Algorithms for Hybrid Optimal Control

Jean-Guillaume Dumas, Aude Rondepierre

Hybrid optimal control

J.-G. DUMAS & A. RONDEPIERRE*[1]
Laboratoire de Modélisation et Calcul
Tour IRMA - BP 53, 38041 Grenoble, France

January 5, 2006

[1]*Corresponding author. Email: Aude.Rondepierre@imag.fr

Abstract

We consider a nonlinear ordinary differential equation and want to control its behavior so that it reaches a target by minimizing a cost function. Our approach is to use hybrid systems to solve this problem: the complex dynamic is replaced by piecewise affine approximations which allow an analytical resolution. The sequence of affine models then forms a sequence of states of a hybrid automaton. Given an sequence of states, we give a hybrid approximation of nonlinear controllable domains and a new algorithm computing a controllable under-approximation. We are then able to traverse the automaton till the target, locally insuring the optimality. Moreover we also present new efficient methods computing a Kalman canonical exact decomposition of any local affine systems.

0.1 Introduction

Aerospace engineering, automatics and other industries provide a lot of optimization problems, which can be described by optimal control formulations: change of satellites orbits, flight planning, motion coordination see e.g. [21] ([30] for more applications in aerospace industry). In general those optimal control problems are fully nonlinear ; since the years 1950-1970, the theory of optimal control has been extensively developed and has provided us with powerful results like dynamic programming (see [6]) or the maximum principle ([32]). A large amount of theory have been developed, and resolutions are mainly numerical.

In this paper, we consider a dynamical system which state is described by the solution of the following ordinary differential equation (ODE):

$$\begin{cases} \dot{X}(t) = f(X(t), u(t)) \\ X(0) = X_0 \end{cases} \tag{1}$$

We present a hybrid algorithm controlling the system (1) from an initial state X_0 at time $t = 0$ to a final state $X_f = 0$ at an unspecified time t_f. To reach this state, we allow the admissible control functions u to take values in a convex and compact polyhedral set $\mathbb{U}_m$ of $\mathbb{R}^m$, in such a way that:

$$J(X, u(.)) = \int_0^{t_f} l(X(t), u(t))dt \tag{2}$$

is minimized.

The dynamic programming approach of optimal control problems is essentially based on the characterization of the value function $J(X, u)$ in terms of a nonlinear Hamilton-Jacobi-Bellman (HJB) equation. The first class of techniques uses a discrete version of the dynamical principle over a time discretization of the (HJB) equation ([4, 9, 7]). However those algorithms are very expensive in high dimension. For instance, if the state dimension is n and the number of discretization points per dimension is 50 (which is the minimum acceptable: 100 could still be a bit sparse), one has to consider 50^n grid points. Despite the development of efficient techniques for the choice of the discretization points like adaptive mesh, computations grow exponentially in the state dimension. Consequently dimension 4 or 5 cannot be exceeded, while e.g. [30] requires treatments of dimensions 6 or 7. Now, with the appearance of viscosity solutions ([14, 11, 5, 12, 13]), approximations of (HJB) equations have been well improved. Unfortunately, regularity conditions are too restrictive, see [4].

The other kind of numerical techniques are based on the Pontryagin Maximum Principle ([32, 8]). This principle provides a pseudo-Hamiltonian formulation of optimal control problems. For many problems like time-optimal linear control problems, adequate solutions have been found e.g. by [8, 31, 30]. However, the main problem is actually the synthesis of optimal feedback, even not

solved for linear systems, except in some very special cases (such as time-optimal problems).

We propose a different approach: the use of hybrid systems. The idea is to approach nonlinear systems like (1) by piecewise affine models that we can analytically studied. Basically, an analytical approach must allow to improve approximations as is done by [24, 19]: the level of details allows to reach a compromise between quantitative quality of the approximation and the computational time. There are many possible linearizations by parts. For instance, one can build a virtual mesh of the phase space and use multi-dimensional interpolation to define an affine approximation of the system in each cell (simplex) of the mesh, see [15, 23, 1] for more details. One can also linearize each equation separately by implicit representation and one-dimensional linearization on each variable. The latter has been done e.g. for biological systems, where simplifications in relation to real data and in regard of simulations of the model are possible, see [19]. We here choose to use a hybrid system modeling, i.e. to approximate the original system (1) by a continuous and piecewise affine one, built on a virtual mesh of the phase space and the control space. We also propose a new way to reduce affine problem to canonical form via an enhanced block Kalman decomposition. To our best knowledge, the latter improves on previously known algorithm by an order of magnitude. Indeed, we show in section 0.2.1 how to use variants of fast characteristic polynomial algorithms to reduce the Kalman decomposition to matrix multiplication. Then, in each cell of the implicitly build automaton, the system is fully linear ($\dot{X}(t) = AX(t)+Bu(t)+c$) and can be solved with mostly analytical tools. A preliminary version of this part of this paper already appeared there: [20], we here use the same ideas but give new and more efficient algorithms for the canonical decomposition and for the boundary computations (see in particular sections 0.2.1 and 0.3.3).

The paper is organized as follows: the first part presents algorithms for the hybrid approximation of nonlinear system (§0.2.1). Next the controllability of the system is studied and an optimized algorithm computing an underapproximation of the controllable set. This set of solutions is now explicitly computed along a path to the target, with an accuracy improving on what was obtained in [33]. We then can propose the complete algorithmic resolution of the optimal control problem by the search of optimized solutions in section 0.4. Section 0.4.3 ends with an application of our algorithms to a 2-dimensional example.

0.2 Hybrid Approximation of Nonlinear Systems

In this section, we want to build a hybrid model approximating the nonlinear dynamic (1): $\dot{X}(t) = f(X(t), u(t))$. The principle is as follows: for a given mesh of the control and state space $\mathbb{R}^n \times \mathbb{U}_m$, we compute a piecewise affine approximation of the nonlinear vector field f. The partition, associated to the

so computed piecewise affine dynamic, defines a hybrid system that we can thus analyze.

In this section, we first compute the piecewise affine approximation of the nonlinear vector field f in §0.2.1. Then, for each affine system, we propose a new way to compute its Kalman canonical form, via block matrix decomposition. From then, we develop algorithms building an implicit mesh of the space $\mathbb{R}^n \times \mathbb{U}_m$ in §0.2.2. Lastly, in §0.2.3, we define our hybrid model of any system (1).

0.2.1 Hybridization

In this paragraph we want to approximate the system (1) by a system:

$$\dot{X}(t) = f_h(X(t), u(t)) \tag{3}$$

where f_h is a piecewise affine approximation of the nonlinear vector field f.

Computation of the piecewise affine approximation

Let $\Delta = (\Delta_i)_{i \in I}$ be a given simplicial mesh of the space $\mathbb{R}^n \times \mathbb{U}_m$. Any affine function in dimension $n + m$ is uniquely defined by its values at $(n + m + 1)$ affine independent points. Thus, as is done e.g. in [23], in each cell of Δ, we can compute an affine approximation f_h of f by interpolation at the vertices of the cell.

Let us consider a simplicial cell Δ_i of Δ, defined as the convex hull of its $(n + m + 1)$ vertices: $\sigma_1, \ldots, \sigma_{n+m+1}$. Let f_i be the affine approximation of f computed by interpolation at the vertices of Δ_i. We state:

$$f_i(X, u) = A_i X + B_i u + c_i = [\ A_i \mid B_i\ ] \begin{bmatrix} X \\ u \end{bmatrix} + c_i$$

The approximation f_i is computed by interpolation of f at the vertices of the cell Δ_i, so that: $\forall j \in \{1, \ldots, n + m + 1\},\ f(\sigma_j) = f_i(\sigma_j)$. Hence, we have:

$$\forall j \in \{1, \ldots, n + m + 1\},\ f(\sigma_j) = [\ A_i \mid B_i\ ]\sigma_j + c_i$$

which could induce e.g.:

$$\forall j = 1, \ldots, n + m + 1,\ f(\sigma_j) - f(\sigma_1) = [\ A_i \mid B_i\ ](\sigma_j - \sigma_1)$$

We thus define M_i the $(n+m) \times (n+m)$ matrix, whose columns are the vectors of vertices: $\{\sigma_j - \sigma_1; j = 2, \ldots, n+m+1\}$ and F_i the $(n+m) \times n$ matrix, whose columns are the vectors of images: $\{f(\sigma_j) - f(\sigma_1); j = 2, \ldots, n + m + 1\}$.

Hence: $F_i = [\ A_i \mid B_i\ ]M_i$. By linear independence of the vertices of the simplex Δ_i, the square matrix M_i is non singular, so that we obtain:

$$\begin{aligned} [\ A_i \mid B_i\ ] &= F_i\ M_i^{-1} \\ c_i &= f(\sigma_1) - [\ A_i \mid B_i\ ]\sigma_1 \end{aligned} \tag{4}$$

Remark 1. *The c_i can be solved as well with any column:*

$$\forall j = 1, \ldots, n+m+1,\ f(\sigma_j) - [\ A_i \mid B_i\ ]\sigma_j = c_i$$

Consequently the piecewise affine approximation of (1) is defined by:

$$f_h(X,u) = A_i X + B_i u + c_i,\ \text{if } (X,u) \in \Delta_i$$

and looks like in figure 1.

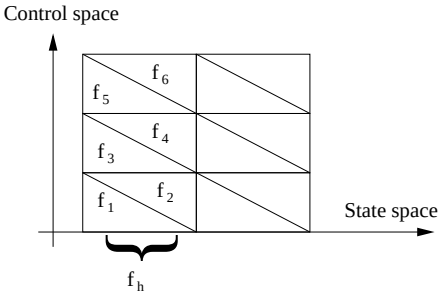


Figure 1: Definition of the piecewise affine approximation

Canonical Transformation

According to the previous hybrid approximation, in each cell of the mesh Δ, the dynamic is affine. Let us consider a general linear control system:

$$\dot{X}(t) = AX(t) + Bu(t) + c,\ X(t) \in \mathbb{R}^n,\ u(t) \in \mathbb{U}_m$$

When the system is rank deficient (i.e. $rk([B\ AB \ldots A^{n-1}B]) < n$), Kalman has shown the existence of a part of the linear system independent of the control (see [26, 25]).

Theorem 1 (([25] Canonical Structure)). *Let A and B be real matrices having respective sizes $n \times n$ and $n \times m$. There exists an invertible $n \times n$ matrix T such that:*

$$T^{-1}AT = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix} \qquad T^{-1}B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}$$

where $r = rk([B\ AB \ldots A^{n-1}B]) = rk([B_1\ A_1B_1 \ldots A_1^{n-1}B_1])$, A_1 is a $r \times r$ square matrix and B_1 a $r \times m$ matrix.

With numerical methods, or numerical inputs computing the canonical structure can be useless. Nevertheless, cases where the control is already linear and the matrices B or even A are known exactly appears quite often. In this case, it is of high interest to extract the rank deficiency and the related uncontrollable parts of the system. Therefore we here propose a new explicit and exact algorithm for the Kalman decomposition. Furthermore, and different from the initial Kalman decomposition, our canonical form returns a nearly diagonal matrix A_1, speeding up the subsequent computations.

Block Kalman Decomposition Our approach is to use block versions of the linear algebra algorithms as in [16] in order to improve the locality of the computations and treat larger problems faster. Indeed, we are then able to compute exactly the rank of the system and use the LQUP decomposition of [17] (nowadays quite as fast as numerical routines) to perform the decomposition. We already presented a block version in [20], but Gilles Villard thus remarked[1] that our fast block algorithms of [18] for the characteristic polynomial could then be applied on the Kalman matrix. To our best knowledge, this algorithm improves on previously known algorithm by an order of magnitude.

The idea, developed in cooperation with C. Pernet, is to treat the matrix K virtually, and never to compute it explicitly as follows:

1. Form a compressed matrix $\overline{K}$ selecting at most n independent columns of K.
2. Compute the Kalman form using $\overline{K}$.

Now, to enable an $O(n^\omega \log(n))$ complexity[2], the computation of $\overline{K}$ is based on the following two remarks:

- A block Gaussian elimination enables a fast computation of r independent columns in a $m \times n$ rank r matrix. Keller-Gehrig used a step-form elimination with complexity $O(n^\omega)$ [28], also known in the literature as "row-echelon", we shown in [18, 17] that a LQUP variant is more efficient in practice. In the following this subroutine will be denoted `ColReducedForm` ([18, algorithm 2.3]).
- It is possible to compute all the iterates of the Kalman matrix with only $O(n^\omega \log(n))$ operations by using Keller-Gehrig's trick:
  Compute $\lceil \log_2(n) \rceil$ squares of A $(A, A^2, \ldots, A^{2^i}, A^{2^{\lceil \log_2(n) \rceil}})$; then all the iterates of a column C can be obtained with only $O(n^\omega \log(n))$ operations with the scheme $V_0 = C, V_{i+1} = [V_i | A^{2^{i-1}} V]$.

To perform the first step of our decomposition, thus start with all the columns of B and interleave both preceding techniques so that V_{i+1} contains only independent columns at each step. This produces the compressed Kalman matrix with only $O(n^\omega \log(n)$ operations. The proof of correctness is the same as the one for the characteristic polynomial just replacing the identity in the latter by our matrix B.

Now, in [18], we were able to remove the $log(n)$ factor in the characteristic polynomial algorithm to the price of loosing the fast matrix multiplication. This gives an $O(n^3)$ complexity but a faster algorithm in practice. We can extend this algorithm to also compute the compressed Kalman matrix as follows:

The last part of the algorithm is then to recover the transformation matrix T. This is done by completing the compressed Krylov matrix to get an invertible

[1] G. Villard, 2005, personal communication

[2] ω is the exponent of fast matrix multiplication, actually between 2.3755 and 3.

Algorithm 1 LUCKM : Compressed Krylov Matrix (LU-Krylov)

Require: A a $n \times n$ matrix, B, a $n \times m$ matrix over a field
Ensure: $(\overline{K}, rank(\overline{K}))$

1: Take the first column of B, $v = B_1$.
2: $\begin{cases} K_1 = \begin{bmatrix} v & Av & A^2v & \dots \end{bmatrix} \\ (L, [U_1|U_2], P) = \texttt{LUP}(K_1^T), r_1 = \text{rank}(K_1) \end{cases}$
{The matrix K_1 is computed on the fly: at most $2r$ columns are computed}
3: **if** $(r_1 = n)$ **then**
4: Return (K_1, r_1)
5: **else**
6: $A' = PAP^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$ where A'_{11} is $r_1 \times r_1$.
7: $A_R = A'_{22} - U_2^T U_1^{-T} A'_{12}$
8: $B' = \begin{bmatrix} U_1^{-T} & 0 \\ -U_2^T U_1^{-T} & I \end{bmatrix} PB$
9: Compute the permutation Q s.t. $B'Q = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$
10: Recursively call $(K_2, r_2) = \texttt{LUCKM}(A_R, Z)$
11: $\overline{K} = \left[\begin{array}{c|c} K_v & P^T \begin{bmatrix} 0 \\ K_2 \end{bmatrix} \end{array} \right]$
12: Return $(\overline{K}, r_1 + r_2)$
13: **end if**

matrix. We use the technique of [18, theorem 2.1]: compute the LUP factorization of $\overline{K}^T$, replace $[U_1 U_2]$, the upper part of U, by $\begin{bmatrix} U_1 & U_2 \\ 0 & Id \end{bmatrix}$ and replace $[L]$ by $\begin{bmatrix} L & 0 \\ 0 & Id \end{bmatrix}$:

Theorem 2. *Let A be a $n \times n$ matrix and B a $n \times m$ matrix $n \times m$. Algorithm 2 is correct and requires $O(n^\omega logn)$ arithmetic operations using Keller-Gehrig's compression or $O(n^3)$ using algorithm 1.*

The proof of this theorem is given in appendix .1. Our implementation and constructive proof of the Kalman decomposition are based on LQUP factorization and block matrix computation. The better locality induced by this block version enables the use of very fast Basic Linear Algebra Subroutines, even with symbolic computations [17]. Therefore the computation time is improved and can nowadays match the numerical performances.

Local canonical transformation of the hybrid model Thanks to the canonical structure theorem 1, any local affine system $\dot{X}(t) = A_q X(t) + B_q u(t) + c_q$ can be replaced by another affine one:

$$\dot{Y}(t) = \begin{bmatrix} A_{q,1} & A_{q,2} \\ 0 & A_{q,3} \end{bmatrix} Y(t) + \begin{bmatrix} B_{q,1} \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} c_{q,1} \\ c_{q,2} \end{bmatrix} \tag{5}$$

Algorithm 2 Fast Block Kalman form

Require: A a $n \times n$ matrix over a field, B, a $n \times m$ matrix
Ensure: r, T, A_1, A_2, A_3, B_1 as in theorem 1

1: $(\overline{K}, r) = \texttt{CompressedKrylovMatrix}(A, B)$
2: **if** (r=n) **then**
3: Return $(n, Id, A, \emptyset, \emptyset, B)$
4: **else**
5: $(L, [U_1 U_2], P) = \mathrm{LUP}(\overline{K}^T)$
6: $\mathrm{T} = \left[\begin{array}{c|c} \overline{K} & P^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \end{array} \right]$
7: $B_1 = L^{-T} U_1^{-T} P B$
8: $A' = P A^T P^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$
9: $C_1 = L^{-T} U_1^{-T} A'_{12}$
10: $C_2 = A'_{22} - U_2^T U_1^{-T} A'_{12}$
11: **for all** j **do**
12: Let t_j be the column indexes in $\overline{K}$ of the last iterate l_j of the jth block.
13: $m_j = l_j L^{-1}_{1 \ldots t_j, 1 \ldots t_j}$
14: **end for**
15: Build the polycyclic matrix H by placing each column vectors m_j at column index t_j and adding 1 on the sub-diagonal on all other columns.
16: Return (r, T, H, C_1, C_2, B_1)
17: **end if**

via the state variable change $Y(t) = T_q^{-1} X(t)$, with $[\ c_{q,1} \mid c_{q,2}\ ]^T = T_q^{-1} c_q$.

Interpolation error of the Hybridisation scheme

In this section, we establish some properties of the piecewise affine approximation f_h. These properties are an extension of those presented in [24, §11.1] for autonomous nonlinear systems like: $\dot{x}(t) = f(x(t))$. The difference, here, in the context of nonlinear control systems $\dot{X}(t) = f(X(t), u(t))$, is that the control function is in general only measurable. In this case, the regularity of the vector field f is lost. We thus need to generalize the latter result to get a good bound on the error produced by our linearization scheme.

Let us recall that h denotes the size of the mesh Δ and is defined by:

$$h = \sup_{i \in I} h_i \quad \text{and:} \quad h_i = \sup_{x,y \in \Delta_i} \|x - y\|$$

where $\|.\|$ is the ∞-norm on R^{n+m}. We first need the continuity of f_h:

Proposition 1. *The vector field f_h is a continuous function on $\mathbb{R}^n \times \mathbb{U}_m$ and locally Lipschitz in X, i.e. for all compact subset Ω in $\mathbb{R}^n$, the restriction of f_h to $\Omega \times \mathbb{U}_m$ is Lipschitz continuous in X, uniformly in u.*

Proof. The continuity of the piecewise affine approximation f_h is the result of the proposition [24, proposition 11.1.3] applied to f and the mesh Δ of $\mathbb{R}^n \times \mathbb{U}_m$. Now, let Ω be a compact subset of $\mathbb{R}^n$. We introduce $\tilde{\Omega} = Conv(\Omega)$ the convex hull of Ω. $\tilde{\Omega}$ is so a compact and convex subset in $\mathbb{R}^n$. By applying [24, proposition 11.1.3] to the domain $\tilde{\Omega}$, we deduce that f_h is L_h-Lipschitz in X over $\tilde{\Omega}$ where: $L_h = \sup_{i \in J(\tilde{\Omega})} \|A_i\|$ and $J(\tilde{\Omega}) = \{i \in I\ ;\ \Delta_i \cap (\tilde{\Omega} \times \mathbb{U}_m) \neq \emptyset\}$. We conclude, since $\tilde{\Omega} \times \mathbb{U}_m$ is a compact in $\mathbb{R}^{n+m}$ as the Cartesian product of two compacts respectively in $\mathbb{R}^n$ and $\mathbb{R}^m$. It follows that $card\ J(\tilde{\Omega}) < +\infty$ and then the Lipschitz constant L_h is well defined. □

Then, from this continuities of f and f_h, we deduce the following bound on the magnitude of f_h:

Lemma 1. *f and f_h are locally bounded in X, uniformly in u on $\mathbb{R}^n \times \mathbb{U}_m$, i.e.:*

$$\forall R > 0,\ \exists C_R > 0,\ \forall (X,u) \in B(0,R) \times \mathbb{U}_m,\ \|f_h(X,u)\| \leq C_R$$

Moreover, if f is bounded on $\mathbb{R}^n \times \mathbb{U}_m$, then f_h is also bounded on $\mathbb{R}^n \times \mathbb{U}_m$.

Proof. The first result is immediate by noticing that f and f_h are continuous over any compact $\bar{B}(0,R) \times \mathbb{U}_m$, for $R > 0$. Let us consider $(X,u) \in \mathbb{R}^n \times \mathbb{U}_m$; there exists a cell $\Delta_i = Conv(\sigma_1, \ldots, \sigma_{n+m+1})$ of the mesh Δ of $\mathbb{R}^n \times \mathbb{U}_m$, such that $(X,u) \in \Delta_i$. The convexity thus gives (X,u) as a linear combination: $\exists (\alpha_j)_{j=1\ldots n+m+1} \in [0,1]^{n+m+1},\ (X,u) = \sum_{j=1}^{n+m+1} \alpha_j \sigma_j$, with $\sum_{j=1}^{n+m+1} \alpha_j = 1$.

In Δ_i, f_h is affine, so we have: $f_h(X,u) = \sum_{j=1}^{n+m+1} \alpha_j f_h(\sigma_j)$. According to the interpolation constraints, we have $\forall j = 1 \dots n+m+1,\ f_h(\sigma_j) = f(\sigma_j)$ and hence:

$$\begin{aligned} \|f_h(X,u)\| &= \|\sum_{j=1}^{n+m+1} \alpha_j f_h(\sigma_j)\| = \|\sum_{j=1}^{n+m+1} \alpha_j f_h(\sigma_j)\| \\ &\leq \sum_{j=1}^{n+m+1} \alpha_j \|f_h(\sigma_j)\| \leq (\sum_{j=1}^{n+m+1} \alpha_j)\ C = C \end{aligned}$$

□

With these key modifications, and under some non restrictive assumptions on the nonlinear vector field f, the interpolation error of [24] can be evaluated with respect to the size h of the mesh Δ as:

Proposition 2 (([24, proposition 11.1.1] Interpolation error)).

If f is L-Lipschitz on $\Omega \times \mathbb{U}_m$, then: $\sup_{(X,u)\in\Omega\times\mathbb{U}_m} \|f(X,u) - f_h(X,u)\| \leq \frac{2L(n+m)}{n+m+1} h$ ■

We can thus see that the key point is the definition of the mesh and of its size h. We now show that a meshing of the whole space is not mandatory. We rather compute the mesh on the fly.

0.2.2 Implicit simplicial mesh

In the previous section, we have seen how to build a piecewise affine approximation of the nonlinear dynamic f for a given simplicial mesh of the state and control domain. So to perform our hybrid approximation, we need now a method to build a simplicial mesh of $\mathbb{R}^n \times \mathbb{U}_m$.
As $\mathbb{R}^n \times U_m$ is not bounded, it's algorithmically inconceivable to mesh the whole space $\mathbb{R}^n \times \mathbb{U}_m$. Our approach is then to implicitly define a mesh of our space, so that the simplicial subdivision is made on the fly.

Let $h > 0$ be the discretization step. There exists several ways to mesh a given domain. To each mesh correspond only one piecewise affine approximation (3) by interpolation. Among all the possible meshes of the domain $\mathbb{R}^n \times U_m$, we want to select those for which $(0,0)$ is a fixed point of the system (3). Indeed, the target is $X_f = 0$ and we want to stay there once we have reached it. Therefore, $u(t_f)$ also has to be zero. A simple way to thus enforce $f_h(0,0) = (0,0)$ is to request the following property:

Property 1. *If $0 \in \Delta_i$, then 0 is a vertex of Δ_i.*

Lemma 2. *Let Δ be a mesh of $\mathbb{R}^n \times \mathbb{U}_m$ that satisfies the property 1. Then $(0,0)$ is a fixed point of the system (3) build over Δ.*

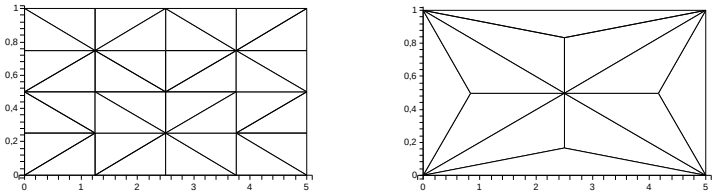
Proof. At the vertices of the mesh, $f(\sigma) = f_h(\sigma)$ by the interpolating constraints and $(0,0)$ is a fixed point of the initial system by definition. □

Moreover, we assume that the mesh Δ also satisfies the following property:

Property 2. *Let $D = (D_q)_{q\in\mathcal{Q}}$ be the projection of Δ over the state space $\mathbb{R}^n$.*

1. *D is a simplicial mesh of $\mathbb{R}^n$.*
2. *Let $(i,j) \in I^2$. We state: $D_i = p^{\perp}_{\mathbb{R}^n}(\Delta_i)$ and $D_j = p^{\perp}_{\mathbb{R}^n}(\Delta_j)$. Then we have:*
$$D_i = D_j \text{ or } \mathring{D}_i \cap \mathring{D}_j = \emptyset$$

The property 2 is a geometrical constraint on the position of the cells Δ_i in $\mathbb{R}^n \times \mathbb{U}_m$. Indeed, we now assume that, to each cell D_i corresponds a "column" of cells Δ_j whose projection over $\mathbb{R}^n$ is exactly D_i (see for example, figure 2a). The latter property will allow us in section 0.4 to have the same output con-



(a) (b)

Figure 2: Examples of meshes, for $n = m = 1$, (a) satisfying the properties 1 and 2 (b) satisfying the properties 1 and not 2.

straints from a state cell D_q for each Δ_i such that: $p^{\perp}_{\mathbb{R}^n}(\Delta_i) = D_q$.

We are now able to define an algorithm computing an implicit mesh of the space $\mathbb{R}^n \times \mathbb{U}_m$. To satisfy property 2, we start by computing an implicit mesh $D = (D_q)_{q\in\mathcal{Q}}$ of the state space. Next, we compute an explicit triangulation of the control domain $\mathbb{U}_m$ (small and bounded). We lastly deduce an implicit simplicial mesh $(\Delta_i)_{i\in I}$ of $\mathbb{R}^n \times \mathbb{U}_m$.

Simplicial mesh of $\mathbb{R}^n$

The state space $\mathbb{R}^n$ is implicitly cut into n-dimensional cubes. Each cube is then meshed into $n!$ simplices and the resulting partition is a mesh of $\mathbb{R}^n$ (see [22],[24, chapter 11]) for more details). Here, a n-cube C is defined by one point $a = (a_i)_{i=1\ldots n} \in \mathbb{R}^n$ and the length h of its edges: $C = [a_1, a_1 + h] \times \cdots \times [a_n, a_n + h]$. This will also be denoted by $C = a + [0,h]^n$. In the same way we define a mesh of $\mathbb{R}^n$ into n-cubes:

Definition 1 ((Mesh of $\mathbb{R}^n$ into n-cubes)). *Let $a = (a_i)_{i=1\ldots n}$ be a given point in $\mathbb{R}^n$ and $h > 0$. Then $(a + kh + [0,h]^n)_{k=(k_1,\ldots,k_n)\in\mathbb{Z}^n}$ is a mesh of $\mathbb{R}^n$ into n-cubes.*

Now, let us consider a n-cube $C = [a_1, a_1 + h] \times \cdots \times [a_n, a_n + h]$. We then introduce $\mathcal{S}_n$ the set of permutations of $\{1, \ldots, n\}$. For all $\varphi \in \mathcal{S}_n$, $D_\varphi = \{(x_1, \ldots, x_n) \in \mathbb{R}^n; 0 \leq x_{\varphi(1)} - a_{\varphi(1)} \leq \cdots \leq x_{\varphi(n)} - a_{\varphi(n)} \leq h\}$ is a simplex in $\mathbb{R}^n$, whose vertices are defined by:

$$\begin{cases} \forall i = 1, \ldots, p, & x_{\varphi(i)} = a_{\varphi(i)} \\ \forall i = p+1, \ldots, n, & x_{\varphi(i)} = a_{\varphi(i)} + h \end{cases}, p = 0, \ldots, n \tag{6}$$

Note that property 1 is satisfied if $a = 0$.

Proposition 3 (([22])). *$(D_\varphi)_{\varphi \in S_n}$ is a mesh of the n-cube C.*

This definition enables us to give the actual size of our mesh:

Corollary 1. *$(D_\varphi)_{\varphi \in S_n}$ is a mesh of size $\sqrt{n}h$.*

Proof. Let $\varphi \in S_n$ and $(x, y) \in D_\varphi$.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = \sqrt{\sum_{i=1}^{n}(x_{\varphi(i)} - y_{\varphi(i)})^2} \leq \sqrt{\sum_{i=1}^{n} h^2} = \sqrt{n}h$$

The size of the mesh is therefore at least equal to $\sqrt{n}h$. Moreover, according to (6), we can easily check that a and $a + h$ must also remain in D_φ for any φ. Now, since $d(a, a + h) = \sqrt{n}h$, we deduce that the mesh is exactly of size $\sqrt{n}h$. □

Triangulation of $\mathbb{U}_m$

$\mathbb{U}_m$ is a convex bounded polyhedron, defined as the convex hull of its vertices. The study of simplicial subdivisions of such polytopes has been extensively developed in recent years and provides us with some efficient tools to compute them (e.g. Delaunay triangulation of the *Qhull* [3] software[3]). To build a regular mesh of size h of $\mathbb{U}_m$ we just first regularly add points in $\mathbb{U}_m$ and then apply the Delaunay triangulation.

Remark 2. *The triangulation of the control domain is computed once and for all at the beginning of the algorithm. Moreover in every control problem, we generally have no initial condition on the control. Therefore in each state cell, we need to compute all the possible related control cells, and the triangulation can not be computed on the fly as for the state. This is possible for the control polytope as it is bounded and usually much smaller than the state space. Also this same explicit triangulation will be reused in every column cell of our hybrid automaton.*

[3] tool for computation of convex hulls, Delaunay triangulation, Voronoi diagrams in 2d, 3d or higher

Simplicial mesh of $\mathbb{R}^n \times \mathbb{U}_m$

We let $(D_i)_i$ and $(U_j)_j$ respectively be the simplicial subdivisions of $\mathbb{R}^n$ and $\mathbb{U}_m$. We build a triangulation $(\Delta_q)_q$ of $\mathbb{R}^n \times \mathbb{U}_m$ **without new vertices** via the Delaunay triangulation (using e.g.*Qhull*). This last criterion guarantees the property 2 to be satisfied and gives the following size of the resulting mesh:

Lemma 3. *Let D_i and U_j be polytopes respectively in $\mathbb{R}^n$ and $\mathbb{R}^m$. Then we have:*

$$diam(D_i \times U_j) = \sqrt{diam(D_i)^2 + diam(U_j)^2}$$

Thanks to this lemma, we can conclude that the our mesh Δ is of size $\sqrt{n+1}h$.

In this section, we have described the main steps of the construction of an implicit simplicial mesh. Next, we will see that this mesh and the corresponding linear approximations, actually define a hybrid approximation of the initial system (1).

0.2.3 Hybrid Automaton

In sections 0.2.1 and 0.2.2, we built a piecewise affine approximation of the nonlinear vector field f over a simplicial mesh $\Delta = (\Delta_q)_{q \in I}$ of the space $\mathbb{R}^n \times \mathbb{U}_m$. Now, in each cell Δ_q, we want to define an affine optimal control problem.

Control constraints computation

We consider a given cell D_q in the state space. We then define $\mathcal{K}(q)$ the set of cells in Δ, whose projection in $\mathbb{R}^n$ is D_q:

$$\mathcal{K}(q) = \{q' \in I \ / \ p^{\perp}_{\mathbb{R}^n}(\Delta_{q'}) = D_q\}$$

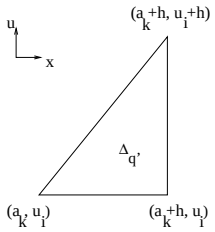
By construction, in each cell Δ_q, the control depends on the state. In this paragraph we want to compute the explicit control constraints. For that, we define $U_{q'}(X)$ to be the set of control constraints in the cell $\Delta_{q'}$ for a given $X \in \mathbb{R}^n$:

$$\forall X \in D_q, \ \forall q' \in \mathcal{K}(q), \ U_{q'}(X) = \{u \in \mathbb{R}^m \ / \ (X, u) \in \Delta_{q'}\} \tag{7}$$

The problem now is to determine the geometrical structure of $U_{q'}(X)$ in $\mathbb{R}^m$. For instance, on figure 3, we can remark that the control domain $U_{q'}(x)$ is a segment (namely an 1-simplex) in $\mathbb{R}$ when $x \in]a_k, a_k + h]$, and is reduced to one point $\{u_i\}$, when $x = a_k$.

Let us take $X \in D_q$ and $q' \in \mathcal{K}(q)$. By definition, $\Delta_{q'}$ is a simplex in $\mathbb{R}^{n+m}$, so that it can be defined by a system of $(n + m + 1)$ affine independent inequalities: $NY + d \geq 0$ where $N \in \mathbb{M}_{n+m+1,n+m}(\mathbb{R})$ and $d \in \mathbb{R}^{n+m+1}$. We define now the left and right parts of N as follows:

$$N_1 = N \begin{bmatrix} I_n \\ 0 \end{bmatrix} \in \mathbb{M}_{n+m+1,n}(\mathbb{R}), \ N_2 = N \begin{bmatrix} 0 \\ I_m \end{bmatrix} \in \mathbb{M}_{n+m+1,m}(\mathbb{R})$$



$$
\begin{array}{c}
(x,u) \in \Delta_{q'} \\
\Updownarrow \\
x \in D_q = [a_k, a_k + h] \\
\text{and} \\
u \in U_{q'}(x) = \{u \in \mathbb{U}_1; 0 \leq u - u_i \leq \\
x - a_k\}
\end{array}
$$

Figure 3: Definition of D_q and $U_{q'}(x)$ in cell $\Delta_{q'}$ for $n = m = 1$

so that: $N = [\ N_1 \mid N_2\ ]$. Thus we have $\forall (X,u) \in \Delta_{q'}, N_1 X + N_2 u + d \geq 0$. And we hence obtain a characterization of $U_{q'}$:

$$
\begin{array}{rcl}
\forall X \in D_q,\ U_{q'}(X) & = & \{u \in \mathbb{R}^m \ /\ (X,u) \in \Delta_{q'}\} \\
& = & \{u \in \mathbb{R}^m \ /\ N_2 u + (N_1 X + d) \geq 0\}
\end{array} \tag{8}
$$

Therefore $U_{q'}(X)$ is determined by a finite number of affine inequalities and is a polyhedral set. Moreover, since Δ_q is bounded by construction, it follows that $U_{q'}(X)$ is a bounded polyhedral set, i.e. a polytope. Furthermore, it also has the following property:

Proposition 4. *For all $X \in \mathring{D}_q$. Then, for all $q' \in \mathcal{K}(q)$, $U_{q'}(X)$ is a m-simplex in $\mathbb{R}^m$.*

The proof is given in appendix .2 ; the idea is to count the number of intersections between the whole control space and the n-faces of the simplex $\Delta_{q'}$.

Now, we can also define a pseudo-convexity property over the control domains. This will enable us in section 0.3.2 to compute them efficiently:

Lemma 4 ((Pseudo-convexity)). *Let q_1, q_2 be indices in $\mathcal{K}(q)$. Then:*

$$
\begin{array}{c}
u_1 \in U_{q_1}(X_1) \text{ and } u_2 \in U_{q_2}(X_2) \\
\Downarrow \\
\forall \alpha \in [0,1],\ \exists q' \in \mathcal{K}(q),\ \alpha u_1 + (1-\alpha) u_2 \in U_{q'}(\alpha X_1 + (1-\alpha) X_2)
\end{array}
$$

Proof. Let us state: $\alpha \in [0,1]$. Thanks to the definition 7, we have:

$$u_1 \in U_{q_1}(X_1),\ u_2 \in U_{q_2}(X_2) \Rightarrow (X_1, u_1) \in \Delta_{q_1},\ (X_2, u_2) \in \Delta_{q_2}$$

By assumptions: $q_1 \in \mathcal{K}(q)$, $q_2 \in \mathcal{K}(q)$, so that: $X_1 \in D_q$ and $X_2 \in D_q$. By convexity of D_q, we deduce: $\alpha X_1 + (1-\alpha) X_2 \in D_q$. Then, by convexity of $\mathbb{U}_m$, we have: $\alpha u_1 + (1-\alpha) u_2 \in \mathbb{U}_m$. Hence:

$$(\alpha X_1 + (1-\alpha) X_2, \alpha u_1 + (1-\alpha) u_2) \in D_q \times \mathbb{U}_m$$

Moreover, by construction of the mesh Δ, we have: $D_q \times \mathbb{U}_m = \bigcup_{q' \in \mathcal{K}(q)} \Delta_{q'}$. Thus, the latter pair must belong to one of the simplices:

$$\exists q' \in \mathcal{K}(q),\ (\alpha X_1 + (1-\alpha) X_2, \alpha u_1 + (1-\alpha) u_2) \in \Delta_{q'}$$

This proves that $\alpha u_1 + (1-\alpha)u_2 \in U_{q'}(\alpha X_1 + (1-\alpha)X_2)$ and the lemma is true. □

Once the control constraints are explicitly determined, we are now able to define a hybrid optimal control problem.

Hybrid automaton definition

Let us introduce the equivalence relation $\sim$ on I defined by:

$$i \sim j \Leftrightarrow p^{\perp}_{\mathbb{R}^n}(\Delta_i) = p^{\perp}_{\mathbb{R}^n}(\Delta_j)$$

We then introduce the index q of the cell D_q in the state space, which is the projection of Δ_i and Δ_j. We can then define the equivalence classes of the relation $\sim$:

Lemma 5. *The equivalence classes for the relation $\sim$ are the sets $\mathcal{K}(q)$, where q is an index of the mesh D of $\mathbb{R}^n$. Moreover: $\forall q$, card $\mathcal{K}(q) < +\infty$.*

In control theory, the control functions are generally measurable and not continuous (at best piecewise continuous). Therefore, every trajectory $(X(.), u(.))$ in the space $\mathbb{R}^n \times \mathbb{U}_m$ is a priori discontinuous, whereas $X(.)$ is a continuous trajectory in the state space. Moreover, figure 4 shows that the notion of tran-

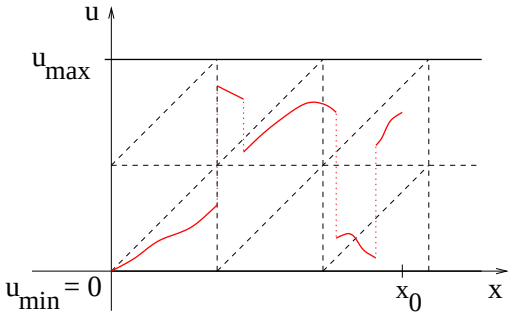


Figure 4: Example of discontinuous trajectory in the space $\mathbb{R} \times [0, u_{max}]$

sitions between two cells of the mesh Δ has no meanings with respect to the control. Indeed, an admissible control function can switch at any time to a given cell Δ_i to another Δ_j, not necessarily adjacent to Δ_i, and without reaching one boundary of the system. This observation leads us to introduce a new class of hybrid systems with inputs:

Definition 2 ((Controlled hybrid automaton)).

1. *$\mathcal{Q}$ the countable set of class representatives for $\sim$ (i.e. the set of indices of the simplices D_q).*

2. $\mathcal{D} = \{D_q \ / \ q \in \mathcal{Q}\}$ *the collection of domains induced by* Δ *over the state space:*

$$\begin{cases} \forall q \in \mathcal{Q},\ \exists q' \in I,\ D_q = p^{\perp}_{\mathbb{R}^n}(\Delta_{q'}) \\ \\ \forall (q,q') \in \mathcal{Q}^2,\ [\overset{\circ}{D_q} \cap \overset{\circ}{D_{q'}} \neq \emptyset \Rightarrow D_q = D_{q'}] \end{cases}$$

3. $\mathcal{U} = \{\mathcal{U}_q \ / \ q \in \mathcal{Q}\} \subset \mathbb{U}_m$ *the collection of control domains induced by* Δ*:*

$$\begin{cases} \mathcal{U}_q = \{u(.) \text{ admissible } / \ \forall X \in D_q,\ \exists q' \in \mathcal{K}(q),\ u(X) \in U_{q'}(X)\} \\ \\ U_{q'}(X) = \{u \in \mathbb{R}^m \ / \ \begin{bmatrix} X \\ u \end{bmatrix} \in \Delta_{q'}\} \end{cases}$$

4. $\mathcal{E} = \{(q,q') \in \mathcal{Q} \times \mathcal{Q} / \ \partial D_q \cap \partial D_{q'} \neq \emptyset\}$ *the transition set.*

5. $\mathcal{F} = \{\mathcal{F}_q \ / \ q \in \mathcal{Q}\}$ *the collection of field vectors of section 0.2.1, where:* $\mathcal{F}_q = \{f_{q'} \ / \ q' \in \mathcal{K}(q)\}$ *and:*

$$\begin{array}{rccl} f_{q'}: & \Delta_{q'} & \rightarrow & \mathbb{R}^n \\ & (X,u) & \rightarrow & A_{q'}X + B_{q'}u + c_{q'} \end{array}$$

6. $\mathcal{G} = \{G_e \ / \ e \in \mathcal{E}\}$ *the collection of the guards:*

$$\forall e = (q,q') \in \mathcal{E}, G_e = \partial D_q \cap \partial D_{q'}$$

7. $\mathcal{R} = \{R_e / e \in \mathcal{E}\}$ *the collection of Reset functions:*

$$\forall e = (q,q') \in \mathcal{E}, \forall x \in G_e, R_e(x) = \{x\}$$

(Here, we do not need to reinitialize the continuous variable x*, since the* D_q *are adjacent).*

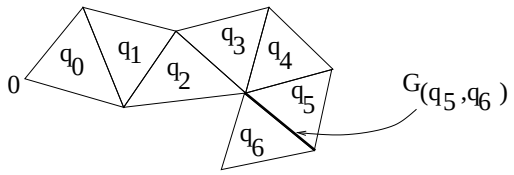


Figure 5: Illustration of the hybrid automaton definition over 7 modes $(q_0, q_1, q_2, q_3, q_4, q_5, q_6)$

Notation 1. *When there is no ambiguity about the trajectory, we will denote* $u(X(t))$ *simply by* $u(t)$*.*

From now on, we will make the following assumption:

Hypothesis 1. *The hybrid automaton $\mathcal{H}$ is assumed not Zeno*[4]

Once the hybrid automaton $\mathcal{H}$ is defined, we can then build a new optimal control problem $(\mathcal{P}_{\mathcal{H}})$ as the hybrid approximation of the initial one (1)-(2): *"Let X_0 be a point of $\mathbb{R}^n$. We want to find a trajectory under the $\mathcal{H}$'s dynamic that steers the initial point X_0 to the target point 0, locally minimizing the cost function $J(X_0, u(.))$."*

Next we propose some methods and algorithms to solve the hybrid problem $(\mathcal{P}_{\mathcal{H}})$. Our analysis is divided in two parts. First in §0.3, we address the controllability problem, namely the existence of (non compulsorily optimal) solutions. Then, in §0.4, we provide an algorithmic resolution of the optimization problem $(\mathcal{P}_{\mathcal{H}})$.

0.3 Approximation of the Controllable Domain

In this section, we focus on the controllability of the nonlinear system (1) towards the origin. For a given initial point $X_0 \in \mathbb{R}^n$ in the state space, we want to study the existence of admissible trajectories of the system that steers X_0 to the considered target 0 at some finite time. The theory of linear control systems without constraints on the control have been extensively developed and provides some powerful results like the Kalman controllability criterion ([26, 27]). Using the implicit functions theorem, the analysis of the local linearization of nonlinear systems gives some results about the local controllability of the system ([29]). In this paper, we present methods and algorithms to approximate the controllable domain of nonlinear control systems by the way of the hybrid model. In §0.3.1, we present a hybrid approach to the controllability of nonlinear systems: the nonlinear controllable domain is approximated by the hybrid one over a finite cells path in the state space. Then, in the following paragraph, we give some useful properties of the hybrid controllable domain, which will enable us in §0.3.3 to define a new algorithm computing an under-approximation of the hybrid controllable domain.

0.3.1 Hybrid approach

In this paragraph, we focus on the notion of controllability to the origin of nonlinear systems:

Definition 3. *$X_0 \in \mathbb{R}^n$ is controllable to 0 if and only if there exists $T \geq 0$ and $u : [0, T] \to \mathbb{U}_m$ measurable, such that the system*

$$\begin{cases} \dot{X}(t) &= f(X(t), u(t)) \\ X(0) &= X_0, \quad X(T) = 0 \end{cases}$$

[4]Zeno executions correspond to an infinite number of switch in a finite time. That often involves problems in the simulation of hybrid system. Indeed the transition times come closer and closer and in simulations, we can not differentiate them any more (see [24, §2.3] and [35]).

admits a solution $X(.)$ over $[0,T]$. The set of controllable points in $\mathbb{R}^n$ is called controllable domain *of the system (1).*

This definition can be applied to any control system, and particularly to the hybrid automaton previously described (see §0.2.3). So, any $X_0 \in \mathbb{R}^n$ is controllable for the hybrid system $\mathcal{H}$ if and only if there exists an admissible trajectory of the system that steers X_0 to the target 0 in some time $T > 0$. Let us now introduce the notion of solution of a hybrid system (inspired from [24, section 2.1]):

Definition 4. *$(X(.), u(.))$ is a solution of the hybrid control problem ($\mathcal{P}_\mathcal{H}$) (i.e. X_0 controllable) if there exists a finite execution $\chi = (\tau, q, X)$ satisfying:*

i. $(q(t_0), X(t_0)) = (q_0, X_0)$ *and* $(q(t_{r+1}), X(t_{r+1})) = (q_r, 0)$.

ii. For all $i \in \{0, \ldots, r\}$, such that: $t_i < t_{i+1}$:
- $\forall t \in]t_i, t_{i+1}[,\ q(t) = q_i$ *and* $X(t) \in D_{q_i}$
- *there exists a measurable control function $u(.) \in \mathcal{U}_{q_i}$ over $[t_i, t_{i+1}]$, such that:*
$$\forall t \in]t_i, t_{i+1}[,\ \dot{X}(t) = f_h(X(t), u(t))$$

iii. $\forall i \in \{1, \ldots, r\},\ X(t_i) \in G_{(q_{i-1}, q_i)}$.

where: $\tau = ([t_i, t_{i+1}])_{i=0\ldots r-1}$ $(t_i \leq t_{i+1})$ and $q = (q_i)_{0 \leq i \leq r}$ is a finite sequence of modes of the automaton $\mathcal{H}$.

Moreover, for any controllable point $X_0 \in \mathbb{R}^n$, the hybrid system may accept several executions that steer X_0 to the target, and several different cell paths $q = (q_i)_i$ between X_0 and 0. So the analysis of the global controllability of the hybrid automaton $\mathcal{H}$ implies to first determine one admissible sequence of modes, on which we then compute the controllable domain for the system $\mathcal{H}$. From now on, we so consider the following assumption illustrated on figure 6:

Hypothesis 2. *Let $q = (q_i)_{i=0\ldots r}$ be a sequence of discrete modes of the automaton $\mathcal{H}$ such that:*

$$\begin{cases} X_0 \in D_{q_0},\ 0 \in D_{q_r} \\ \forall i = 1 \ldots r-1,\ D_{q_i} \cap D_{q_{i+1}} \neq \emptyset \end{cases}$$

$(D_{q_i})_{i=0\ldots r}$ is the sequence of adjacent state cells related to the path q.

For any given sequence of discrete modes $q = (q_i)_{i=0\ldots r}$ satisfying the hypothesis 2, we define:

- a successor function *succ*: $succ_q(q_i) = q_{i+1}$ for $i < r$

- the sequence of modes $\bar{q}$ associated to $q = (q_i)_{i=0\ldots r}$ by time reversal:

$$\bar{q} = (q_{r-i})_{i=0\ldots r} \text{ and } succ_{\bar{q}}(q_i) = q_{i-1} \text{ for } i = 1 \ldots r$$

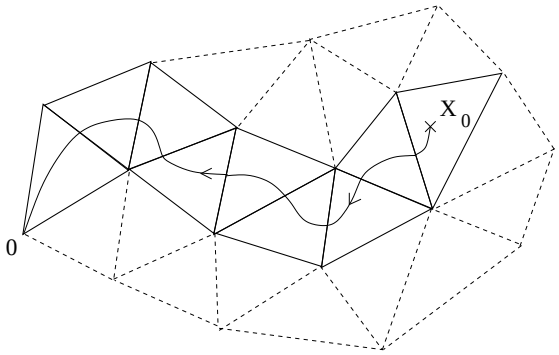


Figure 6: Hybrid controllability

The hypothesis 2 leads us to introduce the domain: $\Omega = \bigcup_{i=0}^{r} D_{q_i} \subset \mathbb{R}^n$ and to focus on the controllability of the nonlinear system (1) inside Ω in the state space. Our hybrid approach consists then in approximating the nonlinear controllable domain on Ω by the hybrid one.

This idea induces a recursive definition of the controllable domain inside the domain Ω, for the nonlinear control system as well as for the hybrid one. It is build inside Ω by time reversal, i.e. from the target 0 to successive cells $D_{\bar{q}_0}, \ldots, D_{\bar{q}_r}$, according to the path $\bar{q}$.

Definition 5 ((Controllable domain $C(\Omega)$)). *Let $q = (q_i)_{i=0\ldots r}$ be a sequence of modes in the automaton $\mathcal{H}$ satisfying the hypothesis 2 and $\Omega = \bigcup_{i=0}^{r} D_{q_i}$. The controllable domain $C(\Omega)$ inside Ω for the considered control system is defined by:*

- *$C(D_{\bar{q}_0})$ is the set of points $X_0 \in D_{\bar{q}_0}$, steerable to the target 0, while remaining within the cell $D_{\bar{q}_0}$.*
- *For all $i \in \{0, \ldots, r-1\}$, $C(\bigcup_{j=0}^{i+1} D_{\bar{q}_j})$ is the union of $C(\bigcup_{j=0}^{i} D_{\bar{q}_j})$ and of the set of points $X_0 \in D_{\bar{q}_{i+1}}$, that are steerable to the current target $C(\bigcup_{j=0}^{i} D_{\bar{q}_j}) \cap G_{(\bar{q}_i, \bar{q}_{i+1})}$, while remaining within the cell $D_{\bar{q}_{i+1}}$.*

0.3.2 Controllable domain of the hybrid automaton

Let q be a discrete mode of the automaton $\mathcal{H}$ satisfying: $0 \in D_q$ ($q = \bar{q}_0$ of definition 5). In this section, we are interested in determining some topological properties of the controllable domain inside the cell D_q under the control constraints $\mathcal{U}_q$. These properties will enable us to compute a good approximation

of the controllable set in section §0.3.3. We now consider the piecewise affine dynamic f_h defined by the hybrid automaton (see §0.2.3). The controllability of the hybrid system $\mathcal{H}$ in the state cell D_q is defined as follows:

Definition 6 ((Controllability in mode q)). *$X_0 \in D_q$ is controllable by the hybrid automaton $\mathcal{H}$ if and only if there exists $T \geq 0$ and $u : [0,T] \to \mathbb{U}_m$ measurable, such that the system:*

$$\left\{ \begin{array}{lcl} \dot{X}(t) & = & f_h(X(t), u(t)) \\ X(0) & = & X_0, \quad X(T) = 0 \end{array} \right.$$

admits a solution $X(.)$ over $[0,T]$ such that:

$$\forall t \in [0,T], \ \left\{ \begin{array}{l} X(t) \in D_q \\ \exists q'(t) \in \mathcal{K}(q), \ u(t) \in U_{q'(t)}(X(t)) \end{array} \right.$$

Proposition 5. *The controllable domain in mode q is convex.*

The proof of proposition 5 has to be completed. This result has been analytically proved for linear systems and piecewise affine systems with constant constraints on the control and could be generalized to our hybrid automaton definition.

Corollary 2. *Let q be a mode of the hybrid automaton $\mathcal{H}$ and $q' \in \mathcal{K}(q)$.*
If $0 \in \Delta_{q'}$ (i.e.: $0 \in D_q$ and $\forall X \in D_q, \ 0 \in U_{q'}(X)$), then the controllable domain under the state and control constraints induced by $\Delta_{q'}$ is convex.

0.3.3 Computation of a controllable under-approximation

In this section, we want to compute the set of controllable points in $\mathbb{R}^n$, i.e. the set of initial points for which the problem $(\mathcal{P}_\mathcal{H})$ admits a solution. In [20], we proposed an algorithm to compute an under-approximation in time $T > 0$ of the controllable set without state constraints. In [33], an extension of this algorithm to piecewise affine systems under constant control constraints is presented. Here these algorithms are developed and widely improved for hybrid systems like $\mathcal{H}$. The principle is as follows: by time reversal, the computation of the controllable set comes down to the computation of the attainable set from the target. In [1], for safety verification, the idea is to compute an over-approximation of the attainable set. They can thus certify that the system can not escape from an admissible set of states. On the contrary, we address the problem of guarantying the controllability of a given initial point. Therefore, we instead compute an under-approximation of the controllable set.

Under-Approximation within a given state cell

In this paragraph, we address the problem of under-approximating the controllable set in the state cell $D_q, \ q \in \mathcal{Q}$. The idea is to compute for each cell $\Delta_{q'}$,

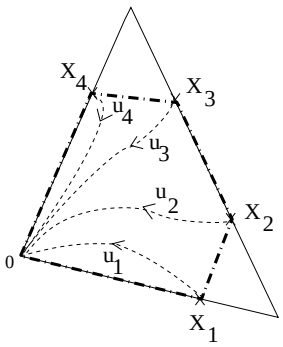


Figure 7: Under-approximation principle in a state cell D_q

$q' \in \mathcal{K}(q)$, some specific controllable points in D_q ; then, thanks to the proposition 5, their convex hull is an under-approximation of the controllable set (see figure 7). The algorithm presented below is inspired by [20, 33]. There, the under approximation is computed for piecewise affine automata with constant control constraints with several discretisation levels. Here, the automaton cells are build for a selected precision h and we choose the same precision for the under-approximation. This is actually simpler and the under-approximation quality is nonetheless preserved !

Notation 2. *From now on, $X[X_0, u]$ denotes the trajectory that goes through X_0 according to the control function $u(.)$.*

Let $\Delta_{q'}$, $q' \in \mathcal{K}(q)$, be a given cell in the state-control space $\mathbb{R}^n \times \mathbb{U}_m$. Now, we want to compute some controllable points in D_q under the control constraints defined by $\mathcal{U}_q$: the first step is to compute the vertices $u_i(X)$ of the control domain $U_{q'}(X)$, for any $X \in D_q$.

Remark 3. *As previously seen in proposition 7, the control constraints in the cell $U_{q'}(X)$ are affine in the state and the control ; so $\forall i$, $u_i(X)$ depends affinely on the state X.*

After that, for each control function u_i, we consider the trajectory $X[0, u_i]$ and compute its intersection, when it exists, by time reversal with the boundary of D_q:

$$X_i = X[0, u_i](T_i) \quad \text{where:} \quad T_i = sup\{t < 0; X[0, u_i](t) \in \partial D_q\}$$

Remark 4. *If the system: $\dot{X}(t) = f_h(X(t), u_i(X(t)))$, $X(0) = 0$ has an equilibrium point in D_q, then: $T_i = -\infty$ and $X_i = \lim_{t \to -\infty} X[0, u_i](t)$ exists and is controllable in mode q.*

Let $\Lambda_q(q')$ be the set of the so computed controllable points in D_q. According to the proposition 5, we deduce:

Proposition 6. $\Lambda_q = Conv(\bigcup_{q' \in \mathcal{K}(q)} \Lambda_q(q'))$ *is an under-approximation of the controllable set in mode* q.

The following algorithm 3 thus computes the under-approximation of the controllable domain in mode q.

Algorithm 3 UnderApproximationCell

Require: $\mathcal{H}$, D_q a given state cell.
Ensure: an under-approximation of the controllable domain in D_q.

1: Initial under-approximation: $\Lambda := \emptyset$;
2: Target: $\mathcal{T} := \{0\}$;
3: $\mathcal{K}(q) := \{q' \in I \ / \ p^{\perp}_{\mathbb{R}^n}(\Delta_{q'}) = D_q\}$ {*Computation of the set of indices of cells* $\Delta_{q'}$ *whose projection on* $\mathbb{R}^n$ *is* D_q.}
4: In each cell $\Delta_{q'}$
5: ***for all*** $q' \in \mathcal{K}(q)$ ***do***
6: Computation of the set of control constraints defined by the list of its vertices
$U_q := ControlConstraints(\Delta_{q'})$;
{For each vertex of $U_{q'}(X)$}
7: **for all** time step i (from 1 to card($U_{q'}(X)$)) **do**
8: i-th vertex of $U_{q'}(X)$: $u := X \rightarrow U_{q'}(X)[i]$;
9: Computation of the trajectory from $\mathcal{T}$ according to $u = U_{q'}(X)[i]$
$X[\mathcal{T}, u](.)$ solution of $\dot{X}(t) = A_{q'}X(t) + B_{q'}u(X(t)) + c_{q'}$, $X(0) = \mathcal{T}$.
10: *Computation of the intersection time between* $X[\mathcal{T}, u](.)$ *and the boundary of* D_q
$T := sup\{t < 0; X[\mathcal{T}, u](t) \in \partial D_q\}$
{*The so-computed point is in the under-approximation*}
11: $\Lambda := \Lambda \cup \{X[\mathcal{T}, u](T)\}$
12: **end for**
13: ***end for***
14: *Return ConvexHull*(Λ).

Under-Approximation over a given path

Let $q = (q_i)_{i=0...r}$ be a given sequence of discrete modes of the hybrid automaton $\mathcal{H}$ satisfying the hypothesis 2, i.e. such that:

$$\begin{cases} 0 \in D_{q_0} \\ \forall i \in [|0, r-1|], \ G_{(q_i, q_{i+1})} \neq \emptyset \end{cases}$$

As similarly done in [33], we are now able to compute an under-approximation of the controllable set over the sequence of adjacent states D_{q_i}. The principle is to start by computing the under-approximation of the controllable set from the target 0 in the first cell q_0 by the algorithm 3. Then, from its intersection with the guard $G_{(q_0,q_1)}$, we pursue the under-approximation, the same way. The only

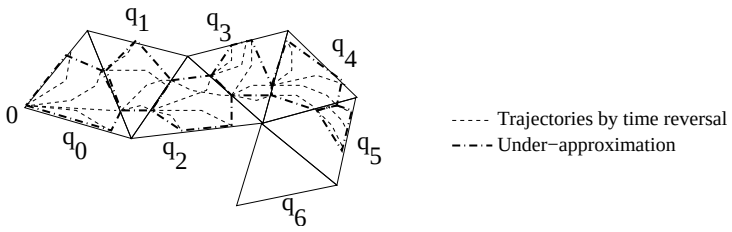


Figure 8: Construction of an under-approximation of the controllable set in a given path $(q_0, q_1, q_2, q_3, q_4, q_5, q_6)$ of discrete modes.

difference is that the reverse starting point is not 0 any more. It is instead the extremal points of the intersection between the guard and the current under-approximation. The algorithm ends when this intersection is empty or when the last state q_r is reached. This algorithm is illustrated on figure 8. The computed under-approximation describes the set of points, which are reachable by time reversal inside a given sequence of cells. It clearly depends on the order of cells inside the cells path (see example 0.3.4).

Controllability of the initial point

Up to now, we have seen how to build an under-approximation of the hybrid controllable domain on a given finite sequence of modes. Now, we come back to the controllability of a given initial point in the state space for the hybrid system. Let $X_0 \in \mathbb{R}^n$ be a given initial point for the system $\mathcal{H}$. According to the hypothesis 2, we consider a finite sequence of discrete modes $q = (q_i)_{i=1\ldots r}$ between X_0 and the target 0 and the domain: $\Omega = \bigcup_{i=1}^{r} D_{q_i}$, on which we study the controllability to 0. The idea is to compute an under-approximation $\Lambda(\Omega)$ of the controllable domain on the domain Ω, then, if $X_0 \in \Lambda(\Omega)$, then X_0 is controllable. Otherwise we can combine an under- and an over-approximation of the controllable set: if X_0 is out of the over-approximation, we can certify that it is not controllable. If X_0 is in between, we can not conclude.

0.3.4 Experimental results

In this section, we consider the example of the nonlinear spring presented in [34, paragraph 7.3.1]:

$$\left\{ \begin{array}{lcl} \dot{x}(t) & = & y(t) \\ \dot{y}(t) & = & -x(t) - 2 * x(t)^3 + u(t) \end{array} \right. \tag{9}$$

where the control constraints are: $\forall t \geq 0,\ |u(t)| \leq 1$. In this section, we want to apply the previously described algorithms to compute an under-approximation of the set of initial positions $(x_0, y_0 = \dot{x}_0)$, from which the spring can be steered

to its equilibrium position $(0, 0)$.

In this example, we perform the under-approximation algorithm on three successive adjacent cells as shown on figure 9-(a): Now, we are given the

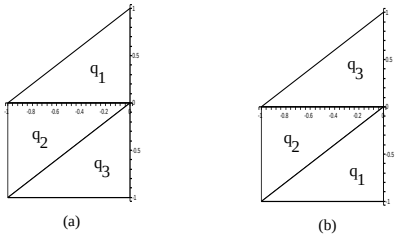


Figure 9: Two different paths inside one given domain in the state space.

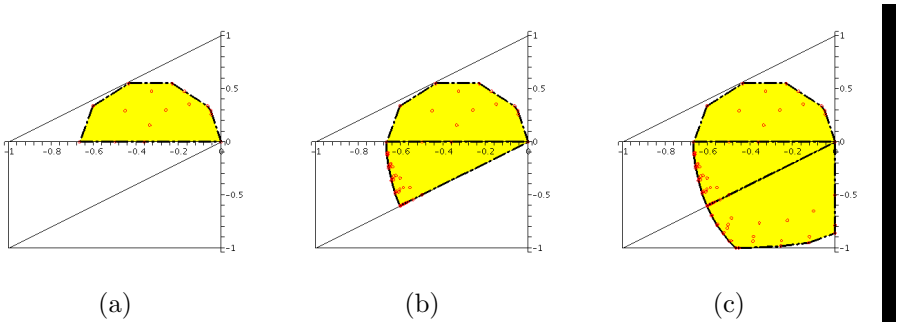


Figure 10: Controllable under-approximations for the nonlinear spring inside the three successive cells in $\mathbb{R}^2$ of figure 9-(a).

same set of adjacent state cells, but in reverse order, as shown on figure 9-(b). We want to compare the controllable under-approximations over these two cells paths. As expected, the resulting under-approximations are different. In fact, the resulting under-approximation is the one computed in the first step (a) on figure 10;
Indeed, each boundary between two adjacent modes can not necessarily be crossed in every direction. In conclusion, when several different paths exist inside one given domain in the state space, we have to compute the under-approximations on all the possible paths.

Example 1 ((Runtimes)).
The under-approximation algorithm 3, implemented in Maple[5]*, is performed*

[5]The maplets are available online at: `www-lmc.imag.fr/lmc-mosaic/Jean-Guillaume.Dumas/SHOC`

inside a given simplex S_n in $\mathbb{R}^n$ for $u \in [-1,1]$ where S_n is the standard simplex of $\mathbb{R}^m$, we have scaled in order to respect the size h of the mesh of the control and state space $\mathbb{R}^n \times [-1,1]$:

$$S_n = Conv(\{(h\delta_{ij})_{j=1\ldots m+1} \ / \ i = 1, \ldots, m+1\})$$

where: δ_{ij} is the Kronecker symbol: $\delta_{ij} = 1$ if $i = j$. Otherwise, when $i \neq j$, $\delta_{ij} = 0$. The considered nonlinear dynamic is: $f(X,u) = (X_1 u, \ldots, X_n u)$.

Table 1: Global timings with Maple 9.5 on a PIV 3.4 GHz: cpu (s) for $h = 1$

n*	2	3	4	5	6	7	8
	3.346 (12)†	4.643 (16)	6.795 (20)	10.611 (24)	16.110 (28)	22.213 (32)	29.978 (36)
	9	10	11	12	13	14	20
	41.691 (40)	54.501 (44)	80.067 (48)	96.205 (52)	120.913 (56)	157.066 (60)	626.884(84)

* n is the dimension of the state space † (.) is the number of cells $\Delta_{q'}$ in one mode

The times given in table 1 take into account:

- *the computation of the set of control domains $U_{q'}(X)$ related to the simplex S_n (their number increases with the dimension).*
- *the hybridisation, i.e. to compute the affine approximation in each resulting state and control cell in $\mathbb{R}^{n+m}$.*
- *the under-approximation computation in each cell.*

0.4 Computation of optimized solutions

This section deals with the algorithmic resolution of the nonlinear optimal control problem (1)-(2) stated in introduction. For a given controllable point $X_0 \in \mathbb{R}^n$, we want to select among all the admissible trajectories that steers X_0 to 0, those that minimize the given cost. We therefore now present some methods and algorithms to approximate the optimal solutions of the nonlinear control problem. First, in paragraph 0.4.1, we describe the hybrid approximation of optimal solutions of the nonlinear problem. Then an algorithm is proposed in §0.4.2 to solve the hybrid optimal control problem. We end the section with a full example showing the overall process.

0.4.1 Hybrid approximation of optimal solutions

In this paragraph we focus on the resolution of the nonlinear optimal control problem stated in introduction:

$(\mathcal{P}_{NL})$ Minimize the cost function $J(X_0, u(.)) = \int_0^{t_f} l(X(t), u(t))dt$ with respect to the control $u(.)$ under the nonlinear dynamic (1):

$$\begin{cases} \dot{X}(t) = f(X(t), u(t)) \\ X(0) = X_0 \end{cases} \qquad X(t_f) = 0$$

and the constraints: $\forall t \in [0, t_f], u(t) \in \mathbb{U}_m$ convex and compact polytope in $\mathbb{R}^m$, such that: $0 \in \mathbb{U}_m$. The final time t_f is unspecified.

We define a hybrid optimal control problem $(\mathcal{P}_{\mathcal{H}})$ by replacing the vector field f by f_h. In section 0.3 we approximated the nonlinear controllable domain by the hybrid one inside a given compact subset Ω of $\mathbb{R}^n$. Now, we consider the *optimal* resolution of the problem $(\mathcal{P}_{NL})$ inside Ω. Our approach is divided in two main steps: we first define the *hybrid optimal* control problem, and then prove that hybrid optimal solutions are good approximations of the nonlinear optimal ones. Let X_0 be a given initial point in the state space $\mathbb{R}^n$ and $q = (q_i)_{i=1...r}$ the finite sequence of discrete modes (satisfying the hypothesis 2), on which X_0 is controllable for the nonlinear system (1): $X_0 \in C_{NL}(\Omega)$ where $\Omega = \bigcup_{i=1}^{r} D_{q_i}$ and $C_{NL}(\Omega)$ is nonlinear controllable domain inside Ω. Any optimal solution inside Ω is then defined step by step from X_0 in mode q_0 towards 0 in mode q_r: for i from 0 to r, we have to solve a local optimal control problem $(\mathcal{P}_{NL}(q_i))$ in each mode q_i of the given path:
$(\mathcal{P}_{NL}(q_i))$ Minimize the cost function $J(X_i, u(.)) = \int_{t_i}^{t_{i+1}} l(X(t), u(t))dt$ with respect to the control $u(.) \in \mathcal{U}_{q_i}$ under the nonlinear dynamic (1):

$$\begin{cases} \dot{X}(t) = f(X(t), u(t)) \\ X(t_i) = X_i \end{cases} \qquad X_{i+1} = X(t_{i+1}) \in G_{(q_i, q_{i+1})}$$

and the constraints: $\forall t \in [t_i, t_{i+1}], (X(t), u(t)) \in \Delta_{q_i}$, where the final time t_{i+1} is unspecified.
We so compute an optimal execution (τ, q, X) (see definition 4) in Ω of the nonlinear optimal control problem $(\mathcal{P}_{NL})$, by stating:

$$\tau = ([t_i, t_{i+1}])_{i=0...r},\ X(t_0) = X_0,\ X(t_{r+1}) = 0,\ G_{(q_r, q_{r+1})} = \{0\}$$

Our purpose is to approximate optimal solutions of $(\mathcal{P}_{NL})$ inside Ω by the optimal solutions of $(\mathcal{P}_{\mathcal{H}})$ inside Ω. Therefore, the following paragraph is devoted to the algorithmic resolution of the hybrid optimal control problem inside Ω.

0.4.2 Computation of hybrid optimal solutions

Let us consider the hybrid optimal control problem $(\mathcal{P}_{\mathcal{H}})$ inside Ω as defined in previous section. Its definition emphasizes the existence of local optimal control problems to solve. There are actually three levels of resolution in our approach. The first one deals with the affine optimal control problems defined in each cell

$\Delta_{q'}$ of the state and control space. Then the second is to solve the optimal control problem in the current mode of the considered path q. The last one is the generic optimal resolution inside the domain Ω, i.e. all over the path q.

Local optimal solutions in each cell $\Delta_{q'}$

Let us consider a mode q_i in our given path q and $q' \in \mathcal{K}(q_i)$ (see definition 2). In this section, we want to compute optimal solutions for the hybrid problem $(\mathcal{P}_{\mathcal{H}})$ in the state and control cell $\Delta_{q'}$. By definition of the automaton $\mathcal{H}$, in $\Delta_{q'}$ we can define a constrained affine optimal control problem $(p_{q_i}(q'))$:
$(p_{q_i}(q'))$ *Minimize the cost function* $J(X_0, u(.)) = \int_0^{t_f} l(X(t), u(t))dt$ *with respect to the control* $u(.) \in \mathcal{U}_q$ *under the dynamic:*

$$\begin{cases} \dot{X}(t) = A_{q'}X(t) + B_{q'}u(t) + c_{q'} \\ X(0) = X_i \end{cases} \qquad X(t_f) \in \mathcal{T}$$

and the constraints: $\forall t \in [0, t_f], (X(t), u(t)) \in \Delta_{q'}$ *for an unspecified final time* t_f.
The target $\mathcal{T}$ *is the guard* $G_{(q_i, q_{i+1})}$ *of* D_{q_i}, *if* $i < r$, $\{0\}$ *otherwise.*

We then are typically in the context of mixed state and control constraints affine optimal control problems. Optimal control under state inequality constraints is a hard and subtle problem. Indeed, some problems with bounded target curves have no generic solving methods, see [31]. A crucial point that would make our problem generically solvable is the shape of our final constraints. From now, we consider that **the target $\mathcal{T}$ is to reach the sub-space generated by the target guard of D_{q_i}.** So, let us now state our optimal control problem $(p_{q_i}(q'))$ in terms of optimization via the Pontryagin maximum principle (see e.g. [8, 10]) where the state constraints induced by D_q are given by the affine inequalities $N_{q_i}X + L_{q_i} \leq 0$. We first introduce the Hamiltonian function

$$H_{q_i}(X, u, \lambda) = l(X, u) + (\lambda^T + \mu^T N_{q_i})(A_{q'}X + B_{q'}u + c_{q'})$$

where μ is a Lagrange multiplier verifying:

$$\forall j,\ \mu_j \begin{cases} = 0 & \text{if } (N_{q_i}X + L_{q_i})_j < 0 \\ > 0 & \text{if } (N_{q_i}X + L_{q_i})_j = 0 \end{cases}$$

We can now formulate the optimization problem thanks to the Pontryagin principle:

Minimize the Hamiltonian function H *with respect to the control variable* $u \in U_{q'}(X)$ *under the constraints:*

$\dot{X}(t) = \frac{\partial H}{\partial \lambda}(X(t), u(t), \lambda(t), \mu(t))$

$\dot{\lambda}^T(t) = \frac{\partial H}{\partial X(t)}(X(t), u(t), \lambda(t), \mu(t))$

$H(X^\star(t), u^\star(t), \lambda^\star(t), \mu^\star(t)) = 0$ *along the optimal trajectory*

$\forall t \geq 0,\ X(t) \in D_{q_i}$, *i.e.:* $N_{q_i} X + L_{q_i} \leq 0$

Transverse condition: $< \lambda(t_f), n_{\mathcal{T}} >= 0$ *where* $n_{\mathcal{T}}$ *is the normal vector to the target* $\mathcal{T}$*".*

The principle of our local solving algorithm is first to solve the affine optimal control problem without state constraints. Useful methods and algorithms have been developed in [20]. Once we have computed the optimal control $\bar{u}^\star$ and the related optimal trajectory $\bar{X}^\star$ towards the target hyper-plan $\mathcal{T}$, two configurations may occur (see figure 11):

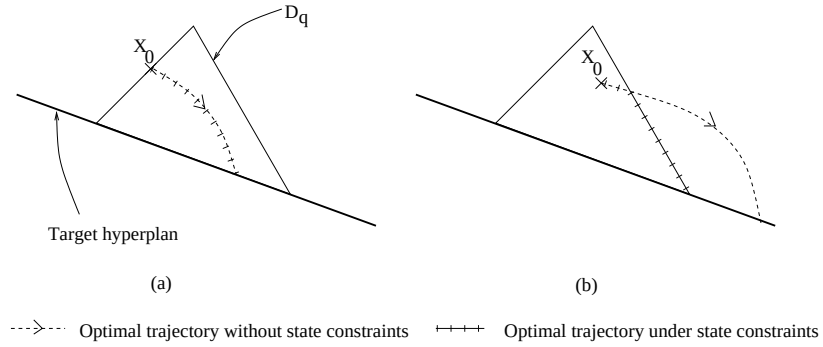


Figure 11: Local optimal solutions from a controllable point X_0 in mode q

1. $\forall t \geq 0,\ \bar{X}^\star(t) \in D_{q_i}$. Then the optimal trajectory without state constraints is optimal for our problem $(p_{q_i}(q'))$, see figure 11-(a).
2. $\exists t \geq 0,\ \bar{X}^\star(t) \notin D_{q_i}$. Then the optimal trajectory without state constraints is optimal for our problem $(p_{q_i}(q'))$, see figure 11-(b).

Optimal control in mode q_i

In this section, we want to solve the optimal control problem in one given mode q_i. By construction of the hybrid automaton (see definition 2), to each mode correspond a finite number of state and control cell in Δ. The main purpose in this paragraph is to manage the optimal resolution of the hybrid problem through this set of cells. Let us consider the cell D_{q_i} in the state space $\mathbb{R}^n$ and the set of cells $\Delta_{q'}$, $q' \in \mathcal{K}(q_i)$. To compute the optimal control from X_i, we have exactly *card* $\mathcal{K}(q_i)$ candidate cells $\Delta_{q'}$. The idea is to solve the local optimal

control problem $(p_{q_i}(q'))$ in each $\Delta_{q'}$ as presented in previous section 0.4.2 ; and then to compute the local optimal trajectory $(X^*_{q'}, u^*_{q'})$ and the related value function $V_{q'}(X_0)$. We end by a finite discrete optimization over these cells ; i.e. find the mode q_i^* such that:

$$V_{q_i^*}(X_0) = \min_{q' \in \mathcal{K}(q_i)} V_{q'}(X_0)$$

However the last optimization step can sometimes be avoided. Indeed, as explained in section 0.4.2, any optimal trajectory $(X(.), u(.))$ evolves along the edges of our mesh Δ of $\mathbb{R}^n \times \mathbb{U}_m$.
Let us take the example of the figure 12 and consider the trajectory between the

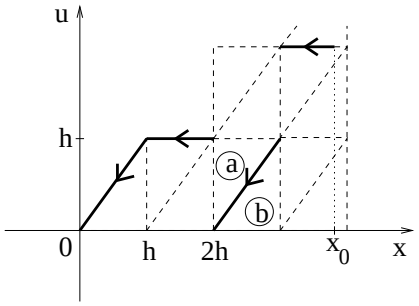


Figure 12: Example of optimal configurations

cells a and b. We assume that we have already computed the optimal control in each cell so that:

- in cell a, the optimal control is constant, equal to h.
- in cell b, the optimal control depends on the state and: $u^* = x - 3h$

As shown on figure 12, the optimal trajectory in cell b evolves along the boundary between the two cells a and b. In consequence, it could belong to both cells a and b. Moreover we have seen that the optimal trajectory in cell b is not optimal in cell a. We then conclude that the cell a is a best choice than the cell b for our optimal control problem. Therefore the discrete optimization can be

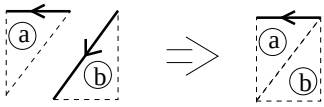


Figure 13: Example of optimal choices between two cells in dimension 2: left the local optimal choices in each cell, right the deduced optimal choice in the column

avoided by the application of that kind of analysis (see figure 13).

Hybrid solver

In regard of previous results, we can now describe the HybridSolver algorithm: as expressed in hypothesis 2, we assume a sequence of state given. The principle of our solving algorithm 4 is to compute column by column a piecewise optimal solution of our hybrid problem along the path.

Algorithm 4 HybridSolver

Require: X_0 and the hybrid automaton $\mathcal{H}$
Require: $q = (q_i)_{i=1..r}$ a sequence of discrete modes s.t. $X_0 \in D_{q_0}$ and $0 \in D_{q_r}$.
Ensure: $\tau = ([t_i, t_{i+1}])$, the sequence of times for entering and exiting a mode.
Ensure: X a trajectory such that (τ, q, X) is a local optimal execution of $\mathcal{H}$
Ensure: u the corresponding optimal control.
Ensure: $V(X_0)$ the corresponding value function.
1: **if** $X_0 \notin$ UnderApproximation(H,q), **then**
2: Return "X_0 may not be controllable".
3: **end if**
4: $t_0 := 0; V := 0;$
{*Piecewise Affine Resolution*}
5: **for all** time step i (from 1 to r) **do**
6: {*Linear Approximations in a mode*}:
Compute the several affine approximations $(A_{q'}, B_{q'}, c_{q'})_{q'}$ of a mode q_i using equation (4)
7: {*Output Condition*}: $target := G_{(q_i, q_{i+1})}$;
8: {*Mixed and State Inequality Constraints*}
$Column := \mathcal{U}_{q_i}; Cell := D_{q_i};$ using equation (8)
9: {*Solve the piecewise affine problem in mode* q_i}
$(\mathcal{P}_{\mathcal{H}}(q_i)) \rightarrow (X(.), u(.), t_f, V_f)$ using section (0.4.2)
10: $X_0 := X(t_f); t_{i+1} := t_i + t_f; V := V + V_f;$
11: **end for**
12: Return (τ, X, u, V).

0.4.3 Example

Let us consider the nonlinear dynamical system:

$$\dot{x}(t) = x^2(t) + x(t)u(t) + u(t)^2 \tag{10}$$

The problem is to control the system (10) in minimum time from a given initial point $X_0 \in \mathbb{R}$ to the origin 0 under the control constraint: $\forall t \geq 0, -1 \leq u(t) \leq 1$.

Exact resolution

Let us consider the vector field: $f(x, u) = x^2 + xu + u^2$ as a polynomial of degree 2 in the variable u. Its discriminant is: $-3x^2 < 0$; we thus deduce:

$\forall (x,u), f(x,u) > 0$. Hence $\forall t \geq 0, \dot{x}(t) > 0$, so that $x(.)$ is increasing and the controllability conditions: $x_0 \leq 0$. We can also numerically check that the exact controllable domain of our problem is $]-\infty, 0]$.

Let us introduce the Hamiltonian: $H(X,u,\lambda) = 1+\lambda(x^2+xu+u^2)$. According to the Pontryagin minimum principle ([32]), we come down to the problem of minimizing H with respect to u under the constraints:

$$\begin{aligned} &\dot{\lambda}(t) = -2x(t)\lambda(t) - \lambda(t)u(t) \\ &H(X(t),u(t),\lambda(t) = 0 \text{ along the optimal trajectory} \end{aligned} \tag{11}$$

Let us first assume that the optimal control takes values in $]-1,1[$. We thus have to solve: $\frac{\partial H}{\partial u} = 0 \Leftrightarrow \lambda x + 2\lambda u = 0$ under the condition: $\frac{\partial^2 H}{\partial u^2} > 0$.

As $H \equiv 0$ along the optimal trajectory, we necessarily have: $\lambda \neq 0$ (otherwise: $H \equiv 1$), so that: $u^* = -\frac{x}{2}$ while $-2 \leq x \leq 2$ (to fulfill the control constraints). In consequence, we can now explicitly solve the system (10) and we obtain:

$$x(t) = 4x_0(4-3x_0t)^{-1} \text{ and } \lambda(t) = -\frac{4}{3}x(t)^{-2}$$

Therefore $\lambda(t) < 0$ and the previous u^* is not minimizing H. Hence: $\forall t \geq 0, u(t) \in \{-1,1\}$.

To choose between $u^* = -1$ and $u^* = 1$, we compute the value function associated to each possible control. The comparison of these functions shows that $u^* = -1$ is the exact solution of our initial nonlinear optimal control problem and the value function is:

$$V(x_0) = -\frac{1}{9}(6\ arctan(\frac{1}{3}(-1+2x_0)\sqrt{3}) + \pi)\sqrt{3}$$

Hybrid Approximation

Let $h = 1/N$ our discretization step. Let us build a simplicial mesh $(D_k)_k$ of $\mathbb{R}$ (see methods in §0.2.2): here we have: $D_k = [kh, (k+1)h]$.

Then the control domain $[-1,1]$ is subdivided into N intervals $[u_i, u_{i+1}],\ i = 0 \ldots N-1$, where: $u_i = -1 + 2ih$.

Lastly we just have to triangulate $[kh,(k+1)h] \times [u_i, u_{i+1}]$ and we can thus define our global mesh Δ of $\mathbb{R} \times [-1,1]$ by:

$$\begin{aligned} \Delta_{k,i}^{(1)} &= \{(x,u); 0 \leq u - u_i \leq x - kh \leq h\} \\ \Delta_{k,i}^{(2)} &= \{(x,u); 0 \leq x - kh \leq u - u_i \leq h\} \end{aligned}$$

So we now perform the hybrid approximation f_h over the cells $\Delta_{k,i}^{(j)},\ j \in \{1,2\}$:

$$f_h(x,u) = a_{k,i}^{(j)}x + b_{k,i}^{(j)}u + c_{k,i}^{(j)}, \text{ for } (x,u) \in \Delta_{k,i}^{(j)}$$

where:

$$
\begin{aligned}
a_{k,i}^{(1)} &= 2kh + h - 1 + 2ih \\
b_{k,i}^{(1)} &= -2 + 3h + kh + 4ih \\
c_{k,i}^{(1)} &= -k^2h^2 + kh - 2ih^2k - 1 + 4ih - 4i^2h^2 - h^2k + 3h - 6ih^2
\end{aligned}
$$

$$
\begin{aligned}
a_{k,i}^{(2)} &= 2kh + 3h - 1 + 2ih \\
b_{k,i}^{(2)} &= kh - 2 + 4ih + 2h \\
c_{k,i}^{(2)} &= -k^2h^2 + kh - 2ih^2k - 1 + 4ih - 4i^2h^2 - 3h^2k + 2h - 4ih^2
\end{aligned}
$$

We have defined the hybrid automaton related to the considered optimal control problem. Next, we will therefore apply the algorithm described in section 0.4.

Let x_0 be a given initial point: $x_0 \in D_k$ with $k = E[\frac{x_0}{h}]$ and a fixed sequence of states $q = (D_j)_{j=0\ldots k}$; we then want to compute the optimal solution of the hybrid problem over the path q:
As presented in §0.4.2, the first step is to solve the local affine optimal control problem in each cell of Δ and we have:

$$
u^* = \begin{cases} u_i & \text{when } (x,u) \in \Delta_{k,i}^{(1)} \\ x + u_i - kh & \text{when } (x,u) \in \Delta_{k,i}^{(2)} \end{cases}
$$

After that, we have to find the optimal cell $\Delta_{k,i}^{(j)},\ j \in \{1,2\},\ i = 0,\ldots,N$ to consider. We then can notice that for all i, $\Delta_{k,i}^{(1)}$ and $\Delta_{k,i}^{(2)}$ play respectively the roles of the cells a and b of the figure 13, so that:

$$
u^* = u_i \text{ when } (x,u) \in \Delta_{k,i}^{(1)} \cup \Delta_{k,i}^{(2)}
$$

Recursively, we can so conclude that over the whole columns on cells $\Delta_{k,i}^{(j)},\ j \in \{1,2\},\ i = 0,\ldots,N$, the optimal choice is the cell $\Delta_{k,0}^{(1)}$ with $u^* = -1$. We can then compute the local value function $V_k(x_0)$. x_0 is then re-initialized to kh in the cell D_{k-1}, and so on.

So our HybridSolver algorithm returns the optimal solution to our hybrid optimal control problem:

$$
u^* = -1 \text{ and } V_h(x_0) = \sum_{j=0}^{k} V_j(x_j) \text{ where: } \begin{cases} x_0 \text{ given} \\ x_j = (k-j)h,\ j = 1\ldots k \end{cases}
$$

Figure 14 shows in dot the real value function $V(x_0)$ and two hybrid value functions for two different discretization steps h. Our approximation converges towards the real curve.

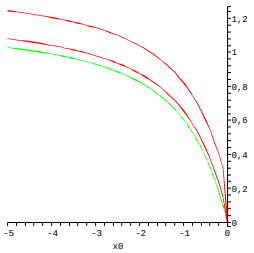


Figure 14: Hybrid approximations of the exact value function (in green dashed-dots) for: $h = .1$ and $h = 0.05$

0.5 Conclusion

In this paper, we address the problem of hybrid optimal control of non-linear dynamical systems. First we propose a hybrid approximation, by way of a hybrid automaton with piecewise affine dynamics, of complex systems. Then we solve the affine optimal control problem and give an explicit and fully analytical algorithm.

This algorithm however guarantees only a local optimization. Next step will be to give a way to find a sequence of cells containing an optimal trajectory. Several directions to solve this problem include:

- Exploring different sequences of states. This could unfortunately induce a combinatorial explosion.
- Partial numerical simulations could give some information on the localization of an optimal trajectory and thus reduce the exploration.
- Replace $l(X(t), u)$, the cost function with an admissible variable change $ds = l$, so that the problem comes down to a time optimal control problem. Finding the optimal sequence would then be to minimize the time to reach 0. To do this, one can use a time reversal, then perform an attainability test on X_0. The first time step when X_0 is reached gives an estimation of the total time and moreover the direction from which it was reached. This direction is the new goal of the affine optimal control. When a guard is reached in this direction, this gives a new point X_1, closer to zero than X_0, and from which we iterate the same loop.
- Another idea would be to perform an optimal control in the whole space with the local affine system. This would also give a possible direction. Unfortunately, convergence is then not guaranteed anymore.

.1 Proof of the correctness of the *Fast Block Kalman form* algorithm

Algorithm `CompressedKrylovMatrix` builds a matrix $\overline{K}$ satisfying

$$A\overline{K} = \overline{K}H$$

where H has a polycyclic Hessenberg form: upper block triangular with companion blocks on the diagonal and upper blocks full of zeroes except on the last column [2, Definition 4]:

$$\overline{K}^{-1}A\overline{K} = \begin{bmatrix} \boxed{\begin{matrix} 0 & & & * \\ 1 & 0 & & * \\ & \ddots & \ddots & * \\ & & 1 & * \end{matrix}} & & & * \\ & & & * \\ & & & * \\ & & & * \\ & \ddots & & \\ & & \boxed{\begin{matrix} 0 & & & * \\ 1 & 0 & & * \\ & \ddots & \ddots & * \\ & & 1 & * \end{matrix}} & \end{bmatrix} \tag{12}$$

Now, we have $AT = \left[\begin{array}{c|c} A\overline{K} & AP^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \end{array}\right]$

and there remains to find C_1 and C_2 such that $AT = T\begin{bmatrix} H & C_1 \\ 0 & C_2 \end{bmatrix}$ i.e. $T\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = AP^T\begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix}$. By writing $A' = PAP^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$ we get

$$T\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = P^T A'\begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} = P^T\begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix}$$

But $T = P^T\begin{bmatrix} U_1^T & 0 \\ U_2^T & I_{n-r} \end{bmatrix}\begin{bmatrix} L^T & 0 \\ 0 & I_{n-r} \end{bmatrix}$ so that

$$\begin{bmatrix} U_1^T & 0 \\ U_2 & I_{n-r} \end{bmatrix}\begin{bmatrix} L^T & 0 \\ 0 & I_{n-r} \end{bmatrix}\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix}$$
$$\begin{bmatrix} U_1^T & 0 \\ U_2^T & I_{n-r} \end{bmatrix}\begin{bmatrix} L^T C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix}$$

This give the following system:

$$\begin{cases} U_1^T L^T C_1 = A'_{12} \\ U_2^T L^T C_1 + C_2 = A'_{22} \end{cases}$$

which solutions are

$$\begin{cases} C_1 &= L^{-T}U_1^{-T}A'_{12} \\ C_2 &= A'_{22} - U_2^T U_1^{-T} A'_{12} \end{cases}$$

And therefore

$$T^{-1}AT = \begin{bmatrix} H & C_1 \\ 0 & C_2 \end{bmatrix},$$

Now, as columns of B are linear combinations of columns of $\overline{K}$, this gives

$$B = T \begin{bmatrix} B_1 \\ 0 \end{bmatrix}$$

And thus $B_1 = L^{-T}U_1^{-T}PB$.

The complexity of the compression is either $O(n\omega \log(n)$ or $O(n^3)$ depending on the chosen method. The remaining steps in algorithm 2 are $n \times n$ LUP factorizations, multiplications and triangular solvings all of which requires at most $O(n^\omega)$ operations.

.2 Proof that the control constraints are m-simplices

Proof. We take $q' \in \mathcal{K}(q)$. The problem is to compute the intersection of the affine subspace $P = \{(X_0, u)/u \in \mathbb{R}^m\}$ in $\mathbb{R}^{n+m}$ and the simplex $\Delta_{q'}$ (see figure 15):

$$U_{q'}(X_0) = \{(X_0, u) \ / \ u \in \mathbb{R}^m\} \cap \Delta_{q'} = P \cap \Delta_{q'}$$

We know that the intersection of an affine subspace with a simplex is a simplex

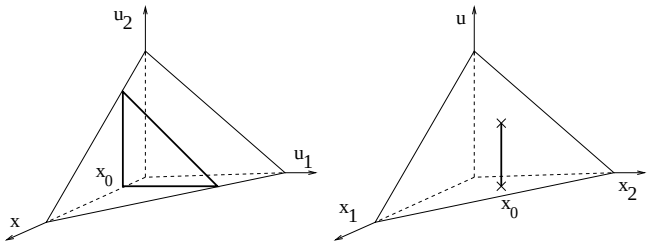


Figure 15: Intersection $P \cap \Delta_{q'}$ for $(n, m) = (1, 2)$ on the left, $(n, m) = (2, 1)$ on the right.

[36, Theorem 1.1]. Therefore $U_{q'}(X_0)$ is a simplex in $\mathbb{R}^m$. There remain to to compute its dimension. The idea is to consider the intersection of the subspace P of dimension m with the n-faces of $\Delta_{q'}$. These intersections, when they exist, are of dimension 0, and define the vertices of $U_{q'}(X_0)$. We then have to prove that there exists exactly $(m + 1)$ different n-faces intersecting P to conclude.

Let $\{X_1, \ldots, X_{n+1}\}$ be the vertices of D_q. By construction, we have: $p^{\perp}_{\mathbb{R}^n}(\Delta_{q'}) = D_q$. We can then introduce the vertices of $\Delta_{q'}$:

$$V_k = \begin{bmatrix} X_{i_k} \\ u_{j_k} \end{bmatrix},\ i_k \in \{1, \ldots, n+1\},\ j_k \in \{1, \ldots, m+1\},\ k = 1, \ldots, n+m+1$$

and: $\{X_{i_k}\ ;\ k = 1, \ldots, n+m+1\} = \{X_1, \ldots, X_{n+1}\}$.
We also assume: $X_0 \in \mathring{D}_q$, hence:

$$\exists!(\alpha_k)_{k=1\ldots n+1} \in]0,1[^{n+1},\ \sum_{k=1}^{n+1} \alpha_k = 1 \text{ et } X_0 = \sum_{k=1}^{n+1} \alpha_k X_k \tag{13}$$

1^{st} step: any n-face of $\Delta_{q'}$ has either one intersection with P, or no one.
Let F be a n-face of $\Delta_{q'}$ defined by the list (renumbered to make the readability easier) of its vertices: $(\begin{bmatrix} X_{i_k} \\ u_{j_k} \end{bmatrix})_{k=1,\ldots,n+1}$ where $i_k \in \{1, \ldots, n+1\}$, $j_k \in \{1, \ldots, m+1\}$.

- 1^{st} *case: the i_k are all different.*
  We then have: $\{X_{i_k}\ /\ k = 1, \ldots, n+1\} = \{X_1, \ldots, X_{n+1}\}$. (13) so becomes: $X_0 = \sum_{k=1}^{n+1} \alpha_{i_k} X_{i_k}$. We state: $X_F = \sum_{k=1}^{n+1} \alpha_{i_k} \begin{bmatrix} X_{i_k} \\ u_{j_k} \end{bmatrix} = \begin{bmatrix} X_0 \\ \sum_{k=1}^{n+1} \alpha_{i_k} u_{j_k} \end{bmatrix}$.
  Then: $X_F \in P$. Moreover, by construction of X_F, $X_0 \in \mathring{D}_q$ involves: $X_F \in \mathring{F}$, hence: $X_F \in P \cap F$.
  By uniqueness of the convex decomposition in a given simplex, we then deduce:
  $$P \cap F = \{X_F\} \text{ and } X_F \in \mathring{F}$$

- 2^{nd} *case: the i_k are not all different.*
  We then deduce: $\{X_{i_k}\ /\ k = 1, \ldots, n+1\} \subsetneq \{X_1, \ldots, X_{n+1}\}$. For $Y \in F$, we so have:
  $$\exists!(\gamma_{i_k})_{k=1\ldots n+1} \in [0,1]^{n+1},\ \sum_{k=1}^{n+1} \gamma_{i_k} = 1 \text{ et } Y = \sum_{k=1}^{n+1} \gamma_{i_k} \begin{bmatrix} X_{i_k} \\ u_{j_k} \end{bmatrix}$$
  However, according to the hypothesis: $X_0 \in \mathring{D}_q$, X_0 depends on all the X_i, $i = 1, \ldots, n+1$. It follows: $\sum_{k=1}^{n+1} \gamma_{i_k} X_{i_k} \neq X_0$, i.e.: $Y \notin P$, and: $F \cap P = \emptyset$.

2^{nd} **step: there exist exactly** $(m+1)$ **n-faces intersecting** P**.**
By construction, we know that: $\{X_{i_k} \ / \ k = 1, \ldots, n+m+1\} = \{X_1, \ldots, X_{n+1}\}$; after renumbering the vertices of $\Delta_{q'}$, we can assume:

$$\forall l \in \{1, \ldots, n+1\},\ V_l = \begin{bmatrix} X_l \\ u_{j_l} \end{bmatrix}$$

- We state: $F_0 = Conv(\{V_1, \ldots, V_{n+1}\})$. By affine independence of $X_1, \ldots, X_{n+1}$ of D_q, the vertices $V_1, \ldots, V_{n+1}$ are also affinely independent, so that F_0 is actually a n-face of $\Delta_{q'}$ intersecting P.
- For $n+2 \leq l \leq n+m+1$ (i.e. m possible values for l), we state:

  $$F_l = Conv((\{V_1, \ldots, V_{n+1}\} - \{V_{i_l}\}) \cup \{V_l\}),\ 1 \leq i_l \leq n+1$$

  where: $V_l = (X_{i_l}, u_{j_l})$ and $V_{i_l} = (X_{i_l}, u_{j_{i_l}})$. As for F_0, F_l is a n-face of $\Delta_{q'}$ which intersects P.

At the end, we so have found $(1+m)$ n-faces of $\Delta_{q'}$ which intersect P, i.e. $(m+1)$ vertices of $U_{q'}(X_0)$. □

Bibliography

[1] E. Asarin, T. Dang, and A. Girard. Reachability of non-linear systems using conservative approximations. In *Proceedings of the 2003 Hybrid Systems: Computation and Control*, pages 20–35. Springer, April 2003.

[2] D. Augot and P. Camion. On the computation of minimal polynomials, cyclic vectors, and Frobenius forms. *Linear Algebra and its Applications*, 260(1–3):61–94, July 1997.

[3] C.B. Barber, D.P. Dobkin, and H.T. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 1996.

[4] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, volume 17 of *Systems & Control: Foundations & Applications*. Birkauser, 1997.

[5] G. Barles. *Solutions de viscosité des équations de Hamilton-Jacobi*, volume 17 of *Mathématiques et Applications*. Springer-Verlag, 1994.

[6] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[7] R. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 & 2. Athena Scientific, 1984.

[8] A.E. Bryson and Y. Ho. *Applied Optimal Control*. Hemisphere, 1975.

[9] I. Capuzzo-Dolcetta. *On a Discrete Approximation of the Hamilton-Jacobi Equation of Dynamic Programming*, volume 10 of *Applied Mathematics and Optimization*, pages 367–377. Springer-Verlag, 1983.

[10] F. H. Clarke. *Optimization and Nonsmooth Analysis*. SIAM Classics in Applied Mathematics, 1990.

[11] M.G. Crandall, L.C. Evans, and P.-L. Lions. Some properties of viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 282(2):487–502, 1984.

[12] M.G. Crandall, H. Ishii, and P.-L. Lions. Uniqueness of viscosity solutions revisited. *Journal of Mathematical Society*, 39:581–596, 1987.

[13] M.G. Crandall, H. Ishii, and P.-L. Lions. A user's guide to viscosity solutions. *Bulletin A.M.S., N.S.*, 27:1–67, 1992.

[14] M.G. Crandall and P.-L. Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[15] J. Della Dora, A. Maignan, M. Mirica-Ruse, and S. Yovine. Hybrid computation. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*. Bernard Mourrain editor, ACM Press, July 2001.

[16] J.-G. Dumas, T. Gautier, and C. Pernet. Finite field linear algebra subroutines. In Teo Mora, editor, *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation,*, pages 63–74. ACM Press, New York, jul 2002.

[17] J.-G. Dumas, P. Giorgi, and C. Pernet. FFPACK: Finite Field Linear Algebra Package. In Jaime Gutierrez, editor, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, Santander, Spain*. ACM Press, New York, July 2004.

[18] J.-G. Dumas, C. Pernet, and Z. Wan. Efficient computation of the characteristic polynomial. In *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, Beijing, China*, 2005.

[19] J.-G. Dumas and A. Rondepierre. Modeling the electrical activity of a neuron by a continuous and piecewise affine hybrid system. In *Proceedings of the 2003 Hybrid Systems: Computation and Control*, pages 156–171. Springer, April 2003.

[20] J.-G. Dumas and A. Rondepierre. Algorithms for symbolic/numeric control of affine dynamical systems. In *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, Beijing, China*, 2005.

[21] R. Fierro, A. K. Das, V.Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. 2001.

[22] H. Freudenthal. Simplizialzerlegungen von beschraeukter flachkeit. *Annals of Math. in Science and Engin.*, 43:580–582, 1942.

[23] A. Girard. Approximate solutions of ordinary differential equations using piecewise linear vector fields. In *Proceedings of the 2002 Computer Algebra in Scientific Computing*. Springer Verlag, September 2002.

[24] A. Girard. *Analyse Algorithmique des Systèmes hybrides*. PhD thesis, Institut National Polytechnique, Grenoble, 2004.

[25] R.E. Kalman. Canonical structure of linear dynamical systems. In *Proceedings of the National Academy of Sciences*, pages 596–600, 1961.

[26] R.E. Kalman. Mathematical description of linear dynamical systems. *Siam Journal on Control*, 1:152–292, 1963.

[27] R.E. Kalman, P.A. Falb, and M. Arbib. *Topics in Mathematical System Theory*. New York: McGraw-Hill, 1969.

[28] W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical computer science*, 36:309–317, 1985.

[29] M.M. Lee and L. Markus. *Foundations of Optimal Control Theory*. Whiley, New York, 1967.

[30] H.J. Pesch. A practical guide to the solutions of real-life optimal control problems. *Parametric Optimization. Control Cybernet*, 23:7–60, 1994.

[31] E.R. Pinch. *Optimal Control and the Calculus of Variations*. Oxford University Press, 1993.

[32] L. Pontryagin, V. Boltiansky, R. Gamkrelidze, and E. Michtchenko. *Théorie mathématique des processus optimaux*. Editions de Moscou, 1974.

[33] A. Rondepierre. Piecewise affine systems controllability and hybrid optimal control. In *Proceedings of the 2005 International Conference on Informatics in Control, Automation and Robotics*, 2005.

[34] E. Trélat. *Contrôle optimal: théorie et applications*. Collection Mathématiques Concrètes. Vuibert, 2005.

[35] J. Zhang, K.H. Johansson, J. Lygeros, and S. Sastry. Zeno hybrid systems. *International Journal of Robust and Nonlinear Control*, 11:435–451, 2001.

[36] G.M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer-Verlag, 1994.