J.A. Stiver, P.J. Antsaklis and M.D. Lemmon, "An Invariant Based Approach to the Design of Hybrid Control Systems," Proc. of the IFAC 13th Triennial World Congress, Vol. J, pp. 467-472, San Francisco, CA, July 1-5, 1996.

AN INVARIANT BASED APPROACH TO THE DESIGN OF HYBRID CONTROL SYSTEMS[1]

James A. Stiver, Panos J. Antsaklis and Michael D. Lemmon

Department of Electrical Engineering,
University of Notre Dame,
Notre Dame, IN 46556

Abstract. The hybrid control systems considered here consist of a continuous-time plant under the control of a discrete event system. Communication between the plant and controller is provided by an interface which can convert signals from the continuous domain of the plant to the discrete, symbolic domain of the controller, and vise-versa. When designing a controller for a hybrid system, the designer may or may not be free to design the interface as well. This paper presents an approach to designing the controller and part of the interface. It is based on using the natural invariants of the system.

Keywords. Hybrid, design, discrete

1. INTRODUCTION

Hybrid systems contain two distinct types of systems, systems with continuous dynamics and systems with discrete event dynamics, that interact with each other. Hybrid control systems typically arise when continuous processes interact with, or are supervised by, sequential machines. Since the continuous and discrete dynamics coexist and interact with each other, it is important to develop models that accurately describe the dynamic behavior of such hybrid systems. In this way it is possible to develop control strategies that take fully into consideration the relation and interaction of the continuous and discrete parts of the system. In the past, models for the continuous and discrete event subsystems were developed separately; the control law was then derived in a rather empirical fashion, except in special cases such as the case of digital controllers for linear time-invariant systems. The study of hybrid control systems is essential in designing sequential supervisory controllers for continuous systems, and it is central in designing intelligent control systems with a high degree of autonomy (Antsaklis *et al.* 1993, Antsaklis 1994). Examples of hybrid control systems are common in practice and are found in such applications as flexible manufacturing, chemical process control, electric power distribution, and computer communication networks. A simple example of a hybrid control system is the heating and cooling system of a typical home. The furnace and air conditioner, along with the heat flow characteristics of the home, form a continuous system which is to be controlled. The thermostat is a simple discrete event system which basically handles the symbols *too hot, too cold,* and *normal.*

The hybrid control systems of interest here consist of a continuous (state, variable) system to be controlled, also called the plant, and a discrete event controller connected to the plant via an interface. In the approach described below, the plant contains all continuous subsys-

[1] The partial financial support of the Army Research Office (DAAH04-95-0600) and the National Science Foundation (MSS-9216559) is gratefully acknowledged.

tems of the hybrid control system, such as any conventional continuous controller that may have been developed, a clock if time and synchronous operations are to be modeled, etc. The controller is an event driven, asynchronous discrete event system (DES), described here by a finite state automaton. The hybrid control system also contains an interface that provides the means for communication between the continuous plant and the DES controller; see Figure 1. The interface receives information from the plant in the form of measurements of a continuous variable, such as the continuous state, and issues a sequence of symbols to the DES controller. It also receives a sequence of control symbols from the controller and issues (piecewise) continuous input commands to the plant.

In general the design of the interface depends not only on the plant to be controlled, but also on the control policies available, as well as on the control goals to be attained. Certain control goals may require, for example, detailed feedback information while for others coarser quantization levels of the signals may be sufficient. The former case corresponds to finer partitioning of the feedback signal space, while the latter corresponds to coarser partitioning. The fact that different control goals may require different types of information about the plant is not surprising, as it is rather well known that to stabilize a system, for example, requires less detailed information about the system's dynamic behavior than to do tracking.

The hybrid control system model presented in this paper is close to the model of (Nerode and Kohn 1993) which was also developed for control purposes (in particular control design; our model is for analysis and design). Other models include (Brockett 1994); see also the earlier references therein. The models in (Back *et al.* 1993, Tavernini 1987) are more general but they are developed primarily for simulation purposes. The model of (Peleties and DeCarlo 1994) is developed for control, but the interface is rather complex. Other approaches include (Benveniste and Guernic 1990, Deshpande and Varaiya 1994, Gennaro *et al.* 1994, Tittus and Egardt 1994, Holloway and Krogh 1992, Grossman and Larson 1992, Kohn and Nerode 1992, Zeigler 1989). The paper by Branicky et al. (Branicky *et al.* 1994) presents a rather detailed comparison of some of the models. Early work also included (Peleties and DeCarlo 1989, Gollu and Varaiya 1989). For recent developments in hybrid systems research see (Grossman *et al.* 1993). Note that early versions of the results discussed in this paper have appeared in (Stiver *et al.* 1994, Stiver *et al.* 1995).

2. HYBRID CONTROL SYSTEM MODELING

A hybrid control system, can be divided into three parts, the plant, interface, and controller as shown in Figure 1. In this model, the plant represents the continuous-time components of the system, while the controller represents the discrete-event portions. The interface is the necessary mechanism by which the former two communicate. A detailed description of the model can be found in (Stiver *et al.* 1994) as well as other earlier publications by the authors.

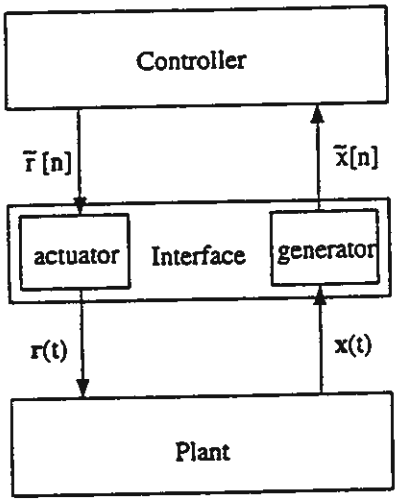

Fig. 1. Hybrid Control System

3. INVARIANT BASED APPROACH

Here a methodology is presented to design the controller and the interface together based on the natural invariants of a plant described by

$$\dot{x}(t) = f(x(t), r(t)) \qquad (1)$$

where certain smoothness assumptions apply.

In particular, this section discusses the design of the generator, which is part of the interface, and the design of the controller. We assume that the plant is given, the set of available control policies is given, and the control goals are specified as follows. Each control goal for the system is given as a starting set and a target set, each of which is an open subset of the plant state space. To realize the goal, the controller must be able to drive the plant state from anywhere in the starting set to somewhere in the target set using the available control policies. Generally, a system will have multiple control goals.

To successfully control the plant, the controller must know which control policy to apply and when to apply it. The controller receives all its information about the

plant from the generator, and therefore the generator must be designed to provide that information which the controller requires.

The following solution to this design problem is proposed. For a given target region, identify the states which can be driven to that region by the application of a single control policy. If the starting region is contained within this set of states, the control goal is achievable via a single control policy. If not, then this new set of states can be used as a target region and the process can be repeated. This will result in a set of states which can be driven to the original target region with no more than two control policies applied in sequence. This process can be repeated until the set of states, for which a sequence of control policies exists to drive them to the target region, includes the entire starting region (provided the set of control policies is adequate as mentioned below).

When the regions have been identified, the generator is designed to tell the controller, via plant symbols, which region the plant state is currently in. The controller will then call for the control policy which drives the states in that region to the target region.

3.1 *Generator Design*

To describe the regions mentioned above, the concept of the flow (Nijmeijer and van der Schaft 1990) is used. Let the flow for the plant be given by $F_k : \mathrm{X} \times \Re \to \mathrm{X}$, where

$$\mathrm{x}(t) = F_k(\mathrm{x}(0), t). \tag{2}$$

The flow represents the state of the plant after an elapsed time of t, with an initial state of $\mathrm{x}(0)$, and with a constant input of $\gamma(\bar{r}_k)$, representing the kth control policy. Since the plant is time invariant, there is no loss of generality when the initial state is defined at $t = 0$. The flow is defined over both positive and negative values of time. The flow can be extended over time using the forward flow function, $F_k^+ : \mathrm{X} \to \mathrm{P}(\mathrm{X})$, and the backward flow function, $F_k^- : \mathrm{X} \to \mathrm{P}(\mathrm{X})$, which are defined as follows.

$$F_k^+(\xi) = \bigcup_{t \geq 0} \{F_k(\xi, t)\} \tag{3}$$

$$F_k^-(\xi) = \bigcup_{t \leq 0} \{F_k(\xi, t)\} \tag{4}$$

The backward and forward flow functions can be defined on an arbitrary set of states in the following natural way.

$$F_k^+(\mathrm{A}) = \bigcup_{\xi \in \mathrm{A}} \{F_k^+(\xi)\} \tag{5}$$

$$F_k^-(\mathrm{A}) = \bigcup_{\xi \in \mathrm{A}} \{F_k^-(\xi)\} \tag{6}$$

where $\mathrm{A} \subset \mathrm{X}$. For a target region, T, $F_k^-(T)$ is the set of initial states from which the plant can be driven to T with the input $\gamma(\bar{r}_k)$. In addition, $F_k^+(T)$ is the set of states which can be reached with input $\gamma(\bar{r}_k)$ and an initial state in T.

Now a generator design procedure can be described using the backward flow function. This is a preliminary procedure, upon which the final design method, developed subsequently, is based. For a given starting region, $S \subset \mathrm{X}$, and target region, $T \subset \mathrm{X}$, use the following algorithm.

(1) If $S \subset T$, stop.
(2) Identify the regions, $F_k^-(T), \forall \bar{r}_k \in \bar{R}$.
(3) Let $T = \bigcup_{\bar{r}_k \in \bar{R}} F_k^-(T)$
(4) Go to 1.

There are two problems associated with this algorithm as stated. First, it will not stop if there is no sequence of available control policies which will achieve the control goal, and second, actually identifying the regions given by the flow functions is quite involved. The first issue is related to the adequacy of the available control policies and will not be dealt with here. The second problem will be addressed. The difficulty in identifying a region given by a flow function is integrating over all the points in the target region. In the generator design procedure developed here, the concentration is on finding a subset of the region $F_k^-(T)$, rather than the region itself. By definition, all the trajectories passing through $F_k^-(T)$ lead to the target region, T, and therefore all the trajectories found in a subset of $F_k^-(T)$ will also lead to the target.

Here, the focus is on identifying subsets of $F_k^-(T)$ which are called *common flow regions*. Common flow regions are bounded by invariant manifolds and an exit boundary. The invariant manifolds are used because the state trajectory can neither enter nor leave the common flow region through an invariant manifold. The exit boundary is chosen as the only boundary through which state trajectories leave the common flow region.

To design the generator, it is necessary to select the set of hypersurfaces, $\{h_i : \mathrm{X} \to \Re \mid i \in I\}$ and the associated functions, $\{\alpha_i : \mathcal{N}(h_i) \to \bar{R} \mid i \in I\}$. These hypersurfaces make up the invariant manifolds and exit boundaries mentioned above, as well as forming the boundary for the target region(s). As the system operates, when the state crosses the ith hypersurface the genera-

tor sends a plant symbol to the controller according to the function α_i.

A target region, T, is specified as

$$T = \{\xi \in \mathrm{X} : \forall i \in I_T, h_i(\xi) < 0\}, \quad (7)$$

where I_T is the index set indicating which hypersurfaces bound the target region. A common flow region, B, is specified as

$$B = \{\xi \in \mathrm{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\}, \quad (8)$$

where I_B is an index set indicating which hypersurfaces form the invariant manifolds bounding B and h_e defines the exit boundary for B.

The goal, of course, is that B should include only states whose trajectories lead to the target region. Figure 2 shows an example of this where $I_T = \{1\}$ and $I_B = \{2, 3\}$. The target region, T, is surrounded by h_1, the common flow region lies between h_2 and h_3 above the exit boundary, h_e.

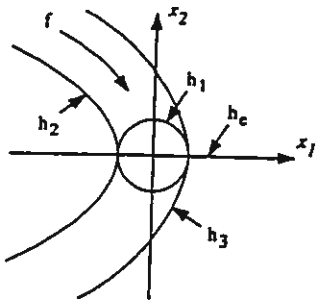

Fig. 2. Target Region and Invariants

Two propositions are now presented which can be used to determine the suitability of a set of hypersurfaces to achieve our goal of identifying a common flow region. In different situations, one of the propositions may be easier to apply than the other. The propositions give sufficient conditions for the hypersurfaces bounding B and T to ensure that all state trajectories in B will reach the target region.

Proposition 1. Given the following:

(1) A flow generated by a smooth vector field, f_k
(2) A target region, $T \subset \mathrm{X}$
(3) A set of smooth hypersurfaces, $h_i, i \in I_B \subset 2^I$
(4) A smooth hypersurface (exit boundary), h_e

such that $B = \{\xi \in \mathrm{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\} \neq \emptyset$. For all $\xi \in B$ there is a finite time, t, such that $F_k(\xi, t) \in T$, if the following conditions are satisfied:

(1) $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
(2) $\exists \epsilon > 0, \nabla_\xi h_e(\xi) \cdot f(\xi) < -\epsilon, \forall \xi \in B$
(3) $B \cap \mathcal{N}(h_e) \subset T$

The second proposition uses a slightly different way of specifying a common flow region. In addition to the invariant manifolds and the exit boundary, there is also a cap boundary. The cap boundary is used to obtain a common flow region which is bounded. So for this case

$$B = \{\xi \in \mathrm{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\}. \quad (9)$$

Proposition 2. Given the following:

(1) A flow generated by a smooth vector field, f_k
(2) A target region, $T \subset \mathrm{X}$
(3) A set of smooth hypersurfaces, $h_i, i \in I_B \subset 2^I$
(4) A smooth hypersurface (exit boundary), h_e
(5) A smooth hypersurface (cap boundary), h_c

such that $B = \{\xi \in \mathrm{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\} \neq \emptyset$ and $\overline{B}$ (closure of B) is compact. For all $\xi \in B$ there is a finite time, t, such that $F_k(\xi, t) \in T$, if the following conditions are satisfied:

(1) $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
(2) $\nabla_\xi h_c(\xi) \cdot f(\xi) < 0, \forall \xi \in B \cap \mathcal{N}(h_c)$
(3) $B \cap \mathcal{N}(h_e) \subset T$
(4) There are no limit sets in $\overline{B}$

Consider the hypersurfaces defined by $\{h_i : i \in I_B\}$. These hypersurfaces must first be invariant under the vector field of the given control policy, f. This can be achieved by choosing them to be integral manifolds of an $n-1$ dimensional distribution which is invariant under f. An $n-1$ dimensional distribution, $\Delta(\mathrm{x})$, is invariant under f if it satisfies

$$[f(\mathrm{x}), \Delta(\mathrm{x})] \subset \Delta(\mathrm{x}), \quad (10)$$

where the $[f(\mathrm{x}), \Delta(\mathrm{x})]$ indicates the Lie bracket. Of the invariant distributions, those that have integral manifolds as we require, are exactly those which are involutive (according to Frobenius). This means

$$\delta_1(\mathrm{x}), \delta_2(\mathrm{x}) \in \Delta(\mathrm{x}) \Rightarrow [\delta_1(\mathrm{x}), \delta_2(\mathrm{x})] \in \Delta(\mathrm{x}). \quad (11)$$

Therefore by identifying the involutive distributions which are invariant under the vector field, f, we have identified a set of candidate hypersurfaces. For details of these relationships between vector fields and invariant distributions, see (Isidori 1989).

Since an $n-1$ dimensional involutive distribution can be defined as the span of $n-1$ vector fields, over each

of which it will then be invariant, and the control policy only gives one vector field, f, there will be more than one family of hypersurfaces which are all invariant under f. The set of all invariant hypersurfaces can be found in terms of $n-1$ functionally independent mappings which form the basis for the desired set of functionals, $\{h_i : i \in I_B\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(x)} = \frac{dx_2}{f_2(x)} = \cdots = \frac{dx_n}{f_n(x)} \qquad (12)$$

where $f_i(x)$ is the ith element of $f(x)$.

3.2 *Controller Design*

Once the interface has been designed, the design of the controller involves two steps. The first step is to construct one subautomaton for each control goal. This is the step which is already determined by the interface design. The second step is the connection of these subautomata to create a single DES controller. This step will depend upon the order in which the simpler control goals are to be achieved. For example, if a chemical process is to produce a sequence of different products, then each subautomaton in the controller would be designed to produce one of the products, and these subautomata would be connected to produce the products in the desired sequence.

The hypersurfaces in the generator divide the state space of the plant into a number of *cells*. Two states are in the same cell exactly when they are both on the same side (positive or negative) with respect to each hypersurface. States which lie on a hypersurface are not in any cell.

The first step in creating the controller is the construction of the subautomata, one for each individual control goal. Each subautomaton is constructed in the following way.

i. Create a controller state to represent each cell.
ii. Place transitions between states which represent adjacent cells.
iii. Label each transition with the plant symbol which is generated by the hypersurface separating the associated cells.

This results in a subautomaton which can follow the progress of the plant state as it moves from cell to cell. Next the controller output function must be designed for each subautomaton.

The controller symbol output by a given controller state depends on which common flow region contains the associated cell. Each common flow region was constructed using a specific control policy, and the control symbol which initiates that control policy should be output by controller states representing cells contained in that common flow region. However, in general, common flow regions will overlap, meaning a given cell can lie in more than one common flow region. In such cases treat the cell as lying within the common flow region which is closest to the target region. Distance, in this case, is the number additional control policies which must be used to reach the target region. If common flow regions are both the same distance, then the choice is arbitrary, though the common flow region which is favored in one case must then be favored in all such cases. States which represent cells not contained in any common flow region or target region will never be visited and can thus be deleted.

Once the individual subautomata have been constructed they must be connected to form a single controller. This can be accomplished by following these steps for each subautomaton.

i. Remove the state(s) which represent cells in the target region as well all transitions emanating from such states.
ii. Connect the dangling transitions to states in the subautomaton which achieves the next desired control goal. The connections will be to the states which represent the same cells as the states which were removed.

In this way, as soon as one control goal is achieved, the system will begin working on the next one. The actual order in which each control goals are pursued is up to the designer.

4. REFERENCES

Antsaklis, P. J. (1994). Defining intelligent control. *IEEE Control Systems Magazine* 14(3), 4–5. Report of the Task Force on Intelligent Control, P. J. Antsaklis Chair.

Antsaklis, P. J., J. A. Stiver and M. D. Lemmon (1993). Hybrid system modeling and autonomous control systems. In: *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, Eds.). Vol. 736 of *Lecture Notes in Computer Science*. pp. 366–392. Springer-Verlag.

Back, A., J. Guckenheimer and M. Myers (1993). A dynamic simulation facility for hybrid systems. In: *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, Eds.). Vol. 736 of *Lecture Notes in Computer Science*. pp. 255–267. Springer-Verlag.

Benveniste, A. and P. Le Guernic (1990). Hybrid dynamical systems and the signal language. *IEEE Transactions on Automatic Control* 35(5), 535–546.

Branicky, M., V. Borkar and K. Mitter (1994). A unified framework for hybrid control. In: *Proceedings of the 33rd IEEE Conference on Decision and Control*. Lake Buena Vista, FL. pp. 4228–4234.

Brockett, R. (1994). Language driven hybrid systems. In: *Proceedings of the 33rd IEEE Conference on Decision and Control*. Lake Buena Vista, FL. pp. 4210–4214.

Deshpande, A. and P. Varaiya (1994). Viable control of hybrid systems. In the Ph.D. Dissertation of the first author. ftp: eclair.eecs.berkeley.edu.

Gennaro, S. Di, C. Horn, S. Kulkarni and P. Ramadge (1994). Reduction of timed hybrid systems. In: *Proceedings of the 33rd IEEE Conference on Decision and Control*. Lake Buena Vista, FL. pp. 4215–4220.

Gollu, A. and P. Varaiya (1989). Hybrid dynamical systems. In: *Proceedings of the 28th Conference on Decision and Control*. Tampa, FL. pp. 2708–2712.

Grossman, R. and R. Larson (1992). Viewing hybrid systems as products of control systems and automata. In: *Proceedings of the 31st Conference on Decision and Control*. Tucson, AZ. pp. 2953–2955.

Grossman, R. L., Nerode, A., Ravn, A. P. and Rischel, H., Eds.) (1993). *Hybrid Systems*. Vol. 736 of *Lecture Notes in Computer Science*. Springer-Verlag.

Holloway, L. and B. Krogh (1992). Properties of behavioral models for a class of hybrid dynamical systems. In: *Proceedings of the 31st Conference on Decision and Control*. Tucson, AZ. pp. 3752–3757.

Isidori, A. (1989). *Nonlinear Control Systems*. 2 ed.. Springer-Verlag. Berlin.

Kohn, W. and A. Nerode (1992). Multiple agent autonomous hybrid control systems. In: *Proceedings of the 31st Conference on Decision and Control*. Tucson, AZ. pp. 2956–2966.

Nerode, A. and W. Kohn (1993). Models for Hybrid Systems: Automata, Topologies, Controllability, Observability. In: *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, Eds.). pp. 317–356. Springer-Verlag.

Nijmeijer, H. and A. J. van der Schaft (1990). *Nonlinear Dynamical Control Systems*. Springer-Verlag. New York.

Peleties, P. and R. DeCarlo (1989). A modeling strategy with event structures for hybrid systems. In: *Proceedings of the 28th Conference on Decision and Control*. Tampa, FL. pp. 1308–1313.

Peleties, P. and R. DeCarlo (1994). Analysis of a hybrid system using symbolic dynamics and petri nets. *Automatica* 30(9), 1421–1427.

Stiver, J. A., P. J. Antsaklis and M. D. Lemmon (1994). Digital control from a hybrid perspective. In: *Proceedings of the 33rd Conference on Decision and Control*. Lake Buena Vista, FL. pp. 4241–4246.

Stiver, J. A., P. J. Antsaklis and M. D. Lemmon (1995). Interface Design for Hybrid Control Systems. Technical Report of the ISIS Group (Interdisciplinary Studies of Intelligent Systems) ISIS-95-001. University of Notre Dame.

Tavernini, L. (1987). Differential automata and their discrete simulators. *Nonlinear Analysis, Theory, Methods, and Applications* 11(6), 665–683.

Tittus, M. and B. Egardt (1994). Control–law synthesis for linear hybrid systems. In: *Proceedings of the 33rd IEEE Conference on Decision and Control*. Lake Buena Vista, FL. pp. 961–966.

Zeigler, B. P. (1989). DEVS representation of dynamical systems: Event based intelligent control. *Proceedings of the IEEE* 77(1), 72–80.