Cutting-Set Methods for Robust Convex Optimization with Pessimizing Oracles

Almir Mutapcic* Stephen Boyd*

Abstract

We consider a general worst-case robust convex optimization problem, with arbitrary dependence on the uncertain parameters, which are assumed to lie in some given set of possible values. We describe a general method for solving such a problem, which alternates between optimization and worst-case analysis. With exact worst-case analysis, the method is shown to converge to a robust optimal point. With approximate worst-case analysis, which is the best we can do in many practical cases, the method seems to work very well in practice, subject to the errors in our worst-case analysis. We give variations on the basic method that can give enhanced convergence, reduce data storage, or improve other algorithm properties. Numerical simulations suggest that the method finds a quite robust solution within a few tens of steps; using warm-start techniques in the optimization steps reduces the overall effort to a modest multiple of solving a nominal problem, ignoring the parameter variation. The method is illustrated with several application examples.

Key words. Robust optimization, cutting-set methods, semi-infinite programming, minimax optimization, games.

*The authors are with the Information Systems Lab, Electrical Engineering Dept., Stanford University, Stanford, CA 94305-9510 USA. Email: {almirm,boyd}@stanford.edu.

1 Introduction

We start with a basic convex optimization problem,

$$\begin{array}{ll} \text{minimize} & f_0(x,u) \\ \text{subject to} & f_i(x,u) \leq 0, \quad i=1,\ldots,m, \end{array} \tag{1}$$

where $x \in \mathbf{R}^n$ is the variable, $f_0, \ldots, f_m$ are convex in x, and $u \in \mathbf{R}^p$ is a vector of problem parameters. The parameter u is used to model uncertainty or variation in problem data values. This uncertainty can come from many sources, *e.g.*, estimation errors, or variation arising in implementation, manufacture, or operation of a system. We are interested in the case when this problem can be efficiently solved, for any fixed value of the parameter u. For example, if f_i are affine in x, then for each u, the problem (1) is a linear program, and therefore easily solved.

Nominal problem. We assume that a nominal value of the parameter u, which we denote $u_{\rm nom}$, is known. The *nominal problem* is

$$\begin{array}{ll} \text{minimize} & f_0(x,u_{\rm nom}) \\ \text{subject to} & f_i(x,u_{\rm nom}) \leq 0, \quad i=1,\ldots,m. \end{array} \tag{2}$$

We let $p^\star_{\rm nom}$ denote its optimal value, and $\mathcal{F}_{\rm nom}$ its feasible set.

Parameter variation and worst-case analysis. We model uncertainty in the parameter u as $u \in \mathcal{U}$, where $\mathcal{U} \subset \mathbf{R}^p$ is a (known) set of possible parameter values, with $u_{\rm nom} \in \mathcal{U}$. We will measure the affect of the uncertainty on a particular choice of the variable x using a worst-case approach, *i.e.*, by the largest value of the objective and constraint functions, over $u \in \mathcal{U}$. We define

$$F_i(x) = \sup_{u \in \mathcal{U}} f_i(x,u), \quad i=0,\ldots,m, \tag{3}$$

and

$$V(x) = \max_{i=1,\ldots,m} F_i(x). \tag{4}$$

We refer to F_0 as the *worst-case objective function*, the functions $F_1, \ldots, F_m$ as the *worst-case constraint functions*, and V as the *maximum constraint violation*. These functions are convex, since each is a supremum of a family of convex functions (indexed by u). Evaluating $F_0(x), \ldots, F_m(x)$ for a given x, *i.e.*, maximizing $f_i(x,u)$ over $u \in \mathcal{U}$ is called *worst-case analysis*. We will also use the term *pessimization*, suggested by Steven G. Johnson, since we are finding the most pessimistic value of the parameter for the objective and each constraint, for a given x. Worst-case analysis can be very difficult, depending on how f_i depends on u, and the particular set $\mathcal{U}$.

A common approach to dealing with parameter uncertainty is to ignore it, and simply solve the nominal problem to obtain a nominal optimal point $x^\star_{\rm nom}$. After this optimization step, a ("posterior") worst-case analysis is carried out on the nominal optimal point $x^\star_{\rm nom}$. If

the worst-case cost $F_0(x^\star_{\rm nom})$ is not much bigger than the nominal cost $p^\star_{\rm nom} = f_0(x^\star_{\rm nom}, u_{\rm nom})$, and if the worst-case constraint violation $V(x^\star_{\rm nom})$ is not too large, the nominal optimal point $x^\star_{\rm nom}$ is judged to be acceptable.

Worst-case robust optimization. In *robust optimization*, the parameter uncertainty is taken into account during the optimization phase. The *robust convex optimization problem* is

$$\begin{array}{ll} \text{minimize} & F_0(x) \\ \text{subject to} & F_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{5}$$

with variable $x \in \mathbf{R}^n$. For brevity, we will refer to (5) as the *robust problem*. We let $p^\star_{\rm rob}$ denote its optimal value, and $\mathcal{F}_{\rm rob}$ its feasible set. Evidently we have $\mathcal{F}_{\rm rob} \subset \mathcal{F}_{\rm nom}$, so $p^\star_{\rm rob} \geq p^\star_{\rm nom}$.

We mention that there are several other methods for taking data and parameter variation into account in an optimization problem. In stochastic optimization, for example, u is modeled as a random variable, and we use the expected value of (some function of) the objective and constraints to form a problem that takes parameter variation into account. In this paper, however, we focus solely on the worst-case robust problem (5).

Since F_i are convex, the robust problem (5) is a convex optimization problem. However, even if the basic problem (1) is readily solved, the robust problem (5) can be difficult to solve. Indeed, just evaluating its objective and constraint function can be hard.

Over the last ten or so years, researchers have shown how to efficiently solve a number of specific cases of the robust problem, by reducing it to a tractable convex optimization problem. One famous example is the robust linear programming problem with ellipsoidal uncertainty. In this problem each f_i is bi-affine in x and u (*i.e.*, affine in each, with the other fixed), and $\mathcal{U}$ is an ellipsoid. This problem can be reduced to a second-order cone program (SOCP), and readily solved (see, *e.g.*, [1, pg. 157], [2], [3, §2.6]). On the other hand, many other robust problems have been shown to be hard to solve exactly; see, *e.g.*, [4, 5]. (These results will be reviewed in more detail in §2.)

Purpose of this paper. In this paper we describe a simple scheme for (possibly approximately) solving the robust problem. The method consists of an alternating sequence of optimization and (possibly approximate) worst-case analysis (pessimization) steps. When the pessimization is exact, *i.e.*, we compute the worst-case functions F_i exactly, the scheme can be shown to solve the robust problem. When the pessimization is approximate, *i.e.*, the worst-case functions are computed approximately, we cannot say that the method solves the robust problem; indeed, we cannot even (surely) determine if a point x is feasible for the robust problem. In this case we can only say that, if the pessimization were exact, we would have solved the robust problem. In a practical sense, this is all we can hope for, when we cannot carry out the pessimization exactly.

Numerical simulations suggest that the method performs well in practice, obtaining good approximate solutions of the robust problem in a small number (typically 10 or so) of optimization-pessimization steps. Using warm-start techniques to solve each optimization

problem starting from the solution of the previous one typically reduces the optimization effort by a significant factor, such as 5 or 10; this, in turn, means that the overall cost of approximately solving the robust problem is a small multiple, such as 3 or 4, of the cost of solving the nominal problem, in which parameter variation is ignored. (We do not include here the cost of the pessimization, which in any case needs to be carried out to certify the final approximate solution.) Thus our message to practitioners is: *The cost of carrying out practical robust optimization is not much more than the cost of carrying out nominal optimization, followed by worst-case analysis.*

The ideas behind the method we describe are not new. Indeed, the method can be related to existing algorithms in semi-infinite programming, cutting-plane methods, active-set methods, and so on. (These connections will be detailed in §2.) Our contribution is to present a simple, unified framework for these "sampling" methods, as applied to practical worst-case robust optimization.

Epigraph form. It will be convenient to work with a simplified, but general, standard form for the robust problem. We introduce a new variable $t \in \mathbf{R}$, and form the epigraph form problem (see, *e.g.*, [1, p.134])

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \sup_{u\in\mathcal{U}} \left(f_0(x,u) - t\right) \leq 0, \\ & F_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

This problem is equivalent to the original robust problem (5). It also has the form of a general robust convex problem, with $n+1$ variables and $m+1$ constraints; unlike our original robust problem, however, it has a linear objective that does not depend on the parameter u. Thus, without loss of generality, we can assume that the objective in the robust problem is linear and certain.

In the sequel, then, we will consider the robust problem in the standard form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array} \tag{6}$$

Outline. In §2 we give an overview of previous and related work. In §3 we introduce the sampled problem, and give some bounds on the optimal value of the robust problem based on the solution of the sampled problem. In §4 we consider the pessimization process in more detail, defining the notion of approximate and exact pessimizing oracles, and giving examples of each. In §5 we give the basic cutting-set method for solving the robust problem, prove its convergence, and describe some variations; we finish with a numerical example for which the exact solution can be computed. In §7 we describe a robust antenna beamforming example in more detail, using an exact oracle that, as far as we know, is new. In §8 we describe a robust state transfer problem. In this problem we do not know an exact oracle, so we use a good approximate oracle; our final solution is then carefully checked for robustness. We give conclusions in §9.

2 Related formulations and previous work

2.1 Regularization and stochastic programming

There are several general approaches to dealing with parameter uncertainty in optimization problems. These can be broadly classified into three groups (which are closely related): regularization, stochastic optimization, and worst-case robust optimization. In *regularization*, a method popularized by Tikhonov [6], we are a bit vague about what the variations are, and simply add an extra cost term to our objective function (or the constraint functions) that penalizes sensitive or nonrobust designs. A more sophisticated regularization method that adds sensitivity penalties based on derivative information is given in [7].

In *stochastic optimization*, we have a stochastic or probabilistic model for parameter variation, and choose x that minimizes (say) the expected objective value, subject to the constraints holding (say) with some probability; see, *e.g.*, the books [8–10] and tutorial [11]. Some methods for solving these problems approximately are closely related to our method; for example, probabilistic sampling or scenario approaches (see, *e.g.*, [12–14]).

2.2 Worst-case robust optimization

The third general approach, which we take in this paper, is *worst-case robust optimization*. Here we model the uncertain parameters as lying in some given set of possible values, but without any known distribution, and we choose a design whose worst-case objective value, over the given set of possible uncertainties, is minimized. The forthcoming book [15] provides a comprehensive treatment of (worst-case) robust optimization.

The first systematic methods for solving classes of convex optimization problems with uncertain parameters were introduced in 1950s; for example, Dantzig considered solutions of uncertain linear programs (LPs) in [16]. In 1973, Soyster considered convex programming problems with set-inclusive constraints [17], in which the worst-case approach is used to solve LPs with box set uncertainty. There has been much recent work on worst-case robust optimization for specific convex optimization problems and associated parameter uncertainty sets. In [4, 18], Ben-Tal and Nemirovski formulated specific classes of robust optimization problems as other convex optimization problems with no uncertainty, which can now be efficiently solved using, for example, interior-point methods [1, Ch. 11], [19–21]. These were followed by many results for specific problem classes or applications; see, *e.g.*, the survey [22]; examples include robust linear programs [2, 23, 24], robust least-squares [25, 26], robust quadratically constrained programs [27], robust semidefinite programs [28], robust conic programming [29], robust discrete optimization [30]. Work focused on specific applications includes robust control [31, 32], robust portfolio optimization [33–36], robust beamforming [37–39], robust machine learning [40], and many others.

For negative results, which show that some robust optimization problems are (for example) NP-hard, see, *e.g.*, [4]. One approach in these cases is to formulate a conservative approximation of the robust problem, using known upper bounds on the worst-case functions F_i, which can be handled exactly.

Another approach, halfway between a stochastic and worst-case robust approach, assumes a probability distribution on the uncertainty set, and samples the constraints. In this case one can prove bounds on the probability of constraint violation; see, *e.g.*, [14, 41].

2.3 Related problems

Semi-infinite programming. We can write the robust problem (6) as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & f_i(x,u) \leq 0 \text{ for all } u \in \mathcal{U}, \quad i = 1, \ldots, m. \end{array} \tag{7}$$

This is a so-called convex *semi-infinite programming* (SIP) problem, since we have an infinite number of constraints, parametrized by i and (the infinite index) u. There is an extensive literature on the theory and numerical methods for the solution of SIP problems; see, *e.g.*, the survey [42] or books [43–45]. Numerical methods for SIP problems can be broadly categorized into three groups (which are themselves related): discretization, exchange, and local reduction methods.

Discretization and exchange methods are of particular interest to us, since our proposed methods can be seen as variants of these methods when applied to solving the robust problem. Discretization and exchange methods are iterative methods that discretize the infinite set of constraints and solve the resulting discretized problem. At each iteration, new constraints are determined by solving an auxiliary problem, and the updated finite problem is solved. This is repeated until some stopping criterion is satisfied.

The basic exchange methods applied to linear SIPs can be related to Kelley's cutting-plane methods [46], and thus can exhibit well-known drawbacks such as slow convergence, infeasible intermediate points, and others [47]. More advanced, rate-preserving discretization methods have been proposed by Polak and He in [48].

The SIP (7) can be viewed as a large-scale optimization problem, so we can apply active-set or column-generation methods; see, *e.g.*, [49, Ch. 4]. Our proposed methods are related to active-set and Bender's decomposition methods [50] as applied to robust problems; see [51] for more details on such interpretations and some experimental results in robust portfolio optimization.

Nonsmooth minimax optimization. The robust problem (5) can be interpreted as a *nonsmooth* minimax optimization problem (*e.g.*, see book [52]), since the objective F_0 and the constraint functions F_i are convex, but generally nonsmooth functions. There are many numerical methods available to solve the nonsmooth optimization problems [53]; for example, subgradient methods [54, 55], feasible-direction methods [56], ellipsoid method [57], analytic center cutting-plane method (ACCPM) [58, 59], and many others.

Most of these methods require evaluation of $F_i(x)$ and a subgradient $g_i \in \partial F_i(x)$ at the current point x. When we can evaluate F_i and g_i in a reasonable time and with reasonable effort, these methods will work, but can be rather slow. Another implementable solution method is the subgradient sampling method introduced in [60, 61].

Game theory. The robust problem (6) can also be interpreted as a zero-sum game between two players, an optimizer who chooses design variables $\tilde{x}$, and an opponent who chooses a worst-case uncertainty $\tilde{u}$ from the uncertainty set $\mathcal{U}$. The goal of the optimizer is to choose the values of x so thatF_i below zero and $c^T x$ is small, while the opponent's goal is the opposite. In this setup we are looking for a saddle-point; see *e.g.*, [4, §4].

We remark that when the functions $f_i(x, u)$ are concave in u for all x, and the uncertainty set $\mathcal{U}$ is convex, we can cast (6) as a convex-concave game, which can be efficiently solved using interior-point methods, as described in, *e.g.*, [1, §10.3.4]. In [62], the authors treat SIP problems (and hence robust problems) as special (Stackelberg) games, and then apply interior-point methods developed for solving Stackelberg games to solve SIP problems. (Hence this approach can be extended to solve robust problems.)

3 Sampled problem

Let

$$\hat{\mathcal{U}}_i = \{u_{i,1}, \ldots, u_{i,K_i}\} \subset \mathcal{U}, \quad i = 1, \ldots, m,$$

be a collection of finite subsets of $\mathcal{U}$, where each subset has K_i samples or scenarios. We will assume that $u_{i,1} = u_{\text{nom}}$, so each of the subsets contains u_{nom}. The associated *sampled worst-case constraint functions* are defined as

$$\hat{F}_i(x) = \max\{f_i(x, u_{i,1}), \ldots, f_i(x, u_{i,K_i})\}, \quad i = 1, \ldots, m.$$

These functions satisfy

$$f_i(x, u_{\text{nom}}) \leq \hat{F}_i(x) \leq F_i(x)$$

for any x. The associated *sampled robust problem* is

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \hat{F}_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array} \tag{8}$$

This sampled problem can be expressed as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & f_i(x, u_{i,j}) \leq 0, \quad j = 1, \ldots, K_i, \quad i = 1, \ldots, m. \end{array} \tag{9}$$

This is a basic convex optimization problem with $K = K_1 + \cdots + K_m$ constraints, and can be solved with an effort that grows linearly with K. Let $p^\star_{\text{samp}}$ denote its optimal value, and $x^\star_{\text{samp}}$ an optimal point. We denote the feasible set of the sampled problem as $\mathcal{F}_{\text{samp}}$.

Lower bound on robust optimal value. We have

$$\mathcal{F}_{\text{rob}} \subset \mathcal{F}_{\text{samp}} \subset \mathcal{F}_{\text{nom}},$$

so

$$p^\star_{\text{rob}} \geq p^\star_{\text{samp}} \geq p^\star_{\text{nom}}.$$

In particular, the optimal value of the sampled problem is a lower bound on the optimal value of the robust problem. It follows that if $V(x^\star_{\text{samp}}) \leq 0$, *i.e.*, $x^\star_{\text{samp}}$ is feasible for the robust problem, then $x^\star_{\text{samp}}$ is optimal for the robust problem.

Upper bound on robust optimal value. We now assume that Slater's condition holds for the robust problem: There exists a point $x_{\rm sla}$ which satisfies $V(x_{\rm sla}) < 0$. In other words, there exists a strictly robustly feasible point. We can use the Slater point $x_{\rm sla}$ to construct a robustly feasible point from $x^\star_{\rm samp}$, with an objective value that can be bounded above. This will give us an upper bound on the optimal value of the robust problem.

Assume $V(x^\star_{\rm samp}) > 0$ (since otherwise, $x^\star_{\rm samp}$ is already optimal for the robust problem). Consider the point

$$\hat{x} = \theta x_{\rm sla} + (1-\theta)x^\star_{\rm samp},$$

where

$$\theta = \frac{V(x^\star_{\rm samp})}{V(x^\star_{\rm samp}) - V(x_{\rm sla})}.$$

Since $\theta \in (0,1)$ and V is convex, we have

$$V(\hat{x}) \leq \theta V(x_{\rm sla}) + (1-\theta)V(x^\star_{\rm samp}) = 0,$$

i.e., $\hat{x}$ is robustly feasible. Its objective value is

$$c^T\hat{x} = \theta c^T x_{\rm sla} + (1-\theta)c^T x^\star_{\rm samp} = p^\star_{\rm samp} + \frac{V(x^\star_{\rm samp})}{V(x^\star_{\rm samp}) - V(x_{\rm sla})} c^T(x_{\rm sla} - x^\star_{\rm samp}).$$

Now we use

$$\frac{c^T(x_{\rm sla} - x^\star_{\rm samp})}{V(x^\star_{\rm samp}) - V(x_{\rm sla})} \leq \beta = \frac{c^T x_{\rm sla} - p^\star_{\rm nom}}{-V(x_{\rm sla})}$$

to get

$$c^T\hat{x} \leq p^\star_{\rm samp} + \beta V(x^\star_{\rm samp}) \leq p^\star_{\rm rob} + \beta V(x^\star_{\rm samp}).$$

Thus, the point $\hat{x}$, which is robustly feasible, is at most $\beta V(x^\star_{\rm samp})$ suboptimal for the robust problem. (Note that the constant β is readily evaluated.) We can also conclude

$$p^\star_{\rm samp} \leq p^\star_{\rm rob} \leq p^\star_{\rm samp} + \beta V(x^\star_{\rm samp}). \tag{10}$$

4 Exact and approximate pessimizing oracles

A *pessimizing oracle* for constraint i is a subroutine or method that evaluates $F_i(x)$, at a given point x, possibly approximately. In other words, it solves the optimization problem

$$\begin{array}{ll} \text{maximize} & f_i(x,u) \\ \text{subject to} & u \in \mathcal{U}, \end{array} \tag{11}$$

with variable u and for a fixed x, possibly approximately. If the oracle solves the problem (11) exactly, we call it an *exact pessimization oracle*; if it solves it approximately, we call it an *approximate pessimization oracle*. By solving the problem (11) approximately, we mean that the method finds a value of u that is in $\mathcal{U}$, but is not guaranteed to maximize $f_i(x,u)$. For an exact pessimization oracle, we let $u^\star(x)$ denote a worst-case parameter, *i.e.*, a solution of (11).

4.1 Some exact oracles

Exact pessimizing oracles rely on analytic formulas for $F(x)$ (and $u^\star(x)$), or some tractable method for solving (11) exactly. In this section we list some typical exact oracles. For brevity, we drop the subscript i from F_i in the reminder of this section.

Finite set of modest size. When $\mathcal{U} = \{u_1, \ldots, u_K\}$ is a finite set of modest size, we evaluate $f(x, u_1), \ldots, f(x, u_K)$, and find an index k for which $f(x, u_k)$ is maximum. Then we take $u^\star(x) = u_k$ and $F(x) = f(x, u_k)$.

Convex hull of finite set. Suppose f is convex in u, for each x, and $\mathcal{U} = \mathbf{conv}\{u_1, \ldots, u_K\}$ is a polyhedron defined as the convex hull of a finite set of modest size. Then the maximum of $f(x, u)$ over $u \in \mathcal{U}$ is the same as the maximum over the given vertices, which we can evaluate by evaluating $f(x, u_1), \ldots, f(x, u_K)$.

Monotone function on a box. Suppose $f(x, u)$ is monotone in each component of u, for each x, and $\mathcal{U}$ is the box

$$\mathcal{U} = \{u \mid |u_i - u_{\mathrm{nom},i}| \leq \rho_i,\ i = 1, \ldots, p\}, \tag{12}$$

where ρ_i gives the radius or half-range of the variation in parameter i. (This type of parameter variation can also be described as $u_i = u_{\mathrm{nom},i} \pm \rho_i$.) Then we have

$$u^\star(x)_i = \begin{cases} u_{\mathrm{nom},i} + \rho_i, & f(x, u) \text{ nondecreasing in } u_i \\ u_{\mathrm{nom},i} - \rho_i, & f(x, u) \text{ nonincreasing in } u_i. \end{cases}$$

Affine dependence over ellipsoid. Suppose that f is affine in u, *i.e.*, $f(x, u) = a(x)^T u + b(x)$, and $\mathcal{U}$ is the ellipsoid $\mathcal{U} = \{u_{\mathrm{nom}} + Pz \mid \|z\|_2 \leq 1\}$. Then

$$F(x) = \sup_{\|z\|_2 \leq 1} a(x)^T (u_{\mathrm{nom}} + Pz) + b(x) = a(x)^T u_{\mathrm{nom}} + b(x) + \|P^T a(x)\|_2,$$

and

$$u^\star(x) = u_{\mathrm{nom}} + PP^T a(x)/\|P^T a(x)\|_2.$$

Affine dependence over polyhedron. Suppose $f(x, u) = a(x)^T u + b(x)$, with $\mathcal{U}$ the polyhedron $\mathcal{P} = \{z \mid Cz \preceq d,\ Gz = h\}$. Then $F(x)$ and $u^\star(x)$ can be found by solving the linear program

$$\begin{array}{ll} \text{maximize} & a(x)^T u + b(x) \\ \text{subject to} & Cu \preceq d, \quad Gu = h. \end{array}$$

(In this case we can also directly express $F(x)$ as the optimal value of the dual linear program, assuming a constraint qualification holds; in some cases, this allows us to solve the robust problem as one large optimization problem.)

Quadratic dependence over quadratic set. Suppose f is quadratic in u, for each x, *i.e.*,

$$f(x,u) = u^T P(x)u + q(x)^T u + r(x),$$

and $\mathcal{U}$ is defined by a quadratic function,

$$\mathcal{U} = \{u \mid u^T \tilde{P}(x)u + \tilde{q}(x)^T u + \tilde{r}(x) \leq 0\}.$$

We can compute $F(x)$ and $u^\star(x)$ exactly using the so-called S-procedure (see, *e.g.*, [1, App. B], [63]).

4.2 Some approximate oracles

Any optimization method can be used as an approximate pessimining oracle, including traditional gradient and the second-order methods [64], local search methods such as Nelder-Mead [65, 66], randomized search [67, 68], derivative-free optimization (DFO) [69], global search methods such as simulated annealing [70], evaluating $f(x,u)$ over a grid on $\mathcal{U}$ [71, §1.1.3], and so on.

An approximate oracle can have an *effort parameter*, which controls the effort the oracle will expend in approximating $F(x)$. For example, the effort parameter could be the number of times a local optimization method is run, from randomly chosen starting points (with the largest value found as our estimate of $F(x)$).

An example of a simple approximate oracle, when $\mathcal{U}$ is a box (12), is given by the choice

$$u^\star(x)_i \approx \begin{cases} u_{\mathrm{nom},i} + \rho_i, & \partial f(x,u)/\partial u_i \geq 0 \\ u_{\mathrm{nom},i} - \rho_i, & \partial f(x,u)/\partial u_i < 0. \end{cases}$$

(This is the maximizer of the first-order approximation of $f(x,u)$ over $\mathcal{U}$.) This point can, of course, be used as the starting point for a local optimization method.

5 Cutting-set methods

5.1 Basic cutting-set method

The algorithm is based on solving a sequence of sampled problems (9), with expanding sets of scenarios $\hat{\mathcal{U}}_i$, $i = 1, \ldots, m$, which are found by pessimization (*i.e.*, worst-case analysis).

Basic Cutting-set Method (CSM).

given stopping tolerance $V^{\mathrm{tol}} > 0$ and $\hat{\mathcal{U}}_i = \{u_{\mathrm{nom}}\}$, $i = 1, \ldots, m$.

repeat

1. *Optimization.*
   Solve sampled problem (9) with $\hat{\mathcal{U}}_i$, $i = 1, \ldots, m$, and return a solution $\tilde{x}$.
2. *Pessimization.*

for $i = 1, \ldots, m$

2a. Call oracle i to evaluate $u_i^\star(\tilde{x})$ and $F_i(\tilde{x})$ (possibly approximately).

2b. If $F_i(\tilde{x}) > 0$, append $u_i^\star(\tilde{x})$ to $\hat{\mathcal{U}}_i$.

3. *Sign-off criterion.* **quit** if $V(\tilde{x}) \leq V^{\mathrm{tol}}$.

This basic CSM alternates between optimization and pessimization steps, until a sign-off criterion based on the maximum constraint violation V is satisfied. In the optimization step, we solve the sampled problem (9) with the current set of scenarios $\hat{\mathcal{U}}_i$ for each constraint. In the pessimization step, we query oracles for each constraint to determine $F_i(\tilde{x})$, its maximum violation, exactly or approximately. (The oracles for the different constraints can be queried independently and in parallel, allowing a linear speedup of step 2; see, *e.g.*, [72].) Whenever pessimization of a constraint finds a value of u that violates a robust constraint, this value is appended to the list of scenarios for that constraint.

When the oracles are exact, we terminate with the certificate of suboptimality given in (10). When the pessimization oracles are approximate, our estimates of $F_i(x)$ and $V(x)$ are underestimates, which of course can lead to premature termination. If the oracles have an effort parameter, the sign-off criterion can be checked with maximum effort, to minimize the chance of premature termination.

We can explain the name of the method. When we add a new point $\tilde{u}$ to one of the sets $\hat{\mathcal{U}}_i$, we impose the additional constraint $f_i(x, \tilde{u}) \leq 0$ in the next optimization step. Our current point $\tilde{x}$ violates this inequality, *i.e.*, $f_i(\tilde{x}, \tilde{u}) > 0$; for every point $z \in \mathcal{F}_{\mathrm{rob}}$, however, we have $f_i(z, \tilde{u}) \leq 0$. Thus, adding the parameter value $\tilde{u}$ has 'cut' points from the set $\mathcal{F}_{\mathrm{samp}}^{(k)}$. When $f_i(x, \tilde{u})$ is affine in x, the cut corresponds to a hyperplane, or linear cut; otherwise, the cut is called nonlinear. In this case, the basic CSM is Kelley's cutting-plane method. The CSM could also be called a nonlinear cutting-plane method, or a cutting-surfaces method; see, *e.g.*, [73].

5.2 Convergence proof

For completeness, we give a convergence proof for the basic CSM. The proof is based on standard ideas and results used in the original proof of Kelley's cutting-plane method in [46], and in alternative proofs [74, §13.7], [75, §14.3.3], [76, §5.4], [71, §3.3].

Assumptions. We assume that the nominal feasible set $\mathcal{F}_{\mathrm{nom}}$ is bounded, and the constraint functions f_i are uniformly Lipschitz continuous in x, on $\mathcal{F}_{\mathrm{nom}}$, *i.e.*, there is a G for which

$$|f_i(x_1, u) - f_i(x_2, u)| \leq G\|x_1 - x_2\|, \tag{13}$$

holds for $i = 1, \ldots, m$, all $x_1, x_2 \in \mathcal{F}_{\mathrm{nom}}$, and all $u \in \mathcal{U}$. Of course we assume that the pessimization oracles are exact.

The assumption that the functions f_i are convex in x for each $u \in \mathcal{U}$ is not needed to prove convergence of CSM; this assumption is needed to ensure that the optimization step is tractable.

Proof. Let $\mathcal{F}_{\rm samp}^{(k)}$ be the feasible set, and $x^{(k)}$ the solution found, for the sampled problem (9) in Step 1, in the kth iteration of the algorithm.

Suppose that for $k = 1, \ldots, K$ the CSM has not terminated, *i.e.*, $V(x^{(k)}) > V^{\rm tol}$. We will derive an upper bound on how large K can be.

Let $u^{(k)}$ be a parameter value added to $\hat{\mathcal{U}}_{i^{(k)}}$ in Step 2, with

$$f_{i^{(k)}}(x^{(k)}, u^{(k)}) = V(x^{(k)}) > V^{\rm tol}. \tag{14}$$

Any $z \in \mathcal{F}_{\rm samp}^{(j)}$, with $j > k$, must satisfy $f_{i^{(k)}}(z, u^{(k)}) \leq 0$, since this constraint is part of the jth sampled problem. In particular, we must have

$$f_{i^{(k)}}(x^{(j)}, u^{(k)}) \leq 0, \tag{15}$$

for $j > k$. Combining (14) and (15) we see that, for $k < j$, we have

$$f_{i^{(k)}}(x^{(k)}, u^{(k)}) - f_{i^{(k)}}(x^{(j)}, u^{(k)}) > V^{\rm tol}.$$

Using the Lipschitz condition (13), we conclude

$$\|x^{(k)} - x^{(j)}\| > V^{\rm tol}/G, \tag{16}$$

for $k < j$. Thus, the minimum distance between any two points $x^{(1)}, \ldots, x^{(K)}$ exceeds $V^{\rm tol}/G$.

Let B_k denote the ball of diameter $V^{\rm tol}/G$ centered at $x^{(k)}$. By (16), these balls do not intersect, so their total volume is K times the volume of one ball, $K\beta_n(V^{\rm tol}/G)^n$, where β_n is the volume of the unit ball in $\mathbf{R}^n$.

Let B be a ball, with radius R, that contains $\mathcal{F}_{\rm nom}$. Then the balls $B_1, \ldots, B_K$ are contained in the ball $\tilde{B}$, which is B, with radius increased by $V^{\rm tol}/G$. If follows that the total volume of the balls $B_1, \ldots, B_K$ cannot exceed the volume of $\tilde{B}$, which is $\beta_n(R+V^{\rm tol}/G)^n$. Thus we have

$$K\beta_n(V^{\rm tol}/G)^n \leq \beta_n(R + V^{\rm tol}/G)^n,$$

from which we conclude

$$K \leq \left(\frac{RG}{V^{\rm tol}} + 1\right)^n.$$

The righthand side gives an upper bound on the number of iterations before the CSM terminates. (We mention that the actual number of iterations typically required is vastly smaller than this upper bound.)

5.3 Variations

5.3.1 Adding constraints

In the basic CSM, we add parameter values for any violated robust constraint. From the covergence proof we see that it suffices to add only one parameter value, corresponding to the worst-case robust constraint violation. Beyond this, we can add any number of parameters, including, at the other extreme, the worst parameter found for each constraint, whether violated or not. In between these two extremes, we can add N worst-case parameters corresponding to the N most violated constraints, where N is algorithm parameter.

5.3.2 Dropping constraints

We can also *drop* constraints, keeping, for example, only a total of N parameter values in the sampled problem, corresponding to the N most violated constraints, where a typical choice of N is between $3n$ and $5n$. The convergence proof presented above does not handle constraint dropping, but it can be extended to handle it; see, *e.g.*, [77].

5.3.3 Linearizing constraints

The next variation involves approximating constraints in order to simplify the solution of the sampled problem. We replace the constraint $f_i(x,u) \leq 0$ with the linear inequality

$$f_i(x^{(k)},u) + g^T(x - x^{(k)}) \leq 0,$$

where $g \in \partial_x f_i(x^{(k)},u)$ is any subgradient of f_i, with respect to x, at the point $x^{(k)}$. Here we still have an outer approximation of the robust problem, and thus a lower bound on $p^\star_{\rm rob}$. (Our convergence proof carries through almost unchanged.) If this is done for all constraints, the sampled problem becomes a linear program.

We do not recommend using the linear approximation of the constraints in cases when the constraint itself can be efficiently handled, since linearization typically yields slower convergence. However, linearization can work well when combined with analytic-center cutting-plane methods (ACCPM) [58, 59], discussed below. Linearization has been used for several types of SIP problems, *e.g.*, a cutting-plane method based on a most violated constraint [78], the Elzinga-Moore central cutting-plane method for solving linear SIPs [79], and convex SIP problems [80].

5.3.4 Regularized methods

The next variation is based on solving a regularized version of the sampled problem (9) at each iteration, such as

$$\begin{array}{ll} \text{minimize} & c^Tx + \gamma\|x - x^{(k)}\|_2^2 \\ \text{subject to} & f_i(x,u_{i,j}) \leq 0, \quad j = 1,\ldots,K_i, \quad i = 1,\ldots,m, \end{array} \tag{17}$$

where $x^{(k+1)}$ denotes an optimal solution of (17), $x^{(k)}$ is the solution of the previous sampled problem, and $\gamma > 0$ is the regularization parameter. Algorithms that use regularization based on quadratic or generalized distance functions are called the *proximal* minimization algorithms; see, *e.g.*, [81, §3.4.3], [72, Ch. 3]. Quadratic regularization (as in (17)) is called the Moreau-Yosida regularization, and the overall scheme in which we regularize a sequence of problem 'approximations' is (one variation on) the bundle method [82, Ch. 7]. Regularization can greatly reduce the number of iterations required. The algorithm converges for any value of the regularization parameter; its choice, however, can affect the rate of the convergence.

Proximal minimization methods have been applied to convex SIP, see, *e.g.*, [83], in which the authors give a convergence proof that can be applied here without much change. For completeness, we give a version of the proof that follows our notation.

We will work with distances to an optimal point $x^\star_{\rm rob}$ of the robust problem. We start with the identity

$$\|x^{(k)} - x^\star_{\rm rob}\|^2 = \|x^{(k)} - x^{(k+1)}\|^2 + 2(x^{(k)} - x^{(k+1)})^T(x^{(k+1)} - x^\star_{\rm rob}) + \|x^{(k+1)} - x^\star_{\rm rob}\|^2,$$

and we have

$$\|x^{(k)} - x^\star_{\rm rob}\|^2 + 2(x^{(k+1)} - x^{(k)})^T(x^{(k+1)} - x^\star_{\rm rob}) \geq \|x^{(k+1)} - x^\star_{\rm rob}\|^2.$$

From the properties of the Moreau-Yosida regularization [82, §7.3.1], we have

$$c^T x^\star_{\rm rob} \geq c^T x^{(k+1)} + \gamma(x^{(k+1)} - x^{(k)})^T(x^{(k+1)} - x^\star_{\rm rob})$$

which gives

$$\|x^{(k+1)} - x^\star_{\rm rob}\|^2 \leq \|x^{(k)} - x^\star_{\rm rob}\|^2 + 2/\gamma(p^\star_{\rm rob} - c^T x^{(k+1)}).$$

If $p^\star_{\rm rob} \leq c^T x^{(k+1)}$, then we have the bound

$$\|x^{(k+1)} - x^\star_{\rm rob}\| \leq \|x^{(k)} - x^\star_{\rm rob}\|,$$

and hence the distance to the optimal set is decreasing, and we can contain the optimal set in a ball around the current solution. Otherwise, we have $p^\star_{\rm rob} > c^T x^{(k+1)}$, *i.e.*, $x^{(k+1)}$ is a lower bound on the optimal value $p^\star_{\rm rob}$, and we can use the arguments in §3 to construct a robustly feasible point $\hat{x}$, which yields a bound on the suboptimality gap

$$c^T x^{(k+1)} \leq p^\star_{\rm rob} \leq c^T x^{(k+1)} + \beta V(x^{(k+1)}).$$

Now, using the Lipschitz condition (13), we know that

$$V(x^{(k)}) \leq G\|x^{(k)} - x^\star_{\rm rob}\|,$$

and as $x^{(k)} \to x^\star_{\rm rob}$, we have $V(x^{(k)}) \to 0$.

5.3.5 Interior-point methods

The next variation is based on approximately solving the sampled problem (9) at each optimization step using interior-point methods. We consider a variation based on the barrier method [1, Ch. 11]: in place of the sampled problem, we solve its *barrier approximation* given by

$$\text{minimize} \quad c^T x + \kappa\phi(x), \tag{18}$$

where $\kappa > 0$ is a parameter, and ϕ is the log barrier

$$\phi(x) = \begin{cases} \sum_{i,j} -\log\left(-f_i(x, u_{i,j})\right) & f_i(x, u_{i,j}) < 0 \\ +\infty & \text{otherwise,} \end{cases} \tag{19}$$

with $j = 1, \ldots, K_i$ and $i = 1, \ldots, m$. ($K = K_1 + \cdots + K_m$ is the total number of constraints in the sampled problem.) The solution of (18), denoted $x^\star(\kappa)$, is called a *central point*,

and can be efficiently obtained using Newton's method. Like regularization, this variation can significantly reduce the number of iterations required. Moreover, warm-start techniques (discussed below) can be used to substantially reduce the cost per iteration.

The proof of convergence given in §5.2 can be applied here almost unchanged. The central point $x^\star(\kappa)$ yields a dual feasible point, and hence a lower bound on the optimal value $p^\star_{\rm samp}$ of the sampled problem given by

$$c^T x^\star(\kappa) - p^\star_{\rm samp} \leq \kappa K.$$

(For more details, see [1, §11.2.2].) From $x^\star(\kappa)$, following §3, we can produce a robustly feasible point $\hat{x}$ and an upper bound on the optimal value $p^\star_{\rm rob}$, given by

$$c^T \hat{x} \leq p^\star_{\rm samp} + \kappa K + \beta V(x^\star(\kappa)) \leq p^\star_{\rm rob} + \kappa K + \beta V(x^\star(\kappa)).$$

Thus, the point $\hat{x}$, which is robustly feasible, is at most $\kappa K + \beta V(x^\star(\kappa))$ suboptimal for the robust problem. We can also conclude that

$$c^T x^\star(\kappa) - \kappa K \leq p^\star_{\rm rob} \leq c^T x^\star(\kappa) + \kappa K + \beta V(x^\star(\kappa)).$$

Given these bounds, we can use the same arguments as before to show that

$$\limsup_{k\to\infty} c^T \hat{x}^{(k)} \leq p^\star_{\rm rob} + \kappa K.$$

5.3.6 Warm-start

In the basic CSM or any of its variations, we solve a sequence of closely related sampled problems. If the optimization method used to solve the sampled problems can use the previously computed optimal variables (and possibly dual variables) to initialize (*i.e.*, *warm-start*) the solution of the current problem, the effort per interation can be substantially reduced. A general rule-of-thumb is that warm-start can reduce the effort per iteration by factor of up to 10 or so (for example, in the barrier method). For more on warm-starting and its benefits, see, *e.g.*, [84, 85].

6 Numerical example

We illustrate the basic CSM and some of its variations by solving a robust problem for which we know an exact solution. We consider a robust linear program (RLP) with ellipsoidal data uncertainty,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \sup_{a_i \in \mathcal{A}_i} a_i^T x \leq b_i, \quad i = 1, \ldots, m, \end{array} \tag{20}$$

where $a_i \in \mathbf{R}^n$ are uncertain but known to belong to ellipsoids $\mathcal{A}_i = \{\bar{a}_i + P_i u \mid \|u\|_2 \leq 1\}$, $b \in \mathbf{R}^m$, and $c \in \mathbf{R}^n$. The uncertainty ellipsoids are centered at the nominal vectors $\bar{a}_i \in \mathbf{R}^n$,

with their shapes described by matrices $P_i \in \mathbf{R}^{n\times n}$ and free parameters $u \in \mathbf{R}^n$. It is well-known that the RLP problem (20) can be reduced to the SOCP [1, p. 157], [2],

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \bar{a}_i^T x + \|P_i^T x\|_2 \leq b_i, \quad i = 1, \ldots, m, \end{array} \tag{21}$$

and therefore efficiently solved.

We put the RLP (20) in our framework as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_i(x) = \sup_{u\in\mathcal{U}} (\bar{a}_i + P_i u)^T x - b_i \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{22}$$

where $\mathcal{U} = \{u \,|\, \|u\|_2 \leq 1\}$. In this case, we have analytic formulas for the exact pessimizing oracles: the worst-case constraint functions are given by

$$F_i(x) = \bar{a}_i^T x + \|P_i^T x\|_2 - b_i,$$

and a worst-case parameter is given by

$$u_i^\star(x) = P_i^T x / \|P_i^T x\|_2.$$

We will also use an approximate oracle, using the nonlinear optimization solver SolvOpt [86], which implements Shor's space-dilated subgradient method [54], using function evaluation and derivative information. To ensure that the pessimization is crude enough to present a challenge to the algorithm, we specify the low relative accuracy of 10%. All sampled problems were solved using CVX [87], which internally calls the conic solver SDPT3 [88].

We generate a synthetic problem instance with $n = 50$ variables, $m = 100$ constraints, and magnitude of data values $\|a_i\|$, b_i, and $\|c\|$ around 1. We also randomly generate positive definite matrices P_i with norm 0.05. (This corresponds to about 5% uncertainty in the problem data.) The robust problem was verified to be feasible and the optimal value $p^\star_{\text{rob}}$ was computed by solving the SOCP (21).

Basic CSM. The lefthand plot in figure 1 shows the maximum constraint violation V versus iteration for the exact oracle (solid, blue line). The righthand plot shows the approximate $\hat{V}(x)$ as computed by the approximate oracle (dashed, red line), and true $V(x)$, which was computed using the worst-case formulas (and not used in the algorithm). The exact oracle results in faster convergence, but more constraints per iteration, since it captures every violating constraint, as seen in figure 2.

Dropping constraints. Here we keep at most $5n = 250$ constraints. The plots in figure 3 show maximum constraint violation V and $\hat{V}$ versus the iteration. We observe that the convergence is slower and more erratic, when compared to the basic CSM; however, we still have convergence to the optimal value. The benefit here is that we never have to solve a sampled problem with more than $5n$ constraints.

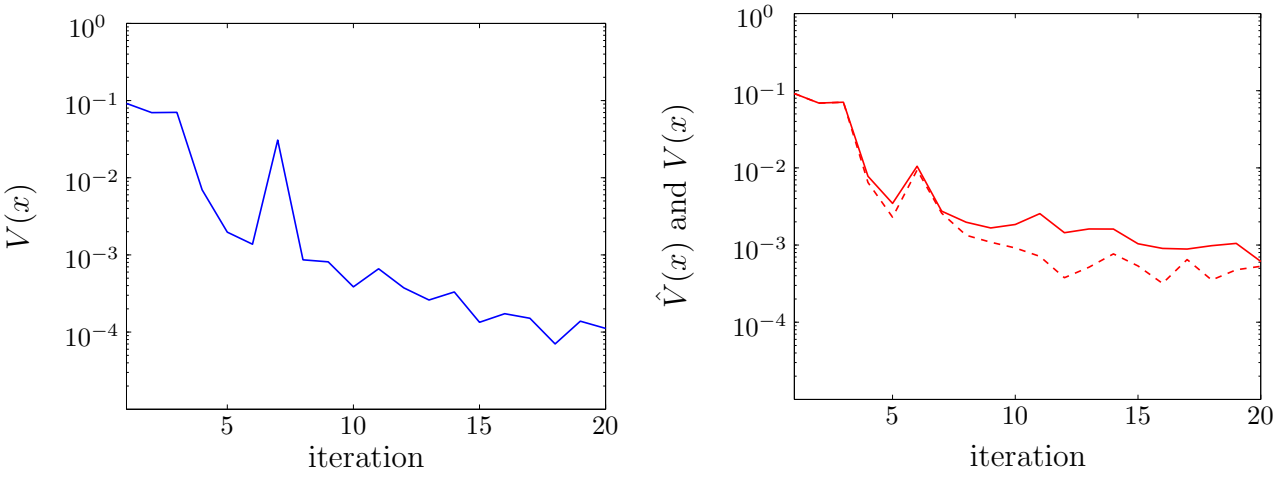

Figure 1: Basic CSM. Maximum constraint violation vs. iteration. *Left.* Exact oracle $V(x)$ (solid, blue line). *Right.* Approximate oracle $\hat{V}(x)$ (dashed, red line) and $V(x)$ (solid, red line).

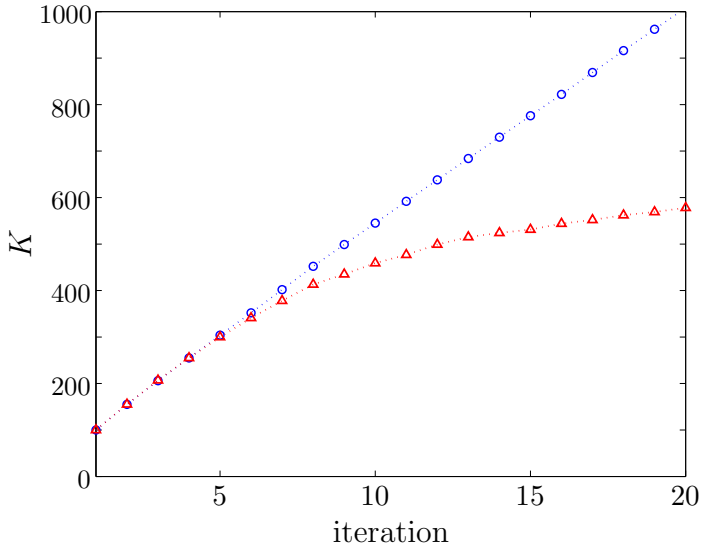

Figure 2: Basic CSM. Total number of scenarios K in $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_m$ vs. iteration for the exact oracles (blue circles) and approximate oracles (red triangles).

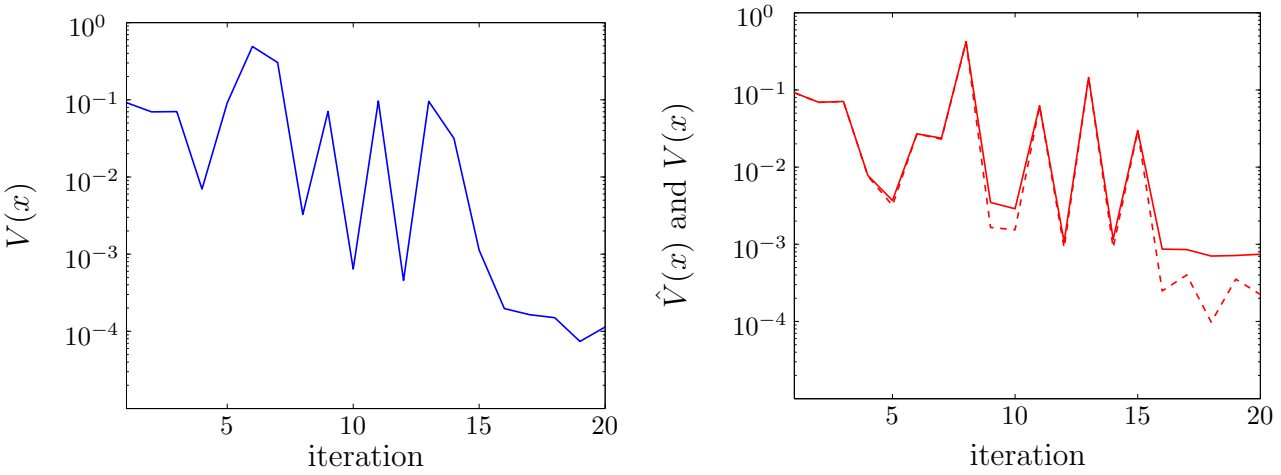

Figure 3: CSM with constraint dropping. Maximum constraint violation vs. iteration. *Left.* Exact oracle $V(x)$ (solid, blue line). *Right.* Approximate oracle $\hat{V}(x)$ (dashed, red line) and $V(x)$ (solid, red line).

Proximal CSM. We use the the proximal CSM , with the regularization parameter $\gamma = 0.1$ (obtained by experimentation). The plots in figure 4 show V and $\hat{V}$ versus the iteration.

In the case of exact oracles, we observe very rapid convergence. The number of constraints for both the exact and approximate oracles does not go much above $5n = 250$ constraints.

Barrier CSM. We use the barrier CSM, with parameter $\kappa = 10^{-4}$. The plots in figure 5 show V and $\hat{V}$ versus iteration. Like the proximal method, the barrier method gives fast convergence. As in the proximal method, the number of constraints for both the exact and approximate oracles does not go much above $5n = 250$ constraints.

We run the barrier CSM with and without warm-start (which yields the same result, of course). The warm-start barrier method was implemented using infeasible start Newton's method; see, *e.g.*, [1, §10.3]. The total number of Newton iterations required to carry out 20 iterations was 622 without warm-start, and 124 with warm-start, which is about five times reduction in the effort. (Both of these counts include the initial iteration which required 32 Newton steps to solve the nominal problem.)

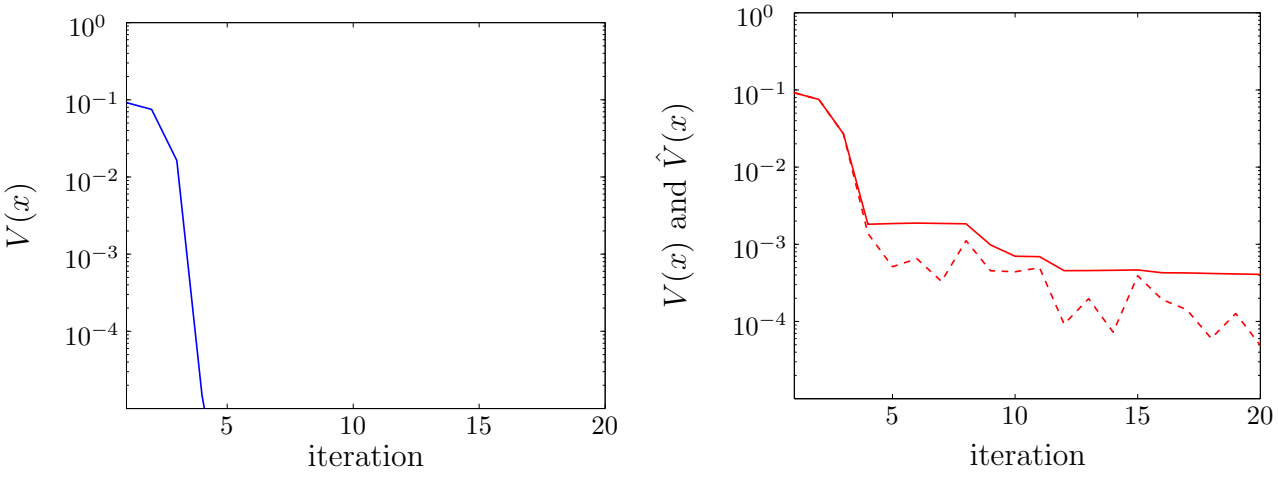

Figure 4: Proximal CSM. Maximum constraint violation vs. iteration. *Left.* Exact oracle $V(x)$ (solid, blue line). *Right.* Approximate oracle $\hat{V}(x)$ (dashed, red line) and $V(x)$ (solid, red line).

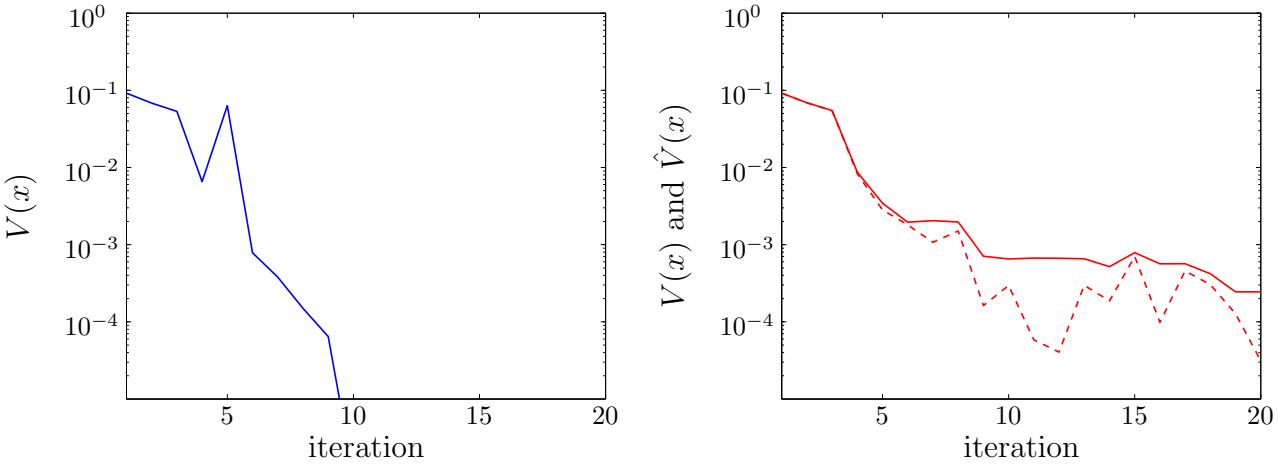

Figure 5: Barrier CSM. Maximum constraint violation vs. iteration. *Left.* Exact oracle $V(x)$ (solid, blue line). *Right.* Approximate oracle $\hat{V}(x)$ (dashed, red line) and $V(x)$ (solid, red line).

7 Robust beamforming with uncertain locations

We consider a beamforming problem with an array of n antennas. We wish to choose the antenna weights $w = (w_1, \ldots, w_n) \in \mathbf{C}^n$ to minimize the maximum array gain over the rejection angles $\theta_1, \ldots, \theta_m$, subject to the real part of the array gain being at least one in a desired direction θ_{des}, *i.e.*,

$$\begin{array}{ll} \text{minimize} & G(w) = \max_{k=1,\ldots,m} |w^* a(\theta_k)| \\ \text{subject to} & \Re(w^* a(\theta_{\mathrm{des}})) \geq 1. \end{array} \tag{23}$$

The array response vector $a(\theta) \in \mathbf{C}^n$ is given by

$$a(\theta)_j = \exp(2\pi i(x_j \cos\theta + y_j \sin\theta)), \quad j = 1, \ldots, n,$$

where $(x_j, y_j) \in \mathbf{R}^2$ is the location of the jth antenna (given in units of the wavelength), and $i = \sqrt{-1}$. The problem data are $\theta_1, \ldots, \theta_m$, θ_{des}, and the antenna positions (x_j, y_j). When all these data are known, the problem (23) can be cast as an SOCP when expressed in terms of the real and imaginary parts of the variables and data; see, *e.g.*, [3, 89].

We will consider the robust beamforming problem with uncertainty in the antenna x- and y-positions,

$$u = (x, y) \in \mathcal{U} = \{(x, y) \mid |x_j - \bar{x}_j| \leq \rho,\ |y_j - \bar{y}_j| \leq \rho\}, \tag{24}$$

where ρ is the position uncertainty. Our problem is then

$$\begin{array}{ll} \text{minimize} & G_{\mathrm{wc}}(w) = \sup_{u\in\mathcal{U}} \max_k |w^* a(\theta_k, u)| \\ \text{subject to} & \inf_{u\in\mathcal{U}} \Re(w^* a(\theta_{\mathrm{des}}, u)) \geq 1, \end{array} \tag{25}$$

where now we show the dependence of a on u explicitly.

We put this problem in our form by introducing a new variable t, and forming the epigraph problem

$$\begin{array}{lll} \text{minimize} & t & \\ \text{subject to} & F_k(w) = \sup_{u\in\mathcal{U}} |w^* a(\theta_k, u)| - t \leq 0, & k = 1, \ldots, m \\ & F_{m+1}(w) = \sup_{u\in\mathcal{U}} (1 - \Re(w^* a(\theta_{\mathrm{des}}, u))) \leq 0. & \end{array} \tag{26}$$

We will use the basic CSM to solve (26), with an exact pessimization oracle we describe below. For work on other robust beamforming problems (but not including this uncertainty model), see [37–39, 90, 91].

7.1 Exact pessimization oracle

In this section we describe a method for computing the worst-case (largest) antenna gain, $\sup_{u\in\mathcal{U}} |w^* a(\theta, u)|$. We first observe that

$$\sup_{u\in\mathcal{U}} |w^* a(\theta, u)| \;=\; \sup_{u\in\mathcal{U}} \sup_{|q|=1} \Re\left(w^* a(\theta, u))q\right)$$

$$
\begin{aligned}
&= \sup_{|q|=1} \sup_{u\in\mathcal{U}} \sum_{j=1}^{n} \Re\left(q\overline{w}_j \exp(2\pi i(x_j c + y_j s))\right) \\
&= \sup_{|q|=1} \sum_{j=1}^{n} \sup_{|x_j-\overline{x}_j|\leq\rho,\ |y_j-\overline{y}_j|\leq\rho} \Re\left(q\overline{w}_j \exp(2\pi i(x_j c + y_j s))\right),
\end{aligned}
$$

where $c = \cos\theta$, $s = \sin\theta$. We now show how to compute

$$
\sup_{|x_j-\overline{x}_j|\leq\rho,\ |y_j-\overline{y}_j|\leq\rho} \Re\left(q\overline{w}_j \exp(2\pi i(x_j c + y_j s)\right), \tag{27}
$$

for any given q. As x_j and y_j range over the box, $x_j c + y_j s$ varies over the interval

$$
(\overline{x}_j c + \overline{y}_j s) \pm \rho(|c| + |s|).
$$

Therefore $q\overline{w}_j \exp(2\pi i(x_j c + y_j s))$ varies over an arc in the complex plane, centered at 0 with radius $|w_j|$ (including, possibly, the whole circle). Its maximum real part is either $|w_j|$, or occurs at one of the arc endpoints, and so is readily calculated.

We can evaluate (27) for $q = \exp(2\pi ik/N)$, for $k = 1, \ldots, N$. The maximum of these values gives an under approximation of (lower bound on) $\sup_{u\in\mathcal{U}} |w^* a(\theta, u)|$; multiplying the lower bound by $1 + (\pi/N)^2$ yields an upper bound, since for any $z \in \mathbf{C}$ we have

$$
|z| \leq \max_{k=1,\ldots,N} \Re(\exp(2\pi ik/N)z) \leq \left(1 + \sin^2(\pi/N)\right)^{1/2} |z| \leq (1 + (\pi/N)^2)|z|.
$$

Thus, the maximum of (27) for $q = \exp(2\pi ik/N)$, for $k = 1, \ldots, N$, is an approximation of $\sup_{u\in\mathcal{U}} |w^* a(\theta, u)|$ within an accuracy that decreases as $1/N^2$.

The same technique can be used to evaluate $\sup_{u\in\mathcal{U}} (1 - \Re(w^* a(\theta_{\rm des}, u)))$.

7.2 Numerical example

We illustrate the method with a numerical example, with $n = 40$ antennas, with their nominal locations generated randomly in the square $[0, 5] \times [0, 5]$. We take $\theta_{\rm des} = 60^\circ$, and pick rejection angles $\theta_k \in \{1^\circ, \ldots, 40^\circ, 80^\circ, \ldots, 360^\circ\}$, *i.e.*, we wish to reject signals arriving from directions outside the beam $(40^\circ, 80^\circ)$. We set the location uncertainty level to $\rho = 0.03$, which corresponds to a maximum phase shift of about $\pm 15^\circ$ at $\theta = 60^\circ$. We set $N = 36$ for the exact pessimizing oracles, which means that our error in computing $F_k(w)$ is no more than 0.76%.

The nominal optimal beamformer achieves an excellent rejection level with the nominal antenna locations: we have $G(w^\star_{\rm nom}) = 2.59 \times 10^{-3}$. The antenna gain, with these antenna weights, is extremely sensitive to uncertainty in the antenna positions. Figure 6 shows the gain pattern of the nominal optimal beamformer given the nominal antenna locations and the worst-case locations.

The nominal rejection level for the robust beamformer is $G(w^\star_{\rm rob}) = 0.205$ and the worst-case rejection level is $G_{\rm wc}(w^\star_{\rm rob}) = 0.212$. Therefore, the robust beamformer performs well (and about the same), over all possible antenna locations parametrized by $u \in \mathcal{U}$. Figure 7

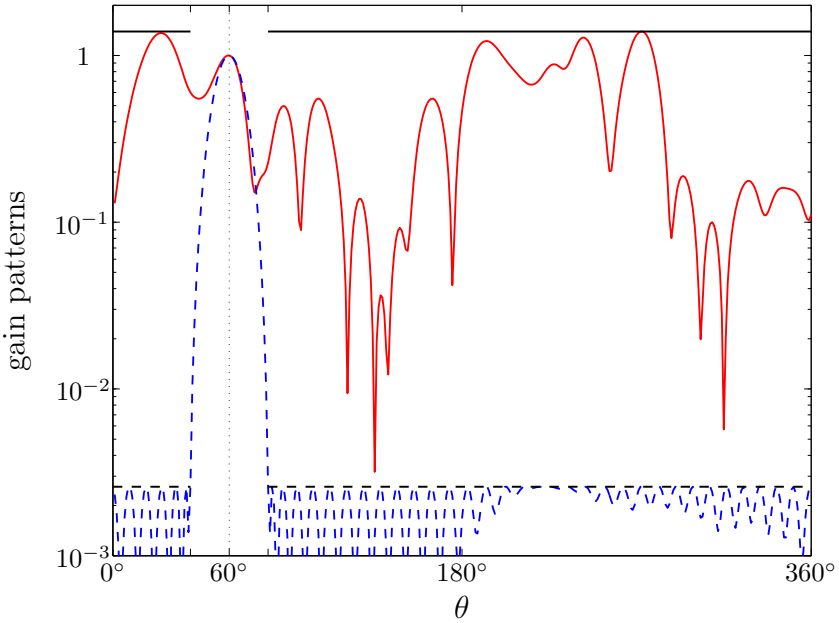

Figure 6: Nominal optimal beamformer gain pattern with nominal antenna locations (dashed blue line), and with worst-case antenna locations (solid red line). Solid and dashed horizontal lines represent the worst-case and nominal rejection levels, respectively.

shows the gain pattern of the robust design, with the nominal and the worst-case antenna locations.

The basic CSM converges with tolerance $V^{\mathrm{tol}} = 10^{-4}$ in 20 iterations; a very good design is obtained in only 10 iterations. Figure 8 shows the maximum constraint violation $V(w)$ versus iteration.

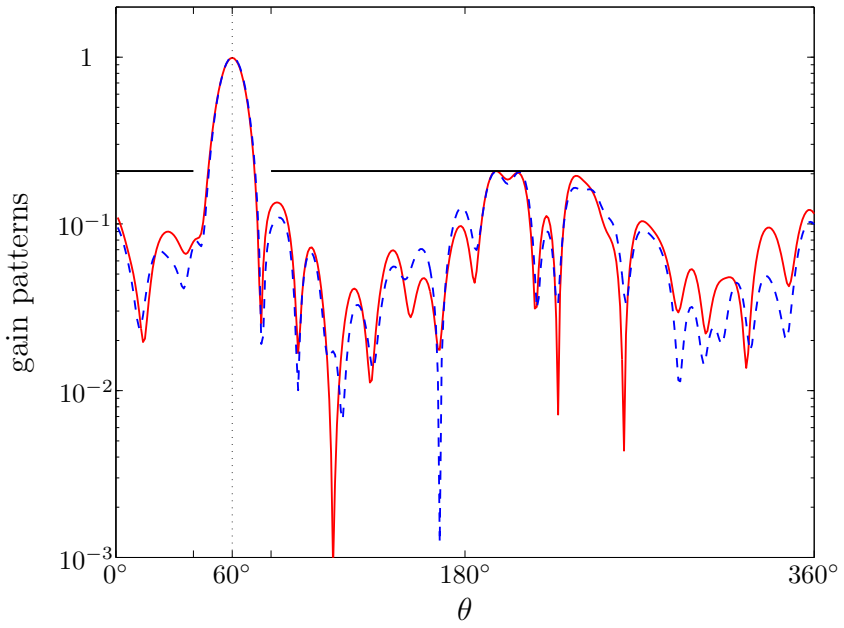

Figure 7: Robust optimal beamformer gain pattern with nominal antenna locations (dashed blue line), and with worst-case antenna locations (solid red line). Solid horizontal line represents the worst-case rejection level (the nominal rejection level is about the same).

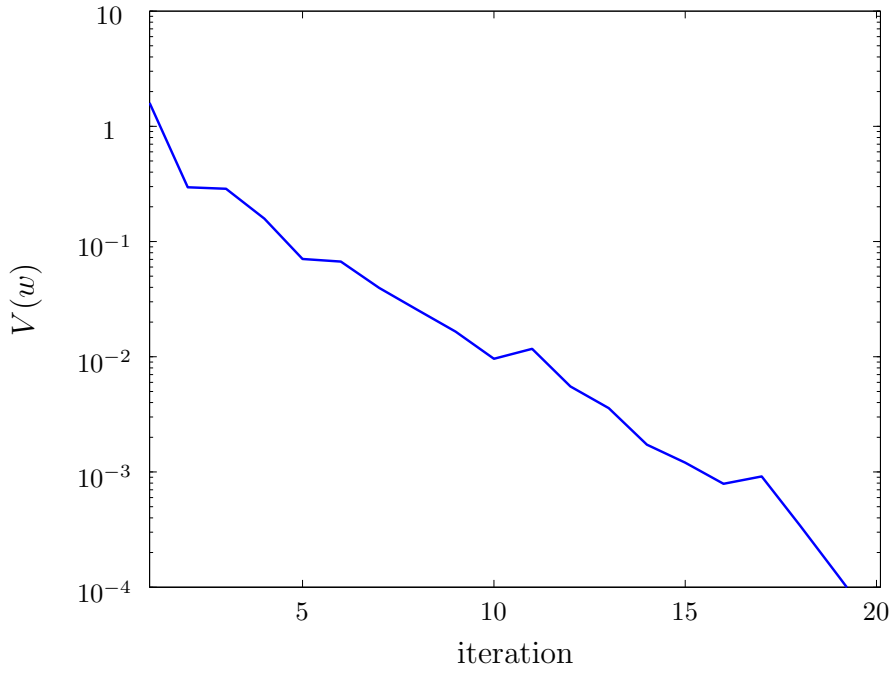

Figure 8: Maximum constraint violation $V(w)$ vs. iteration.

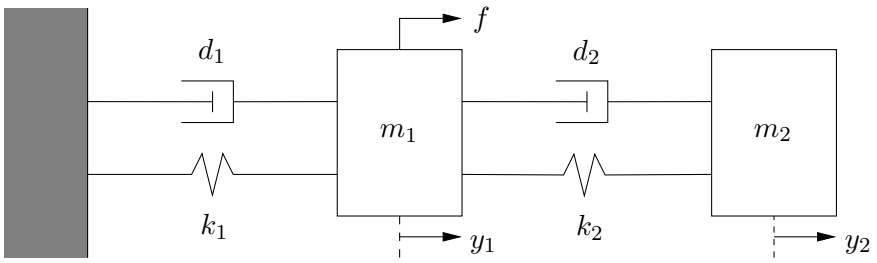

Figure 9: Two mass system connected with strings and dashpots.

8 Robust state transfer

We consider the spring-dashpot-mass system shown in figure 9, where y_1 and y_2 are the displacements of the two masses, m_1 and m_2 are their mass values, f is the force applied to the first mass, k_1 and k_2 are the spring constants, and d_1 and d_2 are the damping coefficients. The dynamics is given by

$$\dot{x}(t) = Ax(t) + Bf(t), \quad x(0) = 0, \tag{28}$$

where $x = (y_1, y_2, \dot{y}_1, \dot{y}_2) \in \mathbf{R}^4$ is the state, $f(t) \in \mathbf{R}$ is the input force, and

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(k_1+k_2)/m_1 & k_2/m_1 & -(d_1+d_2)/m_1 & d_2/m_1 \\ k_2/m_2 & -k_2/m_2 & d_2/m_2 & -d_2/m_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1/m_1 \\ 0 \end{bmatrix}.$$

The input force is piecewise constant over the interval $[0, T]$, where $T = Nh$,

$$f(t) = f_i, \quad ih \leq t < (i+1)h, \quad i = 0, 1, \ldots, N-1.$$

Therefore we have

$$x(T) = \sum_{i=0}^{N-1} A_{\mathrm{d}}^{(N-1-i)} B_{\mathrm{d}} f_i, \tag{29}$$

where

$$A_{\mathrm{d}} = e^{hA}, \qquad B_{\mathrm{d}} = \left(\int_0^h e^{\tau A} \, d\tau \right) B = A^{-1}(A_{\mathrm{d}} - I)B.$$

In the sequel we will let f denote the (discrete) force vector $f = (f_0, \ldots, f_{N-1})$. We note that $x(T)$ is linear in f, for given values of the mass, spring, and damping parameters. But $x(T)$ depends on the parameters in a very complicated way.

Our parameter uncertainty is given by

$$m_i \in [\underline{m}_i, \overline{m}_i], \quad k_i \in [\underline{k}_i, \overline{k}_i], \quad d_i \in [\underline{d}_i, \overline{d}_i].$$

(Such parameter interval uncertainty is often used in the robust control literature [92, 93].)

Our goal is to choose f, with $|f_i| \leq f_{\max}$, that achieves $x(T) \approx x_{\text{des}}$, where x_{des} is some given desired target state, despite the variations in the mass, spring, and damping parameters. Using the Euclidean norm to measure deviation in the final state, we arrive at the robust state transfer problem,

$$\begin{array}{ll} \text{minimize} & \sup \|x(T) - x_{\text{des}}\|_2 \\ \text{subject to} & |f_i| \leq f_{\max}, \quad i = 1, \ldots, N-1, \end{array} \tag{30}$$

where the supremum is over the parameters. We put this in our form as

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \sup \|x(T) - x_{\text{des}}\|_2 - t \leq 0 \\ & |f_i| - f_{\max} \leq 0, \quad i = 1, \ldots, N-1, \end{array} \tag{31}$$

with additional variable t. In this problem only the first constraint is affected by the uncertain parameters.

As a simple approximate pessimization oracle for the first constraint, we evaluate $\|x(T) - x_{\text{des}}\|$ at the 64 vertices of the parameter set. We do not know that the state deviation is maximized over the box at one of its vertices, but it seems likely.

8.1 Numerical example

We take the parameter set to be

$$m_i \in [0.9, 1.1], \quad k_i \in [2.9, 3.1], \quad d_i \in [0.09, 0.11],$$

where the nominal parameters are $m_i = 1$, $k_i = 3$, and $d_i = 0.1$, $i = 1, 2$. We take time horizon $T = 5$, with update interval $h = 0.1$, so $N = 50$. The maximum force is $f_{\max} = 2.5$, and the desired state is $x_{\text{des}} = (0, 1, 0, 0)$.

With the nominal parameter values, it is possible to choose f so that $x(T) = x_{\text{des}}$. The top plot in figure (10) shows one such force. The associated (approximate) worst-case state deviation for this force vector is, however, 0.68.

The basic CSM computes a robust force with very small worst-case (approximate) worst-case violation after 6 iterations. The obtained robust force is shown in figure 10. The (approximate) worst-case state deviation is 0.15, around fifth the value for our nominal force. (The state deviation with nominal parameters is 0.04.) Histograms of the state deviation for both the nominal and the robust forces, over the 64 extreme points, are shown in figure (11), and state trajectories are shown in figure (12).

To verify our approximate pessimization oracle, we ran various local optimization methods from the vertices, and checked 10^5 randomly chosen parameter values, for our nominal and robust forces. We found no parameter that was worse than the worst vertex value. While we cannot claim that our approximate oracle is exact, this seems very likely.

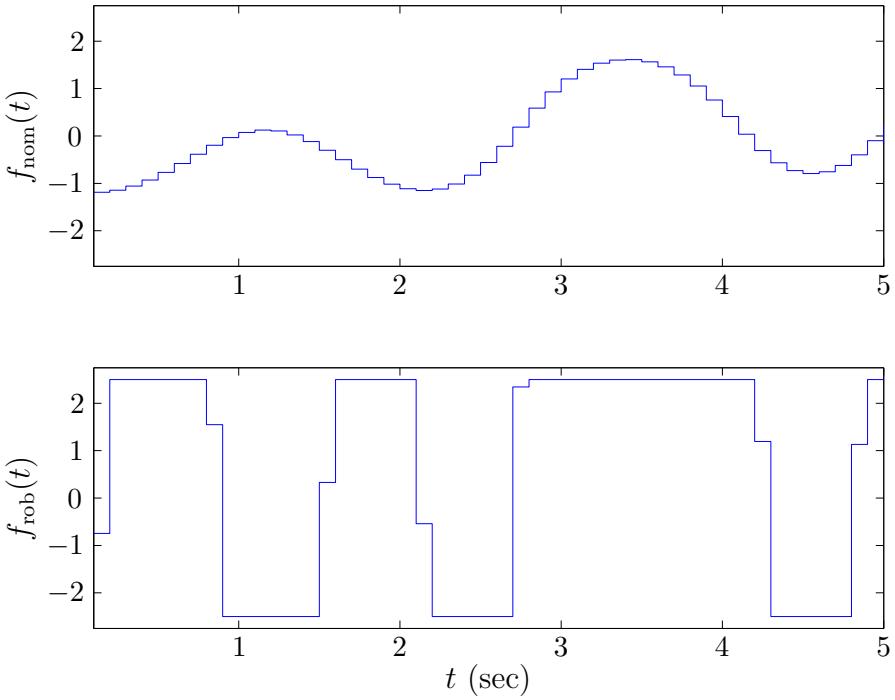

Figure 10: *Top.* Nominal optimal force profile. *Bottom.* Robust force profile.

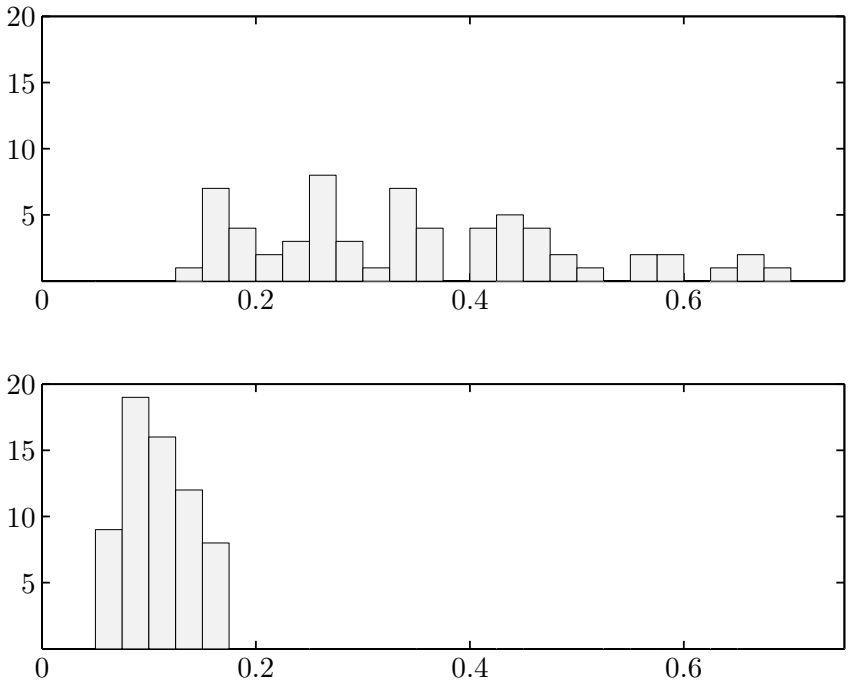

Figure 11: Histogram of state deviation over 64 vertices of the parameter set. *Top.* Nominal optimal force. *Bottom.* Robust force.

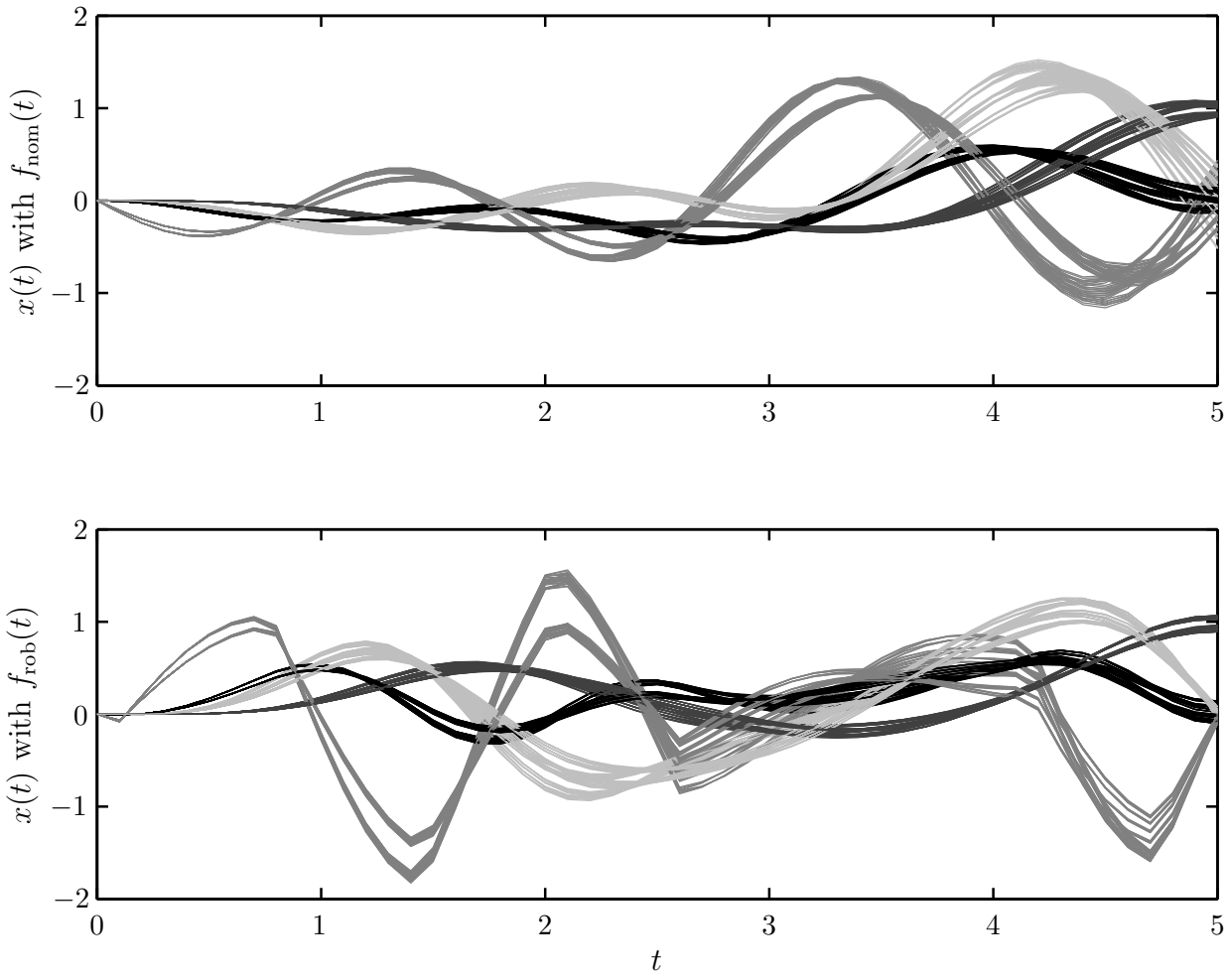

Figure 12: State trajectories for all 64 vertices of the parameter set. The darkest to the lightest curves are $x_1(t)$ through $x_4(t)$. *Top.* With nominal optimal force. *Bottom.* With robust force.

9 Conclusions

We have presented a basic cutting-set method, and several variations, for solving convex worst-case robust optimization problems. These methods alternate between finite-scenario robust optimization steps, and pessimization (worst-case analysis) steps, and so are quite practical when the base problem is convex. Using the barrier or regularized versions of the method, with warm-start techniques to accelerate the solve steps, practical robust solutions can be computed with an effort that is a modest multiple of the effort required to solve the non-robust version of the problem.

Acknowledgments

This work was supported in part by Dr. Dennis Healy of DARPA MTO, under award #N00014-05-1-0700 administered by the Office of Naval Research, by NSF award #0529426, and by AFOSR award #FA9550-06-1-0312.

References

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

[2] A. Ben-Tal and A. Nemirovski, "Robust solutions of uncertain linear programs," *Oper. Res. Lett.*, vol. 25, pp. 1–13, 1999.

[3] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and Its Applications*, vol. 284, pp. 193–228, 1998.

[4] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," *Mathematics of Operations Research*, vol. 23, no. 4, pp. 769–805, 1998.

[5] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization. Analysis, Algorithms, and Engineering Applications.* SIAM, 2001.

[6] A. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Soviet Math. Dokl.*, vol. 4, pp. 1035–1038, 1963.

[7] M. Diehl, H. Bock, and E. Kostina, "An approximation technique for robust nonlinear optimization," *Math. Prog. Series B*, vol. 107, pp. 213–230, June 2006.

[8] A. Prekopa, *Stochastic Programming*. Kluwer Academic Publishers, 1995.

[9] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York: Springer, 1997.

[10] A. Shapiro and A. Ruszczynski, *Lectures on Stochastic Programming*. Book in progress. Available at `www2.isye.gatech.edu/~ashapiro/publications.html`, Nov. 2007.

[11] A. Shapiro and A. Philpott, "A tutorial on stochastic programming." Manuscript. Available at `www2.isye.gatech.edu/~ashapiro/publications.html`, Mar. 2007.

[12] R. T. Rockafellar and R. J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty," *Mathematics of Operations Research*, vol. 16, no. 1, pp. 119–147, 1991.

[13] G. Calafiore and F. Dabbene, eds., *Probabilistic and Randomized Methods for Design under Uncertainty*. Springer-Verlag, 2006.

[14] G. Calafiore and M. Campi, "Uncertain convex programs: randomized solutions and confidence levels," *Mathematical Programming*, vol. 102, no. 1, pp. 25–46, 2005.

[15] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski, *Robust Optimization*. Book in progress. Available at `www2.isye.gatech.edu/~nemirovs/`, Dec. 2007.

[16] G. Dantzig, "Linear programming under uncertainty," *Management Science*, vol. 1, pp. 197–206, 1955.

[17] A. Soyster, "Convex programming with set-inclusive constraints and applications to inexact linear programming," *Oper. Res.*, vol. 21, pp. 1154–1157, 1973.

[18] A. Ben-Tal and A. Nemirovski, "Robust optimization: methodology and applications," *Mathematical Programming Series B*, vol. 92, pp. 453–480, May 2002.

[19] Y. Nesterov and A. Nemirovski, *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. SIAM, 1994.

[20] Y. Ye, *Interior Point Algorithms: Theory and Analysis*. New York: John Wiley, 1997.

[21] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer-Verlag, 2nd ed., 2006.

[22] D. Bertsimas, D. Brown, and C. Caramanis, "Theory and applications of robust optimization." Submitted. Available at `users.ece.utexas.edu/~cmcaram/pubs/RobustOptimizationLV.pdf`, 2007.

[23] D. Bertsimas, D. Pachamanova, and M. Sim, "Robust linear optimization under general norms," *Oper. Res. Lett.*, 2003.

[24] D. Bertsimas and M. Sim, "The price of robustness," *Oper. Res.*, vol. 52, pp. 35–53, January 2004.

[25] L. E. Ghaoui and H. Lebret, "Robust solutions to least-squares problems with uncertain data," *SIAM J. Matrix Anal. Appl.*, vol. 18, no. 4, pp. 1035–1064, 1997.

[26] S. Chandrasekaran, G. Golub, M. Gu, and A. Sayed, "Parameter estimation in the presence of bounded data uncertainties," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, pp. 235–252, 1998.

[27] D. Goldfarb and G. Iyengar, "Robust convex quadratically constrained programming," *Mathematical Programming*, vol. 97, pp. 495–515, August 2003.

[28] L. E. Ghaoui, F. Oustry, and H. Lebret, "Robust solutions to uncertain semidefinite programs," *SIAM J. Optim.*, vol. 9, no. 1, pp. 33–52, 1998.

[29] D. Bertsimas and M. Sim, "Robust conic optimization," *Submitted to Mathematical Programming*, 2004.

[30] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows," *Mathematical Programming*, vol. 98, pp. 49–72, September 2003.

[31] K. Zhou, J. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice-Hall, 1996.

[32] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, vol. 51, pp. 742–753, May 2006.

[33] B. Rustem and M. Howe, *Algorithms for Worst-case Design and Applications to Risk Management*. Princeton University Press, 2002.

[34] L. E. Ghaoui, M. Oks, and F. Oustry, "Worst-case value-at-risk and robust portfolio optimization: A conic programming approach," *Oper. Res.*, vol. 51, pp. 543–556, Jul 2003.

[35] D. Goldfarb and G. Iyengar, "Robust portfolio selection problems," *Mathematics of Operations Research*, vol. 28, pp. 1–38, February 2003.

[36] S.-J. Kim and S. Boyd, "Robust efficient frontier analysis with a separable uncertainty model," *Working paper*, Nov. 2007.

[37] S. Vorobyov, A. Gershman, and Z.-Q. Luo, "Robust adaptive beamforming using worst-case performance optimization: a solution to the signal mismatch problem," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 313–324, 2003.

[38] J. Li, P. Stoica, and Z. Wang, "On robust Capon beamforming and diagonal loading," *IEEE Transactions on Signal Processing*, vol. 51, pp. 1702–1715, July 2003.

[39] R. Lorenz and S. Boyd, "Robust minimum variance beamforming," *IEEE Transactions on Signal Processing*, vol. 53, pp. 1684–1696, May 2005.

[40] G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, M. Jordan, and B. Scholkopf, "A robust minimax approach to classifIcation," *Journal of Machine Learning Research*, vol. 3, no. 3, pp. 555–582, 2003.

[41] D. de Farias and B. V. Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *Mathematics of Operations Research*, vol. 29, no. 3, pp. 462–478, 2004.

[42] R. Hettich and K. Kortanek, "Semi-infinite programming: theory, methods, and applications," *SIAM Review*, 1993.

[43] A. V. Fiacco and K. Kortanek, eds., *Semi-infinite programming and applications. Proceedings.*, vol. 215 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1981.

[44] R. Reemtsen and J.-J. Ruckmann, eds., *Semi-infinite programming*, vol. 25 of *Nonconvex Optimization and its Applications*. Kluwer, 1998.

[45] M. Goberna and M. Lopez, eds., *Semi-infinite programming: Recent advances*, vol. 57 of *Nonconvex Optimization and its Applications*. Kluwer, 2001.

[46] J. E. Kelley, "The cutting plane method for solving convex programs," *Journal of the SIAM*, vol. 8, pp. 703–712, 1960.

[47] R. Hettich, "A review of numerical methods for semi-infinite optimization," *Lecture Notes In Economics and Mathematical Systems 215*, pp. 158–178, 1981.

[48] E. Polak and L. He, "Rate-preserving discretization strategies for semi-infinite programming and optimal-control," *SIAM Journal on Control and Optimization*, vol. 30, no. 3, pp. 548–572, 1992.

[49] L. Lasdon, *Optimization Theory for Large Systems*. MacMillan, 1970.

[50] J. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[51] D. Bienstock, "Histogram models for robust portfolio optimization." Manuscript. Available at `www.columbia.edu/~dano/papers/rqp.pdf`, July 2007.

[52] V. Dem'yanov and V. Malozemov, *Introduction to Minimax*. New York: Wiley, 1974.

[53] V. Dem'yanov and L. Vasil'ev, *Nondifferentiable Optimization*. Optimization Software, Inc., 1985.

[54] N. Shor, *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics, Springer, 1985.

[55] N. Shor, *Nondifferentiable Optimization and Polynomial Problems*. Kluwer Academic Publishers, 1998.

[56] E. Polak, "On the mathematical foundations of nondifferentiable optimization in engineering design," *SIAM Review*, vol. 29, no. 1, pp. 21–89, 1987.

[57] R. Bland, D. Goldfarb, and M. Todd, "The ellipsoid method: a survey," *Operations Research*, vol. 29, no. 6, pp. 1039–1091, 1981.

[58] J.-L. Goffin and J.-P. Vial, "Cutting planes and column generation techniques with the projective algorithm," *Journal of Optimization Theory and Applications*, vol. 65, pp. 409–429, 1990.

[59] J.-L. Goffin and J.-P. Vial, "Convex nondifferentiable optimization: a survey focussed on the analytic center cutting plane method," Tech. Rep. 99.02, Logilab, Geneva, Switzerland, 1999.

[60] J. Burke, A. Lewis, and M. Overton, "Approximating subdifferentials by random sampling of gradients," *Mathematics of Operations Research*, vol. 27, pp. 567–584, 2002.

[61] J. Burke, A. Lewis, and M. Overton, "A robust gradient sampling algorithm for nonsmooth, nonconvex optimization," *SIAM Journal on Optimization*, vol. 25, pp. 751–779, 2005.

[62] O. Stein and G. Still, "Solving semi-infinite optimization problems with interior point techniques," *SIAM J. Control Optim.*, vol. 42, no. 3, pp. 769–788, 2003.

[63] I. Pólik and T. Terlaky, "A survey of the S-Lemma," *SIAM Review*, vol. 49, no. 3, pp. 371–418, 2007.

[64] J. Nocedal and S. Wright, *Numerical optimization*. Springer, New York, 1999.

[65] J. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, 1965.

[66] T. Kolda, R. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, pp. 385–482, Aug. 2003.

[67] B. Gartner, *Randomized Optimization by Simplex-Type Methods*. PhD thesis, Freie Universitat Berlin, Dec. 1995.

[68] D. Bertsimas and S. Vempala, "Solving convex programs by random walks," *J. ACM*, vol. 51, no. 4, 2004.

[69] A. Conn, K. Scheinberg, and P. Toint, "Recent progress in unconstrained nonlinear optimization without derivatives," *Math. Program.*, vol. 79, pp. 397–414, 1997.

[70] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[71] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

[72] Y. Censor and S. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[73] Z.-Q. Luo and J. Sun, "A polynomial cutting surfaces algorithm for the convex feasibility problem defined by self-concordant inequalities," *Computational Optimization and Applications*, vol. 15, no. 2, pp. 167–191, 2000.

[74] D. Luenberger, *Linear and Nonlinear Programming*. Addison-Wesley Publishing, 2 ed., 1984.

[75] S. Boyd and C. Barratt, *Linear Controller Design: Limits of Performance*. Prentice Hall, 1991.

[76] B. Polyak, *Introduction to Optimization*. Optimization Software, Inc., 1987.

[77] J. Elzinga and T. J. Moore, "A central cutting plane algorithm for the convex programming problem," *Mathematical Programming*, vol. 8, pp. 134–145, 1975.

[78] A. Potchinkov and R. Reemtsen, "A globally most violated cutting plane method for complex minimax problems with application to digital filter design," *Numerical Algorithms*, vol. 5, pp. 611–620, 1993.

[79] P. Gribik, "A central cutting plane algorithm for semi-infinite programming problems," in *Semi-Infinite Programming* (R. Hettich, ed.), vol. 15 of *Lecture Notes In Control and Information Sciences*, pp. 66–82, 1979.

[80] K. Kortanek and H. No, "A central cutting plane algorithm for convex semi-infinite programming problems," *SIAM Journal on Optimization*, vol. 3, no. 4, pp. 901–918, 1993.

[81] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[82] A. Ruszczyński, *Nonlinear Optimization*. Princeton University Press, 2006.

[83] A. Kaplan and R. Tichatschke, "Proximal interior point approach for solving convex semi-infinite programming problems," *Journal of Optimization Theory and Applications*, vol. 98, pp. 399–429, 1998.

[84] E. Yildirim and S. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 782–810, 2002.

[85] J. Gondzio and A. Grothey, "Reoptimization with the primal-dual interior point method," *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 842–864, 2003.

[86] A. Kuntsevich and F. Kappel, "SolvOpt, version 1.1." Available at `www.uni-graz.at/imawww/kuntsevich/solvopt/`, June 1997.

[87] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming, version 1.0RC." Available at `www.stanford.edu/~boyd/cvx/`, Oct. 2006.

[88] K.-C. Toh, M. J. Todd, and R. H. Tutuncu, "SDPT3: Matlab software for semidefinite-quadratic-linear programming, ver. 4.0 (beta)." Available at `www.math.nus.edu.sg/~mattohkc/sdpt3.html`, July 2006.

[89] H. Lebret and S. Boyd, "Antenna array pattern synthesis via convex optimization," *IEEE Transactions on Signal Processing*, vol. 45, pp. 526–532, June 1997.

[90] A. Mutapcic, S.-J. Kim, and S. Boyd, "Beamforming with uncertain weights," *IEEE Signal Processing Letters*, vol. 14, no. 5, pp. 352–354, 2007.

[91] S.-J. Kim, A. Magnani, A. Mutapcic, S. Boyd, and Z.-Q. Luo, "Robust beamforming via worst-case SINR maximization," *to appear in IEEE Transactions on Signal Processing*, 2008.

[92] S. Bhattacharyya, H. Chapellat, and L. Keel, *Robust Control: The Parametric Approach*. Prentice-Hall, 1994.

[93] M. A. Dahleh and I. Diaz-Bobillo, *Control of Uncertain Systems: A Linear Programming Approach*. Prentice-Hall, 1995.

[94] G. Calafiore and B. T. Polyak, "Stochastic algorithms for exact and approximate feasibility of robust LMIs," *IEEE Trans. on Automatic Control*, vol. 46, pp. 1755–1759, Nov. 2001.

[95] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in Systems and Control Theory*. Philadelphia: SIAM, 1994.

[96] A. Sayed, V. Nascimento, and S. Chandrasekaran, "Estimation and control with bounded data uncertainties," *Linear Algebra and its Applications*, vol. 284, pp. 259–306, Nov. 1998.