Model Predictive Real-Time Monitoring of Linear Systems

Xin Chen and Sriram Sankaranarayanan
University of Colorado, Boulder, CO **Email:** {xinchen,srirams}@colorado.edu

Abstract—**The predictive monitoring problem asks whether a deployed system is likely to fail over the next T seconds under some environmental conditions. This problem is of the utmost importance in cyber-physical systems and has inspired real-time architectures capable of adapting to such failures upon forewarning. In this paper, we present a linear model-predictive scheme for the real-time monitoring of linear systems governed by time-triggered controllers and time-varying disturbances. The scheme uses a combination of offline (advance) and online computations to decide if a given plant model has entered a state from which no matter what control is applied, the disturbance has a strategy to drive the system to an unsafe region. Our approach is independent of the control strategy used: this allows us to deal with plants that are controlled using model-predictive control techniques or even opaque machine-learning based control algorithms that are hard to reason with using existing reachable set estimation algorithms. Our online computation reuses the symbolic reachable sets computed offline. The real-time monitor instantiates the reachable set with a concrete state estimate, and repeatedly performs emptiness checks with respect to a safety property. We classify the various alarms raised by our approach in terms of what they imply about the system as a whole. We implement our real-time monitoring approach over numerous linear system benchmarks and show that the computation can be performed rapidly in practice. Furthermore, we also examine the alarms reported by our approach and show how some of the alarms can be used to improve the controller.**

I. Introduction

In this paper, we examine the problem of predicting future failures of safety properties within a given time horizon in real-time. The problem of predicting the possibility of a future failure within a time horizon $[0, T]$ is one of the most important problems for Cyber-Physical System (CPS) verification. Doing so in real-time requires us to provide accurate and fast predictions, ideally in time $t \ll T$. The problem setup considers a deployed system from which we receive periodic estimates of the plant state $\mathbf{x}(k\delta)$ and control input $\mathbf{u}(k\delta)$, for time step $\delta > 0$ and $k \in \mathbb{N}$. We will assume that the control inputs are kept constant between two sampling time periods. The disturbances belong to the set $\mathcal{D}$ and control inputs to the set $\mathcal{U}$. Furthermore, we would like the system executions to belong to some safe set $\mathcal{S}$. Thus, the paper considers the following real-time monitoring question:

> Starting from the current state $\mathbf{x}(0)$, is there a (piecewise constant) control input signal $\mathbf{u}(t)$ belonging to $\mathcal{U}$ that can ensure $\mathbf{x}(t) \in \mathcal{S}$ for $t \in [0, T]$, regardless of the disturbances?

If the check fails, we deduce that there are disturbance inputs that can cause the system to be unsafe within the next T seconds. To perform this check, we propose a monitoring algorithm which keeps a "watch list" of reachable set estimates, wherein the i^{th} element of the list overapproximates the possible reachable states for $i\delta$ time ahead into the future, given the information available at current time. At each time step, the monitor checks the safety condition for each reachable set estimate in the list. Upon encountering a new system state, the monitor updates the watch list by removing the head of the list (which corresponds to the current time), and adding a new estimate at the tail (the new end point for our time horizon). In this paper, the reachable state estimates are precomputed offline as reachable state relation and *concretized* in real-time using the most recent state estimate.

A reachable set estimate is represented by a linear expression over the variables that denote the control inputs in the future steps, along with a *zonotope* [34] which accounts for all the possible disturbances. When the monitor progresses by one time step, some of the variables in this expression will be replaced by the new state value and the newly applied control input. Checking for violations involves finding control inputs such that the reachable states are always safe no matter what disturbances from the zonotope are used.

Essentially, the quantifier alternation between the control and disturbance provides a game theoretic interpretation of the monitor, by viewing the maintenance of the safety property as a game between the control and the disturbance, wherein the control is oblivious of the disturbance, but the disturbance can react to the control in an adversarial fashion. In effect, our approach asks if the given prediction is a winning state for the control player.

In this paper, we show that assuming that the initial states and control inputs belong to intervals (or boxes), the set of reachable states also form a zonotope. Furthermore, we resolve the quantifier alternation between the control and disturbance inputs by using a Minkowski difference between the safe set $\mathcal{S}$ and the disturbance zonotope. We show that this difference is efficiently computable, whenever $\mathcal{S}$ belongs to a commonly used type of specification such as a box, halfspace or a strip. The real-time computation reduces to concretizing the symbolic reachable state estimation at each step, updating the disturbance contributions and verifying the nonemptiness of intersections between a zonotope and the updated safe set. This can be solved using a Linear Programming (LP) solver for the general case. However, when the safety set is given as a halfspace or strip, another efficient method using elementary matrix operations can also be used.

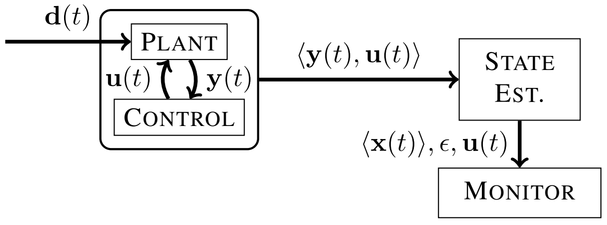
The problem of online monitoring has received increasing attention due to Simplex architectures that are based on real-time computation of a safety envelope that triggers the switch from a high-performance (possibly unsafe) controller to a lower performance (guaranteed safe) controller [32].

We now differentiate our approach from those previously used to solve the real-time safety monitoring problem. The original approach proposed by Seto et al. considers a linear system $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ without disturbances, but with saturation [31], [30]. The approach uses linear matrix inequalities (LMIs) to compute a safe feedback law K and an associated region of attraction in the form of an ellipsoid $\mathcal{E}$ [9]. As long as the state of the system lies (well) within this ellipsoid, the complex controller is used. However, if the system dynamics are about to exit this set, the safety controller is triggered. This approach is improved considerably by using set-based reachability analysis computations that estimate the set of all reachable states in real-time, assuming that the system behaves as a linear hybrid system [6], [7], [23]. The real-time reachability computation problem was explored by Chen et al. for nonlinear systems using a decomposition approach [13]. A real-time reachable set estimate is used to check if the complex controller allowed to execute for the next time step would reach a state that can still allow the system to be recovered by the safe controller, if necessary.

This paper considers a different monitoring problem that is complementary to the previously mentioned ideas. We observe that the increasing use of model-predictive controllers (MPC) and real-time planning in the loop makes offline or online reachability computations involving these controllers quite hard. Thus, rather than reason about the full closed loop involving the plant and a complex controller, we reason just about the plant and the current state $\mathbf{x}(t)$ reached at some time t. Our reasoning incorporates the disturbance model and looks ahead over a small time horizon to check if *some control strategy* is available to keep the system safe, no matter what the disturbance does. Note that the control inputs are oblivious to the disturbance since disturbances are rarely measurable in a direct manner. Rather the control is assumed to be based on a measured/estimated state of the system $\mathbf{x}(t)$. The main advantages of our approach include:

1) It reasons about time-varying disturbance inputs over disturbance ranges.
2) It incorporates actuation limits.
3) Our approach can be used to implement a control scheme that can potentially transition the system into the provably safe operating range of a safe controller.

However, a drawback is that we reason about whether there exists some control input that can maintain the safety property rather than whether the specific control law in use can keep the system safe. This can be addressed by "abstracting" the controller's behaviors to yield constraints over the set of control strategies considered by our approach.



$\mathbf{d}(t)$
PLANT
$\mathbf{u}(t)$
$\mathbf{y}(t)$
CONTROL
$\langle \mathbf{y}(t), \mathbf{u}(t) \rangle$
STATE EST.
$\langle \mathbf{x}(t) \rangle, \epsilon, \mathbf{u}(t)$
MONITOR

Fig. 1. Overall monitoring framework.

Our approach is closely related to robust model-predictive control techniques that consider disturbances to calculate control inputs to minimize a cost function over a time horizon [8]. A key difference lies in the cost function used. At current time t, we focus simply on the feasibility question of whether a control strategy remains available to the system in order to maintain a safety property at time $t + T$. In particular, we do not consider a cost function explicitly. The difference in formulation also explains why our approach is a monitoring rather than a MPC approach. Often, controllers are designed for numerous competing objectives that include performance as well as safety. The focus of our monitoring task is however just on a given safety property.

II. Problem Statement and Approach

In this section, we present an overview of the problem setup and the approach at a high level. Figure 1 shows the overall monitoring framework.

Plant and Control Setup. The system is made up of a plant and a controller that are assumed to be subject to disturbance $\mathbf{d}(t)$. The output $\mathbf{y}(t)$ of the plant is processed using a state estimation technique to yield an accurate state estimate $\mathbf{x}(t)$ with known error bounds $[-\epsilon, +\epsilon]$. The control inputs are assumed to be within known actuation limits inside the interval $\mathcal{U}$. Furthermore, we assume that the system operates with a time period $\delta > 0$, so that for $t \in [(i-1)\delta, i\delta)$ wherein $i \geq 1$, the control $\mathbf{u}(t)$ is kept constant. However, the disturbance $\mathbf{d}(t)$ is unknown and time-varying, but remains within some set $\mathbf{d}(t) \in \mathcal{D}$. We also assume that $\mathcal{D}$ is an interval (or a box).

Monitoring Setup. Let $\mathcal{S}$ be a safety property that we are interested in checking over the time horizon $[(i-1)\delta, (i-1+N)\delta]$, wherein the current time is $t = (i-1)\delta$ and $N \geq 1$ is an integer *lookahead value* specified by the user. We assume that $\mathcal{S}$ can be provided as a box, a halfspace or a strip which will be defined in the next section. Other specifications such as ellipsoids can be handled by simple extensions of our approach but are not considered here.

Definition II.1 (Controllable State). *A state $\mathbf{x}(t_0)$ is said to be* controllable *with respect to the property S and step number N if and only if there exists a control input sequence $\mathbf{c}_1\mathbf{c}_2 \cdots \mathbf{c}_N \in \mathcal{U}^N$ such that for all disturbances $\mathbf{d}(t) \in \mathcal{D}$ over $t \in [t_0, t_0 + N\delta]$, the reachable state $\mathbf{x}(t_0 + N\delta)$ satisfies the safety property. Failing this, we say a state is* uncontrollable.

The monitoring problem asks whether a given state is controllable. The rest of the paper is organized as follows. The notations and basic definitions used in the paper are described in Section III. The real-time monitoring framework along with the algorithms will be presented in Section IV. Section V provides numerous tests to evaluate the performance of our approach.

III. Preliminaries

We denote $\mathbb{R}$ the set of real numbers. Given a set of ordered variables $x_1, \ldots, x_n$, we collectively denote them by $\mathbf{x}$. Similarly, for a (column) vector $\mathbf{x}$, we use x_i to denote its ith component. The transpose of $\mathbf{x}$ is written as $\mathbf{x}^T$. We always use $\dot{x}$ to denote the time derivative dx/dt of x.

We assume that the arithmetic operations such as addition, subtraction, multiplication and division of floating point numbers have $O(1)$ complexity. Furthermore, standard algorithms are used for matrix computation. The complexity of computing $A+B$ for $A, B \in \mathbb{R}^{m\times n}$ is $O(mn)$, and the complexity of computing AB for $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times k}$ is $O(mnk)$. Although the preliminaries and our methods are presented with real-valued matrices, represented as floating point numbers, the result of the paper continues to hold over interval matrices as well.

A. Linear systems

A *linear system* with n state variables $\mathbf{x}$, m control inputs $\mathbf{u}$ and k disturbances $\mathbf{d}$ can be defined by a linear ODE of the form

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + C\mathbf{d}$$

such that $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{n\times k}$.

Given an initial state $\mathbf{x}(0) = \mathbf{x}_0$, the *reachable state* at a time $t > 0$ can be computed by the solution of the ODE:

$$\mathbf{x}(t) = e^{At}\mathbf{x}_0 + \int_0^t e^{A(t-s)}B\mathbf{u}(s)ds + \int_0^t e^{A(t-s)}C\mathbf{d}(s)ds$$

if the control inputs as well as the disturbances are given explicitly.

We use $\mathcal{R}(\mathbf{x}_0, t, \mathbf{u}, \mathbf{d})$ to denote the reachable state from $\mathbf{x}_0$ at the time t with respect to the control inputs $\mathbf{u}$ and disturbances $\mathbf{d}$. Moreover, we also collectively denote the reachable set $\{\mathcal{R}(\mathbf{x}_0, t, \mathbf{u}, \mathbf{d}) \,|\, \mathbf{x} \in X_0, \mathbf{u} \in \mathcal{U}, \mathbf{d} \in \mathcal{D}\}$ by $\mathcal{R}(X_0, t, \mathcal{U}, \mathcal{D})$.

B. Reachable set computation

We review the method to compute reachable sets for linear systems using geometric representations. The disturbances $\mathbf{d}$ are assumed to be uncertain, time-varying, and belonging to a compact (closed and bounded) set $\mathcal{D}$, while the control inputs $\mathbf{u}$ are assumed to be piecewise constant (PWC) and belong to a compact set $\mathcal{U}$. Hence, given an initial set $X_0 \subseteq \mathbb{R}^n$, and PWC control inputs over time intervals of size $\delta > 0$,

$$\mathbf{u}(t) = \mathbf{c}_i \text{ when } t \in [(i-1)\delta, i\delta] \text{ for } i = 1, 2, \cdots, N\,,$$

the reachable set $R(X_0, N\delta, \mathcal{U}, \mathcal{D})$ can be computed using on the following formula

$$e^{AN\delta}X_0 \oplus \int_0^{\delta} e^{A(N\delta-s)}Bds\,\{\mathbf{c}_1\} \oplus \cdots \oplus \int_{(N-1)\delta}^{N\delta} e^{A(N\delta-s)}Bds\,\{\mathbf{c}_N\} \oplus \int_0^{\delta} e^{A(N\delta-s)}C\mathcal{D}ds \oplus \cdots \oplus \int_{(N-1)\delta}^{N\delta} e^{A(N\delta-s)}C\mathcal{D}ds$$

wherein $\oplus$ is the Minkowski sum operator which is defined by $X \oplus Y = \{x + y \,|\, x \in X, y \in Y\}$.

If we denote $\Phi = e^{A\delta}$, $\Psi = \int_0^{\delta} e^{A(\delta-s)}Bds$ and $S_{\mathcal{D}} = \int_0^{\delta} e^{A(\delta-s)}C\mathcal{D}ds$, the above formula can be rewritten as

$$\Phi^N X_0 \oplus \bigoplus_{i=0}^{N-1}(\Phi^i\Psi\{\mathbf{c}_{N-i}\}) \oplus \bigoplus_{i=0}^{N-1}(\Phi^i S_{\mathcal{D}}) \tag{1}$$

since for all $1 \le i \le N$, we have that

$$\int_{(i-1)\delta}^{i\delta} e^{A(N\delta-s)}Bds = e^{(N-i)A\delta}\int_0^{\delta} e^{A(\delta-s)}Bds, \text{ and}$$
$$\int_{(i-1)\delta}^{i\delta} e^{A(N\delta-s)}C\mathcal{D}ds = e^{(N-i)A\delta}\int_0^{\delta} e^{A(\delta-s)}C\mathcal{D}ds$$

When the initial set or the PWC control inputs are not explicitly given, we are still able to compute the reachable set symbolically based on (1) by representing X_0 or $\mathbf{c}_i$ as variables. In the paper, we also call it a *symbolic reachable set*.

Example III.1. *We consider a simple linear system defined by $\dot{x} = -x + u + d$ such that $d = 0$. If the control input u is PWC with the step size 0.02, then the symbolic reachable set at $t = 0.2$ can be computed as*

$$x(0.2) = e^{-0.2}x_0 + \sum_{i=0}^{9}(e^{-0.02i}(1 - e^{-0.02})u_{9-i})$$

wherein x_0 is the variable standing for the initial state, and $u_1, \ldots, u_{10}$ are the variables standing for the control inputs in the 10 time steps respectively.

C. Geometric representations

The techniques presented in this paper rely on geometric objects such as convex polyhedra and zonotopes. We provide a brief introduction to these objects and their manipulations.

Polyhedron. A *halfspace* S in the n-dimensional Euclidean space $\mathbb{R}^n$ is defined by the set satisfying a linear inequality $\mathbf{a}^T\mathbf{x} \le b$, i.e., $S = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{a}^T\mathbf{x} \le b\}$. A *polyhedron* is defined by an intersection of finitely many halfspaces, in other words, it is the set that satisfying all inequalities defining those halfspaces. For example, the polyhedron in Figure 2 can be defined by the intersection of the halfspaces $S_1 = \{(x, y)^T \in \mathbb{R}^2 \,|\, x + y \le 1\}$, $S_2 = \{(x, y)^T \in \mathbb{R}^2 \,|\, -x \le 0\}$ and $S_2 = \{(x, y)^T \in \mathbb{R}^2 \,|\, -y \le 0\}$. Notice that a polyhedron is not necessarily bounded.

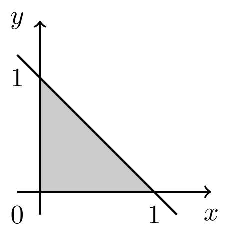


Fig. 2. Example of a polyhedron

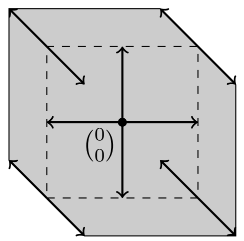


Fig. 3. Example of a zonotope

Box. A (closed) *box* can be simply defined by specifying the range in each dimension. It can be unbounded. For example, $\{(x,y)^T \mid x \geq 6, y \leq 8\}$ defines a 2-dimensional box whose range in the x-dimension is bounded below by 6 while the range in the y-dimension is bounded up by 8. A *unit box* is a box whose range in each dimension is defined by $[-1,1]$.

In the rest of the paper, we assume that the control input set $\mathcal{U}$ as well as the disturbance set $\mathcal{D}$ are both bounded boxes.

Zonotope. Zonotopes are a subclass of symmetric bounded polyhedra [34]. Generally, a zonotope can be viewed as the image of a unit box under an affine mapping. Here, we follow the generator-based definition used by Girard [18].

Definition III.1. *An n-dimensional* zonotope *Z is defined by*

$$Z = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{x} = \mathbf{c} + \sum_{i=1}^{p} \lambda_i \mathbf{g}_i, \lambda_i \in [-1,1] \right\}$$

such that $\mathbf{c} \in \mathbb{R}^n$ is the center, *and $\mathbf{g}_1, \ldots, \mathbf{g}_p \in \mathbb{R}^n$ are the* generators. *We denote Z as $(\mathbf{c}, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$.*

Intuitively, a zonotope $Z = (\mathbf{c}, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$ is the Minkowski sum of $\mathbf{c}$ and the line segments defined by the generators, that is, the line segment defined by $\mathbf{g}_i$ is $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \lambda_i \mathbf{g}_i, \lambda_i \in [-1,1]\}$.

Figure 3 shows an example. The center of the zonotope is the origin $(0,0)^T$, and the generators are $(1,0)^T$, $(0,1)^T$ and $(0.5,0.5)^T$. So, the range of the zonotope can viewed as the result from consecutively bloating the set from the center in both of the positive and negative directions of a generator. Notice that a bounded box is a special zonotope.

Zonotopes are closed under linear transformations and Minkowski sums. Algorithms for the image of a zonotope under linear transformations and Minkowski sums are simple and quite efficient on the generator-based representations.

Given a zonotope $Z = (\mathbf{c}, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$ and a linear map defined by $\mathbf{x}' := M\mathbf{x}$ for a matrix M, the image is also a zonotope which is defined by

$$MZ = (M\mathbf{c}, \langle M\mathbf{g}_1, \ldots, M\mathbf{g}_p \rangle).$$

Given two zonotopes $Z_1 = (\mathbf{c}_1, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$ and $Z_2 = (\mathbf{c}_2, \langle \mathbf{h}_1, \ldots, \mathbf{h}_q \rangle)$. The Minkowski sum is the zonotope defined by the sum of the centers and the union of the generators:

$$Z_1 \oplus Z_2 = (\mathbf{c}_1 + \mathbf{c}_2, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p, \mathbf{h}_1, \ldots, \mathbf{h}_q \rangle).$$

The computation of the linear transformation defined by a matrix $M \in \mathbb{R}^{n' \times n}$ on an n-dimensional zonotope with p generators requires $p+1$ matrix multiplications of the type $M_1 M_2$ such that $M_1 \in \mathbb{R}^{n' \times n}$ and $M_2 \in \mathbb{R}^{n \times 1}$. If we consider the computational complexity of the arithmetic on reals (represented by floating point numbers in the paper) as $O(1)$, the cost of linear transformation on a zonotope is $O((p+1)n'n)$. For computing the Minkowski sum of two n-dimensional zonotopes with p and q generators respectively, it only requires to compute the summation of the center and concatenate the generator lists. Hence, the computation needs one addition of two $n \times 1$ matrices, and then the complexity is $O(n)$.

TABLE I
Computational complexity for computing the terms in $\hat{\mathcal{R}}_N$.

Term	$O(n)$	$O(n^2)$	$O(n^2m)$	$O(n^3)$
Φ_i	0	0	0	$N-1$
$\Phi^i\Psi$	0	0	$N-1$	0
$\bigoplus_{i=0}^{N-1}(\Phi^i B_{\mathcal{D}})$	$N-1$	$(N-1)(n+1)$	0	0

Now we turn to the method for reachable set computation using zonotopes proposed by Girard et al. [18], [20]. The disturbance term $\bigoplus_{i=0}^{N-1}(\Phi^i S_{\mathcal{D}})$ in (1) can be overapproximated by a zonotope $\bigoplus_{i=0}^{N-1}(\Phi^i B_{\mathcal{D}})$ such that $B_{\mathcal{D}}$ is a box enclosure of the one-step disturbance $S_{\mathcal{D}} = \int_0^\delta e^{A(\delta - s)} C\mathcal{D} ds$. It can be evaluated by the zonotope method in [18] or the interval arithmetic [26].

Then the reachable set (1) be overapproximated by

$$R_N = \Phi^N X_0 \oplus \bigoplus_{i=0}^{N-1}(\Phi^i \Psi \{\mathbf{c}_{N-i}\}) \oplus \bigoplus_{i=0}^{N-1}(\Phi^i B_{\mathcal{D}}). \qquad (2)$$

Again, if we want to keep the above set symbolically, the sets X_0, $\{\mathbf{c}_1\}, \ldots, \{\mathbf{c}_N\}$ can be represented by the variables $\mathbf{x}_0$, $\mathbf{u}_1, \ldots, \mathbf{u}_N$ respectively.

We investigate the complexity of computing R_N. Given that Φ, Ψ and $B_{\mathcal{D}}$ are computed in advance with a complexity $\mathcal{C}_{\text{Adv}}$. We may first compute Φ^i for $1 \leq i \leq N$ and keep the results in a hash table. That requires $N-1$ multiplications of two $n \times n$ matrices. Then we compute $\Phi^i \Psi$ for $1 \leq i \leq N-1$ and it requires $N-1$ matrix multiplications of the type $M_1 M_2$ such that $M_1 \in \mathbb{R}^{n \times n}$ and $M_2 \in \mathbb{R}^{n \times m}$ if there are m control inputs. Finally we compute the disturbance zonotope $\bigoplus_{i=0}^{N-1}(\Phi^i B_{\mathcal{D}})$ via N iterations, the ith of which requires to compute a linear transformation defined by Φ^{i-1} on an n-dimensional zonotope $B_{\mathcal{D}}$ with n generators. The result of all iterations will be added up by computing Minkowski sum for $N-1$ times.

Again, if the complexity of real arithmetic is considered as $O(1)$, a summary of the computational complexity of R_N is given in Table I. The columns denote the complexities of those matrix operations. Therefore, the overall complexity is $O(Nn^3 + Nn^2m + \mathcal{C}_{\text{Adv}})$.

D. Intersections of geometric representations

As we mentioned that we need to check the emptiness of a box/zonotope intersection. Although computing such an intersection has already been recognized as a hard task [22],

[21], [19], [2], the emptiness checking can usually be done efficiently. We may use the following methods for the different intersection types.

Zonotope/zonotope intersection. Given two zonotopes $Z_1 = (\mathbf{c}_1, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$ and $Z_2 = (\mathbf{c}_2, \langle \mathbf{h}_1, \ldots, \mathbf{h}_q \rangle)$, the intersection $Z_1 \cap Z_2$ is nonempty if and only if the following linear program has a solution.

$$\text{Find } \lambda_1, \ldots, \lambda_p, \eta_1, \ldots, \eta_q \in [-1, 1] \text{ s.t.}$$
$$z_1 = \mathbf{c}_1 + \sum_{i=1}^{p} \lambda_i \mathbf{g}_i, \; z_2 = \mathbf{c}_2 + \sum_{i=1}^{q} \eta_i \mathbf{h}_i, \text{ and } z_1 = z_2 \,.$$

Zonotope/polyhedron intersection. Given a zonotope $Z = (\mathbf{c}, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$ and a polyhedron $P = \{\mathbf{x} \in \mathbb{R}^n \mid \bigwedge_{i=1}^{q} (\mathbf{a}_i^T \mathbf{x} \leq b_i)\}$. The intersection $Z \cap P$ is nonempty if and only if the following linear program has a solution.

$$\text{Find } \lambda_1, \ldots, \lambda_p \in [-1, 1] \text{ s.t.}$$
$$z = \mathbf{c} + \sum_{i=1}^{p} \lambda_i \mathbf{g}_i, \text{ and } \bigwedge_{i=1}^{q} (\mathbf{a}_i^T \mathbf{z} \leq b_i) \,.$$

Zonotope/strip intersection. The emptiness checking of both of the above intersections requires to solve linear programs, however, when the other set is defined by a *strip* which is the intersection of two homogeneous halfspaces, i.e., of the form $\{\mathbf{x} \in \mathbb{R}^n \mid b_l \leq \mathbf{a}^T \mathbf{x} \leq b_u\}$, we may only need to evaluate the range of the transformed zonotope $\mathbf{a}^T Z$, the range has no intersection with $[b_l, b_u]$ if and only if the zonotope has no intersection with the strip. We assume that $Z = (\mathbf{c}, \langle \mathbf{g}_1, \ldots, \mathbf{g}_p \rangle)$, then the lower and upper bound of $\mathbf{a}^T Z$ can be computed as $\mathbf{a}^T \mathbf{c} - \sum_{i=1}^{p} |\mathbf{a}^T \mathbf{g}_i|$ and $\mathbf{a}^T \mathbf{c} + \sum_{i=1}^{p} |\mathbf{a}^T \mathbf{g}_i|$ respectively. Notice that the range of a zonotope in each dimension can be computed in a similar way.

In the next section, we present our approaches to monitoring the traces or executions of a linear system.

IV. Real-Time Monitoring on System Executions

The monitor keeps a list of reachable set predictions for a time horizon consisting of N steps into the future. For each time step, it performs the following tasks:

1) Updates all existing reachable set predictions in the list according to the current (estimated) system state $\mathbf{s}$ and control input $\mathbf{c}$;
2) Removes the prediction at the head of the list (since it corresponds to the current time), and computes a new prediction at the next Nth time step according to $\mathbf{s}$, $\mathbf{c}$;
3) Checks the controllability for each node in the list;

The monitor also reports an unsafe incident when an unsafe state is found, or an alert when an uncontrollable prediction is detected. Here, the term “real-time” means that there is always an alert sent before an unsafe state occurs.

Therefore, the monitoring task requires to not only compute reachable state predictions but also verify that there is at least one control (input) sequence keeping the system safe in the future steps. As we will later show that such a sequence can be found by solving an $\exists$-$\forall$ formula.

The existing offline reachability computation methods based on numerical simulation [33], [5], predicate abstraction [3], flowpipe construction [17], [11], [10], invariant computation [29], [27], SMT solving [28], [16], [24] and discrepancy functions [14] cannot be directly applied to this task since in these approaches, the future control inputs starting from a given state are not represented symbolically which makes it hard to check the controllability condition.

The reachable set prediction at the next Nth step for a state $\mathbf{s}$ with the first control input $\mathbf{c}$ is computed as the following (overapproximate) symbolic reachable set

$$R = \{\Phi^N \mathbf{s} + \Phi^{N-1} \Psi \mathbf{c} + \sum_{i=0}^{N-2} (\Phi^i \Psi \mathbf{u}_{N-i})\} \oplus \bigoplus_{i=0}^{N-1} (\Phi^i B_{\mathcal{D}}) \quad (3)$$

such that $\mathbf{u}_2 \cdots \mathbf{u}_N$ are the unknown future control inputs for the next N steps. Suppose, at the very next step, the new control input is $\mathbf{c}'$ and the new state of the system is $\mathbf{s}'$, then (3) can be *concretized* to

$$R' = \{\Phi^{N-1} \mathbf{s}' + \Phi^{N-2} \Psi \mathbf{c}' + \sum_{i=0}^{N-3} (\Phi^i \Psi \mathbf{u}_{N-i})\} \oplus \bigoplus_{i=0}^{N-2} (\Phi^i B_{\mathcal{D}}) \,.$$

A prediction can be repeatedly concretized in the above manner as the unknown control inputs $\mathbf{u}_j$ and current state $\mathbf{x}$ are duly replaced by known inputs $\mathbf{c}_j$ and states $\mathbf{s}$. Eventually, we say that a prediction *expires* if all the unknowns in it are *entirely concretized*. This happens at the end of the time horizon of N steps.

Example IV.1. *We consider the system given in Example III.1, i.e., $\dot{x} = -x + u + d$. The control and disturbance sets are defined by $\mathcal{U} = \{u \mid u \in [0, 2]\}$ and $\mathcal{U} = \{d \mid d \in [-0.5, 0.5]\}$. In order to make the explanation concise, we approximately present the coefficients.*

If we set the step size by $\delta = 0.02$, the symbolic reachable set for $N = 3$, i.e., at the time $t = 0.06$, can be computed as

$$R = 0.9418 x_0 + 0.0190 u_1 + 0.0194 u_2 + 0.0198 u_3 + Z_{\mathcal{D}}$$

such that $Z_{\mathcal{D}} \; : \; (0, \langle 0.01, 0.0098, 0.0096 \rangle)$ which is the accumulation of the disturbances.

When the initial state is $x(0) = 5$ and the first control input is $u_1 = 0$, the prediction is concretized to be

$$R_1 = 4.7090 + 0.0194 u_2 + 0.0198 u_3 + Z_1$$

in the first time step, such that $Z_1 = Z_{\mathcal{D}}$.

In the second time step, if the system state is $x(0.02) = 4.9$ and the control input is 1.5 during the step. The prediction R_1 is then concretized to be

$$R_2 = 4.7370 + 0.0198 u_3 + Z_2$$

wherein $Z_2 = (0, \langle 0.01, 0.0098 \rangle)$.

Assume that $x(0.04) = 4.8$ and the control input is 1 in the third time step, the prediction is then entirely concritized to be

$$R_3 = 4.7248 + Z_3$$

such that $Z_3 = (0, \langle 0.01 \rangle)$.

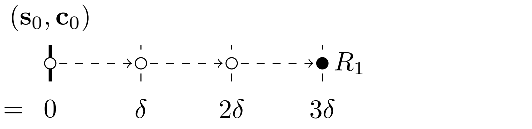
Now we present the detailed monitoring algorithm as follows. The monitor keeps a list $\mathcal{L}_R$ consisting of all reachable set predictions in the future N steps. In each time step, it does the following work.

1. Obtain the current system state $\mathbf{s}$ and the control input $\mathbf{c}$ which will be used in the current time step.
2. Check the safety of $\mathbf{s}$. If it is unsafe, then the monitor reports an unsafe incident.
3. Concretize the predictions in the list $\mathcal{L}_R$ with respect to $\mathbf{s}$ and $\mathbf{c}$.
4. Compute a prediction at the next Nth step with respect to the state $\mathbf{s}$ and the first control input $\mathbf{c}$, and append it into $\mathcal{L}_R$.
5. Check the controllability of all predictions in $\mathcal{L}_R$. If there is an uncontrollable prediction, then the monitor sends out an *alert*.
6. Remove the prediction which is to be expired.

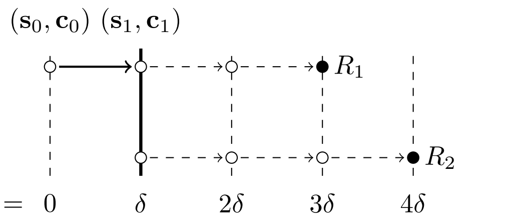
Notice that every prediction will be entirely concretized at its Nth step, so the monitor needs to keep at most N reachable sets in the list.

Figure 4 illustrates the changes of a watch list with $N = 3$ predictions during the first 3 time steps. After the 3rd time step, the list contains the predictions for all of the future N steps.

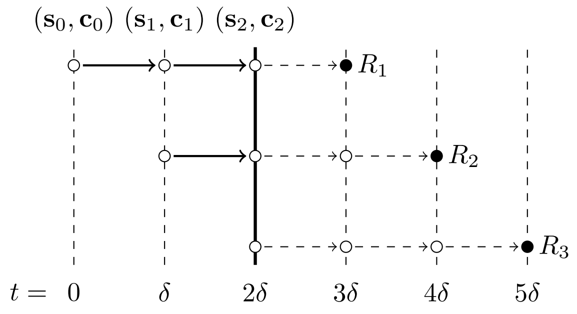
One may point out that the first $N-1$ time steps are missed by our monitoring algorithm. However, we can take them into account by computing their predictions and put them into the list at the very beginning. In the next section, we present our method of verifying the controllability for a prediction.


$(\mathbf{s}_0, \mathbf{c}_0)$
R_1
$t =$ 0 δ 2δ 3δ

(a) Time step at $t = 0$. The prediction R_1 for $t = 3\delta$ is computed according to the current state $\mathbf{s}_0$ and the control input $\mathbf{c}_0$ used in the step.


$(\mathbf{s}_0, \mathbf{c}_0)$ $(\mathbf{s}_1, \mathbf{c}_1)$
R_1
R_2
$t =$ 0 δ 2δ 3δ 4δ

(b) Time step at $t = \delta$. The prediction R_1 is concretized by the current state $\mathbf{s}_1$ and control input $\mathbf{c}_1$. A new predication R_2 for the reachable state from $\mathbf{s}_1$ at $t = 4\delta$ is computed and added to the list.


$(\mathbf{s}_0, \mathbf{c}_0)$ $(\mathbf{s}_1, \mathbf{c}_1)$ $(\mathbf{s}_2, \mathbf{c}_2)$
R_1
R_2
R_3
$t =$ 0 δ 2δ 3δ 4δ 5δ

(c) Time step at $t = 2\delta$. The predication R_1, R_2 are concretized by the current state $\mathbf{s}_2$ and control inputs $\mathbf{c}_2$. A new prediction R_3 for the reachable state from $\mathbf{s}_2$ at $t = 5\delta$ is computed and added to the list. R_1 is to be expired.

Fig. 4. Content of the watch list in the first 3 time steps.

A. Verification of controllability

A reachable set prediction R at the next Nth step for a state $\mathbf{s}$ is of the form

$$\underbrace{\{\Phi^{N-j}\mathbf{s}' + \Phi^{N-j-1}\Psi\mathbf{c}' + \sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i})\}}_{\mathbb{A}(\mathbf{s}',\mathbf{u}_{j+2},\cdots,\mathbf{u}_N)} \oplus \underbrace{\bigoplus_{i=0}^{N-j-1}(\Phi^i B_{\mathcal{D}})}_{\mathbb{B}} \tag{4}$$

for some $0 \leq j \leq N-1$, such that the system evolves from $\mathbf{s}$ to $\mathbf{s}'$ in the first j time steps, and the control input in the $(j+1)$st step is $\mathbf{c}'$. Then, to verify the controllability of the above prediction, we need to find a control sequence $\mathbf{u}_{j+2}\cdots\mathbf{u}_N \in \mathcal{U}^{N-j-1}$ such that the prediction is contained in the safe set $\mathcal{S}$. To do so, we may solve the following $\exists$-$\forall$ formula.

$$(\exists \mathbf{u}_{j+2}, \ldots, \mathbf{u}_N \in \mathcal{U})(\forall \mathbf{z}_D \in \mathbb{B})\mathbb{A}(\mathbf{s}', \mathbf{u}_{j+2}, \cdots, \mathbf{u}_N) \oplus \mathbb{B} \in \mathcal{S} \tag{5}$$

wherein $\mathbb{A}$ and $\mathbb{B}$ are as defined in Eq. (4).

Instead of using a (nonlinear) SAT/SMT solver, we propose an approach to check the satisfiability of a simple predicate γ , if it is satisfiable then so is the formula (5). The method requires to compute a Minkowski difference of the safe set $\mathcal{S}$ and the term $\{\Phi^{N-j}\mathbf{s}' + \Phi^{N-j-1}\Psi\mathbf{c}'\} \oplus \bigoplus_{i=0}^{N-j-1}(\Phi^i B_{\mathcal{D}})$.

The *Minkowski difference* of two sets X, Y is defined by $X \ominus Y = \bigcap_{y\in Y}\{x - y \,|\, x \in X\}$. Intuitively, the sum of any elements in $X \ominus Y$ and Y is always in X. If X is given by a box and Y is a compact set then $X \ominus Y$ is also a box which can be computed based on the following lemma.

Lemma IV.1. *If $X \subseteq \mathbb{R}^n$ is a box and $Y \subseteq \mathbb{R}^n$ is a compact set, then $X \ominus Y$ is also a box. Its lower bound in the ith dimension can be computed as lo(X,i) − lo(Y,i), while the upper bound can be computed as up(X,i) − up(Y,i) such that lo(Z,i) and up(Z,i) denote the lower and upper bound respectively for a set Z in its ith dimension.*

Proof. By the definition of Minkowski difference, the set $X \ominus Y$ is the intersection of the boxes $\{\mathbf{x} - \mathbf{y} \,|\, \mathbf{x} \in X\}$ for $\mathbf{y} \in Y$, then it is also a box.

The lower bound of $X \ominus Y$ in the ith dimension for any $1 \leq i \leq n$ is the maximum lower bound of the ith dimension of $\{\mathbf{x} - \mathbf{y} \,|\, \mathbf{x} \in X\}$ for all $\mathbf{y} \in Y$, and hence it is the value of lo(X,i) − lo(Y,i). The upper bound can be proved analogously. □

The case that X is a strip and Y is a compact set can be handled in a similar way. Assume that $X = \{\mathbf{x} \in \mathbb{R}^n \,|\, b_l \leq \mathbf{a}^T\mathbf{x} \leq b_u\}$, we compute the linear transformation $\mathbf{a}^T Y$, then the Minkowski difference is also a strip which is defined by

$$X \ominus Y = \{\mathbf{x} \in \mathbb{R}^n \mid b_l - y_\ell \leq \mathbf{a}^T\mathbf{x} \leq b_u - y_u\}$$

such that y_ℓ and y_u are the lower and upper bound of $\mathbf{a}^T Y$ respectively.

The Minkowski difference

$$U_{\text{safe}} = \mathcal{S} \ominus \left(\{\Phi^{N-j}\mathbf{s}' + \Phi^{N-j-1}\Psi\mathbf{c}'\} \oplus \bigoplus_{i=0}^{N-j-1} (\Phi^i B_{\mathcal{D}}) \right)$$

is a *safe envelope* for the control inputs, since for any control sequence $\mathbf{u}_{j+2} \cdots \mathbf{u}_N \in \mathcal{U}^{N-j-1}$, if $\sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i}) \in U_{\text{safe}}$, then the prediction is safe, in other words, the control sequence makes the reachable state safe regardless of the disturbances. To ensure the existence of such a sequence, we may check the emptiness of the intersection

$$U_I = U_{\text{safe}} \, \cap \, \left\{ \mathbf{u} \in \mathbb{R}^m \mid \mathbf{u} = \sum_{i=0}^{N-j-2} (\Phi^i\Psi\mathbf{u}_{N-i}), \mathbf{u}_i \in \mathcal{U} \right\} .$$

Hence the controllability predicate is defined by $\gamma : U_I \neq \emptyset$, which can be checked by a LP solver or the methods introduced in Section III.

For entirely concretized predictions, since there is no control variable, we check whether the set is entirely contained in the safe set. If so, the set is controllable, otherwise it is uncontrollable.

Theorem IV.1. *If the predicate γ is satisfiable, then the prediction is controllable.*

Proof. If $U_I \neq \emptyset$, there is at least one control sequence $\mathbf{u}_{j+2} \cdots \mathbf{u}_N \in \mathcal{U}^{N-j-1}$ such that $\sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i}) \in U_{\text{safe}}$. Hence, the set

$$\{\Phi^{N-j}\mathbf{s}' + \Phi^{N-j-1}\Psi\mathbf{c}'\} \oplus \bigoplus_{i=0}^{N-j-1} (\Phi^i B_{\mathcal{D}}) \oplus \{\sum_{i=0}^{N-j-2} (\Phi^i\Psi\mathbf{u}_{N-i})\}$$

containing all possibly reachable state under the control sequence is contained in the safe set $\mathcal{S}$. □

Example IV.2. *Assume that the safe set is defined by $\mathcal{S} = \{x \in \mathbb{R} \mid x \in [4.71, 5]\}$. The control safe envelopes for the 2 reachable set prediction R_1, R_2 which are not entirely concretized in Example IV.1 can be computed as follows.*

For R_1, the range of the set $4.7090 + Z_1$ is $[4.6796, 4.7284]$, and therefore

$$U_{\textit{safe}} = [4.71, 5] \ominus [4.6796, 4.7284] = [0.0304, 0.2716] .$$

For R_2, the range of $4.7370 + Z_2$ is $[4.7172, 4.7568]$, then the safe envelope can be computed as

$$U_{\textit{safe}} = [4.71, 5] \ominus [4.7172, 4.7568] = [-0.0072, 0.2432] .$$

Computational complexity. The concretization in each time step for a reachable set prediction only involves numerical evaluation by real arithmetic. Since the matrices and zonotopes in the symbolic reachable set can be computed offline in advance, the most expensive work online is to compute the matrix multiplications of the type $M_1 M_2$ such that $M_1 \in \mathbb{R}^{n\times n}$ and $M_2 \in \mathbb{R}^{n\times 1}$, and the safe envelope which requires range evaluation for zonotopes with at most N generators. Therefore the concretization task in a time step has a complexity at most quadratic in both n and N. If a LP solver is used for checking the emptiness of U_I, we need to additionally take its complexity into account. Our experiments will show that the monitoring task in one time step can be done very efficiently.

B. Discussion on the verification results

Our monitoring algorithm produces the following verification results in a time step based on the reachable set predications and their control safe envelopes.

(a) *Controllable* - The current state is safe and the set U_I is nonempty.
(b) *Alert* - The current state is safe and the set U_I is empty. That is, the current state might not be controllable.
(c) *Unsafe* - The current state is already unsafe.

We show that our monitoring algorithm always sends out an alert at least one time step before an unsafe incident occurs. Since any reachable state is included by the concretized reachable set prediction one step before, the prediction is not entirely contained in the safe set if there is an unsafe reachable state.

Theorem IV.2. *For $i \geq 2$, if the state $\mathbf{s}_i$ in the ith time step is unsafe, then the previous prediction at the $(i-1)$th step was unsafe.*

However, the converse is not necessarily true. One can imagine a state that is unsafe at the i^{th} step. However, the worst case disturbance does not happen and thus, the system may transition to a controllable state.

We extend the concept of controllability from system states to executions. Given an execution $(\mathbf{s}_1, \mathbf{c}_1), \ldots, (\mathbf{s}_N, \mathbf{c}_N)$ with a step size $\delta > 0$ such that $\mathbf{s}_i$ is the system state at the time $t = (i-1)\delta$, and $\mathbf{c}_i$ is the control input which is used in the ith time step $t \in [(i-1)\delta, i\delta]$. Then the controllability of the execution can be defined by the following 4 cases.

(a) *Controllable* - The states $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1}$ are controllable, and $\mathbf{s}_N$ is safe.
(b) *Alert* - The state $\mathbf{s}_N$ is safe, however at least one of the states $\mathbf{s}_1, \ldots, \mathbf{s}_{N-1}$ causes an alert. Although this is often seen as a false alarm, we will argue are in fact "near misses" where the disturbance inputs could have potentially led to a safety violation.
(c) *Unsafe I* - The state $\mathbf{s}_N$ is unsafe, and only $\mathbf{s}_{N-1}$ causes an alert.
(d) *Unsafe II* - The state $\mathbf{s}_N$ is unsafe, at least one previous state other than $\mathbf{s}_{N-1}$ causes an alert.

The reason to classify Unsafe I and Unsafe II is that Unsafe II represents an early warning for a real error. However Unsafe I only gives the monitor one step before an unsafe incident occurs.

Figure 5 shows some examples for the controllability of an execution. These results may tell us whether a controller is robust under disturbances. If not then whether there is a robust controller with the same control input range.

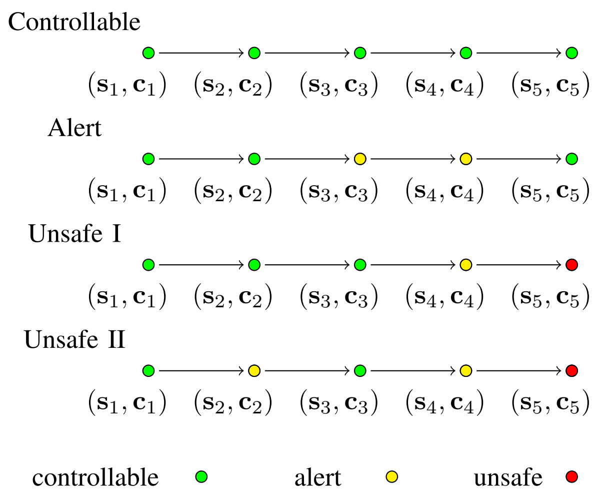

Controllable
$(\mathbf{s}_1,\mathbf{c}_1)$
$(\mathbf{s}_2,\mathbf{c}_2)$
$(\mathbf{s}_3,\mathbf{c}_3)$
$(\mathbf{s}_4,\mathbf{c}_4)$
$(\mathbf{s}_5,\mathbf{c}_5)$
Alert
Unsafe I
Unsafe II
controllable
alert
unsafe

Fig. 5. Controllability of executions

We briefly discuss the feasibility of (a) finding a safe control sequence for a controllable state, and (b) finding a PWC disturbance sequence for an alert state.

Given a reachable set prediction R which is defined by

$$\{\Phi^{N-j}\mathbf{s}'+\Phi^{N-j-1}\Psi\mathbf{c}'+\sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i})\}\oplus\bigoplus_{i=0}^{N-j-1}(\Phi^i B_{\mathcal{D}})$$

such that $0\leq j\leq N-2$. If R is controllable, then a safe control sequence can be found by solving the following linear program:

$$\text{Find } \mathbf{u}_{j+2},\ldots,\mathbf{u}_N\in\mathcal{U} \text{ s.t.}$$
$$\mathbf{u}=\sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i}) \text{ and } \mathbf{u}\in U_{\text{safe}}\,.$$

When R causes an alert or is unsafe, we may treat $\sum_{i=0}^{N-j-2}(\Phi^i\Psi\mathbf{u}_{N-i})$ as a zonotope and introduce new variables to represent disturbances. An unsafe disturbance sequence can be found by solving a similar linear program over those disturbance variables.

V. Experiments

We implemented a prototype tool in C++ based on the library of the tool Flow* [12]. The examples in our experiments are all closed-loop control systems. We add time-varying disturbances to the feedback loop, and monitor the system executions.

A. Benchmarks settings

All of the benchmarks are of the structure given in Figure 6. The plant is defined by a linear system whose state variables are $\mathbf{x}$. The controller reads the output $\mathbf{y}$ of the system by time steps, such that $\mathbf{y}$ is the image of $\mathbf{x}$ under a linear transformation, i.e., $\mathbf{y}=C_{\text{out}}\mathbf{x}$ wherein C_{out} is a constant matrix, and then computes an control input $\mathbf{u}$ for the next time step. The input and output of the controller might be

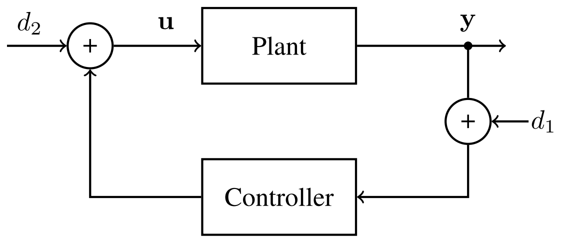


Fig. 6. Structure of the benchmarks

influenced by some disturbances which are denoted by d_1, d_2 in the figure. We give an example as below.

The dynamics of an inverted pendulum can be described by the following ODE [4],

$$\begin{pmatrix}\dot{x}\\ \dot{v}_x\\ \dot{\theta}\\ \dot{v}_\theta\end{pmatrix}=\begin{pmatrix}0&1&0&0\\0&-0.1818&2.6727&0\\0&0&0&1\\0&-0.4545&31.1818&0\end{pmatrix}\begin{pmatrix}x\\ v_x\\ \theta\\ v_\theta\end{pmatrix}+\begin{pmatrix}0\\1.8182\\0\\4.5455\end{pmatrix}u$$

such that x is the position of the cart, v_x is the cart velocity, θ is the angle of the pendulum and v_θ is the angular velocity. The PWC control input u can be designed as

$$u=x+1.6567v_x-18.68540\theta+3.4594v_\theta$$

to stabilize the system, i.e., make the executions of the system converge to zero. Then, we additionally add independent disturbances to the values of x, v_x, θ, v_θ and u.

Table II gives a summary of all our tests. The benchmarks are adapted from the ones described in [25], [4], [15], [1]. For each test, we build a simulator which generates executions of bounded lengths. For example, if the time step is δ and we are interested in the system behavior in the time horizon T, then each execution consists of T/δ elements which are the system states and control inputs at the beginning of the time steps. For the aircraft pitch benchmarks, we use the lookahead value $N=20$, while $N=10$ is used for the other benchmarks.

The tests on the same benchmark are based on slightly different PWC controllers along with the disturbance ranges of different sizes. The controllers are using the same strategy but different parameters which allow them to produce inputs in different ranges. For example, the controller of the inverted pendulum model can be more restrictive if we enlarge the coefficients in u, however it also enlarges the set $\mathcal{U}$. It is the reason why we use large sets for $\mathcal{U}$ in our monitor for more restrictive controllers.

B. Performance evaluation

The performance of our real-time monitoring algorithm is evaluated in the following way. For each benchmark, we define a safe set which is a box. For each test, we perform the

TABLE II
DESCRIPTION OF THE TESTS WITH BENCHMARKS. LEGENDS: T: THE TIME HORIZON IN INTEREST, $|\mathbf{x}|$: # OF STATE VARIABLES, $|\mathbf{u}|$: # OF CONTROL INPUTS, δ: STEP SIZE FOR MONITORING AND PWC CONTROL INPUTS, $\mathcal{U}$: RANGE OF EACH CONTROL INPUT, $\mathcal{D}$: RANGE OF EACH TIME-VARYING DISTURBANCE.

No.	System	T	$\lvert\mathbf{x}\rvert$	$\lvert\mathbf{u}\rvert$	δ	$\mathcal{U}$	$\mathcal{D}$
1	Cruise Control	10	1	1	0.05	[-0.5,1.5]	[-0.1,0.1]
2						[-0.5,1.5]	[-0.5,0.5]
3						[-1.5,2.5]	[-0.5,0.5]
4	Motor Speed	10	2	1	0.05	[-1,2]	[-0.2,0.2]
5						[-1,2]	[-0.5,0.5]
6						[-1.5,2.5]	[-0.5,0.5]
7	Inverted Pendulumn	5	4	1	0.02	[-0.5,1.5]	[-0.5,0.5]
8						[-0.5,1.5]	[-1.5,1.5]
9						[-1,2]	[-1.5,1.5]
10	Aircraft Pitch	20	3	1	0.02	[-6,6]	[-0.1,0.1]
11						[-6,6]	[-0.2,0.2]
12						[-7,7]	[-0.2,0.2]
13	Ball & Beam	5	4	1	0.02	[-35,5]	[-4,4]
14						[-35,5]	[-8,8]
15						[-40,10]	[-8,8]
16	F16	20	4	1	0.02	[-1,1]	[-0.1,0.1]
17						[-1,1]	[-0.2,0.2]
18						[-2,2]	[-0.2,0.2]
19	Suspension	5	5	2	0.02	[-0.8,0.8] [-0.2,0.2]	[-0.2,0.2]
20						[-0.8,0.8] [-0.2,0.2]	[-1,1]
21						[-1.3,1.3] [-0.3,0.3]	[-1,1]

TABLE III
SETTINGS FOR SYSTEM EXECUTIONS. THE VARIABLE x_i DENOTES THE iTH STATE VARIABLE.

Benchmark	Initial set	Safe set $\mathcal{S}$
Cruise Control	$x_i \in [-0.1, 0.1]$	$x_1 \in [-0.2, 3]$
Motor Speed	$x_i \in [-0.1, 0.1]$	$x_1 \in [-0.12, 0.12]$
Inverted Pendulumn	$x_i \in [-0.05, 0.05]$	$x_1 \in [-0.22, 0.12]$ $x_3 \in [-0.1, 0.1]$
Aircraft Pitch	$x_i \in [-0.1, 0.1]$	$x_3 \in [-2.5, 2.5]$
Ball & Beam	$x_i \in [-0.01, 0.01]$	$x_1 \in [-0.011, 0.011]$
F16	$x_i \in [-0.1, 0.1]$	$x_3 \in [-0.4, 0.6]$
Suspension	$x_i \in [-0.01, 0.01]$	$x_3 \in [-0.11, 0.11]$

algorithm on 100 executions and count the numbers of controllable, alert, unsafe I and unsafe II executions respectively by checking all states. Table III gives the safe sets and the initial sets based on which the executions are computed. The experimental results are given in Table IV.

In our tests, we use the LP solver in the GNU GLPK library to check the emptiness of the intersection U_I. When the safe set is also a strip, we also use the method of range evaluation to check the emptiness. In Table IV, we give the best, worst and average time costs of computing one monitoring step, that includes computing and concretizing reachable sets for the future N steps, and checking their controllability. We also provide the standard deviation in each test.

From the table, we can see that even the worst time cost of a test is much lower than the real time step size, so each prediction is computed and verified before its expiration, in order words, no deadline is missing.

The LP solver has a better performance than the range evaluation method in controllability verification. The reason could be that the latter one is implemented in a prototype tool. To further ensure the conservativeness, that is to compute a guaranteed overapproximation for a reachable set, we may use interval arithmetic in the monitor.

VI. CONCLUSION

We introduced a framework to real-time monitor the executions of controlled linear systems. In fact, our method can be extended and applied to dealing with more complex systems, such as Linear Time-Varying (LTV) or even nonlinear systems. The directions of our future work are given as below.

1. Since the reachable set predictions for LTV stochastic systems can also be computed symbolically, we plan to extend our method to verify the controllability of stochastic executions.
2. We plan to verify the properties that can be defined by a richer language in the real-time monitoring framework. For example, linear temporal logic properties.
3. We expect to propose new algorithms to recover a system to a controllable state when some possible uncontrollability in the future is detected.

Acknowledgments: This work was suppored by the US National Science Foundation (NSF) under award numbers 1646556 and 1446900, and the Air Force Research Laboratory (AFRL). All opinions expressed are those of the authors and not necessarily by the NSF or AFRL.

REFERENCES

[1] A. Abate, I. Bessa, D. Cattaruzza, L. C. Cordeiro, C. David, P. Kesseli, and D. Kroening. Sound and automated synthesis of digital stabilizing controllers for continuous plants. In *Proc. of HSCC'17*, pages 197–206. ACM, 2017.
[2] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.
[3] R. Alur, T. Dang, and F. Ivancic. Progress on reachability analysis of hybrid systems using predicate abstraction. In *Proceedings of the 6th International Workshop on Hybrid Systems: Computation and Control (HSCC'03)*, volume 2623 of *LNCS*, pages 4–19. Springer, 2003.
[4] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2011.
[5] S. Bak and P. S. Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proc. of HSCC'17*, pages 173–178. ACM, 2017.
[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] Alberto Bemporad and Manfred Morari. *Robust model predictive control: A survey*, pages 207–226. Springer London, London, 1999.
[9] Stephen Boyd, Laurent ElGhaoui, Eric Feron, and V. Balakrishnan. *Linear Matrix Inequalities in Systems and Control Theory*. Studies in Applied Mathematics. SIAM, 1994.
[10] X. Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.
[11] 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.
[12] 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.

TABLE IV
EXPERIMENTAL RESULTS FOR THE REAL-TIME MONITORING ON 100 EXECUTIONS. LEGENDS: U II: UNSAFE II, U I: UNSAFE I, CONT: CONTROLLABLE, OFFLINE: OFFLINE COMPUTATION TIME, ONLINE: ONLINE COMPUTATION TIME, AVG: AVERAGE TIME COST, MIN: BEST TIME COST, MAX: WORST TIME COST, DEV: STANDARD DEVIATION OF THE TIME COSTS OVER 100 EXECUTIONS.

No.	# of Executions				Offline	Online							
						Using LP				Without LP			
	U II	U I	Alert	Cont		Avg	Min	Max	Dev	Avg	Min	Max	Dev
1	0	0	0	100	1.03e-3	5.47e-4	2.46e-4	1.19e-3	9.30e-5	7.69e-4	5.36e-4	2.08e-3	1.00e-4
2	15	0	85	0	9.98e-4	5.42e-4	2.35e-4	1.61e-3	9.80e-5	7.75e-4	5.34e-4	1.58e-3	1.12e-4
3	0	0	0	100	1.08e-3	5.63e-4	2.45e-4	1.43e-3	1.00e-4	7.99e-4	5.32e-4	1.72e-3	1.33e-4
4	0	0	0	100	2.24e-3	1.04e-3	5.07e-4	2.62e-3	1.77e-4	1.26e-3	1.00e-3	2.52e-3	1.27e-4
5	0	18	0	82	2.24e-3	1.04e-3	5.08e-4	2.86e-3	1.82e-4	1.29e-3	1.03e-3	2.61e-3	1.40e-4
6	0	0	0	100	2.24e-3	1.03e-3	5.09e-4	2.34e-3	1.72e-4	1.27e-3	9.92e-4	2.49e-3	1.42e-4
7	0	0	0	100	7.29e-3	2.11e-3	1.25e-3	4.99e-3	2.56e-4	-	-	-	-
8	1	0	0	99	7.11e-3	2.07e-3	1.28e-3	4.11e-3	1.73e-4	-	-	-	-
9	0	0	0	100	7.03e-3	2.11e-3	1.26e-3	4.74e-3	2.69e-4	-	-	-	-
10	0	0	0	100	7.81e-3	5.91e-3	3.40e-3	1.01e-2	7.30e-4	6.85e-3	5.03e-3	1.27e-2	7.58e-4
11	2	0	18	80	7.26e-3	5.24e-3	3.48e-3	1.02e-2	7.77e-4	6.76e-3	5.04e-3	1.38e-2	7.62e-4
12	0	0	0	100	7.76e-3	6.17e-3	3.34e-3	1.04e-2	7.64e-4	6.67e-3	5.15e-3	1.22e-2	8.55e-4
13	0	0	0	100	5.45e-3	2.20e-3	1.18e-3	4.34e-3	3.27e-4	2.58e-3	2.09e-3	4.97e-3	3.87e-4
14	0	0	5	95	5.37e-3	2.12e-3	1.17e-3	4.15e-3	3.06e-4	2.51e-3	2.09e-3	4.73e-3	3.57e-4
15	0	0	0	100	5.71e-3	2.21e-3	1.16e-3	4.33e-3	3.13e-4	2.58e-3	2.10e-3	5.70e-3	3.35e-4
16	0	0	0	100	7.58e-3	2.24e-3	1.32e-3	4.78e-3	3.29e-4	2.67e-3	2.13e-3	6.96e-3	4.00e-4
17	63	0	35	2	7.64e-3	2.23e-3	1.32e-3	4.62e-3	3.39e-4	2.68e-3	2.13e-3	5.39e-3	3.96e-4
18	0	0	0	100	7.66e-3	2.28e-3	1.32e-3	4.86e-3	3.62e-4	2.65e-3	2.13e-3	4.96e-3	4.18e-4
19	0	0	0	100	1.28e-2	3.77e-3	2.13e-3	7.17e-3	4.52e-4	4.21e-3	3.41e-3	7.73e-3	5.74e-4
20	64	0	36	0	1.18e-2	3.84e-3	2.13e-3	7.73e-3	5.15e-4	4.03e-3	3.38e-3	7.85e-3	5.29e-4
21	0	0	0	100	1.17e-2	3.81e-3	2.12e-3	7.16e-3	5.14e-4	4.05e-3	3.37e-3	8.36e-3	5.42e-4

[13] X. Chen and S. Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 13–24. IEEE Press, Nov 2016.

[14] 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.

[15] G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems (7th edition)*. Pearson, 2014.

[16] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1(3-4):209–236, 2007.

[17] 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.

[18] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control (HSCC'05)*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[19] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, volume 4981 of *LNCS*, pages 215–228. Springer, 2008.

[20] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Proceedings of the 9th International Workshop on Hybrid Systems: Computation and Control (HSCC'06)*, volume 3927 of *LNCS*, pages 257–271. Springer, 2006.

[21] P. Guerra, V. Puig, and M. Witczak. Robust fault detection with unknown-input interval observers using zonotopes. *IFAC Proceedings Volumes*, 41(2):5557 – 5562, 2008.

[22] L. J. Guibas, A. Nguyen, and L. Zhang. Zonotopes as bounding volumes. In *Proc. of SODA'03*, pages 803–812. ACM/SIAM, 2003.

[23] Taylor T. Johnson, Stanley Bak, Marco Caccamo, and Lui Sha. Real-time reachability for verified simplex design. *ACM Trans. Embedd. Comput. Syst.*, 15(2):29, May 2016.

[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] B. Lu. *Linear Parameter-Varying Control of an F-16 Aircraft at High Angle of Attack*. PhD thesis, North Carolina State University, 2005.

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

[27] A. Platzer and J.-D. Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *Proc. of IJCAR'08*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[28] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *Proc. of HSCC'05*, volume 3414 of *LNCS*, pages 573–589. Springer, 2005.

[29] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *Proceedings of the 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *LNCS*, pages 539–554. Springer, 2004.

[30] D. Seto, E. Ferreira, and T. F. Marz. Case study: Development of a baseline controller for automatic landing of an F-16 aircraft using linear matrix inequalities (LMIs), 1999. Technical Report CMU/ SEI-99-TR-020.

[31] D. Seto and L. Sha. A case study on analytical analysis of the inverted pendulum real-time control system, 1999. CMU/ SEI, Tech. Rep.

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

[33] B. Silva, K. Richeson, B. Krogh, and A. Chutinan. Modeling and verification of hybrid dynamical system using checkmate. In *ADPM 2000*. Shaker, 2000.

[34] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1995.