
Jorge Manuel Estrela da Silva

Abstract—The motivations for research on computational methods for the solution of dynamic optimization problems in the hybrid systems framework is briefly described, taking into account the background for dynamical systems with continuous dynamics. The most recent results on computational methods for this kind of problems are described.

Index Terms—Hybrid Systems, Dynamic Optimization, Computational Methods

I. INTRODUCTION

Hybrid dynamical systems, or simply Hybrid Systems (HS) for short, are a major topic of research in the control area nowadays. The interest on this topic grew significantly on the last decade of the 20th century (see for instance [1]–[3]) due to the enormous growth of computer based control systems and embedded systems in general. This kind of applications allowed, on some cases, or demanded, on other cases, new control designs and methodologies. This was motivated by the flexibility and discrete nature of the computer systems as opposed to classic analog control system. However, the employment of the designation Hybrid System in control literature can be traced at least as far back as 1966 [4]. Additionally, there are other works that, in light of the current definition of HS, can be seen as studies on HS, even though they do not employ that designation. One major example is the literature on "switched systems", which is now accepted as a special case of hybrid systems (see for instance [5]). Other such case is the work related in [6], where the authors opt to recast the original theory in the current framework of hybrid systems [7].

As motivated by classical mechanics and theory of differential equations, continuous-time dynamical systems can be modelled as follows:

\[ \dot{x}(t) = f(t, x(t), u(t)) \]  

where \( x(t) = (x_1(t), \ldots, x_n(t)) \in \mathbb{R}^n \) is the state, \( u \in \mathbb{R}^m \) is the vector of system inputs, and \( f(t, x, u) \) is the vector field determining the flow of the dynamical system. However, this formulation is still very general if no assumptions are made regarding \( f(t, x, u) \). For most interesting problems, the function \( f(t, x, u) \) is nonlinear. However, a common requirement for some control methodologies is that \( f(t, x, u) \) should have continuous derivatives on \( x \) and \( u \) (which also implies continuity of \( f(t, x, u) \)). Therefore, systems containing “hard” nonlinearities, such as saturation, hysteresis, etc., can not be treated on such framework.

The main idea of a hybrid system formulation is to model a complex system as a composition of multiple simpler systems, seeking a improved combination of simplicity and expressiveness. Vector fields with “abrupt” changes on their characteristics should be modelled as a set of simpler sub-systems, modeled as in 1. The hybrid system will then jump between different vector fields \( f_q(t, x, u), q \in 0, \ldots, N \), according to a combination of the following: value of the state variables; sequence of previous jumps; value of discrete inputs or occurrence of events. Additionally, jumps (or “resets”) on the value of the continuous state variables \( x_i(t) \) are also encompassed on HS. This allows for discontinuities on the trajectory of the system. The reset is made by simply assigning a value to the desired state variable, as opposed to the cumbersome methods that are required in a classical continuous-time formulation.

The work in [2] and [8] can be considered a landmark on the theory of HS. This work subsumes most of the previous formulations of HS and provides a general formulation and taxonomy that has been widely used as a reference since then.

Dynamic optimization seeks the solution of problems consisting of the minimization, or maximization, of a certain criteria which is a function of the dynamical system’s trajectory, and respective input signal, during a certain time interval. Dynamic optimization of continuous-time systems, even with \( f(x, u) \) being continuously differentiable, is a difficult problem. For general problems, numerical methods, based on partial or full discretization of the continuous-time problem, must be employed [9]. Even after decades, the research of more efficient numerical methods for the computation of the solution of this kind of problem continues [10], [11]. Therefore, it is no surprise that the computation of the solution for a dynamic optimization problem involving an hybrid system is still more complex. Consider the classic continuous-time optimal control problem where one seeks the input signal that optimizes the defined criteria, while meeting all the constraints. On hybrid systems, besides the continuous-time input, one may have to choose at which discrete instants certain events occur. This lends a combinatorial nature to the problem. Consider, a control problem where it is possible to change the dynamics of the system at a certain cost (see motivational problem in [12]), i.e., a hybrid system with controlled transitions. A “brute force” approach would have to consider the change of dynamics at every instant of the considered time horizon, and then calculate the optimal solution for each case, using techniques from dynamic optimization of continuous systems. Additionally, hybrid systems formulations aim to provide a general framework for problems that would be dealt in a cumbersome way by classical methods. One such example is the consideration of state constraints, which are modeled in an elegant way on hybrid systems formulations.
II. BACKGROUND: DYNAMIC OPTIMIZATION METHODS FOR CONTINUOUS-TIME SYSTEMS

Dynamic optimization has a long story [13]. This section presents a brief overview of the two main milestones in modern dynamic optimization: the Pontryagin’s maximum principle [14], and dynamic programming [15], based on the solution of the Hamilton-Jacobi-Bellman partial differential equation.

A. Problem statement

Dynamic optimization for problems with fixed finite time horizon can be stated as follows:

\[ \min \psi(x(T)) + \int_0^T L(t, x(t), u(t))dt \]  \hspace{1cm} (2)

subject to:

\[ \dot{x}(t) = f(t, x(t), u(t)) \] \hspace{1cm} (3)
\[ u(t) \in U(t) \] \hspace{1cm} (4)

Equation 2 is the objective function, where \( L(t, x, u) \) is the running cost. Equation 3 arises naturally from the problem dynamics. There are some variations to this formulation. Obviously any maximization problem can be turned into a minimization by noting that \( \max \psi(x) = \min -\psi(x) \). The final time \( T \) can be normalized to unity, by appropriate time scaling. The objective function 2 can be simply of the form \( \min \psi(x(T)) \) by defining a new state variable \( y(t) \) with flow equation \( \dot{y}(t) = L(t, x(t), u(t)) \). However, as we will see, for numerical computations we usually want to keep the number of state variables as low as possible. In the minimum time problem (or optimal time problem), where \( T \) itself is a decision variable, the problem can be changed to the fixed finite time formulation by considering a new time variable \( s \), a new state variable with flow equation \( \frac{dx}{ds}(s) = \alpha \), where \( \alpha \) can be regarded as a new input variable, and \( \frac{d^2x}{ds^2}(s) = \alpha f(x, u) \), with \( z(s) = x(t(s)) \). Finally, final endpoint constraints (terminal conditions) can be considered, as also state constraints (\( x(t) \) must lie in some set \( S \) for all time horizon).

B. Maximum principle

The Maximum Principle, presented in [14] is one of the main tools in optimal control problems. It provides the following necessary condition for the solution of 2:

\[ H(t, x^*(t), u^*(t), p^*(t)) \leq H(t, x^*(t), u(t), p^*(t)), \forall u(t) \in U(t) \] \hspace{1cm} (5)

where \( x^*(\cdot) \) is the optimal solution to 2, \( u^*(\cdot) \) the optimal input, \( p^*(\cdot) \) the optimal trajectory for the adjoint variables, and \( H(t, x, u, p) \) the Hamiltonian function defined as follows:

\[ H(t, x(t), u(t), p(t)) = \min_{u(t) \in U(t)} p^T f(t, x(t), u(t)) + L(t, x(t), u(t)) \] \hspace{1cm} (6)

This condition is presented as Maximum Principle or Minimum Principle depending whether we are considering a maximization or a minimization problem, respectively.

C. Principle of optimality

The principle of optimality defines the basic idea behind dynamic programming, the other main approach to dynamic optimization problems. It was originally stated as follows [15, p. 83]: “An optimal policy has the property that, whatever the initial state and optimal first decision may be, the remaining decisions constitute an optimal policy with regard to the state resulting from the first decision”.

The mathematical formulation is based on the concept of value function. The value function \( V(t, x) \) gives us the cost\(^1\) to go from a state \( x \) at time \( t \) to the optimal solution. The principle of optimality, for deterministic continuous-time system, can be defined as follows:

\[ V(t, x) = \min \left\{ \int_{t}^{t+\Delta} L(\tau, x(\tau), u(\tau))d\tau + V(t+\Delta, x(t+\Delta))|u(\tau) \in U(\tau), \forall \tau \in [t, t+\Delta] \right\} \] \hspace{1cm} (7)

with \( V(T, x) = \psi(x) \) \hspace{1cm} (8)

In terms of computation, this principle is more easily understood if regarded backwards in time, starting from the terminal condition 8. This mathematical formulation leads to the Hamilton-Jacobi-Bellman equation, presented in the next section.

D. Hamilton-Jacobi-Bellman equation

The Hamilton-Jacobi-Bellman (HJB) equation is a partial differential equation (PDE) of the form

\[ \phi_t(x, t) + H(t, x, u, \phi_x(x, t)) = 0 \] \hspace{1cm} (9)

subject to terminal condition

\[ \phi(x, T) = \psi(x) \] \hspace{1cm} (10)

where \( \psi(x) \) is the same as in 2. In the context of dynamic programming, \( \phi(x, t) \) is the value function. It must be remarked that the value function allows us to find the optimal trajectory from any initial state, by iterative static minimization of the Hamiltonian, and integration of the dynamic equation using the optimal control \( u^*(t) \). Therefore, The HJB equation defines a sufficient condition for the optimal solution. The main difficulty resides in solving the equation in order to find the value function. One of the main obstacles to the process of finding that solution is the fact that in the general case \( \phi_x(x, t) \) is not differentiable. On those cases, 9 has not a solution in classical terms, and an alternative notion of solution must be considered. One such notion is the viscosity solution [16].

E. Numerical methods to solve the HJB

One approach to solve the HJB PDE is to discretize the equation, and to solve the resulting system of coupled nonlinear equation using iterative numerical methods. However

\(^1\) The term “cost” is associated to minimization problems. On his seminal document, Bellman considers a maximization problem and employs the term “return”.
this can be rather expensive. In practice, for dynamic programming inspired algorithms, we confine the computation to a bounded partition of the state space, and define a grid that will discretize the state space over that partition. Notice that for a state space of dimension \( n \) with a regular grid of \( k \) points per dimension, the memory requirements are on the order of \( k^n (O(k^n)) \).

In [10], the authors propose an algorithm for the minimum time optimal trajectory problem\(^2\) with a computational complexity of \( O(M \log(M)) \) where \( M \) is the number of points in the mesh that discretizes the state space. The algorithm computes the solution of a special version of the HJB PDE using a technique coined “ordered upward methods”, where information about the characteristic directions of the PDE is computed as the solution is constructed. That information is used to decide the sequence of points used in the computation, hence the name of the technique. The algorithm is loosely inspired in the Dijkstra’s heap-sort algorithm for computation of the shortest path on a network. It should be remarked that, when considering discrete versions of problems on the continuous domain, Dijkstra’s algorithm does not converge to the continuous solution as the mesh is refined.

The system dynamics are described on the following form:

\[
\dot{x}(t) = g(x(t), a(t))a(t), \|a(t)\| = 1
\]

where \( g: \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^+ \) gives the speed of propagation in the direction defined by the input \( a(t) \) at the state \( x(t) \). Basically, \( g(x(t), a(t))a(t) \) gives the maximum flow of 3 at \( x(t) \) along a chosen direction \( a(t) \). The ratio between the maximum and minimum values of \( g(x, a) \) is designated coefficient of anisotropy, \( \gamma \). The coefficient of anisotropy must be bounded and the computational complexity of the algorithm grows with \( n \). For isotropic problems we have \( \gamma = 1 \). For instance, the system

\[
\begin{align*}
\dot{x}_1(t) &= \cos(u(t)) \\
\dot{x}_2(t) &= \sin(u(t))
\end{align*}
\]

with \( u(t) \) as a scalar input, presents an isotropic problem. On the other hand, the system

\[
\begin{align*}
\dot{x}_1(t) &= u_1(t) \cos(x_3(t)) \\
\dot{x}_2(t) &= u_1(t) \sin(x_3(t)) \\
\dot{x}_3(t) &= u_2(t) \\
|u_1(t)| &\leq U_1, |u_2(t)| \leq U_2
\end{align*}
\]

can not be treated by this algorithm since, depending on \( x \), \( g(x, a) \) is zero for some directions \( a \).

The proposed algorithm is based on the solution of the HJB equation. However, the considered value function \( V(x) \) does not depend on time. It indicates the minimum time to reach a given target set \( \partial \Omega \) starting from \( x \). Therefore the following static HJB equation form is considered:

\[
\begin{align*}
\min_a (V_x(x) \cdot a)g(x, a) &= 1, \quad x \in \Omega \\
V(x) &= \psi(x), \quad x \in \partial \Omega
\end{align*}
\]

where \( \partial \Omega \) is the boundary of \( \Omega \). If the problem is isotropic, 12 simplifies to the so called Eikonal equation. For isotropic problems, other “single-pass” algorithms inspired on Dijkstra’s algorithm were proposed previously [17], [18]. The algorithm can also be applied for time-varying systems \( \dot{x}(t) = g(t, x(t), a(t))a(t) \). On those cases, the following form of the HJB PDE is considered:

\[
\begin{align*}
\min_a (V_x(x) \cdot (-a))g(V(x), x, a) &= 1, \quad x \in \Omega \\
V(x) &= \psi(x), \quad x \in \partial \Omega
\end{align*}
\]

As said above, when considering the solution of 12, \( V(A) \) is the minimum time to reach the target set \( \partial \Omega \) when starting from \( A \). However, if \( V(x) \) is the solution of 14 then \( V(A) \) should be interpreted as the minimum time to reach \( A \) when starting from \( \partial \Omega \). This work is developed in more detail in [19] in order to consider both cases (solution from and to boundary), as also the general HJB PDE 9.

In [20], the authors propose a toolbox for the numerical solution of 9. Several variations of the HJB PDE are supported and several options are provided. Those options allow a trade-off between the accuracy of the solution and the computation time. The algorithm can deal with the general nonlinear continuous system. However, the algorithm requires \( \phi(x, t) \) to be continuous. Therefore, in general, the solution of 9 given by this algorithm is not the value function. The value function must be inferred from the zero level sets of \( \phi(x, t) \), i.e., \( \{x|\phi(x, t) = 0\} \). Because of this \( \phi(x, t) \) is designated by implicit surface function. The algorithm starts by setting \( \phi(x, t) = 0 \) at the target set, and the remaining points as a continuous increasing function of \( x \). At each time step, the zero level set is propagated using upward methods. The user is required to introduce the analytical expression resulting from the static optimization of the Hamiltonian. An implementation of this toolbox for Matlab® (a product from The MathWorks, Inc.) is publicly available.

### III. DYNAMIC OPTIMIZATION OF HYBRID SYSTEMS

In [12], the authors extend the results of [21] (and later in [10], as described in the last section) to general hybrid systems. The main limitation of this algorithm is the class of supported dynamics, which must be of the form 11, with bounded coefficient of anisotropy. This requirement precludes its application to many practical systems. The algorithm is illustrated only for systems with two-dimensional continuous space. No implementation of the algorithm is provided.

In [20], the authors claim that the toolbox can be applied to hybrid systems. However, that is to be done in an ad hoc fashion. In [22], the author presents an example of application for a system with three different modes of operation and autonomous transitions between them.

In [23], [24], the authors propose an algorithm based on the hybrid maximum principle defined in [25] and refined by the authors. The authors cite references to several optimal control problems for hybrid systems and apply the algorithm on several examples of these problems.

In [26], the authors present a numerical solver for hybrid systems without jumps on the continuous state variables. It
must be remarked that this is a major limitation for general hybrid systems, since it limits the algorithm for the class of switched systems. The main advantage of the proposed algorithm is that, under certain assumptions, it may use sequential quadratic programming, avoiding therefore combinatorial explosion. One (sufficient) condition is that the transitions are autonomous, i.e., the jumps between vector fields depend only on the continuous state. However, the algorithm may still avoid combinatorial complexity under weaker conditions, for special cases. The main idea of the algorithm is to consider that the transitions take place over a continuous set of modes instead of the discrete ones, therefore avoiding the combinatorial problem (the problem becomes a continuous one). The algorithm is applied on some examples, where the more complex one consists of motion optimization for an unicycle that may switch between “rolling” and “sliding” modes. The authors do not discuss potential ways of improving the applicability of this algorithm to general hybrid systems.

A last word for sampling-based methods [27], [28], based on random or probabilistic choice of trajectories. In practice, these methods provide a feasible solution without suffering from the time penalty and memory requirements of the above mentioned techniques (Maximum Principle, and Dynamic Programming). Those algorithms scale better with the increase of the number of state variables, while the above methods are prohibitive for systems of dimension greater than four, taking in account current computer systems. However, there is no guarantee that sampling-based methods converge to the optimal solution. Results on the convergence of those methods are not definitive and studies concerning optimality are underway [27]. Additionally, those methods are not well-suited for reach set computation since they rely only on the simulated trajectories.

IV. CONCLUSION

Most authors do not provide publicly available implementations of their algorithms. The results and properties reported on the literature are frequently relative to a particular setting. This makes comparison of different approaches difficult. It would be desirable to agree on a set of benchmark scenarios in order to facilitate comparisons. In [29], the authors propose a benchmark problem for this kind of algorithms.

The main conclusion is that there is still much work regarding a solver that deals efficiently, and in a straightforward way, with all classes of hybrid systems defined in [8]. Approaches inspired on dynamic programming lead to the solution of the HJB PDE. Computational representation of the solution of the PDE is a challenge by itself: for grid discretizations the storage requirements grow exponentially with the number of continuous state variables. Approaches like the one described in [30] for reachability problems may be a possible avenue for an efficient representation. Concerning the the discrete modes of the hybrid system, and the transitions between them, the impact is not so strong. The analyzed algorithms have shown, at most, a linear dependency on the maximum number of transitions departing from discrete modes, for the considered class of systems.

ACKNOWLEDGMENT

The author would like to thank Jo˜ ao Borges de Sousa and Fernando Lobo Pereira for their suggestions.

REFERENCES


