Algorithms for Symbolic/Numeric Control of Affine Dynamical Systems

Jean-Guillaume Dumas
Université Joseph Fourier,
Laboratoire de Modélisation et Calcul,
50 av. des Mathématiques.
B.P. 53 X, 38041 Grenoble, France.
Jean-Guillaume.Dumas@imag.fr

Aude Rondepierre
Université Joseph Fourier,
Laboratoire de Modélisation et Calcul,
50 av. des Mathématiques.
B.P. 53 X, 38041 Grenoble, France.
Aude.Rondepierre@imag.fr

ABSTRACT
We consider a general linear dynamical system and want to control its behavior. The goal is to reach a given target by minimizing a cost function. We provide a new generic algorithm with together exact, symbolic and numerical modules. In particular new efficient methods computing a block Kalman canonical exact decomposition and the optimal solutions are presented. We also propose a new numerical algorithm under-approximating the controllable domain in view of its analytical resolution in the context of singular sub-arcs.

Categories and Subject Descriptors:
I.1.2 [Symbolic and algebraic manipulation]: Algorithms, symbolic computations.
J.1.7 [Computer Applications]: Command and control.

General Terms:
Algorithms, symbolic computation.

Keywords:
Affine Optimal Control Problems, Canonical Transformation, Controllability.

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 [7] ([16] for more applications in aerospace industry). Optimal control has become a more and more challenging domain and its theory has been extensively developed for many years. Nevertheless, the problem of synthesis of optimal feedback is not solved, even for linear systems. In some specific cases like time-optimal control problems, adequate solutions have been found [18, §3],[2, 17, 16]. Also, control theory lacks generic algorithms, specially when singular sub-arcs appear [14, 19, 2]. Furthermore, in “real-life”, optimal control problems are often nonlinear. Therefore most of the algorithms presented here have been developed towards their application to the hybrid control of nonlinear dynamical systems: in [20], we propose a piecewise affine approximation by way of a hybrid automaton. In each cell, the local optimal control problem is affine and techniques developed here will be applied. In this paper we consider a linear dynamical system:

$$\begin{align*}
\dot{X}(t) &= AX(t) + Bu(t) \\
X(0) &= X_0
\end{align*}$$

where $\forall t \geq 0, X(t) \in \mathbb{R}^n$ and $u(t) \in U_m = \{ s_1, \ldots, s_p \} \subset \mathbb{R}^m$ is the control. We want to control the system (1) from an initial state $X_0$ to a target $X_f = 0$ at an unspecified time $t_f$, in such a way that the functional: $J(X_0, u(\cdot)) = \int_{t_0}^{t_f} l(X(t), u(t)) \, dt$ is minimized.

Here, we provide a full implementation analyzing linear optimal control problems as general as possible. Our algorithm is divided in four steps:

1. Canonical transformation (see §2).
2. Approximation of the controllable set (see §3.2).
3. Computation of optimal solutions (see §4).
4. Inverse transformation (see §2.2).

Each step can be done in several different ways and some salient features of our presentation are:

- a new and more efficient implementation by block matrices of the exact computation of Kalman decomposition.
- symbolic computation of the bound of domain, where the optimal control is constant (see e.g. algorithms 5 and 6).
- a new numerical method to compute an under-approximation of the controllable domain.
- an efficient implementation of the optimal solution computation, for a very large class of cost functions using subroutines previously defined (see algorithm 7).

Our approach enables the high dimensions treatment, even when compared to numerical softwares. Indeed, numerical methods developed from the Hamilton-Jacobi-Bellman
(HJB) theory are known to suffer from the dimension: they generally require to generate a grid over a bounded region of the state space. 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. aerospace [16] requires treatments of dimensions 6 or 7. By the use of Hybrid Computation [4] combining numerical analysis and computer algebra, we are now able to deal with high dimensions (see [20, Part II]): first the mesh is made on the fly to reduce the complexity. Then, at the vertices of the mesh, only a linear interpolation of our complex system is performed. In each cell, the system is linear and one need to develop methods as symbolic as possible: basically, an analytical approach must allow to improve the hybrid approximation.

The paper is organized as follows: in section 2, we will develop explicit algorithms to compute exactly the canonical transformation of any linear optimal control problem and then the exact inverse transformation. In section 3, we provide a numerical controllability analysis and then in section 4, the analytical computation of optimal solutions of the canonical problem.

2. CANONICAL TRANSFORMATION

Linear control systems have been widely analyzed. In [13, 12], Kalman considers constant linear optimal control problems without constraints on the control. In this context, we have two well-known results: the first one is a controllability criterion (see [13, 3] for more details), and the second is the following decomposition theorem:

**Theorem 1** ([12] Kalman 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:

\[
T^{-1}AT = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix}, \quad T^{-1}B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}
\]

where \( r = \text{rk}(\begin{bmatrix} AB & \ldots & A^{n-1}B \end{bmatrix}) = \text{rk}(\begin{bmatrix} B_1 & A_1B_1 & \ldots & A_1^{n-1}B_1 \end{bmatrix}) \)

\( A_1 \) is a \( r \) dimensional square matrix and \( B_1 \) a \( r \times m \) matrix.

There exist many numerical algorithms computing the Kalman canonical form of full rank linear dynamical systems. Next, we consider rank deficient systems, for which exact computation of the rank is needed. Therefore we propose a new explicit and exact algorithm for the Kalman decomposition. Our approach is to use block versions of the linear algebra algorithms as in [5] 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 [6] (nowadays quite as fast as numerical routines) to perform the decomposition.

2.1 Block Canonical decomposition

We consider the general linear system (1). Our decomposition is divided into two steps: first we reduce the system to one with a full rank mapping of the control and second apply a LQUP decomposition to the Kalman matrix.

2.1.1 Simplification to \( \text{rk}(B) = m \)

**Lemma 1.** Let us consider the linear system (1). There exists a full rank matrix \( B \in \mathbb{R}^{n \times \text{rk}(B)} \) and a linear mapping \( \Phi \in \mathbb{R}^{\text{rk}(B) \times m} \) such that: \( X(t) = AX(t) + B \Phi u(t) \).

**Proof.** \( b = \text{rk}(B) \). If \( b < m \), then there exists a column permutation \( P \in \mathbb{R}^{m \times m} \) s.t.: \( BP = [\tilde{B}]B_0 \) where \( \tilde{B} \in \mathbb{R}^{n \times b} \) and \( \text{rk}(\tilde{B}) = b \). Moreover, the column vectors of \( B_0 \) are linearly dependent of those of \( \tilde{B} \), i.e.: \( \exists \lambda \in \mathbb{R}^{b \times \text{rank}(\tilde{B})} : B_0 = \lambda \tilde{B} \). Hence: \( B = \tilde{B} \left[ I_n | \lambda \right] P^{-1} \) and \( \Phi = \left[ I_n | \lambda \right] P^{-1} \).

In the following, we will denote by FullRank\((b)\) the algorithm computing \((b, \tilde{B}, \Phi)\) from a matrix \( B \) as in the lemma.

2.1.2 Block Kalman Canonical Form

Now we want to decompose the state space of our linear system into a controllable part and an uncontrollable one. The classical method is to introduce the linear subspace \( W(A,B) = \text{span}(B,AB,\ldots,A^{n-1}B) \) and then prove that \( W \) is the first subspace of \( \mathbb{R}^n \) satisfying both: (i) \( \text{Im}(B) \subset W \) (ii) \( W \) is \( A \)-invariant. The method is then to decompose the state space \( \mathbb{R}^n \) into \( W \oplus W \): one has to compute a basis of the subspace \( W(A,B) \) and to complete it for the whole state space. The matrix \( T \) of theorem 1 would be the change matrix from the canonical basis to the computed basis.

In this paper we propose a new approach via block matrix computation developed in collaboration with C. Pernet: we use the so-called LQUP decomposition of a \( x \times y \) matrix of rank \( r \), where \( U = \begin{bmatrix} U_1 & U_2 \\ 0 & 0 \end{bmatrix} \) is \( x \times y \), \( U_1 \) is an upper triangular \( r \times r \) invertible matrix, \( L \) is \( x \times x \), lower triangular, and \( P \) and \( Q \) are permutation matrices [10].

**Algorithm 1** BlockKalmanForm

**Require:** \( A \) \( n \times n \) matrix, \( B \) \( n \times m \) matrix.

**Ensure:** \( r,T,A_1,A_2,\Lambda,B_1 \) as in theorem 1.

1: \( K = [B AB \ldots A^{n-1}B] \);
2: \( (L,Q,U_1,U_2,P,r) = \text{LQUP}(K^T) \);
3: if \( r = n \) then
4: end if
5: \( \text{Return} \ (n, I_n, A, \emptyset, \emptyset, B) \).

6: Form \( \delta = [I_r | 0] Q^T L \begin{bmatrix} I_r \\ 0 \end{bmatrix}, \) lower triangular.
7: Form \( d = [I_{r+1 \ldots m}] | 0] Q^T L \begin{bmatrix} I_r \\ 0 \end{bmatrix} \).
8: \( G = [I_r | 0] Q^T K^T \).
9: \( C_1 = G(A^T P^T \begin{bmatrix} I_r \\ 0 \end{bmatrix} U_{1-1}^{-1} \delta^{-1}) \).
10: \( C_2 = [0 | I_{n-r}] P(A^T P^T \begin{bmatrix} I_r \\ 0 \end{bmatrix} U_{1-1}^{-1} \delta^{-1}) \).
11: \( C_3 = [0 | I_{n-r}] P A^T P^T \begin{bmatrix} -U_{1-1}^{-1} U_2 \\ I_{n-r} \end{bmatrix} \).
12: \( Q_1 = [I_m | 0] \begin{bmatrix} I_r \\ d d^T \end{bmatrix} \{Q_1 \} m \times r \)
13: \( \text{Return} \ (r, [G | 0] I_{n-r} P^T, C_1^T, C_2^T C_3^T, Q_1^T) \).

**Theorem 2:** Algorithm 1 is correct and its arithmetic complexity is \( O(n^r m^2) \).\(^{1}\)

\(^{1}\)where \( \omega \) is the exponent of matrix multiplication (3 for the classical algorithm and 2.3755 for Coppersmith-Winograd’s)
Proof. [20] It has three parts and is actually another, constructive, proof of Kalman’s theorem:
1. First, use the generalization of the companion matrix decomposition to prove that \( GA^T = C_1G \).
2. Second, use the latter to show that \( T^{-1}AT \) is block triangular.
3. Show that \( T^{-1}B \) has generic rank profile.

Now for the complexity: building the Kalman matrix is \( n \) matrix multiplications \( n \times n \) by \( n \times m \), each requiring \( O(n^{w-1}-m) \) operations. Following [6, Lemma 4.1], the LQUP decomposition requires \( O(n^{w-1}\text{mn}+n) \) operations. Those two costs dominate the remaining operations: two triangular inversions \( O(r^2) \), some permutations and column selections, and small matrix multiplications \( (GA^T) \) is \( O(rn^{w-1}) \) and \( d\delta^{-1} \) is \( O(nmr^{w-1}) \) where \( r \leq n \). □

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 [6]. Therefore the computation time is improved. Moreover if we first apply the algorithm FullRank of paragraph 2.1.1, the system (1) can be replaced by another one:
\[
\dot{Y}(t) = \begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix} Y(t) + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u(t)
\]
via possibly two variable changes: \( \{ Y(t) = T^{-1}X(t), \quad \dot{u}(t) = \Phi u(t) \} \)

Next, we use these decomposition in order to define a canonical optimal control problem, simpler to solve.

2.2 Inverse transformation

In this section, the focus is on the explicit construction of a new linear optimal control problem under the dynamic (2). A new cost function and new state and control spaces have to be constructed and initial solutions have to be recovered.

2.2.1 Control Space

In this paragraph we focus on the construction of a new control space for the linear system (2). By assumptions (see section 1), \( \exists \) control space for the linear system (2). By assumptions (see section 1), \( \exists \) control space for the linear system (2). By assumptions (see section 1), \( \exists \) control space for the linear system (2).

2.2.2 State Space

By construction, the change matrix \( T \) is non singular. Therefore, a trajectory \( Y(\cdot) \) from an initial point \( Y_0 \) corresponds to a trajectory \( X(\cdot) = TY(\cdot) \) from the initial point \( X_0 = TY_0 \). Every trajectory is necessarily related to a control, the table 1 displays the correspondence between each trajectory.

<table>
<thead>
<tr>
<th>Initial Problem (1)</th>
<th>Canonical Problem (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((X(\cdot), u(\cdot)))</td>
<td>((T^{-1}X(\cdot), \Psi u(\cdot)))</td>
</tr>
</tbody>
</table>

Table 1: Matching trajectories

Proof. The key point here is that a trajectory \((X(\cdot), u(\cdot))\) in the \( X \)-space is a solution of the system (1):
\[
\begin{align*}
X(t) & = e^{AT}X_0 + \int_{t}^{\infty} e^{-A\omega}BU(\omega)d\omega \\
T^{-1}X(t) & = e^{(T^{-1}AT)^Tt}X_0 + \int_{t}^{\infty} e^{-(T^{-1}AT)^T\omega}T^{-1}BU(\omega)d\omega \\
T^{-1}X(t) & = e^{(T^{-1}AT)^Tt}X_0 + \int_{t}^{\infty} e^{-(T^{-1}AT)^T\omega}T^{-1}BU(\omega)d\omega \\
\end{align*}
\]

Then \((T^{-1}X(\cdot), \Phi u(\cdot))\) is a solution of (2), i.e. a trajectory in the \( Y \)-space. □

2.2.3 Cost Function

Let \( X_0 \) be a controllable point. The value function related to the initial control problem (1) is defined by: \( V(X_0) = \inf_{u(\cdot)} \int_{0}^{\infty} l(X(t), u(\cdot))dt \). We want to define a new value function \( \tilde{V}(Y_0) = \inf_{\tilde{u}(\cdot)} \int_{0}^{\infty} \tilde{l}(\tilde{Y}(t), \tilde{u}(\cdot))dt \) such that the two related optimal control problems are equivalent.

First, the idea is to define a new cost function \( \tilde{l} \), such that the value function is invariant by canonical transformation (i.e.: \( V(X_0) = \tilde{V}(T^{-1}X_0) \)). In this case, \( \tilde{l}(Y, \tilde{u}) \mapsto l(TY, \Psi \tilde{u}) \) and the new optimal control problem becomes:

\[
\text{“Minimize } \tilde{V}(Y_0, \tilde{u}(\cdot)) = \int_{0}^{\infty} \tilde{l}(\tilde{Y}(t), \tilde{u}(\cdot))dt \text{ with respect to the control } \tilde{u}(\cdot) \text{ under the dynamic (2) and the constraints: } \forall t \geq 0, \tilde{u}(t) \in \left( : \tilde{l}(\tilde{Y}, \tilde{u}) \right).
\]

We then have to verify that optimal solutions of this new problem correspond to optimal solutions of (1):

Proposition 2. Let \((Y^{\ast}(\cdot), \tilde{u}^{\ast}(\cdot))\) be an optimal solution of (3). Then \((TY^{\ast}(\cdot), \Psi \tilde{u}^{\ast}(\cdot))\) is an optimal solution of the initial problem (1) and \( V(TY_0) = \tilde{V}(Y_0) \).

(The proof is by inspection of \( J(X_0, \Psi \tilde{u}^{\ast}(\cdot)) \).

2.2.4 Algorithms

To conclude the section, we describe two algorithms: SimplifySystem and InverseTransformation. From one given optimal control problem, SimplifySystem allows to define the canonical optimal control problem (see §2.1); once this problem is solved, InverseTransformation exactly computes the related optimal solutions of the initial problem (1) by the use of proposition 2. In fact, the algorithms, the pseudo-inverse of \( \Psi \) is given e.g. by [21].

In this section we achieved the transformation of any linear optimal control problem into a canonical one. Moreover we have proved that optimal solutions of the canonical problem give optimal solutions of our initial problem. We have also proposed exact computation algorithms for switching
to one problem to the other. Now, we can work on the canonical problem.

3. CONTROLLABLE DOMAIN

In this section, we consider the canonical optimal control problem previously defined and raise the question of its controllability: how to compute the set of initial points $Y_0$ for which the control problem (2) with the constraints $Y(0) = Y_0$ : $Y(t) = 0$ and $\forall t \geq 0, u(t) \in U_m = \{s_1, \ldots, s_p\} \subset \mathbb{R}^m$ admits a solution.

Let us state: $\forall t \geq 0, Y(t) = (Y_1(t), Y_2(t))$ where: $Y_1(t) \in \mathbb{R}^r$ and $Y_2(t) \in \mathbb{R}^{n-r}$. Thus the state space splits clearly up into an uncontrollable part ($Y_2 = A_3 Y_2$) and a controllable one ($Y_1 = A_1 Y_1 + A_2 Y_2 + B_1 u(t)$). We study the controllability question in the two configurations.

3.1 Stabilization of the uncontrollable part

Let us consider the uncontrollable part:

$$Y_2(t) = A_3 Y_2(t) \tag{4}$$

Clearly, $0 \in \mathbb{R}^{n-r}$ is an equilibrium point of (4). Thus the target $0$ is reachable from anywhere if $0$ is a stable focus of (4). In other words, the matrix $A_3$ has to be stable (all its eigenvalues have negative real parts).

In the following, we prove that the non-stability of $A_3$ involves constraints on $Y_2(0)$, so that we can easily come down to the case of a stable matrix $A_3$: we apply the Schur decomposition to $A_3$ and choose to sort its eigenvalues in such a way that: $\forall i = 1 \ldots k, \text{Re}(\alpha_i) < 0$ and $\forall i = k + 1 \ldots (n-r), \text{Re}(\alpha_i) \geq 0$. Then there exists a unitary $Q \in \mathbb{C}^{n \times n}$ such that: $Q^* A_3 Q = D + N$ where $D = \text{diag}(\alpha_1, \ldots, \alpha_{n-r})$ and $N \in \mathbb{C}^{(n-r) \times (n-r)}$ is strictly upper triangular. Moreover (4) is easily solvable: $\forall t \geq 0, Y_2(t) = e^{A_3 t} Y_2(0)$. Hence:

$$Q^* Y_2(t) = e^{Q^* A_3 Q t} Q^* Y_2(0) = e^{Q^* (D + N) t} Q^* Y_2(0) = e^{Q^* \alpha_1 t} e^{Q^* N t} Q^* Y_2(0) = e^{Q^* \alpha_1 t} e^{Q^* \alpha_{n-r} t} Q^* Y_2(0)$$

Nevertheless we do not need to compute $e^{Nt}$. Indeed, we can recursively show (by starting from $n - r$ to $k + 1$) that: $$\forall t \geq 0, (e^{Q^* (D + N) t}) Y_2(0) = 0$$

Hence: $$\forall t \geq 0, (e^{Q^* \alpha_1 t} e^{Q^* N t} Y_2(0)) Y_2(t) = 0$$

So under the variable change: $Y_2 = (Q^* Y_2)_{1:k}$, the system (2) then becomes:

$$\begin{align*}
\dot{Y}_1(t) &= A_1 Y_1(t) + \bar{A}_2 Y_2(t) + B_1 u(t) \\
\dot{Y}_2(t) &= \bar{A}_3 Y_2(t)
\end{align*}$$

where: $\bar{A}_2 = A_2 Q \begin{bmatrix} I_k \\ 0 \end{bmatrix}$ and $\bar{A}_3 = (D + N) \begin{bmatrix} I_k \\ 0 \end{bmatrix}$ is stable.

We have shown that the analysis of the uncontrollable part of the system (2) leads to define a subspace of the state space, namely $\{(Y_1, Y_2, 0) \in \mathbb{R}^r \times \mathbb{R}^r \times \mathbb{R}^{n-r}\}$. In this subspace, $Y_2(\cdot)$ trajectories converge towards $0$. From now on, we therefore restrict our analysis to a system (2) where the matrix $A_3$ is stable.

3.2 Under-Approximation of the Controllable Domain

Now, we assume w.l.o.g that the points $s_i$ defining the control boundaries are such that: $s_i \notin \text{Conv}_{\text{polytope}}(s_i)$. Therefore, each point $s_i$ is a vertex of the polytope $U_m$ and we have (see §2): $rk(B) = m$, $rk([B\mid AB] \ldots [A^{n-1} B]) = n$.

We want to find the set of controllable points of our system. By time reversal, we come down to the computation of the attainable set from the target point $0$. In [1], for safety verification, the idea is to compute a conservative 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 need a guaranty that $Y_0$ is controllable. Therefore we instead compute an under-approximation of this set.

Let us start by defining the controllable set $C$ in our context:

$$C = \{Y \in \mathbb{R}^n, \exists u : [0, T] \rightarrow U_m, Y = \int_0^T e^{-A \tau} B u(\tau) d\tau\}$$

Indeed, any solution of a linear system $\dot{Y}(t) = AY(t) + Bu(t)$ has the form: $Y(t) = e^{A t} Y(0) + \int_0^t e^{A(t-s)} Bu(s) ds$.

**PROPOSITION 3.** The controllable domain $C$ is a convex subset of the state space.

The main idea of the proof (given in [20]) is to define (by convexity and at maximal time) a new control from that of some controllable points within $C$.

Now we can introduce our under-approximation of the domain by time-reversal of the control polytope:

**COROLLARY 1.** Let $Y_i(\cdot)$ be the trajectory from $0$ by time reversal according to $u$, $i$. If $C(t) = \text{Conv}_{1:k}(Y_i(t))$, then

$$C(t) \subset C \land \forall Y \in C(t), \exists a \text{ control } u, Y = \int_0^t e^{-A \tau} B u(\tau) d\tau$$

Any point in $C(t)$ is said controllable at least in time $t$ and $C(t)$ is an under-approximation of the controllable set in time $t$.

This gives us an algorithm to build our under-approximation in time $T$. Nevertheless for a given time $T$, the quality of the approximation could be very poor (see example 1, figure 1-(a)). To refine it, we choose to discretize the time interval $[0, T]$ in $N$ subintervals. The under-approximation in time $T$ is the convex hull of under-approximations in time $j \times h$ for
$j = 1..N-1$ (where $h = T/N$) and the quality is significantly improved (see example 1, figure 1-(b)). We have thus defined the following algorithm, UnderApproximation, computing a set of controllable points.

**Algorithm 4 UnderApproximation**

**Require:** $A,B,U,T,h$ (where $U = \text{Conv}(s_1, ..., s_p)$).

**Ensure:** an under-approximation with a step $h = T/N$ of the controllable domain in time $T$.

1. ApproxVertices := $\{0\}$
2. for all time step $j$ (from 1 to N) do
   3. for all vertex $s_i$: do
      4. $Y(_i \_j) = \text{trajectory from 0 with } u = s_i$
      5. ApproxVertices := ApproxVertices $\cup \{Y(_i \_j)\}$
   6. end for
3. end for
4. Return ConvexHull(ApproxVertices).

**Example 1 (2D Under-Approximations).** Let us consider the system: $\dot{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix} Y + \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} u$ with $u \in \text{Conv}([0,0]^T,[1,0]^T,[0,1]^T)$. The following figures show in dashes under-approximations of the controllable set (represented in plain line) for three refinements.

![Figure 1: Under-approximations in time $T = 5$ (a) without refinement ($N = 1$) (b) by discretizing ($N = 5$ in dash-dots - $N = 30$ in dashes, nearly superposed)](image)

4. **OPTIMAL SOLUTIONS**

In this section, we present some theoretical results and algorithms for solving linear optimal control problems. The algorithm is as general and symbolic as possible to design optimal controllers. Recall that we want to control a linear system: $\dot{Y}(t) = AY(t) + Bu(t)$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ from a controllable initial state $Y_0$ to a final state $Y_f = 0$ at an unspecified time $t_f$ by choosing the admissible control functions $u \in U_m = \text{Conv}_{\mu}(s_p) \subset \mathbb{R}^m$ in such a way that: $J(Y_0, u(\_)) = \int_0^{\infty} l(\dot{Y}(t), u(t)) \text{d}t$ is minimized. According to the decomposition algorithm developed in section 2, we also assume: $rk(B) = m$ and $rk([B|AB|...|A^{n-1}B]) = n$. To solve this canonical problem, we now introduce the Hamiltonian function: $H(Y, u, \lambda) = l(\dot{Y}(t), u(t)) + \lambda^T AY + \lambda^T Bu$. The pseudo-Hamiltonian formulation of the optimal control problem and the Pontryagin Minimum principle provide us with the following optimization problem [18, §1], [2, §2],[17, §4]:

$P$: “Minimize $H$ with respect to the control variable $u \in U_m$ under the constraints:

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

$$\lambda(t)^T = -\frac{\partial H}{\partial Y}(Y(t), u(t), \lambda(t))$$

and $H(Y^*(t), \lambda^*(t), u^*(t)) = 0$ along the optimal trajectory.”

Our algorithm is divided in two main steps: first, the controllable set is partitioned (see 4.1) in domains, inside which the optimal control is constant. In practice, we propose symbolic algorithms computing the boundaries of these cells (see §4.2). The second step requires to compute an optimal trajectory from an entry point to the target within each cell. In this section, the cost function $l$ is assumed linear in the control: $l(Y, u) = l_0(Y) + l_1(Y)u$. The case where this function is nonlinear is actually simple: indeed, the Hamiltonian optimization problem could be solved by classical tools, since $H_u \equiv \frac{\partial H}{\partial u} = 0$ is then solvable in the control variable $u$.

4.1 **Singular control**

Let us consider the optimization problem $P$. By definition, $P$ is a linear program. It thus admits solutions which may occur on the boundary of the polyhedral set $U_m$. Now, any solution $(Y, u, \lambda)$ of the Hamiltonian system (5) is said to be extremal and distinguish regular and singular solutions:

**Definition 1.** An extremal $(Y(t), u(t), \lambda(t))$ is called **regular** on an interval $[t_0, t_1]$, if there exists $k$ s.t., for almost all $t \in [t_0, t_1]$,

$$l_1(Y(t)) + \lambda^T(t)B(s_i) < \min\{l_1(Y(t)) + \lambda^T(t)B; i \neq k\}$$

Therefore, for any regular extremal $(Y(t), u(t), \lambda(t))$, the optimal control is given by the relation:

$$u(t) = s_i \quad \text{if} \quad l_1(Y(t)) + \lambda^T(t)B(s_i) < \min\{l_1(Y(t)) + \lambda^T(t)B(s_j)\}$$

Consequently one can define a partition of the controllable set (see definition 2) as follows:

**Definition 2.** An optimal trajectory $Y(.)$ belongs to the domain $\Gamma_i$ on a time interval $[t_0, t_1]$ if the condition: $\forall t \in [t_0, t_1], \forall j \in \{1, ..., m\} - \{i\}$, $|l_1(Y(t)) + \lambda^T(t)B(s_i - s_j)| < 0$ holds. Thus at any point of the domain $\Gamma_i$, the optimal control is $u(.) = s_i$ and the related field vector is $AY + BS_i$.

Now, we introduce the switching function $S_{i,j}$, that describes transitions between the domains $\Gamma_i$ and $\Gamma_j$:

**Definition 3** (Switching function).

$$S_{i,j}(t) = H_u(Y(t), u(t), \lambda(t))(s_i - s_j)$$

Then, the single zeros of $S_{i,j}$ give us the switching time between the domains $\Gamma_i$ and $\Gamma_j$. However it may also be possible to find time intervals where the switching function is identically equal to zero. This typically corresponds to the appearance of singular arcs in each face of the polyhedral control set. Thus singular trajectories are:

**Definition 4.** [22] A trajectory $Y(.)$ is called $ij$-singular on a time interval $[t_0, t_1]$ if the condition “$S_{i,j}(t) = 0$ and $\forall k \neq i, j$, $S_{j,k}(t) < 0$” holds for almost all $t \in [t_0, t_1]$. 


Just note that definition could be naturally extended to I-singular trajectories \((I \subset \{1, \ldots, p\})\). According to definitions 2 and 4, we show that I-singular trajectories geometrically correspond to the boundary between \(\Gamma_i\) and \(\Gamma_j\).

On this singular boundary, the optimal control is said to be singular and satisfies:

**Proposition 4.** Let us consider an i-j singular trajectory \(Y(\cdot)\) on a time interval \([t_0, t_1]\). Then:

\[
\forall t \in [t_0, t_1], \quad u(t) \in [s_i, s_j].
\]

Likewise, on an I-singular trajectory, \(u(t) \in \text{Conv}_{k \in I}(s_k)\).

### 4.2 Boundaries computation

At this point of our analysis, we have partitioned our state space in domains delimited by:

- singular boundaries (see e.g. [17, fully optimal problem]),
- mixed and non singular boundaries (see [11, ex. 1]),
- non singular boundaries (see [18, time-optimal problems]).

In our linear control problem, the Hamiltonian has the form:

\[
H(Y, v, \lambda) = H_0(Y, \lambda) + H_1(Y, \lambda)(s_i + (s_i - s_j)v)
\]

where \(v \in [0, 1]\) (since \(u \in [s_i, s_j]\) with proposition 4) to show how to symbolically compute the considered boundary, when it exists.

#### 4.2.1 Switch rules

In this paragraph we briefly describe a method to compute the allowable “switching directions” [11] in the state space. From examination of the sign of \(dH_u(Y(t), v, \lambda(t))\) at switching points (i.e. \(H_u(Y(t), \lambda(t)) = 0\) and \(H(Y(t), v, \lambda(t)) = 0\)), it is possible to determine whether switchings from \(u = s_i\) to \(u = s_j\) are allowed in a given region of the state space.

#### 4.2.2 Singular boundaries

In this paragraph we present a symbolic algorithm computing singular boundaries when they exist. This algorithm is essentially based on the Pontryagin maximum principle [18] and classical results in the theory of singular extrema (see [14, 19, 2] for more details).

We show on table 2 some performances of this algorithm in high dimension where \(U_m\) is a random simplex in \(\mathbb{R}^m\) and \(n = m\). Note that we still have to check that the

<table>
<thead>
<tr>
<th>n</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu (s)</td>
<td>0.10</td>
<td>0.22</td>
<td>0.35</td>
<td>0.56</td>
<td>0.91</td>
<td>1.51</td>
<td>2.43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu (s)</td>
<td>4.21</td>
<td>7.03</td>
<td>10.53</td>
<td>19.06</td>
<td>31.38</td>
<td>53.85</td>
<td>94.18</td>
</tr>
</tbody>
</table>

Table 2: Symbolic singular boundaries timings

so-computed boundary really exist in the controllable domain and that the switching conditions are satisfied: \(\forall k \notin \{i, j\}, S_{i,k} < 0\). However, we show next that these conditions are not always sufficient to determine if a computed singular boundary is valid or not. Such cases appear when the computed singular control explicitly depends on the state \(Y\).

#### Algorithm 5 i-j singular boundary

**Require:** \(i\) and \(j\), indices of the considered \(\Gamma\) domains.

**Require:** \(H(Y, v, \lambda)\).

**Ensure:** \(\varphi\), where \(\varphi(X) = 0\) defines the i-j boundary.

**Ensure:** \(u^*\) the i-j singular optimal control.

**Ensure:** \(\lambda^*\) the optimal Pontryagin parameter.

1: Compute the smallest integer \(K\) such that:

\[
\frac{\partial}{\partial v} \left( \frac{d^2K}{d^2K} H_v \right) \neq 0
\]

where \(H_v = \frac{\partial H}{\partial v}\).

2: if \(\text{the Legendre-Clebsch (LC) condition} [14, 19]:\)

\[
(-1)^K \frac{\partial}{\partial v} \left( \frac{d^2K}{d^2K} H_v \right) \geq 0 \text{ is not satisfied} \text{ then}
\]

3: Return “no singular solution”.

4: end if

5: Solve \((S)\) \(\{H = 0, H_v = 0, \left( \frac{d}{dt} H_v = 0 \right)_{t = 1.2K} \}\) \{(S)\} is linear in \(v\) and \(\lambda\), hence we obtain the exact singular values of \(v\) and \(\lambda\) in relation with \(Y\). The remaining relation gives the equation \((\varphi(Y) = 0)\) of the boundary.

6: Return \((\varphi(Y), s_j + (s_i - s_j)v(Y), \lambda(Y))\).

While the related boundary is bounded, the whole boundary between \(\Gamma_i\) and \(\Gamma_j\) is necessarily also made of a regular part. The next paragraph is devoted to its computation.

#### 4.2.3 Mixed boundaries

In this paragraph we assume that we have already computed the singular boundary between two domains \(\Gamma_i\) and \(\Gamma_j\) and check the existence condition of these boundary. So we have its equation: \(\varphi(Y) = 0\) under the constraint \(0 \leq e(Y) \leq 1\), the singular control \(u^*\) and the related \(\lambda^*\). We now want to compute the related regular boundary (see algorithm 6).

#### Algorithm 6 MixedBoundary

**Require:** \(i\) and \(j\), indices of the considered \(\Gamma\) domains.

**Require:** \(\varphi\) equation of the i-j boundary.

**Require:** \(\lambda^*\) optimal Pontryagin parameter on the i-j boundary.

**Ensure:** a parameterization of the non singular boundary between \(\Gamma_i\) and \(\Gamma_j\).

1: Parameterize the singular boundary (by the implicit functions theorem) \(\psi(\xi)\) (i.e. such that \(\varphi(\psi(\xi)) = 0\)).

2: for \(s \in \{s_i, s_j\}\) do

3: Compute the trajectory \(Y[\psi(\xi), s]\) from \(\psi(\xi)\) by time reversal with \(u = s:\)

\[
Y[\psi(\xi), s](t) = e^{At} \psi(\xi) + e^{At} \int_0^t e^{-Aw} Bsdw
\]

4: Solve the Euler-Lagrange equations (6) with the initial condition \(\lambda(\psi(\xi), s)[0] = \lambda^*(\psi(\xi))\).

5: Compute the first time \(t(\xi) < 0\) for which the switching condition between \(s_i\) and \(s_j\) holds, i.e.: \(S_{i,j}(t) = 0\) (see definition 3). No solution \(t(\xi)\) invalidates the singular boundary so that the boundary between \(\Gamma_i\) and \(\Gamma_j\) is necessarily regular.

6: end for

7: Return the switching curve equations(if they exist):

\[
Y[\psi(\xi), s](t(\xi)) = 0, \quad s \in \{s_i, s_j\}
\]

Consider the system [11, Example 1]:

\[
\begin{align*}
X(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 1 \\ -1 \end{bmatrix} u(t) \\
X(0) &= X_0
\end{align*}
\]
where $X \in \mathbb{R}^2$, $|u| < 1$ and the performance index to be minimized is: $J(X,u(.)) = \int_0^T \tfrac{1}{2} x_1(t)^2 dt$. (Note that (7) already is under its canonical form since $rk([B;AB]) = 2$).

[11] provides the full analysis of the singular controls and the graph of the allowable switching regions [11, fig. 5]. Therefore, the singular boundary is defined by $\phi(x_1, x_2) = x_1 + 2x_2 = 0$ and $-1 \leq x_2 \leq 1$ and the switching function is $S(t) = 2(\lambda_1(t) - \lambda_2(t))$. Due to the constraint $-1 \leq x_2 \leq 1$, the singular boundary does not allow us to draw a partition of the state space. However, we are able to compute all singular boundaries, we can now apply our MixedBoundary algorithm as shown on figure 2:

1. Parametrize the singular boundary: $\psi(\xi) = (-2\xi, \xi)$. 
2. For $u = 1$, compute the trajectory from $\psi(\xi)$ according to $u = 1$ by time reversal: $Y[\psi(\xi), 1](t) = [-\frac{4}{3}t^2 + \xi t + t - 2\xi, -t + \xi]$. 
3. Solve the Euler-Lagrange equations $S(0) = 0$ 
4. Compute the first time $t(\xi)$ such that $S(t(\xi)) = 0$ holds: $t(\xi) = \frac{2}{3} - 2\sqrt{\xi^2 - 3\xi + 3}$ 
5. Repeat steps 3, 4 and 5 with $u = -1$. 
6. Return the parametrized switching curve equation:

$$ \begin{cases} (2\xi \sqrt{\xi^2 - 3\xi + 3} - 2\xi^2 + 6\xi - 6 - 2\sqrt{\xi^2 - 3\xi + 3}, \xi \in [-1, 1]) \\ (2\xi \sqrt{\xi^2 + 3\xi + 3} + 2\xi^2 + 6\xi + 6 + 2\sqrt{\xi^2 + 3\xi + 3}, -\xi - 2\sqrt{\xi^2 + 3\xi + 3}, \xi \in [-1, 1]) \end{cases} $$

Figure 2: MixedBoundary in Gibson’s problem

Remark 1. In some cases the equation $S_{i,j}(t) = 0$ is not solved by Maple, so that we cannot express the time $t(\xi)$ in step 5. Nevertheless, by discretizing the given singular boundary, we can compute a discretization of the regular part of the searched boundary. Then, by interpolation, we find again the whole boundary.

4.2.4 Non Singular Boundaries

In this paragraph, we consider the case where there is no singular or mixed boundary between the two domains $\Gamma_i$ and $\Gamma_j$. The optimal control is then called bang-bang (i.e. piecewise constant with values in $\{s_i, s_j\}$). Let us distinguish two possible configurations:

- For all $(i, j) \in \{1, \ldots, m\}^2$, $i \neq j$, the boundary between $\Gamma_i$ and $\Gamma_j$ is non singular.

In this case, one method is to compute all the switching functions $S_{i,j}$ (see definition 3). After that we can study the zeros of $S_{i,j}$ and deduce the transition time $\tau$ between two of them. By time reversal, we start from the origin and build the switching curve. This method is well described in [17].

- There exists $k \in \{1, \ldots, m\} - \{i, j\}$, such that the boundary between $\Gamma_i$ and $\Gamma_k$ exists and is singular. In this case we come down to the same technique than for mixed boundaries: the idea is to take a parameterization of the singular boundary: the idea is to take a parameterization of the singular boundary between $\Gamma_i$ and $\Gamma_k$. We consider by time reversal the trajectory from a point of this boundary according to $u = s \in \{s_i, s_k\}$ and compute the first negative time for which the switching condition: $S_{i,k}(t) = 0$ holds.

This latter algorithm, NonSingBoundary is based on the following proposition [20]:

**Proposition 5.** Let $Y(.)$ be an optimal trajectory from an initial point $Y_0$ and $u(.)$ the associated optimal control. We assume that there exists a time $T > 0$, such that: $\exists e > 0$, $Y(t)$ regular over $[T - e, T]$ and $Y(T + .)$ is $ij$-singular. Then: $\forall t \in [T - e, T]$, $u(t) \in \{s_i, s_j\}$.

In conclusion, we have proposed three algorithms to compute boundaries between the domains $\Gamma_i$. We can now define a general one Boundary that compute the boundary between two given domains $\Gamma_i$ and $\Gamma_j$: $(\varphi, \omega, u) := Boundary(i, j)$ where $\varphi$ is the equation of the boundary, $\omega$ is defined by: $\omega(Y) = \begin{cases} 1 \text{ if } \varphi(Y) = 0 \text{ is singular} \\ 0 \text{ otherwise} \end{cases}$ and $u^*$ is the related optimal control. We therefore have all the necessary subroutines to solve our problem.

4.3 Linear Optimal Control

In this section, we detail the general algorithm for solving any linear optimal control problem. The principle is as follows: after a virtual partition of the state space in $\Gamma_i$ domains, one follows the trajectory within each cell of the partition. Indeed, in each cell and in every boundaries, the control is known thanks to the algorithms of section 4.2. When the trajectory reaches a boundary, there is a switching of cell and a control change. This goes on till the target is reached.

Once the canonical problem is solved with algorithm 7, we just have to apply the inverse transformation 3 to come down to optimal solutions of our initial control problem.

5. Conclusion

We have presented an algorithm for solving general linear optimal control problems. First we propose an explicit method to transform any problem into a canonical one by the way of a block Kalman decomposition. We have also developed generic algorithms solving the canonical problem even when complex boundaries occur. Yet, two important new features of our algorithms are that we give a full generic implementation and that they are mainly symbolic. Also, all the algorithms presented here have been implemented in Maple and work in high dimensions2.

2The maplets are available online at: www-lmc.imag.fr/lmc-mosaic/Jean-Guillaume.Dumas/SHOC
Algorithm 7 LinearOptimalControl

Require: A, B, Y₀, l(Y,u) = l₀(Y) + l₁(Y)u and {s₁,...,sₘ}.
Ensure: Optimal trajectory, control and value function.
1: V := 0; (Initialize switching functions)
2: Sᵢ,j = ∂H(Y,sᵢ)+(sᵢ−sⱼ)υ, λ) = lᵢ(Y) + λᵀB(sᵢ−sⱼ).
{Virtual partition of the state space}
3: I = [i ∈ [1,m] | l(Y, u, λ) ≠ i, Sᵢ,j < 0 ≠ ∅],
4: for all i ∈ I and j ∈ I such that i ≠ j do
5: (ϕᵢ,j,ωᵢ,j,υᵢ,j) := Boundary(i,j).
6: (Γᵢ)ᵢ∈I is the induced partition of the controllable set.
7: end for
8: {Identification of the domains where u = sᵢ}
9: for all j ∈ I do
10: if ∂Dᵢ := ∪ₖ(Y st. ϕᵢ,k(Y) = 0) then
11: end if
12: end for
13: k := 0; T₀ := 0;
{Within each cell, reach the boundary}
14: while Y₀ ≠ 0 do
15: Find s.t. Y₀ ∈ Γₜ.
16: if Y₀ ∈ ∂Γₜ, then
17: Find s.t. ϕᵢ,j(Y₀) = 0.
18: u := uᵢ,j(Y₀);
19: if ωᵢ,j(Y₀) = 1 then
20: Tₙ₊₁ := Solution of Y[Yₙ,u](t) = 0;
21: break while loop;
22: end if
23: else
24: u := sᵢ;
25: end if
26: Compute Tₙ₊₁ = inf{t > 0; Y[Yₙ,u](t) ∈ ∂Γₜ}
27: Yₙ₊₁ := Y[Yₙ,u(Tₙ₊₁)],
28: u := u for t ∈ [Tₙ,Tₙ₊₁];
29: Y := Y[Yₙ,u] for t ∈ [Tₙ,Tₙ₊₁];
30: V := V + ∫₀ᵀ₀−Tₙ l(Y[Yₙ,u](t))dt
31: end while
32: Return (Y,u*,V)

Further developments already are in progress:
• Complete the whole algorithm for a cost function nonlinear in the control. In this case, the Hamiltonian optimization problem could be solved by classical tools. Indeed, ∂V/∂u = 0 can now be solved in the control variable u.
• The UnderApproximation and solving algorithms have been performed on linear dynamical systems under the canonical form where A₂ = 0 (see §3.2 and section 4). These two algorithms have to be extended to any canonical form (see (2)). In practice, this corresponds to the appearance of a perturbation time function t → A₂eᵣ,t₁,Y₁(0) in the dynamical system. The technique does not change, but practical implementations are slightly more complex.
• The UnderApproximation could be refined and a study of the approximation error is still to be made. The idea is to consider cases where the dynamical system for u = sᵢ admits one (or an infinite number of) equilibrium point Pᵢ, (note that 0 is an equilibrium point when u = 0). The underapproximation can e.g. be completed by the convex hull of trajectories from Pᵢ that go through (or tend towards) Pᵢ by time reversal according to u = sᵢ. Also, a rigorous proof of the convergence of our under-approximation towards the real controllable set has still to be completed.

6. ACKNOWLEDGMENTS

We would like to thank Kevin Hamon for his collaboration and for the work done towards generic algorithms (see [9]).

7. REFERENCES