Optimization Methods and Software
Vol. 00, No. 00, Month 200x, 1–15

RESEARCH ARTICLE

Automatic robust convex programming

Johan Löfberg*

Division of Automatic Control, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden

(*Received February 23rd 2010; final version received 29 June 2010*)

This paper presents the robust optimization framework in the modeling language YALMIP, which carries out robust modeling and uncertainty elimination automatically, and allows the user to concentrate on the high-level model. While introducing the software package, a brief summary of robust optimization is given, as well as some comments on modeling and tractability of complex convex uncertain optimization problems.

Keywords: Robust optimization; Conic programming; Modeling software

1. Introduction

Almost all optimization models are wrong, and we know it. In most cases we can live with this, the performance degradation due to modeling errors is too small to be of practical interest, or the application might be inherently robust against small errors (e.g. control systems where the whole idea is to reduce the impact of uncertainty via feedback). On the other hand, there are cases where small model errors can have disastrous effects, or at least lead to unnecessarily poor performance compared to what could have been obtained if uncertainty explicitly had been taken into account.

A common reason for practitioners to avoid uncertainty is the added complexity of building the robust optimization model, and the lack of knowledge of *how* and *when* this can be done efficiently. The purpose of this paper is therefore two-fold. To begin with, we collect some of the most common approaches to structured robust optimization, and try to pin-point typical classes of problems that can be handled efficiently. However, it is our belief that knowing how to develop and manipulate uncertain models does not necessarily lead to actual implementations of robust optimization models, strong software support is required if we want robust optimization to become common in practice. Thus, we introduce and discuss the software support that is available in modeling language YALMIP to implement these uncertain scenarios. Additionally, we discuss the easily overseen fact that models where the nominal high-level problem can be easily reformulated to linear programs in polynomial time, in some cases automatically by software, and we have methods to derive robust counterparts of uncertain linear programs in polynomial time, the combination of the two does not necessarily lead to a polynomial time problem for uncertain high-level convex programs.

*Email: johanl@isy.liu.se

ISSN: 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/1055678xxxxxxxxxxxxx
http://www.informaworld.com

2. Robust optimization

The basic problem addressed in this paper is uncertain optimization problems.

$$\begin{aligned} &\min_x \max_w f(x,w) \\ &s.t \quad g(x,w) \preceq 0 \quad \forall w \in \mathcal{W} \end{aligned}$$

To keep notation at a minimum, we write $g(x,w) \preceq 0$ here, but keep in mind that these constraints may include element-wise inequalities, equality, second-order and semidefinite cone constraints. In fact, as we will see later, it can also include more advanced constraints, such as sum-of-squares constraints. Since the objective can be moved to the constraints by an epigraph reformulation, we can w.l.o.g assume certain objective function and thus concentrate on uncertain constraints.

In short, there are two ways to address this class of problems. Historically, probabilistic approaches with chance constraints, risk-measures and expected outcomes have been common, being the foundation for the whole stochastic programming field [Pre95]. The solution methods are, due to the intractability of most problems, typically based on some type of sampling from an uncertainty set. The disadvantage with this approach is of course that it only gives an (optimistic) approximation, and thus no guaranteed solutions. It is however possible to obtain statistical confidence results on the solution, in particular in the convex case [CC04].

Recently, a more strict approach has become popular. The paradigm here is to exactly or conservatively convert the problem to a certain problem (a so called robust counterpart), by in some way removing the uncertainty, using methods such as explicit maximization, duality properties, or relaxation methods. These approaches have to a large extent gained popularity due to developments in the field of convex and conic optimization [EOL98, BTN98, BTN99, BTN02] with significant impetus from the robust control field [BEFB94, ZDG95]. It should however be made clear that this approach in no sense is a new concept, exact approaches to uncertain linear programming can be found already in, e.g., [Soy73, Soy74, Fal76]. This paradigm, commonly referred to as a worst-case approach is what we concentrate on in this paper (sometimes also referred to as unknown-but-bounded, or minimax).

Many robust optimization problems fall into standard cases, which can be converted to certain counterparts by standard but error-prone and cumbersome reformulations. Our goal is to describe an extension to the modeling language YALMIP [LÖ04] for modeling robust optimization problems in an intuitive format, and let the software package take care of the reformulations. In other words, it is the robust optimization correspondence of the convex programming reformulations in the "nonlinear operator"-framework in YALMIP and the similar "disciplined convex programming" framework in CVX [GB08]. The outcome of the features presented in this paper is a new optimization problem which solves the worst-case scenario.

$$\begin{aligned} &\min_{x,y} \hat{f}(x,y) \\ &s.t \quad \hat{g}(x,y) \leq 0 \end{aligned}$$

Typically, the problem is changed considerably from the original problem, indicated by the new objective $\hat{f}$, constraints $\hat{g}$ and additional variables y, introduced in order to eliminate the uncertainty. While the original uncertain problem, for instance, is a linear program, the robust counterpart can become a semidefinite program. Even worse, the robust counterpart may not even be a tractable problem. In the follow-

ing sections, we will outline the basic ideas in the proposed robust optimization extension, and introduce the uncertainty scenarios that are supported.

At the initial release of the robust optimization framework in 2005, described in [LÖ8], there was no comparable system publically available. Recently however, alternatives have emerged [SG10]. The emphasis is however different. While YALMIP concentrates on the generality and integration of the robust optimization framework into other modules (parametric programming, sum-of-squares, integer programming etc.), the emphasis in [SG10] is largely on uncertainty descriptions using distributions and moments, and various applications of linear decision rules.

3. Notation

Matrices will generically be denoted using capital letters while vectors are in small letters. Inner product $trace(A^TB)$ will be written as $A \bullet B$. Cone constraints are indicated using the $\succeq$ ($\preceq$) operator. The cone can be either the positive (negative) orthant cone, second order cone, or the cone of semidefinite matrices. When a discussion is limited to the standard orthant, we use $\geq$ ($\leq$). To simplify notation, conic constraints in dual form $C - \sum_i^m A_i w_i \succeq 0$ will be written in the operator form $C - \mathcal{A}^T(w) \succeq 0$. Primal form conic constraints, $A_i \bullet X = b_i, i = 1 \dots m$, will be written $\mathcal{A}(X) = b$. The cones might be direct products of several cones, possibly of different type, and the data is partitioned accordingly. For simplicity though, one may think of all cone constraints as having only one element.

4. The philosophy behind the software implementation

The first and most important idea in the robust optimization framework in YALMIP is that uncertain models should be modeled using exactly the same syntax as certain models. Hence, the only addition to the modeling language is a new command `uncertain`, which declares a set of variables as uncertain.

4.1. *Detection*

When YALMIP encounters a model with uncertain variables, three main steps occur. To begin with, the variables explicitly declared as uncertain are detected, and constraints involving only these variables are separated from the model. These constraints constitute (the initial) uncertainty description.

4.2. *Expansion*

YALMIP then applies an expansion of the remaining model, to model all advanced nonlinear operators, such as absolute values and norms, typically using graph representations. Note that the expansion of expressions that only involve uncertain variables might generate new variables and constraints, which have to be added to the list of uncertain variables, and to the uncertainty description. As an example, if the uncertainty description is defined in the modeling language as $|w|_\infty \leq 1$, the set of uncertain variables and associated uncertainty set after the expansion is performed will be $\{(w,t) : -t \leq w \leq t,\ t \leq 1\}$. The introduction of the auxiliary, and strictly speaking redundant, variable t is a consequence of the way YALMIP models nonlinear operators, such as norms.

A complication in terms of implementation is that the modeling language has to

treat the uncertainty as a constant during the convexity propagation and expansion. For instance, the constraint $|xw| \leq 1$ is not convex, and could thus lead to problems when convexity analysis is performed by YALMIP to decide on how to model the absolute value operators. However, the standard graph model $-1 \leq xw \leq 1$ will be derived, since the modeling language temporarily treats w as a constant and thus sees an affine term inside the absolute value operator. After this expansion has been made, the model is conceptually in YALMIP standard form (no high-level operators such as norms, absolute values, etc.), albeit parameterized in the w variables.

4.3. *Classification and filtering*

At this point, the derivation of a robust counterpart takes place. Depending on the class of uncertain constraints, and the uncertainty model, different approaches are used. The process of removing the uncertainty and deriving a robust counterpart is called the filtering step, and there are currently five *filters* implemented, as outlined in the next section.

Once the filter has been applied, a standard YALMIP model with no uncertainty has been generated. This symbolic model can be solved using any installed solver suitable for the robust problem, or manipulated further by the user.

Important to understand is that the uncertainty modeling and the derivation of robust counterparts is a feature that is (essentially) completely integrated in the infrastructure of YALMIP. Hence, nothing prevents a user to, e.g., define uncertain sum-of-squares problems with combinatorial constraints. The modeling language performs the sum-of-squares compilation, uncertainty removal, and addition of combinatorial constraints separately in a modular fashion.

5. The filters

The mechanism of converting a problem with uncertainty, to the corresponding certain counterpart, is called filtering in the robust optimization framework in YALMIP. At the moment, five filters are implemented. The goal is to extend this list in future versions, but the current set of problem classes are considered the most important in practice.

5.1. *Duality filter*

The duality filter is applicable to element-wise constraints with coefficients linearly parameterized in the uncertain variable, and the uncertainty constrained to an intersection of linear, second order and semidefinite cones. Consider a single element-wise constraint

$$(Aw + b)^T x + (c^T w + d) \leq 0 \ \forall w : E - \mathcal{F}^T(w) \succeq 0 \tag{1}$$

By writing the left-hand term as $(A^T x + c)^T w + (b^T x + d)$ and viewing it as the cost of a conic program in w, it follows from duality theory of conic optimization that the maximum of this function, over w, is less than or equal to 0 if and only

if[1] there exist a Z such that the following condition holds [BTN02]

$$E \bullet Z + b^T x + d \leq 0, \mathcal{F}(Z) = A^T x + c, Z \succeq 0 \tag{2}$$

This is a very general and useful result. A major drawback however is that the filter can lead to a substantial increase of problem size, since a new variable Z, and the associated constraints, have to be introduced for every uncertain constraint. Hence, when possible, more specialized filters should be used.

Note that the strong duality arguments are employed in the w-space. Hence, additional complicating constraints on x, such as integral variables or other convex or nonconvex constraints, does not influence the correctness of the method. Of course, this holds also for the remaining four filters.

5.2. *Enumeration filter*

A classical uncertainty case is constraints where the parameterization is linear in the uncertainty, and the uncertainty is constrained to a polytopic set. To simplify notation, let $\mathcal{A}_w(x)$ denote the parameterized operator $\sum A_i(w)x_i$ where each matrix A_i is linearly parameterized in w. Consider the following uncertain conic constraint

$$(\mathcal{C}^T(w) + D) + (\mathcal{B}^T(x) + \mathcal{A}_w^T(x)) \preceq 0 \quad \forall w : Ew \leq f \tag{3}$$

This is the case that arise, e.g., in stability analysis of polytopic systems [BEFB94], where w corresponds to parameters in an uncertain system and x corresponds to the variables parameterizing a Lyapunov matrix.

From convexity, it follows that it is sufficient to study the vertices of the polytope $Ew \leq f$. Hence, if we let $\{w_i\}$ denote the vertex enumeration of the uncertainty polytope, the robustified constraint is given by the intersection of the conic constraint evaluated at the vertices $\{w_i\}$.

$$(\mathcal{C}^T(w_i) + D) + (\mathcal{B}^T(x) + \mathcal{A}_{w_i}^T(x)) \preceq 0 \tag{4}$$

The problem with this approach is obvious; simple polytopes can generate intractably many vertices. As a trivial example, the unit-cube in $\mathbf{R}^n$ has 2^n vertices.

5.3. *Explicit maximization filter*

Specializing the problem structure further, we arrive at a case where we actually can perform the maximization over the uncertainty analytically. Consider an element-wise constraint linearly parameterized in an uncertainty $w \in \mathbf{R}^r$ which is constrained to a norm-ball.

$$(c^T w + d) + (Aw + b)^T x \leq 0 \quad \forall w : |w|_p \leq 1 \tag{5}$$

The maximum over w can be derived by using the fact that $max_{|w|_p \leq 1} q^T w$ is $|q|_{p^*}$ where $|\cdot|_{p^*}$ denotes the dual norm, $\frac{1}{p} + \frac{1}{p^*} = 1$ [BV04]. Hence, the robust constraint is

$$(b^T x + d) + |c + A^T x|_{p^*} \leq 0 \tag{6}$$

[1] Assuming the uncertainty set is bounded and has a strict interior.

The current implementation exploits this result for the (possibly scaled and translated) 1, 2 and ∞-norm case, where the dual norm is the ∞, 2 and 1-norm respectively. As for complexity, it depends on A. If A is nonzero, an uncertainty constrained by a 1-norm will introduce one new variable and $2r$ constraints for each uncertain constraint (epigraph variable and associated constraints to model the dual ∞-norm). An uncertainty constrained in 2-norm will generate one second-order cone constraint per uncertain constraint, whereas an uncertainty constrained in ∞-norm will be somewhat expensive as it forces us to introduce r new variables and $2r$ constraints, for every uncertain constraint, to model the dual 1-norm. If $A = 0$, all approaches are extremely cheap, as they only require an addition of the constant term $|c|_{p^*}$ to the unperturbed constraint.

5.4. *Pólya filter*

The filters above are all exact, in the sense that the robustified constraints are both sufficient and necessary for the original constraints to be robustly satisfied. Unfortunately, there are not many more cases where simple sufficient and necessary counterparts are available [BTN98]. Instead, one has to rely on conservative approximations. One common case where a simple conservative result is available is polynomially parameterized element-wise or semidefinite constraints, with the uncertainty constrained to a simplex[1].

$$p(x,w) \preceq 0 \quad \forall w : \sum_{i=1}^{m} w_i = 1, w \geq 0 \tag{7}$$

Since w is constrained to a simplex, the constraint is trivially equivalent to $p(x,w)\left(\sum_{i=1}^{m} w_i\right)^N \preceq 0$ for arbitrary N. If we assume that $p(x,w)$ is homogeneous[2] in w, the polynomial $p(x,w)\left(\sum_{i=1}^{m} w_i\right)^N \preceq 0$, when seen as a polynomial in w with coefficients parameterized in x, is non-positive if all coefficients are non-positive. This follows trivially since w is non-negative. Hence, a sufficient condition is

$$\text{coefficients}_w\{p(x,w)(\sum_{i=1}^{m} w_i)^N\} \preceq 0 \tag{8}$$

Note that this analogously also holds for the symmetric matrix polynomial case, i.e., when $p(x,w)$ is a matrix polynomial, with matrix coefficients functions of x.

The reason we denote the filter Pólya, is due to a result of Pólya, stating that for a finite sufficiently large N, the condition is necessary[3] [HLP52]. However, a bound on this sufficiently large N is typically unreasonably large, and depends on the parameterized coefficients, so the necessity result is of no direct use to us. Instead, the user has to specify N and hope that the relaxation is sufficiently tight. Analogous necessity in the matrix case has recently been shown in [Sch05].

An important feature of this relaxation, compared to more advanced schemes using recent developments in convex optimization based relaxations of polynomial

[1] This is not necessary, any polytope can be handled by expressing w as a convex combination of its vertices, if such vertex enumeration is available and tractable. The variable used in the convex combination lives, by definition, on a simplex.

[2] This is not a restriction, since any polynomial can be rendered homogeneous on a simplex by multiplying monomial terms with suitable powers of $\sum_{i=1}^{m} w_i$.

[3] To be precise, the theorem concerns a strict inequality of a homogeneous polynomial, and the restriction on the simplex can be relaxed to any set in the positive orthant excluding the origin.

problems, such as sum-of-squares and moment relaxations [Par03, Las09], is that an uncertain element-wise constraint leads to element-wise constraints in the robust counterpart. In other words, the problem class does not change. Nevertheless, since YALMIP has support for sum-of-squares reformulations and moment relaxations [LÖ9], future versions may have support for stronger relaxations in the robust optimization module, at the cost of increased problem complexity.

5.5. *Elimination filter*

If everything else fails, our last resort is to constrain the decision variables such that the uncertainties disappear from the constraint. Consider a polynomially parameterized constraint, with arbitrary uncertainty description.

$$p(x, w) \preceq 0 \quad \forall w \in \mathcal{W} \tag{9}$$

If we see this as a polynomial in w with coefficients parameterized in x, a trivial sufficient condition is obtained by constraining the coefficients to be zero.

$$p(x, w) \preceq 0, \ \text{coefficients}_w\{p(x, w)\} = 0 \tag{10}$$

Although this condition is trivial, it is actually rather useful in some situations, as mentioned in the LPV control example in Section 7.3. Additionally, the elimination filter is the only option when an equality constraint involves uncertain variables (which typically indicates an incorrect model, or the fact that the user wants to solve a recourse problem and fails to see the difference between assignment and equality in a model)

5.6. *Mixing uncertainty models*

We have so far only discussed the case when a constraint involves only one type of uncertainty. It might however be the case that a constraint is affected by several disjoint sets of uncertainties, say one uncertainty constrained to a norm ball and another uncertainty constrained to a general conic set. One approach to address this is to include the norm-ball description in the conic set, and apply the duality filter on the direct product of the two uncertainties. YALMIP will however take care of each part independently in an attempt to derive as small models as possible.

5.7. *Comparison of filter complexity*

As a simple illustration we now apply the three first approaches to derive a robust counterpart. Consider an uncertain linear programming problem with the constraint ($A \in \mathbf{R}^{m \times n}$)

$$A(x + w) \leq b \quad \forall |w|_\infty \leq 1 \tag{11}$$

This set of uncertain constraints can be robustified using the duality approach, the enumeration approach, and the explicit maximization. The explicit maximization approach will lead to the robustified constraint $Ax + |A|\mathbf{1} \leq b$, where $\mathbf{1}$ denotes a vector of ones and $|A|$ is element-wise absolute value. In other words, the size of the problem remains unaltered. The enumeration approach would lead to the constraints $A(x + v_i) \leq b$, where v_i denotes the 2^n vertices of the unit cube. Hence, the original set of m constraints are replicated 2^n times with different right-hand

sides, thus leading to a quickly growing problem size. Finally, the duality based approach leads to a model with $2nm + n$ variables, nm equality constraints, and $m + 2nm$ inequality constraints.[1]

We now make a small change in our model, and consider

$$A(x + w) \leq b \quad \forall |w|_1 \leq 1 \tag{12}$$

By using the explicit filter, the problem size remains unchanged, since every row in the constraint is robustified to $a_i^T x + |a_i|_\infty \leq b_i$. Using an enumeration of the uncertainty polytope leads to $2n$ vertices, hence the robust counterpart will have $2nm$ constraints in the n original decision variables. The duality approach depends crucially on how the uncertainty set has been modelled. The most naive approach is to use a polytope in the original w-variables. Such a polytope will have 2^n facets, hence a polytopic representation $Ew \leq f$ will have 2^n constraints. Using the same computations as above, we end up with $2^n m + n$ variables, nm equality constraints and $m + 2^n m$ inequality constraints. This derivation assumes that the user explicitly has generated the exponentially large polytope representation of the uncertainty set. A more realistic scenario is that the set is represented in a lifted space, $\{w, t : -t \leq w \leq t, \sum_i^n t_i \leq 1\}$. This is the representation YALMIP would work with if the user simply described the uncertainty as $|w|_1 \leq 1$. This polytope has $2n + 1$ vertices, hence the enumeration approach would yield roughly the same complexity. However, it only has $2n + 1$ facets, although in $2n$ variables. Applying the duality filter leads to a model with $(2n + 1)m + n$ variables, nm equality constraints and $m + (2n + 1)m$ inequalities.

From this simple comparison, it should be clear that the choice of filter can make a huge impact on the resulting problem. The current implementation always tries to apply the explicit maximization scheme first. If this filter not is applicable, the robustification resorts to the enumeration scheme if possible (based on the empirical knowledge that most users solve problems with few uncertain variables). The duality based scheme is only used if necessary, or if an explicit choice is made via an option structure that is available for the user to guide the conversion.

6. Robust convex programming can easily become intractable

In this section, we want to bring forward an often missed fact. Models that easily (polynomial time and size reformulation) can be converted to conic programs can become intractable if uncertainty is involved, even if the uncertainty by it self can be handled in polynomial time on a *given* standard form conic program. In other words, the combination of tractable convex modeling and tractable robust modeling can lead to intractable problems. Notice that we are not talking about the fact that some worst-case problems are intractable (e.g., uncertain semidefinite constraints with semidefinite representable uncertainty), but that unfortunate combinations of epigraph reformulations and uncertainty can lead to troubles.

Let us begin by considering the issue in a fairly general setting, minimizing the

[1]For every row a_i, we have to maximize $a_i^T w$ subject to $-1 \leq w \leq 1$. Since the dimension of w is n, there are $2n$ inequalities in the uncertainty constraint set. There will thus be $2n$ dual variables for each row. The dual variables are constrained by n equality constraints and $2n$ inequality constraints. Summing up and adding the original constraints and variables leads to the result.

worst-case sum of convex functions

$$\min_{x\in\mathcal{X}} \max_{w\in\mathcal{W}} \sum_{i=1}^{m} f_i(x,w) \tag{13}$$

Perhaps not too surprising, this problem is intractable in the general case. Let us now assume that all functions are (polynomial size) linear programming representable (i.e. convex and piecewise affine), the uncertainty set is polytopic and enters affinely, and that there are no constraints on x. We claim that the problem still is intractable. The result follows directly from the following theorem due to Mangasarian and Shiau,

THEOREM 6.1 [MS86] Maximizing $||w||_p$ over $Aw \leq b$ is NP-complete for $p \in [1, \infty)$

Clearly, the function $||w||_1$ is linear programming representable with a size polynomial in the dimension of w (requiring $2n+1$ constraints and n additional auxiliary variables, see Section 5.7). However, the theorem above immediately tells us that the problem $\min_x \max_{Aw\leq b} ||w||_1$ should be intractable (already in the case when x is absent). Hence, it would be highly surprising if we had an efficient approach to handle worst-case optimization of linear programming representable expressions, despite the fact that the uncertainty model it self is easily handled on a given linear program in standard form (in this case, using the polynomially sized duality filter). In the current version of YALMIP, these model scenarios are thus not supported, but the extension is under development.

7. Examples

To illustrate the use of the software package, we will solve a couple of uncertain optimization problems, starting with the general linear programming scenario, and the solve some applied problems from control.

7.1. *Robust linear programming*

As a first example, we implement the robust linear programming problem from Section 5.7, minimize $c^T x$ subject to $A(x+w) \leq b$ where $|w|_\infty \leq 1$.

```
x = sdpvar(n,1);
w = sdpvar(n,1);

C = [A*(x+w) <= b,
      abs(w) <= 1,
      uncertain(w)]

solvesdp(C,c'*x);
```

For a user familiar with YALMIP, the only new modeling construct is the command `uncertain` which declares a variable as uncertain. The actual uncertainty set will be extracted automatically from the model. After extracting the uncertainty model and analyzing the uncertain constraints, a suitable filter will be applied to derive the robust counterpart, which will be solved with a suitable solver, here any installed and recognized linear programming solver. In this case, the explicit maximization filter from Section 5.3 will be applied. Note that according to the discussion in Section 4, the uncertainty model will be expanded to the model $\{(w,t) : -t \leq w \leq$

$t,\ t \leq 1\}$. This uncertainty set only implicitly defines a simple box-bounded set. However, since this case is so common, and the use of the absolute value operator is frequent among users, YALMIP implements some specialized code to analyze the uncertainty set to detect the redundant variables t and project the problem to the standard box-bounded case.

Sim introduces the D-norm in [Sim04]. In its most simple form, this norm induces an uncertainty set which can can be interpreted as a standard ∞-norm ball with corners cut off by intersecting it with a 1-norm ball. How much of the corners that are cut of is controlled with a scalar $0 \leq r \leq n$.

```
C = [A*(x+w) <= b,
      max(norm(w,inf),norm(w,1)/r) <= 1,
      uncertain(w)]
```

This model, although it allows an explicit robust counterpart due to the self-dual properties of the D-norm [Sim04], will be handled using the duality filter since the YALMIP identifies the uncertainty set as nothing but a general polytopic set, and the result is a linear program.

The model in [Soy73] can easily be generated and solved. All we have to do is to add perturbations to the data matrix A and declare them as uncertain and bounded. Once again, the explicit maximization machinery will be invoked and a linear program will be solved.

```
dA = sdpvar(n,m);
C = [(A+dA)*x <= b,
       dAmin <= dA <= dAmax,
      uncertain(dA)]
```

Finally, we recover row-wise ellipsoidal uncertainty model from [BTN99].

```
dA = sdpvar(n,m);
C = [(A+dA)*x <= b,
      sum(dA.*dA,2) <= 1,
      uncertain(dA)]
```

This uncertainty model can be approached using both the explicit maximization filter and the duality filter (YALMIP automatically rewrites the quadratic constraints as second-order cone constraints). Due to the typically lower complexity of the models arising from explicit maximization, it is employed by default. The result is a second-order cone program.

7.2. *Robust optimal control with linear recourse*

As a second example, we address the problem of finite-horizon optimal control of uncertain constrained discrete-time linear systems. Consider the scalar system

$$x_{k+1} = x_k + u_k + w_k \tag{14}$$

The sought control input is constrained $|u_k| \leq 1$ while the disturbance is bounded $|w_k| \leq \frac{1}{5}$. The objective is to minimize the worst-case amplitude of the state over a finite prediction-horizon, i.e., $\max_w \max_i |x_{k+i}|$. However, we assume that the system will run in closed-loop in a receding horizon fashion, hence we should allow u_1 to feed back from future state measurements, i.e., to depend on w_0. One strategy to incorporate this is to parameterize the future input linearly in past disturbances [GW74, LÖ3], which in our setting means we write $u_1 = Lw_0 + v_1$, where L and

v_1 are decision variables. The model is straightforwardly implemented and solved below for the particular initial state $x_0 = 1$.

```
sdpvar u0 L v1 w0 w1
x0 = 1;

x1 = x0 + u0 + w0
u1 = L*w0+v1
x2 = x1 + u1 + w1

C = [uncertain([w0 w1]), -0.2 < [w0 w1] < 0.2, -1 < u0 < 1];

solvesdp(C,norm([x1;x2],inf));
```

The optimal solution turns out to be $u_0 = -1$, $L = -1$ and $v_1 = 0$. In other words, one aggressive step, and then feedback decides the whole input at the future instant. Note that predictions and control parameterizations are defined explicitly, they should not represented using equality constraints.

7.3. *LPV stabilization*

Our task is to compute a stabilizing state-feedback $u(t) = Kx(t)$ for the linear parameter-varying system $\dot{x}(t) = A(\rho)x(t) + Bu(t)$, where A is linearly parameterized in ρ, which is uncertain but known to be constant and constrained to a simplex. Without going into details, a controller that minimizes an upper bound on $\int_0^\infty x(t)^T Q x(t) + u(t)^T R u(t) dt$ can be found by solving the following uncertain semidefinite program in the variable L and the inverse positive semidefinite Lyapunov matrix Y [BEFB94].

$$\max_{L,Y} \max_{\rho} \; -\operatorname{trace}(Y)$$

$$\begin{bmatrix} -(A(\rho)Y + BL) - (A(\rho)Y + BL)^T & Y & L^T \\ Y & Q^{-1} & 0 \\ L & 0 & R^{-1} \end{bmatrix} \succeq 0$$

The feedback matrix can be recovered as $K = LY^{-1}$. This problem can be solved easily using our framework, since it fits into the enumeration scenario (linearly parameterized conic constraint with polytopic uncertainty). However, to make matters more challenging, we complicate the problem slightly. To decrease conservativity in the case when we actually can measure ρ on-line, we could be interested in using a parameterized feedback $L = \sum \rho_i L_i$, and a parameterized inverse Lyapunov function $Y = \sum \rho_i Y_i$. The problem with this parameterization is that the product $A(\rho)Y(\rho)$ yields bilinear terms. Hence, the enumeration scheme cannot be used. However, if the matrix $A(\rho)$ has a particular structure, some terms in Y can still be parameterized, without giving rise to any bilinear terms. This is where the elimination filter comes into play. By simply using a full parameterization, and letting YALMIP derive the robust counterpart, YALMIP will automatically constrain the structure of Y so that no bilinear terms are generated. After this elimination is done, the remaining uncertainty is dealt with using enumeration. A less conservative approach can be obtained by using the Pólya filter introduced in Section 5.4. This approach will allow nonlinear terms in the uncertain semidefinite constraint. The following code illustrates how we would solve a problem in the case when ρ is two-dimensional, for a system with n states and m inputs, using a Polya

filter with $N = 1$.

```
rho = sdpvar(2,1)
A = A0 + A1*rho(1) + A2*rho(2);

Y0 = sdpvar(n,n);Y1 = sdpvar(n,n);Y2 = sdpvar(n,n);
Y = Y0 + rho(1)*Y1 + rho(2)*Y2
L0 = sdpvar(n,m);L1 = sdpvar(n,m);L2 = sdpvar(n,m);
L = L0 + rho(1)*L1 + rho(2)*L2

S = -A*Y-B*L;
C = [[S+S'     Y        L';
      Y      inv(Q)    zeros(n,m);
      L     zeros(m,n) inv(R)] > 0]
C = [C, 0 <= rho, sum(rho) == 1]
C = [C, uncertain(rho)]
options = sdpsettings('robust.polya',1);
solvesdp(C,-trace(Y),options)
```

7.4. *Uncertain quadratic constraint on the unit-simplex*

To more explicitly illustrate the use of the Pólya filter, we study the impact of the relaxation degree N on a simple quadratic constraint. Consider the following set

$$\{(a,b) \mid ax^2 + 2bxy + by^2 \geq 0 \ \forall \ x + y = 1, x \geq 0, y \geq 0\} \tag{15}$$

Our goal here is to study the approximation that arise when we apply the Pólya filter on this constraint. The model is

```
sdpvar a b c x y
p = a*x^2+b*x*y+y^2*a
C = [p > 0, uncertain([x y]), x+y == 1, [x y] > 0];
```

To perform the study, we use the plot capabilities of YALMIP. Note that the set is unbounded, hence, for illustration purpose only, we add artificial bound constraints.

```
C = [C, a < 5, b < 5]
plot(C,[],[0.9 0.9 0.9],[],sdpsettings('robust.Polya',16))
plot(C,[],[0.8 0.8 0.8],[],sdpsettings('robust.Polya',4))
plot(C,[],[0.7 0.7 0.7],[],sdpsettings('robust.Polya',2))
plot(C,[],[0.6 0.6 0.6],[],sdpsettings('robust.Polya',0))
```

The resulting feasible sets are reported in Figure 1.

7.5. *Uncertain sum-of-squares*

As a final example, we solve a problem where we show-case the integration of different modules in YALMIP. Here, illustrated by combining the robust optimization module with the sum-of-squares capabilities of YALMIP, described in detail in [LÖ9].

A nonlinear system is described by the following differential equation.

$$\begin{aligned}\dot{x}_1 &= -\frac{3}{2}x_1^2 - \frac{1}{2}x_1^3 - x_2 \\ \dot{x}_2 &= 6x_1 - wx_2\end{aligned}$$

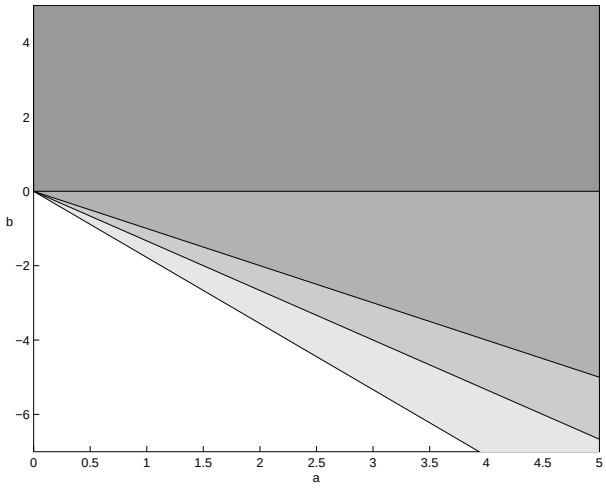

Figure 1. The feasible sets arising after applying Pólya filters of varying relaxation orders. On top (and darkest), the trivial set $(a, b) \geq 0$ generated when $N = 0$. For increasing degrees, the feasible set (partly covered by sets related to lower relaxation degrees) grows and incorporates portions of the lower right orthant.

The model is not known exactly, due to the uncertain parameter $w \in \mathcal{W} = \{w \mid 2 \leq w \leq 5\}$. Our goal is to show that the nonlinear system is stable for any $w \in \mathcal{W}$, and our approach to do this is to construct a polynomial Lyapunov function, and prove robust asymptotic stability using sum-of-squares techniques.

The uncertain variable w does not pose a problem for standard sum-of-squares techniques. Including information about uncertainty in the differential equation can be done relatively easily in a sum-of-squares framework by, e.g., suitable application of the positivstellensatz, [Par03]. However, we will apply a robust optimization approach instead.

To prove stability, we introduce a polynomial Lyapunov function $V(x) = c^T v(x)$ where c is the sought parameterization of the polynomial, and $v(x)$ is a predefined vector of monomials. For stability, we require

$$V(x) > 0 \ \forall x \neq 0$$
$$\dot{V}(x) < 0 \ \forall x \neq 0, \ w \in \mathcal{W}$$

A sum-of-squares approach tries to find symmetric matrices Q_1 and Q_2 such that $V(x) = h(x)^T Q_1 h(x)$ and $\dot{V}(x) = -h(x)^T Q_2 h(x)$, given a polynomial basis $h(x)$. By writing the sum-of-squares problem in image form [Par03], the semidefinite problem that arise will include two constraints, $Q_1(c) \succ 0$ and $Q_2(c, w) \succ 0$.

We begin by defining the variables and basic expressions involved in the problem. Without any deeper thought, we use a fourth order polynomial, and bound the Lyapunov function and its negative derivative from below by quadratic functions to ensure that the functions are positive definite[1].

```
sdpvar x1 x2 w
f = [-1.5*x1^2-0.5*x1^3-x2;
      6*x1-w*x2];
x = [x1;x2];
[V,c] = polynomial(x,4);
dVdt  = jacobian(V,x)*f;
```

[1]Methods to impose strict inequalities in a sum-of-squares setting is a delicate issue beyond the scope of this example.

```
r = x'*x;
```

The sum-of-squares constraints and the uncertainty model are defined and the problem is solved. If feasible, robust asymptotic stability is proved.

```
C = [uncertain(w), 2<=w<=5];
C = [C,sos(V-r),sos(-dVdt-r)];
solvesdp(C,[],[],c);
```

Behind the scenes, YALMIP will derive the matrices $Q_1(c)$ and $Q_2(c, w)$ using its standard sum-of-squares module. Since $Q_2(c, w)$ is linear in w and w is described by a polytope, the enumeration filter is applicable and is used to eliminate the uncertainty. The solution $V(x)$ is a quartic polynomial with 11 non-zero elements. By exploiting the built-in support for mixed-integer conic programming, we can alternatively search for a solution with minimal cardinality using the `nnz` operator. For the mixed-integer programming representation of the cardinality constraint to be numerically sound, we bound the decision variables.

```
solvesdp([C, -100< c < 100],nnz(c));
```

The solution returned is $V(x) = 89.22x_1^2 - 49.21x_1x_2 + 14.77x_2^2 + 0.12x_2^4$.

8. Conclusion

A software framework for robust optimization has been presented. The modeling language in YALMIP allows users to concentrate on the application model, while YALMIP takes care of reformulations required to remove uncertainty in the problem and compute robust solutions. The automatic support for robustification of uncertain models allows extremely rapid experimentation and prototyping with different uncertainty models and methods to handle these.

The implementation is currently limited to a small number of standard uncertainty cases, albeit they have to be considered the most common cases found in practice. Additional scenarios will hopefully be available in future versions.

In addition to more uncertainty scenarios, the framework will hopefully be extended to support a broader class of optimization problems, such as general convex problems, and uncertain geometric programs. The current version is primarily meant for problems with (mixed-integer) conic representable constraints.

References

[BEFB94] S. Boyd, L. El Ghaoui, E. Feron, and V Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics. SIAM, 1994.

[BTN98] A. Ben-Tal and A. Nemirovski. Robust Convex Optimization. *Mathematics of Operations Research*, 23(4):769 – 805, 1998.

[BTN99] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[BTN02] A. Ben-Tal and A. Nemirovski. Robust Optimization - Methodology and Applications. *Mathematical Programming (Series B)*, 92(3):453–480, 2002.

[BV04] S Boyd and L Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CC04] G. Calafiore and M. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2004.

[EOL98] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

[Fal76] J. E. Falk. Exact Solutions of Inexact Linear Programs. *Operations Research*, 24(4):783 – 787, 1976.

[GB08] M. Grant and S Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control (a tribute to M. Vidyasagar)*, volume 371 of *Lecture Notes in Control and Information Sciences*, pages 95–110. Springer, 2008.

[GW74] S. J. Garstka and R. J. B. Wets. On decision rules in stochastic programming. *Mathematical Programming*, 7(1):117–143, December 1974.

[HLP52] G. Hardy, J. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, second edition, 1952.

[LÖ3] J. Löfberg. Approximations of closed-loop MPC. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 1438–1442, Maui, Hawaii, 2003.

[LÖ4] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the 13th IEEE International Symposium on Computer Aided Control System Design*, Taipei, Taiwan, 2004.

[LÖ8] J. Löfberg. Modeling and solving uncertain optimization problems in YALMIP. In *Proceedings of the 17th IFAC World Congress on Automatic Control*, Seoul, South Korea, 2008.

[LÖ9] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, 2009.

[Las09] J.-B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Optimization Series. Imperial College Press, 2009.

[MS86] O. Mangasarian and T Shiau. A variable-complexity norm maximization problem. *SIAM Journal on Algebraic and Discrete Methods*, 7(3):455–461, 1986.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[Pre95] A. Prekopa. *Stochastic programming*. Mathematics and Its Applications. Kluwer Academic Publishers, 1995.

[Sch05] C. W. Scherer. Relaxations for Robust Linear Matrix Inequality Problems with Verifications for Exactness. *SIAM Journal on Matrix Analysis and Applications*, 27(2), 2005.

[SG10] M. Sim and J. Goh. Robust Optimization Made Easy with ROME, Working paper. 2010.

[Sim04] M. Sim. *Robust Optimization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[Soy73] A L Soyster. Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. *Operations Research*, 21:1154–1157, 1973.

[Soy74] A. L. Soyster. A Duality Theory for Convex Programming with Set-Inclusive Constraints. *Operations Research*, 22(4):892 – 898, 1974.

[ZDG95] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1995.