ConPAHS - A Software Package for Control of Piecewise-Affine Hybrid Systems

Pieter Collins, Luc Habets, Anton Kuut, Margreet Nool, Mihaly Petreczky, and Jan H. van Schuppen
Centrum voor Wiskunde en Informatica (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract—The software package ConPAHS facilitates control design of continuous-time piecewise-affine hybrid systems on polytopes. For the control objective of reaching a particular state from a specified initial state, the output of the control law is a piecewise-affine control law. After a short review of the control theory for this problem, the paper presents the functional specification of ConPAHS, the object oriented software principles used, the program structure, and the design choices. Three examples including simulations are provided to illustrate the use of ConPAHS.

I. INTRODUCTION

ConPAHS is a software package for the control design of hybrid systems. A hybrid system is a dynamic system with both discrete and continuous dynamics. In this paper attention is restricted to continuous-time piecewise-affine hybrid systems (PAHS) on polytopes in which the discrete dynamics is described by an automaton and, at each discrete state, the continuous dynamics is described by a continuous-time affine system on a polytope.

Since the reachability problem for piecewise-affine hybrid systems is in general undecidable, see [15], the only effective method for control synthesis is to find sufficient conditions for the solution of the reachability control problem. Such sufficient conditions for control of piecewise-affine hybrid systems on polytopes have been developed by several of the authors, see [8], [9], [10], [11].

Consider for control of a PAHS the reachability control objective of reaching a particular terminal state from an initial state. The approach to this problem consists of a control law both for the discrete and the continuous level. At the continuous level one considers for an affine system on a polytope the control-to-facet approach consisting of synthesizing an affine control law which forces the state to leave the polytope in finite time through one of a prespecified set of exit facets or to remain inside the polytope forever. An application of the control-to-facet approach is the control of the idle speed of a car, see [1].

There is a need for a software package which for large-scale examples computes the control law for the reachability control objective. The software package ConPAHS is a collection of programs for this control design including an input facility, functions to compute control laws, and functions to simulate the resulting closed-loop system.

The current version of the ConPAHS package is written in MATLAB which is the most-used computer language in the research community of control. The package uses the CDD polyhedral library of Fukuda [5], [6], [7]. A planned version of ConPAHS will be written in C++ and is expected to be more efficient. A publically-available version of ConPAHS, including a user manual, is scheduled for a first release in the summer of 2006, and will be made available for download via the web from a page accessible from http://www.cwi.nl/~schuppen.

The novelty of the software package ConPAHS with respect to the software packages Hybrid Toolbox and MPT listed below concern (a) the restriction to continuous-time piecewise-affine hybrid systems (MPT and Hybrid Toolbox concern only discrete-time PAHS) and (b) the control synthesis approach based on control-to-facet (MPT and Hybrid Toolbox use a reduction to a computational problem of mixed-integer programming type). The Hybrid Toolbox was developed by the research group of A. Bemporad at the University of Sienna in Italy, for discrete-time piecewise-affine hybrid systems on polytopes, see [2]; and The Multi-Parametric Toolbox (MPT) [12] was developed by the research group of M. Morari at the ETH in Zürich, Switzerland for discrete-time piecewise-affine hybrid systems on polytopes, see [3]. The developers of ConPAHS have been inspired by the package MPT.

The remainder of the paper is structured as follows. In Section II, we describe the control problem, and present an outline for the computation of a control law. In Section III the ConPAHS specification and implementation are discussed. The structure of the program and the design choices are presented in Section IV. Examples of the use of the software are given in Section V. Concluding remarks are stated in Section VI.

II. CONTROL OF PIECEWISE-AFFINE HYBRID SYSTEMS

The theoretical background of the algorithms in ConPAHS has been developed in several papers (see [8], [9], [10], [11]). In this section, an overview is presented of the problem statements and results in these papers.

Let \( n \in \mathbb{N} \), and consider the \( n \)-dimensional Euclidean space \( \mathbb{R}^n \). A polyhedral set is a subset of \( \mathbb{R}^n \), that is described by a finite number of linear inequalities. If a polyhedral set is bounded, it is called a polytope. A face of a polyhedral set is the intersection of the polyhedral set with one of its supporting hyperplanes. If the dimension of this intersection
is $n - 1$, then the face is called a facet, and if the dimension is 0, then the face is called a vertex. Alternatively, a polytope may be characterized as the convex hull of its set of vertices. This is called the explicit representation of a polytope, whereas the representation in terms of linear inequalities is called implicit. A polytope is full-dimensional if its vertex set contains a subset of $n + 1$ affinely independent points. A full-dimensional polytope with exactly $n + 1$ vertices is called a simplex.

**Definition 2.1:** A (continuous-time) affine system on a polytope consists of an affine system $\dot{q} = (A, B, a)$, a polytope $X \subset \mathbb{R}^n$ and a polyhedral set $U \subset \mathbb{R}^n$. The evolution of the state $q$ satisfies the affine ordinary differential equation

$$\dot{q}(t) = Aq(t) + Bu(t) + a, \quad q(t_0) = q_0, \quad (1)$$

with $u(t) \in U$. In particular it is assumed that differential equation (1) remains valid as long as the state $q$ is contained in the state polytope $X$.

**Definition 2.2:** A (continuous-time) piecewise-affine hybrid system on polytopes consists of an automaton $(Q, E, f)$ in combination with a $|Q|$-tuple of affine systems $\dot{q}_q = (A_q, B_q, a_q)$ defined on polytopes $X_q \subset \mathbb{R}^n$, $\forall q \in Q$, with input in the polyhedral set $U \subset \mathbb{R}^n$. The automaton and the affine systems interact via guard sets in the boundary of the polytope, $G_q(q) \subset \partial X_q$, that consist of facets of $X_q$, and via affine reset maps $R_q(q) : G_q(q) \rightarrow X_{q(e)}$ as follows.

Given a discrete location $q \in Q$, the continuous state $x_q$ evolves according to (1) for the affine system $\dot{q}_q = (A_q, B_q, a_q)$, i.e. with $x_q \in X_q$ and $u \in U$. Whenever the continuous state leaves polytope $X_q$ by crossing the guard set $G_q(q)$, the discrete event $e$ occurs, and a discrete transition takes place according to the transition function

$$f : \subseteq (Q \times E) \rightarrow Q,$$

if $x_q := \lim_{s \rightarrow t_1} x_q(s) \in G_q(q)$, then $q^e = f(q, e)$. At the new discrete location $q^e$, the evolution of the new continuous state $x_{q^e}$ is described by (1) for the affine system $\dot{q}_{q^e} = (A_{q^e}, B_{q^e}, a_{q^e})$, and with initial value $x_{q^e}(t_1) = x_q(t_1)$, determined by the exit point $x_q$ and the affine reset map $R_{q^e}(q)$.

By choosing an appropriate input function $u$, the continuous component of a hybrid trajectory can be influenced directly. Therefore also the switching behavior of the discrete component depends (indirectly) on the continuous input $u$. The goal is to design a piecewise-affine control law, such that the closed-loop hybrid system achieves a specified control objective.

**Definition 2.3:** A continuous piecewise-affine state feedback for an affine system (1) on a polytope $X$ is a continuous function $k : X \rightarrow U$, for which there exists a partitioning of $X$ in polytopes $X_1, \ldots, X_r$, such that for all $j = 1, \ldots, r$, the function $k|_{X_j}$ is affine; i.e. there exists a matrix $F_j$ and a vector $g_j$ such that for all $x \in X_j$: $k(x) = F_j x + g_j$.

**Definition 2.4:** An admissible piecewise-affine hybrid control law is a family $\mathcal{X} = \{k_q \mid q \in Q\}$, where each $k_q : X_q \rightarrow U$ is a continuous piecewise-affine state feedback.
solving the linear equations
\[
\begin{bmatrix}
v_{T,1}^T & 1 & \vdots & \vdots & F_{T}^T_{1} \\
v_{T,2}^T & & \vdots & & \vdots \\
v_{T,N+1}^T & & 1 & & g_{T}^T \\
\end{bmatrix}
= \begin{bmatrix}
u_{T,1}^T \\
\vdots \\
u_{T,N+1}^T 
\end{bmatrix},
\]
for \( F_{T} \) and \( g_{T} \). Furthermore, the piecewise-affine state feedback \( k_{i} \), obtained by combining the affine state feedbacks in each simplex of the triangulation, is a continuous piecewise-affine state feedback (no discontinuities between adjacent simplices) and a solution to Problem 2.6.

Once the continuous control problem of reaching a facet has been solved, one has to combine the piecewise-affine state feedbacks in each discrete location in an intelligent way in order to obtain a solution to the reach-avoid problem. To determine the appropriate exit facet for each discrete location, one may use the same backward recursion algorithm as described in [11], [8].

### III. THE CONPAHS PACKAGE - SPECIFICATION AND IMPLEMENTATION

**A. Functional specification** The first functionality is the input of a specific piecewise-affine hybrid system in the form of an automaton and for each discrete state of the automaton, a polytope and an affine system. The input also includes a specification of the initial state and the terminal state and the numerical specification of the simulation.

The second functionality is the computation of admissible control laws for the reachability control problem for PAHS, including: (1) Computations of the representations and properties of a polytope, including a Delaunay triangulation. (2) Computation per polytope of admissible input vectors at vertices. (3) Computation, per simplex of the polytope, of the control law and of the closed-loop system. (4) Abstraction of the hybrid system in the form of an automaton. (5) A reachability search in the automaton according to the dynamic programming algorithm.

The third functionality is the simulation of a PAHS, usually the closed-loop system. Because of the particular character of such a system, standard simulation packages cannot be used directly.

ConPAHS’s current version, as of January 2006, can compute a control law for the control-to-facet problem with a single exit facet for a continuous-time single affine system on a polytope. The following extensions of the software package are planned: computation of a piecewise-affine control law for a piecewise-affine hybrid system on polytopes; control with multiple exit facets; and computation of a single affine control law per polytope.

**B. Implementation issues** The software package is built on the principles of object-oriented design, see [14], which is one of the mainstream paradigms of software engineering. Its main idea is to formulate the task of the program through interaction of objects. Object-oriented program design is intended for designing larger programs which should be easy to modify and extend. It is especially useful for developing program libraries, because such libraries tend to have a long lifetime and may be frequently changed and/or extended.

ConPAHS is currently implemented in MATLAB and the standard constructions of MATLAB for object-oriented design are used. Unfortunately, the built-in support for object-oriented programming within MATLAB is not particularly satisfactory, and apart from the built-in numerical routines, code written in MATLAB tends not to be particularly efficient. For these reasons, a re-implementation of the library in C++ is under development. C++ is one of the most widespread programming languages, with good support for object-orientation and which provides the opportunity for efficient implementations of algorithms. The C++ version of the package will be still provided with a MATLAB interface to facilitate its use.

The software package ConPAHS makes use of the CDD library of Fukuda [5], [7]. This library is implemented in both C and C++, and implements a wide range of operations on polyhedral sets.

### IV. THE CONPAHS PACKAGE - DESCRIPTION

**A. System** The structure of the program is outlined in Fig. 1. The main object the user works with belongs to the class of AffineSystemOnPolytope and describes an affine system on a polytope. To create this object, one has to provide a polytope as the object from class Polytope, and an affine control system in the form of object of class AffineSystem. An object Polytope can be created as the convex hull of the rows of the matrix \( V \in \mathbb{R}^{n \times N} \) which form the vertices of \( P \), \( P = \text{convh}(V) \) (explicit representation), or, by defining a set of inequalities
\[
P = \{ x \in \mathbb{R}^n | Cx \leq w \},
\]
with \( C \in \mathbb{R}^{M \times n} \) and \( w \in \mathbb{R}^M \) (implicit representation). The subroutine call of Polytope can compute one representa-
tion from the other. Moreover, the hyperplane representation of an object Polytope will always be a minimal representation. In this respect, the ConPAHS object Polytope differs from the MPT object Polytope.

An affine system as defined in equation (1) is implemented by the class AffineSystem. An object of this class has the following fields: (i) A, an n-by-n matrix A, (ii) B, in case B is non-empty, the n-by-m matrix B, (iii) a, an n-by-1 vector a.

A triangulation of P into simplices $S_1, \ldots, S_r$ is computed using the delaunay routine of MATLAB, and is described by the class TriangulationOfPolytope. The TriangulationOfPolytope command with as input an object of class Polytope returns a triangulation in the form of a set of simplices, where each simplex is described by the indices of the vertices of the polytope $P$. The result is an object which contains the vertices of the polytope stored in the matrix $V$, the number of simplices $r$, the set of simplices $S_1, \ell = 1, \ldots, r$, and an array with the global reference numbers of the vertices of these simplices, and an array of facets of these simplices. ConPAHS includes routines to show the contents of an object, its fields and their values. Moreover, for dimensions $n = 2$ and $n = 3$ it offers the possibility of plotting the polytope and its triangulation.

B. Control Design For computing a control law which drives the state to a particular exit facet $H$, one has to specify an input vector $u_j \in U$, for each vertex $v_j$, $j = 1, \ldots, N$ such that linear inequalities (2,3) of Theorem 2.7 are satisfied. The computation of all input vectors at the vertices of polytope $P$ in $\mathbb{R}^n$ requires the solution of $N$ systems of at least $n$ linear inequalities. Such systems of inequalities describe polyhedral sets which may be unbounded. In the case that the input vector $u_j$ may be chosen from an unbounded polyhedral set, artificial bounds are introduced. Next, ConPAHS chooses as a value for $u_j$, the center of the largest ball inscribed in the bounded polytope $U_j$ of admissible input vectors. The solution exists for a user-specified exit facet $H$, when for $H$ the inequalities (2) are satisfied and for every facet $K \neq H$ containing $v_j$ condition (3) holds.

A control law for a polytope can be thought of as a family of control laws; each element of the family defines a control law on a particular simplex. These simplices are obtained by a triangulation of the polytope, on which the system is defined. Control laws are represented by objects of the class ControlLaw and each control law on a simplex is an object of the class LocalControlLaw. Given a triangulation of $P$ into $r$ simplices, the local control law for each simplex $S_\ell, \ell = 1, \ldots, r$ is computed by solving the linear equations (4). In MATLAB the vector $(F_\ell | g_\ell)^T$ is computed as the solution of (4) in a numerical reliable way. The effective rank is determined from the QR decomposition with pivoting.

The object ControlLaw is defined for an affine system on a polytope; it combines the data of object LocalControlLaw and the information on the partitioning of $P$ into simplices described by the class TriangulationOfPolytope.

The closed-loop system obtained by applying the control law to the original system has a structure different from the original open-loop system. It is defined on a set of adjacent simplices instead of one polytope and on each simplex the vector field takes a different form. An affine system on a simplex is represented as an element of class LocalClosedLoopSystem and the closed-loop system itself is an object of the class ClosedLoopSystem. An object LocalClosedLoopSystem, with as input arguments the objects AffineSystem and LocalControlLaw, describes the autonomous affine system

$$x(t) = (A + BF_\ell)x(t) + (a +Bg_\ell) = A_\ell x(t) + B_\ell,$$  

where $A, B$ and $a$ originate from the AffineSystem, and $F_\ell$ and $g_\ell$ from the object LocalControlLaw corresponding to simplex $S_\ell$.

The last part of the control design covers the creation of an object of the class ClosedLoopSystem, with as input arguments the objects AffineSystem and ControlLaw. The fields of the class ClosedLoopSystem are: (i) $r$ which denotes the number of simplices, (ii) LCSL$(1:r)$, an array of objects LocalClosedLoopSystem, (iii) Vertices$(1:r)$, an array of global vertices, and (iv) Facets$(1:r)$, an array of facets.

C. Simulation For each simplex of the triangulation of polytope $P$, a local control law as well as a local closed-loop system is computed. Also the simulation will be executed per simplex. For that purpose the classes SimulationOnClosedLoopSystem and SimulationOnLocalClosedLoopSystem have been created. The simulation on the closed-loop system (6) starts with the integration of the closed-loop system on the simplex $S_\ell$, that contains the initial state $x_0$, until the boundary of $S_\ell$ is reached.

Currently, the MATLAB function ODE45, a Runge-Kutta method with a variable time step, is used to integrate the system within a simplex $S_\ell$. The result is a sequence of points $x_i = x(t_i), t = 1, \ldots, s$ giving approximations to the exact evolution at the time interpolation points $x_i$. The program halts the integration if it detects that $x_{i+1} \not\in S_\ell$, i.e., $C_i x_{i+1} + w_i > 0$, for some $i_1, 1 \leq i_1 \leq i_2$. If $\|x_{i+1} - x_i\| < \varepsilon$ for some (small) tolerance $\varepsilon$, then integration is continued in the simplex $S_i$ containing $x_{i+1}$ using the affine control law $u = F_i x + g_i$. Otherwise, the program discards $x_{i+1}$, and continues integrating from $x_i$ at time $t_i$ with a reduced step size, halting again if it leaves $S_i$. This process is repeated until the program finds points $x_{in} \approx x(t_{in})$ and $x_{out} \approx x(t_{out})$ such that

$$x_{in} \in S_\ell, \quad x_{out} \not\in S_\ell \quad \text{and} \quad \|x_{in} - x_{out}\| < \varepsilon,$$  

where $\varepsilon$ is a user-specified tolerance. Note that the iteration process for reaching the boundary of $S_\ell$, based on condition (7), may lead to another exit facet than the one obtained after the first iteration.
An object `ClosedLoopSystem` contains information on all simplices in which the polytope has been partitioned. It is relatively simple to determine in which simplex the simulation will continue, i.e., in the simplex for which the set of inequalities
\[
C_j x_{\text{out}} \leq w_j \quad \text{for} \quad j \neq \ell
\]
is satisfied, where \( C_j \) and \( w_j \) form the implicit representation of simplex \( S_j \) (cf. definition (5)). Only those simplices that have a common facet with simplex \( S_\ell \) will be considered as possible candidates. Obviously, the set of simplices can be reduced if the facet \( H_{\ell,\text{exit}} \), through which the state \( x \) leaves \( S_\ell \) is known. In that case only simplices with facet \( H_{\ell,\text{exit}} \) are of interest.

V. EXAMPLES

A. Extended example of a two-dimensional system

Consider the system
\[
x = \begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix} x + \begin{pmatrix} 1 \\ -4 \end{pmatrix} u + \begin{pmatrix} 0.2 \\ 0.3 \end{pmatrix}
\]
on a 2D unit square with extreme points \( v_1 = (0, 0), v_2 = (1, 0), v_3 = (0, 1), \) and \( v_4 = (1, 1). \) The MATLAB program below lists the steps to be taken from the initialization of the polytope \( P \) and the affine system \( S \) through the simulation process, see also Section IV. For simplicity the calls to functions which generate output and make plots are not included. In addition, default values are applied where necessary.

```matlab
% Define polytope P by its V-representation
V=[0 0; 1 0; 0 1; 1 1]; P=Polytope(V);

% Define the affine system by A, B, and a
A=[3 -1; -2 2]; B=[1; -4]; a=[0.2; 0.3];
S=struct('A',A,'B',B,'a',a);
S=AffineSystem(S);
sAP=AffineSystemOnPolytope(sA,P);

% Partition a polytope into simplices
ToP=TriangulationOfPolytope(P);

% Initialize ExitFacet and Constraints U
ExitFacet=4; ConstraintsU= []; InputV=InputVectorsAtVertices(sAP,...
ExitFacet,ConstraintsU);

% Compute the ControlLaw
CL=ControlLaw(ToP,InputV);

% Compute the ClosedLoopSystem
CLS=ClosedLoopSystem(sA,CL);

% Simulation
xin = [1e-5;0]'; [xk_out,ODEValues]= ... SimulationOnClosedLoopSystem(xin,CLS);
```

First the polytope and the system are initialized. We remark that the function call \( P=\text{Polytope}(V) \) includes the computation of both polytope representations (see Section II).

Even in case \( P \) is a simplex, the Delaunay triangulation of \( P \) must be carried out to create an object of class `TriangulationOfPolytope`. The polytope \( P \) in this example can be triangulated in two ways. The first one is shown in Figure 2, where the simplices \( S_1 \) and \( S_2 \) with explicit representation \((v_4,v_3,v_1)\) and \((v_4,v_2,v_1)\) are achieved, respectively. The other possibility delivers the simplices \((v_1,v_2,v_3)\) and \((v_2,v_3,v_4)\).

Numerical experiments show that exit facet \( F_4 \) (through \((v_2,v_4)\)) results into a closed-loop system such that for an arbitrary initial state \( x_0 \) the terminal state lies on \( F_4 \) without first leaving the polytope through another facet. By choosing the constraints empty, the lower and upper bound for each input vector \( u_1,u_2,u_3,u_4 \) achieve the default values. If there exists a solution, then field \( u = [u_1^T,u_2^T,u_3^T,u_4^T] \) of object \( \text{InputV} \) satisfies the conditions (2,3). For this example the values \( u_1 = -\frac{1}{10}, u_2 = -1\frac{3}{10}, u_3 = 2\frac{9}{10} \) and \( u_4 = 2\frac{3}{10} \) are obtained.

The computation of the control law on polytope \( P \) by a call of `ControlLaw` includes the computation of the control laws on the simplices \( S_1 \) and \( S_2 \). For simplex \( S_1 \) the solution of system (4) results into the vectors \( F_1 = (-\frac{29}{20},-\frac{27}{20}) \) and \( g_1 = (-\frac{1}{10}) \) and for simplex \( S_2 \) the vectors \( F_2 = (-\frac{13}{4}, \frac{7}{20}) \) and \( g_2 = (-\frac{1}{10}) \) are obtained. Analogously, both simplices have their own local closed-loop system, included in object `CLS`: the affine systems (6) \( S_1 \) and \( S_2 \) are respectively specified by
\[
A_1 = \begin{pmatrix} 2.6375 & 1.9625 \\ -0.5500 & -9.8500 \end{pmatrix} \quad \text{and} \quad B_1 = \begin{pmatrix} 0.1375 \\ 0.5500 \end{pmatrix}
\]
\[
A_2 = \begin{pmatrix} 1.2500 & 3.3500 \\ 5.0000 & -15.4000 \end{pmatrix} \quad \text{and} \quad B_2 = \begin{pmatrix} 0.1375 \\ 0.5500 \end{pmatrix}
\]
In this example the initial state is chosen equal to \( x_0 = (10^{-5}, 0) \), a point in \( S_2 \). The threshold value \( \varepsilon \) used in (7) equals \( 10^{-8} \). The simulation trajectory starts in simplex \( S_1 \), reaches the internal facet \((v_1, v_4)\) at \((1.333 \times 10^{-3}, 1.333 \times 10^{-5})\), enters simplex \( S_2 \). Then the state \( x \) reenters simplex \( S_1 \) at \((4.547 \times 10^{-2}, 4.547 \times 10^{-2})\). Finally, \( P \) is left in terminal state \((1.0, 3.153)\). The trajectory \( x(t) \) is plotted in Figure 2, together with seven other trajectories.

**B. Example in dimension three.** Consider the system

\[
\dot{x} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} x + \begin{pmatrix} 1 & 0 \\ 2 & 2 \\ 0 & 1 \end{pmatrix} u + \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \tag{11}
\]

on a three-dimensional cube with extreme points \( v_1 = (1, 1, 1), \ldots, v_8 = (-1, -1, -1) \), as shown in Figure 3. As exit facet the plane through \((v_2, v_4, v_5, v_8)\) is chosen, the left side of the cube in Figure 3. The triangulation used consists of six simplices. Choose a set of four initial states for the simulation. Three trajectories leave the polytope through simplex \( S_2 \), the fourth leaves through \( S_1 \).

**C. Example in dimension seven.** Consider the system

\[
\dot{x} = Ax + Bu + g, \tag{12}
\]

where \( A = \text{Diag}((-1, -2, -3, -4, -3, -2, -1)^T) \), and

\[
B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 6 & 5 & 2 & 3 \\ 4 & 1 & 5 & 3 \\ 2 & 6 & 3 & 7 \\ 1 & 4 & 2 & 1 \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} -1 \\ 0 \\ -4 \\ 0 \\ -3 \\ -2 \end{pmatrix}, \tag{13}
\]

which lives on a seven dimensional unit simplex with vertices \( v_i = e_i \), where \( e_i \) is the \( i \)-th unit vector in \( \mathbb{R}^7 \), and the origin \( v_8 = (0, 0, 0, 0, 0, 0, 0)^T \). As exit facet choose the hyperplane opposite to \( v_8 \). Again, an admissible piecewise-affine control law was computed, and four initial states chosen for simulation. The computations were performed in a fraction of a second.

**VI. CONCLUDING REMARKS**

The paper presents the software package ConPAHS for computation of control laws for continuous-time piecewise-affine hybrid systems on polytopes. The package is planned to be released later in 2006 to several research groups for testing on realistic systems of engineering applications.

**ACKNOWLEDGMENTS**

Financial support in part by the European Commission via the project Control and Computation (CC; IST-2001-33520) is herewith gratefully acknowledged. The financial support of the Centrum voor Wiskunde en Informatica (CWI) is also acknowledged.

**REFERENCES**


