Robust Steady-State Target Calculation for Model Predictive Control

Dean E. Kassmann and Thomas A. Badgwell
Dept. of Chemical Engineering, Rice University, Houston, TX 77005

Robert B. Hawkins
Advanced Control Systems, Aspen Technology, Inc., Houston, TX 77077

In practice, model predictive control (MPC) algorithms are typically embedded within a multilevel hierarchy of control functions. The MPC algorithm itself is usually implemented in two pieces: a steady-state target calculation followed by a dynamic optimization. A new formulation of the steady-state target calculation is presented that explicitly accounts for model uncertainty. When model uncertainty is incorporated, the linear program associated with the steady-state target calculation can be recast as a second-order cone program. This article shows how primal-dual interior-point methods can take advantage of the resulting structure. Simulation examples illustrate the effect of uncertainty on the steady-state target calculation and demonstrate the advantages of interior-point methods.

Introduction

Model predictive control (MPC) refers to a class of computer control algorithms that compute a sequence of manipulated variable adjustments in order to optimize the future behavior of a plant. As its name suggests, an MPC algorithm uses an explicit model to predict how a process will evolve in time. The prediction is used to determine optimal control moves that will bring the process to a desired steady state. The optimal control moves are computed using an on-line optimization which is, in general, a full-blown nonlinear program (NLP). In practice, linear models are most often used and the resulting optimizations are linear or quadratic programs. While originally developed to address the needs of power plants and petroleum refineries, MPC technology is now used in a wide variety of applications ranging from food processing to pulp and paper (Qin and Badgwell, 1997). This remarkable success has captured the attention of both the industrial and academic control communities.

In modern processing plants, MPC is implemented as part of a multilevel hierarchy of control functions. Figure 1 shows a representative control hierarchy. At the top of the structure, a plant-wide optimizer determines optimal steady-state settings for each unit in the plant. These may be sent to local optimizers at each unit which run more frequently or consider a more detailed unit model than is possible at the plant-wide level. The unit optimizer computes an optimal economic steady state and passes this information to the MPC algorithm for implementation. The MPC algorithm must move the plant from one constrained steady state to another, while minimizing constraint violations along the way. Figure 1 also shows that the MPC portion can be further divided into a steady-state calculation and a dynamic calculation.

The goal of the steady-state target calculation is to recalculate the targets from the local optimizer every time the MPC steady state controller executes. This must be done because disturbances entering the system or new input information from the operator may change the location of the optimal steady state. This is a standard idea that dates back to the early days of optimal control theory (Kwakernaak and Sivan, 1972). Cutler et al. (1983) discuss this idea in the context of MPC. More recently, this problem has been investigated by Muske and Rawlings (1993), Muske (1997), and Rao and Rawlings (1999). These authors assume a perfect model and cast the steady-state target algorithm as a quadratic program. Today, the separation of the MPC algorithm into a steady-state and dynamic calculation is a common part of industrial MPC technology. Since the steady-state algorithm must run more frequently than the local optimizer, it uses a less detailed model. In practice, it uses a steady-state version of the

Correspondence concerning this article should be addressed to T. A. Badgwell.
Current address of D. E. Kassmann and T. A. Badgwell: Aspen Technology, Inc., Advanced Control Systems; Houston, TX 77077.

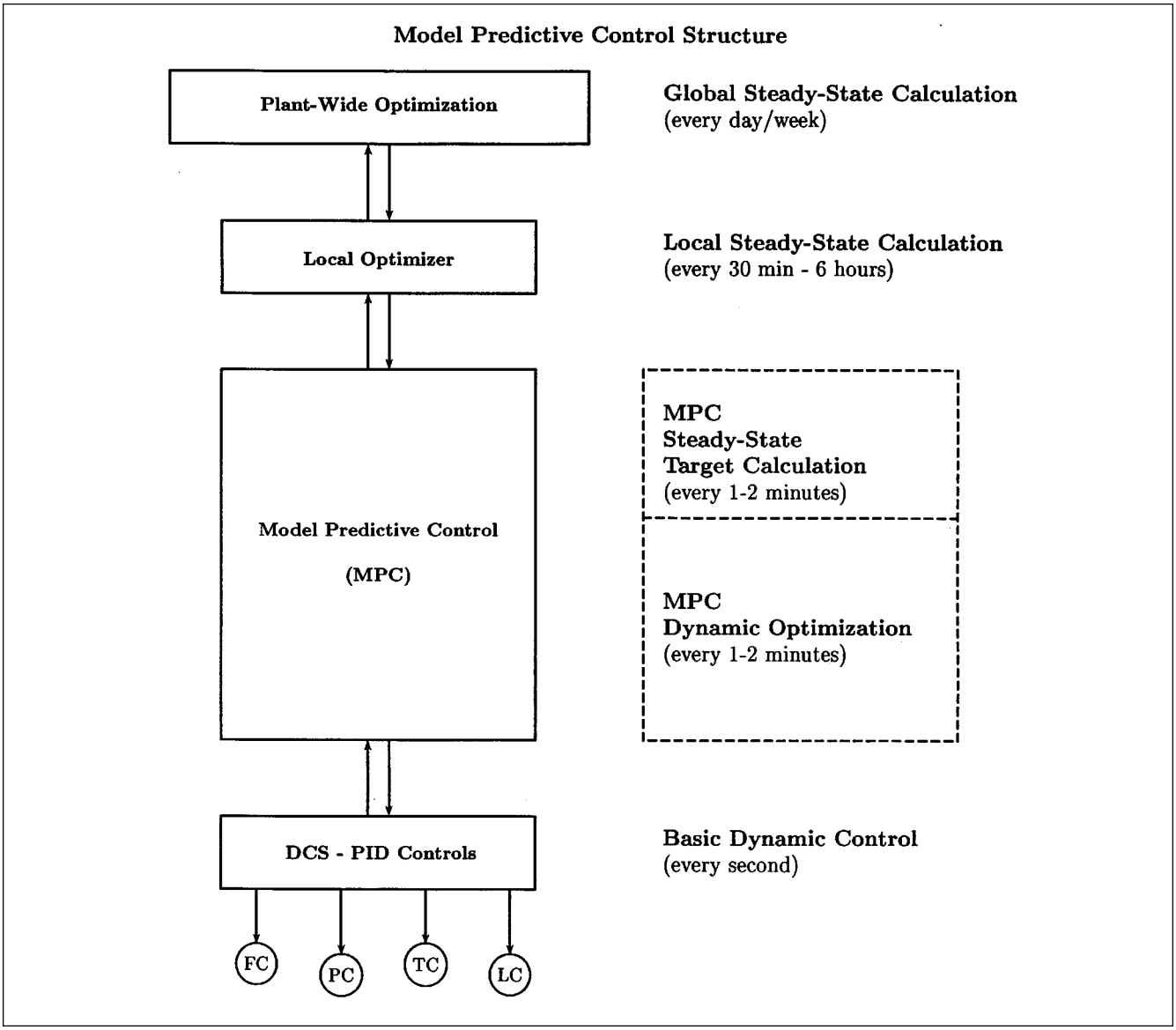

Figure 1. MPC control hierarchy.

dynamic model used for the dynamic optimization.

The recalculated optimal steady state is then passed to the dynamic MPC algorithm which determines how to best go from one constrained steady state to the next. The dynamic MPC calculation (dynamic constraint control) has been studied extensively (see Qin and Badgwell (1997) and the references therein). The steady-state target optimization can be thought of as an alternative way to incorporate feedback into the MPC algorithm without explicitly including it in the dynamic MPC calculation.

In this work, we assume the MPC calculation is separated into dynamic and steady-state calculations with the steady-state optimization driven by economics. We limit our discussion to stable linear systems. This approach, however, can be slightly modified to handle integrating systems as well. Further, we assume the model used in the steady-state calculation is not known exactly. This leads to a robust steady-state target calculation in which the controller has the ability to rigorously account for model mismatch.

The outline of this article is as follows. First, we describe the nominal steady-state target calculation. Next, we give several uncertainty descriptions derived from process identification. We then describe the robust LP and address the numerical issues associated with implementing the algorithm in real time. Finally, we provide two illustrative simulation examples.

Nominal LP

Because most models in MPC applications are linear, with linear economics driving the controller, the steady-state target calculation commonly takes the form of a linear program (LP).

Any stable linear model, whether it be state-space, step response, impulse response, or other, can be cast at steady state in the following form

$$\Delta y = G\Delta u.$$

Here, $\Delta y \in \mathbb{R}^m$ represents the change between the current steady-state output and the open-loop value of the output and steady state, and $\Delta u \in \mathbb{R}^n$ represents the change between the current steady-state input and its open-loop value at steady state

$$\Delta u = u_s - u_{ol},$$
$$\Delta y = y_s - y_{ol}.$$

Here, y_s and u_s are vectors containing the future steady-state outputs and inputs, respectively. The vectors y_{ol} and v_{ol} are the open-loop values of y and u at steady state

$$u_{ol} = u_{k-1},$$
$$y_{ol} = f(v_{ol}).$$

The open-loop value of the input is simply its value at the last time step. The open-loop value of the output is the predicted output value at steady state when the input is held constant at its previous value. Here f represents the static model of the system.

The matrix $G \in \mathbb{R}^{m \times n}$ is the *steady-state gain matrix* for the system. The LP finds optimal steady-state targets by minimizing an economic objective

$$J_s = c^T u_s + d^T y_s, \tag{1a}$$

while maintaining the relationship between the steady-state inputs and outputs

$$\Delta y = G\Delta u, \tag{1b}$$

respecting input and output constraints arising from process specifications

$$\underline{u} \le u_s \le \bar{u} \tag{1c}$$

$$\underline{y} \le y_s \le \bar{y}, \tag{1d}$$

and ensuring the resulting solution does not force the dynamic MPC calculation to become infeasible

$$N_c \Delta \underline{u} \le \Delta u \le N_c \Delta \bar{u} \tag{1e}$$

In Eqs. 1c and 1d, $\underline{u}$ and $\bar{u}$ are minimum and maximum bounds for u_s, respectively, with similar notation employed for the outputs y_s. In Eq. 1e, N_c refers to the control horizon of the dynamic MPC calculation, and $\Delta \underline{u}$ and $\Delta \bar{u}$ are the minimum and maximum bounds for the rate of change constraints in the dynamic MPC calculation. Equation 1e ensures the steady-state target is compatible with the rate of change or velocity constraints in the dynamic MPC calculation.

To avoid real-time infeasibilities in the LP, it is common to recast the *hard* output constraints $\underline{y} \le y_s \le \bar{y}$ in Eq. 1d as *soft* constraints by adding slack variables $\bar{\epsilon} \ge 0 \in \mathbb{R}^m$, and $\underline{\epsilon} \ge 0 \in \mathbb{R}^m$ that allow for some amount of violation in the constraint

$$-\underline{\epsilon} + \underline{y} \le y_s \le \bar{y} + \bar{\epsilon}. \tag{2}$$

The sizes of the violations are minimized by appending the slacks to the objective function

$$J_s = c^T u_s + d^T y_s + \bar{\epsilon} + \underline{\epsilon}. \tag{3}$$

Additionally, a bias is introduced into the model to incorporate feedback. It is assumed that the difference between the model prediction and the measured output at the current time is due to a constant step disturbance at the output. While other types of disturbance models can be incorporated into the MPC framework (Muske and Rawlings, 1993), the constant output step disturbance assumption is standard in industrial applications (Qin and Badgwell, 1997). The model bias $b \in \mathbb{R}^m$ is based on a comparison between the current predicted output y and the current measured output $\hat{y} \in \mathbb{R}^m$,

$$b = \hat{y} - y. \tag{4}$$

The bias is added to the model

$$\Delta y = G\Delta u + b. \tag{5}$$

The nominal steady-state target calculation is then

$$\min_{u_s, y_s, \epsilon} c^T u_s + d^T y_s + \bar{\epsilon} + \underline{\epsilon}$$

subject to

$$\begin{aligned} y_s &= y_{ol} + G(u_s - u_{ol}) + b \\ \underline{u} &\le u_s \le \bar{u} \\ N_c \Delta \underline{u} &\le \Delta u \le N_c \Delta \bar{u} \\ -\underline{\epsilon} + \underline{y} &\le y_s \le \bar{y} + \bar{\epsilon} \\ 0 &\le \bar{\epsilon} \\ 0 &\le \underline{\epsilon} \end{aligned} \tag{6}$$

It is common to calculate the moves Δu and Δy, instead of the inputs and outputs directly. The nominal steady-state target calculation can be expressed in velocity form as

$$\min_{\Delta u, \Delta y, \epsilon} c^T \Delta u + d^T \Delta y + e^T \epsilon$$

subject to

$$\begin{aligned} \Delta y &= G\Delta u + b \\ A_u \Delta u &\le b_u \\ A_y \Delta y &\le b_y + \epsilon \\ \epsilon &\ge 0 \end{aligned} \tag{7}$$

where $e \in \mathbb{R}^{2m}$ is a vector which penalizes output deviations. For our discussion, we assume without loss of generality that

$$e = [1\,1 \ldots 1]^T;$$

however, e may be arbitrarily large in practice. The vector ϵ is comprised of stacking the upper and lower slack vectors

$$\epsilon = \left[\bar{\epsilon}^T \underline{\epsilon}^T\right]^T \tag{8a}$$

and $A_u \in \mathbb{R}^{4n \times n}$ and $b_u \in \mathbb{R}^{4n}$ are given by

$$A_u = \begin{bmatrix} I \\ -I \\ I \\ -I \end{bmatrix} \quad b_u = \begin{pmatrix} \bar{u} - u_{ol} \\ -\underline{u} + u_{ol} \\ N_c \Delta \bar{u} \\ -N_c \Delta \underline{u} \end{pmatrix}, \tag{8b}$$

while $A_y \in \mathbb{R}^{2m \times m}$ and $b_y \in \mathbb{R}^{2m}$ are given by

$$A_y = \begin{bmatrix} I \\ -I \end{bmatrix} \quad b_y = \begin{pmatrix} \bar{y} - y_{ol} \\ -\underline{y} + y_{ol} \end{pmatrix}. \tag{8c}$$

Additionally, since Δy depends linearly upon Δu, the entire problem can be expressed in terms of Δu and ϵ, reducing the number of decision variables. The resulting LP can be cast in standard form and passed to an optimization algorithm such as the simplex method (Chvátal, 1983).

Uncertainty Descriptions

Our objective is to develop a robust LP that explicitly accounts for uncertainty in the steady-state gain matrix G of the nominal steady-state target calculation. The mathematical parameterization of possible plants is known as the *uncertainty description.* It can take many different forms and can be given parametrically or statistically.

Ellipsoid uncertainty

In model identification, if the process noise and model parameters are assumed to be normally distributed or Gaussian variables, the natural uncertainty description is an ellipsoidal bound on the parameters (Bard, 1974). Illustrated in Figure 2, the parameter vector $\theta = [\theta_1 \ldots \theta_{n_\theta}]^T \in \mathbb{R}^{n_\theta}$ is assumed to lie in the set

$$\theta \epsilon \Theta \stackrel{\text{def}}{=} \left\{\theta : (\theta - \theta_c)^T W (\theta - \theta_c) \le 1\right\}, \tag{9}$$

describing the joint confidence region. The center of the ellipse θ_c is the mean of the normal distribution, and the symmetric positive-definite matrix W defines the size and orientation of the ellipsoid. In particular, the square roots of the reciprocals of the eigenvalues of W are the lengths of the semi-axes of the ellipsoid, and the eigenvectors of W give the directions of the semi-axes.

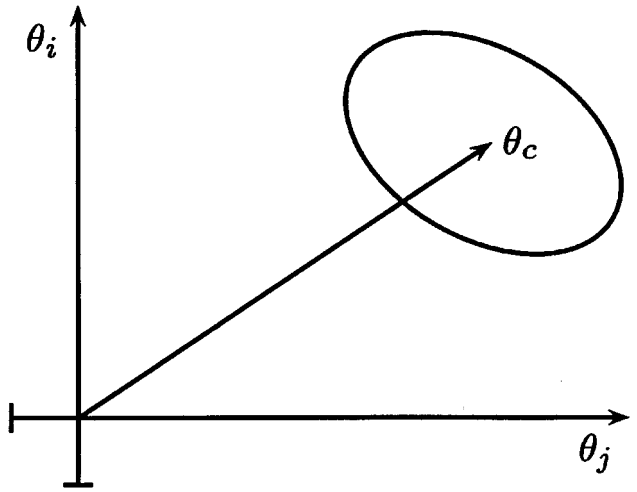

Figure 2. Elliptic parameter uncertainty.

The confidence region defined by the ellipsoid in Eq. 9 can be parameterized by a vector s that sweeps out a unit sphere

$$\theta \in \Theta \stackrel{\text{def}}{=} \left\{\theta_c + \beta V^{1/2} s : \| s \| \le 1\right\}. \tag{10}$$

Here, we have replaced the matrix W with its statistical interpretation

$$W = V^{-1}/\beta^2$$

The matrix V is the covariance or estimate of the covariance of the estimated parameters. The term β is related to the radius of the ellipse and is given by

$$\beta = \Phi^{-1}(1 - \alpha)$$

where $\Phi(x)$ is the univariate normal distribution function with zero mean and unit variance

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-1/2 s^2} ds.$$

and $1 - \alpha$ is the confidence or probability level. We can always guarantee a value of β to exist for any probability level $1 - \alpha$, even when V is only an estimate of the true covariance (Bard, 1974). For Eq. 10 to remain convex, $\alpha \le 0.5$, or equivalently, $\beta \ge 0$.

The uncertainty description given by Eq. 10 is more general than Eq. 9 as it allows V to be rank deficient. When the covariance matrix V does not have full rank, the ellipsoid becomes degenerate, collapsing in specific directions corresponding to parameters that are known exactly. The directions in which the ellipse collapses correspond to the eigenvectors (or semi-axes) with associated eigenvalues that are zero (that is, those directions for which a nonzero z causes $V^{1/2} z$ to be zero).

A more formal discussion of ellipsoid uncertainty is presented by Ben-Tal and Nemirovski (1996), who describe ellip-

soid bounds in the general context of convex programming, and by Boyd et al. (1998) who motivate the use of ellipsoid uncertainty descriptions for use in optimal control.

In the steady-state target calculation, the uncertain parameters θ are the elements of the steady-state gain matrix G. Let g_i be the column vector describing the i^{th} row of G

$$G = [\, g_1\ g_2 \cdots g_m \,]^T. \tag{11}$$

Assume for the moment that the outputs are independent random variables (no output is correlated to another). In this case, we can construct a block diagonal covariance matrix for the process gains made up of the individual covariance matrices for each row of the gain matrix. In particular, if V_i is the covariance matrix corresponding to the ith output, then the covariance for the entire matrix is given by

$$V = \begin{bmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_m \end{bmatrix}. \tag{12}$$

If the outputs are actually cross-correlated, then the off-diagonal elements of V will be nonzero. In any case, we can assume without loss of generality that g is nominally $\tilde{g}$ and has covariance V, yielding

$$g \in \varepsilon \stackrel{\text{def}}{=} \left\{ \tilde{g} + \beta V^{1/2} s : \| s \| \le 1 \right\}, \tag{13}$$

where ε defines the ellipsoid of the entire matrix. This is the most natural way to pose the problem as it results in *constraint-wise* uncertainty. In the nominal LP, the gain matrix G is premultiplied by the (possibly dense) matrix A_y in the output constraint. This results in a linear combination of outputs. The ith component of the output constraint is given by

$$\sum_j a_{ij} g_i^T \Delta u \le b_{y_i}.$$

In the general case, we require ellipsoids of the type above to capture the uncertainty in the constraint. If, however, most of the elements of A_y are zero (Eq. 8c), as is the case for simple bounds on the outputs, it is possible to simplify the elliptic uncertainty description.

For simple bounds, there is a constant scalar factor γ premultiplying the ith constraint

$$\gamma g_i^T \Delta u \le b_{y_i}.$$

Each component of the constraint depends only upon a single row of the gain matrix. Thus, we can define an ellipsoid for only a single row. Let the gain for the ith row nominally be $\tilde{g}_i$. The covariance is V_i. The ellipsoid is given by

$$g_i \in \varepsilon_i \stackrel{\text{def}}{=} \left\{ \tilde{g}_i + \beta V_i^{1/2} s : \| s \| \le 1 \right\}. \tag{14}$$

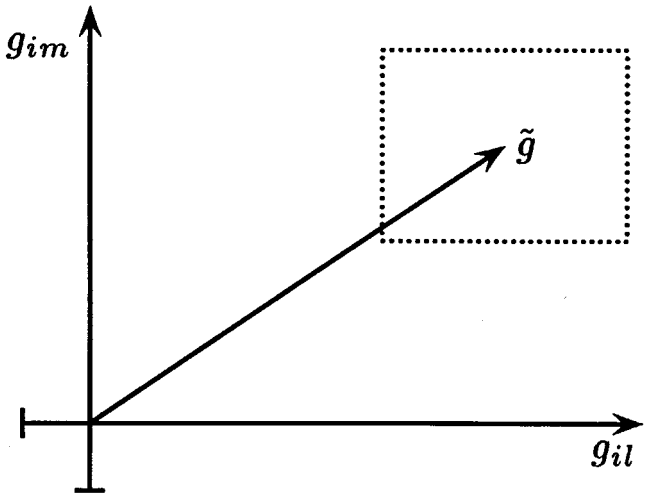

Figure 3. Box uncertainty.

Additionally, there are instances when it is desirable to approximate the ellipsoidal uncertainty as either a polytope or simple box.

Box uncertainty

If the ellipsoidal constraint can be interpreted in terms of a joint confidence region, then the box constraint can be interpreted in terms of joint confidence intervals (or as approximations to the ellipse). Simple box constraints, illustrated in Figure 3, give the minimum and maximum bounds of the ellipse. More complicated polytopes may also be used in an attempt to approximate the ellipse. A general convex polytope, including box constraints, can be expressed as bounds on some linear combination of the elements g_{ij} of G. The uncertainty description $\mathfrak{B}$ is then given by

$$g \in \mathfrak{B} \stackrel{\text{def}}{=} \left\{ g : A_g g \le b_g \right\}. \tag{15}$$

Box constraints, when used to approximate elliptic constraints as explained above, are more conservative than ellipsoidal constraints and, in turn, lead to a more conservative control action. Through Eqs. 13, 14, and 15, we have described three potential uncertainty descriptions $\mathfrak{U}$ for use in the robust LP.

Robust LP

The robust LP explicitly accounts for uncertainty in the steady-state gain matrix G of the nominal algorithm. Because the solution of the LP is always guaranteed to lie on the boundary of the feasible set, uncertainty in the gain matrix has a direct consequence on the steady-targets passed to the dynamic MPC calculation. The feasible set $\tilde{\mathfrak{F}}$ of the nominal LP, ignoring slacks and bias for the moment, is

$$\Delta u \in \tilde{\mathfrak{F}} = \left\{ \Delta u : \begin{array}{l} A_u \Delta u \le b_u \\ A_y \Delta y = A_y G \Delta u \le b_y \end{array} \right\} \tag{16}$$

The uncertainty in G results in an uncertainty about the location of the output constraint, and as a result the optimal targets. To guarantee that the output constraint is feasible for any realizable value of the gain, we must modify the feasible set $\tilde{\mathfrak{F}}$

$$\Delta u \in \mathfrak{F} = \left\{ \Delta u: \begin{array}{l} A_u \Delta u \le b_u \\ A_y \Delta y = A_y G \Delta u \le b_y, \forall G \in \mathfrak{U} \end{array} \right\} \quad (17)$$

We require the output constraint to hold for all possible values of the gain. The difference between the nominal feasible set $\tilde{\mathfrak{F}}$, and the robust feasible set $\mathfrak{F}$ is illustrated in Figure 4 for a simple 2-by-1 example with ellipsoid uncertainty. Although the constraints become nonlinear, the set $\mathfrak{F}$ remains convex.

Output constraints in the nominal problem can correspond to critical variables—the temperature limit on a reactor vessel or the maximum allowable feed rate into a unit. Uncertainty or *fuzziness* in the constraint means that even though the nominal value may be binding at the upper or lower limit, the actual value may be outside the constraint region. The constraint set $\mathfrak{F}$ restricts inputs to only those which guarantee that the output constraint will not be violated.

Model uncertainty forces the problem structure to change. The linear constraints so commonly used in control theory become nonlinear. In order to ensure that output constraints are enforced for all possible gains, the nominal LP must be replaced with the following semi-infinite program

$$\min_{\Delta u, \epsilon} c^T \Delta \underline{u} + d^T \tilde{G} \Delta u + e^T \epsilon$$

subject to

$$\begin{aligned} A_y G \Delta u &\le b_y + \epsilon - A_y b, \forall G \in \mathfrak{U} \quad (18) \\ A_u \Delta u &\le b_u \\ \epsilon &\ge 0 \end{aligned}$$

which we refer to as the *robust LP*. It must be solved at every time-step during the execution of a standard MPC algorithm. Solution times must be on the order of tens of seconds for it to be used in a real-time setting. We have chosen to cast it in terms of only Δu and ϵ, having removed the problem dependency on Δy. This was done for simplicity and to illustrate the structure of the problem; it can just as well be written so that the outputs appear explicitly as an equality constraint. In the objective of the LP, we minimize the output corresponding to the nominal gain $\tilde{G}$. The objective is to drive the process toward the economic optimum using the best guess for the gain of the process, but to restrain the inputs to ensure the constraints will not be violated for any realizable value of the gain (those values captured by the uncertainty description). The next section discusses how to formulate Eq. 18 as an optimization problem over a second-order cone.

Problem formulation

To illustrate the points of this section, we need only consider a simplified version of Eq. 18

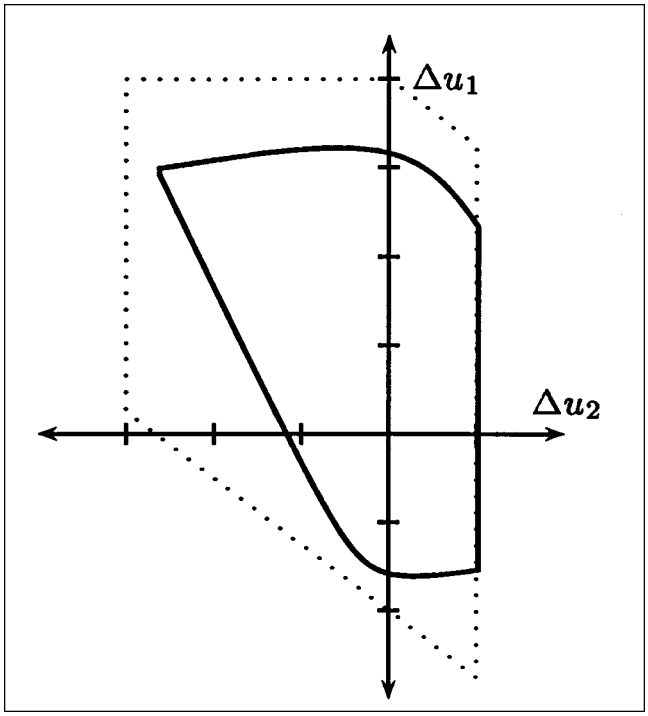

Figure 4. Feasible regions for the robust LP at a given time step.

····: $\tilde{\mathfrak{F}}$ for nominal problem; —— $\mathfrak{F}$ for elliptic uncertainty.

$$\begin{aligned} &\min_x c^T x \\ &\text{subject to} \quad (19) \\ &AGx \le b, \forall G \in \mathfrak{U}. \end{aligned}$$

This problem falls into a more general class of problems known as *semi-infinite programs*. Consider the following general semi-infinite program

$$\begin{aligned} &\min_x f(x) \\ &h(x, \theta) \le 0 \quad \text{for all} \quad \theta \in \mathfrak{U}, \quad (20) \end{aligned}$$

where $f: \mathbb{R}^n \to \mathbb{R}$ and $h: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. The vector θ is generally a set of parameters and, for most engineering problems, $\mathfrak{U}$ is closed and convex. The vector x is finite while the set $\mathfrak{U}$ is infinite. A finite number of variables appear in infinitely many constraints. Equation 20 occurs in many different fields ranging from robotics to structural design. Hettich and Kortanek (1993) provide a review of various areas to which semi-infinite programming has been applied, as well as current theory and solution methods.

The most straightforward way to handle constraints of the type in Eq. 20 is to note that

$$h(x, \theta) \le 0 \quad \forall \theta \in \mathfrak{U} \quad (21)$$

is equivalent to

$$\max_\theta \left[h(x, \theta), \theta \in \mathfrak{U} \right] \le 0.$$

If we define

$$H(x) \stackrel{\text{def}}{=} \max_{\theta} h(x, \theta),\ \theta \in \mathfrak{U}, \tag{22}$$

then the semi-infinite program (Eq. 20) becomes

$$\min_{x} f(x)$$
$$H(x) \le 0 \tag{23}$$

under certain regularity assumptions (Hettich and Jongen, 1977).

For most engineering problem, $H(x)$ is not differentiable with respect to x. The equivalence between Eq. 20 and Eq. 23 arises out of the fact that locally the two problems are equivalent, while globally they have very different properties. The review by Polak (1987) discusses this problem class in the framework of engineering design and treats $H(x)$ as a general nondifferentiable function. The review by Hettich and Kortanek (1993), on the other hand, treats $H(x)$ as a locally differentiable function in the framework of semi-infinite programming. Optimization techniques must be tailored for each problem class.

Many robust optimization problems can be cast as semi-infinite optimization problems. For some classes of problems the constraint (Eq. 21) is interpreted in a stochastic framework. A common goal is to satisfy the constraint in a probabilistic sense

$$\text{prob}\left[h(x; \theta) \le 0\right] \ge 1 - \alpha, \tag{24}$$

with $\theta \in \mathbb{R}^m$ a randomly distributed variable with a given mean and variance. This classical way of dealing with uncertainty as a probabilistic constraint is known as *stochastic programming*. In fact, many robust optimization problems first arose as stochastic optimization problems. Stochastic programming is commonly used in the fields of operations research and managerial science, whereas semi-infinite programming is applied more often to engineering problems.

These ideas have recently been applied to model predictive control. Schwarm and Nikolaou (1997) have used stochastic programming as a way to address uncertainty in output constraint satisfaction for dynamic MPC. They consider an impulse response model with coefficients having a given mean and variance. They pose the standard output constraint as a probabilistic constraint which in turn is cast as an equivalent deterministic constraint. The optimization is then solved using nonlinear programming methods. The deterministic constraint they pose is actually a second-order cone. The techniques we discuss in the section on Numerical Implementation for the robust LP can be used just as effectively on the dynamic portion of MPC with probabilistic output constraints. Instead of using general nonlinear programming methods, it is possible to use recently developed interior-point methods to solve this optimization problem.

Schwarm and Nikolaou (1998) also extend this approach to a stochastic program with recourse. Instead of simply asking that the constraints hold to a given probability, they minimize the expectation that the constraint will be violated. These approaches for addressing output constraint satisfaction show great promise.

While there are various optimal control problems to which semi-infinite programming has been applied (Reemtsen and Rückmann, 1998), only recently has the use of semi-infinite programming been applied to model predictive control. In robust model predictive control, the objective is to guarantee closed-loop stability in the presence of model mismatch. Badgwell (1997) has shown that forcing the cost function of the dynamic calculation to form a nonincreasing sequence for all plants in an uncertainty description produces a robustly stabilizing algorithm. If the uncertainty description is continuous, this approach results in a semi-infinite constraint (Ralhan, 1998). For set uncertainty, the optimization takes the form of a semi-definite program (Kassmann and Badgwell, 1997).

The ideas of both stochastic programming and semi-infinite programming share a common thread for linear problems. This is especially the case for the robust LP. For a more detailed discussion of stochastic programming, refer to Prékopa (1995) and Kall and Wallace (1994). The text by Hettich and Zencke (1982), although in German, and the text by Reemtsen and Rückmann (1998) are good sources of information on semi-infinite programming.

Before discussing the case of ellipsoid uncertainty in the robust LP, let us consider the simpler case when the uncertainty description is given by a convex, bounded polytope or box constraints. If the objective of a semi-infinite optimization is to minimize a linear function subject to a set of linear constraints when there is uncertainty in the problem data, the semi-infinite LP can be cast as a standard LP with more constraints (Dantzig, 1963). If we assume the uncertain parameter θ corresponds to the elements of the matrix A and the goal is to

$$\min_{x} c^T x$$
subject to
$$Ax \le b \quad \forall A \in \mathfrak{B},$$

where $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$, then a straightforward way to solve the problem is to simply enumerate all the possible values of A. The reason that is a valid approach relies on the fact that the solution of an LP will always lie on the boundary of the feasible set. So, by enumerating all the possible values, the semi-infinite problem can be replaced by a finite problem with an increased number of inequality constraints. For every row $i = 1, \ldots, m$ of A, there are 2^n possible permutations of the entries, created by replacing the n elements with their upper and lower bounds, respectively. Doing so replaces the previous problem with the following problem having $m\ 2^n$ constraints

$$\min_{x} c^T x$$
subject to
$$A^* x \le b,$$

where A^* is the matrix created by enumerating all the possible combinations of A. The problem size therefore grows exponentially. For a 10-by-10 system with simple bounds on the

outputs, this corresponds to over 20,000 constraints that would need to be added to the nominal LP. For real-time applications such as control, this may be unacceptable since constraints need to be dropped or added as sensors fail or transmitters go off-line.

If the simplex method is used to solve the problem, it can be expected to slow down as it is forced to consider a larger and larger number of vertices. Other methods, such as interior-point methods, will likely produce better solution times as the number of inequalities increases. There is a limit, however, to the usefulness of this approach because for large semi-infinite problems, this will produce enormous equivalent finite problems. In this case, using ellipsoid uncertainty, in addition to possibly providing less of a performance penalty, may result in shorter solution times.

When the semi-infinite problem (Eq. 20) is convex with the constraints appearing as cones, Ben-Tal and Nemirovski (1994, 1996) have shown that the problem can be recast as a finite dimensional convex optimization. This is based in part on the work by Nesterov and Nemirovski (1994) who introduced the idea of a second-order cone representable function (in other words, a function that can be cast as a second-order cone). They showed that a semi-infinite optimization with a second-order cone representable constraint can be better interpreted as an optimization over a second-order cone.

Second-order cone programming

A general second-order cone program (SOCP) has the following form

$$\begin{aligned} &\min_x f^T x \\ &\quad \| A_i x + b_i \| \le c_i^T x + d_i, \quad i = 1, \ldots, N \\ &\quad Gx = g, \end{aligned} \tag{25}$$

where $x \in \mathbb{R}^n$, $b_i \in \mathbb{R}^m$, and $g \in \mathbb{R}^p$. $\| \cdot \|$ is the standard Euclidean norm; and A_i, c, c_i, d_i, and G are of appropriate dimension. A wide variety of nonlinear convex optimization problems can be cast as SOCPs (Lobo et al., 1998), including linear and quadratic programs as special cases. The robust LP falls into this category.

The two principal advantages to casting the robust LP as a SOCP are:

- First and foremost, the original semi-infinite optimization is recast in the form of a standard optimization problem with a finite dimensional constraint.
- Secondly, there has been tremendous activity in extending primal-dual interior-point methods to this more general problem class—resulting in new efficient solution methods.

The primary reference for interior-point methods for second-order cone programming is the text by Nesterov and Nemirovski (1994). Boyd, Crusius, and Hansson (1998) show how SOCPs can be used in optimal control. They describe a robust optimal control problem in which the ℓ_∞ norm of the cost function (that is, the peak tracking error) is minimized for uncertain impulse response coefficients in a finite impulse response model. They show that the problem can be solved efficiently as a SOCP. They also illustrate how SOCPs can be used in the optimal design of feedback controllers.

These applications are based on the fact that an uncertain linear constraint can be cast as a second-order cone. Ben-Tal and Nemirovski (1994) consider the general linear program

$$\begin{aligned} &\min_x c^T x \\ &\text{subject to} \\ &a_i^T x \le b_i \quad \forall a_i \in \varepsilon_i,\ i = 1, \ldots, m, \end{aligned}$$

with uncertainty in the data a_i described by an ellipsoid

$$a_i \in \varepsilon_i \stackrel{\text{def}}{=} \left\{ \tilde{a}_i + \beta V_1^{1/2} s : \| s \| \le 1 \right\}.$$

From semi-infinite programming, the constraint above is equivalent to a maximization over the ellipsoid

$$\max\left\{ a_i^T x : a_i \in \varepsilon_i \right\} = \tilde{a}_i^T x + \beta \| V_i^{1/2} x \| \le b_i, \tag{26}$$

which is nothing other than a second-order cone. The result is the following robust LP

$$\begin{aligned} &\min_x c^T x \\ &\text{subject to} \\ &\tilde{a}_i^T x + \beta \| V_i^{1/2} x \| \le b_i \end{aligned} \tag{27}$$

for $i = 1, \ldots, m$.

It has been known for some time in stochastic programming that a probabilistic random linear constraint (that is, Eq. 24 for linear $h(x, \theta)$ and a normal distribution) can be cast as an equivalent nonlinear constraint (see, for example, §10.4 of Whittle (1971); §10.3 of Prékopa, 1995). However, the nonlinear equivalent was not recognized as a second-order cone. Lobo et al. (1998) in their summary of SOCP applications make this connection explicit.

Consider the linear probabilistic constraint

$$\text{prob}\left[a_i^T x \le b_d \right] \ge 1 - \alpha.$$

If a_i has mean $\tilde{a}_i$ and covariance V_i, then $a_i^T x$ has mean $\tilde{a}_i^T$ and variance $x^T V_i x$. The constraint can be written as an equivalent constraint with zero mean and unit variance

$$\text{prob}\left[\frac{a_i^T x - \tilde{a}_i^T x}{\sqrt{x^T V_i x}} \le \frac{b_i - \tilde{a}_i^T x}{\sqrt{x^T V_i x}} \right] \ge 1 - \alpha.$$

Thus, the probability can be given by

$$\Phi\left(\frac{b_i - \tilde{a}_i^T x}{\sqrt{x^T V_i x}} \right) \ge 1 - \alpha,$$

or, equivalently

$$\tilde{a}_i^T x + \Phi^{-1}(1 - \alpha) \| V^{1/2} x \| \le b_i,$$

which is a second-order cone constraint. While this is a special case of the general probabilistic constraint, it is still very important. A general probabilistic constraint (Eq. 24) is very computationally expensive since it involves a semi-infinite, multivariate probability integral to evaluate the associated probability distribution function. A second-order cone, however, can be evaluated quite efficiently.

We can now interpret the robust steady-state target calculation probabilistically. The semi-infinite constraint in

$$\min_x c^T x$$
$$\text{subject to}$$
$$AGx \le b, \quad \forall G \in \mathfrak{U},$$

can be thought of as requiring the output constraints to hold for some uncertainty in the process gains to a probability $1-\alpha$.

To see this explicitly, we rewrite the semi-infinite constraint component-wise

$$\sum_j a_{ij} g_j^T x \le b_i \qquad G \in \mathfrak{U}, \quad i = 1, \ldots, m. \tag{28}$$

which can be rewritten as

$$g^T (D_i x) \le b_i \quad \forall g \in \varepsilon$$

where D_i is defined by

$$D_i = \left[\text{diag}(a_{i1}\, e)\, \text{diag}(a_{i2}\, e) \ldots \text{diag}(a_{im}\, e)\right]^T,$$

and g is the vector of the rows of G stacked lengthwise. The vector e contains all ones. We have taken the uncertainty description $\mathfrak{U}$ to be the ellipsoidal uncertainty description ε. From Eq. 26, the constraint becomes

$$\tilde{g}^T (D_i x) + \beta \| V^{1/2} D_i x \| \le b_i. \tag{29}$$

which is a second-order cone.

The result is the following problem

$$\min_x c^T x$$
$$\text{subject to}$$
$$\tilde{g}^T (D_i x) + \beta \| V^{1/2} D_i x \| \le b_i$$

for $i = 1, \ldots, m$. We can now cast the full robust steady-state target calculation in model predictive control (Eq. 18) as a SOCP

$$\begin{aligned} &\min_{\Delta u, \epsilon} c^T \Delta u + d^T \tilde{G} \Delta u + e^T \epsilon \\ &\text{subject to} \\ &\tilde{g}^T (D_i \Delta u) + \beta \| V^{1/2} D_i \Delta u \| + a_i^T b \le b_{y_i} + \epsilon_i \\ &A_u \Delta u \le b_u \\ &\epsilon \ge 0. \end{aligned} \tag{30}$$

for $i = 1, \ldots, m$. This optimization problem must be solved at every controller execution.

Dynamic algorithm

In the examples section, we will send the targets calculated by the nominal and robust LP to a dynamic controller. In this section we briefly explain the dynamic control algorithm that we will use. The algorithm is a variation of the quadratic dynamic matrix control (QDMC) (Garcí and Morshedi, 1986) algorithm with an extra 'sum of moves' constraint and soft output constraints. The quadratic program associated with QDMC is

$$\begin{aligned} &\min_{\delta u, \epsilon} \delta u^T H \delta u - g^T \delta u \\ &\text{subject to} \\ &\underline{u} - u_{k-1} \le L \delta u \le \bar{u} - u_{k-1} \\ &\underline{y} - y^p \le A \delta u \le \underline{y} - y^p \end{aligned} \tag{31}$$

where $\delta u \in \mathbb{R}^{nc}$ is the future control move, n is the number of inputs, and c is the control horizon. The Hessian H and gradient g are given by

$$H = A^T \Gamma^T \Gamma A + \Lambda^T \Lambda \quad \text{and} \quad g = A^T \Lambda^T \Lambda \hat{e}.$$

A is the *dynamic matrix;* $\Lambda = w_y I$ is the matrix of output error weights; and $\Gamma = w_u I$ is the matrix of input movement weights or move suppression weights. The vectors $\bar{u}$ and $\underline{u}$ are the upper and lower input bounds, respectively. Likewise, $\bar{y}$ and $\underline{y}$ are the upper and lower bounds on the outputs. The vector y^p is the future output prediction and u_{k-1} is the input at the previous sample time. The vector $\hat{e}$ contains the open-loop future errors for the controlled variables.

In our formulation, we use soft output constraints and append a constraint that forces the sum of moves calculated by the algorithm to be equal to the total move target calculated by the steady-state algorithm

$$W \delta u = \Delta u_{\text{ref}}$$

where W is a matrix that sums the individual moves. This, however, can cause problems. The addition of this end-point constraint causes the QP to become very ill-conditioned. It is analogous to the problems caused by end-point constraints in state-space MPC algorithms. Instead, we satisfy the constraint in a least-squares sense and append it to the objective

$$\min \delta u^T H \delta u - g^T \delta u + (W \delta u - \Delta u_{\text{ref}})^T \Omega (W \delta u - \Delta u_{\text{ref}}).$$

The modified QP that we solve is then

$$\begin{aligned} &\min_{\delta u, \epsilon} \delta u^T \hat{H} \delta u + \epsilon^T \Sigma \epsilon - \hat{g}^T \delta u \\ &\text{subject to} \\ &\underline{u} - u_{k-1} \le \quad L \delta u \quad \le \underline{u} - u_{k-1} \\ &\underline{y} - y^p - \underline{\epsilon} \le \quad A \delta u \quad \le \bar{y} - y^p + \bar{\epsilon} \end{aligned} \tag{32}$$

where $\epsilon = (\underline{\epsilon}, \bar{\epsilon})$ and the new Hessian $\hat{H}$ and gradient $\hat{g}$ are given by

$$\hat{H} = A^T \Gamma^T \Gamma A + \Lambda^T \Lambda + W^T \Omega W \text{ and } \hat{g} = A^T \Lambda^T \Lambda \hat{e} - W^T \Omega \Delta u_{\text{ref}}$$

Numerical Implementation

In this section we introduce the main ideas behind primal-dual interior-point methods for second-order cone programming. This is by no means an exhaustive summary or discussion; our objective is to provide a more complete picture of the robust LP and to facilitate the use of interior-point methods as a tool in model predictive control. For a broader perspective, see the review by Wright (1998).

The most promising algorithms for solving second-order cone programs are primal-dual interior-point methods. The goal in any interior-point method is to reduce the objective function while keeping the iterates strictly feasible with respect to the inequality constraints, and in the limit of a solution satisfy the equality constraints as well. The application of these methods has been extended from their original use in solving LPs to numerous other optimization paradigms such as quadratic and semi-definite programming.

One of the reasons why researchers are focusing more attention on second-order cone programming is that any quadratically constrained quadratic program can be recast as a SOCP. In fact, linear programming, quadratic programming, second-order cone programming, and semi-definite programming are all special cases of optimizations over symmetric or self-scaled cones (Nesterov and Nemirovski, 1994; Nesterov and Todd, 1997; Nesterov and Todd, 1998). Many engineering and control applications can be cast as semi-definite programs (Vandenberghe and Boyd, 1996; Wu et al., 1996) or second-order cone programs (Boyd et al., 1998). This attention has led to the development of new optimization algorithms. The duality theory for these problems gives rise to a general complementarity condition. That condition combined with primal and dual feasibility form a square nonlinear system of equations which in principle determine the optimal solution.

Other authors have also proposed using interior-point methods specifically for model predictive control. Wright (1997) showed that the quadratic program associated with the dynamic portion of MPC can be interpreted as a monotone linear complementary problem (mLCP) and solved using interior point methods. Rao et al. (1998) showed that the problem structure of the mLCP can be exploited yielding a matrix factorization with a cost that grows linearly instead of cubicly with horizon length. Albuquerque et al. (1997) have shown that these methods can outperform active-set methods for both control and simulation applications.

In order to explain how the primal-dual interior-point methodology is used, consider the following SOCP

$$\begin{aligned} &\min_{\hat{x}} f^T \hat{x} \\ &\text{subject to} \\ &\| A_i \hat{x} + b_i \| \le c_i^T \hat{x} + d_i, \quad i = 1, \ldots, n \\ &G\hat{x} = g \end{aligned} \tag{33}$$

The first step is to recast Eq. 33 in standard form

$$\begin{aligned} &\min_{x} c^T x \\ &\text{subject to} \\ &Ax = b \\ &x \ge_{\kappa} 0 \end{aligned} \tag{34}$$

where the optimization has been cast in terms of the new decision variable $x \in \mathbb{R}^K$ defined as

$$x = \begin{pmatrix} x_1^T & \ldots & x_n^T \end{pmatrix}^T$$

with

$$x_i = \begin{pmatrix} x_{i0} \\ x_{i1} \end{pmatrix} = \begin{pmatrix} c_i^T \\ A_i \end{pmatrix} \hat{x} + \begin{pmatrix} d_i \\ b_i \end{pmatrix}$$

and $x_{i0} \in \mathbb{R}$ and $x_{i1} \in \mathbb{R}^n$. The data of the problem are given in terms of the original data. Let

$$R = \begin{pmatrix} c_1 & A_1^T & c_2 & A_2^T & \ldots & c_n & A_n^T \end{pmatrix}^T$$

and

$$r = \begin{pmatrix} d_1 & b_1^T & d_2^T b_2 & \ldots & d_n^T b_n \end{pmatrix}^T$$

We assume R^{-1} exists, then

$$\begin{aligned} c &= R^{-1} f, \\ b &= g - R^{-1} r, \\ A &= G R^{-1}. \end{aligned}$$

In the case that the columns of R are not linearly independent, then either the dual problem is not feasible, or it can be reduced to another problem with fewer variables. If the dual is not feasible, the primal problem will be unbounded.

The notation $x \ge_{\kappa} 0$ means $x \in \mathcal{K}$ where $\mathcal{K}$ is the Cartesian product of second-order cones

$$\mathcal{K} = \mathcal{K}_1 \times \ldots \times \mathcal{K}_n.$$

The ith second-order symmetric cone is given by

$$\mathcal{K}_i = \left\{ x_i + (x_{i0}, x_{i1}) \mid x_{i0}^2 - x_{i1}^T x_{i1} \ge 0, x_{i0} \ge 0 \right\}$$

The dual of Eq. 34 is

$$\begin{aligned} &\max_{\underline{y}, \underline{z}} b^T y \\ &\text{subject to} \\ &A^T y + z = c \\ &z \ge_{\kappa^*} 0 \end{aligned} \tag{35}$$

where $z \in \mathbb{R}^{\mathcal{K}}$ and $y \in \mathbb{R}^{\mathcal{K}}$ are the dual variables or Lagrange multipliers for the problem. The vector z possesses a structure similar to that of x.

$$z = \begin{pmatrix} z_1^T \dots z_n^T \end{pmatrix}^T$$

with

$$z_i = \begin{pmatrix} z_{i0} \\ z_{i1} \end{pmatrix}$$

and $z_{i0} \in \mathbb{R}$ and $z_{i1} \in \mathbb{R}^n$. The notation $z \geq_{\kappa^*} 0$ means $z \in \mathcal{K}^*$ where $\mathcal{K}^*$ is the dual cone defined by

$$\mathcal{K}^* = \mathcal{K}_1^* \times \dots \times \mathcal{K}_n^*.$$

with

$$\mathcal{K}_i^* = \left\{ z_i = (z_{i0}, z_{i1}) | z_{i0}^2 - z_{i1}^T z_{i1} \geq 0, z_{i0} \geq 0 \right\}.$$

While this notation may at first seem burdensome, it casts the SOCP (and, in fact, any optimization over a self-scaled cone) in a framework analogous to linear programming. This is more than just a convenient change of variables. It helps to extend the theory associated with linear programming to a much larger class of problems of engineering significance.

We now define the duality gap μ; it plays a central role in interior point algorithms. It is defined to be the difference between the primal and dual objective functions

$$\mu = c^T x - b^T y = c^T x - (Ax)^T y = x^T z. \tag{36}$$

It can be regarded in a sense as a measure of the distance from optimality. The primal problem is said to be strictly feasible if there exists a point x for which the inequality constraints of the primal problem hold with strict inequality: $x >_{\kappa} 0$. Likewise, the dual problem is said to be strictly feasible if there exists a point z for which the inequality constraints of the dual problem hold with strict inequality: $z >_{\kappa^*} 0$.

Nesterov and Nemirovski (1994) showed that the duality gap is zero at optimality if the primal and dual problems for a second-order cone program are strictly feasible. The proof for this is given in their text. It is well known that this also holds true for any optimization over self-scaled cones. This fact motivated the development of so-called primal-dual interior-point methods. The primary goal of any interior-point method is to reach optimality by driving the duality gap to zero while satisfying primal and dual feasibility, as well as complementarity (sometimes called the complementarity slackness condition).

The complementarity condition for the SOCP is given by

$$x \circ z = 0,$$

where the binary operation o is defined as follows

$$x \circ z \overset{\text{def}}{=} \begin{pmatrix} x^T z \\ x_{i0} z_{i1} + x_{i0} z_{i1} \end{pmatrix}.$$

for $x \in \mathbb{R}^{\mathcal{K}}$ and $z \in \mathbb{R}^{\mathcal{K}}$. The binary operation forms what is known as a Euclidean Jordan algebra (Faybusovich, 1997a,b, 1995; Faraut and Koráyi, 1994) or, more generally, an associative algebra (Monteiro and Tsuchiya, 1998) with close connections to interior-point algorithms. The algebra can be viewed as the result of partitioning the vector x_i into a scalar component x_{i0} and a vector component x_{i1}. Associated with an element $x \in \mathbb{R}^{\mathcal{K}}$ is the matrix mat(x) defined to be diag$[X_1, \dots X_n]$ with

$$X_i = \begin{pmatrix} x_{i0} & x_{i1}^T \\ x_{i1} & x_{i0} I \end{pmatrix}.$$

We can now see that

$$x \circ z = \text{mat}(x) z = \text{mat}(z) x.$$

This allows us to write Karush-Kuhn-Tucker (KKT) conditions for the primal and dual problems as follows

$$F(x, y, z) = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ Xz \end{pmatrix} = 0, \quad (x, z) \geq_{\kappa} 0. \tag{37}$$

where we have denoted $X = \text{mat}(x)$. Applying any Newton type method on Eq. 37 yields the following linear system

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} b - Ax \\ c - A^T y - z \\ Xz \end{pmatrix}. \tag{38}$$

where $Z = \text{mat}(z)$.

The nonlinear complementarity condition of Eq. 37, $Xz = 0$, is generally relaxed in some specific way to keep the iterates away from the boundary and inside the second-order cone. The specific way in which complementarity is relaxed produces different search directions and thus different algorithms. A large class of these algorithms has been shown to be solvable in polynomial time (Schmieta and Alizadeh, 1998; Monteiro and Tsuchiya, 1998), that is, the effort required to solve the problem to a given tolerance can be represented as a polynomial in the problem data.

Given some relaxed form of Eq. 38, there are two general classes of primal-dual interior point methods: path-following algorithms and potential reduction algorithms. We will consider the potential reduction algorithm of Nesterov and Nemirovski (1994). This method was used to solve the examples of the next section.

A full explanation of Nesterov and Nemirovski's potential reduction algorithm can be found in the article by Lobo et al. (1998). If x_{i1} is zero, the SOCP reduces to an LP and this algorithm reduces to Ye's potential reduction algorithm (Ye, 1991) for linear programming. The potential function for the algorithm is given by

$$\psi(x, z) = \left(2N + \nu\sqrt{2N}\right) \ln \mu - \sum_{i=1}^{n} \left(\ln \gamma(x_i) + \ln \gamma(z_i)\right) - 2N \ln N, \tag{39}$$

where ν is an adjustable parameter and $\ln \gamma$ is the barrier function

$$\gamma(x_i) = \gamma(x_{i0}, x_{i1}) = \begin{cases} \left(x_{i0}^2 - \| x_{i1} \|\right) & \text{for } x \geq_{\kappa} 0 \\ \infty & \text{otherwise.} \end{cases}$$

The potential function keeps the iterates away from the boundary by acting as a barrier for the pair (x, z), balancing duality and feasibility. As ψ approaches $-\infty$, μ approaches 0, and (x, z) approaches optimality.

The actual steps of the algorithm are as follows:

(1) Generate a search direction by solving a linear system similar to Eq. 38 (see Nesterov and Nemirovski (1994) or Lobo et al. (1998) for details).

(2) Perform a plane search on the potential function to determine the fraction of step to take.

(3) Update the iterates.

One characteristic of potential-reduction algorithms is that under mild conditions, they guarantee a fixed amount of decrease in the potential function at each iteration. They also tend to perform well in practice.

We conclude this section by mentioning that there are several direct extensions of path-following algorithms from linear programming to second-order cone programming. The algorithm of Monteiro and Tsuchiya (1998) for second-order cone programming is a direct extension of Mizuno et al.'s (1993) predictor-corrector algorithm for linear-programming. Also, the algorithm by Alizadeh and Schmieta (1997) is a direct extension of Mehrota's (1992) LP predictor-corrector algorithm. For other nonlinear interior-point algorithms for solving SOCPs, see Vanderbei and Yuritan (1998).

Examples

In this section we consider two examples that illustrate the nature of the robust LP. The examples show instances when, because of model mismatch, the targets calculated by the nominal LP result in poor control, while the targets calculated by the robust LP provide much improved control. The simulations were run on a desktop PC with an Intel Pentium II processor using MATLAB.

The SOCP corresponding to the robust steady-state target calculation (Eq. 30) was solved using the routine SOCP by Lobo et al. (1997). The SOCP routine makes use of Nesterov and Nemirovski's (1994) potential reduction algorithm. The nominal and enumerated LPs are solved using a variation of the simplex method (Dantzig et al., 1955; Grace, 1995). the quadratic program in the dynamic calculation is solved using an active set SQP method with a BFGS update for the Hessian (Grace, 1995).

SISO example

Consider the simple single-input, single-output (SISO) steady-state model given by

$$(y_k - y_{k-1}) = g(u_k - u_{k-1}) + b,$$

where u_k and y_k are the input and output at time k, g is the model gain, and b is the model bias. Assume that the gain is nominally

$$\tilde{g} = \frac{1}{2},$$

but the true plant gain is given by

$$g_{\text{act}} = 2,$$

which is assumed to lie in some ellipsoid. Also, assume unit constraints on the input and output

$$-1 \leq y \leq 1,$$
$$-1 \leq u \leq 1.$$

Let the objective be given by

$$J_s = u - 3y.$$

The coefficients of u and y are such that the objective wants to drive the process toward large inputs and outputs.

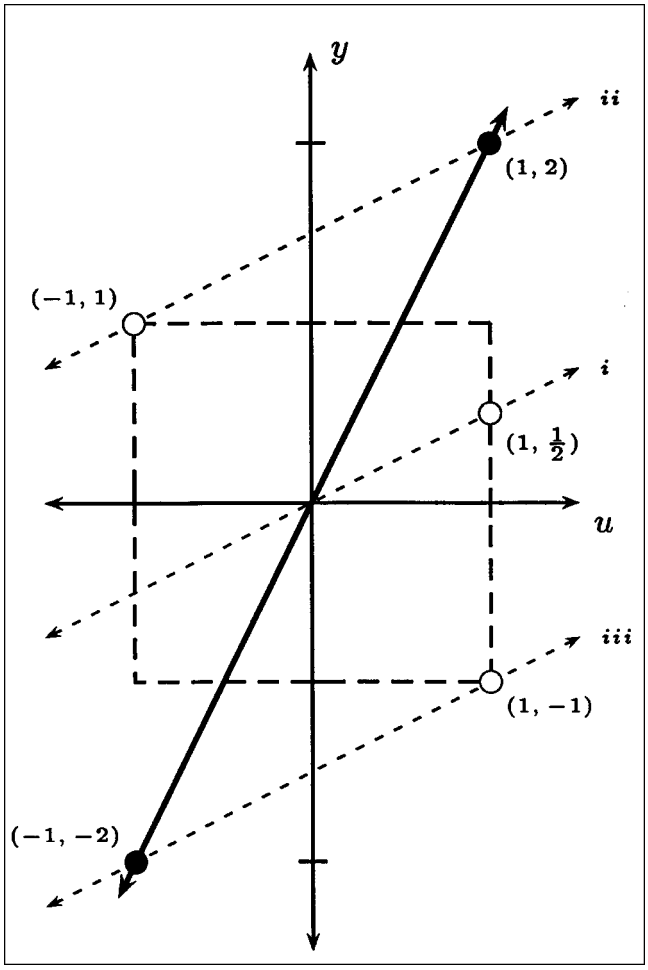

Figure 5. Phase portrait for SISO example.

——: true plant; – – –: nominal models.

For the sake of simplicity, we focus initially on steady-state control only. This means that the MPC control consists of only a steady-state target calculation, and the sample time is chosen long enough that the process reaches steady state between each calculation.

The simulation for the steady-state controller with the above data is shown in Figure 5. This is a phase portrait of the iterates as they evolve in time. The box in the figure represents the feasible region defined by the input and output constraints. The solid line represents the true plant and has a slope $m = 2$. The dotted lines represent the nominal model at different time steps (including bias), each with a slope $m = 1/2$.

We now trace the evolution of the closed-loop system. In Figure 5, the open circles correspond to the predicted input and output. The filled circles correspond to the actual input and output. The numbers next to each circle indicate the value of the input output pair (u, y). If, at time zero, the plant is at the origin with zero bias, then the model, indicated by line *i*, predicts the optimal steady-state should lie at $(1, 1/2)$. However, because of mismatch, once the input $u = 1$ is injected, the true plant is at $(1, 2)$. Next bias is included in the model. This raises the nominal model line *i* to the new position *ii*. The new model has an intercept of $b = 3/2$. This forces the model to agree with measured output. The optimal steady-state solution using the nominal model *ii* for this time-step is at $(-1, 1)$. Because of model error, however, the true plant is actually at $(-1, -2)$. We again update the bias, yielding line *iii*. The new solution is at $(1, -1)$. Injecting $u = 1$ into the plant we see that the process is moved in the opposite direction and the closed-loop system begins to cycle. This cycling is illustrated in Figure 6 where the inputs and outputs are plotted as function of time. For this case, the nominal LP drives the input from one side of the feasible space to the other at each time step. The output shows similar behavior except that it is *never feasible at the sample times*. Now consider what would happen if the nominal LP were replaced with the proposed robust LP.

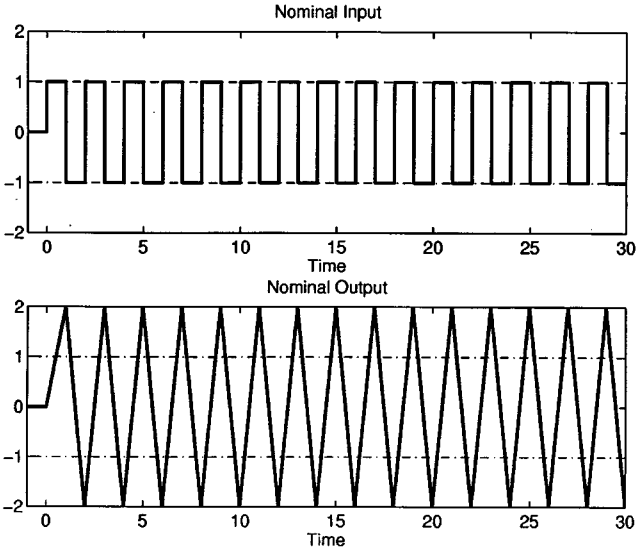

Figure 6. Steady-state only controller: nominal LP.

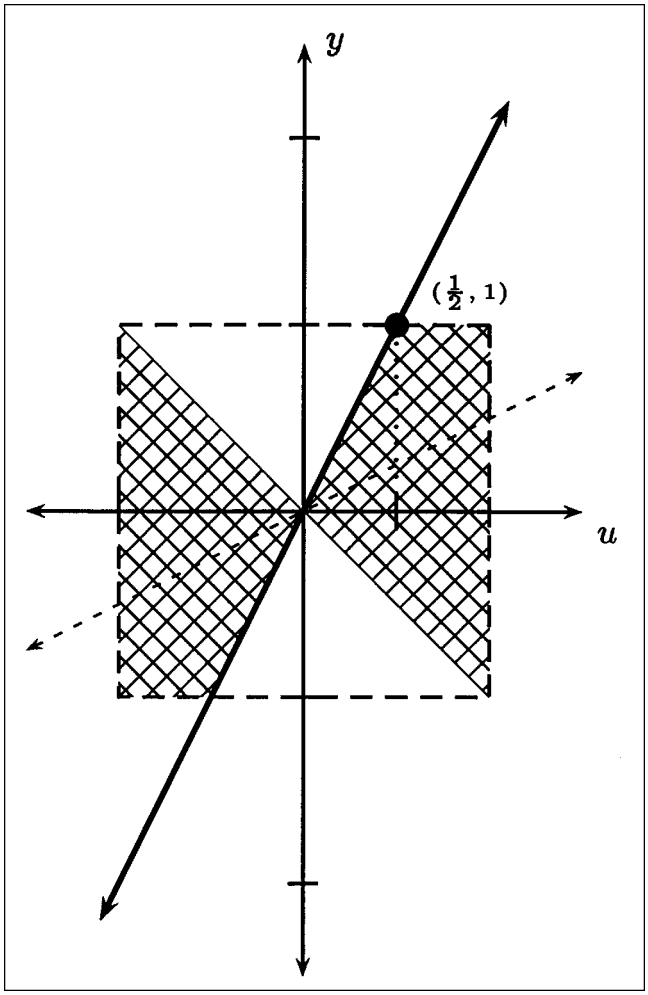

Figure 7. Phase portrait for SISO example with gain uncertainty.

——: true plant; – – –: nominal model; hashed region-realizable gains.

Assume that we know more than just the mean model gain. If we know to a 95% probability that the gain is between 2 and -1 (or equivalently that $\Delta g = 1\ 1/2$), then we can control the process more efficiently with the robust LP. The solution of the robust LP can be found graphically (see Figure 7). The hashed region in the figure represents the gain uncertainty. The optimal steady-state solution is the input for which any realizable output remains feasible: $(u, y) = (1/2, 1)$. At the next iteration when bias is included in the calculation, the steady state remains the same (see Figure 8). We can see that the robust LP prevents oscillations in the input and output, and prevents violation of the output constraint.

We now include dynamics in the problem. For simplicity, let the system be first order with a time constant of 3 for both the nominal model and plant

$$\tilde{g} = \frac{1}{3s+1}0.5, \quad g_{\text{act}} = \frac{1}{3s+1}2.0$$

We use a step-response model with 30 coefficients to describe both the nominal model and the plant. The dynamic

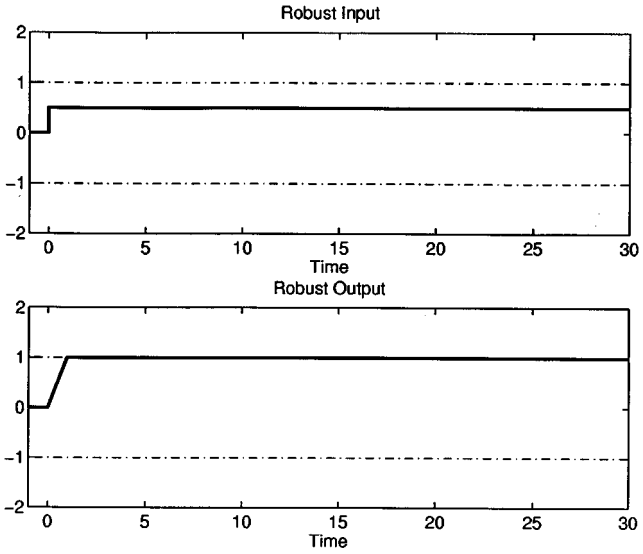

Figure 8. Steady-state only controller: Robust LP.

calculation uses the modified version of the QDMC algorithm discussed earlier with a control horizon of $c = 5$, a prediction horizon of $p = 35$, and simple tuning weights of $w_u = 0.1$ and $w_y = 0.5$.

As illustrated in Figure 9, the nominal LP cycles because of the model mismatch, while the robust LP does not. While the cycling is not as strong as in the steady-state controller, the mechanism producing the cycling is the same. The steady-state algorithm is trying to find a target that agrees with the feedback it has received while satisfying the constraints. Because the model that it is using is not correct, the steady-state target iterates end up 'bouncing' around their correct values. The input, in fact, is forced to cross from one side of its operating region to the other. The output is not maximized as the LP costs dictate. The robust LP, on the other hand, moves the system to the best operating point given the uncertainty of the problem.

While this example is somewhat pathological, it demonstrates the importance of addressing uncertainty in the steady-state target calculation. Next, we consider a more complicated, multiple-input, multiple-output example.

Shell fundamental control problem

Let us consider the performance of the robust LP on a modified subset of the Shell fundamental control problem (Prett and Morari, 1987) illustrated in Figure 10. The process is a heavy oil fractionator in which a gaseous feed stream is separated by removing heat. We will only consider the two-by-two subsystem

$$y = \begin{bmatrix} \dfrac{4.05e^{-30s}}{50s+1} & \dfrac{1.77e^{-30s}}{60s+1} \\ \dfrac{5.39e^{-20s}}{50s+1} & \dfrac{5.72e^{-15s}}{60s+1} \end{bmatrix} u. \qquad (40)$$

The outputs $y = [y_1\ y_2]^T$ are the top end point y_1 and side end point y_2. The inputs $u = [u_1\ u_2]^T$ are the top draw u_1 and side draw u_2. The time constants and dead times are given in units of minutes and the variables are all normalized. The problem has a known steady-state gain uncertainty

$$G = \tilde{G} \pm \Delta G \qquad (41)$$

where

$$\tilde{G} = \begin{bmatrix} 4.05 & 1.77 \\ 5.39 & 5.72 \end{bmatrix} \quad \text{and} \quad \Delta G = \begin{bmatrix} 2.11 & 0.39 \\ 3.29 & 0.57 \end{bmatrix}. \qquad (42)$$

The model is linear and first-order with dead-time. We assume the system dynamics are known exactly and investigate

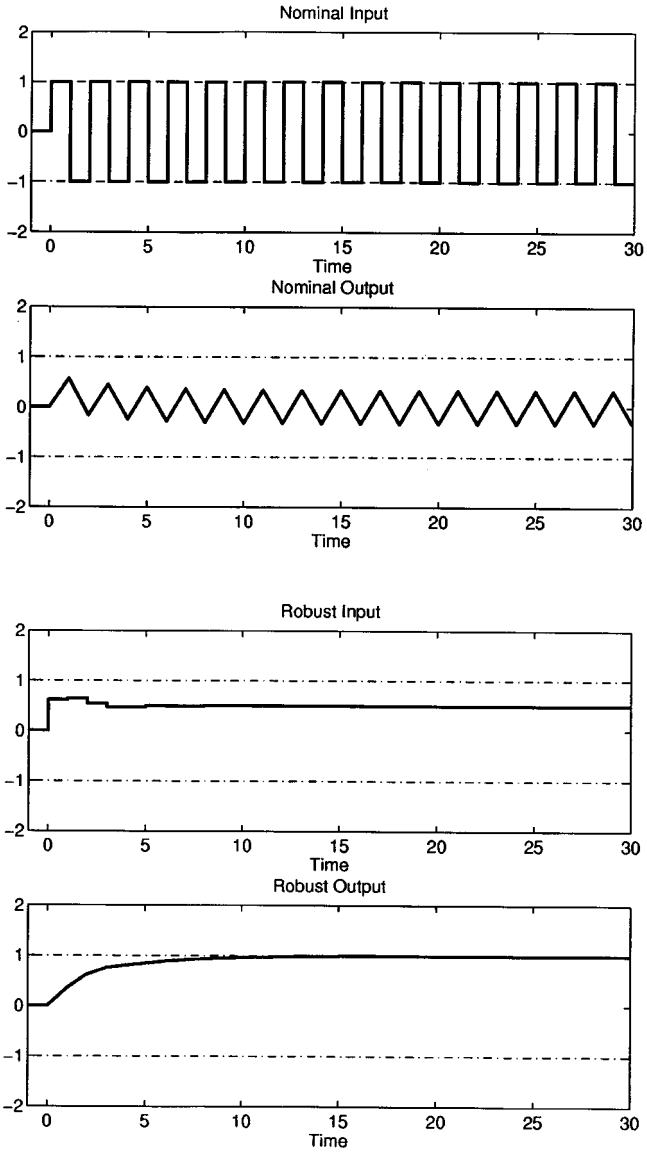

Figure 9. Conventional MPC controller using the nominal LP and robust LP.

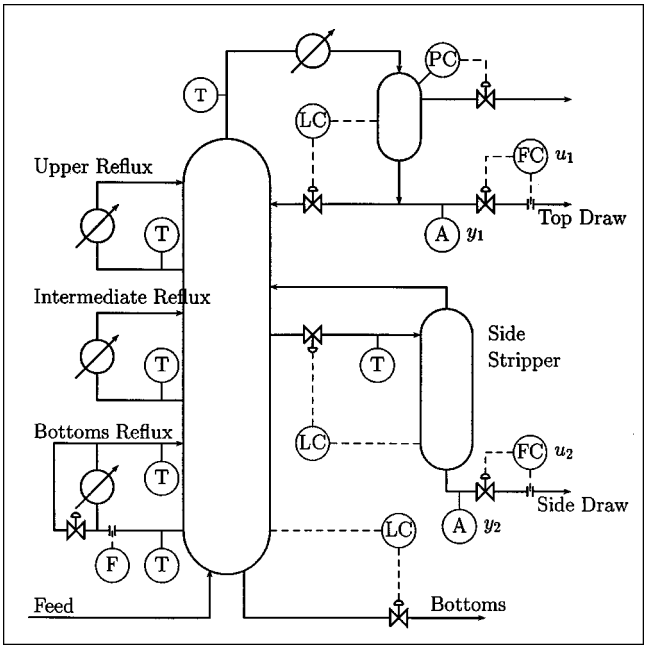

Figure 10. Heavy oil fractionator.

the effect of gain uncertainty on the closed-loop system. The system (Eq. 40) will be represented using a 60 coefficient step-response model with a 5 min sampling time. The constraints of the system are

$$\begin{aligned} -0.5 \le u_1 \le 0.5 \\ -0.5 \le u_2 \le 0.5 \\ -0.5 \le y_1 \le 0.5 \\ -0.5 \le y_2 \le 0.5 \end{aligned} \tag{43}$$

Because the system is linear, we have arbitrarily set the initial operating point to be the origin $(u, y) = (0,0)$. The LP costs are such that the end points are maximized

$$c = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \qquad d = -\begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}. \tag{44}$$

In the simulations we consider the behavior of the closed-loop system when the steady-state targets are calculated by the nominal LP, the robust LP, and the enumerated LP. Equation 42 is a box uncertainty description. The uncertainty for the robust LP is formed by inscribing an ellipse in the box defined by Eq. 42. The enumerated LP uses the box uncertainty directly and refers to the case in which the output constraint is enumerated for all values of the gain (see the subsection on problem formulation). The targets are sent to the dynamic algorithm described earlier. The dynamic algorithm implements input, but no output constraints and is tuned to yield good performance fo the nominal model. We only show plots for the nominal and robust LP, since the enumerated LP solution is similar to that for the robust LP.

Figure 11 shows the closed-loop response when we assume the model given by Eq. 40 exactly describes the plant. The system starts at the origin, and the LP costs drive the system toward the upper output constraints. Both outputs in the nominal controller converge to their upper constraints. The robust controller pushes the side end point to its upper limit, but finds a different optimal steady-state operating point for the top end point. The robust LP, unlike the nominal LP, does not always push the system to the intersection of constraints. In this case, the robust LP has pushed the system to the best operating point given the uncertainty in the problem.

Now, consider the case when the plant and model differ. Assume that the steady-state gain of the system is actually

$$G_{\text{act}} = \begin{bmatrix} 2.30 & 1.87 \\ 8.59 & 5.22 \end{bmatrix}. \tag{45}$$

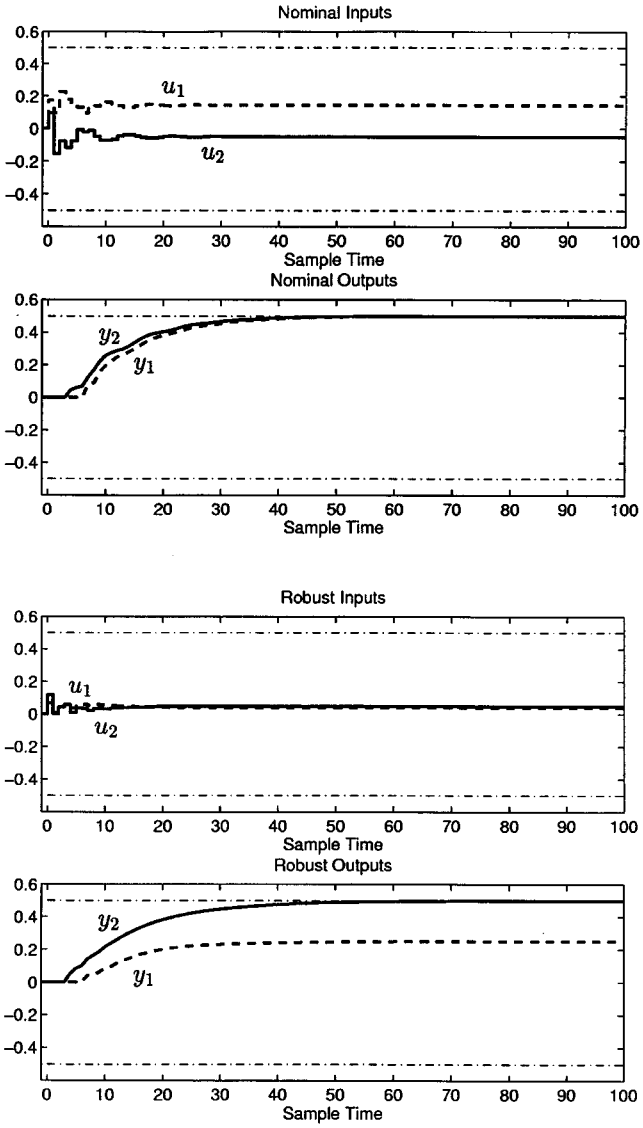

Figure 11. Closed-loop response of the system with targets calculated via the nominal LP and robust LP for a perfect model and nonzero uncertainty.

$N = 60$, $c = 20$, $p = 85$, $w_u = 2.5$, $w_y = 1.0$.

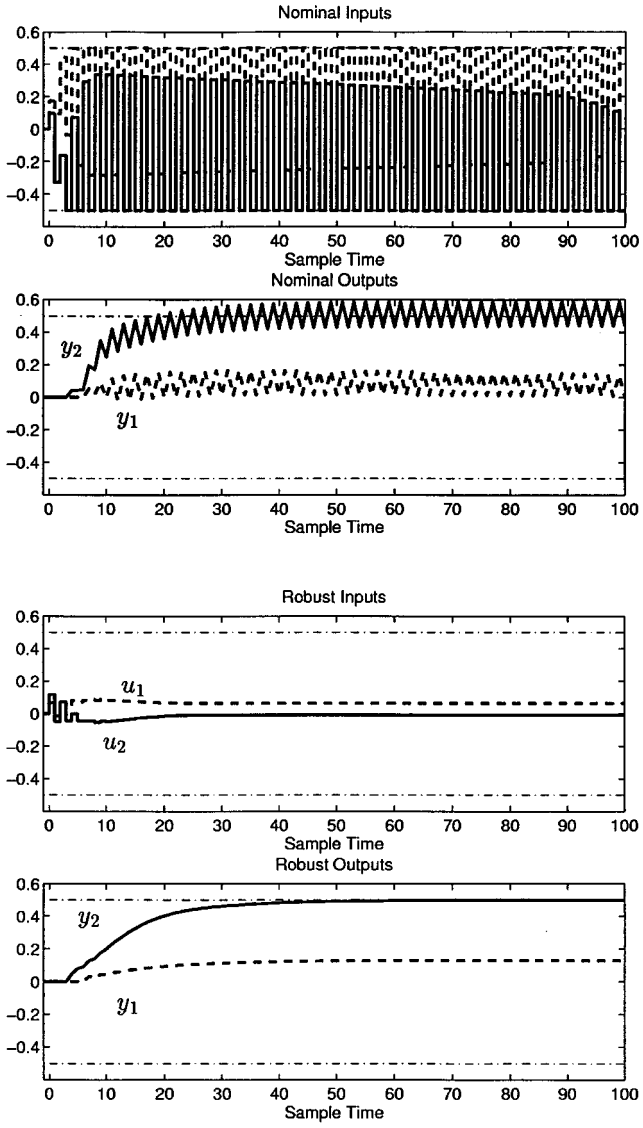

Figure 12. Closed-loop response of the system with targets calculated via the nominal LP and robust LP in the presence of model uncertainty.

$N = 60,\ c = 20,\ p = 85,\ w_u = 2.5,\ w_y = 1.0.$

which lies within the uncertainty description (Eq. 41). We will use this value of the true plant gain for the remaining simulations. Figure 12 shows the closed-loop response. The nominal controller goes unstable, but the robust controller is still stabilizing. This is in part due to the fact that the determinant of the model gain has the wrong sign

$$\det G_{\text{act}} = -4.03, \qquad \det \tilde{G} = 13.57.$$

The plant moves in a direction opposite that of the model prediction, forcing the controller to go unstable. The robust algorithm successfully moves the top end point to its upper limit and again moves the side end point to an unconstrained steady state. While the robust algorithm cannot guarantee closed-loop stability when the determinant of the gain changes the sign, this shows that it can be less sensitive to errors of this type.

If we increase the move suppression weight w_u, we should be able to impart robustness to the nominal controller and make the closed-loop system stable at the cost of a performance loss. With the same plant gain (Eq. 45) and increased move suppression $w_u = 3$, we obtain the simulation shown in Figure 13. The nominal controller is now stable, but the side end point is minimized instead of maximized. The nominal controller, while predicting that the end points should both be maximized, actually has moved the system in the wrong direction because of error in the model gain. The robust controller is still stable with the new tuning and maximizes both end points as the LP costs dictate.

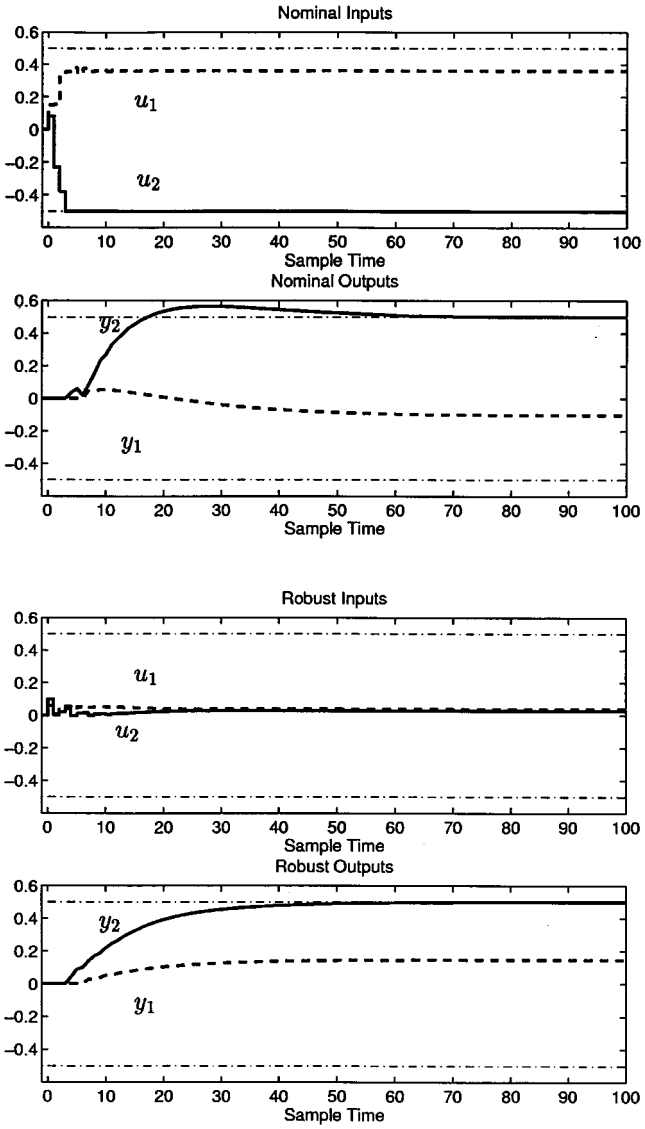

Figure 13. Closed-loop response of the system with targets calculated via the nominal LP and robust LP in the presence of model uncertainty.

$N = 60,\ c = 20,\ p = 85,\ w_u = 3.0,\ w_y = 1.0.$

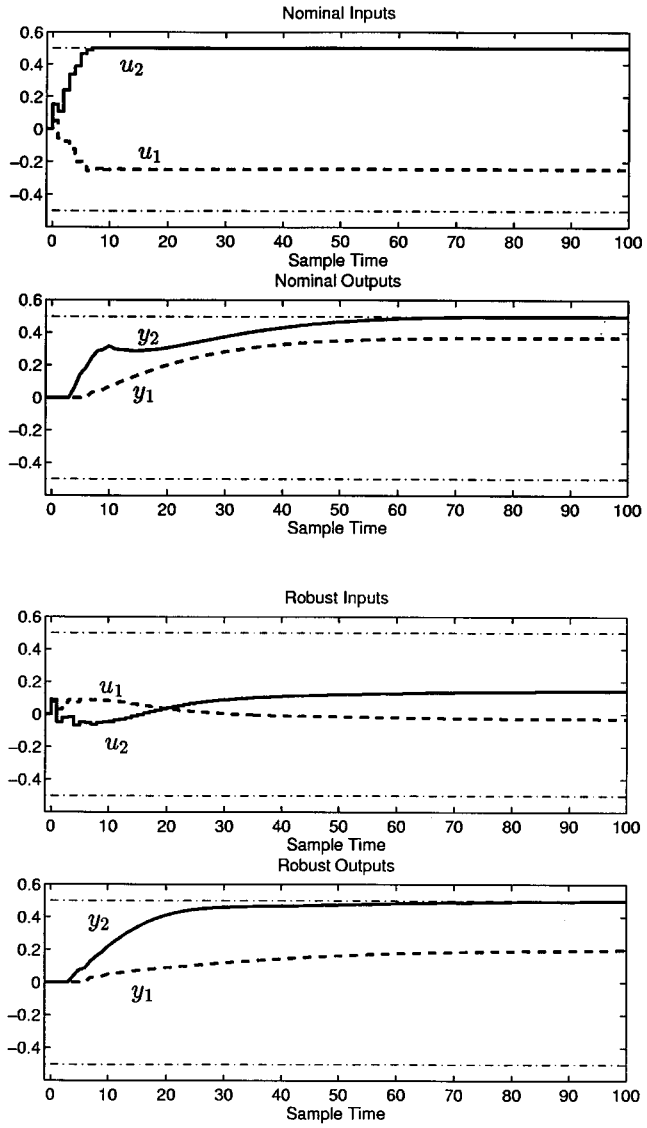

Figure 14. Closed-loop response of the system with targets calculated via the nominal LP and robust LP for a perfect model and nonzero uncertainty.

$N = 60,\ c = 20,\ p = 85,\ w_u = 3.0,\ w_y = 1.0.$

Finally, assume that we can identify the actual plant perfectly, but $\Delta G \neq 0$. Assume

$$\tilde{G} = G_{\text{act}} \qquad \text{with} \qquad \Delta G = \begin{bmatrix} 0.75 & 0.25 \\ 1.25 & 0.25 \end{bmatrix} \tag{46}$$

with G_{act} given by Eq. 45. Figure 14 shows the resulting simulation. The nominal algorithm with a perfect model now correctly pushes the closed-loop system in the correct direction. The robust controller continues to move the system to the best operating point given the uncertainty of the problem. If we were to decrease the uncertainty, the trajectories in the robust simulation would become closer and closer to those in the nominal simulation. In the limit of zero uncertainty $\Delta G \rightarrow 0$, the robust simulation is identical to the nominal simulation.

Table 1. Average Solution Times vs. Problem Size for the Nominal, Robust and Enumerated LP of Shell Fundamental Control Problem

	Nominal LP	Robust LP	Enumerated LP
No. of variables	6	6	66
No. of constraints	12	12	132
Time (s)	$(2.1 \pm 0.8) \times 10^{-2}$	$(3.4 \pm 0.6) \times 10^{-2}$	$(7.5 \pm 0.3) \times 10^{-1}$

All these simulations were run using the enumerated LP as well. As mentioned previously, for this example, the control profile for the enumerated LP is very similar to the control profile for the robust LP. This is generally not the case, and the two controllers can produce different closed-loop responses. Because the enumerated LP contains an increased number of constraints caused by enumerating all possible combinations of the gain, it is slower to solve. Table 1 shows the average solution time of all three algorithms for this example. The error shown in the number represents one standard deviation.

The solution time for the robust LP is roughly 50% higher than that of the nominal algorithm. The particular SOCP routine used for this example takes no advantage of structure and requires strictly feasible initial primal and dual points. By exploiting structure and using a single-stage method, it is likely that solution times for the robust LP can be reduced. The enumerated LP uses a simplex method. Because there are an increased number of constraints, it is slower by over an order of magnitude (even for this two-by-two example).

For these simulations, the controller using targets calculated by the robust LP provided better control than the corresponding controller using targets calculated by the nominal LP. The robust algorithm was able to effectively handle model mismatch, even for the case when the determinant of the gain changed sign.

Conclusions

When investigating the theoretical properties of model predictive control, it is nearly universally assumed that the steady-state target received by the dynamic controller does not change. In practice, however, the target results from a steady-state optimization whose purpose is to reject disturbances and drive the system to some economic optimum. Mismatch between the model used in the target calculation and the true system can cause very poor control.

For the case of a steady-state target calculation based on a linear program, we have shown one way that model uncertainty can be incorporated. For a linear model with elliptic uncertainty on the model parameters, the result is a robust target calculation which takes the form of a second-order cone program (SOCP). This SOCP can greatly improve control by rigorously accounting for modeling uncertainty. The resulting SOCP structure can be exploited to develop efficient numerical solutions based on primal-dual interior-point methods. Several simulations have been provided to demonstrate the improvement possible with this approach.

Acknowledgments

The authors would like to thank Andy Speziale for many helpful and stimulating discussions. This work was supported by the Texas Higher Education Coordinating Board, the National Science Foundation, and Aspen Technology, Inc.

Literature Cited

Albuquerque, J. S., V. Gopal, G. H. Staus, L. T. Biegler, and B. E. Ydstie, "Interior-Point SQP Strategies for Structured Process Optimization Problems," *Computers in Chem. Eng.*, **21**(Suppl), S853 (1997).

Alizadeh, F., and S. H. Schmieta, "Optimization with Semidefinite, Quadratic and Linear Constraints," Technical Report RRR 23-97, Rutgers Center for Operations Research, Rutgers Univ., Piscataway, NJ (1997).

Badgwell, T. A., "Robust Model Predictive Control of Stable Linear Systems," *Int. J. of Control*, **68**, 797 (1997).

Bard, Y., *Nonlinear Parameter Estimation*, Academic Press, New York (1974).

Ben-Tal, A., and A. Nemirovski, "Robust Solutions of Uncertain Linear Programs via Convex Programming," Technical Report, Technion Inst. of Technol., Haifa, Israel (1994).

Ben-Tal, A., and A. Nemirovski, "Robust Convex Optimization," Technical Report, Technion Inst. of Technol., Haifa, Israel (1996).

Boyd, S., C. Crusius, and A. Hansson, "Control Applications of Nonlinear Convex Programming," *J. of Process Control*, **8**, 5-6, 313 (1998).

Chvátal, V., *Linear Programming*, W. H. Freeman and Co., New York (1983).

Cutler, C., A. Morshedi, and J. Haydel, "An Industrial Perspective on Advanced Control," *AIChE Meeting*, Washington, DC (1983).

Dantzig, G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ (1963).

Dantzig, G. B., A. Orden, and P. Wolfe, "The Generalized Simplex Method for Minimizing a Linear Form Under Linear Inequality Constraints," *Pacific J. Math*, **5**, 183 (1955).

Faraut, J., and A. Koráyi, *Analysis on Symmetric Cones*, Oxford University Press, New York (1994).

Faybusovich, L., "Jordan Algebras, Symmetric Cones and Interior-Point Methods," Dept. of Mathematics, Univ. of Notre Dame, Notre Dame, IN (1995).

Faybusovich, L., "Euclidean Jordan Algebras and Interior-Point Algorithms," *Positivity I*, Kluwer Academic Publishers, Boston, p. 331 (1997a).

Faybusovich, L., "Linear Systems in Jordan Algebras and Primal-Dual Interior-Point Algorithms," *J. of Comput. and Appl. Math.*, **86**, 149 (1997b).

García, C. E., and A. M. Morshedi, "Quadratic Programming Solution of Dynamic Matrix Control (QDMC)," *Chem. Eng. Comm.*, **21**, 73 (1986).

Grace, A., *MATLAB Optimization Toolbox User's Guide*, The Mathworks, Inc., Natick, MA (1995).

Hettich, R., and H. T. Jongen, "On First and Second Order Conditions for Local Optima for Optimization Problems in Finite Dimensions," *Methods of Operations Res.*, **23**, 82 (1977).

Hettich, R., and K. O. Kortanek, "Semi-Infinite Programming: Theory, Methods, and Applications," *Siam Rev.*, **35**, 380 (1993).

Hettich, R., and P. Zencke, *Numerishce Methoden der Approximation und Semi-Infiniten Optimierung*, Teubner Studienbücher Mathematik, Stuttgart (1982).

Kall, P., and S. W. Wallace, *Stochastic Programming*, Chap. 4, Wiley, New York (1994).

Kassmann, D., and T. A. Badgwell, "Interior Point Methods in Robust Model Predictive Control," *AIChE Meeting*, Los Angeles (1997).

Kwakernaak, H., and R. Sivan, *Linear Optimal Control Systems*, Wiley, New York (1972).

Lobo, M. S., L. Vandenberghe, and S. Boyd, SOCP: Software for Second-Order Cone Programming, Beta Version, available on the World Wide Web at http://www.stanford.edu/~ boyd/SOCP.html (1997).

Lobo, M. S., L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of Second-Order Cone Programming," *Linear Algebra Appl.*, **284**, 193 (Nov. 1998).

Mehrotra, S., "On the Implementation of a Primal-Dual Interior-Point Method," *SIAM J. Control and Optimiz.*, **3**, 575 (1992).

Mizuno, S., M. J. Todd, and Y. Ye, "On Adaptive-Step Primal-Dual Interior-Point Algorithms for Linear Programming," *Math. of Oper. Res.*, **8**, 964 (1993).

Monteiro, R. D., and T. Tsuchiya, "Polynomial Convergence of Primal-Dual Algorithms for the Second-Order Cone Program Based on the MZ-Family of Directions," Technical Report Memorandum No. 617, Inst. of Statist. Math., Tokyo, Japan (1998).

Muske, K. R., "Steady-State Target Optimization in Linear Model Predictive Control," *Proc. Amer. Control. Conf.*, sponsored by IFAC, Albuquerque, NM (1997).

Muske, K. R., and J. B. Rawlings, "Model Predictive Control with Linear Models," *AIChE J.*, **39**, 262 (1993).

Nesterov, Y., and A. Nemirovski, *Interior-Point Polynomial Methods in Convex Programming," Studies in Applied Mathematics*, Vol. 13, SIAM, Philadelphia (1994).

Nesterov, Y. E., and M. J. Todd, "Self-Scaled Barriers and Interior-Point Methods for Convex Programming," *Math. Oper. Res.*, **22**, 1 (1997).

Nesterov, Y. E., and M. J. Todd, "Primal-Dual Interior-Point Methods for Self-Scaled Cones," *SIAM J. on Optimization*, **8**, 324 (1998).

Polak, E., "On The Mathematical Foundations of Nondifferentiable Optimization in Engineering Design," *SIAM Rev.*, **29**, 21 (1987).

Prékopa, A., *Stochastic Programming*, Chapters 10 and 11, Kluwer Academic Publishers, Boston (1995).

Prett, D. M., and M. Morari, *The Shell Process Control Workshop*, Butterworths, Stoneham, MA (1987).

Qin, S. J., and T. A. Badgwell, "An Overview of Industrial Model Predictive Control Technology," J. C. Kantor, C. E. García, and B. Carnahan, eds., *Fifth Int. Conf. on Chem. Process Control*, AIChE Symp. Ser. **316** (93), 232 (1997).

Ralhan, S., "Robust Model Predictive Control of Linear Finite Impulse Response Plants," Master's Thesis, Rice Univ., Houston (1998).

Rao, C. V., and J. B. Rawlings, "Steady States and Constraints in Model Predictive Control," *AIChE J.*, **45** (1999).

Rao, C. V., S. J. Wright, and J. B. Rawlings, "Application of Interior-Point Methods to Model Predictive Control," *J. of Optimiz. Theory and Appl.*, **99**, 723 (1998).

Reemtsen, R., and J.-J. Rückmann, eds., *Semi-Infinite Programming*, Kluwer Academic Publishers, Boston (1998).

Schmieta, S. H., and F. Alizadeh, "Associative Algebras, Symmetric Cones and Polynomial Time Interior-Point Algorithms," Technical Report RRR 17-98, Rutgers Center for Operations Research, Rutgers Univ., Piscataway, NJ (1998).

Schwarm, A. T., and M. Nikolaou, "Chance Constrained Model Predictive Control," *AIChE Meeting*, Los Angeles (1997).

Schwarm, A. T., and M. Nikolaou, "Robust Output Constraint Satisfaction in MPC: Optimizing Under Uncertainty," *AIChE Meeting*, Miami Beach (1998).

Vandenberghe, L., and S. Boyd, "Semidefinite Programming," *SIAM Rev.*, **38**, 49 (1996).

Vanderbei, R. J., and H. Yuritan, "Using LOQO to Solve Second-Order Cone Programming Problems," Technical Report SOR-98-9, Statistics and Operations Research, Princeton University, Princeton, NJ (1998).

Whittle, P., *Optimization Under Constraints*, Wiley Series in Probability and Mathematical Statistics, Wiley, New York (1971).

Wright, M. H., "The Interior-Point Revolution in Constrained Optimization," Technical Report 98-04-09, Computing Science Research Center, Bell Lab., Murray Hill, NJ (1998).

Wright, S. J., "Applying New Optimization Algorithms to Model Predictive Control," J. C. Kantor, C. E. García, and B. Carnahan, eds., *Fifth Int. Conf. Chem. Process Control*, AIChE Symp. Ser. 316 (93), 147 (1997).

Wu, S., S. Boyd, and L. Vandenberghe, "FIR Filter Design via Semidefinite Programming and Spectral Factorization," *Proc. of IEEE Conf. on Decision and Control*, San Diego, CA (1996).

Ye, Y., "An $O(n^3 L)$ Potential-Reduction Algorithm for Linear Programming," *Math. Programming*, **50**, 239 (1991).

Manuscript received July 6, 1999, and revision received Dec. 10, 1999.