ROBUST LINEAR PROGRAMMING AND OPTIMAL CONTROL

Lieven Vandenberghe [*,1] **Stephen Boyd** [**]
Mehrdad Nouralishahi [*,1]

[*] *Electrical Engineering Department, UCLA*
[**] *Electrical Engineering Department, Stanford University*

Abstract: The paper describes an ef£cient method for solving an optimal control problem that arises in robust model-predictive control. The problem is to design the input sequence that minimizes the peak tracking error between the ouput of a linear dynamical system and a desired target output, subject to inequality constraints on the inputs. The system is uncertain, with an impulse response that can take arbitrary values in a given polyhedral set. This problem can be formulated as a robust linear programming problem with structured uncertainty. The presented method is based on Mehrotra's interior-point method for linear programming, and takes advantage of the problem structure to achieve a complexity that grows linearly with the control horizon, and increases as a cubic polynomial as a function of the system order, the number of inputs, and the number of uncertainty parameters.

Keywords: Linear programming. Convex optimization. Model-predictive control.

1. INTRODUCTION

We describe an ef£cient method for solving the optimal control problem

$$
\begin{array}{ll}
\text{min.} & \sup\limits_{\|\rho\|_\infty \le 1} \max\limits_{t=1,\ldots,N} |c(\rho)^T x(t) - y_{\text{des}}(t)| \\
\text{s.t.} & x(1) = Ax_0 + Bu(0) \\
& x(t+1) = Ax(t) + Bu(t), \\
& \qquad\qquad 1 \le t \le M-1 \\
& x(t+1) = Ax(t), \;\; M \le t \le N-1 \\
& -\mathbf{1} \preceq u(t) \preceq \mathbf{1}, \;\; 0 \le t \le M-1,
\end{array}
\qquad (1)
$$

where $c(\rho) = c_0 + \rho_1 c_1 + \cdots + \rho_p c_p$, and $N \ge M$. The problem data are $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, $x_0 \in \mathbf{R}^n$, the vectors $c_k \in \mathbf{R}^n$, $k = 0,\ldots,p$, and the sequence $y_{\text{des}}(t)$, $t = 1,\ldots,N$. The optimization variables are $u(0), \ldots, u(M-1) \in \mathbf{R}^m$, and $x(1), \ldots, x(N) \in \mathbf{R}^n$, where $u(t)$ and $x(t)$ are the input and the state of a discrete-time linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \quad x(0) = x_0.$$

The constraints also include componentwise upper and lower bounds $-\mathbf{1} \preceq u(t) \preceq \mathbf{1}$ on the input. More complicated constraints, such as input slew rate constraints or terminal state constraints, are readily included, but we will omit them for the sake of simplicity.

To motivate the objective, we £rst consider the special case with $p = 0$, for which the cost function reduces to

$$\max_{t=1,\ldots,N} |c_0^T x(t) - y_{\text{des}}(t)| \qquad (2)$$

We interpret $c_0^T x(t)$ as the output of the system at time t, and $y_{\text{des}}(t)$ as a given desired output, that we want to follow as closely as possible. The problem is to £nd the input sequence $u(0), \ldots, u(M)$ that minimizes the peak tracking error (2), subject to the constraints $-\mathbf{1} \preceq u(t) \preceq \mathbf{1}$. When $p = 0$, we will refer to problem (1) as the *output tracking problem*.

Problem (1) is an extension of the output tracking problem in which we include uncertainty in the system parameters. More speci£cally, we assume that the

[1] Partially supported by the NSF under grants ECS-9733450 and CCR-9714002.

system output is given by $y(t) = c(\rho)^T x(t)$, where the vector c is an af£ne function of some parameter $\rho \in \mathbf{R}^p$, which is unknown but bounded, with components between -1 and 1. Alternatively, we can say that the impulse response coef£cients $h(1), h(2), \ldots$ have the form

$$h(t) = h_0(t) + \rho_1 h_1(t) + \cdots + \rho_p h_p(t),$$

where ρ is unknown with $\|\rho\|_\infty \leq 1$, and $h_k(t)$ is de£ned as $h_k(t) = c_k^T A^{t-1} B$. In problem (1) we minimize the *worst-case* peak tracking error, considering all possible values of ρ. We therefore refer to the problem as the *robust output tracking problem*.

The robust output tracking problem (1) has been applied in robust model-predictive control by (Allwright and Papavasiliou 1992) and (Zheng and Morari 2000). Both papers use a linear programming (LP) formulation (see §5), and solve the resulting LP using general-purpose solvers. The purpose of this paper is to discuss a more ef£cient algorithm, and show that the cost of solving the robust problem is not much higher than the cost of solving the nonrobust problem.

More speci£cally, we will see that if we reformulate problem (1) as an LP, using the formulation of (Allwright and Papavasiliou 1992) and (Zheng and Morari 2000), we obtain an LP with $N(n+p) + Mm + 1$ variables, $2(N(n+p) + Mm)$ inequalities, and Nn equality constraints. We can therefore expect that the computational effort in a general-purpose LP method strongly depends on the control horizons N and M, and that the cost of solving the robust problem ($p > 0$) is much higher than the cost of solving the nonrobust problem ($p = 0$). The main contribution of this paper is to show that we can take advantage of problem structure and reduce the cost per iteration of an LP interior-point method to $2Npn^2 + 3Nn^3 + M(4mn^2 + 4m^2 n + m^3/3)$ ¤oating-point operations (¤ops). In other words, the computational complexity grows *linearly* with N and M, and the complexity of the robust problem is comparable to the complexity of the nonrobust problem.

Numerical algorithms for linear and quadratic programming have been applied to optimal control since the 60s, and are widely used in model-predictive control, see (Morari and Lee 1999, Rawlings 2000). More recently, it was pointed out in (Boyd *et al.* 1998) that new interior-point methods for nonlinear convex optimization (for example, for second-order cone programming or semide£nite programming) allow us to ef£ciently solve a much wider class of optimal control problems, including, for example, problems with uncertain system models, or nonlinear constraints on inputs and states. It was also noted that the resulting convex optimization problems are usually quite large, and may require special-purpose interior-point implementations that take advantage of problem structure.

We £nd two basic approaches in the literature on numerical implementation of interior-point methods for control. Both approaches focus on speeding up the solution of the large sets of linear equations that need to be solved at each iteration, in order to compute the search directions. A £rst idea is to use conjugate gradients to solve these linear systems (Boyd *et al.* 1994, Hansson 2000). Many different types of structure can be exploited this way, often resulting in a speedup by several orders of magnitude. Unfortunately, the performance of the conjugate gradient method is also very sensitive to the problem data, and in general requires good preconditioners. Moreover, the excellent convergence properties of general-purpose interior-point implementations (typically 10–50 iterations) often degrade when conjugate gradients is used to compute search directions. The second approach is less general, but much more reliable, and is based on direct, non-iterative, methods for solving the linear systems fast. Wright, Rao, and Rawlings (Wright 1993, Rao *et al.* 1998) and Hansson (Hansson 2000) have studied quadratic programming formulations of optimal control problems with linear constraints. They show that the Riccati recursion of (unconstrained) linear-quadratic optimal control can be used to compute the search directions in an interior-point method fast, *i.e.*, at a cost that is linear in the control horizon, and cubic in the system dimensions. The results of this paper can be viewed as an extension of the quadratic programming method of (Rao *et al.* 1998) to the robust output tracking problem (1).

Notation We denote by $\mathbf{S}^n$ the space of symmetric matrices of size $n \times n$. The symbols $\succeq$, $\succ$, $\preceq$, and $\prec$ denote componentwise inequality between vectors, or matrix inequality, depending on the context. For example, if $x \in \mathbf{R}^n$, then $x \succeq 0$ means $x_k \geq 0$ for $k = 1, \ldots, n$; if $x \in \mathbf{S}^n$, it means x is positive semide£nite. The symbol $\mathbf{1}$ denotes a vector with all its components equal to one. If $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^p$, then $(x, y) \in \mathbf{R}^{n+p}$ denotes the vector $(x, y) = [x^T \; y^T]^T$.

2. LINEAR-QUADRATIC OPTIMAL CONTROL

In this section we review the classical method for solving the linear-quadratic optimal control problem

$$\begin{array}{ll} \text{min.} & \sum_{t=1}^{N} (\frac{1}{2} x(t)^T Q(t) x(t) - q(t)^T x(t)) \\ & \quad + \sum_{t=0}^{M-1} (\frac{1}{2} u(t)^T R(t) u(t) - r(t)^T u(t)) \\ \text{s.t.} & x(1) = Ax_0 + Bu(0) \\ & x(t+1) = Ax(t) + Bu(t), \\ & \qquad\qquad\qquad 1 \leq t \leq M-1 \\ & x(t+1) = Ax(t), \;\; M \leq t \leq N-1. \end{array}$$

The variables are $u(t)$, $t = 0, \ldots, M-1$ and $x(t)$, $t = 1, \ldots, N$. The weights $Q(t) \in \mathbf{S}^n$ and $R(t) \in \mathbf{S}^m$ are given and satisfy $Q(t) \succeq 0$ and $R(t) \succ 0$ for all t. We also assume that $N \geq M$. To simplify notation, we will express the problem as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{q}^T\mathbf{x} + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} - \mathbf{r}^T\mathbf{u} \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{b} \end{array} \quad (3)$$

where $\mathbf{x} = (x(1), \ldots, x(N)) \in \mathbf{R}^{Nn}$, $\mathbf{u} = (u(0), \ldots, u(M-1)) \in \mathbf{R}^{Mm}$ and

$$\mathbf{b} = (-Ax_0, 0, \ldots, 0) \in \mathbf{R}^{Nn}$$
$$\mathbf{q} = (q(1), \ldots, q(N)) \in \mathbf{R}^{Nn}$$
$$\mathbf{r} = (r(0), \ldots, r(M-1)) \in \mathbf{R}^{Mm}$$
$$\mathbf{A} = \begin{bmatrix} -I & 0 & \cdots & 0 & 0 \\ A & -I & \cdots & 0 & 0 \\ 0 & A & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A & -I \end{bmatrix} \in \mathbf{R}^{Nn \times Nn}$$
$$\mathbf{B} = \begin{bmatrix} B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \in \mathbf{R}^{Nn \times Mm}$$
$$\mathbf{Q} = \begin{bmatrix} Q(1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q(N) \end{bmatrix} \in \mathbf{S}^{Nn}$$
$$\mathbf{R} = \begin{bmatrix} R(0) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R(M-1) \end{bmatrix} \in \mathbf{S}^{Mm}.$$

The quadratic optimization problem (3) can be solved by introducing a Lagrange multiplier $\mathbf{y} \in \mathbf{R}^{Nn}$, associated with the equality constraints. The optimality conditions are

$$\begin{bmatrix} 0 & \mathbf{A} & \mathbf{B} \\ \mathbf{A}^T & \mathbf{Q} & 0 \\ \mathbf{B}^T & 0 & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{q} \\ \mathbf{r} \end{bmatrix}, \quad (4)$$

which is a symmetric inde£nite set of $2Nn + Mm$ equations in $2Nn + Mm$ variables. It follows from our assumptions ($\mathbf{Q} \succeq 0$, $\mathbf{R} \succ 0$) that the coef£cient matrix is nonsingular.

The familiar Riccati recursion from optimal control (Anderson and Moore 1990) can be interpreted as a very ef£cient method for solving equations of the form (4), by taking advantage of the block structure of **A**, **B**, **Q**, and **R**. The computational complexity is

$$3Nn^3 + M(4mn^2 + 4m^2n + m^3/3)$$

¤ops. See (Rao *et al.* 1998, Wright 1993, Vandenberghe *et al.* 2001) for details.

3. MEHROTRA'S METHOD

The algorithms presented in the next two sections are based on Mehrotra's method, one of the most popular algorithms for linear programming. For the sake of conciseness we only give a high level description of the method (more details can be found in (Wright 1997, Vandenberghe *et al.* 2001)). We consider LPs of the form

$$\begin{array}{ll} \text{minimize} & d^T\tilde{x} \\ \text{subject to} & G\tilde{x} \preceq g \\ & H\tilde{x} = h. \end{array} \quad (5)$$

The variable is $\tilde{x} \in \mathbf{R}^n$. The problem data are $d \in \mathbf{R}^n$, $G \in \mathbf{R}^{m\times n}$, $g \in \mathbf{R}^m$, $H \in \mathbf{R}^{p\times n}$, $h \in \mathbf{R}^p$.

Mehrotra's method is an iterative method and typically converges in about 10–50 iterations, almost independently of the problem dimensions and data. The main computation in each iteration is the solution of a linear system of the form

$$\begin{bmatrix} -D & 0 & G \\ 0 & 0 & H \\ G^T & H^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\tilde{z} \\ \Delta\tilde{y} \\ \Delta\tilde{x} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}, \quad (6)$$

where the matrix D is positive diagonal with values that change at each iteration.

As a practical rule of thumb, we can therefore say that the cost of solving the LP (5) equals the cost of solving about 10–50 sets of linear equations of the form (6).

4. THE OUTPUT TRACKING PROBLEM

We now return to problem (1). We £rst describe an ef£cient method for the special case $p = 0$, and defer the general problem to Section §5. Following the matrix notation introduced in §2, we write the nonrobust problem as

$$\begin{array}{ll} \text{min.} & \|\mathbf{C}_0\mathbf{x} - \mathbf{y}_{\text{des}}\|_\infty \\ \text{s.t.} & -\mathbf{1} \preceq \mathbf{u} \preceq \mathbf{1} \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{b} \end{array} \quad (7)$$

where **A**, **x**, **u**, **b** are de£ned as in §2, and

$$\mathbf{C}_0 = \begin{bmatrix} c_0^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & c_0^T \end{bmatrix} \in \mathbf{R}^{N\times Nn}$$
$$\mathbf{y}_{\text{des}} = (y_{\text{des}}(1), \ldots, y_{\text{des}}(N)).$$

Problem (7) is readily formulated as an LP

$$\begin{array}{ll}\text{min.} & w \\ \text{s.t.} & \begin{bmatrix} \mathbf{C}_0 & 0 & -\mathbf{1} \\ -\mathbf{C}_0 & 0 & -\mathbf{1} \\ 0 & I & 0 \\ 0 & -I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ w \end{bmatrix} \preceq \begin{bmatrix} \mathbf{y}_{\text{des}} \\ -\mathbf{y}_{\text{des}} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} \\ & \begin{bmatrix} \mathbf{A} & \mathbf{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ w \end{bmatrix} = \mathbf{b}. \end{array} \tag{8}$$

The variables are $\mathbf{x}$, $\mathbf{u}$, and a scalar w. The LP has the form (5), so solving it ef£ciently requires solving a sequence of linear equations of the form (6). It can be shown (Vandenberghe *et al.* 2001) that these equations reduce to a set of equations of the form

$$\begin{bmatrix} 0 & \mathbf{A} & \mathbf{B} & 0 \\ \mathbf{A}^T & \mathbf{Q} & 0 & \mathbf{d} \\ \mathbf{B}^T & 0 & \mathbf{R} & 0 \\ 0 & \mathbf{d}^T & 0 & \gamma \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y} \\ \Delta\mathbf{x} \\ \Delta\mathbf{u} \\ \Delta w \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ r_4 \end{bmatrix}, \tag{9}$$

where $\mathbf{R}$ is positive diagonal, and

$$\mathbf{Q} = \mathbf{C}_0^T \mathbf{D}_0 \mathbf{C}_0, \; \mathbf{d} = \mathbf{C}_0^T \tilde{\mathbf{D}}_0 \mathbf{1}, \; \gamma = \mathbf{Tr}\, \mathbf{D}_0,$$

with $\mathbf{D}_0$ positive diagonal, and $\tilde{\mathbf{D}}_0$ diagonal.

Note that eliminating Δw from (9) would result in a 3×3 block matrix with a large dense matrix $\mathbf{Q} - (1/\gamma)\mathbf{d}\mathbf{d}^T$ in the $(2,2)$-position. Instead of eliminating Δw, we therefore solve two equations

$$\begin{bmatrix} 0 & \mathbf{A} & \mathbf{B} \\ \mathbf{A}^T & \mathbf{Q} & 0 \\ \mathbf{B}^T & 0 & \mathbf{R} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}_1 \\ \Delta\mathbf{x}_1 \\ \Delta\mathbf{u}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix},$$
$$\begin{bmatrix} 0 & \mathbf{A} & \mathbf{B} \\ \mathbf{A}^T & \mathbf{Q} & 0 \\ \mathbf{B}^T & 0 & \mathbf{R} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}_x \\ \Delta\mathbf{x}_2 \\ \Delta\mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{d} \\ 0 \end{bmatrix},$$

and then make a linear combination to satisfy the last equation, *i.e.*, calculate the solution of (9) as

$$\begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{u} \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{x}_1 \\ \Delta\mathbf{y}_1 \\ \Delta\mathbf{u}_1 \end{bmatrix} - \Delta w \begin{bmatrix} \Delta\mathbf{x}_2 \\ \Delta\mathbf{y}_2 \\ \Delta\mathbf{u}_2 \end{bmatrix}$$

where $\Delta w = (r_4 - \mathbf{d}^T \Delta\mathbf{x}_1)/(\gamma - \mathbf{d}^T \Delta\mathbf{x}_2)$. The equations (10) have exactly the same form as (4). Moreover $\mathbf{R}$ is positive diagonal, and $\mathbf{Q}$ is block diagonal with positive semide£nite diagonal blocks

$$Q(t) = D_0(t) c_0 c_0^T, \; t = 1, \ldots, N,$$

where the diagonal elements of $\mathbf{D}_0$ are denoted by $D_0(t)$. We can therefore apply the Riccati method described in §2, and solve (9) in roughly

$$3Nn^3 + M(4mn^2 + 4m^2 n + m^3/3) \text{ ¤ops.}$$

5. THE ROBUST TRACKING PROBLEM

The method of the previous paragraph can be extended to the robust tracking problem (1), which can be expressed concisely as

$$\begin{array}{ll} \text{min.} & \sup_{\|\rho\|_\infty \le 1} \|(\mathbf{C}_0 + \sum_{i=1}^{p} \rho_i \mathbf{C}_i)\mathbf{x} - \mathbf{y}_{\text{des}}\|_\infty \\ \text{s.t.} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{b} \\ & -\mathbf{1} \preceq \mathbf{u} \preceq \mathbf{1} \end{array}$$

where

$$\mathbf{C}_i = \begin{bmatrix} c_i^T & 0 & \cdots & 0 \\ 0 & c_i^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_i^T \end{bmatrix} \in \mathbf{R}^{N \times Nn}.$$

The other matrices and vectors are de£ned as before. This problem can be formulated as an LP

$$\begin{array}{ll}\text{min.} & w \\ \text{s.t.} & \begin{bmatrix} \mathbf{C}_0 & 0 & \mathbf{E} & -\mathbf{1} \\ -\mathbf{C}_0 & 0 & \mathbf{E} & -\mathbf{1} \\ \mathbf{C} & 0 & -I & 0 \\ -\mathbf{C} & 0 & -I & 0 \\ 0 & I & 0 & 0 \\ 0 & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \\ w \end{bmatrix} \preceq \begin{bmatrix} \mathbf{y}_{\text{des}} \\ -\mathbf{y}_{\text{des}} \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}, \\ & \begin{bmatrix} \mathbf{A} & \mathbf{B} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \\ w \end{bmatrix} = \mathbf{b}, \end{array}$$

where

$$\mathbf{E} = \begin{bmatrix} I & I & \cdots & I \end{bmatrix} \in \mathbf{R}^{N \times Np}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \vdots \\ \mathbf{C}_p \end{bmatrix}$$

and $\mathbf{v} \in \mathbf{R}^{Np}$ is an auxiliary variable.

As in §4 it can be shown that Mehrotra's applied to this LP reduces to solving 10–50 linear systems of the form (9), where $\mathbf{R}$ is positive diagonal, and $\mathbf{Q}$ is block diagonal with diagonal blocks

$$Q(t) = \sum_{i=0}^{p} \frac{D_i(t)^2 - \tilde{D}_i(t)^2}{D_i(t)} c_i c_i^T + \frac{1}{\sum_{i=0}^{p} D_i(t)^{-1}} \left(\sum_{i=0}^{p} \frac{\tilde{D}_i(t)}{D_i(t)} c_i \right) \left(\sum_{i=0}^{p} \frac{\tilde{D}_i(t)}{D_i(t)} c_i \right)^T,$$

where $D_i(t) > 0$ and $\tilde{D}_i(t)$ change at each iteration. The cost of forming $\mathbf{Q}$ is approximately $2Npn^2$ ¤ops, ignoring lower-order terms. The total number of ¤ops per iteration of Mehrotra's method is therefore about

$$2Npn^2 + 3Nn^3 + M(4mn^2 + 4m^2 n + m^3/3).$$

6. NUMERICAL RESULTS

Both algorithms have been implemented in Matlab (Version 6) and tested on a 933 Mhz Pentium III running Linux. Table 1 summarizes the results for a family of (nonrobust) output tracking problems with randomly generated problems (using Matlab's `drss` function). The £rst four columns give the problem dimensions. Columns 5–7 give the number of variables, inequalities, and equality constraints in the corresponding LPs (8). The last three columns give the number of iterations to reach a relative error of 0.1%, the total CPU time, and the CPU time per iteration. Table 2 summarizes the results of a similar experiment for the robust output tracking problem.

The results con£rm that the number of iterations grows slowly with problem size, and typically ranges between 10 and 50. From the last column it is also clear that the CPU time per iteration grows linearly with N and M. Within the range of dimensions considered here ($n \le 40$, $m, p \le 20$), the cost per iterations appears to grow more slowly with n, m, and p, than predicted by the theory (which predicts a cubic increase).

Comparing the two tables we note that the cost of solving the robust output tracking problem is only slightly higher than the cost of solving the nonrobust problem, despite the fact that the corresponding LPs are much larger.

7. CONCLUSION

We have described ef£cient methods for solving a constrained linear optimal control problem and its robust counterpart. The methods are based on a primal-dual interior-point method for linear programming, and take a number of iterations that typically ranges between 10 and 50 and appears to grow very slowly with problem size. The cost per iteration is dominated by the solution of a large, structured set of linear equations. By exploiting problem structure, we are able to reduce these linear equations to the solution of an unconstrained quadratic linear optimal control problem, which can be solved ef£ciently by the well-known Riccati recursion.

We have compared in detail the cost of solving the output tracking problem and its robust counterpart. The main contribution of the paper is to show that, despite the size differences of the equivalent LPs, the robust output tracking problem can be solved at a cost that is not much higher than the nonrobust problem.

The techniques discussed here extend to a variety of related problems, for example, problems with an ℓ_1-objective (Rao and Rawlings 2000) or a quadratic objective, problems with additional convex constraints such as slew rate constraints and terminal state constraints, and problems with ellipsoidal uncertainty.

8. REFERENCES

Allwright, J. C. and G. C. Papavasiliou (1992). On linear programming and robust model-predictive control using impulse-responses. *Systems and Control Letters* pp. 159–164.

Anderson, B. and J. B. Moore (1990). *Optimal Control: Linear Quadratic Methods*. Prentice-Hall.

Boyd, S., C. Crusius and A. Hansson (1998). Control applications of nonlinear convex programming. *Journal of Process Control* **8**(5-6), 313–324. Special issue for papers presented at the 1997 IFAC Conference on Advanced Process Control, June 1997, Banff.

Boyd, S., L. Vandenberghe and M. Grant (1994). Ef£cient convex optimization for engineering design. In: *Proceedings IFAC Symposium on Robust Control Design*. pp. 14–23.

Hansson, A. (2000). A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Trans. Aut. Control* **45**, 1639–1655.

Mayne, D. Q., J. B. Rawlings, C. V. Rao and P. O. M. Scokaert (2000). Constrained model predictive control: stability and optimality. *Automatica* **36**, 289–814.

Morari, M. and J. H. Lee (1999). Model predictive control: past, present and future. *Computers and Chemical Engineering* **23**, 667–682.

Rao, C. V. and J. B. Rawlings (2000). Linear programming and model predictive control. *Journal of Process Control* **10**, 283–289.

Rao, C. V., S. J. Wright and J. B. Rawlings (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications* **99**(3), 723–757.

Rawlings, J. B. (2000). Tutorial overview of model predictive control. *IEEE Control Systems Magazine* **20**, 38–52.

Vandenberghe, L., S. Boyd and M. Nouralishahi (2001). Robust linear programming and optimal control. `www.ee.ucla.edu/~vandenbe`.

Wright, S. J. (1993). Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications* **77**, 161–187.

Wright, S. J. (1997). *Primal-Dual Interior-Point Methods*. SIAM. Philadelphia.

Zheng, A. and M. Morari (2000). Robust control of linear time-varying systems with constraints. *International Journal of Robust and Nonlinear Control* **10**, 1063–1078.

dimensions				LP dimensions			# iters	CPU time	time/iter
N	M	n	m	#vars.	#ineqs.	#eqs.		(seconds)	(seconds)
100	50	5	2	601	1200	1000	8	0.9	0.1
500	450	5	2	3401	6800	5000	9	6.9	0.8
1000	950	5	2	6901	13800	10000	11	17.5	1.6
2000	1950	5	2	13901	27800	20000	13	42.9	3.3
100	50	10	5	1251	2500	2000	10	1.3	0.1
500	450	10	5	7251	14500	10000	14	12.8	0.9
1000	950	10	5	14751	29500	20000	11	20.4	1.9
2000	1950	10	5	29751	59500	40000	14	54.3	3.9
100	50	20	10	2501	5000	4000	13	2.4	0.2
500	450	20	10	14501	29000	20000	15	18.6	1.3
1000	950	20	10	29501	59000	40000	16	41.3	2.6
2000	1950	20	10	59501	119000	80000	21	113.9	5.4
100	50	40	20	5001	10000	8000	17	6.6	0.4
500	450	40	20	29001	58000	40000	15	40.0	2.7
1000	950	40	20	59001	118000	80000	21	121.0	5.8
2000	1950	40	20	119001	238999	160000	27	304.1	11.3

Table 1. Number of iterations and CPU times for a family of output tracking problems with randomly generated data.

dimensions					LP dimensions			# iters	CPU time	time/iter
N	M	n	m	p	#vars.	#ineqs.	#eqs.		(seconds)	(seconds)
100	50	5	2	2	801	1600	1000	10	1.3	0.1
500	450	5	2	2	4401	8800	5000	12	10.4	0.9
1000	950	5	2	2	8901	17800	10000	10	17.6	1.8
2000	1950	5	2	2	17901	35800	20000	12	44.5	3.7
100	50	10	5	5	1751	3500	2000	9	1.4	0.2
500	450	10	5	5	9751	19500	10000	13	13.7	1.1
1000	950	10	5	5	19751	39500	20000	13	28.4	2.2
2000	1950	10	5	5	39751	79500	40000	16	71.5	4.5
100	50	20	10	10	3501	7000	4000	20	4.1	0.2
500	450	20	10	10	19501	39000	20000	16	21.7	1.4
1000	950	20	10	10	39501	79000	40000	21	62.0	3.0
2000	1950	20	10	10	79501	159000	80000	21	126.8	6.0
100	50	40	20	20	7001	14000	8000	18	7.9	0.4
500	450	40	20	20	39001	78000	40000	23	69.8	3.0
1000	950	40	20	20	79001	158000	80000	24	152.7	6.4
2000	1950	40	20	20	159001	318000	160000	22	297.7	13.5

Table 2. Number of iterations and CPU times for a family of robust output tracking problems with randomly generated data.