R-RoMulOC: A unified tool for randomized and robust multiobjective control*

Mohammadreza Chamanbaz
Singapore University of Technology and Design, Singapore

Fabrizio Dabbene
CNR-IEIIT, Politecnico di Torino, Italy

Dimitri Peaucelle
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France

Roberto Tempo
CNR-IEIIT, Politecnico di Torino, Italy

February 24, 2015

Abstract
In this paper, we describe the salient features of a newly released toolbox for Matlab in dealing with systems with uncertainties. The toolbox combines classical robust multiobjective design techniques with the recently developed randomized approach, providing to the user a wide range of tools for performing deterministic and probabilistic analysis and design.
Keywords
Robust control; randomized methods; systems with uncertainties; LMIs

1 Introduction

In the past years, the field of analysis and design of systems in the presence of uncertainty has witnessed an increase of interest, due to the development of novel and efficient theoretical and numerical tools, see [21].

In particular, two main paradigmatic approaches to robustness (ability of the system to maintain stability and performance under large variations of the system parameters) have gained popularity. The worst-case, or deterministic, paradigm aims at guaranteeing a desired level of performance *for all* system's configurations. This approach has largely benefited from the introduction of the

*Research partly funded by a CNR-CNRS bilateral project No. 134562

linear matrix inequalities (LMIs) formalism, which led to many important results, allowing to tackle a large variety of performances and uncertainty models. The corresponding numerical tools have been collected in a MATLAB toolbox named Robust MultiObjective Control toolbox (ROMULOC) [19]. The toolbox provides some simple functions for manipulating uncertain systems and building LMI optimization problems related to robust multiobjective control problems.

The deterministic approach can be seen as "pessimistic," in the sense that the certified (and guaranteed) performance is usually significantly worse than the actual worst case performance, due to unavoidable conservatism of the developed methodologies. This motivated the introduction of a probabilistic approach [25, 7], which consists in testing a finite number of configurations among the infinitely many admissible ones. This approach is said to be "optimistic" in the sense that if a level of performance is valid for all tested cases, it may not hold for some of the unseen instances. However, rigorous theoretical results, based on large-deviation inequalities, have been derived to bound the probability of performance violation. This theory has now reached a sufficient level of maturity, and the main algorithms have been coded in the Randomized Algorithms Control Toolbox (RACT) [26] which can be freely downloaded from `http://ract.sourceforge.net/pmwiki/pmwiki.php/`. This toolbox allows the user to define and manipulate various types of probabilistic uncertainties, providing efficient sampling algorithm for the different uncertainty types commonly encountered in robust control. Furthermore, it includes sequential and batch randomized algorithms for control systems design.

This paper presents another step in the direction of providing a complete and manageable set of tools in dealing with uncertain systems. The R-ROMULOC toolbox represents an effort which merges the functionalities of the two toolboxes in an integrated framework. The idea behind the toolbox is that the user inputs the system he/she is designing only once, exploiting the well tested formalism of ROMULOC. Then, both deterministic and probabilistic methods can be applied on the same system, efficiently passing from a deterministic to a probabilistic description of the uncertain parameters simply by changing a definition.

As the two tools from which it originates, R-ROMULOC is freely distributed, and can be downloaded at `http://projects.laas.fr/OLOCEP/rromuloc/`. We refer the interested reader to this webpage for a detailed list of references to the various worst-case and probabilistic methods which are coded in R-ROMULOC.

The paper is organized as follows. First the different uncertainty models are presented in Section 2, together with some discussion on the available algorithms for generating random samples. In Section 3 the algorithms for deterministic and probabilistic performance analysis are briefly summarized, with the aid of a simple numerical example. Finally, multiobjective design problems are presented in Section 4. Conclusions are drawn along with a description of future developments.

2 Systems with uncertainty in R-RoMulOC

In this section, we present the two types of systems with uncertainties that can be handled in R-RoMulOC. For each of them, we discuss the sets in which the uncertainties lie (sets that are used in deterministic methods), and outline the different options for drawing random samples according to predefined distributions (sampling that is used in randomized methods).

2.1 Polytopic type

Systems in R-RoMulOC are defined in state-space as

$$\begin{pmatrix} \sigma x \\ z \\ y \end{pmatrix} = \begin{bmatrix} A & B_w & B_u \\ C_z & D_{yw} & D_{zu} \\ C_y & D_{yz} & D_{yu} \end{bmatrix} \begin{pmatrix} x \\ w \\ u \end{pmatrix} = M \begin{pmatrix} x \\ w \\ u \end{pmatrix} \tag{1}$$

where σ represents s-operator (time derivative) in continuous time systems and z-operator (time shift) in discrete time systems; w/z represents a pair of vectors that define some input-to-output performance; the pair u/y represents the input/output vectors used for control. For a precise description of the systems matrices involved in (1) the reader can refer e.g. to [27].

The most simple class of uncertain systems is when M is an interval matrix $M \in [\underline{M},\ \overline{M}]$, where the notation means that the coefficients m_{ij} of M are all independent and bounded in the intervals $[\underline{m}_{ij},\ \overline{m}_{ij}]$. This class of models is allowed in R-RoMulOC and is a special case of the more general class of parallelotopic models defined by

$$M = M_0 + \sum_{a=1}^{\bar{a}} q_a M_a \quad , \quad q_a \in [-1,\ 1],$$

where the matrices $M_a,\ a = 1, \ldots, \bar{a}$ correspond to the axes of the polytope. From a geometric point of view parallelotopes are generalizations of parallelograms in the space of matrices, while interval matrices are parallelotopes with sides that are orthogonal to the canonical basis matrix vectors (with one non-zero element). For these parallelotopic models R-RoMulOC allows uniform sampling that amounts to sample uniformly each q_a assumed independent.

Another way of representing parallelotopes is to define them as the convex hull of the $2^{\bar{a}}$ vertices obtained when taking the extremal values of the q_a coefficients ($q_a = \pm 1$). Parallelotopes are from this point of view subcases of the more general polytopic modeling where the uncertain matrices are affine in the uncertainties q defined on the unitary simplex

$$M = \sum_{v=1}^{\bar{v}} q_v M^{[v]} \quad , \quad q_v \geq 0 \quad , \quad \sum_{v=1}^{\bar{v}} q_v = 1.$$

and where $M^{[v]}$ are the vertices. Sampling uniformly over the polytope is a very difficult task [25] and is not coded in R-RoMulOC. An approximation is

coded that samples over the simplex but, except for the case of systems defined as parallelotopes or polytopes with three or less vertices, the result is not a uniform sampling over the set.

2.2 LFT type

The second type of systems with uncertainties goes beyond the restriction of affine dependence imposed by polytopic models. By means of the so-called linear-fractional transformation (LFT), it allows to represent any matrix $M(\Delta)$ in (1) which is rational in the uncertainty Δ. The notation for LFT type uncertain models in R-RoMulOC is as follows

$$\begin{pmatrix} \sigma x \\ z_\Delta \\ z \\ y \end{pmatrix} = \begin{bmatrix} A & B_\Delta & B_w & B_u \\ C_\Delta & D_{\Delta\Delta} & D_{\Delta w} & D_{\Delta u} \\ C_z & D_{z\Delta} & D_{zw} & D_{zu} \\ C_y & D_{y\Delta} & D_{yw} & D_{yu} \end{bmatrix} \begin{pmatrix} x \\ w_\Delta \\ w \\ u \end{pmatrix} = H \begin{pmatrix} x \\ w_\Delta \\ w \\ u \end{pmatrix} \quad (2)$$

with uncertainties entering as a feedback loop $w_\Delta = \Delta z_\Delta$ in which Δ is a block diagonal matrix of "elementary" uncertainties that can be scalars q_i or matrix valued Δ_i, and that can be repeated on the diagonal. Closing the feedback-loop in the w_Δ/z_Δ channel allows to model a system of the form (1) with $M(\Delta)$ rational in the diagonal elements of Δ. In R-RoMulOC the user is supposed to enter the model in this LFT format. Tools for building such LFT type representation are available in Matlab, see the LFR-toolbox [15] and the Robust Control Toolbox [2].

R-RoMulOC can handle several type of uncertainties in the LFT formulation. It includes interval, parallelotopic and polytopic matrix uncertainties, as well as real or complex valued 2-norm bounded, passive or more general quadratically constrained dissipative operators. Deterministic robust methods have been developed for all these types of uncertainties and are coded in R-RoMulOC. Concerning randomized methods, sampling is possible assuming uniform distributions for interval, parallelotopic and bounded quadratically constrained uncertainties. Within randomized methods some other uncertainties are available, namely p-norm bounded uncertainties with uniform distributions [] and uncertainties with Gaussian distributions [3].

Example 1 *To illustrate the LFT type of models we provide below the display in* Matlab *of one example that is fully described in the code that can be downloaded at the following address* `http://projects.laas.fr/OLOCEP/rromuloc/rocond2015.html`.

```
>> usys
Uncertain model : LFT
-------- WITH --------
name: mechanical system in R-Romuloc demo #1
                n=8     md=6    mw=2    mu=2
 n=8    dx  =   A*x +  Bd*wd +  Bw*w +  Bu*u
pd=7    zd  =  Cd*x + Ddd*wd + Ddw*w + Ddu*u
```

```
pz=1     z  =  Cz*x + Dzd*wd
continuous time ( dx : derivative operator )
-------- AND  --------
diagonal structured uncertainty
size: 6x7 | nb blocks: 4 | independent blocks: 4
 wd = diag( #2 #3 #4 #6 ) * zd
index   size    repeat. RorC        nature
#2  1x2 1   real    LTI {X,Y,Z}-dissipative
#3  2x2 1   real    LTI norm-bounded by 0.25
#4  1x1 1   real    LTI interval 1 param
#6  2x2 1   real    LTI polytope 3 vertices
```

In this example the LTI part of the LFT is an 8th order plant with two disturbance inputs and one performance output. The uncertainty Δ is composed of four blocks. The first one is a 1-by-2 matrix constrained in some ellipsoid, the second is a 2-by-2 matrix in a 2-norm ball, the third is a scalar defined in an interval and the last one is a 2-by-2 matrix constrained in a polytope of 3 vertices. In this example the uncertainties are not repeated. By default when sampling the uncertainties a uniform distribution is assumed over the sets. Random samples $\Delta^{(i)}$ of the uncertainty and samples of the uncertain system evaluated at $\Delta^{(i)}$ can be obtained entering the following commands.

```
>> usample(usys.uncertainty)
ans =
-0.17  0.06     0     0     0     0     0
    0     0  0.02  0.13     0     0     0
    0     0 -0.19 -0.09     0     0     0
    0     0     0     0 -0.67     0     0
    0     0     0     0     0  0.20  0.22
    0     0     0     0     0     0  0.11
>> usample(usys)
               n=8    mw=2
 n=8    dx  =   A*x +  Bw*w
pz=1     z  =  Cz*x
continuous time ( dx : derivative operator )
```

We remark that, contrary to the RACT toolbox, R-RoMulOC is limited to plants that are rational in the uncertainties, due to the LFT representation. In RACT the definition of more general non-rational dependence of the plant on the uncertainty is possible, at the expense of a more involved syntax, which requires to define the plant in a separate MATLAB file. The advantage of the LFT formulation adopted in R-RoMulOC, besides its better usability, is that the sampling procedure does not involve file-calls, operation that is computationally demanding when repeated many times.

3 Analysis problems R-RoMulOC can solve

The first main feature of R-RoMulOC is to provide in a unified framework different available tools for analyzing the robust performance of the uncertain

systems defined in Section 2. In particular, it is possible to check if several performance criteria, as for instance $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norms, impulse-to-peak response, pole location, etc., hold either robustly or with a guaranteed level of probability.

In the next few subsections, we present different techniques used in R-RoMulOC, which depend on the type of uncertainty appearing in the system description and the type of techniques used for solving the problems. We first discuss the probabilistic approach to robust analysis, presenting the algorithms for assessing in a probabilistic way the performance of the system.

3.1 Randomized Worst Case Performance

Given a system affected by parametric uncertainty $\Delta \in \mathbf{\Delta}$, randomized algorithms can be used to assess the so-called *probabilistic worst-case performance*, that is a level of performance guaranteed in probability. To this end, one can introduce a performance function $f(\Delta)$, representing for instance the $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norms of the uncertain system. Then, letting $\varepsilon \in (0,1)$ and $\delta \in (0,1)$ be assigned probability levels, the objective of the randomized algorithm is to return with probability at least $1-\delta$ a performance level $\widehat{\gamma}_N \leq \sup_{\Delta \in \mathbf{\Delta}} f(\Delta)$ such that

$$\Pr\{f(\Delta) \leq \widehat{\gamma}_N\} \geq 1 - \varepsilon. \tag{3}$$

That is, with probability higher than $1-\epsilon$ the system enjoys a performance lever lower than $\widehat{\gamma}_N$. Such randomized algorithm can be derived based on a Monte-Carlo algorithm ([17, 16]). The procedure is formally presented in Algorithm 1.

Algorithm 1 Randomized Worst-Case Performance

1. Choose an integer N satisfying (4).
2. Extract a multisample $\Delta^{1\dots N} = \{\Delta(1), \dots, \Delta^{(N)}\}$ according to the underlying probability distribution.
3. Return the empirical maximum

$$\max_{i=1,\dots,N} f(\Delta^{(i)}).$$

The sample bound N used in Algorithm 1 is derived in [24] stating that if N is chosen according to

$$N \geq \frac{\log \frac{1}{\delta}}{\log \frac{1}{1-\varepsilon}} \tag{4}$$

then $\widehat{\gamma}_N$ satisfies (3) with probability at least $1-\delta$.

Example 1 (cont.): The goal is to apply Randomized Algorithm 1 to compute the probabilistic worst-case $\mathcal{H}_\infty$ norm of the uncertain system. Choosing the probability levels ε and δ to be 10^{-3} and 10^{-6} respectively, the sample bound

(4) becomes 13,809. The computation using R-RoMulOC of an empirical worst-case $\mathcal{H}_\infty$ norm is done as follows.

```
>> quiz_Pr = ctrpb('analysis', 'rand')+hinfty(usys);
>> opts=randsettings('epsilon',1e-3,'delta',1e-6);
>> H_Pr=solvesdp(quiz_Pr,opts)
Hinfty norm = 2.49742 is the worst case estimate
The probability P for an other sample to violate the estimate is ...
    such that
  Prob{ P > 0.001 } < 1e-06
```

The computation[1] takes about 170 seconds where 95 seconds are spent in generating random samples of the uncertainty, 75 seconds are spent in computing $\mathcal{H}_\infty$ norms and only 3 seconds are spent in computing the plant model for given samples of the uncertainties.

3.2 Randomized Performance Verification

Randomized algorithms can be used for probabilistic performance verification. In such a case the objective is to estimate the probability of a given level γ of performance being satisfied, for instance estimating the probability of instability or the probability that the $\mathcal{H}_\infty$ norm of the system is below a given level. Given the performance function $f(\Delta)$ and associated level γ, the randomized Algorithm 2 returns with probability at least $1-\delta$ an estimate $\widehat{\mathbf{p}}_N(\gamma)$ such that

$$\left|p(\gamma) - \widehat{\mathbf{p}}_N(\gamma)\right| < \varepsilon$$

where $p(\gamma) = \Pr\{f(\Delta) \leq \gamma\}$.

Algorithm 2 Randomized Performance Verification

1. Choose an integer N satisfying (5).
2. Extract a multisample $\Delta^{1\ldots N} = \{\Delta^{(1)}, \ldots, \Delta^{(N)}\}$ according to the underlying probability distribution.
3. Return the empirical probability

$$\widehat{\mathbf{p}}_N(\gamma) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\Delta^{(i)})$$

where $\mathbb{I}(\Delta^{(i)}) = \begin{cases} 1 \text{ if } f(\Delta^{(i)}) > \gamma \\ 0 \text{ otherwise} \end{cases}$.

[1]The computation time is given here in terms of CPU time as reported by the `profile` function in Matlab.

The sample bound N used in Algorithm 2 is chosen based on the well-known Chernoff bound ([10])

$$N \geq \frac{1}{\varepsilon^2} \log \frac{2}{\delta}. \tag{5}$$

Example 1 (cont.): Let us apply now Randomized Algorithm 2 to evaluate the probability of the $\mathcal{H}_\infty$ norm of the uncertain system exceeding the level $\gamma = 2.948$. Selecting probability levels $\varepsilon = 10^{-2}$ and $\delta = 10^{-3}$), the sample bound (5) becomes $26,492$. The verification using R-RoMulOC of the probability of $\mathcal{H}_\infty$ norm less than γ is performed as follows.

```
>> quiz_Pr2 = ctrpb('analysis', 'rand')+hinfty(usys, 2.498);
>> opts=randsettings('epsilon',1e-2,'delta',1e-2);
>> solvesdp(quiz_Pr2,opts)
Hinfty norm < 2.498 over all uncertainties with estimated ...
    probability P_est=99.996%
The true probability P is such that
  Prob{ |P-P_est| > 0.01 } < 0.01
```

The computation takes about 321 seconds where 179 seconds are spent in generating random samples of the uncertainty and 141 seconds are spent in computing $\mathcal{H}_\infty$ norms. It is planned to improve this computation time by avoiding the computation of $\mathcal{H}_\infty$ norms for each sample, by rather simply checking whether the $\mathcal{H}_\infty$ norm is below the prescribed level.

3.3 Deterministic Worst-Case Performance

As seen in the example above, the randomized tools allow, by means of sampling, to assess robust performances in a probabilistic sense. As already discussed, this is an optimistic point of view on the robustness problem, in the sense that the probability for the level of performance to be violated is small but it is non-zero. The deterministic approach, on the contrary, provides a pessimistic view point on the problem: it guarantees that a level of performance will not be violated whatever the uncertainties are. But this level is in general conservative.

The deterministic methods implemented in R-RoMulOC are based on Lyapunov-type certificates. For example, proving the stability of a continuous-time system $\dot{x} = Ax$ can be done by finding a solution P to the LMI problem

$$P \succ 0, \quad A^T P + PA \prec 0.$$

A classical robust version of this test for uncertain systems is the search of a common certificate P for all uncertainties. In the case of uncertain polytopic systems this problem boils down to the search of a common P for a subset of vertices [1]

$$P \succ 0, \quad A^{[v]T} P + PA^{[v]} \prec 0 \quad , \quad \forall v = 1, \ldots, \bar{v}.$$

In case of LFT models the formulas are slightly more complex and involve both a Lyapunov certificate and some quadratic separator Θ which is structured

according to the structure and contents of the Δ operator. The LMIs are of the following type

$$P \succ 0, \quad N_A^T \begin{bmatrix} 0 & P \\ P & 0 \end{bmatrix} N_A \prec N_C^T \Theta N_C$$

with appropriately defined matrices N_A and N_C and LMI constraints on Θ.

Example 1 (cont.): We now apply these deterministic results involving a parameter independent Lyapunov function to compute a deterministic upper bound on the worst-case $\mathcal{H}_\infty$ norm of the uncertain system. The computation using R-RoMulOC is done as follows.

```
>> quiz_pilf = ctrpb('analysis', 'PILF')+hinfty(usys);
>> H_pilf=solvesdp(quiz_pilf)
Hinfty norm < 2.86659 assessed
```

The computation time takes 6.5 seconds on the same computer when solving the LMIs with SDPT3 solver. The obtained value is clearly conservative when compared to the probabilistic worst-case values obtained above.

A significant difference between the deterministic upper-bound and the probabilistic estimate is that in the latter case uncertainties are assumed to be parametric (constant) while the former deterministic formulas apply even in the case of time-varying or non-linear uncertain operators. The gap is thus justified. A time-varying uncertainty that stays in the prescribed bounds may produce a larger L_2-induced norm (which is the more appropriate definition than "$\mathcal{H}_\infty$ norm" when dealing with time-varying or non-linear systems) than parametric time-invariant uncertainty.

3.4 Improved Deterministic Worst-Case Performance

In case uncertainties are known to be parametric, the literature offers many results for assessing robustness with the help of parameter-dependent Lyapunov functions. In R-RoMulOC two such results are coded. For polytopic systems we allow the search for Lyapunov certificates of the form $P(\xi) = \sum_{v=1}^{\bar{v}} \xi_v P^{[v]}$, based on the so-called S-Variables approach described in [12]. In the case of LFT systems, we allow the search for Lyapunov certificates of the type $P(\Delta) = R(\Delta)^T \hat{P} R(\Delta)$ where $R(\Delta)$ is affine in the rational expression $\Delta(I - D_{\Delta\Delta}\Delta)^{-1} C_\Delta$. These are extensions of results from [13, 20].

Example 1 (cont.): We apply these improved deterministic results involving parameter-dependent Lyapunov functions to compute a deterministic upper bound on the worst-case $\mathcal{H}_\infty$ norm of the system with parametric uncertainties. The computation using R-RoMulOC is done as follows.

```
>> quiz_pdlf = ctrpb('analysis', 'PDLF')+hinfty(usys);
>> H_pdlf=solvesdp(quiz_pdlf)
Hinfty norm < 2.53004 assessed
```

The computation takes 9.3 seconds on the same computer when solving the LMIs with SDPT3 solver. The obtained value is still conservative when compared to the probabilistic worst-case values obtained above, but with a significantly reduced gap.

Notice that the $\mathcal{H}_\infty$ norm of a system defined by matrices A, B, C, D is exactly equal to the $\mathcal{H}_\infty$ norm of the dual system defined by the matrices A^T, C^T, B^T, D^T. However, in terms of an upper bound computation with parameter-dependent Lyapunov functions, the LMIs constructed for the dual system are different. The Lyapunov certificate involved in LMIs for the dual system correspond to the search of some $P_d(\Delta) = (R_d(\Delta)\hat{P}_d R_d(\Delta)^T)^{-1}$ for the primal system. Where, this time the matrix $R_d(\Delta)$ is affine in $\Delta(I - D_{\Delta\Delta}\Delta)^{-1}B_\Delta$. The computation on the dual system is in R-RoMulOC as follows (notice the conjugate transpose sign on `usys'` that makes all the difference.)

```
>> quiz_pdlf_d = ctrpb('analysis', 'PDLF')+hinfty(usys');
>> H_pdlf_d=solvesdp(quiz_pdlf_d)
Feasibility is not strictly determined
Worst constraint residual is -2.61404e-08 < 0
Hinfty norm < 2.51475 might hold
```

This example illustrates a disadvantage of LMI methods, related to numerical difficulties associated with the solver. In the considered case, the solver stopped with a non strictly feasible solution. Nevertheless, the provided value is rather good as attested by the following verification.

```
>> quiz_pdlf_d2 = ctrpb('analysis', 'PDLF')+hinfty(usys', 2.516);
>> solvesdp(quiz_pdlf_d2)
Hinfty norm < 2.516 assessed
```

3.5 Perspectives for analysis

Ongoing work on analysis tools in R-RoMulOC is in several directions. One direction is the coding of further improved deterministic LMI-based results involving Lyapunov certificates with higher order dependency with respect to parameters. The results will be those obtained via the system augmentation techniques described in [12] for the case of polytopic uncertainties, and in [20, 18] for the case of LFTs. This extension of R-RoMulOC will also be the opportunity to extend the modeling to descriptor systems and mixed polytopic-LFT models.

Another direction is to consider randomized tools for the analysis of uncertain systems with time-varying uncertainties. As exposed in Subsection 3.3, searching for a common Lyapunov certificate for all values of the uncertainties provides a solution for the case of time-varying or non-linear uncertain operators. A probabilistic point of view on this problem is the design of a common Lyapunov matrix that solves the LMI constraints for a large number of samples over the uncertainty set. Such probabilistic LMI-design problems are discussed in the next section.

4 Design problems R-RoMulOC can solve

There is a number of deterministic and randomized approaches for solving design problems in R-RoMulOC. In the next few subsections, we explain these design methods and provide numerical examples for comparison. We limit our attention to state-feedback design.

4.1 Deterministic multiobjective design

The LMI-based deterministic results coded in the R-RoMulOC toolbox are extensions of the common Lyapunov certificate results exposed in the previous section, applied to the dual system in closed-loop with state-feedback $u = Kx$. In case of polytopic uncertain models these conditions write as

$$P_d \succ 0, \quad (A^{[v]} + B^{[v]}K)P_d + P_d({A^{[v]}}^T + K^T {B^{[v]}}^T) \prec 0$$

and are linear in the design variables when applying the well known linearizing change of variable $S = KP_d$. Applying the Lyapunov shaping paradigm [23], this framework allows to build conditions for multiobjective state-feedback design. An example of such multiobjective problem is shown below, where the requirements are to find a state feedback that guarantees the worst-case $\mathcal{H}_\infty$ norm of the closed-loop system to be smaller than one, and at the same time places all closed-loop poles inside a disc of radius 10 and centered at the origin.

```
>> quiz_sf_d = ctrpb('state-feedback')...
+hinfty(usys,1)+dstability(usys,region('disc',0,10));
>> K_d=solvesdp(quiz_sf_d)
Hinfty norm < 1 assessed
D-stability assessed for region Disk centered at 0 with radius 10
K_d =
   -2.6565    0.0015   -1.8447   -0.3888    0.0219   -0.5358   ...
       -0.5831    0.0309
   -0.0174   -2.7051    0.8036   -1.3358   -0.0436    0.3575   ...
       -0.2607   -0.1363
```

The computation takes less than 9 seconds and the LMI problem is defined by 100 decision variables and 87 rows.

4.2 Randomized scenario approach

The main idea in the scenario approach, which has been introduced in [4, 5], is to reformulate a semi-infinite convex optimization problem as a sampled convex optimization problem subject to a finite number of random constraints extracted form the uncertainty set. The scenario approach is a non-sequential randomized method which is capable of solving uncertain optimization problems. It is non-sequential in the sense that the problem is solved in one shot and hence the computational complexity can be determined a priori, i.e. is before the algorithm starts, as opposed to the sequential approaches where the sample

complexity is known after the algorithm terminates. In the following example we solve the state-feedback problem using the scenario approach.

```
>> quiz_sf_s = ctrpb('state-feedback','rand')...
+hinfty(usys,1)+dstability(usys,region('disc',0,10));
>> opts=randsettings('epsilon',1e-1,'delta',1e-2,...
'method','scenario');
>> K_s=solvesdp(quiz_sf_s,opts)
Hinfty norm < 1 assessed
D-stability assessed for region Disk centered at 0 with radius 10
Assessemnts are in a probabilistic sense:
  Prob{ P > 0.1 } < 0.01
where P is the probability for an other sample to violate the LMIs
K_s =
   -2.6702    0.0133   -1.7103   -0.1282    0.0264   -0.5434   ...
       -0.4340    0.0423
   -0.0087   -2.6735    0.4810   -0.4544   -0.0323    0.2438   ...
       -0.1321   -0.1131
```

Based on the selected probabilistic levels ε, δ and on the number of design variables (52 in this case), the number of samples of the uncertain plant generated is equal to 937. It takes about 30 seconds to generate these samples and 34 seconds to define the 937 LMI constraints relative to the $\mathcal{H}_\infty$ and pole location specifications for each closed-loop. The LMIs with the 52 variables and 23,433 rows are solved with SDPT3 solver in 104 seconds. The overall computation time is 184 seconds.

4.3 Randomized sequential approaches

The sequential randomized method in robust controller design includes two main steps:1) Probabilistic Oracle and 2) Update Rule.

Having a candidate solution, a *probabilistic oracle* [24] performs a random check to assess the probabilistic performance of a *candidate solution*. In the case the oracle is successful, the algorithm is terminated and the candidate solution is declared as a *probabilistic robust solution*. Otherwise, the oracle returns a violation certificate to be used in the *update step* for updating the candidate solution. The key point in the probabilistic oracle is to derive the minimum number of sample one needs to check to assess the probabilistic performance of the candidate solution. We remark that the probabilistic oracle used in R-RoMulOC is in agreement with randomized worst-case performance methods of Algorithm 1.

The *update* step is a deterministic procedure based on convex optimization. Having a violation certificate from the oracle, this step exploits the convexity of the problem and computes a new candidate solution to be checked by the oracle. The update rule can be designed based on stochastic gradient [22, 8], ellipsoid [14] and cutting plane [6, 11] iterative methods. In R-RoMulOC the stochastic approximation method based on gradient iteration is used to solve uncertain state-feedback problem.

We remark that the sequential randomized method presented here is capable of solving feasibility problems.

```
>> quiz_sf_g = ctrpb('state-feedback','rand')...;
+hinfty(usys,1)+dstability(usys,region('disc',0,10));
>> opts=randsettings('epsilon',1e-1,'delta',1e-2,...
'method','gradient','Nout',1e5);
>> K_g=solvesdp(quiz_sf_g,opts)
Looking for a probabilistic solution to uncertain LMIs defined by
  Prob{ P > 0.1 } < 0.01
where P is the probability for an other sample to violate the LMIs
Iter #5532 : Oracle passed 213 tests, solution is found!
Hinfty norm < 1 assessed
D-stability assessed for region Disk centered at 0 with radius 10
K_g =
   -2.5934    0.0236   -2.6593   -0.1027    0.0252   -0.6761   ...
       -0.5166    0.0147
   -0.0142   -2.5837    0.6174   -0.3878   -0.0439    0.2908   ...
       -0.1494   -0.1213
```

In this example, a probabilistic solution is found in 5,532 steps of the gradient algorithm. At the last step, the probabilistic oracle successfully checks 213 random samples. The overall computation time is of 3,387 seconds, 2,543 of which are spent in generating random samples.

From the examples and in particular from the last gradient based sequential approach result, one notices that most of the computation time is spent in generating random samples of the uncertainties. When going into the details, most of that computation time is due to sampling over dissipative and norm-bounded uncertainties. Improving the computation needed for generating samples over matrix type uncertainties will have a significant influence on the computation time of the probabilistic methods.

4.4 Perspective for design

The approach based on the scenario approach can be computationally demanding when the probabilistic levels are stringent. New sequential randomized methods capable of solving optimization problems are introduced in [9]. The proposed method is of sequential nature, where in each update step a reduced size scenario problem is solved.

5 Conclusion

The paper highlights the main features of a newly developed toolbox for analysis and design of systems in the presence of uncertainty. R-RoMulOC provides various tools for uncertain systems, using both deterministic as well as randomized methods. It combines the user friendly interface and the LMI based deterministic algorithms of RoMulOC with the powerful randomized methods

provided by RACT. The presented toolbox allows the user to compare randomized and deterministic methods using a unified framework, and to choose the best method which fits his/her requirement.

Acknowledgments

Acknowledgments to all those who contributed to R-RoMulOC in many different ways: D. Arzelier, A. Bortott, G. Calafiore, G. Chevarria, E. Gryazina, B. Polyak, P. Shcherbakov, M. Sevin, P. Spiesser, and A. Tremba.

References

[1] T. Alamo, R. Tempo, D.R. Ramrez, and E.F. Camacho. A new vertex result for robustness problems with interval matrix uncertainty. 57:474–481, 2008.

[2] G. Balas, R. Chiang, A. Packard, and M. Safonov. *Robust Control User's Guide*. MathWorks, 2014.

[3] G. Calafiore, F. Dabbene, and R. Tempo. Radial and uniform distributions in vector and matrix spaces for probabilistic robustness. In D.E. Miller and L. Qiu, editors, *Topics in Control and its Applications*, pages 17–31. Springer London.

[4] G.C. Calafiore and M.C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102:25–46, 2004.

[5] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51:742–753, 2006.

[6] G.C. Calafiore and F. Dabbene. A probabilistic analytic center cutting plane method for feasibility of uncertain LMIs. *Automatica*, 43:2022–2033, 2007.

[7] G.C Calafiore, F. Dabbene, and R. Tempo. Research on probabilistic methods for control system design. *Automatica*, 47:1279–1293, 2011.

[8] G.C. Calafiore and B. T Polyak. Stochastic algorithms for exact and approximate feasibility of robust LMIs. *IEEE Transactions on Automatic Control*, 46:1755–1759, 2001.

[9] M. Chamanbaz, F. Dabbene, R. Tempo, V. Venkataramanan, and Q-G. Wang. Sequential randomized algorithms for convex optimization in the presence of uncertainty. *IEEE Transactions on Automatic Control*, 2013. Submitted for publication,.

[10] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23:493–507, 1952.

[11] F. Dabbene, P. S. Shcherbakov, and B. T. Polyak. A randomized cutting plane method with probabilistic geometric convergence. *SIAM Journal on Optimization*, 20, 2010.

[12] Y. Ebihara, D. Peaucelle, and D. Arzelier. *S-Variable Approach to LMI-Based Robust Control*. Communications and Control Engineering. Springer, 2015.

[13] T. Iwasaki and G. Shibata. LPV system analysis via quadratic separator for uncertain implicit systems. *IEEE Transactions on Automatic Control*, 46:1195–1208, 2001.

[14] S. Kanev, B. De Schutter, and M. Verhaegen. An ellipsoid algorithm for probabilistic robust controller design. *Systems & Control Letters*, 49:365–375, 2003.

[15] J. Magni. User manual of the linear fractional representation toolbox version 2.0. Technical report, Technical report, ONERA - Systems Control and Flight Dynamics, 2005. Available at http://www.onera.fr/smac/.

[16] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[17] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44:335–341, 1949.

[18] D. Peaucelle. *Integral Quadratic Separation and its use for robust control*. Habilitation à diriger des recherches, Université Toulouse, III Paul Sabatier, 2014.

[19] D. Peaucelle and D. Arzelier. Robust multi-objective control toolbox. In *Proceedings of the CACSD Conference*, Munich, Germany, 2006.

[20] D. Peaucelle, D. Arzelier, D. Henrion, and F. Gouaisbaut. Quadratic separation for feedback connection of an uncertain matrix and an implicit linear transformation. *Automatica*, 43:795–804, 2007.

[21] I.R. Petersen and R. Tempo. Robust control of uncertain systems: Classical results and recent developments. *Automatica*, 50:1315–1335, 2014.

[22] B. T. Polyak and R. Tempo. Probabilistic robust design with linear quadratic regulators. *Systems & Control Letters*, 43:343–353, 2001.

[23] C. Scherer, P. Gahinet, and M. Chilali. Multiobjective output-feedback control via LMI optimization. *IEEE Transactions on Automatic Control*, 42:896–911, 1997.

[24] R. Tempo, E.-W. Bai, and F. Dabbene. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. *Systems and Control Letters*, 30:237–242, 1997.

[25] R. Tempo, G.C. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems: With Applications*. Springer, 2nd edition, 2013.

[26] A. Tremba, G.C. Calafiore, F. Dabbene, E. Gryazina, B. Polyak, P. Shcherbakov, and R. Tempo. RACT: randomized algorithms control toolbox for MATLAB. In *Proc. 17th World Congress of IFAC, Seoul*, pages 390–395, 2008.

[27] K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall New Jersey, 1996.