Decomposed Reachability Analysis for Nonlinear Systems

Xin Chen
University of Colorado, Boulder, CO
xinchen@colorado.edu

Sriram Sankaranarayanan
University of Colorado, Boulder, CO
srirams@colorado.edu

***Abstract*—We introduce an approach to conservatively abstract a nonlinear continuous system by a hybrid automaton whose continuous dynamics are given by a decomposition of the original dynamics. The decomposed dynamics is in the form of a set of lower-dimensional ODEs with time-varying uncertainties whose ranges are defined by the hybridization domains. We propose several techniques in the paper to effectively compute abstractions and flowpipe overapproximations. First, a novel method is given to reduce the overestimation accumulation in a Taylor model flowpipe construction scheme. Then we present our decomposition method, as well as the framework of on-the-fly hybridization. A combination of the two techniques allows us to handle much larger, nonlinear systems with comparatively large initial sets. Our prototype implementation is compared with existing reachability tools for offline and online flowpipe construction on challenging benchmarks of dimensions ranging from 7 to 30.**

I. Introduction

In this paper, we present a more scalable flowpipe construction technique for the reachability analysis of nonlinear continuous systems, with applications to Cyber-Physical Systems (CPS). Given the model of a hybrid system with initial sets, sets of disturbance inputs and sets of model parameters, the flowpipe construction technique seeks to compute the set of all reachable time trajectories over a given time horizon T. This enables the verification of bounded real-time properties. Flowpipe construction for linear hybrid systems has been quite successful with efficient tools such as SpaceEx [18]. However, a closer examination of the state-of-the-art approaches for *nonlinear systems* reveals two principal drawbacks: (a) many approaches do not scale beyond systems with 5$\sim$8 state variables, and (b) the overestimation error can be quite large, even for systems with few variables and small initial sets, leading to overapproximations that may not be useful.

In this paper, we present a flowpipe construction scheme using Taylor model arithmetic [8], that exploits the structure of the terms in the given differential equation model to *decompose* a flowpipe construction for a larger system into flowpipe construction tasks over smaller subsystems. The main idea is to abstract the system by replacing carefully selected state variables in the RHS of the Ordinary Differential Equation (ODE) by intervals. As a result, these variables are assumed to be time-varying uncertainties that lie inside those intervals, which are taken to be an assumption. The variables are also selected, so that the resulting abstract system can be decomposed into a set of smaller, independent subsystems with no state variables in common (other than a common notion of time). The integration is performed with the assumption that the abstracted variables are inside the interval. If the check passes, we prove that the resulting flowpipe is indeed a valid overapproximation of the dynamics over a time step. In this case, the flowpipes computed are then used to iteratively refine/narrow the intervals, and thus reduce the overapproximation error of the computed flowpipe. Failing this check, the assumption intervals are enlarged and/or the time step of integration is shortened. As a result, our decomposition approach resembles an on-the-fly hybridization scheme wherein only selected variables are hybridized to decompose the system structurally.

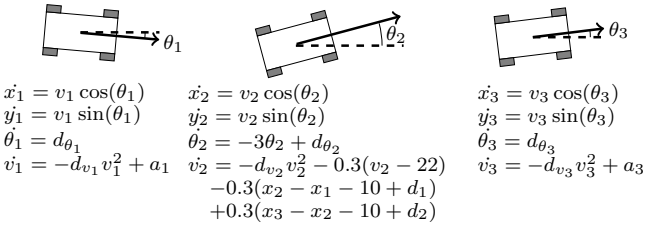


Fig. 1. Car platooning model showing three cars V_1, V_2 and V_3 from left to right, implementing a potentially unsafe control scheme for avoiding collisions.

However, using time-varying uncertainties may lead to heavy accumulation of overapproximation error which is also called *overestimation* in flowpipe construction. To avoid this, we propose a method to symbolically track the remainder over multiple steps.

We implement this approach on top of the tool Flow* and compare with related tools including Flow* [11], CAPD [23] and VNODE-LP [31] for both offline and online flowpipe computation. The comparisons are based on some challenging benchmarks with 7 to 30 system variables.

Motivation. As a motivating example, consider the problem of monitoring a safety critical control system to check if starting from a current state $\vec{x}(t)$, the system is guaranteed to remain within a safe region S in the time interval $[t, t+T]$. For real-time monitoring applications, we require that the reachable set can be computed rapidly, within the time horizon T so that a switch to a safe control scheme can be made [7], [36].

Figure 1 shows an adaptive cruise control model for a "platoon" of three vehicles, that controls the middle vehicle V_2

TABLE I
DESCRIPTION OF DISTURBANCE INPUTS IN THE CAR PLATOONING MODEL.

Time-varying uncertainties	Source	Range
$d_{\theta_1}, d_{\theta_2}, d_{\theta_3}$	steering disturbance	$[-5°, 5°]$
a_1, a_3	acceleration of V_1, V_3	$[-0.45, 0.45]$
$d_{v_1}, d_{v_2}, d_{v_3}$	drag/friction coefficients	$[0.09, 0.1]$
d_1, d_2	sensor disturbances	$[-0.05, 0.05]$

from colliding with vehicle V_1, V_3 that are behind and ahead of it, respectively. Each vehicle is modeled using the state variables $(x_i, y_i, v_i, \theta_i)$ for $i \in \{1, 2, 3\}$ to denote its position x_i, y_i, velocity v_i and orientation θ_i.

Figure 1 shows the ODE for each of the cars. The cars V_1, V_3 are assumed to be driven by humans. We assume that the accelerations and steering angles are uncertain. The middle car is actively controlled to avoid collisions. We assume that the middle car is equipped with a radar to estimate the positions x_1, x_3 of the leading and trailing cars continuously, but with time-varying estimation errors. It then uses proportional control on its acceleration to avoid collisions, wherein the collision free region is defined by $x_2 - x_1 > 2 \wedge x_3 - x_2 > 2$.

The overall model has 12 state variables in all, and is coupled by the feedback involving x_1, x_3 for the acceleration of the second car. The model also involves time-varying disturbances shown in Table I.

At the beginning of each time period of $T = 0.99 < 1$ second, an estimate of the current state with uncertainty is available to the online monitor. The goal is to predict all possible reachable states of the system within T to decide if a collision is imminent. If yes, evasive action or fall back to a safe controller is taken, following the principle of a Simplex architecture [36]. Such an application requires a fast flowpipe construction scheme, that is capable of computing the reachable set within T rapidly, ideally requiring $t \ll T$ computation time. As reported in Section V, the techniques presented in this paper can exploit the "loose coupling" between the states of V_2 and those of V_1, V_3 to achieve computation times ranging in $[0.68, 0.81]$ seconds under various initial conditions. In comparison, the original Flow* tool under the same settings takes around $10\times$ more time.

Related work. Overapproximating solutions for nonlinear ODEs plays an important role in the safety analysis of nonlinear hybrid systems and in the control of CPS in a provably safe manner [6]. Some existing tools have already shown their applicability to some nontrivial safety problems, such as Flow* [11], iSAT [16], dReach [24], NLTOOLBOX [38], C2E2 [15], HyCreate [5], and CORA [1]. Frameworks like HYST provide convenient interfaces for performing complex real-time CPS verification tasks using these tools [4]. However, it is still a difficult task to handle the continuous system defined by a nonlinear ODE beyond 10 or more variables, especially when the initial set is relatively large, and the unsafe set (property) lies close to the boundary of the precise reachable set. In contrast, computing flowpipes for linear ODEs have been shown to scale using symbolic representations such as support functions [18] and polynomials [34].

Whereas many approaches discussed this far perform offline verification, Bak et al. present an application to "online verification" for implementing a Simplex architecture [7]. Their approach checks if the system will continue to satisfy its specification for a given real-time horizon T, starting from its current measured/estimated state. A key requirement for such an application is that the running time for the flowpipe construction be less than T, so that it can be implemented in real time. Whereas the approach of Bak et al. tackles linear systems, we demonstrate online verification for nonlinear systems through a combination of Taylor model flowpipe construction and the decomposition technique presented here.

Approaches to nonlinear flowpipe construction rely on higher-order interval arithmetic [8], [10] to minimize a potentially expensive gridding of the state space. However, the size of the Taylor model required to maintain a given precision and avoid error accumulation can grow exponentially in the number of system variables in the worst case. Alternatively, many approaches *hybridize* the system by approximating it, to simplify the dynamics to piecewise affine or constant hybrid systems [20], [17], [3], [2], [12], [13]. Our method combines both the techniques of symbolic representations and hybridization. However, we avoid expensive flowpipe representations through a combination of lower-dimensional polynomials with symbolic remainder tracking. Furthermore, unlike the existing hybridization methods [3], [2], [13], we do not hybridize the entire state space. Instead, we target specific terms in the ODEs to remove key dependencies between system variables that will decompose the system.

Compositional methods have been applied to computing approximate abstractions [35] and discrepancy functions [21] for continuous and hybrid systems. Decomposition methods were considered in stability analysis [37] and obtaining differential invariants [33]. However, in this paper, we present a novel approach combining *decomposition* and *hybridization* to efficiently compute flowpipe overapproximations for large-scale systems. If the given system is already provided as a composition of smaller components, it is possible to use these components to define a decomposition. However, many large models are often not provided in such a way. Nevertheless, we observe that hybridizing few select variables allows us to decompose the model into much smaller submodels.

The rest of the paper is organized as follows. Section II presents the preliminaries including the method of computing Taylor model flowpipes for nonlinear ODEs. The treatment of time-varying uncertainties is presented in Section III. We use a symbolic remainder to minimize the accumulation of overapproximation error. In Section IV, we first introduce our decomposition method on ODEs, and then show the framework of computing partial hybridizations for obtaining decompositions. We compare our prototype tool with some related tools on a set of benchmarks in Section V.

II. PRELIMINARIES

In the paper, we use $\mathbb{R}$ for the set of reals. A set of ordered variables $x_1, \ldots, x_n$ or a vector $(x_1, \ldots, x_n)$ are collectively represented by $\vec{x}$. Given a vector or a vector-valued function f, we denote f_i or $f[i]$ the ith component of f.

Definition II.1 (Continuous system)**.** *An n-dimensional* continuous system $\mathcal{S}$ *is defined by an ODE $\dot{\vec{x}} = f(\vec{x})$ such that $\vec{x}$ is a vector representation of the state variables, and $f : \mathbb{R}^n \to \mathbb{R}^n$ defines the* vector field *which associates each state $\vec{c} \in \mathbb{R}^n$ a derivative vector $f(\vec{c}) \in \mathbb{R}^n$.*

An execution of a continuous system is a solution of its ODE. Given a continuous system $\mathcal{S} : \dot{\vec{x}} = f(\vec{x})$ and an initial condition $\vec{x}(0) = \vec{x}_0$, we denote the solution at some time $t \geq 0$ by $\varphi_f(\vec{x}_0, t)$ or $\vec{x}(t)$ if $\vec{x}(0)$ is given in the context. Throughout this paper we assume that f is at least locally Lipschitz continuous, that is, there exists some nonempty open set Ω containing $\vec{x}_0$ and a real value $L_f \geq 0$ such that $\|f(\vec{x}_1) - f(\vec{x}_2)\| \leq L_f \|\vec{x}_1 - \vec{x}_2\|$ for all $\vec{x}_1, \vec{x}_2 \in \Omega$. Here, $\|\cdot\|$ denotes the infinity norm and L_f is called a *Lipschitz constant* of f in Ω. Then, if the flow $\varphi_f(\vec{x}_0, t)$ exists, it is unique. If the initial value of $\vec{x}$ is given by a set X_0, we collectively denote the set of the solutions by $\varphi_f(X_0, t)$ for $t \geq 0$. We also call the set of solutions over a bounded time interval a *flowpipe*. We call a state $\vec{s}$ *reachable* in some time interval Δ if there is $t \in \Delta$ such that $\vec{s} = \varphi_f(\vec{x}_0, t)$ for some initial state $\vec{x}_0$. In the rest of the paper, we will refer to continuous systems and ODEs interchangeably.

Since nonlinear ODEs may not have known analytic closed-form solutions, we resort to overapproximation methods. Widely used overapproximate representations include intervals and Taylor models.

Interval arithmetic. A closed and bounded interval (also called a *box*) $\{x \in \mathbb{R} \,|\, a \leq x \leq b\}$ is denoted by $[a, b]$. The operations on reals can be extended to handling intervals. For example, the interval addition and multiplication are defined by $[a, b] + [c, d] = [a + c, b + d]$ and $[a_1, b_1] \cdot [a_2, b_2] = [\min\{a_1 \cdot a_2, a_1 \cdot b_2, b_1 \cdot a_2, b_1 \cdot b_2\}, \max\{a_1 \cdot a_2, a_1 \cdot b_2, b_1 \cdot a_2, b_1 \cdot b_2\}]$ respectively. Intervals are used as overapproximate representations for reals in numerical computation. They can also be organized as vectors or matrices. An interval vector (or matrix resp.) V, denotes a set of vectors (matrices), wherein $v \in V$ iff each entry of v is contained in the corresponding interval entry of V. We refer to the textbook by Moore & Cloud for further details [30]. In the paper, we also call interval vectors intervals or boxes, and use $B[i]$ to denote the ith component of an interval vector B.

Definition II.2 (Taylor model [8])**.** *An order k* Taylor model (TM) *is a pair $(p(\vec{x}), I)$ wherein p is a polynomial of degree k over $\vec{x}$, and I is the remainder interval. The variables $\vec{x}$ are associated with an interval range D which is called the domain of the TM.*

TMs can be viewed as higher-order intervals, such that a part of the uncertainty is represented by polynomials. They are used to provide overapproximations for sets of smooth or continuous functions. A (vector-valued) function $f(\vec{x})$ with $\vec{x} \in D$ is *overapproximated* by the TM $(p(\vec{x}), I)$ iff $\forall \vec{x} \in D.\ f(\vec{x}) \in p(\vec{x}) + I$. Likewise, a TM can also be used to represent the set given by its image: $\{\vec{y} \mid \vec{y} \in p(\vec{x}) + I \text{ for some } \vec{x} \in D\}$. TMs are closed under operations such as addition, multiplication, and integration (see [28]). Given functions f, g that are overapproximated by TMs (p_f, I_f) and (p_g, I_g) respectively. A TM for $f + g$ can be computed as $(p_f + p_g, I_f + I_g)$, and an order k TM for $f \cdot g$ can be computed as $(\, p_f \cdot p_g - r_k\,,\, I_f \cdot B(p_g) + B(p_f) \cdot I_g + I_f \cdot I_g + B(r_k)\,)$ wherein $B(p)$ denotes an interval enclosure of the range of p, and the *truncated part* r_k consists of the terms in $p_f \cdot p_g$ of degrees $> k$.

A. Taylor model flowpipes

Given a continuous system $\mathcal{S} : \dot{\vec{x}} = f(\vec{x})$ and an initial set $\vec{x}(0) \in X_0$ such that X_0 is represented by a TM or interval, the method of TM integration is to compute a finite set of TM flowpipes $(p_1, R_1), \ldots, (p_N, R_N)$ such that $(p_i(\vec{z}, t), R_i)$ is an overapproximation of $\varphi_f(\vec{z}, t + (i-1)\delta)$ with $\vec{z} \in X_0$, $t \in [0, \delta]$, for $1 \leq i \leq N$. We recall the approach to compute these approximations [8], [28]. For the ith integration step, the *local initial set* is X_0 for $i = 1$, and computed as a TM $X_{i-1} = (p_{i-1}(\vec{z}, \delta), R_{i-1})$ for $i \geq 2$. Then, the ith TM flowpipe can be obtained from the following steps.

(1) Compute the order k Taylor expansion $\Phi_i(\vec{y}, t)$ at $t = 0$ for the ODE solution $\varphi_f(\vec{y}, t)$, with the domain $\vec{y} \in X_{i-1}$.

(2) Find a proper remainder I_i such that $\varphi_f(\vec{y}, t)$ is overapproximated by $(\Phi_i(\vec{y}, t), I_i)$ with $t \in [0, \delta]$. It can be done by verifying the contractiveness[1] of the Picard operator on $(\Phi_i(\vec{y}, t), I_i)$ (see [8], [10]).

(3) Compute the ith TM flowpipe (p_i, R_i) by evaluating $(\Phi_i(X_{i-1}, t), I_i)$ using TM arithmetic.

The polynomial part of a TM flowpipe is a vector-valued polynomial. Its jth component defines a polynomial approximation for the flowmap φ_f in the jth dimension. In the rest of the paper, we also call flowpipe overapproximations *flowpipes* if it is clear in the context that they are overapproximations.

B. Symbolic versus interval representation for initial sets.

A TM flowpipe keeps the initial set symbolically by the n variables $\vec{z}$, and that results in a representation size at least as large as that of a high order polynomial of n variables, and could be exponential in n. On the other hand, the interval-based integration method [32] uses Interval Taylor Series (ITS) to represent a flowpipe, in which the initial set is represented by its interval enclosure and the representation is only a univariate polynomial in t with interval coefficients. Although the size of an ITS is much smaller than that of a TM in general, it can hardly track a flow accurately when the initial set is relatively large. In that case, one may have to perform a subdivision on the initial set and do integration for each piece, and that often costs much more time than computing TMs. Some experimental comparisons are given in [9].

[1] The resulting TM is contained in the input TM.

In this paper, we introduce an approach to partially represent the initial set symbolically such that some of the variables in $\vec{z}$ are replaced by their intervals. The selection of the replaced variables are handled by our *decomposition method* presented in Section IV.

C. Time-varying uncertainties.

Our approach will deal with the nonlinear ODE terms by means of time-varying parameters inside an interval. This requires careful handling during the TM integration. The standard TM integration technique is further extended to dealing with time-varying uncertain parameters in [9]. Surprisingly, checking the contractiveness of the Picard operator with all time-varying uncertainties replaced by their interval bounds suffices to handle these parameters. Since the solution is unique in the situation where each uncertainty is given by a continuous function of t, and TMs are set-based representation for continuous functions, the contractiveness means all unique solutions are included by the resulting TM.

Theorem II.1 ([9]). *Given a continuous system $\mathcal{S} : \dot{\vec{x}} = f(\vec{x}, \vec{u})$ wherein $\vec{u}$ are time-varying uncertainties and bounded by $\mathcal{U} \in \mathbb{IR}^m$. If the Picard operator $\mathbb{P}_f(g)(\vec{y}, t) = \vec{y} + \int_0^t f(g(\vec{y}, s), \mathcal{U})\, ds$ is contractive on the TM $(p(\vec{y}, t), I)$ with $\vec{y} \in X$ and $t \in [0, \delta]$, then $(p(\vec{y}, t), I)$ is an overapproximation of the solutions of the uncertain ODE from X in the time interval $[0, \delta]$.*

Although the above theorem provides a way to compute TM flowpipes for uncertain ODEs, the remainder part of each TM flowpipe is often large and the overestimation can easily accumulate along flowpipe construction. We provide the following example to show that even the uncertainties are very small, it is still hard to compute overapproximations for the solutions.

Example II.1. *The model of Higgins-Sel'kov Oscillator is defined by the ODE $\dot{S} = v_0 - S \cdot k_1 \cdot r(P)$, $\dot{P} = S \cdot k_1 \cdot r(P) - k_2 \cdot P$ wherein the typical values for the parameters are $v_0 = 1$, $k_1 = 1$, $k_2 = 1.00001$, and the simplest expression for $r(P)$ is P^2. Since the model describes a class of enzyme reactions such that S, P are the concentrations of substrate and product respectively, it is possible for the parameters to have additive time-varying uncertainties. We assume that all uncertainties are within the interval $[-0.0002, 0.0002]$, and therefore the dynamics becomes an ODE with time-varying uncertain coefficients. We consider the initial set $S(0) \in [1.99, 2.01]$, $P(0) \in [0.99, 1.01]$ and try to compute TM flowpipes in the time horizon $[0, 10]$. We employ a stepsize of 0.02 and a TM order of 6. The tool Flow* [11] fails to wrap the reachable set at time 4.44 because of the remainder explosion. We then reduce the stepsize to 0.002 and increase the order to 7, however the tool still terminates with the same failure at the time 4.510 with 182 second computation time.*

The main problem here is the accumulation of overestimation which makes it hard to produce a flowpipe within the required error interval bounds and polynomial degrees. In the next section, we present a novel method to reduce this accumulation using symbolic remainder representation.

III. Reduction of overestimation

One of the main difficulties of flowpipe overapproximation is to reduce the accumulation of overestimation. Since a flowpipe is computed based on the previous one, the overestimation is also propagated. For linear ODEs, since the closed-form solution is known, we may either compute each flowpipe independently (see [9]), or use a symbolic flowpipe representation, such as support functions [26], or TMs with symbolic remainders [9]. However, neither of the schemes can be applied to dealing with nonlinear ODEs.

In this section, we introduce a method to reduce the overestimation accumulation in TM flowpipe construction. The purpose is to better deal with the ODEs with time-varying uncertainties, since their TM flowpipes are often with large remainder parts. Based on this method, we can more effectively compute TM flowpipes in the decomposition scheme that will be introduced in Section IV.

Since a local initial set in an integration step is computed by bloating the image of the previous local initial set under a polynomial transformation, we may split that transformation into linear and nonlinear parts, and try to reduce the overestimation accumulation under the linear part.

In the ith integration step, as we described in Section II-A, the local initial set X_{i-1} is given by $p_{i-1}(\vec{z}, \delta) + R_{i-1}$, wherein $\vec{z} \in X_0$, X_0 is the initial set and δ is the time stepsize. Thus, the local initial set X_i for the next time step is computed as the result of $\Phi_i(p_{i-1}(\vec{z}, \delta) + R_{i-1}, \delta) + I_i$, wherein Φ_i is the Taylor expansion for the solution in the ith step, and I_i is the remainder interval for Φ_i. We expand the expression of X_i as below,

$$\begin{aligned} p_i(\vec{z}, \delta) + R_i =& \Phi_i(p_{i-1}(\vec{z}, \delta) + R_{i-1}, \delta) + I_i \\ =& \Phi_{i,L}(p_{i-1}(\vec{z}, \delta) + R_{i-1}, \delta) + \\ & \underbrace{\vec{c}_i + \Phi_{i,N}(p_{i-1}(\vec{z}, \delta) + R_{i-1}, \delta) + I_i}_{S_i(\vec{z}) + J_i} \end{aligned} \tag{1}$$

wherein $\Phi_{i,L}$ and $\Phi_{i,N}$ are the linear and nonlinear part respectively of Φ_i, and $\vec{c}_i$ is the constant part. The constant and nonlinear part as well as the remainder I_i can be computed as a TM $S_i(\vec{z}) + J_i$. If the expression is recursively expanded, we then obtain that $p_i(\vec{z}, \delta) + R_i$ is equivalent to

$$\begin{aligned} & \Phi_{i,L}(\Phi_{i-1,L}(p_{i-2}(\vec{z}, \delta) + R_{i-2}, \delta) + S_{i-1}(\vec{z}) + J_{i-1}, \delta) \\ & + S_i(\vec{z}) + J_i \\ =& \Phi^\delta_{i,L} \cdot \Phi^\delta_{i-1,L} \cdot \cdots \cdot \Phi^\delta_{1,L}(X_0) + \mathbb{S}_i + \mathbb{J}_i \end{aligned} \tag{2}$$

such that $\Phi^\delta_{j,L}(\cdot)$ denotes the linear transformation defined by $\Phi_{j,L}(\cdot, \delta)$, and

$$\begin{aligned} \mathbb{S}_i &= S_i(\vec{z}) + \Phi^\delta_{i,L} \cdot S_{i-1}(\vec{z}) + \cdots + \Phi^\delta_{i,L} \cdot \cdots \cdot \Phi^\delta_{2,L} \cdot S_1(\vec{z}) \\ \mathbb{J}_i &= J_i + \Phi^\delta_{i,L} \cdot J_{i-1} + \cdots + \Phi^\delta_{i,L} \cdot \cdots \cdot \Phi^\delta_{2,L} \cdot J_1 \end{aligned}$$

Therefore, if we are able to represent J_j for $1 \le j \le i$ symbolically, there is no overestimation accumulation in computing

$\mathbb{J}_i$, and R_i can be made smaller than that in the original method. Here, we use support functions as the symbolic representation. We give an algorithm for the above scheme, the input ODE may have time-varying uncertainties. Unlike the original method, in the ith step for $i > 1$, a local initial set *does not* directly take the remainder R'_{i-1} from the previous flowpipe, it computes a *smaller* one R_i symbolically based on $J_1, \ldots, J_i$.

Algorithm 1 Flowpipe construction with symbolic remainders

Input: ODE: $\dot{x} = f(\vec{x})$, initial set $\vec{x}(0) \in X_0$, stepsize $\delta > 0$, TM order $k \geq 1$, step number N

Output: Overapproximation for the reachable set in the time interval $[0, N\delta]$

$\Phi_L := \emptyset$; # queue for $\Phi^{\delta}_{i,L}$
$\mathcal{J} := \emptyset$; # queue for J_i
$\mathcal{R} := \emptyset$;
for $i = 1$ to N **do**
  Compute the Taylor approximation Φ_i;
  Compute a proper remainder I_i for Φ_i;
  Compute the TM $S_i + J_i$ by (1);
  $p_i(\vec{z}, t) + R'_i := \Phi_i(X_{i-1}, t) + I_i$; # the ith TM flowpipe
  $\mathcal{R} := \mathcal{R} \cup \{p_i(\vec{z}, t) + R'_i\}$;
  $R_i := J_i$;
  for $j = 1$ to $i - 1$ **do**
    $\Phi_L[j] := \Phi^{\delta}_{i,L} \cdot \Phi_L[j]$;
  end for
  Add $\Phi^{\delta}_{i,L}$ to the end of Φ_L;
  for $j = 2$ to i **do**
    $R_i := R_i + \Phi_L[j] \cdot \mathcal{J}[j-1]$;
  end for
  Compute $X_i = p_i(\vec{z}, \delta) + R_i$ by (2);
  Add J_i to the end of $\mathcal{J}$;
end for
return $\mathcal{R}$;

Theorem III.1. *The TM flowpipes computed by Algorithm 1 form an overapproximation of the reachable set in the time interval of* $[0, N\delta]$.

Difference from the preconditioning techniques. The preconditioning techniques [29] proposed for TMs have a different purpose. Applying a precondition to a local initial set helps to obtain a small remainder interval for the local flow, that is the the remainder I_i in (1). However, the purpose of our method is to limit the accumulation of overestimation along flowpipe construction. Hence, it can be used in a combination with the existing preconditioning techniques.

Maximum size of the queues for $\Phi^{\delta}_{i,L}$ and J_i. The complexity of Algorithm 1 is quadratic in N which is the number of steps. To reduce it, we introduce $\mathcal{M}$ to be the maximum size of the queues for $\Phi^{\delta}_{i,L}$ and J_i. If the size reaches $\mathcal{M}$ after a step, then the queues will be cleared and the remainder R_i in the next step will be computed nonsymbolically by the standard TM integration method. Hence, the overestimation accumulates, under linear transformation, every $\mathcal{M}$ steps, and the algorithm complexity is quadratic in $\mathcal{M}$ but linear in N. When $\mathcal{M} = 0$, the algorithm coincides with the standard TM integration method, and larger $\mathcal{M}$ gives better accuracy.

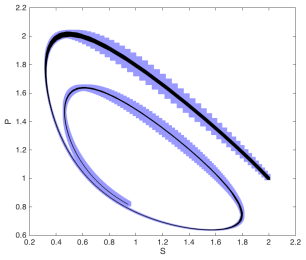


Fig. 2. Flowpipe of the Higgins-Sel'kov Oscillator with time-varying uncertainties.

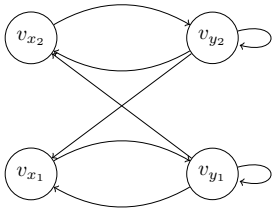


Fig. 3. Variable dependency graph of the coupled Van der Pol system.

We revisit the Higgins-Sel'kov Oscillator in Example II.1. We apply the above algorithm with the stepsize 0.02, TM order 6 and $\mathcal{M} = 400$. The computed TM flowpipes along with the numerical simulations in the time horizon $[0, 10]$ are shown in Figure 2. The time cost is only **18** seconds. The algorithm can also be used with adaptive techniques [11].

IV. System decomposition

Given a nonlinear system with n state variables, a TM flowpipe of it is represented as $(p(\vec{z}, t), R)$ such that $\vec{z}$ represents in the initial set. In the worst case, p could have as many as $\binom{n+k+1}{k}$ terms which make its computation intractable when the dimension n is large. Although we proposed in [9] that a TM can be simplified by moving "small" terms into the remainder part, those terms should still be computed before simplification and that may not essentially improve the scalability.

In this section, we introduce a hybridization method such that in each hybridization domain, the original ODE is overapproximated by a set of "independent" lower-dimensional ODEs with time-varying uncertainties, and then a computed TM flowpipe is essentially a set of lower-dimensional TMs. The decomposed relations are however kept by the ranges of the uncertainties, so that the approximation (hybridization) error can be arbitrarily reduced by the refinement in the dimensions of the decomposed variables. Also the stability of the original system can be preserved. In Section V, we show that such refinement is often not necessary if the decomposition-hybridization approach is applied along with the overestimation reduction scheme presented in the previous section.

A. Dependency graph of variables.

Given an n-dimensional system $\mathcal{S} : \dot{x}_1 = f_1, \ldots, \dot{x}_n = f_n$. We denote $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ the *dependency graph* of the state variables such that $V_{\mathcal{S}}$ consists of n nodes $v_{x_1}, \ldots, v_{x_n}$ each of which is corresponded to a variable, $E_{\mathcal{S}}$ defines the edges such that $(v_{x_i}, v_{x_j}) \in E_{\mathcal{S}}$ iff x_j occurs in f_i.

Example IV.1. *The dynamics of a coupled Van der Pol system is defined by* $\dot{x}_1 = y_1$, $\dot{y}_1 = y_1 - y_1x_1^2 - 2x_1 + x_2$, $\dot{x}_2 = y_2$, $\dot{y}_2 = y_2 - y_2x_2^2 - 2x_2 + x_1$. *The dependency graph of the variables is given in Figure 3.*

B. Decomposition of variable dependency graphs

The complexity of a TM flowpipe can be studied from the variable dependency graph. Proposition IV.1 tells that if the derivative of x_i does not depend on x_j then z_j does not occur in the x_i-dimension of any TM flowpipe. In other words, the single TM in the x_i-dimension of a flowpipe has at most $n-1$ variables. Therefore, if we can decompose the dependency graph into K components which are disconnected from each other, then the system ODE can be accordingly decomposed into K lower-dimensional ODEs which are called a *decomposed ODE (or system)*.

Proposition IV.1. *Given an* n*-dimensional system* $\mathcal{S} : \dot{x}_1 = f_1, \ldots, \dot{x}_n = f_n$. *If there is no path from* v_{x_i} *to* v_{x_j} *in* $G_{\mathcal{S}}$, *then the* x_i*-dimension of any TM flowpipe does not contain* z_j *which represents the range of* $x_j(0)$.

We want to decompose an n-dimensional system $\mathcal{S}$ while breaking as few dependencies as possible. One feasible way is to perform a balanced (K, L) partitioning of its variable dependency graph into K clusters with at most L nodes each, while the cut size is minimized [19]. It can be computed by solving the following integer linear program:

$$\min\{\textstyle\sum_{(v_{x_i},v_{x_j})\in E_{\mathcal{S}}} e_{i,j}\} \quad \text{s.t.}$$
$$\text{(i)} \quad \textstyle\sum_{k=1}^{K} v_{i,k} = 1, \quad \text{for } 1{\le}i{\le}n,$$
$$\text{(ii)} \quad \textstyle\sum_{i=1}^{n} v_{i,k} \le L, \quad \text{for } 1{\le}k{\le}K,$$
$$\text{(iii)} \quad e_{i,j} \ge v_{i,k} - v_{j,k}, \quad \text{for } 1{\le}i,j{\le}n \wedge i{\neq}j \wedge 1{\le}k{\le}K,$$
$$\text{(iv)} \quad v_{i,k}, e_{i,j} \in \{0,1\}, \quad \text{for } 1{\le}i,j{\le}n \wedge 1{\le}k{\le}K.$$

The value of $v_{i,k}$ is 1 when v_{x_i} belongs to the kth cluster, otherwise $v_{i,k} = 0$. The value of $e_{i,j}$ is 1 when the edge (v_{x_i}, v_{x_j}) is in the cut of the partitioning, i.e., (v_{x_i}, v_{x_j}) is removed by decomposition. The property (i) requires that each node can only belong to one cluster. The property (ii) requires that each cluster can only have at most L nodes. The property (iii) tells that $e_{i,j}$ is 1 iff the nodes v_{x_i} and v_{x_j} belong to different clusters. Notice that we only consider the value of $e_{i,j}$ if (v_{x_i}, v_{x_j}) is an edge in $G_{\mathcal{S}}$. Such a problem can be exactly solved by an integer programming tool such as Z3 [14], or approximately solved by greedy algorithms.

Given a system $\mathcal{S} : \dot{x}_1 = f_1, \ldots, \dot{x}_n = f_n$, and the values of K, L. We obtain a cut set which consists of the edges removed in the (K, L) partitioning of the graph $G_{\mathcal{S}}$. Then a decomposed ODE $\dot{x}_1 = g_1, \ldots, \dot{x}_n = g_n$ is computed as follows. For $1 \le i \le n$, g_i is computed from replacing x_j in f_i by u_j for all $1 \le j \le n$ if (v_{x_i}, v_{x_j}) is in the cut set. Then the decomposed ODE can be collectively represented as $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ which is essentially a set of lower-dimensional ODEs with uncertain parameters $\vec{u}$. We call those replaced variables *decomposed variables*.

As an example, the system in Example IV.1 can be decomposed to $\dot{x}_1 = y_1$, $\dot{y}_1 = y_1 - y_1x_1^2 - 2x_1 + u_2$, $\dot{x}_2 = y_2$, $\dot{y}_2 = y_2 - y_2x_2^2 - 2x_2 + u_1$ by removing the edges (v_{y_1}, v_{x_2}) and (v_{y_2}, v_{x_1}). The decomposed variables are x_1, x_2.

C. Hybridization with decomposition

We introduce the method to construct an overapproximate hybrid automaton (or hybridization) for a continuous system based on its decomposition.

Definition IV.1. *An* overapproximate hybrid automaton $\mathcal{A}_D$ *is denoted by a tuple* $\langle \vec{q}, \vec{x}, \vec{u}, g, \mathcal{T}, \mathcal{U}_{\vec{q}}, X_0 \rangle$ *wherein* $\vec{q}$ *are the* N *ordered discrete modes,* $\vec{x}$ *are the* n *state variables,* $\vec{u}$ *are the* n *time-varying uncertainties,* $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ *is the decomposed dynamics such that the range of* $\vec{u}$ *in* q_i *is defined by the* predicate $\mathcal{U}_{q_i}(\vec{u})$, *we also call those ranges* hybridization domains. *The set* $\mathcal{T}$ *consists of time-triggered switches such that there is only one discrete transition from* q_i *to* q_{i+1} *for* $1 \le i \le N-1$, *and it is enabled and must be executed when* $t = i\delta$ *for a stepsize* $\delta > 0$. X_0 *is the initial set. Additionally, we also have* t *as the global timer. Notice that only the uncertainties corresponding to the decomposed variables are constrained by the predicates* $\mathcal{U}_{\vec{q}}$.

An *execution* of $\mathcal{A}_D$ is a *piecewise differentiable* function $\varphi_D(t)$ with $\varphi_D(0) \in X_0$ such that for all $1 \le i \le N$, $\varphi_D(t - (i-1)\delta)$ with $t \in [(i-1)\delta, i\delta]$ is a solution of the ODE $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ w.r.t. $\vec{x}(0) = \varphi_D((i-1)\delta)$ while $\vec{u}(t)$ is a continuous function bounded by the hybridization domain in q_i. We call a state $\vec{v}$, which is a valuation of the variables, *reachable* if there is some $t \in [0, N\delta]$ and $\vec{v} = \varphi_D(t)$.

Given an n-dimensional system $\mathcal{S} : \dot{\vec{x}} = f(\vec{x})$, its decomposition $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ and a stepsize $\delta > 0$, we compute a hybridization $\mathcal{A}_D$ whose reachable set is an overapproximation of the reachable set of $\mathcal{S}$ in $[0, N\delta]$. Starting with the initial set X_0, we iteratively construct a mode with its hybridization domain and compute a flowpipe there.

We assume $x_{n_1}, \ldots, x_{n_\gamma}$ are the decomposed variables. In the ith iteration for $i \ge 1$, assuming that the local initial set is given by X_{i-1}. We do the following steps to construct the hybridization domain $\mathcal{U}_{q_i}$ and compute a flowpipe.

1. Estimate a hybridization domain. We evaluate an interval enclosure $B(X_{i-1})$ for X_{i-1}, and then bloat it to $\overline{B}_{i-1}$ by pushing the upper and lower bounds in all dimensions outward by a distance $\sigma > 0$. Then the hybridization domain $\mathcal{U}_{q_i}$ is estimated to be $\mathcal{U}_{q_i} : \bigwedge_{j=n_1,\ldots,n_\gamma}(a_j \le u_j \le b_j)$ such that for each j, a_j is the lower bound of $B(X_{i-1})[j]$ if the derivative of x_j is positive when $\vec{x} \in \overline{B}_{i-1}$, otherwise a_j is the lower bound of $\overline{B}_{i-1}[j]$; b_j is the upper bound of $B(X_{i-1})[j]$ if the derivative of x_j is negative when $\vec{x} \in \overline{B}_{i-1}$, otherwise it is the upper bound of $\overline{B}_{i-1}[j]$. The signs of the derivatives can be conservatively checked by interval arithmetic.

2. Compute a valid TM flowpipe. We compute a TM flowpipe $(\Phi_i(\vec{y}, t_D), I_i)$ with $\vec{y} \in X_{i-1}$ and $t_D \in [0, \delta]$ for the solutions of $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ while $\vec{u}$ are time-varying uncertainties whose range satisfies $\mathcal{U}_{q_i}$. We next validate the TM, that is, we verify that the TM is "entirely" contained in the hybridization domain. Since $u_{n_1}, \ldots, u_{n_\gamma}$ are the overapproximate substitutions of $x_{n_1}, \ldots, x_{n_\gamma}$, if the jth dimension of the

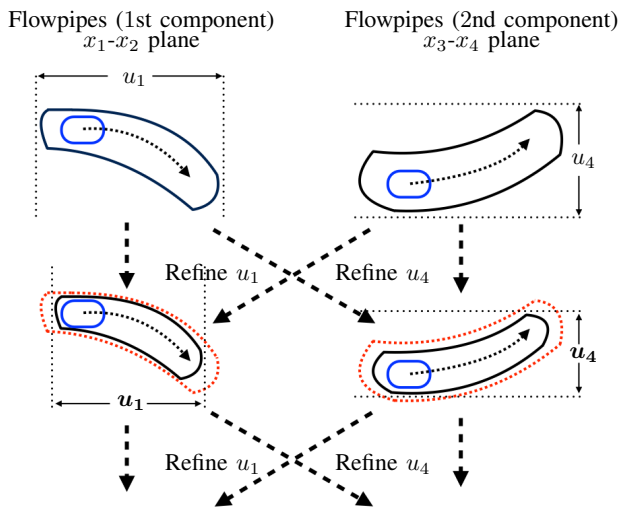


Fig. 4. Mutual refinement for the flowpipes of the components

flowpipe is contained in the range of u_j for $j = n_1, \ldots, n_\gamma$, then the flowpipe is an overapproximation of the original system reachable set. To validate it, for $j = n_1, \ldots, n_\gamma$, if the derivative of x_j is positive (negative resp.), then we only need to ensure that the upper (lower resp.) bound of $(\Phi_i(\vec{y}, t_D), I_i)$ is smaller (larger resp.) than the upper (lower resp.) bound of $\mathcal{U}_{q_i}$ in that dimension. Otherwise we verify both sides. If the TM flowpipe is not validated, we may go to the previous step and use a larger estimation σ.

3. Refine the hybridization domain and the TM flowpipe. We propose a mutual refinement method to reduce the hybridization domain as well as the TM flowpipe. We evaluate the range $\texttt{Rng}(x_j)$ of the flowpipe in the jth dimension for all $j = n_1, \ldots, n_\gamma$. Since the flowpipe is valid, we must have that $\texttt{Rng}(x_j) \subseteq [a_j, b_j]$. Then we reduce $[a_j, b_j]$ to $\texttt{Rng}(x_j)$ in $\mathcal{U}_{q_i}$ for all $j = n_1, \ldots, n_\gamma$, and compute a smaller TM flowpipe based on the contracted hybridization domain. We repeat this step until no big improvement is made. Since the range of an uncertainty is refined by one component and then fed back to some others in each refinement iteration, it can be viewed as that the components are mutually refined.

4. Define the switch and compute the next local initial set. If $i > 1$, we define a switch from q_{i-1} to q_i with the switch condition $t = (i-1)\delta$. If $i < N$, the local initial set for the next time step can be computed by the method presented in Section III.

We give an example to illustrate the mutual refinement method based on a 4-dimensional system decomposed to $(\dot{x}_1, \dot{x}_2) = g_1(x_1, x_2, u_4)$ and $(\dot{x}_3, \dot{x}_4) = g_2(x_3, x_4, u_1)$. The decomposed variables x_1, x_4 are replaced by u_1, u_4 in the components. The refinement of the TM flowpipe in a step is illustrated in Figure 4. The top most flowpipes are guaranteed valid w.r.t. the estimation of hybridization domain, they are the result of Step 2. The local initial set is denoted by blue boundary. We compute the range $\texttt{Rng}(x_1)$ of x_1 and the range $\texttt{Rng}(x_4)$ of x_4 in the first and second component respectively. Since the combined flowpipe is contained in the hybridization domain, those ranges must be contained in the ranges of u_1 and u_4 respectively. We then reduce the hybridization domain to $u_1 \in \texttt{Rng}(x_1)$ and $u_4 \in \texttt{Rng}(x_4)$, then feed each of the ranges to the other component where a refined flowpipe can be computed. We repeat this mutual refinement procedure until no big improvement can be made.

Fixing an order, assuming that the time complexity of computing one n-dimensional flowpipe is $\mathcal{C}_{\text{fp}}(n)$ which is exponential in n in the worst case. If the system is decomposed into K components such that $n_1, \ldots, n_K$ are their dimensions, then the complexity of computing a decomposed flowpipe is at most $\mathcal{C}_{\text{Est-HD}} + \sum_{i=1}^{K} \mathcal{C}_{\text{fp}}(n_i) + \lambda \cdot (\sum_{i=1}^{K} \mathcal{C}_{\text{fp}}(n_i) + \mathcal{C}_{\text{Ref-HD}})$ wherein $\mathcal{C}_{\text{Est-HD}}$ is the cost of estimating a hybridization domain (Step 1), λ is the number of refinement iterations (Step 3), $\mathcal{C}_{\text{Ref-HD}}$ is the cost of refining a hybridization domain. In practice, we may set a constant upper bound for λ, and only contract the remainder of the flowpipe instead of re-computing it in each refinement iteration. Besides, it is also possible to more accurately estimate the hybridization domain based on a Lyapunov function if it can be obtained easily.

Our decomposition-hybridization algorithm *terminates* when either N modes are computed or no validated TM flowpipe could be computed when σ exceeds a user-specified upper bound. Also, adaptive stepsizes could be used here.

Theorem IV.1. *The hybrid automaton computed by the decomposition-hybridization method is an overapproximation of the original system in the time horizon* $[0, N\delta]$. *The TM flowpipes form an overapproximation for the original system reachable set in the time horizon* $[0, N\delta]$.

Proof. For the first statement, it is sufficient to show that the solution φ_f of $\dot{\vec{x}} = f(\vec{x})$ w.r.t. $\vec{x}(0) \in X_{i-1}$ for $1 \le i \le N$ in the time interval $[0, \delta]$ is also a solution of $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ with some continuous functions $\vec{u}$ bounded by $\mathcal{U}_{q_i}$. Assume that the decomposed variables are $x_{n_1}, \ldots, x_{n_\gamma}$. Since all $x_{n_1}, \ldots, x_{n_\gamma}$ are continuous and bounded by $\mathcal{U}_{q_i}$ in the time $[0, \delta]$, we have that φ_f is the solution of $\dot{\vec{x}} = g(\vec{x}, \vec{u})$ with $u_j(t) = x_j(t)$ for $j = n_1, \ldots, n_\gamma$. The second statement can be directly deduced, since the TM flowpipes are overapproximations of the hybrid automaton reachable set. □

D. Hybridization error

We assume that a continuous system $\dot{\vec{x}} = f(\vec{x})$ with the initial set X_0 is decomposed and hybridized by $\langle \vec{q}, \vec{x}, \vec{u}, g, \mathcal{T}, \mathcal{U}_{\vec{q}}, X_0 \rangle$, then the dynamics error is bounded by

$$\mathcal{E}_D = \sup_{1 \le i \le N} \{\sup\{\|f(\vec{x}) - g(\vec{x}, \vec{u})\| \mid \vec{x} \in \mathcal{F}_i, \vec{u} \in \mathcal{U}_{q_i}\}\}$$
$$\le \sup_{1 \le i \le N} \{\sup\{\|g(\vec{x}, \vec{u}_1) - g(\vec{x}, \vec{u}_2)\| \mid \vec{x} \in \mathcal{F}_i, \vec{u}_1, \vec{u}_2 \in \mathcal{U}_{q_i}\}\}$$

such that $\mathcal{F}_i$ denotes the ith flowpipe. More intuitively, $\mathcal{E}_D$ gives the maximum difference of the derivatives of the original and the decomposed systems in the domains $\mathcal{F}_1, \ldots, \mathcal{F}_N$. Notice that the error bound can be arbitrarily improved by refining

$\mathcal{U}_{q_1}, \ldots, \mathcal{U}_{q_N}$, and only the dimensions of the decomposed variables are involved which is unlike the other hybridization methods that often need to consider all dimensions of the state space. Our hybridization error is given as below. It is similar to the form given in [3] except that only decomposed dimensions are involved.

Theorem IV.2 (Hybridization error)**.** *For any solution $\vec{x}_1(t)$ of $\dot{\vec{x}} = f(\vec{x})$ and any execution $\vec{x}_2(t)$ of $\langle \vec{q}, \vec{x}, \vec{u}, g, \mathcal{T}, \mathcal{U}_{\vec{q}}, X_0 \rangle$ with $\vec{x}_1(0) = \vec{x}_2(0) \in X_0$, we have that*

$$\|\vec{x}_1(t) - \vec{x}_2(t)\| \leq \frac{\mathcal{E}_D}{L_f}(e^{L_f t} - 1), \text{ for all } t \in [0, N\delta] \quad (3)$$

such that L_f is the Lipschitz constant of f in $\bigcup_{i=1}^{N} \mathcal{F}_i$.

Proof. Given two piecewise differentiable functions $\vec{z}_1(t)$ and $\vec{z}_2(t)$ over $t \in [0, N\delta]$, such that for all $t \in [0, N\delta]$, $\vec{z}_1(t), \vec{z}_2(t) \in \bigcup_{i=1}^{N} \mathcal{F}_i$, $\|f(\vec{z}_1(t)) - \dot{\vec{z}}_1(t)\| \leq \epsilon_1$ wherein $\vec{z}_1$ is differentiable, and $\|f(\vec{z}_2(t)) - \dot{\vec{z}}_2(t)\| \leq \epsilon_2$ wherein $\vec{z}_2$ is differentiable. By *fundamental inequality* [22], we have that

$$\|\vec{z}_1(t) - \vec{z}_2(t)\| \leq \|\vec{z}_1(0) - \vec{z}_2(0)\| e^{L_f t} + \frac{\epsilon_1 + \epsilon_2}{L_f}(e^{L_f t} - 1)$$

for all $t \in [0, N\delta]$. Therefore, we set $\vec{z}_1 = \vec{x}_1$, $\vec{z}_2 = \vec{x}_2$, and then the inequality (3) is proved. □

Based on the above theorem, we also have that if the solutions of the original system reach a region A, which could be an attractor, in the time $[0, N\delta]$, then for any $\epsilon > 0$, if we set $\mathcal{E}_D = \frac{\epsilon L_f}{e^{L_f N\delta} - 1}$, all executions of the decomposed system reach the neighborhood $\{\vec{x} \,|\, \exists \vec{a} \in A, \|\vec{x} - \vec{a}\| \leq \epsilon\}$ of A. It tells that at least the bounded time stability of the original system is preserved by our method.

Although smaller hybridization domains could be computed using smaller stepsizes, the size of a hybridization domain is bounded below by the size of the local initial set in that mode, and it can not be arbitrarily refined without performing a subdivision. However, we will show by experiments that the overall hybridization error can still be well controlled if we use the symbolic remainder representation scheme presented in Section III.

E. Running example

We apply our decomposition-hybridization method on the following example $\dot{x} = x + 2y + 0.1y^2$, $\dot{y} = -6x - 10y - 0.1x^2$. We decomposed the two dimensions, and consider the initial set $x(0) \in [0.9, 1.1]$, $y(0) \in [0.9, 1.1]$. We choose the stepsize $\delta = 0.001$, the TM order 4, $\mathcal{M} = 200$ and $\sigma = 0.02$. The decomposed dynamics is $\dot{x} = x + 2u_y + 0.1u_y^2$, $\dot{y} = -6u_x - 10y - 0.1u_x^2$ wherein u_x, u_y are the replacements of x and y respectively. Their ranges in each step denotes the value ranges of x, y there and define a hybridization domain. By our method, the first hybridization domain is computed as $\mathcal{U}_{q_1} : u_x \in [0.899662, 1.103422] \wedge u_y \in [0.883274, 1.101697]$. After 1000 steps, the hybridization domain $\mathcal{U}_{q_{1000}}$ is $u_x \in [1.001221, 1.230034] \wedge u_y \in [-0.768629, -0.623624]$. When we reach q_{4000}, the hybridization domain is computed as $u_x \in [0.520089, 0.662552] \wedge u_y \in [-0.410163, -0.322654]$. In Figure 5, we show all computed flowpipe overapproximations in the time horizon $[0, 5]$.

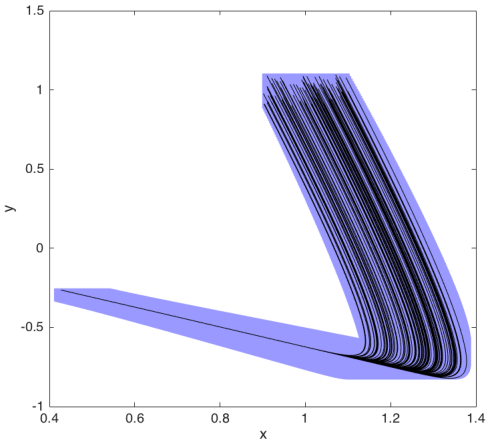


Fig. 5. TM flowpipes for the running example in the time horizon $[0, 5]$.

We can see that although the decomposed dynamics for x shows instability without the input u_y (since the eigenvalue is 1 which is positive), it is actually stabilized by u_y during hybridization, since the original system is stable and u_y is a piecewise interval overapproximation of y.

V. Experiments

We implemented a prototype tool based on the computation library of Flow*. The performance of our decomposition-hybridization method is evaluated by both *offline* and *online* flowpipe construction.

A. Offline flowpipe construction

We construct 14 challenging tests based on 10 benchmarks for evaluating both efficiency and accuracy of the methods to compute flowpipe overapproximations for nonlinear systems. The efficiency of each test is reflected by the running time in seconds, while the accuracy is assessed by checking whether the overapproximation set at a time T exceeds a specified target set or not. If it does not, then the accuracy requirement is fulfilled. Otherwise the result is too coarse.

Experimental setting. The experiments are designed as follows. For each test, we specify an initial set as well as a target set with a time T to a system model. We try to prove, using each tool, that all reachable set at time T is contained in the target set, that is the property $\forall \vec{x}_0 \in X_0.(\varphi_f(\vec{x}_0, T) \in \mathcal{S})$ wherein X_0 is the initial set and $\mathcal{S}$ is the target set. Notice that the set $\mathcal{S}$ could be a real-time property that some event must happen at T for all system executions. In our tests, T is set to be the end of the time horizon, since overestimation eventually accumulates during validated integration for nonlinear ODEs (see [29]), and it is reasonable to assess the overall accuracy just at the last flowpipe. The refinement iteration number λ is bounded by 50 in all tests. Our platform is a laptop equipped with an Intel i7 CPU and 16GB memory. A summary of the

experimental results is given by Table II. More explanations are given as follows.

The tools considered by us are Flow*, VNODE-LP, CAPD, dReach, NLTOOLBOX, C2E2 and HyCreate. The comparison with a more user-friendly version of CORA will be our future work.

Tuning the setting for a tool. We briefly outline the process of obtaining parameter settings for each tool we compared against. For each benchmark instance and tool, we attempt to prove the given property with the computationally cheapest setup for the tool. If this fails, we consider more expensive settings of the parameter values by trial and error until either (a) the property is proved or (b) the tool does not terminate in 1 hour. If no successful setting can be found in this manner, we subdivide the initial set into D intervals along each system dimension. Such a subdivision simplifies the reachability problem. We repeat the parameter selection process, attempting to prove the reachability for each piece. In Table II, a $D_{\min}$ cell gives the subdivision size and also denotes that the tool fails to prove the reachability property on a D-subdivision of the initial set for any $1 \leq D < D_{\min}$.

We were unable to use dReach, NLTOOLBOX, C2E2 or HyCreate to successfully tackle any of our benchmarks within the given running time limit of 1 hour. Since dReach and C2E2 are built on top of the CAPD solver, we directly compare our method with CAPD. In all of the tests, we let VNODE-LP automatically select stepsizes and orders, and let CAPD automatically select stepsizes with fixed order 20 which gives the best numerical stability in our experience.

The system models along with the definitions of initial and target sets are described as below.

Laub-Loomis model. The Laub-Loomis model described in [25], [38] has 7 variables, the dynamics is given by

$$\begin{cases} \dot{x}_1 &= 1.4x_3 - 0.9x_1 \\ \dot{x}_2 &= 2.5x_5 - 1.5x_2 \\ \dot{x}_3 &= 0.6x_7 - 0.8x_2x_3 \\ \dot{x}_4 &= 2 - 1.3x_3x_4 \\ \dot{x}_5 &= 0.7x_1 - x_4x_5 \\ \dot{x}_6 &= 0.3x_1 - 3.1x_6 \\ \dot{x}_7 &= 1.8x_6 - 1.5x_2x_7 \end{cases}$$

We consider the initial set that is a box of width W_0 (along all dimensions) centered at $x_1(0)$=1.2, $x_2(0)$=1.05, $x_3(0)$=1.5, $x_4(0)$=2.4, $x_5(0)$=1, $x_6(0)$=0.1, $x_7(0)$=0.45, as suggested in [38]. We define target sets $\mathcal{S}_{0.02} : x_5 \in [0.265, 0.275] \wedge x_7 \in [0.316, 0.326]$, $\mathcal{S}_{0.1} : x_5 \in [0.255, 0.285] \wedge x_7 \in [0.305, 0.335]$ and $\mathcal{S}_{0.2} : x_5 \in [0.23, 0.31] \wedge x_7 \in [0.29, 0.35]$ for W_0=0.02, 0.1, 0.2 respectively. We require that the computed overapproximation set at $t = 10$ should be entirely contained in the target set. The system is decomposed as $\{x_1, x_4, x_5\}$, $\{x_2, x_3, x_6, x_7\}$ in all tests. For $W_0 = 0.2$, we overlap the flowpipes computed by our method and Flow* in Figure 6. It can be seen that the result generated by the decomposition-hybridization method is only slightly larger, but the time cost is much less.

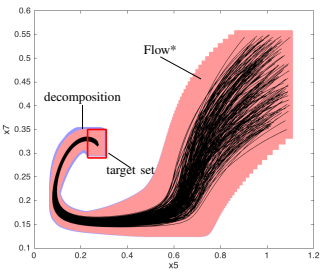


Fig. 6. Flowpipes for the Laub-Loomis model ($W_0 = 0.2$). Numerical simulations are in black.

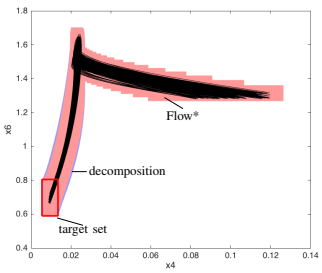


Fig. 7. Flowpipes for the genetic model ($W_0 = 0.04$). Numerical simulations are in black.

Genetic model. We consider the genetic model described in [39]. It is a 9-dimensional system. We adapt some of the constant parameters and derive the dynamics

$$\begin{cases} \dot{x}_1 = 50x_3 - 0.1x_1x_6 \\ \dot{x}_2 = 100x_4 - x_1x_2 \\ \dot{x}_3 = 0.1x_1x_6 - 50x_3 \\ \dot{x}_4 = x_2x_6 - 100x_4 \\ \dot{x}_5 = 5x_3 + 0.5x_1 - 10x_5 \\ \dot{x}_6 = 50x_5 + 50x_3 + 100x_4 - x_6 \cdot (0.1x_1 + x_2 + 2x_8 + 1) \\ \dot{x}_7 = 50x_4 + 0.01x_2 - 0.5x_7 \\ \dot{x}_8 = 0.5x_7 - 2x_6x_8 + x_9 - 0.2x_8 \\ \dot{x}_9 = 2x_6x_8 - x_9 \end{cases}$$

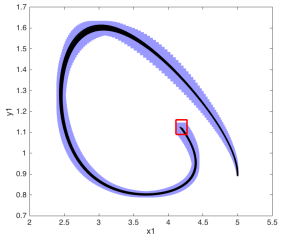
The initial set under our consideration is a box of width W_0 centered at $x_1(0)$=1, $x_2(0)$=1.3, $x_3(0)$=0.1, $x_4(0)$=0.1, $x_5(0)$=0.1, $x_6(0)$=1.3, $x_7(0)$=2.5, $x_8(0)$=0.6, $x_9(0)$=1.3. We consider W_0=0.01, 0.02 and 0.04. We define the target sets $\mathcal{S}_{0.01} : x_4 \in [0.0089, 0.0102] \wedge x_6 \in [0.676, 0.717]$, $\mathcal{S}_{0.02} : x_4 \in [0.0081, 0.0111] \wedge x_6 \in [0.653, 0.740]$, $\mathcal{S}_{0.04} : x_4 \in [0.0055, 0.0135] \wedge x_6 \in [0.590, 0.805]$ corresponding to the three values of W_0 from small to large respectively. The system is decomposed as $\{x_1, x_3, x_5\}$, $\{x_2, x_4, x_7\}$, $\{x_6, x_8, x_9\}$. For $W_0 = 0.04$, we overlap the flowpipes computed by the decomposition-hybridization method and Flow* in Figure 7. Similar to the previous benchmark, the result of our method is only slightly larger but the time cost is much less.

Coupling of cells. We consider the model of N coupled cells studied by Wolf and Heinrich [40]. The dynamics of the ith cell is defined by a 2-dimensional ODE $\dot{x}_i = 4 - x_i y_i^2$, $\dot{y}_i = x_i y_i^2 - 3.84y_i - 3.2 \cdot (x_i - C)$ such that C is the extracellular concentration of the coupling metabolite whose derivative is given by $\dot{C} = \frac{0.64}{N} \cdot (\sum_{i=1}^{N} y_i - N \cdot C)$. The whole system consists of $2N + 1$ state variables if there are N cells, and the dependency graph of the variables is fully connected: this means all pairs of variables are connected in the graph. For $1 \leq i \leq N$, we define the initial condition for the ith component as $x_i(0) \in [4.98 + 0.01i, 5 + 0.01i]$, $y_i(0) \in [0.88 + 0.01i, 0.9 + 0.01i]$, such that all components have different behaviors for any scale, and the first component has different behaviors in different scales. We define the target sets $\mathcal{S}_8 : x_1 \in [4.08, 4.24] \wedge y_1 \in [1.06, 1.13]$, $\mathcal{S}_{10} : x_1 \in [4.09, 4.25] \wedge y_1 \in [1.07, 1.14]$, $\mathcal{S}_{12} : x_1 \in [4.10, 4.26] \wedge y_1 \in$

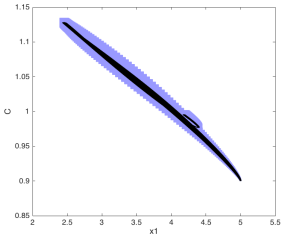
TABLE II

EXPERIMENTAL RESULTS. LEGEND - **VAR**: # OF VARIABLES, **TIME**: TIME COST IN SECONDS TO PROVE THE REACHABILITY PROPERTY, δ: STEPSIZE, k: TM ORDER, σ: BLOATING DISTANCE (SECTION IV), $\mathcal{M}$: QUEUE SIZE LIMIT (SECTION III), **T.O.**: DOES NOT TERMINATE IN 1 HOUR.

				CAPD		VNODE-LP		Flow*			decomposition-hybridization				
#	benchmark	var	T	$D_{\min}$	time	$D_{\min}$	time	δ	k	time	δ	k	σ	$\mathcal{M}$	time
1	Laub-Loomis ($W_0 = 0.02$)	7	10	4	1401	3	67	0.02	4	32	0.02	4	0.1	100	**18**
2	Laub-Loomis ($W_0 = 0.1$)	7	10	10	T.O.	6	T.O.	0.02	4	113	0.02	4	0.1	150	**31**
3	Laub-Loomis ($W_0 = 0.2$)	7	10	20	Fail	10	Fail	0.02	4	557	0.02	4	0.1	200	**87**
4	genetic ($W_0 = 0.01$)	9	2	3	T.O.	2	131	0.002	4	162	0.002	4	0.15	80	**97**
5	genetic ($W_0 = 0.02$)	9	2	4	T.O.	3	T.O.	0.002	4	216	0.002	4	0.15	80	**102**
6	genetic ($W_0 = 0.04$)	9	2	9	T.O.	5	T.O.	0.002	4	560	0.002	4	0.15	80	**133**
7	coupling of cells ($N = 8$)	17	3	10	Fail	4	T.O.	0.001	5	T.O.	0.01	4	0.1	300	**389**
8	coupling of cells ($N = 10$)	21	3	10	Fail	4	T.O.	0.001	5	T.O.	0.01	4	0.1	300	**598**
9	coupling of cells ($N = 12$)	25	3	10	Fail	4	T.O.	0.001	5	T.O.	0.01	4	0.1	300	**1053**
10	coupling of cells ($N = 14$)	29	3	10	Fail	4	T.O.	0.001	5	T.O.	0.01	4	0.05	300	**1775**
11	coupled oscillators ($N = 3$)	15	3	10	Fail	4	T.O.	0.003	4	295	0.01	4	0.05	100	**117**
12	coupled oscillators ($N = 4$)	20	3	10	Fail	5	T.O.	0.001	6	Fail	0.01	4	0.05	150	**195**
13	coupled oscillators ($N = 5$)	25	3	10	Fail	5	T.O.	0.001	6	Fail	0.01	4	0.05	150	**326**
14	coupled oscillators ($N = 6$)	30	3	10	Fail	5	T.O.	0.001	6	Fail	0.01	4	0.05	150	**515**

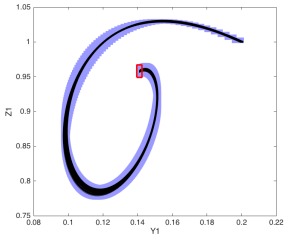


(a) Projection in the x_1-y_1 plane.

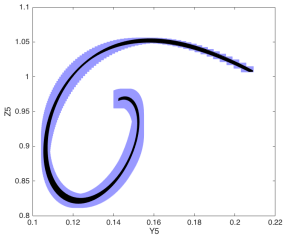


(b) Projection in the x_1-C plane.

Fig. 8. Flowpipes of the coupling cells ($N = 14$). Red box denotes the target set. Numerical simulations are in black.



(a) Projection in the Y_1-Z_1 plane.



(b) Projection in the Y_5-Z_5 plane.

Fig. 9. Flowpipes of the coupled nonidentical oscillators ($N = 6$). Red box denotes the target set. Numerical simulations are in black.

$[1.08, 1.15]$, $\mathcal{S}_{14} : x_1 \in [4.11, 4.27] \wedge y_1 \in [1.09, 1.16]$ for $N = 8, 10, 12, 14$ respectively. Projections of the flowpipe overapproximations for $N = 14$ are shown in Figure 8. In order to make Flow* complete the computation, we have to use the stepsize 0.001 and TM order 5, but it costs more than 1 hour in each test.

Coupled nonidentical oscillators. Another scalable benchmark is the coupled nonidentical oscillator studied in [27]. The model consists N oscillators each of which has 5 state variables after our polynomialization, $\dot{X}_i = 0.1U_i - 3X_i + \frac{10}{N}\sum_{j=1}^{N} V_j$, $\dot{Y}_i = 10X_i - 2.2Y_i$, $\dot{Z}_i = 10Y_i - 1.5Z_i$, $\dot{V}_i = 2X_i - 20V_i$, $\dot{U}_i = -5U_i^2 Z_i^4 (10Y_i - 1.5Z_i)$. Notice that the model is composed in a different way from the previous case study. For $1 \leq i \leq N$, the initial set of the ith component is given by $X_i(0) \in [-0.003 + 0.002i, -0.001 + 0.002i]$, $Y_i(0) \in [0.197 + 0.002i, 0.199 + 0.002i]$, $Z_i(0) \in [0.997 + 0.002i, 0.999 + 0.002i]$, $V_i(0) \in [-0.003 + 0.002i, -0.001 + 0.002i]$, $U_i(0) \in [0.497 + 0.002i, 0.499 + 0.002i]$. Then the components in a scale do not have the same behavior, and the first component behaves differently in different scales. We define the target set $\mathcal{S}_3 : Y_1 \in [0.1395, 0.1425] \wedge Z_1 \in [0.952, 0.970]$, $\mathcal{S}_4 : Y_1 \in [0.1395, 0.1425] \wedge Z_1 \in [0.951, 0.969]$, $\mathcal{S}_5 : Y_1 \in [0.1395, 0.1425] \wedge Z_1 \in [0.950, 0.968]$, $\mathcal{S}_6 : Y_1 \in [0.1395, 0.1425] \wedge Z_1 \in [0.949, 0.967]$ for $N = 3, 4, 5, 6$ respectively. Projections of the flowpipe overapproximations for $N = 6$ are shown in Figure 9. Flow* can prove the safety for $N = 3$ with a longer running time but fails in the other tests.

B. Online Flowpipe Construction

We repeat the experiments over the benchmarks in Table II using an "online" setting for real-time applications. We compute flowpipes for a small time horizon $T < 1$ second from an initial set of states estimated from current sensor measurements. We use an initial set rather than a single initial state to consider uncertainties from the sensor measurement and state estimation.

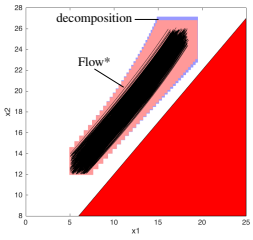
We first motivate the choice of a real time horizon. Each model is based on varying units of time. For instance, for the Laub-Loomis benchmark the time unit is $t = 1$ minute. Therefore, we use a time horizon $T = 0.01$ (minute) corresponding to RT $= 0.6$ seconds in real time. For models where the time unit is 1 hour, we set $T = 0.0002$ (hour) corresponding to RT $= 0.72$ seconds in real time. For a feasible online verification, we require that the flowpipe be constructed within a time $t_f <$ RT, so that the system behavior for a time horizon RT can be estimated in real time. The experimental results for the online verification are shown in Table III. The real-time horizons for each benchmark are reported in Table III under column RT.

Next, we describe the experimental methodology used. We

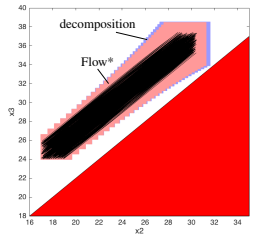
compare our approach against the tools CAPD, VNODE-LP, and Flow*. The desired accuracy of the flowpipe construction affects the time taken to construct it. Therefore, for each test, we adjust the settings so that the tools generate overapproximation sets of comparable accuracy at the end of the time horizon, and then compare the computational time required. The accuracy of an overapproximation is evaluated by its *width*, which is the maximum width of a bounding box in all dimensions. We say that two overapproximations are "similar" (or "comparable"), if their widths are within 10% of each other. We consider the accuracy of Flow* as the baseline, and adjust the subdivision sizes for CAPD and VNODE-LP to provide comparable widths while minimizing their running time. Furthermore, in order to make a more direct comparison between Flow* and our method, we use the same parameter settings for each benchmark.

The decomposition technique presented in this paper is clearly more efficient: our method is able to compute flowpipes within time that is smaller than the real time for all but two of the benchmarks. The original Flow* tool without decomposition takes much longer for comparable accuracy. On the other hand, CAPD and VNODE-LP could not produce any comparably accurate result within 200 seconds for most of the tests.

Vehicle platoon. We apply the decomposition-hybridization method to compute flowpipes for the vehicle platoon model in an online fashion. The initial position of the ith car is given by $x_i \in [c_i, c_i + 2]$, $y_i \in [-0.1, 0.1]$ for all $i \in \{1, 2, 3\}$. The initial velocities of the first and third car are in the range $[20, 21]$ m/s, whereas the second car is initially moving at $[c_4, c_4 + 1]$ m/s. In the beginning, the steering angles for all cars are in the range $[-5°, 5°]$. We consider 100 randomly generated values for c_1, c_2, c_3, c_4 according to $c_1 \in [0, 5]$, $c_2 \in [12, 17]$, $c_3 \in [24, 29]$ and $c_4 \in [19, 21]$, respectively. We evaluate the maximum and minimum time cost of computing the flowpipes for $0.99 < 1$ second in real time. For this time horizon, we wish to verify that the distance in the x-dimension between two consecutive cars is safe: $x_2 - x_1 > 2, x_3 - x_2 > 2$. We compare our method with Flow* based on the stepsize 0.03 and TM order 4. The decomposition-hybridization method requires computation time inside the range $[0.68, 0.81]$ seconds. However, the original Flow* requires computation time in the range $[6.2, 8.2]$ seconds, about 10 times slower. Figure 10 plots the positions of the cars for two instances.



(a) c_1=5, c_2=12, c_3=24, c_4=19



(b) c_1=0, c_2=17, c_3=24, c_4=21

Fig. 10. Flowpipes computed by Flow* and our method for the vehicle platoon model. Numerical simulations are in black, and unsafe set is in red.

VI. Conclusion and Future Work

Thus, we have described an approach to decompose a large monolithic system into smaller ones through hybridization. In doing so, we also provide a solution to tackle the associated problem of dealing with error accumulation. Our experimental evaluation compares our implementation of the decomposition approach with related tools to demonstrate superior performance in terms of time for offline and online verification. In particular, we demonstrate successful treatment of nonlinear systems with up to 30 state variables. Future work will consider better decomposition schemes and the application of this idea to handle larger and more challenging benchmarks.

Acknowledgments: We thank the reviewers for their detailed comments. This work was supported in part by the US National Science Foundation (NSF) under award number CPS-1446900. All opinions expressed are those of the authors and not necessarily of the NSF.

References

[1] M. Althoff. An introduction to cora 2015. In *Proc. of ARCH'15*, volume 34 of *EPiC Series in Computer Science*, pages 120–151. EasyChair, 2015.

[2] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of CDC'08*, pages 4042–4048. IEEE, 2008.

[3] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Inf.*, 43(7):451–476, 2007.

[4] S. Bak, S. Bogomolov, and T. T. Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In *Proc. of HSCC'15*, pages 128–133. ACM, 2015.

[5] S. Bak and M. Caccamo. Computing reachability for nonlinear systems with HyCreate. In *Demo and Poster Session in HSCC'13*, 2013.

[6] S. Bak, A. Greer, and S. Mitra. Hybrid cyberphysical system verification with simplex using discrete abstractions. In *Proc. of RTAS'10*, pages 143–152. IEEE Computer Society, 2010.

[7] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha. Real-time reachability for verified simplex design. In *Proc. of RTSS'14*, pages 138–148. IEEE Computer Society, 2014.

[8] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361–369, 1998.

[9] X. Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.

[10] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of RTSS'12*, pages 183–192. IEEE Computer Society, 2012.

[11] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 258–263. Springer, 2013.

[12] T. Dang, C. Le Guernic, and O. Maler. Computing reachable states for nonlinear biological models. In *Proc. of CMSB'09*, volume 5688 of *LNCS*, pages 126–141. Springer, 2009.

[13] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *Proc. of HSCC'10*, pages 11–20. ACM, 2010.

[14] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proc. of TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[15] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *Proc. of TACAS'15*, volume 9035 of *LNCS*, pages 68–82. Springer, 2015.

[16] A. Eggers, N. Ramdani, N. Nedialkov, and M. Fränzle. Improving sat modulo ode for hybrid systems analysis by combining different enclosure methods. In *Proc. of SEFM'11*, volume 7041 of *LNCS*, pages 172–187. Springer, 2011.

TABLE III

COMPUTING FLOWPIPES IN REAL TIME. LEGEND - **RT**: REAL TIME IN SECONDS, **TIME**: TIME COST IN SECONDS, D: SUBDIVISION SIZE, W_T: WIDTH OF THE REACHABLE SET OVERAPPROXIMATION AT TIME RT, **T.O.**: DOES NOT TERMINATE IN 200 SECONDS, **OTHERS**: SAME AS THOSE IN TABLE II.

			CAPD			VNODE-LP			Flow*				decomposition-hybridization				
benchmark	var	RT	D	time	W_T	D	time	W_T	δ	k	time	W_T	δ	k	σ	time	W_T
Laub-Loomis ($W_0 = 15$)	7	0.6	3	5.65	17.039	3	1.76	17.037	2.5e-3	5	1.22	**16.920**	2e-3	5	0.5	**0.21**	17.028
genetic ($W_0 = 40$)	9	0.72	3	115	42.129	3	41.7	42.127	2.5e-5	3	1.84	**42.107**	2.5e-5	3	0.8	**0.17**	42.117
coupling of cells ($N = 8$, $W_0 = 4$)	17	0.6	2	T.O.	–	2	T.O.	–	2e-3	3	2.49	**5.397**	2e-3	3	0.1	**0.27**	5.566
coupling of cells ($N = 10$, $W_0 = 4$)	21	0.6	2	T.O.	–	2	T.O.	–	2e-3	3	6.11	**5.420**	2e-3	3	0.1	**0.38**	5.592
coupling of cells ($N = 12$, $W_0 = 4$)	25	0.6	2	T.O.	–	2	T.O.	–	2e-3	3	11.5	**5.442**	2e-3	3	0.1	**0.51**	5.618
coupling of cells ($N = 14$, $W_0 = 4$)	29	0.6	2	T.O.	–	2	T.O.	–	2e-3	3	22.1	**5.465**	2e-3	3	0.1	**0.69**	5.644
coupled oscillators ($N = 3$, $W_0 = 1.2$)	15	0.72	2	T.O.	–	2	T.O.	–	4e-5	3	0.81	**1.353**	4e-5	3	0.05	**0.32**	**1.354**
coupled oscillators ($N = 4$, $W_0 = 1.2$)	20	0.72	2	T.O.	–	2	T.O.	–	4e-5	3	1.81	**1.355**	4e-5	3	0.05	**0.49**	**1.355**
coupled oscillators ($N = 5$, $W_0 = 1.2$)	25	0.72	2	T.O.	–	2	T.O.	–	4e-5	3	4.17	**1.356**	4e-5	3	0.05	**0.69**	**1.357**
coupled oscillators ($N = 6$, $W_0 = 1.2$)	30	0.72	2	T.O.	–	2	T.O.	–	4e-5	3	5.02	**1.358**	4e-5	3	0.05	**0.95**	**1.359**

[17] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Proc. of HSCC'05*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.

[18] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.

[19] J. L. Gross and J. Yellen. *Graph theory and its applications, 2nd edition*, volume 29 of *Textbooks in Mathematics*. Chapman and Hall/CRC, 2005.

[20] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: the next generation. In *Proc. of RTSS'95*, pages 56–65. IEEE Computer Society, 1995.

[21] Z. Huang and S. Mitra. Proofs from simulations and modular annotations. In *Proc. of HSCC'14*, pages 183–192. ACM, 2014.

[22] J. H. Hubbard and B. H. West. *Differential Equations: A Dynamical Systems Approach*. Springer, 1991.

[23] T. Kapela, M. Mrozek, P. Pilarczyk, D. Wilczak, and P. Zgliczyński. Capd - a rigorous toolbox for computer assisted proofs in dynamics. Technical report, Jagiellonian University, 2010.

[24] S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *Proc. of TACAS'15*, volume 9035 of *LNCS*, pages 200–205. Springer, 2015.

[25] M. T. Laub and W. F. Loomis. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Molecular Biology of the Cell*, 9:3521–3532, 1998.

[26] C. Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. PhD thesis, Université Joseph Fourier, 2009.

[27] C. Li, L. Chen, and K. Aihara. Synchronization of coupled nonidentical genetic oscillators. *Physical Biology*, 3(1):37, 2006.

[28] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *J. Pure and Applied Mathematics*, 4(4):379–456, 2003.

[29] K. Makino and M. Berz. Suppression of the wrapping effect by taylor model-based verified integrators: Long-term stabilization by preconditioning. *International Journal of Differential Equations and Applications*, 10(4):353–384, 2005.

[30] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.

[31] N. S. Nedialkov. Implementing a rigorous ode solver through literate programming. In A. Rauh and E. Auer, editors, *Modeling, Design, and Simulation of Systems with Uncertainties*, volume 3 of *Mathematical Engineering*, chapter Mathematical Engineering, pages 3–19. Springer Berlin Heidelberg, 2011.

[32] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[33] A. Platzer and E. M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design*, 35(1):98–120, 2009.

[34] P. Prabhakar and M. Viswanathan. A dynamic algorithm for approximate flow computations. In *Proc. of HSCC'11*, pages 133–142. ACM, 2011.

[35] M. Rungger and M. Zamani. Compositional construction of approximate abstractions. In *Proc. of HSCC'15*, pages 68–77. ACM, 2015.

[36] L. Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, 2001.

[37] D. Šiljak. *Large-scale dynamic systems: stability and structure*. North Holland, 1978.

[38] R. Testylier and T. Dang. Nltoolbox: A library for reachability computation of nonlinear dynamical systems. In *Proc. of ATVA'13*, volume 8172 of *LNCS*, pages 469–473. Springer, 2013.

[39] J. M. G. Vilar, H. Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proc. of the National Academy of Sciences of the United States of America*, 99(9):5988–5992, 2002.

[40] J. Wolf and R. Heinrich. Dynamics of two-component biochemical systems in interacting cells; synchronization and desynchronization of oscillations and multiple steady states. *Biosystems*, 43(1):1–24, 1997.