Multi-parametric toolbox (MPT)

Conference Paper *in* Lecture Notes in Computer Science · January 2004

Source: DBLP

Michal Kvasnica
Slovak University of Technology in Bratislava

Multi-Parametric Toolbox (MPT)

Michal Kvasnica, Pascal Grieder, Mato Baotić, and Manfred Morari

Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH)
CH-8092 Zurich, Switzerland,
{kvasnica,grieder,baotic,morari}@control.ee.ethz.ch

Abstract. A Multi-Parametric Toolbox (MPT) for computing optimal or suboptimal feedback controllers for constrained linear and piecewise affine systems is under development at ETH. The toolbox offers a broad spectrum of algorithms compiled in a user friendly and accessible format: starting from different performance objectives (linear, quadratic, minimum time) to the handling of systems with persistent additive disturbances and polytopic uncertainties. The algorithms included in the toolbox are a collection of results from recent publications in the field of constrained optimal control of linear and piecewise affine systems [10,13, 4,9,16,17,15,14,7].

1 Introduction

Optimal control of constrained linear and piecewise affine (PWA) systems has garnered great interest in the research community due to the ease with which complex problems can be stated and solved. The aim of the *Multi-Parametric Toolbox* (MPT) is to provide efficient computational means to obtain feedback controllers for these types of constrained optimal control problems in a MATLAB programming environment. By multi-parametric programming a linear or quadratic optimal control problem is solved off-line as a function of the initial state as a parameter. The associated solution takes the form of a PWA state feedback law. In particular, the state-space is partitioned into polyhedral sets and for each of those sets the optimal control law is given as one affine function of the state. In the on-line implementation of such controllers, computation of the controller action reduces to a simple set-membership test, which is one of the reasons why this method has attracted so much interest in the research community.

As shown first for quadratic [8] and then for linear [4] objectives, a feedback controller may be obtained for constrained linear systems by applying multi-parametric quadratic and linear programming techniques, respectively. The multi-parametric algorithms for constrained finite time optimal control (CFTOC) of linear systems contained in the MPT are based on [1] and are similar to [29]. Both [1] and [29] give algorithms that are significantly more efficient than the original procedure proposed in [8].

It is current practice to approximate the constrained infinite time optimal control (CITOC) by receding horizon control (RHC) - a strategy where the

R. Alur and G.J. Pappas (Eds.): HSCC 2004, LNCS 2993, pp. 448–462, 2004.

CFTOC problem is solved at each time step, and then only the initial value of the optimal input sequence is applied to the plant. The main problem of RHC is that it does not, in general, guarantee stability or constraint satisfaction. In order to obtain these properties, certain conditions have to be added to the original problem [25]. The extensions to guarantee these properties are a part of the MPT. It is furthermore possible to impose a minimax optimization objective which allows for the computation of robust controllers for linear systems subject to polytopic uncertainties and additive disturbances [6,20]. As an alternative to computing suboptimal stabilizing controllers, the procedures to compute the infinite time optimal solution for constrained linear systems [13] are also provided.

Optimal control of piecewise affine systems has received great interest in the research community since PWA systems represent a powerful tool for approximating non-linear systems and because of their equivalence to hybrid systems [18]. The algorithms for computing the feedback controllers for constrained PWA systems were presented for quadratic and linear objectives in [10] and [3] respectively, and are also included in this toolbox. Instead of computing the feedback controllers which minimize a finite time cost objective, it is also possible to obtain the infinite time optimal solution for PWA systems [2].

Even though the multi-parametric approaches rely on off-line computation of a feedback law, the computation can quickly become prohibitive for larger problems. This is not only due to the high complexity of the multi-parametric programs involved, but mainly because of the exponential number of transitions between regions which can occur when a controller is computed in a dynamic programming fashion [10,21]. The MPT therefore also includes schemes to obtain sub-optimal controllers of low complexity for linear and PWA systems as presented in [16,14,15,17].

In addition to control tools, the toolbox also provides extensive functionality for polytope manipulation [31]: convex hulls, convex unions and envelopes, Minkowski sums, Pontryagin differences, as well as many other operations can be performed efficiently by MPT. Most of the functionality supports both single polytopes and non-convex unions thereof. Other commercial software, such as the Geometric Bounding Toolbox (GBT) [30], provides similar functionality on the level of polytope manipulation. MPT explicitly takes advantage of object-oriented programming to provide a transparent and easy to use interface. The toolbox is provided free and is based on state of the art optimization packages (compatibility with CPLEX [19], NAG [26], SeDuMi [28], CDD [11] and more).

2 Problem Description and Properties

Polytopic (or, more general, polyhedral) sets are an integral part of multi-parametric programming. For this reason we give some definitions and fundamental operations with polytopes. For more details we refer reader to [31].

Definition 1 (polyhedron). *A convex set $\mathcal{Q} \subseteq \mathbb{R}^n$ given as an intersection of a finite number of closed half-spaces $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Q^x x \leq Q^c\}$, is called* polyhedron.

Definition 2 (polytope). *A bounded polyhedron* $\mathcal{P} \subset \mathbb{R}^n$

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}, \tag{1}$$

is called polytope.

It follows from the above definitions that every polytope represents a convex, compact (i.e., bounded and closed) set. We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$ is *full dimensional* if $\exists x \in \mathbb{R}^n : P^x x < P^c$. Furthermore, if $\|(P^x)_i\| = 1$, where $(P^x)_i$ denotes i-th row of a matrix P^x, we say that the polytope $\mathcal{P}$ is *normalized.* One of the fundamental properties of a polytope is that it can also be described by its vertices

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_P} \alpha_i V_P^{(i)},\ 0 \leq \alpha_i \leq 1,\ \sum_{i=1}^{v_P} \alpha_i = 1\}, \tag{2}$$

where $V_P^{(i)}$ denotes the i-th vertex of $\mathcal{P}$, and v_P is the total number of vertices of $\mathcal{P}$. We will henceforth refer to the half-space representation (1) and vertex representation (2) as $\mathcal{H}$ and $\mathcal{V}$ representation respectively.

Definition 3 (face). *Linear inequality $a'x \leq b$ is called* valid *for a polyhedron $\mathcal{P}$ if $a'x \leq b$ holds for all $x \in \mathcal{P}$. A subset $\mathcal{F}$ of a polyhedron is called a* face *of $\mathcal{P}$ if it is represented as $\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid a'x = b\}$, for some valid inequality $a'x \leq b$. The faces of polyhedron $\mathcal{P}$ of dimension* 0, 1, $(n-2)$ *and* $(n-1)$ *are called vertices, edges, ridges and facets, respectively.*

We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$ is in a *minimal representation* if the removal of any of the rows in $P^x x \leq P^c$ would change it (i.e., there are no redundant halfspaces). It is straightforward to see that a normalized, full dimensional polytope $\mathcal{P}$ has a *unique* minimal representation. This fact is very useful in practice. Normalized, full dimensional polytopes in minimal representation allow us to avoid any ambiguity when comparing them and very often speed-up other polytope manipulations. We will now define some of the basic manipulations on polytopes.

2.1 Polytope Manipulation

The Set-Difference of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a union of polytopes $\mathcal{R} = \bigcup_i \mathcal{R}_i$

$$\mathcal{R} = \mathcal{P} \setminus \mathcal{Q} := \{x \in \mathbb{R}^n \mid x \in \mathcal{P}, x \notin \mathcal{Q}\}. \tag{3}$$

The Pontryagin-Difference of two polytopes $\mathcal{P}$ and $\mathcal{W}$ is a polytope

$$\mathcal{P} \ominus \mathcal{W} := \{x \in \mathbb{R}^n \mid x + w \in \mathcal{P},\ \forall w \in \mathcal{W}\}. \tag{4}$$

The Minkowski addition of two polytopes $\mathcal{P}$ and $\mathcal{W}$ is a polytope

$$\mathcal{P} \oplus \mathcal{W} := \{x + w \in \mathbb{R}^n \mid x \in \mathcal{P},\ w \in \mathcal{W}\}. \tag{5}$$

The convex hull of a union of polytopes $\mathcal{P}_i \subset \mathbb{R}^n$, $i = 1, \dots, p$, is a polytope

$$\operatorname{hull}\left(\bigcup_{i=1}^{p} \mathcal{P}_i\right) := \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{p} \alpha_i x_i,\ x_i \in \mathcal{P}_i,\ 0 \le \alpha_i \le 1,\ \sum_{i=1}^{p} \alpha_i = 1\}. \tag{6}$$

The envelope of two $\mathcal{H}$-polyhedra $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \le P^c\}$ and $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Q^x x \le Q^c\}$ is an $\mathcal{H}$-polyhedron

$$\operatorname{env}(\mathcal{P}, \mathcal{Q}) = \{x \in \mathbb{R}^n \mid \bar{P^x} x \le \bar{P^c},\ \bar{Q^x} x \le \bar{Q^c}\}, \tag{7}$$

where $\bar{P^x} x \le \bar{P^c}$ is the subsystem of $P^x x \le P^c$ obtained by removing all the inequalities not valid for the polyhedron $\mathcal{Q}$, and $\bar{Q^x} x \le \bar{Q^c}$ are defined in the similar way with respect to $Q^x x \le Q^c$ and $\mathcal{P}$ [7].

2.2 Multi-parametric Programming

This section first covers some of the fundamentals of multi-parametric programming for linear systems before restating results for PWA systems. Consider a discrete-time linear time-invariant system

$$x(t+1) = Ax(t) + Bu(t) \tag{8a}$$
$$y(t) = Cx(t) + Du(t) \tag{8b}$$

with $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. Let $x(t)$ denote the state at time t and $x_{t+k|t}$ denote the predicted state at time $t + k$ given the state at time t. For brevity we denote $x_{k|0}$ as x_k. Let u_k be the computed input for time k, given $x(0)$. Assume now that the states and the inputs of the system in (8) are subject to the following constraints

$$x \in \mathbb{X} \subset \mathbb{R}^n, \qquad u \in \mathbb{U} \subset \mathbb{R}^m \tag{9}$$

where $\mathbb{X}$ and $\mathbb{U}$ are compact polyhedral sets containing the origin in their interior, and consider the constrained finite-time optimal control (CFTOC) problem

$$J_N^*(x(0)) = \min_{u_0, \dots, u_{N-1}} ||Q_f x_N||_\ell + \sum_{k=0}^{N-1} ||Ru_k||_\ell + ||Qx_k||_\ell \tag{10a}$$
$$\text{subj. to} \quad x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U}, \qquad \forall k \in \{1, \dots, N\}, \tag{10b}$$
$$x_N \in \mathcal{X}_{set}, \tag{10c}$$
$$x_0 = x(0),\ x_{k+1} = Ax_k + Bu_k,\ \forall k \in \{0, \dots, N-1\}, \tag{10d}$$
$$\text{if } \ell = 2, \text{then } Q = Q' \succeq 0, \quad Q_f = Q_f' \succeq 0, \quad R = R' \succ 0 \tag{10e}$$

where (10c) is a user defined set-constraint on the final state which may be chosen such that stability of the closed-loop system is guaranteed [25]. The cost (10a) may be linear (e.g., $\ell \in \{1, \infty\}$) [4] or quadratic (e.g., $\ell = 2$) [8] whereby the matrices Q, R and Q_f represent user-defined weights on the states and inputs.

Definition 4. *We define the N-step feasible set $\mathcal{X}_f^N \subseteq \mathbb{R}^n$ as the set of initial states $x(0)$ for which the CFTOC problem (10) is feasible, i.e.*

$$\mathcal{X}_f^N = \{x(0) \in \mathbb{R}^n \mid \exists (u_0, \dots, u_{N-1}) \in \mathbb{R}^{Nm}, \\ x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U},\ \forall k \in \{1, \dots, N\}\}. \quad (11)$$

For a given initial state $x(0)$, problem (10) can be solved as an LP or QP for linear or quadratic cost objectives respectively. However, this type of on-line optimization may be prohibitive for control of fast processes. As shown in [8,4, 9], problem (10) can be solved for all parameters $x(0) \in \mathcal{X}_f^N$ to obtain a feedback solution with the following properties,

Theorem 1. *[8,9] Consider the CFTOC problem (10). Then, the set of feasible parameters $\mathcal{X}_f^N$ is convex, the optimizer $U_N^* : \mathcal{X}_f^N \to \mathbb{R}^{Nm}$ is continuous and piecewise affine (PWA), i.e.*

$$U_N^*(x(0)) = F_r x(0) + G_r \quad \textit{if} \quad x(0) \in \mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\},\ r = 1, \dots, R \quad (12)$$

and the optimal cost $J_N^* : \mathcal{X}_f^N \to \mathbb{R}$ is continuous, convex and piecewise quadratic ($\ell = 2$) or piecewise linear ($\ell \in \{1, \infty\}$).

According to Theorem 1, the feasible state space $\mathcal{X}_f^N$ is partitioned into R polytopic regions, i.e., $\mathcal{X}_f^N = \{\mathcal{P}_r\}_{r=1}^R$. An approach was presented in [8], but more efficient algorithms for the computation are given in [1,29]. With sufficiently large horizons or appropriate terminal set constraints (10c) the closed-loop system is guaranteed to be stabilizing for receding horizon control [13,25]. However, no robustness guarantees can be given. This issue is addressed in [20,6] where the authors present minimax methods which are able to cope with additive disturbances

$$x(t+1) = A(\lambda)x(t) + B(\lambda)u(t) + w(t), \qquad w(t) \in \mathcal{W}, \quad (13)$$

where $\mathcal{W}$ is a polytope with the origin in its interior. The minimax approach can be applied also when there is polytopic uncertainty in the system dynamics,

$$\varOmega := \mathrm{conv}\big\{[A^{(1)}|B^{(1)}], [A^{(2)}|B^{(2)}], \dots, [A^{(L)}|B^{(L)}]\big\}, \quad [A(\lambda)|B(\lambda)] \in \varOmega, (14)$$

i.e., there exist L nonnegative coefficients $\lambda_l \in \mathbb{R}$ ($l = 1, \dots, L$) such that

$$\sum_{l=1}^{L} \lambda_l = 1\ , \qquad [A(\lambda)|B(\lambda)] = \sum_{l=1}^{L} \lambda_l [A^{(l)}|B^{(l)}]. \quad (15)$$

The set of admissible λ can be written as $\varLambda := \{x \in [0,1]^L \mid ||x||_1 = 1\}$. In order to guarantee robust stability of the closed loop system, the objective (10a) is modified such that the feedback law which minimizes the worst case is computed, hence the name *minimax* control.

The results in [8] were extended in [5,10,3] to compute the optimal explicit feedback controller for PWA systems of the form

$$x(k+1) = A_i x(k) + B_i u(k) + f_i, \tag{16a}$$
$$\text{s.t. } L_i x(k) + E_i u(k) \leq W_i, \quad i \in I \tag{16b}$$
$$\text{if } [x'(k) \;\; u'(k)]' \in \mathcal{D}_i \tag{16c}$$

whereby the dynamics (16a) with the associated constraints (16b) are valid in the polyhedral set $\mathcal{D}_i$ in (16c). The set $I \subset \mathbb{N}$, $I = \{1, \ldots, d\}$ represents all possible dynamics, and d denotes the number of different dynamics. Henceforth, we will abbreviate (16a) and (16c) with $x(k+1) = f_{PWA}(x(k), u(k))$. Note that we do not require $x(k+1) = f_{PWA}(x(k), u(k))$ to be continuous. The optimization problem considered here is then given by (10) whereby the dynamics (10d) and constraints in (9) are replaced by (16).

For PWA systems, the constraint (10c) is user-specified and imposed on the terminal state in order to guarantee stability [24,14,9]. As an alternative, the infinite horizon optimal controller for PWA systems guarantees stability as well [2]. In order to robustify controllers with respect to additive disturbances, a minimax approach is taken [21] which is identical to what was proposed for linear systems [20,9].

All multi-parametric programming methods suffer from the curse of dimensionality. As the prediction horizon N increases, the number of partitions R ($\mathcal{X}_f^N = \{\mathcal{P}_r\}_{r=1}^R$) grows exponentially making the computation and application of the solution intractable. Therefore, there is a clear need to reduce the complexity of the solution. This was tackled in [15,16,14] where the authors present two methods for obtaining feedback solutions of low complexity for constrained linear and PWA systems. The first controller drives the state in minimum time into a convex set $\mathcal{X}_{set}$, where the cost-optimal feedback law is applied [16,14]. This is achieved by iteratively solving one-step multi-parametric optimization problems. Instead of solving one problem of size N, the algorithm solves N problems of size 1, thus the decrease in both on-line and off-line complexity. This scheme guarantees closed-loop stability. If a linear system is considered, an even simpler controller may be obtained by computing a controller for prediction horizon $N = 1$, with the additional constraint that $x_1 \in \mathcal{X}_f^N$ [16,17]. In order to guarantee stability of this closed-loop system, an LMI analysis is performed which aims at constructing a Lyapunov function [15,17].

3 MPT Content

3.1 References

Aside from standard polytopic-manipulation functions, at this stage the toolbox contains the algorithms presented in the following references. Additions will be made in the future to cover a wider spectrum of the literature.

- Set difference computation and convexity recognition of the union of polyhedra [7].

- Multi-parametric solvers for quadratic objectives for constrained linear [1] and PWA [10] systems.
- Invariant set computation of linear [12] and PWA systems [15].
- Computation of invariant sets for linear [22] and PWA systems [27] subject to bounded disturbances.
- Stability analysis of PWA systems through LMIs [15].
- Robust stability analysis of PWA systems through LMIs [17].
- Multi-parametric computation of robust feedback controllers for constrained linear systems [6].
- Multi-parametric computation of low complexity controllers for constrained linear [16] and PWA [14] systems.

In order to utilize the full functionality of MPT, an LMI solver is needed therefore. The SDP solver interface Yalmip [23] is included in MPT and we recommend to its use in combination with SeDuMi [28], though other LMI solvers are compatible as well (see [23] for details). In order to perform certain polytopic manipulations (e.g., vertex-enumeration, Minkowski-Addition) in dimensions larger than 3, it is necessary to have CDD [11] installed. CDD [11] is also included in MPT. The MPT furthermore supports different LP and QP solvers, namely MATLAB's `linprog` and `quadprog`, the LP and QP solvers of the Numerical Algorithms Group (NAG) as well as the CPLEX, GLPK and CDD solvers. If the user wishes to utilize another LP or QP solver, a simple modification to the functions `mpt_solveLP` and `mpt_solveQP` will do.

3.2 Classes and Basic Polytope Manipulations

The toolbox defines a new class `polytope` inside the MATLAB programming environment along with overloaded operators which are presented [1] in Table 1. The functions for polytope manipulations are given in Table 2. All functions take either polytopes or unions thereof as an input argument which is illustrated in the following example:

Example 1.

```
>> P=polytope([eye(2);-eye(2)],[1 1 1 1]');      %Create Polytope P
>> W=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]'); %Create Polytope W
>> DIF=P-W;                                    %Pontryagin difference P-W
>> ADD=P+W;                                    %Minkowski addition P+W
>> plot(ADD, P, DIF, W);                       %Plot polytopes
```

The resulting plot is depicted in Figure 1(a). When a `polytope` object is created, the constructor automatically normalizes its representation and removes all redundant constraints. Note that all elements of the polytope class are private and can only be accessed as described in the tables. Furthermore, all information on a polytope is stored in the internal polytope structure. In this way unnecessary

[1] Please note that Tables 1-3 list only a part of the functions available in the toolbox. Check the MPT manual for more details.

Table 1. Short overview of overloaded operators for the class `polytope`.

`P=polytope(Px,Pc)`	Constructor for creating the polytope $P = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$.
`[ , ]`	Concatenation of polytopes into a (non-convex) union.
`P == Q, P ~= Q`	Check if two polytopes are equal ($P = Q$), or not-equal ($P \neq Q$), respectively.
`P > Q, P < Q`	Check if $P \supset Q$ or $P \subset Q$, respectively.
`P & Q`	Intersection of two polytopes, $P \cap Q$.
`P | Q`	Convex union of polytopes, $P \cup Q$.
`P + Q, P - Q`	Minkowski sum, $P \oplus Q$ and Pontryagin difference, $P \ominus Q$.
`P \ Q`	Set difference operator.

Table 2. Functions defined for class `polytope`.

`V=extreme(P)`	Computes extreme points (vertices) of a polytope P.
`E=envelope(P,Q)`	Computes envelope E of two polytopes P and Q.
`P=hull(PA)` `P=hull(V)`	Computes hull of a polytope array `PA` or hull of an array of vertices V.
`plot(P)`	Plots a given polytope or polytope array in 2D or 3D.
`P=range(Q,A,f)`	Affine transformation of a polytope. $P = \{Ax + f \in \mathbb{R}^n \mid x \in Q\}$
`P=domain(Q,A,f)`	Compute polytope that is mapped to Q. $P = \{x \in \mathbb{R}^n \mid Ax + f \in Q\}$

repetitions of the computations during polytopic manipulations in the future can be avoided. More functions on polytopes are given in Table 2 and are illustrated in the following example.

Example 2.

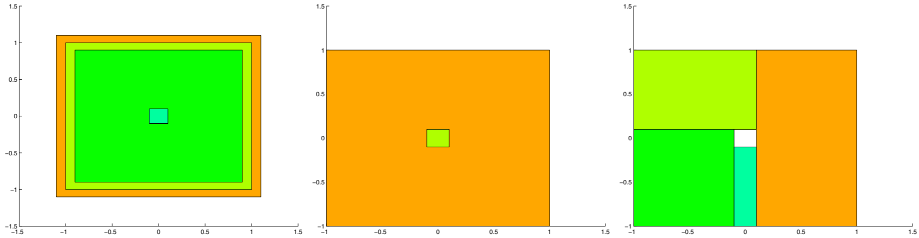
```
>> P=polytope([eye(2);-eye(2)],[1 1 1 1]');      %Create Polytope P
>> Q=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]'); %Create Polytope Q
>> D=P\Q;                       %Compute set difference between P and Q
>> U=D|Q;                       %Compute union of D and Q
>> U==P                         %Check if two polytopes are equal
    ans=1
```

The polytopes P and Q are depicted in Figures 1(b) and 1(c). The `hull` function is overloaded such that it takes both elements of the `polytope` class as well as matrices of points as input arguments.

3.3 Control Functions

This subsection will give a brief overview of the main control functions which are provided with the MPT. All functions may be called by using the accessor function

```
[ctrlStruct]=mpt_Control(sysStruct,probStruct,Options)
```



(a) The result of the `plot` call in Example 1.

(b) The sets P and Q in Example 2.

(c) The sets $P \setminus Q$ in Example 2.

Fig. 1. Results obtained for Examples 1 and 2.

which takes the structures defined in Section 3.5 as parameters and automatically calls one of the functions described below depending on the parameters which were passed. Every function returns a controller structure `ctrlStruct` which contains the regions over which the feedback law is unique, i.e. $u = F_i x + G_i$ if $x \in P(i)$. The different functions for obtaining these solutions are:

`[ctrlStruct]=mpt_optControl(sysStruct,probStruct,Options)`:
: This function solves a constrained optimal control problem as defined in (10) for linear and quadratic cost objectives for linear systems with the method proposed in [1,4].

`[ctrlStruct]=mpt_optInfControl(sysStruct,probStruct,Options)`:
: This function computes a solution to the constrained infinite-time optimal control problem for linear systems and quadratic cost objectives using the algorithm described in [13].

`[ctrlStruct]=mpt_iterative(sysStruct,probStruct,Options)`:
: This function applies the minimum time computation scheme described in [16]. The function returns the stabilizing minimum time controller as well as the controller obtained at the final iteration. The final controller guarantees feasibility for all time, but stability still needs to be checked with `mpt_getPWQLyapFunction` (see Section 3.4). This scheme can also be used to obtain robust solutions by setting the appropriate flags [15].

`[ctrlStruct]=mpt_iterativePWA(sysStruct,probStruct,Options)`:
: This function applies the minimum time computation scheme described in [14]. The function returns the stabilizing controller represented by the controller structure `ctrlStruct`. This scheme can also be used to obtain robust solutions for PWA systems affected by additive disturbance by setting the appropriate flags [21].

3.4 Analysis Functions

Various scripts which serve to plot the obtained results as well as analysis functions are included in the toolbox. Some of these function are vital in obtaining stability and feasibility properties for the low complexity controllers [16,15,14]. The corresponding functions are given in Table 3.

Table 3. Functions used for analysis purposes.

`mpt_getPWQLyapFct`	Computes a PWQ Lyapunov function for a given closed-loop system.
`mpt_getCommonLyapFct`	Computes a common quadratic Lyapunov function for a set of linear systems.
`mpt_infset`	Calculates the maximal (robust) positively invariant set for an LTI system
`mpt_infsetPWA`	Computes the maximal (robust) positive invariant subset for PWA systems

3.5 Structures and Objects

As indicated in the previous sections, the toolbox utilizes three main structures: the system structure `sysStruct`, the problem structure `probStruct`, and the controller structure `ctrlStruct`. The details of these structures are given in Tables 4, 5, and 6. System structure describes dynamics of the system and input/output constraints. The problem structure serves to state properties of a problem the user wants to solve. Finally, the controller structure encapsulates results of the calculation and can be later used to evaluate control actions.

Table 4. Fields of the system (`sysStruct`) structure.

`A, B, C, D, f, g`	State-space dynamic martices in (8) and (16a). Set elements to empty if they do not apply.
`umin, umax`	Bounds on inputs `umin` $\leq$ u(t) $\leq$ `umax`.
`dumin, dumax`	Bounds on `dumin` $\leq$ u(t)-u(t-1) $\leq$ `dumax`.
`ymin, ymax`	Constraints on the outputs `ymin` $\leq$ y(t) $\leq$ `ymax`.
`noise`	A `polytope` bounding the additive disturbance, i.e. `noise`=$\mathcal{W}$ in (13).
`Aunc, Bunc`	Cell arrays containing the vertices of the polytopic uncertainty (14).
`guardX, guardU, guardC`	Polytope cell array defining where the dynamics are active (for PWA systems). $\mathcal{D}_i = \{(x,u) \mid \texttt{guardXi}\ x + \texttt{guardUi}\ u \leq \texttt{guardCi}\}$.

3.6 Examples

In order to obtain a feedback controller, it is necessary to specify both a system as well as the problem. We demonstrate the procedure on a simple second-order double integrator, with bounded input $|u| \leq 1$ and output $||y(k)||_\infty \leq 5$:

Example 3.

```
>> sysStruct.A=[1 1; 0 1];       %x(k+1)=Ax(k)+Bu(k)
>> sysStruct.B=[0 1];            %x(k+1)=Ax(k)+Bu(k)
>> sysStruct.C=[1 0; 0 1];       %y(k)=Cx(k)+Du(k)
>> sysStruct.D=[0;0];            %y(k)=Cx(k)+Du(k)
```

Table 5. Fields of the problem (`probStruct`) structure.

`Q, R`	Weighting matrices in the cost function.
`N`	Prediction horizon. Values: 1,2,3, ...
`norm`	1/2/inf norm solution. Values: 1/2/inf. Default `2`.
`subopt_lev`	Which level of sub-optimality to use (0 = optimal solution, 1 = minimum-time solution, 2 = sub-optimal solution). Default `0`.
`x0bounds`	Impose constraints also on x_0? Values: 1 (yes)/0 (no). Default `0`.
`Tconstraint`	Which terminal constraint to use (0 = no, 1 = LQR invariant set, 2 = user defined terminal set). Default `1`.
`P_N`	Terminal cost ($x'_N P_N x_N$). Only applies if `Tconstraint`$\neq 1$.
`Tset`	A `polytope` defining the terminal set ($x_N \in$ `Tset`). Only applies if `Tconstraint` = 2.

Table 6. Fields of the controller structure (`ctrlStruct`).

`Pn`	Polyhedral partition over which the control law is defined.
`Fi,Gi`	Cell arrays containing the PWA control law, i.e. `U = Fi{m} x + Gi{m}`
`Ai,Bi,Ci`	Cell arrays containing value function `1/2 x'Ai{m}x + Bi{m}x + Ci{m}`
`Pfinal`	The maximum controllable set as a polytope object.
`details`	More details about the solution

```
>> sysStruct.umin=-1;           %Input constraints umin<=u(k)
>> sysStruct.umax=1;            %Input constraints u(k)<=umax
>> sysStruct.ymin=[-5 -5]';     %Output constraints ymin<=y(k)
>> sysStruct.ymax=[5 5]';       %Output constraints y(k)<=ymax
```

For this system we will now formulate the problem with quadratic cost objective in (10) and a prediction horizon of $N = 5$:

```
>> probStruct.norm=2;           %Quadratic Objective
>> probStruct.Q=eye(2);         %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.R=1;              %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.N=5;              %...over the prediction horizon 5
>> probStruct.subopt_lev=0;     %Compute optimal solution
```

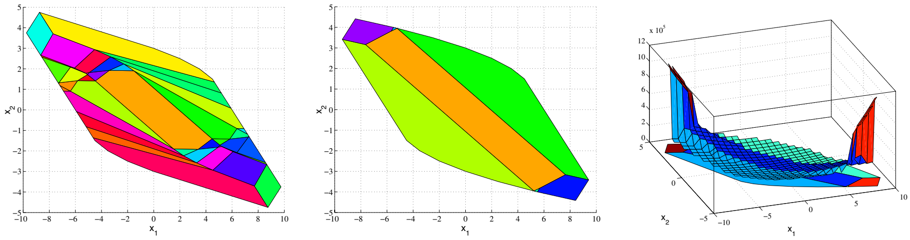
If we now call

```
%Compute feedback controller
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct);
>> mpt_plotPartition(ctrlStruct)
```

the controller[2] for the given problem is returned and plotted (see Figure 2(a)). If we wish to compute a low complexity solution, we can run the following:

```
>> probStruct.subopt_lev=2;        %Compute low complexity solution.
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct);
```

[2] Calculated in 1.2 seconds on a 2.4 GHz Pentium 4 machine using Matlab R12.1



(a) The $N = 5$ step optimal feedback solution.

(b) The iterative low complexity solution for the double integrator.

(c) Lyapunov function for the low complexity solution.

Fig. 2. Results obtained for Example 3.

```
>> [Q,L,C,feasible]=mpt_getPWQLyapFct(ctrlStruct);
>> feasible
     ans=1
>> mpt_plotPartition(ctrlStruct)
>> mpt_plotPWQ(ctrlStruct.Pn,Q,L,C)
```

The resulting partition[3] and Lyapunov function is depicted in Figures 2(b) and 2(c) respectively. In the following we will solve the PWA problem introduced in [24] by defining two different dynamics which are defined in the left- and right half-plane of the state space respectively.

Example 4.

```
%Polytopic state constraints Hx(k)<=K
>> H=[-1 1; -3 -1; 0.2 1; -1 0; 1 0; 0 -1];
>> K=[  15;    25;     9;    6;   8;   10];

%System Dynamics 1/2: x(k+1)=Ax(k)+Bu(k)+f
>> syst.A{1} = [1 0.2; 0 1]; syst.B{1} = [0; 1]; syst.f{1} = [0; 0];
>> syst.A{2} = [0.5 0.2; 0 1]; syst.B{2} = [0; 1]; syst.f{2} = [0.5; 0];

%System Dynamics 1/2: y(k)=Cx(k)+Du(k)+g
>> syst.C{1} = [1 0]; syst.D{1} = [0]; syst.g{1} = [0];
>> syst.C{2} = [1 0]; syst.D{2} = [0]; syst.g{2} = [0];

%Dynamics 1/2 defined in guardA x <= guardC
>> syst.guardA{1} = [1 0; H]; syst.guardC{1} = [  1; K];
>> syst.guardA{2} = [-1 0; H]; syst.guardC{2} = [  -1; K];

%Input/Output constraints for dynamic 1 and 2
>> syst.umin = -1; syst.umax = 1; syst.ymin = -10; syst.ymax = 10;
```

[3] Calculated in 1.9 seconds on a 2.4 GHz Pentium 4 machine using Matlab R12.1

we can now compute the low complexity feedback controller by defining the problem as follows:

```
>> probl.norm=2;              %Quadratic Objective
>> probl.Q=eye(2);            %Objective: min_U J=sum x'Qx + u'Ru...
>> probl.R=0.1;               %Objective: min_U J=sum x'Qx + u'Ru...
>> probl.subopt_lev=1;        %Compute low complexity controller.
```

and calling the control function,

```
>> [ctrlStruct]=mpt_Control(syst,probl);
>> mpt_plotPartition(ctrlStruct)
```

The result[4] is depicted in Figure 3.

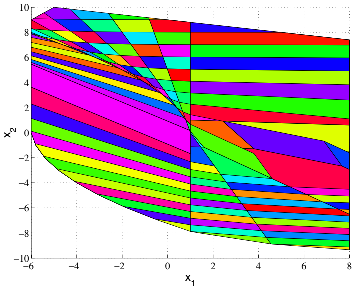


Fig. 3. Controller partition obtained for Example 4.

4 Outlook

The version of MPT presented here is an initial release which we hope to continually extend as new advances in the field of optimal control are made. We encourage all researchers which are active in related fields to contribute code to the toolbox such that the community as a whole has access to the latest developments. Details on how to contribute as well as the latest versions of MPT and a more detailed manual can be obtained from:

`http://control.ee.ethz.ch/~hybrid/mpt/`

Acknowledgment. We would like to thank all contributors to the toolbox which are not in the authors list. Specifically (in alphabetical order): Alberto Bemporad, Francesco Borrelli, Frank J. Christophersen, Eric Kerrigan, Marco

[4] Obtained after 24.9 seconds on a 2.4 GHz Pentium 4 machine using Matlab R12.1

Lüthi, Saša V. Raković, Fabio Torrisi and Kari Unneland. A special thanks goes to Komei Fukuda (cdd) and Johan Löfberg (Yalmip) for allowing us to include their respective packages in the distribution. Thanks to their help MPT truly is an 'unpack-and-use' toolbox.

References

1. M. Baotić. An efficient algorithm for multi-parametric quadratic programming. Technical Report AUT02-04, Automatic Control Laboratory, ETH Zurich, Switzerland, February 2002.
2. M. Baotić, F. J. Christophersen, and M. Morari. Infinite time optimal control of hybrid systems with a linear performance index. In *Proc. of the Conf. on Decision and Control, Maui, Hawaii, USA*, December 2003.
3. M. Baotić, F.J. Christophersen, and M. Morari. A new Algorithm for Constrained Finite Time Optimal Control of Hybrid Systems with a Linear Performance Index. In *European Control Conference*, Cambridge, UK, September 2003.
4. A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of LP-based model predictive control. In *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.
5. A. Bemporad, F. Borrelli, and M. Morari. Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.
6. A. Bemporad, F. Borrelli, and M. Morari. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Trans. Automatic Control*, 48(9):1600–1606, 2003.
7. A. Bemporad, K. Fukuda, and F.D. Torrisi. Convexity recognition of the union of polyhedra. *Computational Geometry*, 18:141–154, April 2001.
8. A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.
9. F. Borrelli. *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, 2003.
10. F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. In *Proc. 2003 American Control Conference*, Denver, Colorado, USA, June 2003.
11. K. Fukuda. *Cdd/cdd+ Reference Manual*, December 1997. `www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html`.
12. E. G. Gilbert and K. Tin Tan. Linear systems with state and control constraints: the theory and applications of maximal output admissible sets. *IEEE Trans. Automatic Control*, 36(9):1008–1020, 1991.
13. P. Grieder, F. Borrelli, F.D. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. In *Proc. 2003 American Control Conference*, Denver, Colorado, USA, June 2003.
14. P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. Low complexity control of piecewise affine systems with stability guarantee. *Submitted*, 2003.
15. P. Grieder, M. Lüthi, P. Parillo, and M. Morari. Stability & feasibility of receding horizon control. In *European Control Conference*, Cambridge, UK, September 2003.
16. P. Grieder and M. Morari. Complexity reduction of receding horizon control. In *Proc. 42nd IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2003.

17. P. Grieder, P. Parillo, and M. Morari. Robustness of receding horizon control. In *Proc. 42nd IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2003.
18. W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.
19. ILOG, Inc. *CPLEX 7.0 User Manual.* Gentilly Cedex, France, 2000.
20. E. C. Kerrigan and J. M. Maciejowski. Robustly stable feedback min-max model predictive control. In *Proc. 2003 American Control Conference*, Denver, Colorado, USA, June 2003.
21. E. C. Kerrigan and D. Q. Mayne. Optimal control of constrained, piecewise affine systems with bounded disturbances. In *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, dec 2002.
22. I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Egineering*, 4:317–367, 1998.
23. J. Löfberg. *Yalmip.* `http://www.control.isy.liu.se/~johanl/yalmip.html`.
24. D. Q. Mayne and S. Raković. Model predictive control of constrained piecewise affine discrete-time systems. *Int. J. of Robust and Nonlinear Control*, 13(3):261–279, April 2003.
25. D. Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.
26. Numerical Algorithms Group, Ltd. *NAG Foundation Toolbox for MATLAB 6.* Oxford, UK, 2002.
27. S. Raković, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari. Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances. 2003. Submitted.
28. J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, pages 625–653, October 1999.
29. P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *Proc. 40th IEEE Conf. on Decision and Control*, December 2001.
30. S. M. Veres. Geometric Bounding Toolbox (GBT) for MATLAB, 2003.
31. G. M. Ziegler. *Lectures on Polytopes.* Springer, 1994.