CHARACTERISATION OF RECEDING HORIZON CONTROL FOR CONSTRAINED LINEAR SYSTEMS

María M. Serón, Graham C. Goodwin and José A. De Doná

ABSTRACT

This paper characterises the geometric structure of receding horizon control (RHC) of linear, discrete-time systems, subject to a quadratic performance index and linear constraints. The geometric insights so obtained are exploited to derive a closed-form solution for the case where the total number of constraints is less than or equal to the number of degrees of freedom, represented by the number of control moves. The solution is shown to be a partition of the state space into regions for which an analytic expression is given for the corresponding control law. Both the regions and the control law are characterised in terms of the parameters of the open-loop optimal control problem that underlies RHC and can be computed off line. The solution for the case where the total number of constraints is greater than the number of degrees of freedom is addressed via an algorithm that iteratively uses the off-line solution and avoids on-line optimisation.

KeyWords: Receding horizon control, predictive control, constraints.

I. INTRODUCTION

Receding horizon control (RHC) is a method that, at each sampling instant, computes the current control input by solving an open-loop optimal control problem. The initial state for the optimisation is taken to be the current state of the system, and future states are predicted using a system model. The optimal control sequence resulting from the optimisation is an open loop strategy. However, this is converted into a feedback strategy by applying only the first element of this sequence and then repeating the whole procedure at the next sampling instant when new measurements of the system states are obtained. A recent survey of the field may be found in [8].

An essential feature of RHC is the ability to directly handle constraints, including constraints on the control inputs, system outputs and/or internal states. This feature has been one of the keys to its success in industrial applications [9]. Constraints are simply included as additional conditions to be satisfied in the optimisation problem that is solved at each sampling instant.

In general, constrained optimisation problems do not have a closed-form solution and hence they are typically solved on-line using numerical optimisation methods. Recently, however, there has been interest in studying the underlying structure of RHC, and in obtaining off-line solutions to RHC problems. So, in a sense, the gap between on-line RHC and conventional off-line control methods is being reduced. Since the receding horizon technique is known to perform well in practice, there are many reasons why it would be desirable to have an off-line solution. Firstly, closed-form off-line solutions potentially provide a better understanding of the inherent structure that the optimal receding horizon strategy imposes on the controller. This would also lead to other insights, for example, the fundamental limitations faced in the design of RHC controllers. Secondly, having a complete off-line characterisation of the RHC law implies that the solution is a priori known for all states and this allows verification of the full spectrum of behaviour prior to implementation.

Results pertaining to off-line solutions of RHC problems have been recently reported in [4,5,13,2,1] and [6]. In [4,5], the authors have shown that the receding horizon control problem has a closed-form solution in a local, but nontrivial, region of the state space. This solution is identical to clipping the unconstrained solution. Independently, [13] studied special cases of RHC that take closed forms. In the particular case of horizon N = 1...
they show that the receding horizon control problem has a closed-form solution that agrees with that obtained in [4,5]. In [2], the authors establish, using multi-parametric quadratic programming properties, that the RH control solution for linear systems subject to linear constraints and quadratic cost is a continuous and piecewise affine function of the state. They also present an off-line algorithm that computes the explicit solution numerically. In [6] a suboptimal explicit RH solution is developed that reduces the computational complexity of the explicit solution for large horizons.

Our results agree with those of [2,1] but were derived independently and rely on different techniques. Our approach consists of transforming the open-loop optimal control problem that underlies RH into an equivalent quadratic programme (QP), and then using geometric arguments to solve the latter problem. The resulting solution consists of a partition of the state space into regions in which the corresponding control law has an affine analytical form. Both the regions and the control law are characterised in terms of the parameters of the underlying open-loop optimal control problem. In this way, RH is presented as a piece-wise affine switching strategy that can be pre-computed off-line. The proposed solution applies directly to the cases where the number of control moves, then our algorithm gives an affine analytical form. Both the regions and the control law are characterised in terms of the parameters of the underlying open-loop optimal control problem. In this way, RH is presented as a piece-wise affine switching strategy that can be pre-computed off-line. The proposed solution applies directly to the cases where the number of state variables is greater than or equal to the number of degrees of freedom, represented by the number of control moves. When the total number of constraints is greater than or equal to the number of degrees of freedom, the solution is addressed via an algorithm that iteratively uses the off-line solution and avoids on-line optimisation.

Regarding numerical implementation, the algorithm of [2,1] explores the state space recursively to find the final region partition. On the other hand, our algorithm computes the solution in the “input space” (see Section 3) for all combinations of constraints and selects those that have a nonempty projection on the state space. If the number of state variables is greater than or equal to the number of control moves, then our algorithm gives directly the complete region partition without need for the elimination of empty regions. Examples indicate that there is a slight, but non-negligible, off-line computational advantage over the results of Bemporad et al. for short horizons, although a strict comparison should be weighted with a variety of factors such as computer speeds, coding details, etc.

Once the state space partition and control law are computed off-line, they can be stored for on-line use within a “table look-up” implementation of RH. Preliminary results on numerical properties of the table look-up implementation of the RH solution and its comparison with traditional methods for on-line optimisation (such as active constraint methods for quadratic programming) are reported in [12].

The remainder of the paper proceeds as follows. In Section 2 we formulate the problem under consideration. In Section 3 we describe the technique that we employ for its solution, namely, a geometric view of quadratic programming. The solution for the case where the number of linear constraints is less than or equal to the number of control moves is given in Section 4. The solution for more general cases is addressed in Section 5. Finally, conclusions are given in Section 6.

A preliminary version of this result, covering the input constraint case only, was reported in [11].

II. PROBLEM FORMULATION

The system model is given by

$$x(k + 1) = Ax(k) + Bu(k), \quad k = 0,1,2,... \quad (1)$$

$$y(k) =Cx(k), \quad (2)$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^n$ is the input vector, and $y(k) \in \mathbb{R}^p$ is the “output” vector or combinations of states of interest. We assume that $(A,B)$ is stabilisable and that the matrices $B$ and $C$ have full column and row ranks, respectively.

For this model, at time $k$, the RH technique poses the following finite-horizon open-loop optimal control problem: given the current state measurement $x(k) = x$, find the $N$-move control sequence $U = \{u(k),u(k+1),...,u(k+N-1)\}$ that minimises the performance index:

$$V_N(x,U) = \sum_{i=k}^{k+N-1}[x^T(i)Qx(i) + u^T(i)Ru(i)] + x^T(k+N)Px(k+N). \quad (3)$$

In (3), $N$ is the prediction horizon; $Q \geq 0$ and $R > 0$ are the state and control weighting matrices, respectively, and $x^T Px + Pu^T R u$, is the terminal cost function.

We will express (1) and (3) in a more convenient form. To this end, we collect $N x$ and $N x$ vectors in the following $(Nn \times 1)$ and $(Nn \times 1)$ vectors, respectively:

$$x = \begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N) \end{bmatrix} u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

Then, from (1), we can write

$$x = \Gamma u + \Omega \xi, \quad (4)$$

where $s(k) = x$ and

Once the state space partition and control law are computed off-line, they can be stored for on-line use within a “table look-up” implementation of RH. Preliminary results on numerical properties of the table look-up implementation of the RH solution and its comparison with traditional methods for on-line optimisation (such as active constraint methods for quadratic programming) are reported in [12].

The remainder of the paper proceeds as follows. In Section 2 we formulate the problem under consideration. In Section 3 we describe the technique that we employ for its solution, namely, a geometric view of quadratic programming. The solution for the case where the number of linear constraints is less than or equal to the number of control moves is given in Section 4. The solution for more general cases is addressed in Section 5. Finally, conclusions are given in Section 6.

A preliminary version of this result, covering the input constraint case only, was reported in [11].

II. PROBLEM FORMULATION

The system model is given by

$$x(k + 1) = Ax(k) + Bu(k), \quad k = 0,1,2,... \quad (1)$$

$$y(k) =Cx(k), \quad (2)$$

where $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^n$ is the input vector, and $y(k) \in \mathbb{R}^p$ is the “output” vector or combinations of states of interest. We assume that $(A,B)$ is stabilisable and that the matrices $B$ and $C$ have full column and row ranks, respectively.

For this model, at time $k$, the RH technique poses the following finite-horizon open-loop optimal control problem: given the current state measurement $x(k) = x$, find the $N$-move control sequence $U = \{u(k),u(k+1),...,u(k+N-1)\}$ that minimises the performance index:

$$V_N(x,U) = \sum_{i=k}^{k+N-1}[x^T(i)Qx(i) + u^T(i)Ru(i)]$$

$$+ x^T(k+N)Px(k+N). \quad (3)$$

In (3), $N$ is the prediction horizon; $Q \geq 0$ and $R > 0$ are the state and control weighting matrices, respectively, and $x^T Px + Pu^T R u$, is the terminal cost function.

We will express (1) and (3) in a more convenient form. To this end, we collect $N x$ and $N x$ vectors in the following $(Nn \times 1)$ and $(Nn \times 1)$ vectors, respectively:

$$x = \begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N) \end{bmatrix} u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

Then, from (1), we can write

$$x = \Gamma u + \Omega \xi, \quad (4)$$

where $s(k) = x$ and...
\[
\Gamma = \begin{bmatrix} B & 0 & \ldots & 0 & 0 \\ AB & B & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{n-1}B & A^{n-2}B & \ldots & AB & B \end{bmatrix}, \quad \Omega = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^n \end{bmatrix}
\]

Using (4), (5) and
\[
Q = \text{diag}[Q, \ldots, Q, P], \\
R = \text{diag}[R, \ldots, R],
\]
we can express the performance index (3) as
\[
V_x (x, u) = x^T Q x + x^T Q x + u^T R u
\]
\[
= x^T Q x + [\Gamma u + \Omega x]^T Q [\Gamma u + \Omega x] + u^T R u
\]
(6)

In (6), \( \bar{V} \) is independent of \( u \) and
\[
W = \Gamma^T Q \Gamma + R, \quad F = \Gamma^T Q \Omega.
\]
We will consider the minimisation of (3) for (1) (equivalently, (6)) under the following linear constraints
\[
Lu \leq M,
\]
where the matrix \( L \) and the vector \( M \) have the form
\[
L = \begin{bmatrix} \Phi \\ -\Phi \end{bmatrix}, \quad M = \begin{bmatrix} \Delta \\ \Delta \end{bmatrix} x.
\]
In (9), \( \Phi \in \mathbb{R}^{m \times q} \) has full row rank \( q \leq N m, \Delta \in \mathbb{R}^q \) and \( \Lambda \in \mathbb{R}^{q \times m} \). The integer \( q \) is the constraint horizon for the RHC problem.

The structure of \( L \) and \( M \) in (9) easily accommodates typical constraint requirements, such as, for example, magnitude constraints on the inputs or outputs. We show this in the following remark.

**Remark 2.1.** Let \( y_i, i = 1, 2, \ldots, p, p \leq m, \) be the \( i \)-th component of the output vector \( y \) and let \( C_j \) be the \( j \)-th row of the matrix \( C \). Suppose that we are interested in the following two types of constraints:

- **Magnitude constraints** on the input of the form
\[
\begin{aligned}
|u_{i}(k+j)| & \leq \sigma_{iq}, & \quad \sigma_{iq} & > 0, & \quad i = 1, 2, \ldots, m, \\
\hline
& j = 0, 1, \ldots, N-1.
\end{aligned}
\]

- **Magnitude constraints** on the output of the form
\[
\begin{aligned}
|y_{i}(k+j)| & \leq \Delta_{jq}, & \quad \Delta_{jq} & > 0, & \quad i = 1, 2, \ldots, p, \\
& j = r_i, r_i + 1, \ldots, N,
\end{aligned}
\]

where \( r_i \) is the relative degree of the \( i \)-th output, i.e., the lowest integer \( \nu \) such that \( C_{\nu} A^{\nu-1} B \neq 0 \). Note that using (1), (2), and the definition of relative degree, we can express (11) in terms of the input as
\[
|C_i A^{i-1} B u(t+k)| \leq \Delta_{aj}, \quad \Delta_{aj} > 0, (12)
\]

for \( i = 1, 2, \ldots, p \) and \( j = r_i, r_i + 1, \ldots, N \).

Constraints of the form (10) can be expressed as linear constraints on \( u \) of the form (8) simply by choosing, \( \Phi = I_{\text{inner}}, \Delta = 0_{\text{inner}} \) and, \( \Delta = \sigma \), where
\[
\sigma = [\sigma_{iq}, \ldots, \sigma_{iq}, \sigma_{jq}, \ldots, \sigma_{jq}]
\]

To represent (12) in the form (8), we start by constructing the \( p \times N \) matrix
\[
D = \begin{bmatrix} 0_{(i+q-1), \nu} & \Delta_{v_{1}} & \ldots & \Delta_{v_{N}} \\
0_{(i+q-1), \nu} & \Delta_{v_{2}} & \ldots & \Delta_{v_{N}} \\
\vdots & \vdots & \ddots & \vdots \\
0_{(i+q-1), \nu} & \Delta_{v_{p}} & \ldots & \Delta_{v_{p}} \\
\end{bmatrix},
\]

where \( D_i \) denotes the \( i \)-th column of \( D \). We then form the matrices
\[
\Phi = \begin{bmatrix} CB & 0 & \ldots & 0 & 0 \\
CB & CB & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
CA^{n-1}B & CA^{n-2}B & \ldots & CAB & CB
\end{bmatrix},
\]
\[
\Lambda = \begin{bmatrix} CA \\
CA^2 \\
\vdots \\
CA^n \\
\end{bmatrix}, \quad \Delta = \begin{bmatrix} D_1 \\
D_2 \\
\vdots \\
D_p \end{bmatrix}
\]

We next remove any linearly dependent rows of \( \Phi \) and the corresponding rows of \( \Lambda \), and discard all zero entries of \( \Delta \). For simplicity, we preserve the same notation for the matrices so obtained. The resulting \( \Phi \) has full row rank, equal to \( q \), say. It is then easy to verify that, using the resulting matrices (14), the inequality constraint (8) is equivalent to (12).

The problem of minimising (3) for (1) (equivalently, (6)) subject to (8) is denoted in compact form by
\[
\mathcal{P}_x (x) : \begin{bmatrix} \nu_x \nu (x) \\
\nu_u (x) \end{bmatrix} = \min \begin{bmatrix} \nu_x (x, u) \\
\nu_u (x, u) \end{bmatrix},
\]
where
\[
\nu_x (x, u) = \arg \min \nu_u (x, u),
\]
is the resulting optimal control vector, and $V^\text{OPT}(x)$ is the optimal value function, that is, the value of (6) (equivalently, (3)) at $u = u^\text{OPT}(x)$.

The RHC technique then proceeds as follows. At time $k$, and with initial state $x(k) = x$, $\mathcal{P}(x)$ is solved. The first control move $u^\text{OPT}(k; x)$ in (16) is the control applied to the system (1) at time $k$, that is,

$$u(k) = u^\text{OPT}(k; x),$$

and then the whole procedure is repeated at time $k+1$ with the new initial state $x(k+1) = x$. In this way the RHC state feedback control law $K_d(\cdot)$ is defined by (17). Since the model (1) and the performance index (3) are time invariant, the resulting RHC feedback law is also time invariant. Hence, the initial time in the open-loop optimal control problem may be taken to be $k = 0$, that is,

$$K_d : x \mapsto u^\text{OPT}(0; x).$$

Exponential stability of the system (1)-(2) with the RHC control law (15)-(17) can be guaranteed when the terminal cost function $x^T P x$ in (3) is properly chosen and the constraints in (10)-(11) are fixed with time (i.e., $\sigma_l = \sigma$ and $\Delta_l = \Delta$, for all $j$) see e.g., [14,3,10,8]). This can be done by ensuring that the horizon $N$ is sufficiently large such that the predicted terminal state $x^\text{OPT}(k+N; x)$ (where $x^\text{OPT}(\cdot; x)$ denotes the optimal finite-horizon trajectory that corresponds to the optimal control sequence (16) for initial state $x(k) = x$) belongs to a "terminal constraint set" that satisfies certain conditions. A lower bound on $N$ such that the above terminal property is satisfied for all initial conditions in a given compact set is derived in [2] based on the algorithm of [3].

It is, in general, impossible or difficult to pre-compute the mapping $K_d(\cdot)$ in closed form. Thus, common implementations of RHC compute numerically and on-line, at time $k$ and for the initial state $x(k) = x$, the optimal control move $K_d(x)$ rather than pre-computing the control law $K_d(\cdot)$. In Section 4 we will investigate cases for which it is possible to derive, analytically, an explicit expression for the mapping $K_d(\cdot)$.

III. THE GEOMETRY OF QUADRATIC PROGRAMMING

The finite-horizon optimal control problem $\mathcal{P}(x)$ in (15) is a non-dynamic quadratic programme (QP), that is, a problem that comprises the minimisation of a quadratic cost function (cf. (6)) subject to linear constraints (cf. (8)). Numerical solutions to RHC typically use active set or, more recently, interior point methods to solve this problem (see, for example, [7]). In this paper, instead, we will exploit the geometry of QP to obtain an analytical solution to the problem.

Since (6) is a quadratic function of $u$, its minimum, in the unconstrained case, is obtained by differentiating it with respect to $u$ and equating the result to zero. This yields

$$u^\text{OPT}(x) = -W^\top F x,$$

where the subscript UC stands for unconstrained. We observe that (19) defines a transformation

$$u = W^\top F x,$$

between the state-space coordinates $x \in \mathbb{R}^n$ and the space of $Nm$-vector control coordinates $u \in \mathbb{R}^{Nm}$.

In the constrained case, the geometry of QP is the key to obtaining an analytical solution of (15). Consider the equation

$$W^\top F u + 2u^\top F x = c$$

where $c$ is a constant. This defines ellipsoids in $\mathbb{R}^{Nm}$ centred at $u^\text{OPT UC}(x) = W^\top F x$. Also, (8) defines a constraint polyhedron in $\mathbb{R}^{Nm}$, $R_0$ say, inside which the optimal constrained solution $u^\text{OPT}(x)$ must lie. Then (15) can be regarded as finding the smallest ellipsoid that intersects the boundary of $R_0$ and $u^\text{OPT}(x)$ is the point of intersection. This is illustrated in Fig. 1 for the case $N = 2$, $m = 1$ (single input) and $\sigma_l = \sigma$ in (10) (i.e., input constrained case). In this case the constraint polyhedron is a square in $\mathbb{R}^2$ centred at the origin.

Consider now the following transformation

$$\tilde{u} = W^\top u,$$

In the new coordinates defined by (22), the constraint polyhedron $R_0$ is mapped into another polyhedron, denoted also by $R_0$ for simplicity of notation, and the ellipsoids (21) take the form of spheres centred at $u^\text{OPT UC}(x) = W^\top F x$. Thus (15) is transformed into the problem of finding the point in $R_0$ that is closest to $u^\text{OPT UC}(x)$ in Euclidean distance. This is qualitatively illustrated in Fig. 2 for the case $N = 2$, $m = 1$ (single input) and $\sigma_l = \sigma$ in (10).

Note from this figure that the constraint polyhedron induces a partition of the input space into regions; the analytical expression for the point $\tilde{u}^\text{OPT}(x)$ will depend on the region in which the unconstrained solution $u^\text{OPT UC}(x)$ lies. Thus, to obtain such analytical expression, we need to partition $\mathbb{R}^2$ into nine regions. The first region
is the parallelogram \( R_0 \) and the remaining regions, denoted by \( R_1 \) to \( R_8 \), are delimited by lines that are normal to the faces of the parallelogram and pass through its vertices, as shown in Fig. 2. Then, the optimal constrained solution \( \mathbf{u}^{\text{opt}}(x) \) is determined by the region in which the optimal unconstrained solution \( \mathbf{u}^{\text{opt}}(x) \) lies, in the following way: First, it is clear that \( \mathbf{u}^{\text{opt}}(x) = \mathbf{u}^{\text{opt}}_{\text{uc}}(x) \), if \( \mathbf{u}^{\text{opt}}_{\text{uc}}(x) \in R_0 \), that is, the optimal constrained solution coincides with the optimal unconstrained solution in \( R_0 \). Next, the optimal constrained solution in each of the regions \( R_1, R_3, R_5 \) and \( R_7 \) is simply equal to the vertex that is contained in the region. Finally, the optimal constrained solution in the regions \( R_2, R_4, R_6 \) and \( R_8 \) is defined by the orthogonal projection of \( \mathbf{u}^{\text{opt}}_{\text{uc}}(x) \) onto the faces of the parallelogram. This can be seen from Fig. 2, where a case in which the solution falls in \( R_8 \) is illustrated. Note that the only feasible points in each region \( R_i, i > 0 \), are those that lie on its intersection with the boundary of the constraint polyhedron.

Once the analytical expression for the solution is obtained in the \( \mathbf{u} \)-coordinates as described above, the solution of (15) readily follows via the transformations (20), (22), namely

\[
\mathbf{u} = -W^{-1/2} F x. \tag{23}
\]

Notice that (23) projects the solution obtained above in the \( \mathbf{u} \)-coordinates onto the state space, by defining the corresponding partition in the \( x \)-coordinates.

In the following section we will formalise and generalise this procedure to higher dimensional spaces.

### IV. SOLUTION OF \( P_N(x) \)

As qualitatively outlined in Section 3, we will first use the transformation (22) to derive the solution in the \( \mathbf{u} \)-coordinates using geometric tools, and then employ the transformation (23) to obtain the solution, in the state space, of the constrained optimisation problem \( P_N(x) \) in (15).

Also as before, the constraint polyhedron in the \( \mathbf{u} \)-coordinates is mapped into a convex constraint polyhedron in the \( \mathbf{u} \)-coordinates. Notice that the dimension of the constraint polyhedron is \( q (q = \text{rank} \Phi) \). To each hyperface of the constraint polyhedron, we will associate an index \( \mathcal{N} = 1, ..., q \), to denote the number of variables in the corresponding hyperface that take limit values. (For example, \( \mathcal{N} = q \) corresponds to a vertex.) To each of the \( \mathcal{N} \) indices and the corresponding hyperface, we will then associate regions of \( \mathcal{N} \)-constrained regions in these regions, \( \mathcal{N} \)-constrained regions are active. For example, if \( \Phi, \Lambda \) and \( \Delta \) in (9) are given by (14), then each \( \mathcal{N} \)-constrained region corresponds to \( \mathcal{N} \) elements of the optimal output vector\(^1\) \( y^{\text{opt}}(x) = \Phi \mathbf{u}^{\text{opt}}(x) + \Lambda x \) equal to \( \pm \) the corresponding elements of \( \Delta \) in (14).

For example, for the case \( q = \mathcal{N} = 2 \) in Fig. 2, regions \( R_2, R_4, R_6 \) and \( R_8 \) are \( 1 \)-constrained regions, and regions \( R_1, R_3, R_5 \) and \( R_7 \) are \( 2 \)-constrained regions. In the \( \mathbf{u} \)-coordinates, the solution is then obtained by partitioning \( \mathbb{R}^\mathcal{N} \) into a region inside the constraint polyhedron,

\(^1\)Note that after removing linearly dependent rows of \( \Phi \) and the corresponding rows of \( \Lambda \) in (14), each output \( y \) appears in \( y^{\text{opt}}(x) \) only at and after a time equal to its relative degree.
where the optimal constrained solution \( \tilde{u}^{\text{opt}}(s) \) coincides with the optimal unconstrained solution \( u^{\text{opt}}(x) = -W^{-1/2}Fx \), (cf. (19) and (22)), and, outside the constraint polyhedron, into the \( N \)-constrained regions, where the optimal constrained solution \( \tilde{u}^{\text{opt}}(x) \) is given by the point on the corresponding \( N \)-polyhedron that is closest, in the Euclidean norm, to the elements of \( x \). The set \( \mathcal{E} \) is the set of the first \( q \) natural numbers: 
\[
\mathcal{E} = \{ 1, 2, \ldots, q \}. 
\]

The ordered set \( \ell \) of \( N \) indices, \( 1 \leq N \leq q \), selected from \( \mathcal{E} \):
\[
\ell = \{ \ell_1, \ell_2, \ldots, \ell_N \} \quad (24) 
\]
where
\[
\ell_i \in \{ 1, \ldots, q-(N-1) \}, 
\quad \ell_i \in \{ \ell_i+1, \ldots, q-(N-2) \}, \ldots, 
\quad \ell_i \in \{ \ell_{s_i}+1, \ldots, q-(N-k) \}, \ldots, 
\quad \ell_i \in \{ \ell_{s_i}+1, \ldots, q \}. 
\]
The set \( \ell \) identifies the constraints that are active in each region.

• The set difference
\[
s = \mathcal{E} - \ell = \{ s_1, s_2, \ldots, s_{N^*} : s_i \notin \mathcal{E} \} 
\]
selected. If \( A \) is a vector \((n_2 = 1)\) we write \( A(\tilde{\mathcal{E}}) \) with a single argument. For example, for the matrix \( \Phi \) defined in (14) and the sets (24) and (25), \( \Phi(\ell, s) \) is the submatrix of \( \Phi \) formed by selecting the rows with indices in \( \ell \) and the columns with indices in \( s \).

Let \( \Phi^s \) be the right inverse of \( \Phi \) (e.g., its pseudoinverse) and let \( \Phi^v \) be any \( Nm \times (Nm - q) \) matrix such that its columns are normal to the rows of \( \Phi \) and the matrix \( [\Phi^s, \Phi^v] \) is a basis of \( \mathbb{R}^{nm} \) (e.g., the last \( Nm - q \) columns of the matrix \( \Phi \) of the singular value decomposition \( \Phi = U^* S^* V^* \)). We define the matrix
\[
\Phi(\ell, s) = [\Phi^s, \Phi^v]. 
\]

Note that \( \Phi(\ell, s) \) is \((q \times 1)\) (the empty matrix) if \( q = Nm \) and \( s = \emptyset \).

Next, we introduce the matrices
\[
L_{\ell, s} \triangleq \begin{bmatrix} l \end{bmatrix} 
\quad \text{if} \quad \Phi(\ell, s) = [\cdot, \cdot], 
\]

\[
L_{\ell, s} \triangleq [\Phi(s, s)W^* \Phi(s, s)]^{-1}\Phi^sW, 
\]

where \( W \) is given in (7), \( \Phi(s, s) = [\Phi(s, s \cup \ell) \Phi] \) and \( s \cup \ell \) is the union of \( s \) and \( \ell \). Note that, from its definition, if \( \Phi(\ell, s) \neq \emptyset \), then
\[
L_{\ell, s} \Phi = L_{\ell, s} [\Phi^s(\ell, s) \Phi^v] = I_{Nm-q}. 
\]

where \( L_{\ell, s} \) denotes the identity matrix of dimension \( r \times r \). We also define, for \( k = 1, 2, \ldots, N \), the row vector
\[
L_{\ell, s} \triangleq L_{\ell, s} \ell_{s-k+1}. 
\]

Given \( \ell \) defined in (24) and \( \Delta \) in (14), consider the vector \( \Lambda(\ell) \) and rename its elements \( \Delta_1, \Delta_2, \ldots, \Delta_N \). We define the set \( V(\ell) \) as the set of vertices in \( \mathbb{R}^N \) of the polyhedron \([-\Delta_1, \Delta_1] \times [-\Delta_2, \Delta_2] \times \cdots \times [-\Delta_N, \Delta_N] \). For example, if \( \Delta(\ell) = [\Delta_1, \Delta_2]^T \), then
\[
V(\ell) = \left[ \begin{array}{cc} \Delta_1 & \Delta_2 \\ -\Delta_1 & -\Delta_2 \\ \vdots & \vdots \\ -\Delta_N & -\Delta_N \end{array} \right]. 
\]

Then, given \( \ell \) and \( s \) defined in (24) and (25), respectively, and a vertex \( u \in V(\ell) \), we define the \((q \times 1)\) vectors
\[
\Delta^s = \begin{bmatrix} I_{s(s)} & 0 \\ I_{s(s)} & \Delta(s) \end{bmatrix}, 
\]

\[
\Delta^v = \begin{bmatrix} I_{s(s)} & 0 \\ I_{s(s)} & -\Delta(s) \end{bmatrix}. 
\]

For example, let \( q = 5, N = 3, \ell = \{ 2, 3, 5 \}, s = \{ 1, 4 \}, u = [\Delta_1, -\Delta_2, -\Delta_3] \in V(\ell) \) and rename \( \Delta_{1, 2} \) the
elements of $\Delta(x)$. Then $\Delta^*_q$ and $\Delta_q^*$ are given by

$$
\Delta^*_q = [\Delta_1 - \Delta_2 \prec \Delta_3 \prec \Delta_4],
$$

$$
\Delta_q^* = [-\Delta_1 \prec \Delta_2 \prec \Delta_3 \prec \Delta_4].
$$

We also need the following result.

**Lemma 4.1.** Let $1 \leq N \leq q - 1$, and consider any triple $(\ell, s, u)$, where $\ell$ and $s$ are defined in (24) and (25), respectively, and $u \in V(s, \ell)$. Construct $\Delta^*_q$ and $\Delta_q^*$ as in (31) and (32), and consider the $N$-hyperplane in $\mathbf{u}$-coordinates

$$
\Phi(\ell, \cdot) W^{-1/2} \mathbf{u} = \lambda (\ell, \cdot),
$$

with $\Phi$ and $\Lambda$ from (14), and $W$ from (7). Let $\Phi'$ be a right inverse of $\Phi$ and let $L_\ell$ be as defined in (27). Then each of the equalities

$$
L_W^{1/2} \mathbf{u} = L_\ell \Phi'(\Delta^*_q - \Lambda x),
$$

$$
L_W^{1/2} \mathbf{u} = L_\ell \Phi'(\Delta_q^* - \Lambda x),
$$

defines a hyperplane normal to the $N$-hyperplane (33). Moreover, each of the hyperplanes defined by the first $q - N$ rows of (34) and (35) contains one of the $(N + 1)$-hyperplanes that border the intersection of (33) with the constraint polyhedron.

**Proof.** First, it is easy to show that the rows of the matrix $\Phi(\ell, \cdot) W^{-1/2}$ are normal to the rows of the matrix $\Phi(\ell, \cdot) W^{1/2}$. Indeed, using the definition of $\Phi'(\ell, \cdot)$ from (26), we have

$$
\Phi(\ell, \cdot) W^{1/2} \Phi' = \Phi(\ell, \cdot) [\Phi'(\cdot, s) \Phi'] = 0_{s \times (q - \min(q, s))}.
$$

The first zero on the right hand side is because $\Phi \Phi' = L_\ell$, and $s$ and $\ell$ are disjoint sets. The second zero is because, by construction, the columns of $\Phi'$ are normal to the rows of $\Phi$. Since, from (27), $L_W^{-1/2} = [\Phi'(\cdot, s) \Phi']^{-1} \Phi(\ell, \cdot) W^{1/2}$, then (36) shows that each of (34), (35) defines a hyperplane normal to the $N$-hyperplane defined by (33).

Next, we observe that, in the $u$-coordinates, a vector $\hat{u}$ that belongs to each $(N + 1)$-hyperplane that borders the intersection of (33) with the constraint polyhedron is such that the vector $\Phi \hat{u} + \Lambda x$ has the elements with indices in $\ell$ equal to $u_\ell$ and the element with index $s_k$, $k = 1, 2, \ldots, q - N$, equal to $\Delta(s_k) - \Delta(s_k)$, or $-\Delta(s_k)$. It thus satisfies

$$
\Phi(\ell, \cdot) \hat{u} = \lambda (\ell, \cdot) x,
$$

$$
\Phi(s_k, \cdot) \hat{u} = \pm \Delta(s_k) - \Delta(s_k) x, \quad k = 2, \ldots, 1 - N.
$$

Consider first the case $\Phi(s_k, \cdot) \hat{u} = \Delta(s_k) - \Delta(s_k) x$. Any such $\hat{u}$ can be expressed as

$$
\hat{u} = \Phi(\cdot, \ell) [u - \Lambda (\ell, \cdot) x] + \Phi'(\cdot, s) \Delta(s_k) - \Delta(s_k) x]
$$

$$
+ \Phi'(\cdot, s) z, \quad z = \Phi' z,
$$

where $s - s_k = [s_1, s_2, \ldots, s_k, s_{k+1}, \ldots, s_{q-1}]$, and where $z_i \in R^{+}$. Here, $z_i$ are some vectors. Equivalently

$$
\hat{u} = \Phi(\cdot, \ell) [u - \Lambda (\ell, \cdot) x] + \Phi'(\cdot, s) u_i + \Phi' z,
$$

where

$$
u_i = \begin{bmatrix} I_{s \times (s - k)} & \Delta(s_i) - \Delta(s_k) x \\ I_{s \times (s - k)} z_i \end{bmatrix},
$$

and where $s_k - k = [1, 2, \ldots, k - 1, k + 1, \ldots, q - N]$. We will show that row $k$, $k = 1, 2, \ldots, q - N$, of equality (34) is satisfied by all the vectors $\hat{u} = \Phi W^{1/2} \mathbf{u}$, with $\mathbf{u}$ of the form (37), (38). Using (37) in the left hand side of (34) yields

$$
L_W^{1/2} \mathbf{u} = L_\ell \hat{u}
$$

$$
= L_\ell \Phi'(\cdot, \ell) [u - \Lambda (\ell, \cdot) x] + L_\ell [\Phi'(\cdot, s) \Phi'] U_i z_i
$$

$$
= L_\ell \Phi'(\cdot, \ell) [u - \Lambda (\ell, \cdot) x] + U_i z_i
$$

where the last line follows from (29). On the other hand, using (31) we can evaluate the right hand side of (34) as

$$
L_\ell \Phi'(\cdot, \ell) [u - \Lambda (\ell, \cdot) x] + \Delta(s_k) - \Delta(s_k) x 
$$

where

$$
\Phi(\ell, \cdot) W^{-1/2} \Phi' = \Phi(\ell, \cdot) [\Phi'(\cdot, s) \Phi'] = 0_{s \times (q - \min(q, s))}.
$$

Noting that row $k$ of (38) is equal to $\Delta(s_k) - \Delta(s_k)$, it is immediate to see that row $k$ of (39) is equal to row $k$ of (40), for $k = 1, 2, \ldots, q - N$. Thus, the hyperplane defined by row $k$, $k = 1, 2, \ldots, q - N$, of equality (34) contains the $(N + 1)$-hyperplane that corresponds to the element with index $s_k$ of the vector $\Phi \hat{u} + \Lambda x$ equal to $\Delta(s_k)$. A similar analysis holds for equality (35). The result then follows.

The following theorem gives the solution of $P_\ell(x)$ in (15) with $\ell$ and $M$ given in (9).

**Theorem 4.2 (Closed-Form Solution of $P_\ell(x)$).** Consider the matrices defined in (26)-(28). For $\mathcal{N} = 1, 2, \ldots, q$ where $q = \text{rank} \Phi$, form the sets $\ell$ and $s$ as in (24) and (25), respectively; for each pair $(\ell, s)$ consider all vertices $u = [u_1, u_2, \ldots, u_{s-1}] \in V(s, \ell)$; for each
triple \((\ell, s, t)\) form \(\Delta_T^s\) and \(\Delta_T^t\) as in (31) and (32). The triple \((\ell, s, t)\) defines the region \(X^T_{\ell,s,t}\) characterised by (41).

Then, if \(x \in X^T_{\ell,s,t}\), the optimal constrained control \(u^{OPT}(x)\) in (16) is given by

\[
u^{OPT}(x) = \begin{bmatrix} \Phi(\ell) \backslash \Lambda(\ell) \\ L_x \end{bmatrix} x + \begin{bmatrix} \Phi(\ell) \\ L_x \end{bmatrix}^T u - \begin{bmatrix} \begin{bmatrix} \Phi(\ell) \backslash \Lambda(\ell) \\ L_x \end{bmatrix}^T W \end{bmatrix} u^{OPT}(x) = -W^{-1}Fx,
\]

inside the constraint polyhedron defined by \(-\Delta \leq \langle -\Phi W^{-1} - \Lambda \rangle x \leq \Delta\), where \(\Delta\) is given in (14).

**Proof.** The solution is obtained by partitioning \(\mathbb{R}^{N\ell}\), outside the constraint polyhedron, in \(N\)-constrained regions, parameterised by the triples \((\ell, s, t)\). By considering all possible combinations of these triples, it is easy to see that the number of \(N\)-constrained regions is \(\left\lfloor \frac{q}{N} \right\rfloor^2\), and that there is a total of \(N\) regions outside the constraint polyhedron.

We now show that each \(N\)-constrained region is characterised by (41).

**1-Constrained Regions.**

These are delimited by:

**Limit (a).** The hyperplane that contains the corresponding \(1\)-hyperface. For \(\ell = \{\ell\}\), this hyperplane is defined by

\[
\Phi(\ell)W^{-1}u + \Lambda(\ell)x = \begin{cases} \Delta(\ell) & \text{if } u = (\ell) \\ -\Delta(\ell) & \text{if } u = -\Delta(\ell) \end{cases},
\]

Then the inequality that defines limit (a) in the state space is obtained upon substitution of (23) in (45) as

\[
[-\Phi(\ell)W^{-1}F + \Lambda(\ell)x] \begin{cases} \geq \Delta(\ell) & \text{if } u = \Delta(\ell) \\ \leq -\Delta(\ell) & \text{if } u = -\Delta(\ell) \end{cases}
\]

(46)

Note that for \(s \cup \ell_t = \ell_s\) and so replacing \(s \cup \ell_t\) in (26) we obtain \(\Phi(\ell_s) = [\Phi(\ell_t)^{-1}, \ldots, \Phi(\ell_t)^{-1}, W]\), which is non-singular. Hence, from (28)

\[
L_{\ell,s} = [\Phi(\ell_t)^{-1}, \ldots, \Phi(\ell_t)^{-1}, W]^{-1} \Phi(\ell_s) = [\Phi(\ell_t)^{-1}]^{-1},
\]

and thus,

\[
[L_{\ell,s} (I_s) - \Phi(\ell_s)] = [I_s, 0_{\Phi(\ell_t)^{-1}}] - [I_s, 0_{\Phi(\ell_t)^{-1}}] = 0_{q \times q}
\]

The above implies \(L_{\ell,s} (I_s) = \Phi\), and hence the first set of inequalities in (41) (for \(k = N = 1\) and \(\ell_t = 1, 2, \ldots, q\)) are equal to (46).

**Limit (b).** Each hyperplane normal to the \(1\)-hyperface and containing one of the \(2\)-hyperfaces that border the \(1\)-hyperface.

From Lemma 4.1 these hyperplanes are defined by the first \(q - 1\) rows of (34) and (35). Then, substituting (23) in (34) and (35) shows that the inequalities that define limit (b) in the state space are equal to the second set of inequalities in (41).

**\(N\)-Constrained Regions.**

These are delimited by:

**Limit (a).** Each hyperplane that contains the corresponding \(\bar{N}\)-hyperface and is normal to one of the \((\bar{N} - 1)\)-hyperfaces that border the \(\bar{N}\)-hyperface.

From Lemma 4.1, each row of the equalities

\[
L_{\ell,s}W^{-1}u = L_{\ell,s} \Phi(\Delta^s - \Lambda) \quad (47)
\]

defines a hyperplane that is normal to the \((\bar{N} - 1)\)-hyperface that corresponds to the vector \(\Phi \Delta - \Lambda\) having the element with index \(\ell_i, i \neq k\) equal to the \(k\)th element of \(u\). Also, from the proof of Lemma 4.1 we know that the rows of (47)

\[
L^W_{\ell,s}W^{-1}u = L_{\ell,s} (\ell_t - k + 1)\Phi(\Delta^s - \Lambda) \quad (48)
\]

\[
L^W_{\ell,s}W^{-1}u = L_{\ell,s} (\ell_t - k + 1)\Phi(\Delta^s - \Lambda) \quad (49)
\]
define the hyperplanes that contain the $N$-hyperface corresponding to $\{\Phi \dot{u} + \Lambda x(\ell_s) = \Delta \ell_s\}$ and the $N$-hyperface corresponding to $\{\Phi \dot{u} + \Lambda x(\ell_s) = -\Delta \ell_s\}$, respectively. By varying $k$ from 1 to $N$, we consider all the $(N-1)$-hyperfaces that border the $N$-hyperface. It is then clear that limit (a) is given by the first set of inequalities in (41).

**Limit (b).** Each hyperplane normal to the $N$-hyperface and containing one of the $(N+1)$-hyperfaces that border the $N$-hyperface.

From Lemma 4.1 these hyperplanes are defined by the first $q-N$ rows of (34) and (35). Then, substituting (23) in (34) and (35) shows that the inequalities that define limit (b) in the state space are equal to the second set of inequalities in (41).

**q-Constrained Regions.**

In these regions the solution lies on a vertex of the constraint polyhedron and hence the region is defined by limit (a) only.

**Limit (a).** The derivation of this limit is the same as for the $N$-constrained regions with $N < q$.

**Limit (b).** Since $N = q$, we have that the second set of inequalities in (41) is not present in this case.

We now show that the optimal control inside each region (41) has the form (42). Indeed, the optimal constrained control in each of the $N$-constrained regions is obtained by intersecting the $N$-hyperface (33) with the hyperplane normal to it and passing through $\bar{u}^{\text{opt}}(x)$. That is, $\bar{u}^{\text{opt}}(x)$ satisfies both equation (33), or equivalently, in terms of the $u$-coordinates

$$
\Phi(\ell_s)u^{\text{opt}}(x) = v - A(\ell_s)x,
$$

and the equation (cf. (34), (35))

$$
L_s W^{-1/2} u^{\text{opt}}(x) = L_s W^{-1/2} u^{\text{opt}}_{uc}(x)
$$

or, in terms of the $u$-coordinates, and substituting $u^{\text{opt}}_{uc}(x) = -W^{-1}Fx$ from (19):

$$
L_s u^{\text{opt}}(x) = -L_s W^{-1}Fx.
$$

Combining (48) and (49) proves that the optimal solution is given by (42).

Finally, it is straightforward to see that the solution in the region inside the constraint polyhedron defined by (44) is the unconstrained solution (43).

The theorem is then proved.

Theorem 4.2 gives the solution to the finite-horizon optimal control problem (15) that RHC solves at each sampling instant. The RHC law (18) is then simply obtained by selecting the first $m$ elements of $\bar{u}^{\text{opt}}(x)$ in (42) in each region of the form (41). Specifically, if $x \in X_1$, the RHC law (17) is obtained from (42) as

$$
K_c(x) = [I_m \quad 0 \cdots 0] \bar{u}^{\text{opt}}(x).
$$

Thus, to retrieve the complete solution in the state space, we compute the region partition as described in Theorem 4.2. If $n < Nm$, then the transformation $u^{\text{opt}}_{uc}(x) = -W^{-1}Fx$ spans a lower dimensional subspace of $R^{Nm}$ and so some of the regions $X_{k_s}$ in (41) will be empty. Hence, the partition has to be post-processed to eliminate redundant inequalities and empty regions.

For clarity, we present the result of Theorem 4.2 in the following algorithm.

**Algorithm**

**A. Solution of $\mathcal{P}_N(x)$ in (15) with (9)**

Initialise variables with the unconstrained solution:

- Build the matrices of inequalities (a) in (41) and store in gains.
- for $N = 1$ to $q$ do
  - for each $\ell_s$ as in (24), (25) do
    - Build $L_s$ as in (27).
    - Build the set $V_1(\ell)$ as explained after (30).
    - $X_{k_s} \leftarrow \emptyset$.
      - for each $u \in V_1(\ell)$ do
        - Build $\Delta_c$ and $\Delta_1$ as in (31) and (32).
        - Build the matrices of inequalities (b) in (41) and store in $X_{k_s}$.
      - end for
      - if $X_{k_s} \neq \emptyset$ then
        - Store $X_{k_s}$ in gains.
        - Compute controller matrices from (42), (50) and store in gains.
      - end if
    - end for
  - end for
- end for
Table 1. Computation times (in seconds) for random examples.

(a) Algorithm A

<table>
<thead>
<tr>
<th>q</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.11</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>3</td>
<td>0.27</td>
<td>0.33</td>
<td>0.22</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>1.04</td>
<td>1.65</td>
<td>1.65</td>
<td>1.32</td>
</tr>
</tbody>
</table>

(b) Algorithm of Bemporad et al.

<table>
<thead>
<tr>
<th>q</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.44</td>
<td>0.49</td>
<td>0.55</td>
<td>1.43</td>
</tr>
<tr>
<td>3</td>
<td>1.15</td>
<td>2.08</td>
<td>1.75</td>
<td>5.06</td>
</tr>
<tr>
<td>4</td>
<td>2.31</td>
<td>5.87</td>
<td>3.68</td>
<td>15.93</td>
</tr>
</tbody>
</table>

We have implemented Algorithm A in Matlab 4. The computation of each polyhedral region \( X_k \) in (41) is performed using the Geometric Bounding Toolbox for Matlab 4 (by S. Veres and S. Hermansmeyer), which gives a compact representation of the regions and detects the empty ones. The total number of regions that have to be computed (including the empty ones) is \( 1 + N_e = 3^q \) (with \( N_e \) as in the statement of Theorem 4.2). Compare with \( N_e \leq N_{max} = \sum_{i=0}^{q} 3^i \) in [2], e.g., for \( q = 3 \), we compute \( N_e = 27 \) regions whereas the upper bound of [2] is \( N_{max} = 31288 \) (although \( N_{max} \) is a worst-case estimate that can be conservative). In Table 4(a) we show computation times obtained for several random examples of single-input systems \((m = 1)\) of different orders \( n = 2, 3, 4, 5 \), with input constraints of the form (10), for which we varied the prediction horizon \( N \) (note that \( q = Nm = N \) in this case). These were evaluated using Matlab 4 on a Pentium II-450 MHz computer. For comparison, we reproduce computation times reported in [2], evaluated using Matlab 5.3 on a Pentium III-650 MHz computer. Although this comparison should be considered with caution due to the variations of computer platform speed, coding details, etc, it suggests that both methods have similar performance for short constraint horizons. For large constraint horizons and small \( n \) the efficiency of our algorithm decreases since the number of empty regions increases. The algorithm of [2], on the other hand, may perform better in cases where the upper bound \( N_{max} \) is indeed overestimating the number of regions actually examined.

We illustrate the result with a numerical example.

Example 4.1. Consider the system

\[
y = \frac{2}{s^2 + s + 2} u, \tag{51}
\]

which we discretise using a zero-order hold and a sampling period of 0.1. The resulting discrete-time system has a realization (1) with matrices

\[
A = \begin{bmatrix} 0.8955 & -0.1897 \\ 0.9048 & 0.9903 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0948 \\ 0.0004 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 2 \end{bmatrix}.
\]

In the performance index (3) we take \( N = 4, Q = C^T C \) and \( R = 0.01 \). The terminal cost matrix \( P \) is chosen as the solution of the algebraic Riccati equation

\[
P = A^T PA + Q - K^T R K,
\]

where \( K = \Phi^{-1} B^T PA \) and \( R = \Phi R + R^T P \). We consider input constraints of the form (10), with \( m = 1, N = 4 \), and \( \sigma = \sigma = 2 \). These choices yield constraints of the form (8), (9), with \( \Phi = I_n, \Lambda = 0_{n \times 2} \) and \( D = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix} \).

The state-space partition for this case, computed using Algorithm A, is shown in Fig. 3(a). A “zoom” of this partition is shown in Fig. 3(b) to display the smaller regions in more detail. The region denoted by \( X_i \) is the constraint polyhedron; regions \( X_1, X_2 \) and \( X_3 \) are 1-constrained regions; regions \( X_1 \) and \( X_2 \) are 2-constrained regions; region \( X_i \) is a 3-constrained region; finally, \( X_i \) is the union of 1, 2, 3, and 4-constrained regions.

The resulting RHC law (50) is

\[
K_i(x) = G_i x + h_i, \quad \text{if} \ x \in X_i, \quad i = 0, ..., 7, \tag{53}
\]

where

\[
G_0 = [-4.4650 \quad 13.5974], \quad h_0 = 0
\]
\[
G_1 = 0, \quad h_1 = -2
\]
\[
G_2 = [-5.6901 \quad 15.9529], \quad h_2 = -0.7894
\]
\[
G_3 = [-4.9226 \quad 13.8202], \quad h_3 = -0.4811
\]
\[
G_4 = [-4.5946 \quad 13.3346], \quad h_4 = -0.2684
\]
\[
G_5 = [-6.6778 \quad 16.8644], \quad h_5 = -1.7057
\]
\[
G_6 = [-5.1778 \quad 13.4855], \quad h_6 = -0.9355
\]
\[
G_7 = [-7.4034 \quad 16.8111], \quad h_7 = -2.6783
\]

and similar expressions in the remaining unlabeled regions, which can be obtained by symmetry.

The state space trajectory for the discretised system with initial condition \( x(0) = [-1.2, 0.53]^T \) and control (53) is shown in Fig. 4. The trajectory starts in region \( X_1 \) and moves successively into regions \( X_0, X_1, X_1, X_0 \) and stays in \( X_0 \) thereafter. Table 4.1 shows the trajectory points \( x(k) \) for \( k = 0, ..., 6 \), the regions \( X_i \) such that \( x(k) \in X_i \) and the corresponding RHC controls computed using (53).
Table 2. Example 4.1: Trajectory $x(k)$, $k = 0, \ldots, 6$, regions $X_i$ such that $x(k) \in X_i$, and corresponding RHC controls (53).

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x(k)$</th>
<th>Region $X_i$</th>
<th>RHC control $u(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$[-1.2000, 0.5300]^{T}$</td>
<td>$X_4$</td>
<td>$[-4.5946, 13.3346]{x(k)} - 0.2684$</td>
</tr>
<tr>
<td>1</td>
<td>$[-1.3480, 0.4023]^{T}$</td>
<td>$X_5$</td>
<td>$[-5.1778, 13.4855]{x(k)} - 0.9355$</td>
</tr>
<tr>
<td>2</td>
<td>$[-1.2247, 0.2735]^{T}$</td>
<td>$X_5$</td>
<td>$[-6.6778, 16.8644]{x(k)} - 1.7057$</td>
</tr>
<tr>
<td>3</td>
<td>$[-0.9722, 0.1637]^{T}$</td>
<td>$X_1$</td>
<td>$[0, 0]{x(k)} + 2.0000$</td>
</tr>
<tr>
<td>4</td>
<td>$[-0.7120, 0.0796]^{T}$</td>
<td>$X_1$</td>
<td>$[0, 0]{x(k)} + 2.0000$</td>
</tr>
<tr>
<td>5</td>
<td>$[-0.4630, 0.0209]^{T}$</td>
<td>$X_0$</td>
<td>$[-4.4650, 13.5974]{x(k)}$</td>
</tr>
<tr>
<td>6</td>
<td>$[-0.2495, -0.0146]^{T}$</td>
<td>$X_0$</td>
<td>$[-4.4650, 13.5974]{x(k)}$</td>
</tr>
</tbody>
</table>

Finally, to see how the partitions are affected by the constraint horizon, we take, successively, $N = 2$, $N = 3$, $N = 4$ and $N = 5$ in the performance index (3). The state-space partitions corresponding to each value of $N$ are shown in Fig. 5.

V. SOLUTION OF $\mathcal{P}_N(x)$ FOR MORE GENERAL CASES

The solution of $\mathcal{P}_N(x)$ in (15) given in Section 4 covers many cases of practical interest. Indeed, by varying the matrices $\Phi$ and $\Lambda$ in (9), we can consider general types of constraints, such as combinations of input, output, magnitude and rate constraints. The result presented so far, however, is limited by the allowed number $q = \text{rank} \Phi$ of constraints, which is restricted to be less than or equal to the total number of control moves $Nm$.

We turn now to the case where there are more constraints than the number of controls moves. We will consider $\mathcal{P}_N(x)$ under constraints of the form (8) with

$$L = \begin{bmatrix} L_o & 0 \\ -L_o & 0 \end{bmatrix}, \quad M = \begin{bmatrix} M_x & \cdots & -M_x \\ \vdots & \ddots & \vdots \\ M_x & \cdots & M_x \end{bmatrix},$$

where $L_o$ and $M$ are matrices of appropriate dimensions. Then, we have

$$x(k+1) = Ax(k) + Bu(k) + e(k),$$

where $e(k)$ is the disturbance vector, $A$ and $B$ are matrices of appropriate dimensions, and $x(k)$ is the state vector. The solution of $\mathcal{P}_N(x)$ in (15) given in Section 4 covers many cases of practical interest. Indeed, by varying the matrices $\Phi$ and $\Lambda$ in (9), we can consider general types of constraints, such as combinations of input, output, magnitude and rate constraints. The result presented so far, however, is limited by the allowed number $q = \text{rank} \Phi$ of constraints, which is restricted to be less than or equal to the total number of control moves $Nm$.

We turn now to the case where there are more constraints than the number of controls moves. We will consider $\mathcal{P}_N(x)$ under constraints of the form (8) with

$$L = \begin{bmatrix} L_o & 0 \\ -L_o & 0 \end{bmatrix}, \quad M = \begin{bmatrix} M_x & \cdots & -M_x \\ \vdots & \ddots & \vdots \\ M_x & \cdots & M_x \end{bmatrix},$$

where $L_o$ and $M$ are matrices of appropriate dimensions. Then, we have

$$x(k+1) = Ax(k) + Bu(k) + e(k),$$

where $e(k)$ is the disturbance vector, $A$ and $B$ are matrices of appropriate dimensions, and $x(k)$ is the state vector.

Finally, to see how the partitions are affected by the constraint horizon, we take, successively, $N = 2$, $N = 3$, $N = 4$ and $N = 5$ in the performance index (3). The state-space partitions corresponding to each value of $N$ are shown in Fig. 5.
where $L_a$, $M_a$ and $M_b$ have dimensions $t \times Nm$, $t \times 1$ and $t \times n$, respectively, with $t > Nm$. For example, combined input and output constraints of the form (10) and (11) are obtained with the selections

$$L_a = \begin{bmatrix} I_{Nm} \\ \Phi \end{bmatrix}, \quad M_a = \begin{bmatrix} \sigma \\ \Lambda \end{bmatrix}, \quad M_b = \begin{bmatrix} 0_{n \times t} \\ \Lambda \end{bmatrix},$$

where $\sigma$ is defined in (13), and $\Phi$, $\Lambda$, $\Delta$ are as given in (14) (after removing any linearly dependent rows of $\Phi$ and the corresponding rows of $\Lambda$, and discarding all zero entries of $\Delta$).

Each set of constraints, on the input and on the output, defines a constraint polyhedron, $R_i^u$ and $R_i^y$, respectively, and the optimal solution must lie inside the polyhedron defined by the intersection $R_i^u \cap R_i^y$. Note that $R_i^u \cap R_i^y$ can be empty for some values of the initial state $x$ because, in the $u$-coordinates, the input constraint polyhedron $R_i^u$ is fixed and centered at the origin (the rows of $M_a$ corresponding to input constraints are zero) while the output constraint polyhedron $R_i^y$ “moves” with $x$. In these cases, there is no feasible solution for those values of the state $x$. In the cases where the intersection $R_i^u \cap R_i^y$ is non-empty, the constraint polyhedron that it defines can have a complex representation that makes it difficult to extend the analytical approach of the previous sections. This is true in general, i.e., the closed-form solution in the case where there are more constraints than the number of controls moves would require to find a representation for the constraint polyhedron defined by the intersection of all constraints. Instead of doing this, we will provide an algorithm that finds the solution by considering all possible sets of $Nm$ constraints and uses for each set the results of the previous section.

Assume that $L_a$ in (54) has rank $q \leq Nm$ and consider each of the

$$\eta = \begin{pmatrix} t \\ q \end{pmatrix}$$

sets of $q$ rows taken from $L_a$ and the corresponding ones from $M_a$ and $M_b$. Denote the resulting submatrices by $L_j^u$, $M_j^u$ and $M_j^b$, respectively, $j = 1, 2, \ldots, \eta$. We
assume that $L^*_j$ has rank $q$. Then $\mathcal{P}_j(x)$ together with each of the $\eta$ sets of constraints defines a finite-horizon optimal control sub-problem having $q \leq Nm$ linearly independent constraints. (If rank $L^*_j < q$ for some $j$s then the corresponding sets can be discarded since even if any of them provides the optimal solution, the latter will also be picked by a set of constraints of rank $q$. Hence $\eta$ is actually an upper bound on the number of sub-problems to consider). The algorithm is then as follows.

B. Solution of $\mathcal{P}_j(x)$ in (15) with (54)

I. Off-line computations.

Initialise storage variables: $\text{REGIONS} \leftarrow \emptyset$, $\text{GAINS} \leftarrow \emptyset$.

for $j = 1$ to $\eta$ do

   (i) Select the $j$th sets of regions and gains from $\text{REGIONS}$ and $\text{GAINS}$ and evaluate $u^{\text{OPT}}(x)$ using (42).

   (ii) Build $L$ and $M$ as in (54) with $\Phi = L^*_j$, $\Lambda = M^*_j$, and $\Delta = M_j$.

   (iii) If $L u^{\text{OPT}}(x) \leq M$ (i.e., $u^{\text{OPT}}(x)$ satisfies all constraints) then

      (iii.i) Let $j = j + 1$, $u^* = u^{\text{OPT}}(x)$.

   else

      (iii.ii) Let $j = j + 1$.

end if

II. On-line computations for each state $x$.

Let $j = 1$ and $u^* = \text{infeasible}$.

while $j \leq \eta$ do

   (i) Select the $j$th sets of regions and gains from $\text{REGIONS}$ and $\text{GAINS}$ and evaluate $u^{\text{OPT}}(x)$ using (42).

   (ii) Build $L$ and $M$ as in (54) with $\Phi = L^*_j$, $\Lambda = M^*_j$, and $\Delta = M_j$.

   if $L u^{\text{OPT}}(x) \leq M$ (i.e., $u^{\text{OPT}}(x)$ satisfies all constraints) then

      (ii.i) Let $j = j + 1$, $u^* = u^{\text{OPT}}(x)$.

   else

      (ii.ii) Let $j = j + 1$.

end if

end while

Then, the optimal constrained control is $u^{\text{OPT}}(x) = u^*$ (which may be infeasible, if the outcome of the if evaluation in Algorithm B is false for $j = 1, \ldots, \eta$). Observe that in step (iii), due to the convexity of the constraint polyhedron, the algorithm terminates when a solution $u^{\text{OPT}}(x)$ is found which satisfies all constraints, since such a solution is the global minimum.

Hence, whereas Algorithm A gives the solution in closed form, the proposed Algorithm B requires a set of (at most) $\eta$ iterations for each $x$, and does not therefore produce a closed-form solution. It is interesting to compare its “on-line” performance with that of conventional methods for on-line optimisation (such as active constraint methods for quadratic programming). Towards this end, we provide the results of a test based on random examples of single-input systems ($m = 1$) of different orders $n = 2, 3, 4, 5$, with input and output constraints of the form (55), for prediction horizons $N = 2, 3, 4$. For each system we grid the state space in an area of interest containing the origin, and, for each point of the grid, we evaluate the optimal solution $u^{\text{OPT}}(x)$ using both Algorithm B and Matlab’s QP. At step (i) of the on-line part II of Algorithm B, we use a simple table look-up implementation that retrieves the optimal solution $u^{\text{OPT}}(x)$ by searching all the regions of the $j$th partition in one step. Then, for each algorithm, we measure computation time and number of floating point operations (flops), and average the latter two measures over the grid. For Algorithm B we only consider the on-line part II, since the off-line computations (part I) are only performed once and the results stored. The results of the test are shown in Table 3. We observe from this table that for the cases $n = 2, 3, 5$, there is a slight advantage of Algorithm B over QP, whereas for the case $n = 4$, QP performs a little better than Algorithm B. Overall, the results of the test indicate that the numerical performance of both algorithms is comparable, at least for problems with short horizon. For larger horizons, one could improve Algorithm B’s performance by implementing the table look-up at each iteration in a binary search tree. Examples reported in [12] show that, by employing a binary search tree for the table look-up, the algorithm presented here significantly outperforms QP in a number of situations.

We now illustrate the use of Algorithm B with an example that combines input and output constraints.

**Example 5.1.** Consider again the system of Example 4.1. In the performance index (3), we take $N = 2$, $Q = C^T C$ and $R = 0.01$. The terminal cost matrix $P$ is chosen as the

<table>
<thead>
<tr>
<th>$n$</th>
<th>$N$</th>
<th>Alg. B</th>
<th>QP</th>
<th>Alg. B</th>
<th>QP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>3.7</td>
<td>172</td>
<td>727</td>
</tr>
<tr>
<td>3</td>
<td>1.6</td>
<td>4.1</td>
<td>380</td>
<td>1915</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.4</td>
<td>4.9</td>
<td>1126</td>
<td>3738</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
<td>6.5</td>
<td>58</td>
<td>678</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
<td>6.7</td>
<td>64</td>
<td>1782</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3.4</td>
<td>7.2</td>
<td>70</td>
<td>3531</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
<td>6.0</td>
<td>1348</td>
<td>868</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5.2</td>
<td>6.4</td>
<td>4342</td>
<td>2210</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>7</td>
<td>12068</td>
<td>4131</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16.2</td>
<td>19.8</td>
<td>106</td>
<td>856</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18.0</td>
<td>23.1</td>
<td>131</td>
<td>2240</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16.0</td>
<td>20.3</td>
<td>154</td>
<td>4152</td>
<td></td>
</tr>
</tbody>
</table>
solution of the algebraic Riccati equation (52). We consider combined input and output constraints of the form (10) and (11), with $m = p = 1$, $N = 2$, $\sigma_0 = \sigma = 2$, $r_1 = 1$ and $\Delta_0 = \Delta = 0.5$. These choices yield constraints of the form (8), (54), (55), with $t = 4$ and $q = 6$ in (56), so, for each $x$, Algorithm B has, at most, six iterations that use the following matrices:

$$
L_i = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix},
M_i = \begin{bmatrix}
2 & 0 \\
0 & 0
\end{bmatrix},
M_i^+ = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix},
L_i^+ = \begin{bmatrix}
1 & 0 \\
0 & 0
\end{bmatrix},
M_i^+ = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix},
M_i^+ = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix},
L_i^+ = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix},
M_i^+ = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix}. 
$$

The solutions for each of the cases $j = 1, 3, 4, 5, 6$ follow by direct application of Algorithm A, yielding state-space partitions similar to those of the top left plot in Fig. 5 (in fact, the latter plot corresponds precisely to the case $j = 1$). The case $j = 2$, on the other hand, can be discarded because the matrix $L_2$ in (57) has rank $1 < q = Nm = 2$.

For a time-domain simulation, we start with the initial condition $x(0) = [-1.6773, 0.2161]^T$. For times $k = 0, 1, \ldots, 5$, the optimal solution $u^{OPT}(k) = [2, 2]$ is provided by case $j = 1$, corresponding to the matrices in the first line of (57) (the other cases give, in fact, infeasible solutions). At time $k = 6$, the algorithm has also only one iteration because the optimal solution $u^{OPT}(x) = [2, 2]$ is provided by the case $j = 1$ (at this time, however, cases $j = 2, 3$ give the same feasible solution, and the remaining cases give infeasible solutions). For $k \geq 7$, the solution is unconstrained. The time-domain simulations are displayed in Fig. 6 for both the output $y$ and the control input $u$.

VI. CONCLUSIONS

For linear, time-invariant, discrete-time models with a quadratic performance index and linear constraints, we have presented a closed-form expression for the global solution to the RHC problem. The resulting solution consists of a partition of the state space into regions in which the corresponding control law has an affine analytical form. Both the regions and the control law are characterised in terms of the parameters of the open-loop optimal control problem that underlies RHC. The solution is applicable to cases where the total number of constraints is less than or equal to the number of degrees of freedom, represented by the number of control moves. When the total number of constraints is greater than the number of degrees of freedom, the solution is addressed via an algorithm that iteratively uses the off-line solution and avoids on-line optimisation.

REFERENCES


Maria M. Seron was born in Rosario, Argentina in 1963. She received the Electronic Engineer degree from the Universidad Nacional de Rosario, Argentina in 1988 and the Ph.D. degree from The University of Newcastle, Australia in 1996. From 1988 to 1992 she was a lecturer at the Department of Electrical Engineering, Universidad Nacional de Rosario, Argentina. She held post-doctoral positions with the Department of Electrical Engineering, The University of Newcastle, Australia, with the Centre for Systems Engineering and Applied Mechanics (CESAME), Catholic University of Louvain, Belgium, and with the Center for Control Engineering and Computation, University of California, Santa Barbara, USA. From 1999 to 2002 she was an Associate Professor with the Department of Electronic Engineering, Universidad Nacional de Rosario, Argentina. She is currently an Australian Research Fellow with the Department of Electrical Engineering and Computer Science, University of Newcastle, Australia.

Graham C. Goodwin obtained a B.Sc (Physics), B.E (Electrical Engineering), and Ph.D. from the University of New South Wales. From 1970 until 1974 he was a lecturer in the Department of Computing and Control, Imperial College, London. Since 1974 he has been with the Department of Electrical and Computer Engineering, The University of Newcastle, Australia. He is the co-author of seven monographs; four edited volumes, and several hundred technical papers.

Graham Goodwin is the recipient of several international prizes including the USA Control Systems Society 1999 Hendrik Bode Lecture Prize, a Best Paper award by IEEE Trans. Automatic Control, and Best Engineering Text Book award from the International Federation of Automatic Control. He is currently Professor of Electrical Engineering and Associate Director of the Centre for Integrated Dynamics and Control at the University of Newcastle, Australia. Graham Goodwin is the recipient of an ARC Federation Fellowship; a Fellow of IEEE; an Honorary Fellow of Institute of Engineers, Australia; a Fellow of the Australian Academy of Science; a Fellow of the Australian Academy of Technology, Science and Engineering; a Member of the International Statistical Institute; and a Fellow of the Royal Society, London.

José A. De Doná was born in Argentina in 1965. He received his B.E. degree from Universidad Nacional del Comahue, Argentina in 1989, and his Ph.D. degree from the University of Newcastle, Australia in 2000. During the year 2000 he held
a postdoctoral position from the European Nonlinear Control Network in the Universities of Liege and Gent, Belgium. During the year 2001 he held a Research Academic position with the Centre for Integrated Dynamics and Control, The University of Newcastle, Australia. Since 2002 he works as a Lecturer in the School of Electrical Engineering and computer Science, The University of Newcastle, Australia.