A Matlab toolbox for robust feedforward design and robustness analysis in the presence of LTI/LTV uncertainties

Clément Roos and Gilles Ferreres
ONERA–CERT/DCSD
System Control and Flight Dynamics Department
BP 4025, F–31055 Toulouse Cedex, France
croos@onera.fr, ferreres@onera.fr

July 2005

This paper presents the *Robust Feedforward Design Toolbox (RFD Toolbox)*, a Matlab toolbox for robust feedforward design and robustness analysis in the presence of LTI/LTV uncertainties (available at *http://www.cert.fr/dcsd/cdin/roos/*). It is organised as follows: section 1 states the problem, while section 2 gives an overview of the proposed algorithms; sections 3, 4 and 5 then set out basic and advanced features of the toolbox. A detailed presentation of the method is available in [2, 3].

1 Problem statement

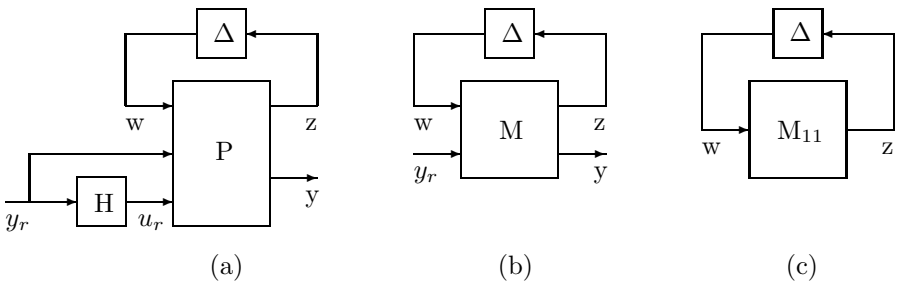

Figure 1: The design scheme (a), the standard interconnection structure (b) and the special case of robust stability (c).

The relation between these 3 structures is given below. If $P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \end{bmatrix}$, then:

$$M = \begin{bmatrix} P_{11} & P_{12} + P_{13}H \\ P_{21} & P_{22} + P_{23}H \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad (1)$$

1.1 Robust feedforward design

With reference to figure 1.a, the issue is to design a feedforward controller $H(s)$ which minimises the L_2 induced norm of the transfer function $T_{y_r \to y}$ between the reference input y_r and y despite model uncertainties in $\Delta = diag(\Delta_1, \Delta_2)$, where $\Delta_1 = diag(\delta_i^{TI} I_{r_i})$ contains LTI parametric uncertainties δ_i^{TI} and $\Delta_2 = diag(\delta_i^{TV} I_{q_i})$ contains arbitrarily time varying parametric uncertainties δ_i^{TV}. Neglected dynamics could also be accounted for, and each normalised time invariant or time varying parametric uncertainty satisfies $\delta_i \in [-1,\ 1]$. As a consequence let the unit ball $B\Delta = \{\Delta \mid \overline{\sigma}(\Delta) < 1\}$. The issue is to minimise (an upper bound of) γ under the induced L_2 norm constraint:

$$\|T_{y_r \to y}\|_{iL_2} \leq \gamma \ \ \forall \Delta \in B\Delta \quad (2)$$

1.2 Robust performance analysis

With reference to figure 1.b, the issue is to minimise (an upper bound of) γ under the L_2 induced norm constraint (2) for a fixed feedforward controller.

1.3 Robust stability analysis

With reference to figure 1.c, the issue is either:

- to compute a guaranteed robustness margin for the interconnection of $M_{11}(s)$ with Δ, i.e. to minimise (an upper bound of) β so that $M_{11}(s) - \Delta$ remains stable $\forall \Delta \in \frac{1}{\beta} B\Delta$.
- or to test whether the interconnection of $M_{11}(s)$ with Δ remains stable $\forall \Delta \in \frac{1}{\beta_{max}} B\Delta$, where β_{max} is fixed.

2 Overview of the algorithms

2.1 Robust feedforward design

The following proposition is just an application of the robustness analysis theory:

Proposition 2.1 *Assume that $M(s)$ is asymptotically stable. Let $D_1(\omega) = D_1^*(\omega) > 0$ and $G_1(\omega) = G_1^*(\omega)$ some frequency-dependent scaling matrices whose structure fits the one of Δ_1. Let then $D_2 = D_2^* > 0$ and $G_2 = G_2^*$ some constant scaling matrices whose structure fits the one of Δ_2, assuming moreover that D_2 is a real matrix and G_2 an imaginary matrix. Let $\mathcal{D}(\omega) = diag(D_1(\omega), D_2, I)$ and $\mathcal{G}(\omega) = diag(G_1(\omega), G_2, 0)$. As a sufficient condition (2) is satisfied if:*

$$M_\gamma(j\omega)\mathcal{D}(\omega)M_\gamma^*(j\omega) + j\left(\mathcal{G}(\omega)M_\gamma^*(j\omega) - M_\gamma(j\omega)\mathcal{G}(\omega)\right) < \mathcal{D}(\omega) \;\; \forall\omega \in [0,\ +\infty) \quad (3)$$

with $M_\gamma = M \begin{bmatrix} I & 0 \\ 0 & \frac{I}{\gamma} \end{bmatrix}$.

The next proposition then reformulates inequality (3) as a convex infinite dimensional optimization problem:

Proposition 2.2 *With reference to proposition 2.1 let $D(\omega) = diag(D_1(\omega), D_2)$ and $G(\omega) = diag(G_1(\omega), G_2)$. Let then $H(s) = \sum_{i=1}^N \theta_i H_i(s)$, where filters $H_i(s)$ are fixed while the θ_i are the design parameters. As a sufficient condition (2) is satisfied if there exist θ_i and frequency dependent scaling matrices $D(\omega)$ and $G(\omega)$ satisfying $\forall\omega \in [0,\ +\infty)$ (the ω dependence is dropped out to alleviate the notations):*

$$\begin{bmatrix} D - P_{11}DP_{11}^* + j\left(P_{11}G - GP_{11}^*\right) & -(P_{11}D + jG)P_{21}^* & P_{12} + P_{13}\sum_{i=1}^N \theta_i H_i \\ \star & \gamma I - P_{21}DP_{21}^* & P_{22} + P_{23}\sum_{i=1}^N \theta_i H_i \\ \star & \star & \gamma I \end{bmatrix} > 0 \quad (4)$$

where $\star$ denotes the conjugate part of the hermitian matrix.

Remark: Consider the initial optimization problem of section 1.1 and let Γ the associated minimal value of γ. Let γ^* the minimal value of γ satisfying the sufficient condition of proposition 2.2. It is worth emphasizing that γ^* is just an upper bound of Γ. Nevertheless the conservatism is usually (very) reasonable in practice.

The idea is to solve a finite dimensional optimization problem corresponding to a frequency gridding, as usually done in μ analysis.

The following optimal algorithm is introduced in this context:

1. Let $(\omega_i)_{i\in[1,\ N]}$ an initial (small size) frequency gridding.
2. Solve the optimization problem of proposition 2.2 on the frequency gridding. Let $\gamma_{LB,N}$ the minimised value, and $H(s)$, D_2 and G_2 the associated values of the feedforward controller and of the constant scaling matrices.

3. For these values let $\gamma = (1+\epsilon)\gamma_{LB,N}$ with $\epsilon > 0$, and check (3) with a μ frequency sweeping technique [1]. If this is satisfied γ is an upper bound of γ^* and the global minimum is computed with a satisfactory accuracy ϵ. Otherwise let $\tilde{\omega}$ a worst-case value of the frequency, where (3) is not satisfied. Include $\tilde{\omega}$ in the gridding and go back to step 2.

A parameter $\alpha > 0$ is also introduced to release the constraints between optimization at step 2 and validation at step 3, which are performed for $\Delta \in (1+\alpha)B\Delta$ and $\Delta \in B\Delta$ respectively. It allows to guarantee convergence and to reduce the number of iterations.

The LMI optimization performed at step 2 of this optimal algorithm can be computationally very demanding. A suboptimal algorithm is introduced in this context, allowing to optimize first w.r.t. the frequency dependent scalings $D_1(\omega)$ and $G_1(\omega)$ with the Matlab routine *mu.m* of the μ-Analysis and Synthesis Toolbox, and then w.r.t. the design parameters θ_i and constant scalings D_2, G_2 by solving inequality (4) with LMI tools.

1. Let $(\omega_i)_{i\in[1,\,N]}$ an initial (small size) frequency gridding and $H(s)$, D_2, G_2 the initial values of the feedforward controller and constant scaling matrices (see *Remark* (i) below for comments about how to initialize these parameters).

2a. For the current values $H(s)$, D_2, G_2 of the feedforward controller and constant scaling matrices, perform an optimization on the gridding w.r.t. $D_1(\omega_i)$ and $G_1(\omega_i)$ to minimize γ in inequality (3). This is done by first transforming the LMI formulation (3) into a $\overline{\sigma}$ one [2, 3] and then performing a dichotomy search on γ.

2b. For the values $D_1(\omega_i)$ and $G_1(\omega_i)$ determined at step 2a, solve the optimization problem of proposition 2.2 w.r.t. θ_i, D_2 and G_2. Let $\gamma_{N_{min}}$ the minimized value of γ and $H(s)$, D_2, G_2 the corresponding values of the feedforward controller and constant scaling matrices. If $\gamma_{N_{min}} < (1-\eta)\gamma_N$, with γ_N the value of γ at the beginning of step 2a and $\eta > 0$ a given threshold, it means that it is worth going on with the optimization and γ minimization process. In this case, let $\gamma_N = \gamma_{N_{min}}$ and go back to step 2a. Otherwise, let $\gamma_N = \gamma_{N_{min}}$ and continue to step 3.

3. Let $\gamma = (1+\epsilon)\gamma_N$ with $\epsilon > 0$. Considering the values of $H(s)$, D_2, G_2 determined at step 2b, check (3) on the whole frequency range with a μ frequency sweeping technique. If this is satisfied, stop since γ is an upper bound of γ^*. Otherwise let $\tilde{\omega}$ a worst-case value of the frequency, where (3) is not satisfied. Include $\tilde{\omega}$ in the gridding and go back to step 2a.

Remarks:
(*i*) The initial values of $H(s)$, D_2 and G_2 can be arbitrarily chosen, e.g. all design parameters equal to zero, $D_2 = I$ and $G_2 = 0$. An other solution is to slightly modify the algorithm by performing a first optimal iteration using the optimal algorithm and then switching to the suboptimal algorithm. The initialization of the parameters and scalings can thus be avoided without penalizing the global computational cost, since the choice of a sufficiently small initial frequency gridding guarantees that the number of variables remains reasonable when solving the LMI optimization problem of proposition 2.2.
(*ii*) The choice of η conditions the relevance of γ_N at the end of step 2b. A smaller value of η indeed increases the number of iterations between steps 2a and 2b and thus leads to a less conservative value of γ_N. This parameter as well as the tuning parameter of the routine *mu.m* (see the help of *mu.m* in the μ-Analysis and Synthesis Toolbox) enable to achieve a trade-off between the computational time and the accuracy of the final upper bound of γ^*.

Note that unlike the optimal algorithm, it is not guaranteed that the value of γ_N determined at the end of iteration 2b is a lower bound of γ^*. Thus it is not possible to quantify the accuracy of the final upper bound of γ^*. Nevertheless, the following heuristics can be applied to combine both optimal and suboptimal algorithms to determine guaranteed lower and upper bounds of γ^* with sufficient accurary while keeping a reasonable computational time:

1. Perform the suboptimal algorithm to determine a guaranteed upper bound γ_{UB} of γ^*.
2. Perform the optimal algorithm to determine a guaranteed lower bound γ_{LB} of γ^* and stop as soon as $\gamma_{LB} > (1 - \xi)\gamma_{UB}$, with $\xi > 0$ the desired accuracy. The algorithm can be initialized with the frequency gridding and the values of $H(s)$, D_2 and G_2 obtained at the end of step 1.

Remark: The frequency gridding can be tightened at the end of the suboptimal algorithm to reduce the computational time of the optimal algorithm. A parameter $0 \leq \nu \leq 1$ is introduced in this context [2, 3].

2.2 Robust performance analysis

Robust performance analysis can directly be performed using the algorithms detailed in section 2.1 by removing the design parameters θ_i from the optimization process, i.e. by setting $M_{12} = P_{12}$ and $M_{22} = P_{22}$ and solving (3) instead of (4) at step 2, which proves more efficient because of the lower complexity of the resulting LMI.

2.3 Robust stability analysis

Robustness margin computation

Proposition 2.1 is first adapted:

Proposition 2.3 *Assume that $N(s)$ is asymptotically stable. Let $D_1(\omega) = D_1^*(\omega) > 0$ and $G_1(\omega) = G_1^*(\omega)$ some frequency-dependent scaling matrices whose structure fits the one of Δ_1. Let then $D_2 = D_2^* > 0$ and $G_2 = G_2^*$ some constant scaling matrices whose structure fits the one of Δ_2. Assume that D_2 is a real matrix and G_2 an imaginary matrix. Let $\mathcal{D}(\omega) = diag(D_1(\omega), D_2)$ and $\mathcal{G}(\omega) = diag(G_1(\omega), G_2)$. As a sufficient condition the interconnection $N(s) - \Delta$ remains stable $\forall \Delta \in \frac{1}{\beta} B\Delta$ if:*

$$N(j\omega)\mathcal{D}(\omega)N^*(j\omega) + j\left(\mathcal{G}(\omega)N^*(j\omega) - N(j\omega)\mathcal{G}(\omega)\right) < \beta^2\mathcal{D}(\omega) \;\; \forall \omega \in [0,\; +\infty) \quad (5)$$

Let β^* the minimal value of β satisfying inequality (5). A guaranteed robustness margin is obtained as the inverse of β^*. It is worth emphasizing that (5) is just a sufficient condition of robust stability. Thus the inverse of β^* is only a lower bound of the robustness margin.

The algorithms detailed in section 2.1 can then be adapted to compute a guaranteed robustness margin over the frequency range by replacing the optimisation problem of proposition 2.2 by the generalized eigenvalue problem of proposition 2.3.

Robustness test

The previous algorithms dedicated to the robustness margin computation are adapted as follows:

- The optimization performed at step 2 is interrupted as soon as β falls below a target value $(1 - \tau)\beta_{max}$ with $\tau > 0$ and the algorithm continues to step 3. If at the end of step 2, β remains greater than β_{max}, it means that the sufficient condition of robust stability (5) cannot be achieved and the algorithm stops.
- The validation at step 3 is performed with a test value $\beta = \beta_{max}$ instead of $\beta = (1+\epsilon)\beta_N$ in the case of a robustness margin computation. If all frequencies are eliminated, the interconnection $N(s) - \Delta$ remains stable $\forall \Delta \in \frac{1}{\beta_{max}} B\Delta$ and thus robust stability is achieved. Otherwise, a worst-case value of the frequency is added to the gridding and the algorithm goes back to step 2.

3 Installation of the toolbox

First note that the μ-Analysis and Synthesis Toolbox, the LMI Control Toolbox and the Control System Toolbox are required. Several routines from the Skew Mu Toolbox [1] are also necessary. They are gathered in the specific subdirectory RFDT/Routines_SMT to avoid a complete installation of this toolbox. Note also that the LFR Toolbox [4] can prove useful to put a system under an LFT form. To install the toolbox, follow the instructions below:

- download the file RFDT.zip (*http://www.cert.fr/dcsd/THESES/roos*) and place it in your MatlabTools directory.
- unzip the file with the command `unzip RFDT.zip` ; a directory RFDT will be created.
- edit `paths_init.m` and enter the root directories where you placed the RFD Toolbox and eventually the LFR Toolbox.
- run `paths_init.m`.

You are now ready to use the toolbox, whose content is described in the table below. For the sake of conciseness, only the main Matlab routines are listed. Read sections 4 and 5 and the `README` file to get more information about these routines.

Subdirectories	Files
RFDT	demo_feedforward.m
	demo_performance.m
	demo_stability.m
	paths_init.m
	README
RFDT/Feedforward	feedforward_design.m
	feedforward_design_on_gridding.m
RFDT/Feedforward/Applications	...
RFDT/Performance	robust_performance.m
	robust_performance_on_gridding.m
RFDT/Performance/Applications	...
RFDT/Stability	robust_stability.m
	robust_stability_on_gridding.m
RFDT/Stability/Applications	...
RFDT/Routines	...
RFDT/Routines_SMT	...

Table 1: Subdirectories and main Matlab routines

4 Basic use of the main Matlab routines

4.1 Robust feedforward design

The main routine is `feedforward_design.m`. It performs a convex design of a robust feedforward controller H(s), which ensures H_∞/L_2 performance in the face of LTI and arbitrarily time-varying model uncertainties, as described in section 2.1. With reference to figure 1.a, the considered LFT model is $P(s) - \Delta$.

Syntax:

```
[gamma,sys_H,gridding,xopt]=...
feedforward_design(sys_lft,blk,design{,options,target,init});
```

Inputs:

- `sys_lft`: state-space representation of the plant $P(s)$ (ss format) with inputs $\begin{pmatrix} w \\ y_r \\ u_r \end{pmatrix}$ and outputs $\begin{pmatrix} z \\ y \end{pmatrix}$ listed in this specific order. y_r and y must have the same dimension, so if the real dimension of the reference input y_r is lower than the dimension of the output y, fictitious inputs must be added to y_r and the plant $P(s)$ must be adapted accordingly.
- `blk`: matrix that describes the structure of the model perturbation Δ, which must be square. The first two columns follow the standard of the Mu-Analysis and Synthesis Toolbox whereas the third one defines the type of uncertainty (0 = LTI and 1 = time-varying).
- `design`: structure that contains design specifications:
  - `list_poles`: poles of the feedforward filters $H_i(s)$; in the case of complex poles, only one has to be specified.
  - `gridding`: initial (small size) frequency gridding $(\omega_i)_{i\in[1,\,N]}$.
  - `interval`: frequency interval on which feedforward design is performed.
- `options`: optional structure that contains all tuning parameters.
- `target`: if `target` is nonzero, the algorithm stops as soon as the value of γ on the gridding becomes greater than `target`.

- `init`: optional structure that contains a user-defined initialization of the algorithm:
  - `theta`: column vector with all the design parameters θ_i (default: $\theta_i = 0$).
  - `D2` and `G2`: constant scaling matrices associated with the time-varying uncertainties (default: $D_2 = I$ and $G_2 = 0$).

The use of the optional parameters `options`, `target` and `init` is detailed in section 5.

Outputs:

- `gamma`: guaranteed lower and upper bounds γ_{LB} and γ_{UB} of γ^*. If no lower/upper bound has been computed, `gamma(1)/gamma(2)` is set to 0/`Inf`.
- `sys_H`: optimal feedforward controller $H(s)$ (ss format).
- `gridding`: final frequency gridding.
- `xopt`: structure that contains the optimal values of `theta`, `D2` and `G2`.

Launch the routine `demo_feedforward.m` for an easy start. A valid mat-file is required that contains the following variables: `sys_lft`, `blk`, `list_poles`, `gridding` and `interval`. Then select between:

- feedforward filters $H_i(s)$ can be added progressively during the design process (0) or simultaneously at the beginning (1).
- either the suboptimal algorithm (0), the optimal algorithm (1) or the heuristics (2) can be applied (the use of the heuristics is detailed in section 5.2).

Finally validate the default tuning parameters or see section 5.1 for comments on how to adjust them.

4.2 Robust performance analysis

The main routine is `robust_performance.m`. It performs a robust H_∞/L_2 performance analysis in the presence of LTI and arbitrarily time-varying model uncertainties, as described in section 2.2. With reference to figure 1.b, the considered LFT model is $M(s) - \Delta$.

Syntax:

```
[gamma,gridding,xopt]=...
robust_performance(sys_lft,blk,interval,gridding{,options});
```

Inputs:

See section 4.1 except for `sys_lft`, which is the state-space representation of the plant $M(s)$ (ss format) with inputs $\begin{pmatrix} w \\ y_r \end{pmatrix}$ and outputs $\begin{pmatrix} z \\ y \end{pmatrix}$ listed in this specific order. y_r and y must have the same dimension.

Outputs:

See section 4.1 except for `xopt`, which is a structure that contains the optimal values of `D2` and `G2`.

Launch `robust_performance.m` for an easy start. A valid mat-file is required that contains the following variables: `sys_lft`, `blk`, `interval` and `gridding`. Then choose between suboptimal (0) or optimal (1) algorithms. Finally validate the default tuning parameters or see section 5.1 for comments on how to adjust them.

4.3 Robust stability analysis

The main routine is `robust_stability.m`. It performs a robust stability analysis in the presence of LTI and arbitrarily time-varying model uncertainties, as described in section 2.3. With reference to figure 1.c, the considered LFT model is $M_{11}(s) - \Delta$.

Syntax:

```
[beta,gridding,xopt]=...
robust_stability(sys_lft,blk,interval,gridding{,test_value,options});
```

Inputs:

See section 4.1 except for:

- `sys_lft`: state-space representation of the plant $M_{11}(s)$ (ss format) with input w and output z.
- `test_value`: if `test_value` $= 0$, a guaranteed robustness margin is computed on the given frequency interval ; if `test_value` > 0, it is checked whether $\beta^* \leq \beta_{max}$ with $\beta_{max} =$ `test_value`, i.e. whether the robustness margin is greater than $\frac{1}{\beta_{max}}$.

Outputs:

- `beta`: guaranteed lower and upper bounds β_{LB} and β_{UB} of β^*. If no lower/upper bound has been computed, `beta(1)/beta(2)` is set to 0/`Inf`. In the case of a robustness test, `beta` is set to `[]` if $\beta^* >$ `test_value`.
- `gridding`: final frequency gridding.
- `xopt`: structure that contains the optimal values of `D2` and `G2`.

Launch `robust_stability.m` for an easy start. A valid mat-file is required that contains the following variables: `sys_lft`, `blk`, `interval` and `gridding`. Then choose between:

- guaranteed robustness margin computation (`test_value` $= 0$) or robustness test (`test_value` > 0).
- suboptimal (0) or optimal (1) algorithm.

Finally validate the default tuning parameters or see section 5.1 for comments on how to adjust them.

5 Advanced use of the toolbox

5.1 Tuning parameters

`options` is a structure that can be specified when launching `feedforward_design.m` (F), `robust_performance.m` (P) or `robust_stability.m` (S). It contains all the tuning parameters:

- `addfilters` (F): feedforward filters are added progressively during the design process (0) or simultaneously at the beginning (1) (default=1).
- `optimal` (FPS): optimal (1) or suboptimal (0) algorithm (default=0 for a robustness test and 1 otherwise).
- `first_opt` (FPS): if the suboptimal algorithm is applied, either the optimal (1) or the suboptimal (0) algorithm can be used to perform the first iteration (default=1).
- `tol_dichotomy` (FPS): required accuracy on γ or β for the dichotomy search performed at step 2a (default=0.05).
- `option_mu` (FPS): tuning parameter of mu.m for the computation of the frequency dependent scalings performed at step 2a (default='c').

- `options_mincx` (FP): options of mincx.m for the LMI optimization performed at step 2b (default=[0.01 0 0 0 1]). See the help of *mincx.m* in the LMI Control Toolbox.
- `options_gevp` (S): options of gevp.m for the LMI optimization performed at step 2b (default=[0.05 20 1e5 0 1] for a robustness margin computation or [0.05 0 1e5 0 1] for a robustness test). See the help of *gevp.m* in the LMI Control Toolbox.
- `alpha` (FP): parameter that allows to release the constraints between optimization and validation ($0 \leq \alpha \leq 0.01$; default = 0.01).
- `tau` (S): in the case of a robustness test, LMI optimization stops as soon as β falls below a target value $(1-\tau)\beta_{max}$ (default=0.05).
- `eta` (FPS): if the suboptimal algorithm is applied, the algorithm goes on with the γ or β minimization process of step 2 as long as $\gamma_{N_{min}} < (1-\eta)\gamma_N$ (default=0.05).
- `intervals_valid` (FPS): validation at step 3 is performed either on the whole frequency interval (0) or only on the intervals where the scalings $D_1(\omega_i)$, $G_1(\omega_i)$ determined at step 2 are not valid (1) (default=1).
- `epsilon` (FPS): maximal gap between the lower and upper bounds of the objective γ^* ; at the end of the algorithm, $\gamma_{UB} \leq (1+\epsilon)\gamma_{LB}$ or $\beta_{UB} \leq (1+\epsilon)\beta_{LB}$ (default=0.05 ; useless for a robustness test).
- `method_valid` (FPS): validation at step 3 is performed either with LMI tools (1) or with mu.m (0) (default=0 for a robustness test and 1 otherwise).
- `options_valid` (FPS): options of gevp.m in the case of an LMI validation (default=[1e-5 0 1e5 10 1]) or of mu.m (default='c') in the case of a $\overline{\sigma}$ validation.
- `reduction` (FPS): parameter that allows (1) or not (0) to minimize the value of ϵ and therefore of γ_{UB} or β_{UB} at the end of the algorithm (default=1 ; useless for a robustness test).
- `nu` (F): if the heuristics are applied, this parameter allows to reduce the size of the frequency gridding at the end of the suboptimal algorithm ($0 \leq \nu \leq 1$; $\nu = 1$ means no reduction ; default=0.1).

Remark: `method_valid` enables to achieve a trade-off between the computational time and the convergence of the algorithms: if validation at step 3 is performed with mu.m instead of LMI tools, the computational time will be lower but the algorithms may sometimes fail to converge owing to the suboptimality of mu.m.

5.2 Heuristics

This section explains how to perform the heuristics described in section 2.1. The suboptimal algorithm is first applied. The parameter ν can be defined to tighten the frequency gridding at the end of this first step:

```
options.optimal=0;
options.nu=0.1;
[gamma_UB,sys_H,gridding,xopt]=...
feedforward_design(sys_lft,blk,design,options);
```

`gamma_UB(2)` is a guaranteed upper bound of γ^* on the frequency gridding. Then the optimal algorithm is applied. It is initialized with the tightened frequency gridding and the values of $H(s)$, D_2 and G_2 obtained at the end of step 1 that are respectively contained in `gridding` and `xopt`. The value of `target` is also defined so that the algorithm stops as soon as $\gamma > (1-\xi)\gamma_{UB}$ on the gridding.

```
options.optimal=1;
design.gridding=gridding;
init=xopt;
xi=0.05;target=(1-xi)*gamma_UB(2);
[gamma_LB,sys_H,gridding,xopt]=...
feedforward_design(sys_lft,blk,design,options,target,init);
```

`gamma_LB(1)` is a guaranteed lower bound of γ^*. Finally, `gamma` is obtained as follows:

```
gamma=[gamma_LB(1) gamma_UB(2)];
```

References

[1] G. Ferreres and J.M. Biannic. A Skew Mu Toolbox (SMT) for robustness analysis. *Available at http://www.cert.fr/dcsd/idco/perso/Ferreres/index.html and http://www.cert.fr/dcsd/idco/perso/Biannic/mypage.html*, 2004.

[2] G. Ferreres and C. Roos. Efficient convex design of robust feedforward controllers. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6460–6465, Seville, Spain, December 2005.

[3] G. Ferreres and C. Roos. Robust feedforward design in the presence of LTI/LTV uncertainties. *International Journal of Robust and Nonlinear Control*, 17(14):1278–1293, 2007.

[4] J-F. Magni. User manual of the Linear Fractional Representation Toolbox (version 2.0). *Available at http://www.cert.fr/dcsd/idco/perso/Magni/*, 2006.