \gamma$$

and $\|KS\|_\infty < 20$. We choose the following $L_d(s)$ that contains the unstable pole of the plant model and meets the Nyquist stability criterion.

$$L_d(s) = 2\frac{s+1}{s(s-1)}$$

and $W_3 = 0.05$. Then we minimize γ under $\|W_3 K S\|_\infty < 1$.

```
s=tf('s');
G=(s+1)*(s+10)/((s+2)*(s+4)*(s-1));
Ld=2*(s+1)/s/(s-1);

W{1}=2/(20*s+1)^2;
W{2}=0.8*(1.1337*s^2+6.8857*s+9)/((s+1)*(s+10));
W{3}=tf(0.05);

phi=conphi('PID',0.01);
hinfper=conper('Hinf',W,Ld);
w=logspace(-1,4,500);
opt=condesopt('gamma',[0.01 2 0.001],'lambda',[1 1 0 0],'w',w);

K=condes(G,phi,hinfper,opt)

Optimization terminated.
gamma=0.78151

Zero/pole/gain:
16.9564 (s+2.037) (s+26.19)
-----------------------------
        s (s+100)
```

The same problem can be solved without approximation of the multiplicative uncertainty circle by an octagon by setting the following options:

```
opt.nq=[];
opt.yalmip='on';
K=condes(G,phi,hinfper,opt)

No problems detected
gamma=0.72224

Zero/pole/gain:
19.0488 (s+2.227) (s+21.47)
-----------------------------
        s (s+100)
```

The obtained results are better and have less conservatism but the computation time is much larger.

Smith Predictor controller design with H_∞ performance

Consider the family of plants:

$$P_i(s) = \frac{e^{-\tau_i s}}{(5s+1)(10s+1)}$$

where τ_i belongs to the set $\{4.5, 5, 5.5\}$.

Gain-scheduled controller design for a domestic condensing boiler

$G_1(s)$ to $G_6(s)$ are six first order identified models concerning a domestic condensing boiler in different water flow rates $\theta = [8; 7; 6; 5; 4; 3]$ lit./min. The objective is to compute a gain-scheduled PI controller with a gain margin of 2, phase margin of 60°.

```
s=tf('s');
G{1}= exp(-6.2*s) * 0.00932/(31.84*s + 1);
G{2}= exp(-6.02*s) * 0.01032/(34.08*s + 1);
G{3}= exp(-6.69*s) * 0.01169/(34.76*s + 1);
G{4}= exp(-9.76*s) * 0.01391/(37.62*s + 1);
G{5}= exp(-12*s) * 0.01700/(57.42*s + 1);
G{6}= exp(-15.2*s) * 0.0216/(62.17*s + 1);

phi=conphi('PI');
per=conper('GPhC',[2,60]);

theta=[8;7;6;5;4;3];

opt=condesopt('np',2,'gs',theta);


K=condes(G,phi,per,opt)

K{1}+theta K{2}+theta^2 K{3}

Optimization terminated.
K{1}=

Zero/pole/gain:
149.8348 (s-0.01179)
---------------------
         s

K{2}=

Zero/pole/gain:
-24.775 (s-0.02971)
-------------------
         s
```

```
K{3}=

Zero/pole/gain:
5.3054 (s+0.01414)
-------------------
        s
```

Simultaneous stabilization (example from Robust Control Toolbox of Matlab)

A set of seven unstable models are given:

```
p{1} = tf(2,[1 -2]);
p{2} = p{1}*tf(1,[.06 1]);              % extra lag
p{3} = p{1}*tf([-.02 1],[.02 1]);       % time delay
p{4} = p{1}*tf(50^2,[1 2*.1*50 50^2]);  % HF resonance
p{5} = p{1}*tf(70^2,[1 2*.2*70 70^2]);  % HF resonance
p{6} = tf(2.4,[1 -2.2]);                % pole/gain migration
p{7} = tf(1.6,[1 -1.8]);
```

A bandwidth of 4.5 rad/s is desired leading to the following weighting filter :

```
desBW = 4.5; W{1}= makeweight(500,desBW,0.33);
```

A noise filter is also defined as :

```
NF = (10*desBW)/20;  % numerator corner frequency
DF = (10*desBW)*50;  % denominator corner frequency
Wnoise = tf([1/NF^2  2*0.707/NF  1],[1/DF^2  2*0.707/DF  1]);
W{2} = Wnoise/abs(freqresp(Wnoise,10*desBW));
```

An (n+1)-th order controller with integral action is designed using a generalized basis function:

```
phi=conphi('Generalized',logspace(-2,2,n),'s',1/s);
```

The poles of the controller are logarithmically spaced between 0.01 and 100. In the first step one initial controller is designed for $P_1(s)$:

```
s=zpk('s');

Ld0=10*(s+2)/s/(s-2);

n=5;
```

```
phi=conphi('Generalized',logspace(-2,2,n),'s',1/s);

per0=conper('Hinf',W,Ld0);

w=logspace(-2,3,1000);
opt=condesopt('w',w);

K0=condes(p{1},phi,per0,opt);
```

and then is used to compute $L_d(s)$ for all models:

```
for j=1:7,
    Ld1{j}=K0*p{j};
    per{j}=conper('Hinf',W,Ld1{j});
end
```

Then an H_∞ controller can be designed using the following commands:

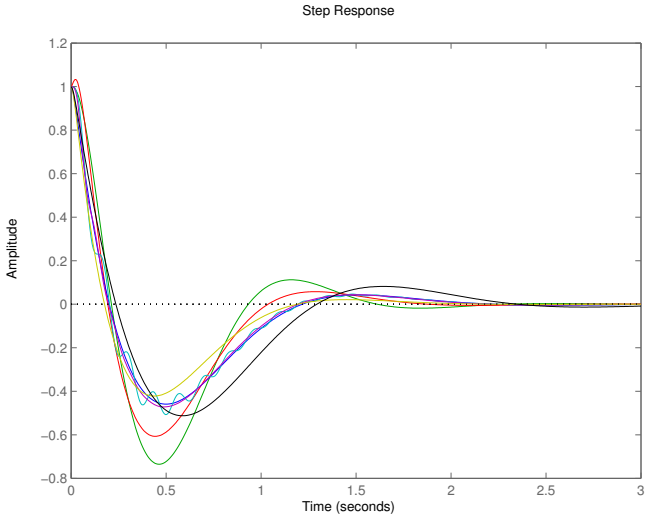
```
opt.gamma=[0.1,5,0.01];
K=condes(p,phi,per,opt);
```

$$K = \frac{385.9471(s+12.46)(s+1.679)(s+0.8785)(s+0.14)(s+0.009792)}{s(s+0.01)(s+0.1)(s+1)(s+10)(s+100)}$$

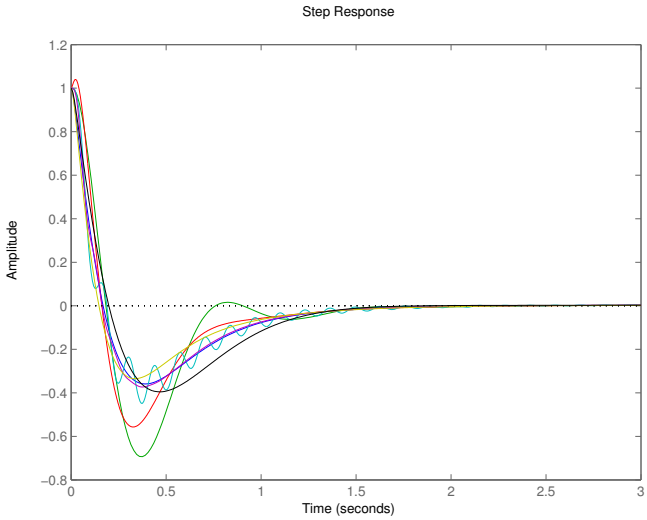
It is interesting to see that the same code can be used for computing a second order controller ($n = 1$) that gives also satisfactory results (robust control toolbox of Matlab ends up with a sixth order controller). For $n = 1$ we obtain:

$$K = \frac{325.3633(s+2.154)}{s(s+100)}$$

The disturbance response of the closed-loop system is given for the second-order controller and 6-th order controller for the sake of comparison with the results of robust control toolbox of Matlab.



Disturbance response of the 2nd order controller



Disturbance response of the 6-th order controller

Bibliography

[1] C. J. Doyle, B. A. Francis, and A. R. Tannenbaum. *Feedback Control Theory*. Mc Millan, New York, 1992.

[2] G. Galdos, A. Karimi, and R. Longchamp. H_∞ controller design for spectral MIMO models by convex optimization. *Journal of Process Control*, 20(10):1175 – 1182, 2010.

[3] A. Karimi and G. Galdos. Fixed-order H_∞ controller design for nonparametric models by convex optimization. *Automatica*, 46(8):1388–1394, 2010.

[4] A. Karimi, M. Kunze, and R. Longchamp. Robust controller design by linear programming with application to a double-axis positioning system. *Control Engineering Practice*, 15(2):197–208, February 2007.

[5] M. Kunze, A. Karimi, and R. Longchamp. Gain-scheduled controller design by linear programming. In *European Control Conference*, pages 5432–5438, July 2007.

[6] M. Sadeghpour, V. de Oliveira, and A. Karimi. A toolbox for robust PID controller tuning using convex optimization. In *IFAC Conference in Advances in PID Control*, Brescia, Italy, 2012.