EPiC Series in Computing

Volume 74, 2020, Pages 209–241

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

Analysis of Real-Time Control Systems using First-Order Continuization *

Maximilian Gaukler

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
max.gaukler@fau.de

Abstract

Experience Report: Real-Time control systems can be difficult to analyze due to the mixture of discrete-time and continuous-time dynamics. This difficulty is particularly pronounced if the timing is non-periodic, e.g., due to network or execution effects. Still, most control loops behave similar to a purely continuous-time system disturbed by a small discretization error, which is exploited by Bak and Johnson (2015) in the method of *Continuization*. This paper uncovers limitations of that work and presents an extension, First-Order Continuization, based on a new formal framework that recovers previous results and eases future development.

1 Introduction

From coffee machines to self-driving cars, our lifes are increasingly depending on the correct function of real-time control systems. While it may be dismissed as bad luck if sometimes a cup of coffee is brewed too cold, safety-critical functions such as lane keeping of an autonomous vehicle must work *provably correct* under all specified conditions. Typically, correctness of the controller design is not guaranteed by construction, i.e., some of the relevant safety properties such as stability, state bounds and response time require additional verification. In practice, this verification is often approximated by a multitude of randomized simulations and real-life test runs. However, these only cover a finite number of the infinitely many possible scenarios. A promising solution for full coverage is *set-valued reachability analysis* [6], which no longer considers individual trajectories, but an outer bound of the set of possible states over time.

As real-time control systems combine a discrete-time digital controller and a continuous-time physical environment, they are ill-suited for standard methods of reachability analysis [4, 5, 9]. The main reason is that discrete transitions are treated separately from continuous evolution and often incur approximations due to set operations. The error from these approximations is particularly pronounced if the control loop admits uncertain timing due to varying network delays or execution times. One solution is to tailor reachability analysis to the specific structure of digital control loops [1, 5]. While this can provide good results for a very specific class of

*Acknowledgement: Thanks to Jiapeng Wu for showing the feasibility of an early version of first-order continuization in his Master's thesis under supervision of the author.

systems, it comes at the cost that one can no longer draw from the decades of experience condensed in existing reachability analysis tools. Therefore, for example, an extension from linear to nonlinear systems is no longer as easy as enabling a configuration option in an existing tool, but could require months to years of coding and research. An alternative approach that avoids this problem is *Continuization* [4, 2]: If the mixed discrete-time and continuous-time original system behaves "smoothly enough", it can be approximated by a continuous-time system with "smooth" behavior that can be analyzed efficiently with existing tools. Formally, the discontinuous solution of the original hybrid system is approximated by the continuous solution of a differential inclusion, and the approximation error is bounded.

This paper extends the work on Continuization by Bak and Johnson [4] towards a broader class of digital control loops. It presents a step towards solving the benchmark problems of [9] by reachability analysis that so far have only been solved by LMI-based methods [8]. The key contents are the following: Section 3 develops a specialized formal framework for proving Continuization schemes using abstractions of hybrid automata and a modified equivalence relation. Section 4.1 restates prior work by Bak and Johnson, here termed *Zero-Order Continuization*, in the new formal framework, correcting some errors in the details and uncovering a previously unknown limitation: This method is limited to state feedback, e.g., proportional controllers. Hence, it cannot handle controllers with internal dynamics, e.g., integral controllers, for which a novel extension named *First-Order Continuization* is developed in Section 4.2. Experiments in Section 5 show that this method is feasible with some limitations that are discussed in Section 6.

2 Notation

Definitions are denoted $a := b$, meaning "a defined as b". Time arguments are mostly omitted if their value is t, i.e., $x(t)$ may be abbreviated as x, but $x(kT)$ is never abbreviated. For a state x, the time derivative is denoted $\dot{x}$, and the successor state of a discrete state transition ("jump") is denoted x'.

The closed ball of radius r and appropriate dimension n is

$$B_r := \{x \in \mathbb{R}^n \mid |x| \leq r\}. \tag{1}$$

Minkowski addition $\oplus$, set-valued multiplication $\otimes$ and scaling of sets are defined as

$$A \oplus B := \{a+b \mid a \in A, b \in B\}, \quad A \otimes B := \{ab \mid a \in A, b \in B\} \quad \text{and} \quad cA := \{ca \mid a \in A\}. \tag{2}$$

The smallest closed multidimensional interval (box) containing the set S is denoted $\square S$. The cartesian product is $\times$. Closed intervals are denoted $[a; b]$, open intervals as $(a; b)$.

For a function $f(x)$ with non-scalar argument or result, the partial derivative $\partial f(x)/\partial x$ is generalized as the Jacobian matrix

$$\frac{\partial \begin{bmatrix} f_1(x_1, x_2, \dots) & f_2(x_1, x_2, \dots) & \dots \end{bmatrix}^\top}{\partial \begin{bmatrix} x_1 & x_2 & \dots \end{bmatrix}^\top} := \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 & \dots \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 & \dots \\ \dots & & \end{bmatrix}. \tag{3}$$

3 Abstraction of Hybrid Automata

This section presents the theoretical foundation that will later be used prove Continuization.

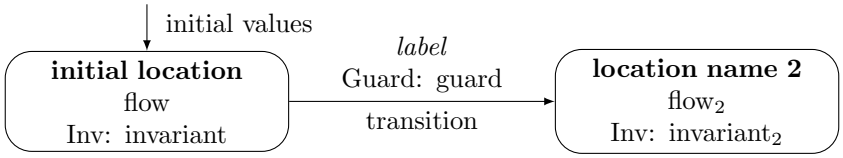

Figure 1: Legend for hybrid automata used in this publication

3.1 Definition of Hybrid Automaton

In this paper, hybrid automata analogous to the definition of Henzinger [10] are used to describe the mixed continuous and discrete behavior of digital control loops. An introduction for readers unfamiliar with the subject can be found in [6]. A formal mathematical definition is detached to Appendix A.1, as it is only relevant for the proofs.

Definition 3.1 (Graphical Notation for Hybrid Automata)**.** Hybrid automata are graphically depicted per Fig. 1. Irrelevant labels and location names, as well as disabled transitions (guard=*false*) may be omitted. State components not mentioned in a transition function remain unchanged, e.g., the empty transition function represents the identity transition $x' = x$. ◁

3.2 Continuous-Variable Reachability and Equivalence

To check if two control loops have the same physical behavior, this paper introduces adapted definitions for the reachable set and equivalence of hybrid automata. In contrast to the conventional definitions, an automaton remains equivalent after a transformation of the state vector or after renaming discrete transition labels.

Definition 3.2 (Reachable Set of a Continuous Variable)**.** Let H be a hybrid automaton with $x \in \mathbb{R}^n$ as the vector of continuous state variables. Then, "$x = \xi$ is reachable by H at t", abbreviated as $\xi \in \mathit{Reach}_x(H, t)$, iff there is a trajectory of H that has a duration of t and ends with (l, ξ), where the location l is irrelevant.

For reference, in the formalism of Henzinger [10, Def. 1.4 and 1.5], this definition reads

$$\mathit{Reach}_x(H,t) := \Big\{ x \mid \exists Tr, j: \quad Tr = \langle \mathit{label}_i, (\mathit{location}_i, x_i)\rangle_{i\geq 1} \text{ is an initialized trajectory of } H, \quad x_j = x, \quad \sum_{i=1}^{j} \mathit{duration}(\mathit{label}_i) = t \Big\} \tag{4}$$

with "initialized trajectory" and "duration" as defined there. ◁

Definition 3.3 (Generalized Notation for the Reachable Set)**.** To extend the previous notation, denote the reachable set of a function $z = h(x)$ of the state vector x as

$$R_z(H,t) := \{h(\xi) \mid \xi \in \mathit{Reach}_x(H,t) \cap \operatorname{dom} h\}. \tag{5}$$

Herein, $\operatorname{dom} h \subseteq \mathbb{R}^n$ denotes the domain of h, i.e., undefined results are excluded.

The reachable set over a time range $T \subseteq [0, \infty)$ is defined as

$$\mathit{Reach}_x(H,T) := \bigcup_{t\in T} \mathit{Reach}_x(H,t), \tag{6}$$

and the reachable set over all time as

$$Reach_x(H) := Reach_x(H, [0, \infty)). \tag{7}$$

◁

Comparison to Linear Impulsive Systems In hybrid automata, time does not elapse during discrete transitions. Therefore, if a discrete transition occurs at t_1, "$x(t_1)$" is ambiguous as it refers to both the value x^- before and x^+ after a transition. Formally, the set $Reach_x(H, t_1)$ will then have (at least) two elements x^- and x^+. In this respect, hybrid automata are different from Linear Impulsive Systems (LIS), where, for a typical definition

$$\dot{x}(t) = f(x), \quad t \neq t_i, \tag{8a}$$
$$x(t_i) = g_i(x(t_i^-)), \quad i \in \mathbb{N}, \tag{8b}$$

$x(t_1)$ is the unique value after the transition, while the value before is referenced by the left-side limit $x(t_1^-)$. While this notation may be more intuitive, it is no longer possible if multiple transitions occur at one time instant, e.g., if $t_1 = t_2$. For example, LIS cannot be directly used for systems with multiple asynchronous sensors [8, Section 6.1].

Definition 3.4 (Continuous Equivalence)**.** Two hybrid automata H_1, H_2 are x-equivalent ($H_1 =_x H_2$) iff the corresponding reachable sets for $x(t)$ are the same for all times t:

$$H_1 =_x H_2 \quad :\Leftrightarrow \quad Reach_x(H_1, t) = Reach_x(H_2, t)\ \forall t \tag{9}$$

Matching (5), this notation includes the case that H_1 and H_2 have different state vectors x and $\tilde{x}$ with a given, possibly non-invertible transformation $x = q(\tilde{x})$. ◁

Loosely speaking, x-equivalence considers all discrete transition labels and locations as hidden internal behavior and only compares for equality of $x(t)$. Next, we define a weakened variant of this relation by replacing "=" with "⊇":

Definition 3.5 (Continuous Abstraction)**.** A hybrid automaton H_2 x-abstracts a hybrid automaton H_1 ($H_2 \supseteq_x H_1$) iff all values of the continuous variable x reachable by H_1 are also reachable by H_2 at the same time t:

$$H_1 \subseteq_x H_2 : \quad \Leftrightarrow \quad H_2 \supseteq_x H_1 \quad :\Leftrightarrow \quad Reach_x(H_2, t) \supseteq Reach_x(H_1, t)\ \forall t \tag{10}$$

◁

Every automaton H_2 with $H_2 \supseteq_x H_1$ is a safe, i.e., pessimistic, approximation of H_1: If H_2 stays within a given safe x-region, then H_1 will certainly do, while the converse is not guaranteed.

Lemma 3.6 (Operator Properties of Continuous Equivalence and Abstraction)**.** *The operators $\subseteq_x$, $=_x$ and $\supseteq_x$ obey the same rules as $\subseteq$, $=$ and $\supseteq$. In particular,*

$$H_1 \subseteq_x H_2 \subseteq_x H_3 \quad \Rightarrow \quad H_1 \subseteq_x H_3, \tag{11}$$
$$H_1 =_x H_2 \quad \Leftrightarrow \quad H_1 \subseteq_x H_2 \ \wedge\ H_2 \subseteq_x H_1 \tag{12}$$
$$\textit{and} \quad H_1 =_x H_2 \quad \Leftrightarrow \quad H_2 =_x H_1. \tag{13}$$

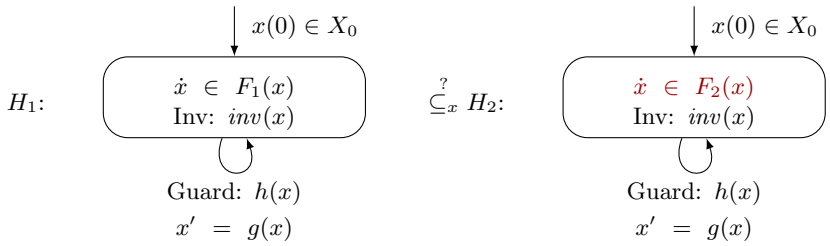

Figure 2: Automata of Theorem 3.7

3.3 Abstraction of Hybrid Automata Within Bound

The point of Continuization is to avoid the expensive analysis of the original automaton and instead only consider a simplified variant. However, as will be seen later, this simplification requires a bound on the reachable states, which at first leads to a chicken-and-egg problem: For simple analysis, a bound is required, which in turn requires the analysis result. To break this circle, [4] uses a scheme that can be summarized as follows:

1. A complicated automaton H_1 is abstracted (simplified) under the assumption that the state remains in a certain set X.

2. An outer bound Y of the reachable set of the simplified automaton H_2 is computed.

3. The assumption is validated only by checking if Y is "smaller than" X, without knowledge of the original reachable set R. If $Y < X$, then $R < Y < X$. Here, "$<$" denotes an appropriate condition for "smaller than", possibly stricter than $\subseteq$.

While this may seem trivially true at first sight, some caution is required to avoid circular reasoning. For example, consider the following naive claim: 1. Assume $R \subseteq X$. 2. Analysis under this assumption yields Y. 3. Claim: If $Y \subseteq X$, then $R \subseteq Y \subseteq X$.

However, this is a false conclusion: 1. Under the assumption $R \subseteq X$, a trivial abstraction of any H_1 is $H_2 =$"$x(t) \in X$". 2. This yields $Y = X$. 3. As then $Y \subseteq X$, the above claim would state that $R \subseteq X$. This result is clearly wrong, as one could show any bound X for any automaton. A correct and formally based version is given in the following theorem:

Theorem 3.7. *Let H_1, H_2 be given from Fig. 2, $X \subseteq \mathbb{R}^n$ be a set and $\epsilon > 0$ such that*

$$F_1(x) \subseteq F_2(x) \quad \forall x \in X \oplus B_\epsilon \quad \text{and} \tag{14}$$

$$\underbrace{Reach_x(H_2)}_{Y} \subseteq X. \tag{15}$$

Then, $H_1 \subseteq_x H_2$, so $\underbrace{Reach_x(H_1)}_{R} \subseteq \underbrace{Reach_x(H_2)}_{Y}$.

Proof Sketch. The proof uses the continuity of continuous evolution. In graphical terms, x in H_1 cannot escape X without passing through the "buffer zone" of width ϵ around X. As the abstraction is still valid there, this escape would be caught in $Reach_x(H_2)$. Appendix A.2 contains the full proof as well as a counterexample for $\epsilon = 0$. □

Usage: Given H_1, choose a set (e.g., interval) X, preferably as a rough guess of $Reach_x(H_1)$ from numerical simulations. Apply the broadening from $F_1(x)$ to $F_2(x)$. For example,

$$F_1(x) = \{-x + \sin(x)\} \quad \rightsquigarrow \quad F_2(x) = \{-x\} \oplus \{\sin(\xi) \mid x_{\min} - \epsilon \leq \xi \leq x_{\max} + \epsilon\} \tag{16}$$

is a valid broadening for $X = [x_{\min}; x_{\max}]$ and $\epsilon > 0$. Analyze the resulting simplified automaton H_2 to check whether $Reach_x(H_2) \subseteq X$. If the condition is true, then $Reach_x(H_1) \subseteq Reach_x(H_2) \subseteq X$. Else, one can try again with X changed to an enlarged version of $Reach_x(H_2)$, such as $X = Reach_x(H_2) \oplus B_r$ with some bloating radius $r \geq 0$.

4 Continuization

The idea of Continuization is to abstract a system with mixed discrete and continuous behaviour by a purely continuous approximation plus an error bound [2, 4]. For a controller that is designed in continuous time and then discretized for implementation, Continuization can be loosely interpreted as "inverting" this discretization.

4.1 Zero-Order Continuization

A first variant of Continuization given by Bak and Johnson in [4] applies to a control loop with the physical plant

$$\dot{x}_p(t) = f_p(x_p(t), x_c(t)) \tag{17}$$

and a *zero-order-hold* state feedback controller

$$x_c(t) = f_c(x_p(kT)), \quad kT \leq t < (k+1)T \tag{18}$$

with fixed period T. The corresponding hybrid automaton H_0 is shown in Fig. 3 (top left). (Note that the original publication [4] permits that f_c may additionally depend on x_c. However, as will be discussed later, this turns out to be infeasible.)

As illustrated in Fig. 4, the key idea is that zero-order sample-and-hold with high sampling rate approximates a direct connection plus some small error. Here,

$$x_c(t) = f_c(x_p(kT)) \approx f_c(x_p(t)). \tag{19}$$

The approximation error $\omega := x_c - f_c(x_p)$ can be bounded by, loosely speaking, the product of the period T and the maximum time derivative of $-f_c(x_p(t))$. For the following explanation, assume that a bound $\omega \in \Omega$ is known. Then, substituting $x_c = f_c(x_p) + \omega$ and $\omega \in \Omega$ in (17) yields a purely continuous-time system

$$\dot{x}_p(t) \in \{f_p(x_p(t), f_c(x_p(t)) + \omega) \mid \omega \in \Omega\} \tag{20}$$

for x_p with bounded disturbance, hence the name *Continuization*. Analysis of this new system is typically easy compared to the original automaton with its mixture of discrete and continuous-time dynamics. With minor modifications, the approach extends to more realistic timing schemes involving uncertain periods or skipped executions [4], where analyzing the original automaton would be even more difficult. However, for simplicity, this publication will only consider the periodic case. A second simplifying assumption in the following derivations, however without loss of generality, is that the initial state is exactly known and there is no disturbance, so that the behavior of the control loop is uniquely determined.

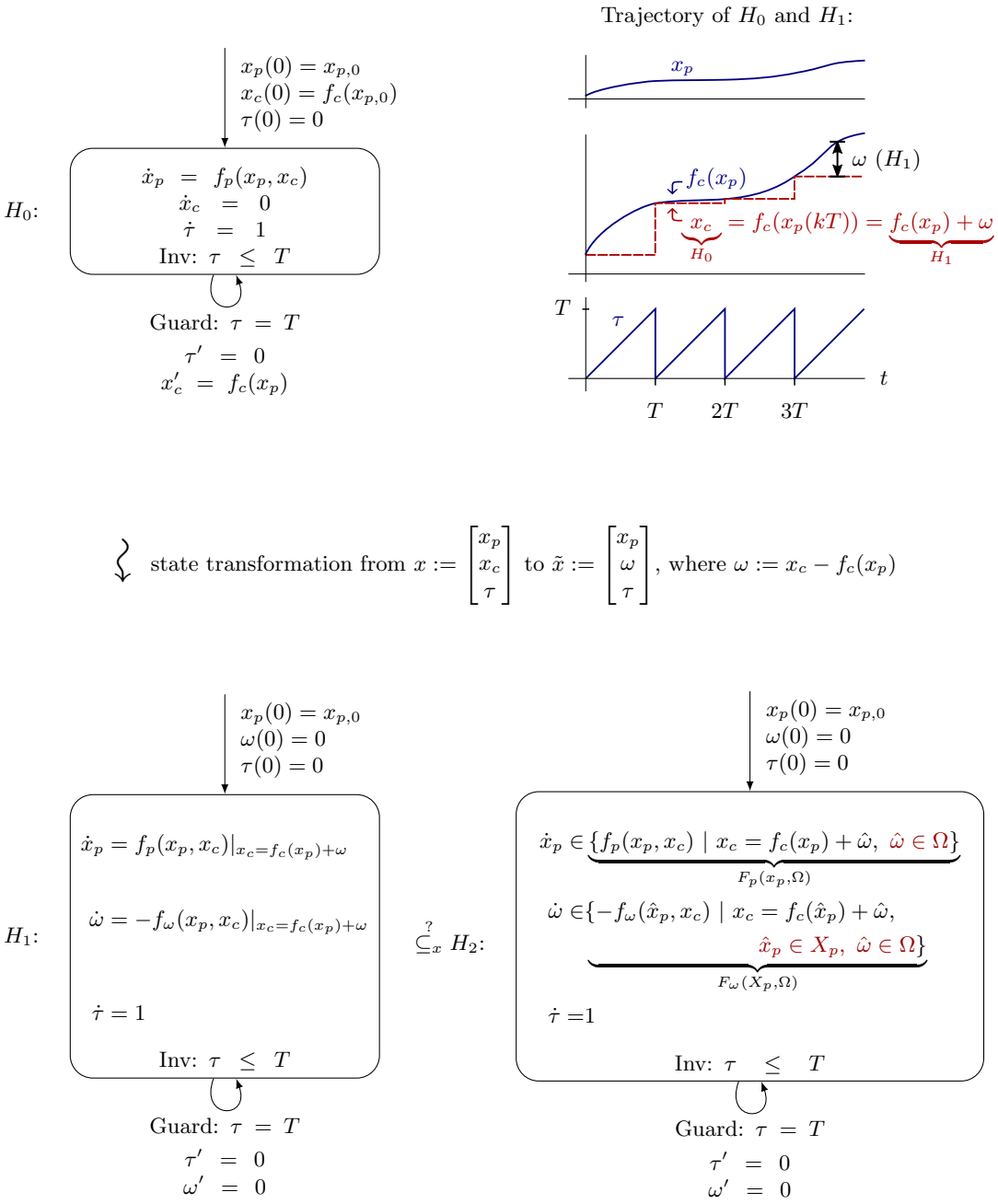

Figure 3: Hybrid automata of control loop and continuization in Theorem 4.1

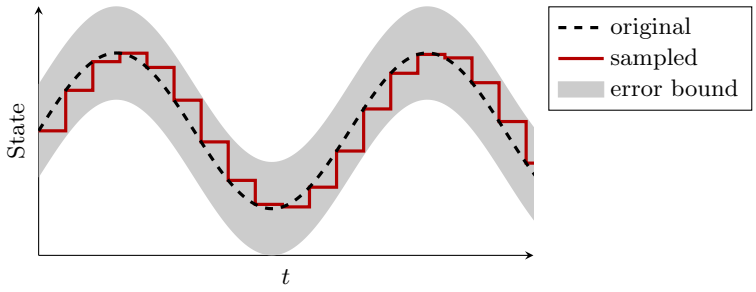

Figure 4: Key idea of Zero-Order Continuization: A sample-and-hold signal can be approximated by its continuous original plus a bounded error.

Formal Derivation: The above idea is now implemented using the concepts of continuous-variable equivalence and bounded abstractions outlined in Section 3. Consider the controller given by H_0 in Fig. 3. Transforming the state vector from $x = \begin{bmatrix} x_p^\top & x_c^\top & \tau \end{bmatrix}^\top$ to $\tilde{x} = \begin{bmatrix} x_p^\top & \omega^\top & \tau \end{bmatrix}^\top$ yields H_1, which uses the error $\omega := x_c - f_c(x_p)$ instead of the sampled state x_c. This is illustrated by the trajectory in Fig. 3 (top right). Assuming that the derivative $\partial f_c(x_p)/\partial x_p$ exists, the dynamics of ω are

$$\dot{\omega} = \dot{x}_c - \frac{\partial f_c(x_p)}{\partial x_p}\dot{x}_p = -\frac{\partial f_c(x_p)}{\partial x_p} f(x_p, x_c) = \underbrace{-\frac{\partial f_c(x_p)}{\partial x_p} f(x_p, f_c(x_p) + \omega)}_{=:-f_\omega(x_p, f_c(x_p)+\omega)}, \tag{21}$$

except for the discrete transitions at $t = kT$. The negative sign of f_ω is chosen to simplify a comparison with [4]. The remaining dynamics and transitions follow immediately from the state transformation. Consequently, $H_0 =_x H_1$ with

$$x = q(\tilde{x}) := \begin{bmatrix} x_p^\top & (f_c(x_p) + \omega)^\top & \tau \end{bmatrix}^\top. \tag{22}$$

Next, Theorem 3.7 is applied to H_1, which has

$$\tilde{X}_0 = \{\begin{bmatrix} x_p(0)^\top & 0^\top & 0 \end{bmatrix}^\top\}, \tag{23a}$$

$$F_1(\tilde{x}) = \left\{ \begin{bmatrix} f_p(x_p, f_c(x_p) + \omega) \\ -f_\omega(x_p, f_c(x_p) + \omega) \\ 1 \end{bmatrix} \right\}, \tag{23b}$$

$$h(\tilde{x}) = (\tau = T), \tag{23c}$$

$$g(\tilde{x}) = \begin{bmatrix} x_p^\top & 0^\top & 0 \end{bmatrix}^\top, \tag{23d}$$

$$inv(\tilde{x}) = (\tau \leq T). \tag{23e}$$

Choose the sets $\underline{X}_p$ and $\underline{\Omega}$ as a (possibly wrong) guess for the bounds of x_p and ω. Define the bloated bounds

$$X_p = \underline{X}_p \oplus B_\epsilon \tag{24}$$

$$\Omega = \underline{\Omega} \oplus B_\epsilon \tag{25}$$

and the combined, non-bloated bound

$$\tilde{X} = \underline{X}_p \times \underline{\Omega} \times \mathbb{R}. \tag{26}$$

Now, to get from H_1 to H_2 (Fig. 3, bottom right), the dependency of $\dot{x}_p$ on ω is replaced with the bound $\omega \in \Omega$:

$$\dot{x}_p \in \{ f_p(x_p, f_c(x_p) + \hat{\omega}) \mid \hat{\omega} \in \Omega \} =: F_p(x_p, \Omega). \tag{27}$$

Also, all dependencies of $\dot{\omega}$ are also replaced with their bounds:

$$\dot{\omega} \in \{ -f_\omega(\hat{x}_p, f_c(\hat{x}_p) + \hat{\omega}) \mid \hat{\omega} \in \Omega, \hat{x}_p \in X_p \} =: F_\omega(X_p, \Omega). \tag{28}$$

Thereby, F_1 is broadened to

$$F_2(\tilde{x}) = F_p(x_p, \Omega) \times F_\omega(X_p, \Omega) \times \{1\}. \tag{29}$$

(It is also valid to use an outer approximation of F_2, e.g., as interval). By construction,

$$F_1(\tilde{x}) \subseteq F_2(\tilde{x}) \quad \forall \tilde{x} \in \tilde{X} \oplus B_\epsilon, \tag{30}$$

so the only missing condition for Theorem 3.7 is $Reach_{\tilde{x}}(H_2) \subseteq \tilde{X}$. The automaton H_2 (Fig. 3) consists of three mostly separate subsystems, so its reachable set can be computed as follows:

- $x_p(t)$ is independent of τ and ω and follows the continuous-time differential inclusion

  $$\dot{x}_p \in F_p(x_p, \Omega), \quad x_p(0) = x_{p,0}. \tag{31}$$

  Therefore it is within

  $$X'_p := Reach_{x_p}\left(\begin{Bmatrix} \dot{x}_p \in F_p(x_p, \Omega), \\ x_p(0) = x_{p,0} \end{Bmatrix} \right). \tag{32}$$

- $\tau(t) = t \bmod T$ is reset to zero every T seconds, so $\tau \in [0; T]$.

- $\omega(t)$ is also reset to zero every T seconds. Inbetween it grows by

  $$\dot{\omega} \in F_\omega(X_p, \Omega), \tag{33}$$

  so, for $kT < t < (k+1)T$,

  $$\omega(t) = \int_{kT}^{t} \dot{\omega}(\tau)\, \mathrm{d}\tau \overset{\text{Lemma A.5 (Appendix A.3)}}{\in} [0; T] \otimes \square F_\omega(X_p, \Omega) =: \Omega'. \tag{34}$$

If $X'_p \subseteq \underline{X}_p$ and $\Omega' \subseteq \underline{\Omega}$, then, by $[0; T] \subseteq \mathbb{R}$, the condition $Reach_{\tilde{x}}(H_2) \subseteq \tilde{X}$ of Theorem 3.7 is fulfilled and therefore, $H_1 \subseteq_{\tilde{x}} H_2$. As there is a mapping $x = q(\tilde{x})$, also $H_1 \subseteq_x H_2$. □

The result is summarized by the following theorem:

Theorem 4.1 (Zero-Order Continuization)**.** *From Fig. 3, consider the control loop H_0 with state vector x and its modifications H_1 and H_2 with state vector $\tilde{x}$. Let f_c be a differentiable function. Then, $H_0 =_x H_1$ with*

$$x = q(\tilde{x}) := \begin{bmatrix} x_p^\top & (f_c(x_p) + \omega)^\top & \tau \end{bmatrix}^\top. \tag{35}$$

Also, $H_1 \subseteq_x H_2$ if $X'_p \subseteq \underline{X}_p$ and $\Omega' \subseteq \underline{\Omega}$. Herein, $\underline{X}_p$ and $\underline{\Omega}$ are arbitrarily chosen and represent a (possibly wrong) guess for the set of x_p and ω. X'_p and Ω' are determined by (32) and (34), and all remaining variables are as given in the above derivation.

Proof. See the above derivation. □

Remark 4.2 (Iterative Analysis). If the condition does not hold, one can retry with an enlarged version of the result as new assumption, e.g., $\underline{X}_p = X'_p \oplus B_r$ and $\underline{\Omega} = \Omega' \oplus B_r$ with an arbitrary bloating radius $r \geq 0$. ◁

Remark 4.3 (Comparison with Prior Work). Theorem 4.1 mostly, but not exactly matches the results of [4]. Particularly, [4] also permits that f_c depends on x_c, which will be discussed in the following chapter. The remaining differences are less severe and can be judged as typing errors with high certainty, as detailed in Appendix B. ◁

Remark 4.4. As X'_p can be determined without knowing $\underline{X}_p$, it is possible to avoid guessing $\underline{X}_p$: First, choose $\underline{\Omega}$ and compute X'_p. Then, use $\underline{X}_p = X'_p$ to finally determine Ω'. ◁

Lemma 4.5. *If the conditions of Theorem 4.1 hold, the following statements are true:*

- (31) *is an abstraction of* $x_p(t)$.
- $x_p(t) \in X'_p$ *for all* t.
- $\omega(t) \in \Omega'$ *and therefore* $x_c(t) = f_c(x_p(t)) + \omega(t) \in \{f_c(x_p(t))\} \oplus \Omega'$ *for all* t.

Conjecture 4.6. *There exist stable control loops (of the form* H_0 *per Fig. 3) whose stability can never be shown by Theorem 4.1. This particularly affects control loops whose discrete-time execution differs significantly from quasi-continuous, i.e., arbitrarily fast, execution.*

4.2 First-Order Continuization

The method of Zero-Order Continuization introduced in the previous section is appropriate for static state-feedback controllers, i.e., if the control signal u

$$u[k] = f(x_p[k]) \tag{36}$$

is determined only from the plant state x_p, where $[k]$ denotes the discrete-time value obtained at $t = kT$. Zero-Order Continuization approximates arbitrarily fast operation of the controller, i.e., $T \to 0$ with unchanged f. However, this approximation makes no sense if the controller has internal dynamics such as a state observer or, for example, an integral part

$$x_c[k+1] = \underbrace{x_p[k] + x_c[k]}_{f_c(x_p, x_c)} \approx \frac{1}{T} \int x_p \, \mathrm{d}t, \tag{37a}$$

$$u[k] = K x_c[k]. \tag{37b}$$

Here, $T \to 0$ leads to an infinite speed of integration, so Zero-Order-Continuization fails. The contrary is implied by [4, Theorem 1], where f_c may depend on x_c, however, the counterexample in Appendix B suggests this is rather a typing error.

As a remedy, the idea of Continuization is extended to a *first-order*-hold approximation, as sketched in Fig. 5: Following the idea of [9, Section 4.1], a linear interpolation $x_{int}(t)$ between neighboring samples $x_c(kT)$ and $x_c((k+1)T)$ is constructed and then differentiated:

$$x_{int}(t) := (1 - \alpha(t))\, x_c[k] + \alpha(t)\, x_c[k+1], \quad \alpha(t) := \frac{t - kT}{T}, \quad kT \leq t \leq (k+1)T \tag{38}$$

$$\Rightarrow \quad \dot{x}_{int} = \frac{1}{T}(x_c[k+1] - x_c[k]) = \frac{1}{T}(f_c(x_c[k], x_p[k]) - x_c[k]). \tag{39}$$

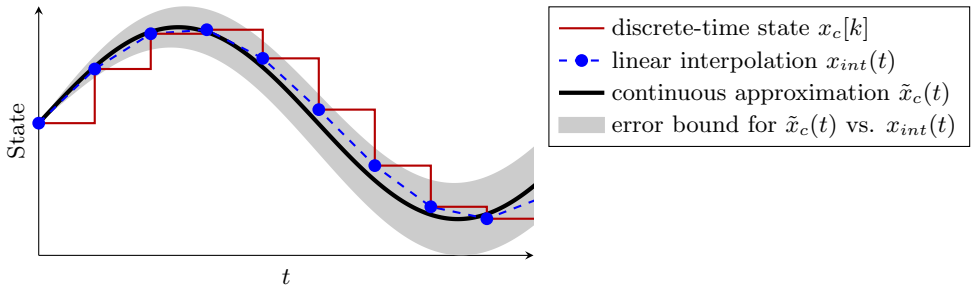

Figure 5: Principle of First-Order Continuization: A discrete-time system is smoothed by linear interpolation and then approximated as a continuous-time differential equation. The error between the continuous approximation and the original interpolation is modeled as disturbance.

Assuming that x_c and x_p change slowly leads to approximate continuous-time dynamics

$$\dot{\tilde{x}}_c \approx \frac{1}{T}(f_c(\tilde{x}_c, x_p) - \tilde{x}_c), \tag{40}$$

whose solution $\tilde{x}_c$ closely resembles the linear interpolation x_{int} of the discrete-time signal x_c. For the exemplary integral controller (37), this yields

$$\dot{\tilde{x}}_c \approx \frac{1}{T}x_p, \tag{41}$$

which recovers the desired integrator dynamics: The discrete-time integrator has been continuized. Formal application of this idea requires an appropriate error bound, for which the difference between $\dot{\tilde{x}}_c$ and $\dot{x}_{int}$ is considered as a disturbance on the continuous dynamics.

Theorem 4.7 (First-Order Continuization)**.** *From Fig. 6, consider the control loop H_0 with state vector x and its modifications H_1 and H_2 with state vector $\tilde{x}$. (Unlike in Theorem 4.1, there are no specific requirements on f_c.)*

Then, $H_0 =_x H_1$ with $x = q(\tilde{x})$. Additionally, $H_1 \subseteq_x H_2$ if $X'_{pc} \subseteq \underline{X}_{pc}$ and $\Delta' \subseteq \underline{\Delta}$. Herein, $\underline{\Delta}$ and $\underline{X}_{pc}$ are arbitrarily chosen and represent a (possibly wrong) guess for the set of $\delta_{pc} := \begin{bmatrix}\delta_p^\top & \delta_c^\top\end{bmatrix}^\top$ and $x_{pc} := \begin{bmatrix}x_p^\top & \tilde{x}_c^\top\end{bmatrix}^\top$. X'_{pc} and Δ' are defined as

$$Reach_{x_{pc}}(H_2) = Reach_{x_{pc}}\begin{pmatrix}\dot{x}_{pc} \in F_{pc}(x_{pc}, \Delta)\\ x_p(0) = x_{p,0}\\ \tilde{x}_c(0) = x_{c,0}\end{pmatrix} =: X'_{pc}, \tag{42}$$

$$Reach_{\delta_{pc}}(H_2) = Reach_{\delta_{pc}}\left(\begin{pmatrix}\dot{\delta}_{pc} \in F_\delta(X_{pc}, \Delta)\\ \delta_{pc}(0) = 0\end{pmatrix}, [0;T]\right) \subseteq [0;T] \otimes \square(F_\delta(X_{pc}, \Delta)) =: \Delta', \tag{43}$$

where $X_{pc} := \underline{X}_{pc} \oplus B_\epsilon$, $\Delta := \underline{\Delta} \oplus B_\epsilon$, and F_{pc}, F_δ are given in H_2 in Fig. 6.

Proof Sketch. $H_0 =_x H_1$ with the coordinate transformation $x = q(\tilde{x})$ can be shown by considering a trajectory of H_1, as sketched in Fig. 6 (top right). By construction of H_1, $x_p^- := x_p + \delta_p$ is the newest sample of $x_p(kT)$ and $x_c^- := \tilde{x}_c + \delta_c$ is the previous sample of x_c, so that $f_c(x_p^-, x_c^-)$ reconstructs the current sample of x_c. Additionally, $\tilde{x}_c$ is a linear interpolation between the samples of x_c with a time delay of T. A detailed proof is detached to Appendix A.4.

The proof concering $H_1 \subseteq_x H_2$ is analogous to Theorem 4.1 and therefore omitted. □

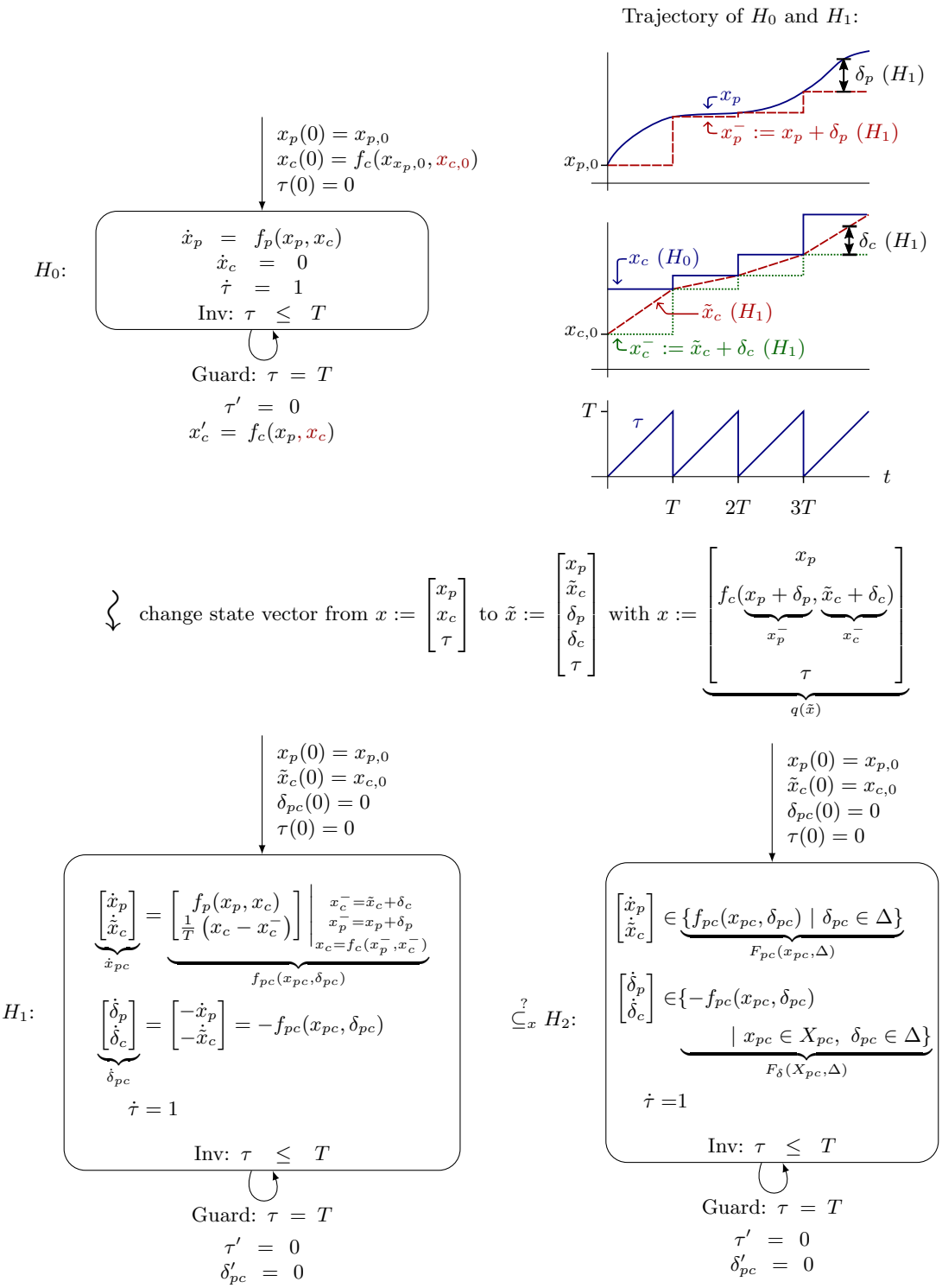

Figure 6: Hybrid automata of control loop and continuization in Theorem 4.7

5 Experiments

This section evaluates an implementation of First-Order Continuization, using SpaceEx [7] and HYST [3] for reachability analysis and the Python mpmath library [11] for interval arithmetic. Open-source code is available at https://github.com/qronos-project/timing-stability-lmi/.

In the experiments, the linear case is considered, which can be summarized as

$$\dot{x}_p = f_p(x_p, x_c) = A_p x_p + B_{pc} x_c, \tag{44a}$$
$$x_c' = f_c(x_p, x_c) = B_{cp} x_p + A_c x_c \quad \text{at } t = kT. \tag{44b}$$

Zero-Order Continuization is not evaluated, as this was already done in [4]. The implementation mostly follows Section 4.2; further details are given in Appendix C.

The following examples are based on Example C1 from [9], which represents a part of the angular rate control of a quadcopter: The angular rate x_p with dynamics

$$\dot{x}_p = u/J_x, \quad y = x_p, \quad x_p(0) \in [-1; 1], \quad J_x = 9.0359 \cdot 10^{-6} \tag{45}$$

and control input u is controlled by a PI-controller approximating $u = -K_P x_p - K_I \int x_p \, \mathrm{d}t$ with $K_P = 3.6144 \cdot 10^{-3}$ and $K_I = 2.5557 \cdot 10^{-4}$. The controller period $T = 0.01$ is chosen such that the behavior differs from the continuous equivalent by about 20 percent [9, Fig. 7]. The discretized implementation incurs a one-period processing delay from x_p to u. However, the formulation used here in Section 4.2 does not directly admit this delay, as there is no extra sample-and-hold mechanism between $x_c(t)$ and the plant input $u(t)$. To avoid extra delay states, the example is simplified by dropping the delay:

Example C3 (PI discrete). Choosing $x_{c,1} \approx \int x_p \, \mathrm{d}t$ and $x_{c,2} \approx x_p$ leads to

$$A_p = 0, \qquad B_{pc} = \frac{1}{J_x} \begin{bmatrix} -K_I & -K_P \end{bmatrix}, \qquad B_{cp} = \begin{bmatrix} T \\ 1 \end{bmatrix}, \qquad A_c = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{46}$$

Continuization of this system fails: No valid error bound can be determined, as the iteration does not converge even for small T ($T = 0.001$, whereas the original example C1 has $T = 0.01$). Presumably this is due to the zero eigenvalue of A_c, as this discrete-time dead-beat behavior has no proper continuous equivalent. This variant is therefore excluded from further analysis.

Example C4 (P continuous, I discrete). As a simple academic example, the proportional part of the controller is changed to continuous time and merged with the plant. Note that this is not equivalent to the previous example. The remaining controller is a discrete-time integrator.

$$A_p = \frac{-K_P}{J_x}, \qquad B_{pc} = \frac{-K_I}{J_x}, \qquad B_{cp} = T, \qquad A_c = 1, \qquad T = 0.003 \tag{47}$$

For $T = 0.003$, Continuization shows stability and reasonable bounds, as will be illustrated later. For larger periods, the bounds grow until Continuization starts to fail near $T = 0.01$.

Example C5 (P as discrete lowpass, I discrete). As a more realistic variant, the proportional feedback is lowpass-filtered in discrete time with an equivalent time constant of $T_L = 0.1$:

$$A_p = 0, \quad B_{pc} = \frac{1}{J_x} \begin{bmatrix} -K_I & -K_P \end{bmatrix}, \quad B_{cp} = \begin{bmatrix} T \\ 1 - e^{-T/T_L} \end{bmatrix}, \quad A_c = \begin{bmatrix} 1 & 0 \\ 0 & e^{-T/T_L} \end{bmatrix}, \quad T = 0.001 \tag{48}$$

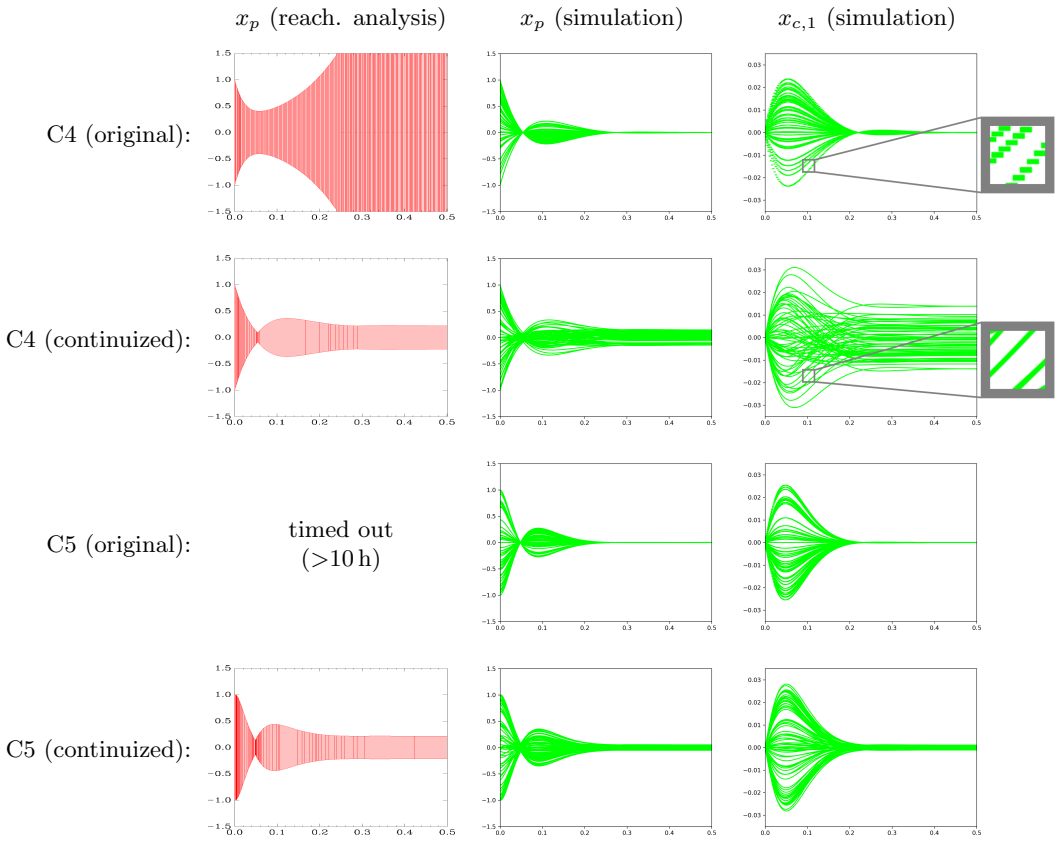

Figure 7: Reachability analysis and randomized simulations for the example systems

For $T_L \to 0$, this would be equivalent to Example C3. With the smaller period of $T = 0.001$, results are similarly positive as in the previous example.

To illustrate the results, the behavior of the original and continuized systems for C4 and C5 is shown in Fig. 7 for a horizon of $t = 0.5$ seconds. The left column shows the outer approximation of the reachable set $Reach_{x_p}(H_i, t)$ of the plant state $x_p(t)$ determined using SpaceEx. The remaining columns show randomized simulations of x_p (middle) and the first controller state $x_{c,1}$ (right) using PySim from the HYST/Hypy toolset. It can be seen that the original systems are stable but ill-suited for reachability analysis: The approximated reachable set diverges (C4) or cannot be computed within ten hours (C5), although the simulations indicate stability. It should be noted that because this paper is restricted to strictly periodic controllers, it should still be possible to solve these systems without Continuization with some analysis tools and optimized parameters. Mainly, one could exploit that the transitions occur at fixed times. However, this benefit disappears for an extension to uncertain timing [9].

After Continuization, reachability analysis is easy: SpaceEx returns a useful result with little pessimism compared to the simulations. The only downside is the pessimism incurred by Continuization itself. Particularly, some steady-state uncertainty remains, while the original system converges to zero. This is because the assumed disturbance bound is constant over time, which could however be mitigated by "restarting" with a lower disturbance bound after some time has passed [4, Lemma 2].

6 Conclusion and Outlook

Results This paper considers the analysis of digital control loops whose behavior is similar to a continuous-time differential equation. This paper presented a new formal framework for Continuization, a method to determine this continuous equivalent and use it for analysis. Two variants were derived: *Zero-Order Continuization* is equivalent to previous work by Bak and Johnson. Unlike assumed previously, this method is limited to static controllers, i.e., controllers with only feedthrough and no dynamics. As a complement, the novel method of *First-Order Continuization* is presented. For the first time, this method allows the Continuization of controllers with internal dynamics, e.g., an integral part or lowpass filter. Experiments show it can be successfully applied for dynamics controllers *without feedthrough* if the period is small enough to approximate continuous-time behavior. However, controllers with dynamics *and* feedthrough can not be solved yet.

Open Questions The results open up a number of questions and possible future developments: Is it possible to analyze controllers with dynamics *and* feedthrough, e.g., the common PI controller, by combining Zero- and First-Order Continuization? Can the fundamental restriction to small periods be reduced? A possible direction of research could be higher-order Continuization, analogous to higher-order (e.g., Runge-Kutta) numerical integration methods.

For the sake of brevity, this paper only considered strictly periodic execution. An extension to relaxed timing schemes is of high practical importance, e.g., to uncertain periods [4] or uncertain input/output timing [9], as these are difficult or impossible to analyze with other methods. The former was already solved in [4] for Zero-Order Continuization. Both cases should be solvable in the formal framework presented here by adding additional sample-and-hold states analogous to δ_p and δ_c.

Regarding the class of systems, the presented formal framework should allow for generalization with little effort: The theory already covers nonlinear dynamics, and the results should hold unchanged for the case of bounded disturbance. Similarly, the results should extend to hybrid plants and controllers (cf. Conjecture A.4). For these hybrid systems, reachability analysis as employed here is particularly suitable, while other methods such as Lyapunov functions are rather difficult.

In summary, First-Order Continuization and the underlying formal framework are a promising step towards the verification of real-time control systems with uncertain timing.

References

[1] Mohammad Al Khatib, Antoine Girard, and Thao Dang. Stability verification and timing contract synthesis for linear impulsive systems using reachability analysis. *Nonlinear Analysis: Hybrid Systems*, September 2016.

[2] Matthias Althoff, Soner Yaldiz, Akshay Rajhans, Xin Li, Bruce H. Krogh, and Larry Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, November 2011.

[3] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST. In *Proceedings of the 18th International Conference on Hybrid Systems Computation and Control - HSCC '15*. ACM Press, 2015.

[4] Stanley Bak and Taylor T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *2015 IEEE Real-Time Systems Symposium*. IEEE, December 2015.

[5] Viktorio Semir el Hakim and Marco Jan Gerrit Bekooij. Stability verification of self-timed control systems using model-checking. In *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, August 2018.

[6] Goran Frehse. An introduction to hybrid automata, numerical simulation and reachability analysis. In *Formal Modeling and Verification of Cyber-Physical Systems*, pages 50–81. Springer Fachmedien Wiesbaden, 2015.

[7] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer Berlin Heidelberg, 2011.

[8] Maximilian Gaukler, Günter Roppenecker, and Peter Ulbrich. Details and proofs for: Stability analysis of multivariable digital control systems with uncertain timing, 2019.

[9] Maximilian Gaukler and Peter Ulbrich. Worst-case analysis of digital control loops with uncertain input/output timing. In *ARCH19. International Workshop on Applied veRification for Continuous and Hybrid Systems*, 2019.

[10] Thomas A. Henzinger. The theory of hybrid automata. In M. Kemal Inan and Robert P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, chapter 13, pages 265–292. Springer, 2000.

[11] Fredrik Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.1)*, December 2018. `http://mpmath.org/`.

A Formal Details and Proofs

A.1 Formal Definition of Hybrid Automata

Definition A.1 (Hybrid Automaton)**.** The semantics of hybrid automata in this paper are based on the work of Henzinger [10, Definitions 1.4 and 1.5], with some minor simplifications.

A hybrid automaton is defined by

- A finite set Loc of discrete-valued locations ("modes")
- A vector $x \in \mathbb{R}^n$ of continuous-valued state variables (abbreviated as "states" with some abuse of terminology)
- Discrete transitions ("jumps") from $i \in Loc$ to $j \in Loc$ with
  - a non-numeric transition label $\alpha_{i,j} \notin \mathbb{R}$, e.g., a letter of the alphabet, whose value is not further relevant in this paper,
  - a guard condition $guard_{i,j}(x) \in \{true, false\}$ to describe if the transition is enabled, and

  - a transition function $\mathit{trans}_{i,j}(x) \subseteq \mathbb{R}^n$ modeling the change of continuous variables due to the transition.

  The discrete transition from $x = x$ in location i to $x = x'$ in location j is possible (formally, the relation $(i, x) \xrightarrow{\alpha_{i,j}} (j, x')$ is true) iff $\mathit{guard}_{i,j}(x)$ is true and $x' \in \mathit{trans}_i(x)$.

- Continuous dynamics in a location i with
  - an invariant condition $\mathit{inv}_i(x) \in \{\mathit{true}, \mathit{false}\}$ and
  - a dynamics (flow) $\mathit{flow}_i(x, \dot{x}) \in \{\mathit{true}, \mathit{false}\}$, e.g., an ordinary differential equation.

  The continuous transition in location i from x to x' with duration $\delta \geq 0$ is possible (formally, the relation $(i, x) \xrightarrow{\delta} (i, x')$ is true) iff there is a solution (witness) $\xi(t) : [0; \delta] \mapsto \mathbb{R}^n$ such that
  - $\xi(0) = x$ and $\xi(\delta) = x'$,
  - $\mathit{flow}_i(\xi(t), \dot{\xi}(t))$ and $\mathit{inv}_i(\xi(t))$ are true on $(0; \delta)$ and
  - $\xi(t)$ is differentiable on $(0; \delta)$ and continuous on $[0; \delta]$.

  In particular, the invariant may be violated at the start and end of the transition, and $(i, x) \xrightarrow{0} (i, x)$ is always true. Note that here, t refers to the local time since the start of the transition.

- An initial set $\mathit{Init} \subseteq \mathit{Loc} \times \mathbb{R}^n$: Trajectories start at all $(l, x) \in \mathit{Init}$ with $\mathit{inv}_l(x) = \mathit{true}$. The restriction to $\mathit{inv}_l(x) = \mathit{true}$ serves no purpose in this paper, except that it ensures compatibility with Henzinger's definition.

A trajectory of the automaton is defined as a chain of transitions $(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} (l_n, x_n)$ of arbitrary length $n \geq 0$ with $(l_0, x_0) \in \mathit{Init}$ and $\mathit{inv}_{l_0}(x_0) = \mathit{true}$. The duration of this trajectory is defined as the sum

$$\mathit{duration} = \sum_{i=1}^{n} \begin{cases} \beta_i, & \beta_i \in \mathbb{R} \text{ (continuous transition)}, \\ 0, & \text{else (discrete transition)} \end{cases} \tag{49}$$

of the durations of all continuous transitions.

Should any of the involved functions become undefined, e.g., if *flow* contains a division by zero, then this is treated as *false* or the empty set $\{\}$, forcing an end of the trajectory (deadlock) if no other transition is possible. ◁

A.2 Bounded Abstractions: Theorem 3.7

Proof of Theorem 3.7. Assume the given conditions (14) and (15) are true. Consider any trajectory

$$(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_n} (l_n, x_n) \tag{50}$$

of H_1 to show by induction that every transition $(l_i, x_i) \xrightarrow{\beta_{i+1}} (l_{i+1}, x_{i+1})$ of this trajectory also exists in H_2 and consequently $\mathit{Reach}_x(H_2) \supseteq \mathit{Reach}_x(H_1)$.

Induction Assumption $IA(i)$**:** For given i, where $0 \leq i \leq n$, the first i transitions of the trajectory (50) of H_1 are also a valid trajectory of H_2, i.e.,

$$(l_0, x_0) \xrightarrow{\beta_1} (l_1, x_1) \xrightarrow{\beta_2} \ldots \xrightarrow{\beta_i} (l_i, x_i) \tag{51}$$

is also trajectory of H_2. This implies that $(l_0, x_0) \in \mathit{Init}(H_2)$ and $x_i \in \mathit{Reach}_x(H_2)$.

Start of Induction: H_1 and H_2 have the same initial set and invariant, so their trajectories start the same. Hence, $IA(0)$ is true.

Induction Step: Assume $IA(i)$ to show $IA(i+1)$: By $IA(i)$, $x_i \in \mathit{Reach}_x(H_2)$, so $x_i \in X$. Consider the following cases:

- The next transition $(l_i, x_i) \xrightarrow{\beta_i} (l_{i+1}, x_{i+1})$ is a discrete transition. Then, it is present both in H_1 and H_2, because the invariant does not affect the discrete transitions.

- The next transition $(l_i, x_i) \xrightarrow{\delta} (l_{i+1}, x_{i+1})$ is a continuous transition with duration $\delta \geq 0$. By the definition of continuous transitions, $l_i = l_{i+1}$ and there is a witness $\xi(t)$ defined on $0 \leq t \leq \delta$ such that $\xi(t)$ is continuous on $[0;\delta]$ and differentiable on $(0;\delta)$,

$$\xi(0) = x_i, \tag{52}$$
$$\xi(\delta) = x_{i+1}, \tag{53}$$
$$\dot{\xi}(t) \in F_1(\xi(t)) \quad \forall t \in (0;\delta) \quad \text{and} \tag{54}$$
$$\mathit{inv}(\xi(t)) = \mathit{true} \qquad \forall t \in (0;\delta). \tag{55}$$

  To show that $\xi(t)$ is a witness for the same transition in H_2, it must be shown that also

$$\dot{\xi}(t) \in F_2(\xi(t)) \quad \forall t \in (0;\delta). \tag{56}$$

  - If $\delta = 0$, the transition is the trivial transition $(l_i, x_i) \xrightarrow{0} (l_i, x_i)$, which is also present in H_2.
  - If $\delta > 0$ and $\xi(t) \in X \oplus B_\epsilon$ for all $t \in [0;\delta]$, then also $\dot{\xi}(t) \in F_1(\xi(t)) \subseteq F_2(\xi(t))$ for all $t \in (0;\delta)$. Consequently, $\xi(t)$ is witness of a continuous transition $(l_i, x_i) \xrightarrow{\delta} (l_i, x_{i+1})$ of H_2.
  - The remaining case is that $\delta > 0$ and $\xi(t) \notin X \oplus B_\epsilon$ for some $t \in [0;\delta]$. Informally, ξ has left X by a distance of more than ϵ. Then, there is a time of violation t_1 with $0 \leq t_1 \leq \delta$ such that $\xi(t_1) \notin X \oplus B_\epsilon$. By the induction assumption, $x_i \in \mathit{Reach}_x(H_2)$, so $\xi(0) \in \mathit{Reach}_x(H_2)$. With

$$\mathit{Reach}_x(H_2) \overset{(15)}{\subseteq} X \subset X \oplus B_\epsilon \tag{57}$$

    it follows that $\xi(0) \in X \oplus B_\epsilon$, so $t_1 > 0$.

    In summary, $\xi(t)$ is continuous, starts at $\xi(0) \in X$ and reaches $\xi(t_1) \notin X \oplus B_\epsilon$ at $t_1 > 0$. Consequently, as illustrated in Fig. 8, there is a time t_0 between 0 and t_1 such that ξ has already left X, but only by a distance of at most ϵ: There exists t_0 with $0 < t_0 < t_1$, $\xi(t_0) \notin X$ and $\xi(t) \in X \oplus B_\epsilon \forall t \in [0;t_0]$.

Consider the same $\xi(t)$ defined on a shortened timespan $0 \le t \le t_0$. As it fulfills all conditions given in the definition, it is witness of a shorter continuous transition $(l_i, x_i) \xrightarrow{t_0} (l_i, \xi(t_0))$ of H_1. Within that shorter timespan, $\xi(t)$ stays within $X \oplus B_\epsilon$, so it fulfills all conditions of the previous case for $\delta = t_0$. Therefore, H_2 has the same transition $(l_i, x_i) \xrightarrow{t_0} (l_i, \xi(t_0))$.

This leads to a contradiction: Because of this transition, $\xi(t_0) \in Reach_x(H_2)$. By the assumed condition (15) of the theorem, $Reach_x(H_2) \subseteq X$, so $\xi(t_0) \in X$. However, in the current case, $\xi(t_0) \notin X$, so the case is impossible.

Conclusion $IA(i)$ holds for all $i \ge 0$ up to the length n of the given trajectory of H_1. Therefore, any trajectory of H_1 is a trajectory of H_2, so $H_1 \subseteq_x H_2$. □

Remark A.2. The ϵ-boundary is required, which is illustrated by the following example with "impossible" flow:

$$X_0 = \{0\}, \qquad F_1(x) = \{1\}, \qquad X = [-1; 1], \qquad F_2(x) = \begin{Bmatrix} \{1\}, & x \le 1, \\ \{-1\}, & x > 1 \end{Bmatrix},$$
$$h(x) = \mathit{false}, \qquad g(x) = x, \qquad \mathit{inv}(x) = \mathit{true}. \tag{58}$$

H_1 is equivalent to $\dot{x} = 1$, $x(0) = 0$, which yields $x(t) = t$ for $t \ge 0$. H_2 has the solution $x(t) = t$ only within $0 \le t \le 1$. At $t = 1$, there is a deadlock, so the trajectories do not continue any further, as will be explained later. Consequently, $Reach_x(H_1) = [0; \infty)$ and $Reach_x(H_2) = [0; 1] \subseteq X$. However, for $\epsilon = 0$, the theorem would state that $H_1 \subseteq_x H_2$, which is clearly false.

To see the deadlock of H_2 at $t = 1$, consider all possible transitions starting from $x(1) = 1$. First, there are no possible discrete transitions. Second, there is a continuous transition of duration $\delta = 0$, but it has no effect on x and t, so it can be ignored. Lastly, there is also no continuous transition of duration $\delta > 0$: This would require a witness $\xi(t)$ with $\xi(0) = x(1) = 1$ that is differentiable on $(0; \delta)$ and fulfills $\dot{\xi}(t) \in F_2(\xi(t))$ on $(0; \delta)$. The derivative of a differentiable function is continuous, but $\dot{\xi} \in F_2(\xi) \subseteq \{+1, -1\}$, so the only two candidates for $\dot{\xi}(t)$ are the constant functions $\dot{\xi}(t) = \pm 1$. Consequently, the only two candidate witnesses are $\xi(t) = 1 \pm t$. Consider an arbitrary time $t \in (0; \delta)$ to see that neither candidate fulfills $\dot{\xi}(t) \in F_2(\xi(t))$: For the case "+", $\xi > 1$, so $\dot{\xi} = 1 \notin F_2(\xi) = \{-1\}$. For the case "−", $\xi < 1$, so $\dot{\xi} = -1 \notin F_2(\xi) = \{1\}$. Hence, there is no valid witness, so H_2 has no solutions beyond $t = 1$. ◁

Remark A.3. Theorem 3.7 makes no restrictions on the set X. The set may be non-connected, non-convex or could even contain isolated points. Similarly, no special restrictions apply on the dynamics $F(x)$, such as the existence or continuity of solutions. This generality is a benefit of defining hybrid automata by a chain of transitions. ◁

Conjecture A.4. *Theorem 3.7 can be generalized to hybrid automata with multiple modes in a way such that there is a different assumption for each mode. As a naive proof sketch, treat the mode number as an additional continuous state variable with zero derivative and then merge all modes into one whose flow, invariant, guard and transition depend on the mode number. Similarly, merge the set of assumptions by introducing the mode number as new dimension. Then apply the theorem in its current form.*

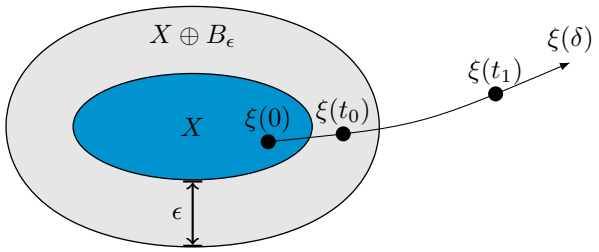

Figure 8: A continuous trajectory from inside X to outside $X \oplus B_\epsilon$ must first pass through the "boundary" $(X \oplus B_\epsilon) \setminus X$.

A.3 Zero-Order-Continuization

The following lemma is required for the proof of Theorem 4.1:

Lemma A.5. *Let $f : \mathbb{R} \mapsto \mathbb{R}^n$ be an integrable function and F be a set. If $f(t) \in F$ for all $t \in [0;T]$, then*

$$\int_0^t f(\tau)\,\mathrm{d}\tau \;\in\; [0;T] \otimes \operatorname{conv} F \quad \text{for all } t \in [0;T], \tag{59}$$

where conv *denotes the convex hull.*

Proof Sketch. Assume $t \in [0;T]$. Consider the integral as weighted sum of $n \to \infty$ summands,

$$\int_0^t f(\tau)\,\mathrm{d}\tau = t \underbrace{\sum_{i=0}^{n-1} \frac{1}{n} f\left(\frac{t}{n}\right)}_{(*)}. \tag{60}$$

In this product, the first factor t is in $[0;T]$ by assumption. The second factor $(*)$ is a convex combination of values of $f(\tau) \in F$, which by definition is inside the convex hull of F. □

For a fast pessimistic approximation, the convex hull can be replaced by the interval hull.

Remark A.6. To see that taking the convex hull (or interval hull) is necessary, consider

$$F = \left\{ \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix} \right\}, \quad f(t) = \begin{bmatrix}1\\0\end{bmatrix} \text{ for } 0 \le t < \frac{T}{2}, \quad f(t) = \begin{bmatrix}0\\1\end{bmatrix} \text{ for } \frac{T}{2} \le t \le T \tag{61}$$

and $T = 1$, for which

$$\int_0^T f(\tau)\,\mathrm{d}\tau = \frac{T}{2}\begin{bmatrix}1\\0\end{bmatrix} + \frac{T}{2}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\\frac{1}{2}\end{bmatrix} \tag{62}$$

is not contained in

$$[0;T] \otimes F = \left\{ \begin{bmatrix}t\\0\end{bmatrix}, \begin{bmatrix}0\\t\end{bmatrix} \;\middle|\; 0 \le t \le 1 \right\}. \tag{63}$$

◁

A.4 First-Order-Continuization

Proof that $H_0 =_x H_1$ in Theorem 4.7. Consider the value of $\tilde{x}$ along a trajectory of H_1 to show that there is a one-to-one correspondence $x = q(\tilde{x})$ (as given in Fig. 6 on page 220) with the state x along a trajectory of H_0. The following proof proceeds by induction, extending the trajectory by one transition in every induction step. For the sake of readability, the proof steps are not strictly formalized, but sketched in terms of $x(t)$ and $\tilde{x}(t)$. To clarify the meaning of $x(t)$ at discrete transitions, $x(t^-)$ refers to the value before and $x(t^+)$ to the value after a possible discrete transition at t. This notation is possible because at most one discrete transition occurs at a given time instant. As a special case, $t = 0^-$ refers to the initial state, and $t = 0^+$ to the state after a possible discrete transition following the initial state.

Assumption: Well-Behaved Automaton For simplicity, assume that in H_0

1. every trajectory can be extended to infinite duration,
2. every continuous transition can be extended to duration T and
3. all functions are defined globally.

This assumption excludes some problematic scenarios, but should be without loss of generality. As sketched in the following, if there is a "problem", it will be the same for H_0 and H_1, and except for that point in time, the proof can be reused.

1. Trajectories ending due to impossible flow (cf. H_2 in Remark A.2), finite escape time or a similar deadlock will admit the same effect in H_0 and H_1 at the same time.
2. Piecewise-continuous solutions due to discontinuous derivatives are built from piecing together multiple continuous transitions (cf. the definition in Appendix A.1). This is the same in H_0 as in H_1. For each piece, the proof can be reused.
3. Trajectories of H_0 ending due to a transition with undefined f_c will render the state transformation $q(\tilde{x})$ of H_1 undefined from that time on, as it also contains f_c. However, this is not a problem since the derivative in H_1 will also become undefined, so both automata reach a deadlock.

Induction Assumption $IA(k)$: For given $k \geq 0$, the trajectory is consistent until $t = kT^+$, i.e.,

$$x(t^+) = q(\tilde{x}(t^+)) \quad \text{for } 0 \leq t \leq kT, \tag{64}$$

$$x(t^-) = q(\tilde{x}(t^-)) \quad \text{for } 0 \leq t \leq kT. \tag{65}$$

H_0 and H_1 share the state names x_p and τ, though it must still be proven that these actually have equal value. For τ, it is obvious that both automata behave exactly the same. For x_p, however, this is not immediately clear, so its values will be distinguished by x_p^{H0} and x_p^{H1}.

Start of Induction ($k = 0$): At $t = 0$, the initialization is consistent, and no discrete transition occurs:

$$x(0^-) = x(0^+) = \begin{bmatrix} x_{p,0} \\ f_c(x_{x_p,0}, x_{c,0}) \\ 0 \end{bmatrix} = q(\tilde{x}_0) = q(\tilde{x}(0^-)) = q(\tilde{x}(0^+)). \tag{66}$$

Therefore, $IA(0)$ holds.

Induction Step: Assume $IA(k)$, where $k \geq 0$.

1. *Continuous Evolution from $t = kT^+$ to $t = (k+1)T^-$:* The first goal is to show that the continuous evolution is consistent per

$$x(t) = q(\tilde{x}(t)) \quad \text{for} \quad kT < t < (k+1)T. \tag{67}$$

This will be shown in two steps that are repeated for each component x_p, x_c and τ:

- The "initial condition" at $t = kT^+$ is consistent per $IA(k)$, which implies $x(kT^+) = q(\tilde{x}(kT^+))$.
- The flow is consistent as well, i.e.,

$$\dot{x} = \frac{\mathrm{d}q(\tilde{x}(t))}{\mathrm{d}t} \quad \text{for} \quad kT < t < (k+1)T. \tag{68}$$

(a) *Consistency of τ:* As noted before, τ is obviously equal in both automata.

(b) *Consistency of x_c:* The following steps show that for the evolution from $t = kT^+$ to $t = (k+1)T^-$, the state x_c in H_0 is consistent with its counterpart $f_c(x_p^{H1}+\delta_p, \tilde{x}_c+\delta_c)$ in H_1.

i. *Initial Condition:* For the initial condition, $IA(k)$ implies that

$$x_c(kT^+) = f_c(x_p(kT^+) + \delta_p(kT^+), \tilde{x}_c(kT^+) + \delta_c(kT^+)). \tag{69}$$

ii. *Flow:* Next, the time derivatives of x_c and its equivalent in H_1 are shown to be equal. For the continuous evolution between $t = kT^+$ and $t = (k+1)T^-$, the following holds: In H_0,

$$\dot{x}_c = 0, \tag{70}$$

and in H_1,

$$\left.\begin{aligned} \dot{x}_c^- &= \dot{\tilde{x}}_c + \dot{\delta}_c = 0 \\ \dot{x}_p^- &= \dot{x}_p^{H1} + \dot{\delta}_p = 0 \end{aligned}\right\} \Rightarrow \frac{\mathrm{d}f_c(\overbrace{x_p^{H1} + \delta_p}^{\text{const.}}, \overbrace{\tilde{x}_c + \delta_c}^{\text{const.}})}{\mathrm{d}t} = 0. \tag{71}$$

Therefore, the x_c-component of the flow consistency (68) is true:

$$\dot{x}_c = \frac{\mathrm{d}\overbrace{f_c(x_p^{H1} + \delta_p, \tilde{x}_c + \delta_c)}^{x_c\text{-component of } q(\tilde{x})}}{\mathrm{d}t} \quad \text{for} \quad kT < t < (k+1)T. \tag{72}$$

iii. *Conclusion:* As the flow and initial condition are consistent, (69) can be extended to the whole duration of the continuous evolution:

$$x_c(kT^+) = x_c((k+1)T^-) = x_c(t) = f_c(x_p^{H1}(t) + \delta_p(t), \tilde{x}_c(t) + \delta_c(t)) \quad \text{for} \quad kT < t < (k+1)T. \tag{73}$$

This result will now be used to show the consistency of x_p.

(c) *Consistency of x_p:* Analogous to the considerations for x_c, consider x_p from $t = kT^+$ to $t = (k+1)T^-$:

i. *Initial Condition:* By $IA(k)$, $x_p^{H0}(kT^+) = x_p^{H1}(kT^+)$, so the initial condition matches.

ii. *Flow:* In H_0,

$$\dot{x}_p^{H0} = f_p(x_p^{H0}, x_c). \tag{74}$$

In H_1,

$$\dot{x}_p^{H1} = f_p(x_p^{H1}, f_c(x_p(t) + \delta_p(t), \tilde{x}_c(t) + \delta_c(t))) \overset{(73)}{=} f_p(x_p^{H1}, x_c) \quad \text{for} \quad kT < t < (k+1)T. \tag{75}$$

iii. *Conclusion:* Therefore, x_p evolves consistently:

$$x_p^{H0}(t) = x_p^{H1}(t) \quad \text{for} \quad kT < t < (k+1)T. \tag{76}$$

From items 1a to 1c, it follows that

$$x(t^-) = x(t^+) = q(\tilde{x}(t^-)) = q(\tilde{x}(t^+)) \quad \text{for} \quad kT < t < (k+1)T, \tag{77}$$

and also

$$x((k+1)T^-) = q(\tilde{x}((k+1)T^-)). \tag{78}$$

Together with $IA(k)$, this shows (65) for $k+1$.

2. *Helper Variables x_p^- and x_c^- in H_1:*

(a) *Preliminary Considerations:* At $t = 0$, there is no discrete transition, so

$$\delta_p(0^+) = \delta_p(0^-) = 0 \quad \text{and} \tag{79}$$
$$\delta_c(0^+) = \delta_c(0^-) = 0. \tag{80}$$

Due to the discrete transition at $t = kT$, $k \geq 1$,

$$\delta_p(kT^+) = 0 \tag{81}$$
$$\delta_c(kT^+) = 0. \tag{82}$$

Because $\tilde{x}_c$ and x_p^{H1} are unaffected by discrete transitions,

$$\tilde{x}_c(t^+) = \tilde{x}_c(t^-) \quad \text{and} \tag{83}$$
$$\tilde{x}_p^{H1}(t^+) = x_p^{H1}(t^-) \tag{84}$$

hold for all t. Therefore, for any $k \geq 0$,

$$x_p^-(kT^+) = x_p^{H1}(kT^+) + \delta_p(kT^+) \overset{(79),(81)}{=} x_p^{H1}(kT^+) \overset{(84)}{=} x_p^{H1}(kT^-) \tag{85}$$

and

$$x_c^-(kT^+) = \tilde{x}_c(kT^+) + \delta_c(kT^+) \overset{(80),(82)}{=} \tilde{x}_c(kT^+) \overset{(83)}{=} \tilde{x}_c(kT^-). \tag{86}$$

(b) *Interpretation of x_p^-:* The variable $x_p^- := x_p^{H1} + \delta_p$ remains constant at the current sample-and-hold value of x_p: For $kT < t < (k+1)T$,

$$\dot{x}_p^- = \dot{x}_p^{H1} + \dot{\delta}_p = 0 \tag{87}$$

$$\Rightarrow x_p^-(t) = x_p^-(kT^+) = x_p^{H1}(kT^+) + \delta_p(kT^+) \overset{(85)}{=} x_p^{H1}(kT^-) + 0 \tag{88}$$

$$\overset{IA(k)}{=} x_p^{H0}(kT^-). \tag{89}$$

(c) *Interpretation of x_c^-:* Similarly, $x_c^- := \tilde{x}_c + \delta_c$ is a sample-and-hold value of $\tilde{x}_c$: For $kT < t < (k+1)T$,

$$\dot{x}_c^- = \dot{\tilde{x}}_c + \dot{\delta}_c = 0 \tag{90}$$

$$\Rightarrow x_c^- = x_c^-(kT^+) \overset{(86)}{=} \tilde{x}_c(kT^+). \tag{91}$$

(d) *Interpolator state $\tilde{x}_c$:* The state $\tilde{x}_c$ will now be shown to be a specific linear interpolation of x_c: Within the period $kT < t < (k+1)T$,

$$\dot{\tilde{x}}_c = \frac{1}{T}\Big(\underbrace{f_c(x_p + \delta_p, \tilde{x}_c + \delta_c)}_{\overset{(73)}{=} x_c(kT^+)} - \underbrace{(\tilde{x}_c + \delta_c)}_{\overset{(91)}{=} \tilde{x}_c(kT^+)} \Big) = \text{const.} \tag{92}$$

holds. Integrating this result over the period leads to

$$\tilde{x}_c((k+1)T^-) = \tilde{x}_c(kT^+) + \int_{kT^+}^{(k+1)T^-} \dot{x}_c(\tau)\,\mathrm{d}\tau \tag{93}$$

$$\overset{(92)}{=} \tilde{x}_c(kT^+) + T \cdot \frac{1}{T}\left(x_c(kT^+) - \tilde{x}_c(kT^+)\right) \tag{94}$$

$$= x_c(kT^+) \tag{95}$$

$$\overset{(73)}{=} x_c((k+1)T^-). \tag{96}$$

In summary, $\tilde{x}_c$

- is continuous (83),
- matches x_c at the *end* $t = (k+1)T^-$ of the period (96), just before the next controller execution, and
- has constant derivative within the period (92).

For the next sentence, assume that the above holds for all k and not just for the current value of k. Then, the above defines a linear interpolation with supporting points at $t = (k+1)T^-$.

(e) *Summary:* If IA(k) is later shown to hold for all k, then the trajectory of H_1 matches the illustration in Fig. 6 (page 220, top right):

- $x_p^-(t) := x_p^{H1}(t) + \delta_p(t)$ is the newest sample-and-hold value of x_p.
- $x_c^-(t) := \tilde{x}_c(t) + \delta_c(t)$ is the *previous* discrete-time value of x_c.
- Consequently, $f_c(x_p^-, x_c^-)$ reconstructs the *current* value of x_c.
- $\tilde{x}_c(t)$ is linear interpolation of x_c with supporting points at $t = (k+1)T^-$, i.e., with a delay of (almost) one period.

3. *Discrete Transition at $t = (k+1)T$:* Next, it is shown that the transition of H_0 and H_1 at $t = (k+1)T$ is consistent with $x = q(\tilde{x})$. In H_0, this transition is

$$x_p^{H0}((k+1)T^+) = x_p^{H0}((k+1)T^-), \tag{97a}$$
$$x_c((k+1)T^+) = f_c(x_p^{H0}((k+1)T^-),\, x_c((k+1)T^-)), \tag{97b}$$
$$\tau((k+1)T^+) = 0. \tag{97c}$$

In H_1, the corresponding transition is

$$x_p^{H1}((k+1)T^+) = x_p^{H1}((k+1)T^-), \tag{98a}$$
$$\tilde{x}_c((k+1)T^+) = \tilde{x}_c((k+1)T^-) \tag{98b}$$
$$\delta_c((k+1)T^+) = 0, \tag{98c}$$
$$\delta_p((k+1)T^+) = 0, \tag{98d}$$
$$\tau((k+1)T^+) = 0. \tag{98e}$$

Therefore,

$$q(\tilde{x}((k+1)T^+)) \tag{99}$$
$$= \begin{bmatrix} x_p^{H1}((k+1)T^+) \\ f_c(x_p^{H1}((k+1)T^+) + \delta_p((k+1)T^+),\ \tilde{x}_c((k+1)T^+) + \delta_c((k+1)T^+)) \\ \tau((k+1)T^+) \end{bmatrix} \tag{100}$$
$$\overset{(98)}{=} \begin{bmatrix} x_p^{H1}((k+1)T^-) \\ f_c(x_p^{H1}((k+1)T^-) + 0,\ \tilde{x}_c((k+1)T^-) + 0) \\ 0 \end{bmatrix} \tag{101}$$
$$\overset{(78)}{=} \begin{bmatrix} x_p^{H0}((k+1)T^-) \\ f_c(x_p^{H0}((k+1)T^-),\ \tilde{x}_c((k+1)T^-)) \\ 0 \end{bmatrix} \tag{102}$$
$$\overset{(96)}{=} \begin{bmatrix} x_p^{H0}((k+1)T^-) \\ f_c(x_p^{H0}((k+1)T^-),\ x_c((k+1)T^-)) \\ 0 \end{bmatrix} \tag{103}$$
$$\overset{(97)}{=} \begin{bmatrix} x_p^{H0}((k+1)T^+) \\ x_c((k+1)T^+) \\ \tau((k+1)T^+) \end{bmatrix} = x((k+1)T^+). \tag{104}$$

Due to this, (77) and (78), the trajectory is consistent up to $t = (k+1)T^+$, so (64) holds for $k+1$.

4. *Summary:* Equations (64) and (65) hold for $k+1$, as shown by (77), (78) and (104). Therefore, $IA(k+1)$ is true.

Conclusion $IA(k)$ holds for all $k \geq 0$. Therefore, $H_0 =_x H_1$ with $x = q(\tilde{x})$. □

B Comparison with Results of Bak and Johnson (2015)

This section presents two examples to highlight differences between [4] and Theorem 4.1, and to justify the following claim: *The intended meaning of [4, Lemma 1] matches Theorem 4.1; all problematic differences can be judged as typing errors in [4].* Due to the nature of this section, it heavily references [4] and requires familiarity with that paper.

B.1 Definitions and Notation

A periodic discrete transition of a hybrid automaton (Model 1 in [4]) is denoted

$$x' = f(x) \text{ at } t = kT. \tag{105}$$

A vector notation for multidimensional intervals is defined as

$$\left[\begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix}; \begin{bmatrix} b_1 \\ b_2 \\ \vdots \end{bmatrix}\right] := [a_1; b_1] \times [a_2; b_2] \times \dots. \tag{106}$$

For vectors, the maximum and minimum are applied elementwise.

B.2 Example F1: Controller with Internal Dynamics

The following example illustrates that, as claimed in Section 4.2, Zero-Order Continuization can fail if the controller update depends on the controller state x_c.

B.2.1 Original Control Loop

Let the control loop be given by [4, Model 1] with the parameters

$$\dot{x}_p = x_c, \tag{107a}$$
$$x_c' = \mathit{controller\_update}(x_p, x_c) = 2x_c \quad \text{at } t = kT, k > 0, \tag{107b}$$
$$x_p(0) = 1, \tag{107c}$$
$$x_c(0) = 2, \tag{107d}$$
$$T > 0. \tag{107e}$$

This is a simple academic example with an unstable controller and no physical feedback. It illustrates similar problems that occur with PI controllers or any other controllers with internal dynamics. Moving these internal dynamics from the discrete-time to the continuous part of the system is typically not possible, particularly not for uncertain timing.

The explicit solution of this system is

$$x_c(kT) = 2^{k+1}, \quad k \geq 0, \tag{108}$$
$$x_p(kT) = 1 + \sum_{i=1}^{k} 2^i T, \quad k > 0. \tag{109}$$

B.2.2 Continuous Approximation

Following the definition of [4], the continuous approximation is

$$\dot{x}_p = x_c \tag{110a}$$

$$\dot{x}_{c,continuous} = \frac{d}{dt} controller\_update(x_p, x_c) = 2\dot{x}_c \tag{110b}$$

It is unclear how to handle $\dot{x}_c$, and this case occurs in none of the examples of [4]. Two interpretations seem plausible:

1. Handle it similar to how $\dot{x}_p$ is handled in [4, Section IV.B]: Insert the dynamics of the original system, i.e., $\dot{x}_c = 0$.
2. Set $\dot{x}_c = \dot{x}_{c,continuous}$ and solve the equation for $\dot{x}_{c,continuous}$.

For this specific example, both interpretations lead to the same result

$$\dot{x}_{p,continuous} = x_{c,continuous}, \tag{111a}$$

$$\dot{x}_{c,continuous} = 0, \tag{111b}$$

$$x_{p,continuous}(0) = 1, \tag{111c}$$

$$x_{c,continuous}(0) = controller\_update(x_p(0), x_c(0)) = 4. \tag{111d}$$

The explicit solution of this is

$$x_{c,continuous} = 4, \tag{112a}$$

$$x_{p,continuous} = 1 + 4t, \tag{112b}$$

which is suspicious because it is radically different from the explicit solution of the original system. Also, it does not depend on T, whereas $x_p(t)$ and $x_c(t)$ grow faster if T is decreased. Plots for $T = 1$ are shown in Fig. 9.

B.2.3 Continuized Abstraction

Next, the error bound is determined to construct the safe approximation (continuized abstraction). Here, [4, Lemma 1] states that $\omega \in [-T; 0] \otimes [K_{min}; K_{max}]$, where K bounds the rate of change of $\dot{x}_c$ in the *continuous approximation* (not in the abstraction, as discussed later). In loose notation, $\ddot{x}_{c,continuous} \in K$. Maybe this is a misinterpretation and should instead be $\dot{x}_{c,continuous}$. Here, this does not matter as both are equally zero.

Therefore, $K = 0$ and $\omega = 0$, so [4] states that the continuized abstraction is the same as the continuous approximation. This leads to a contradiction to [4, Theorem 1] (Soundness of Continuization), which states that if $K = 0$, then $Reach(x_{p,continuous}) \supseteq Reach(x_p)$. However, for $t = 4T$ this does not hold, since

$$x_{p,continuous}(t) = 1 + 16T \neq x_p(t) = 1 + 30T \tag{113}$$

(cf. Fig. 9 with $T = 1$). Probably, $controller\_update(x_p, x_c)$ must not depend on x_c.

B.3 Interpretation of Bak and Johnson's Lemma 1 without $\dot{x}_c$

For now, put aside the issue with x_c and consider the remaining case.

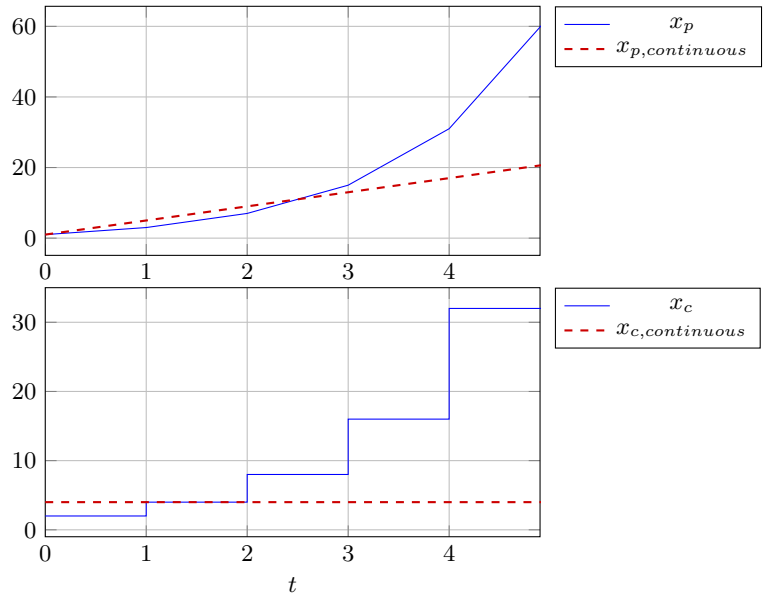

Figure 9: Original system and continuous approximation without error of Example F1 for $T = 1$. (Source code can be found in the subdirectory `notes/continuization-counterexample-f1-f2/` of the repository referenced in Section 5.)

B.3.1 Original Hybrid Automaton

For simplicity, only consider the periodic case. Define the controller as

$$\dot{x}_p = f_p(x_p, x_c), \tag{114a}$$
$$x_c' = f_c(x_p) \quad \text{at } t = kT, \tag{114b}$$
$$x_p(0) = x_{p,0}, \tag{114c}$$
$$x_c(0) = f_c(x_{p,0}). \tag{114d}$$

Definition B.1 (Continuous Approximation)**.** The continuous approximation of (114) is defined in [4] as

$$\dot{\tilde{x}}_p = f_p(\tilde{x}_p, \tilde{x}_c) \tag{115a}$$
$$\dot{\tilde{x}}_c = \left(\frac{d}{dt} f_c(x_p)\right)\Big|_{\dot{x}_p = f_p(x_p, x_c),\ x_p = \tilde{x}_p,\ x_c = \tilde{x}_c} = \underbrace{\frac{\partial f_c(\tilde{x}_p)}{\partial \tilde{x}_p} f_p(\tilde{x}_p, \tilde{x}_c)}_{= f_\omega(\tilde{x}_p, \tilde{x}_c)}, \tag{115b}$$
$$\tilde{x}_p(0) = x_{p,0}, \tag{115c}$$
$$\tilde{x}_c(0) = f_c(x_{p,0}). \tag{115d}$$

◁

(For consistency with Section 4.2, the continuized controller state is here named $\tilde{x}_c$, while in [4] it is named c.) This approximation may be arbitrarily bad: Its stability is neither necessary nor sufficient for the stability of (114).

B.3.2 Equivalent Rewriting of Original Automaton

For comparison with [4], a part of the derivation from Section 4.1 will now be repeated in explicit notation: Defining the error $\omega := x_c - f_c(x_p)$ leads to

$$\omega' = 0 \text{ for } t = kT, \tag{116}$$

$$\dot{\omega} = -\frac{d}{dt} f_c(x_p) = \underbrace{-\frac{\partial f_c(x_p)}{\partial x_p} f_p(x_p, x_c)}_{-f_\omega(x_p, x_c)} \quad \text{for } t \neq kT. \tag{117}$$

Note the negative sign of f_ω here, resulting from the definition of f_ω in Section 4.1. Due to this definition, f_ω matches $\dot{\tilde{x}}_c$ (which is named $\dot{c}$ in [4]), the "derivative of the cyber variable in the continuous approximation", up to a change of arguments from (x_p, x_c) to $(\tilde{x}_p, \tilde{x}_c)$. [4, Lemma 1] claims that a bound on ω can be obtained from "rate of change of the derivative of c_i [here: $\tilde{x}_c$] in the continuous approximation", i.e.,

$$\omega \overset{?}{\in} [-T; 0] \otimes \left[\min_{\substack{t \\ \text{trajectories of } \textbf{cont. approx.}}} \frac{\mathrm{d}}{\mathrm{d}t} f_\omega(\tilde{x}_p(t), \tilde{x}_c(t)); \quad \max_{\substack{t \\ \dots}} \frac{\mathrm{d}}{\mathrm{d}t} f_\omega(\tilde{x}_p(t), \tilde{x}_c(t)) \right]. \tag{118}$$

However, the derivation of (34) using Lemma A.5 leads to

$$\omega = \int_{kT}^{t} -f_\omega(x_p(\tau), x_c(\tau))\ d\tau \quad \text{for } kT \leq t < (k+1)T \tag{119}$$

$$\Rightarrow \omega \in [0; T] \otimes \{-1\} \otimes \left[\min_t f_\omega(x_p(t), x_c(t)); \quad \max_t f_\omega(x_p(t), x_c(t)) \right] \tag{120}$$

$$= \underbrace{[-T; 0] \otimes \left[\min_{\substack{t \\ \text{traj. of } \textbf{original automaton}}} f_\omega(x_p(t), x_c(t)); \quad \max_{\substack{t \\ \dots}} f_\omega(x_p(t), x_c(t)) \right]}_{\Omega}, \tag{121}$$

which shows two differences:

- [4, Lemma 1] speaks of "rate of change of the derivative of" $\tilde{x}_c$, so $\ddot{\tilde{x}}_c$, whereas here only the *first* derivative $\dot{\tilde{x}}_c = f_\omega$ is considered. However, the proof and examples in [4] suggest this is merely a typing error.
- Here, x_c and x_p must be from the original system, not the continuous approximation. Also, in the examples in [4], the states are taken from the abstraction, unlike in the Lemma, which refers to the continuous approximation. A counterexample is shown later in Appendix B.4.

B.3.3 Continuized Abstraction

As a final step, which is roughly the same in [4] and here, the dynamics of ω in the original system are discarded and replaced with a bound $\omega \in \Omega$. Here, it yields the continuized abstraction

$$\dot{x}_p = f_p(x_p, x_c) = f_p(x_p, f_c(x_p) + \omega) \tag{122a}$$

$$\omega \in \Omega. \tag{122b}$$

The variable x_c is no longer required as a state because it can be obtained from

$$x_c = \omega + f_c(x_p). \tag{123}$$

Contrary, in [4] the continuous abstraction (in the following denoted $\hat{x}_p, \hat{x}_c$) is obtained from the continuous approximation by replacing $\tilde{x}_c$ with $\tilde{x}_c + \omega$:

$$\dot{\hat{x}}_p = f_p(\hat{x}_p, \hat{x}_c + \omega) \tag{124a}$$

$$\dot{\hat{x}}_c = \left(\frac{d}{dt} f_c(x_p)\right)\Big|_{\dot{x}_p = f_p(x_p, x_c),\ x_p = \hat{x}_p,\ x_c = \hat{x}_c + \omega} = \underbrace{\frac{\partial f_c(\hat{x}_p)}{\partial \hat{x}_p} f_p(\hat{x}_p, \hat{x}_c + \omega)}_{= f_\omega(\hat{x}_p, \hat{x}_c + \omega)}, \tag{124b}$$

$$\hat{x}_p(0) = x_{p,0}, \tag{124c}$$

$$\hat{x}_c(0) = f_c(x_{p,0}), \tag{124d}$$

$$\omega \in \Omega. \tag{124e}$$

It can be seen that $\hat{x}_c = f_c(\hat{x}_p)$, as the initial condition $\hat{x}_c(0)$ matches

$$\hat{x}_c(0) = f_c(x_p(0)) \tag{125}$$

and the dynamics are the same due to

$$\frac{d}{dt} f_c(\hat{x}_p) = \frac{\partial f_c(\hat{x}_p)}{\partial \hat{x}_p} \dot{\hat{x}}_p \overset{(124a)}{=} \frac{\partial f_c(\hat{x}_p)}{\partial \hat{x}_p} f_p(\hat{x}_p, \hat{x}_c + \omega) \overset{(124b)}{=} \dot{\hat{x}}_c. \tag{126}$$

Consequently, this state $\hat{x}_c$ is not actually required as it can computed directly from $\hat{x}_p$.

B.3.4 Iterative Analysis

The concept of assuming bounds, analyzing under this assumption and then checking the bound (cf. Theorem 3.7) is similarly used in the examples given in [4]. It is therefore not discussed further.

B.4 Example F2: Error Bounds

The following example highlights why it is not sufficient to derive bounds on ω merely from the continuous approximation:

B.4.1 Original Control Loop

Let the control loop be given by [4, Model 1] with the parameters

$$\dot{x}_p = b x_c, \quad b \in \mathbb{R}, b > 0, \tag{127a}$$

$$x_c' = -x_p \text{ at } t = kT, \quad T = 1, \tag{127b}$$

$$x_p(0) = 1, \tag{127c}$$

$$x_c(0) = -1. \tag{127d}$$

The explicit solution of this system is

$$x_c(t) = -x_p(kT), \quad kT \le t < (k+1)T, \tag{128a}$$

$$x_p(t) = x_p(t_0) + b\int_{t_0}^{t} x_c(\tau)dt \quad \forall t_0, \tag{128b}$$

$$= x_p(kT) - b\,x_p(kT)\cdot(t-kT), \quad kT \le t \le (k+1)T, \tag{128c}$$

$$x_p((k+1)T) = x_p(kT)\cdot(1-bT). \tag{128d}$$

This is exponentially stable iff $|1-bT|<1$, i.e., for $0<bT<2$. It is diverging in an infinitely increasing oscillation for $bT>2$. To examine that problematic case, $b=3$ and $T=1$ are considered in the following.

B.4.2 Continuous Approximation

The continuous approximation is

$$\dot{\tilde{x}}_p = b\tilde{x}_c, \tag{129a}$$

$$\dot{\tilde{x}}_c = -b\tilde{x}_c, \tag{129b}$$

$$\tilde{x}_p(0) = 1, \tag{129c}$$

$$\tilde{x}_c(0) = -1. \tag{129d}$$

Its solution is

$$\tilde{x}_c(t) = e^{-bt}\tilde{x}_c(0) = -e^{-bt} \in [-1;0] \tag{130a}$$

and, due to symmetry, $\tilde{x}_p = -\tilde{x}_c$. For completeness, an explicit computation follows:

$$\tilde{x}_p(t) = \tilde{x}_p(0) + b\int_0^t \tilde{x}_c(\tau)d\tau \tag{130b}$$

$$= \tilde{x}_p(0) + b\left(\frac{1}{-b}e^{-bt} - \frac{1}{-b}e^0\right)\tilde{x}_c(0) \tag{130c}$$

$$= 1 + (-e^{-bt}+1)(-1) \tag{130d}$$

$$= 1 + e^{-bt} - 1 \tag{130e}$$

$$= e^{-bt} \in [0;1]. \tag{130f}$$

Therefore, the continuized system is exponentially stable for $b>0$. This is true for $b=3$, so the continuized system is stable while the original is unstable, as illustrated by the simulation in Fig. 10.

B.4.3 Applying Lemma 1

Apply [4, Lemma 1], interpreted as follows: Let K be an interval bound on the *first* derivative of $\tilde{x}_c$ in the *continuous approximation*:

$$\dot{\tilde{x}}_c = -b\tilde{x}_c \in [0;b] = K \tag{131}$$

(Using the second derivative instead would also lead to a bounded interval K.) Then,

$$\omega \in \Omega = [-T;0] \otimes K = [-bT;0]. \tag{132}$$

This result is dubious, since x_p is unbounded for $b=3$, so its sampling error ω should be unbounded as well. Next, examine the resulting abstraction:

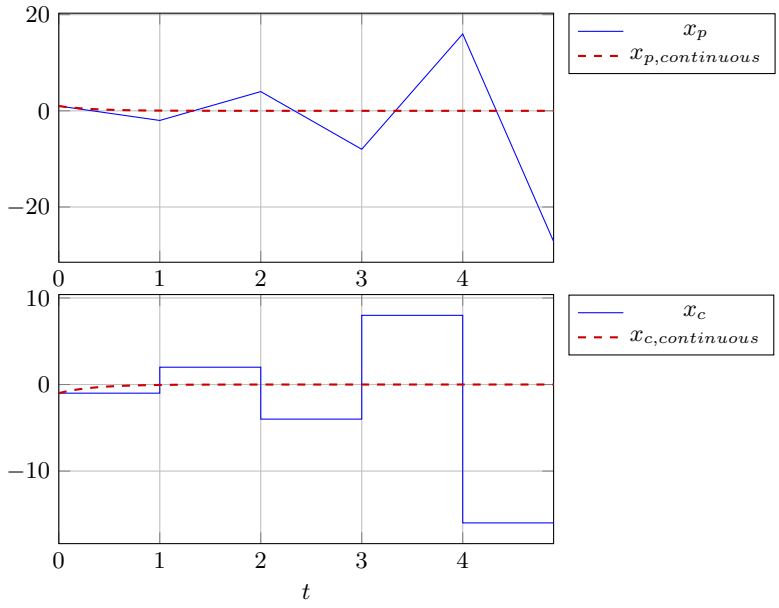

Figure 10: Plots for Example F2: Original System and continuous approximation without error. (Source code can be found in the subdirectory `notes/continuization-counterexample-f1-f2/` of the repository referenced in Section 5.)

B.4.4 Continuized Abstraction

The Continuized Abstraction (124) is

$$\dot{\hat{x}}_p = b\hat{x}_c + b\omega, \tag{133a}$$
$$\dot{\hat{x}}_c = -b\hat{x}_c - b\omega, \tag{133b}$$
$$\hat{x}_p(0) = 1, \tag{133c}$$
$$\hat{x}_c(0) = -1, \tag{133d}$$
$$\omega \in \Omega = [-bT; 0]. \tag{133e}$$

Again, due to symmetry, $\hat{x}_p = -\hat{x}_c$. This exemplifies the claim from Appendix B.3.3 that the $\hat{x}_c$-state is redundant per $\hat{x}_c = f_c(\hat{x}_p)$.

To derive bounds on the states, consider the solutions of $\hat{x}_c$: The $\hat{x}_c$-system is decoupled from $\hat{x}_p$, so it is one-dimensional. For this simple system, the extremal solutions are for $\omega = \omega_{\min}$ and $\omega = \omega_{\max}$: Because there is no oscillation or overshoot, time-varying ω can not make it worse. The maximum of ω is $\omega = 0$, for which the previous result (130a) can be reused:

$$\hat{x}_c(t) = e^{-bt}\hat{x}_c(0) = -e^{-bt} \in [-1; 0]. \tag{134}$$

The minimum of ω is $\omega = -bT$, which yields the step response of a first-order lowpass plus the decaying initial condition:

$$\hat{x}_c(t) = e^{-bt}\hat{x}_c(0) + (1 - e^{-bt})(-1)(-bT) \in [-1; 0] \oplus [0; bT] = [-1; bT] \tag{135}$$

As said before, $\hat{x}_p = -\hat{x}_c$, so $\hat{x}_p$ is also bounded. According to Lemma 1, this means that $\hat{x}_p$ is bounded for any finite b and T. However, $\hat{x}_p$ is unbounded for $b = 3$, which is a contradiction.

B.5 Conclusion

If these thoughts are correct, then [4, Lemma 1] should be modified as follows (Additions and ~~deletions~~ are highlighted):

Lemma B.2 (Sampling Deviation using Lipschitz Constant)**.** *Assume a state-feedback controller controller_update$_i(x_p, x_c) = g_i(x_p)$, i.e., the controller must not depend on x_c. Then, given interval bounds, $K = [K_{\min}, K_{\max}]$ on ~~the rate of change of~~ the derivative of c_i, [i.e., $f_\omega(x_p, x_c) \in K$ with f_ω per (117)], in the ~~continuous approximation~~ sampled CPS, and the period of the associated strictly-periodic (Model 1 from [4, Fig. 1]) controller, T , a sampling deviation function is $\omega_i = [-T, 0] \otimes K$.*

In summary, even though the results of [4] appear to contain some typing errors if taken literally, their intended meaning correctly describes Zero-Order Continuization in the same spirit as Section 4.1.

C Implementation Details

This section gives some details on the implementation of the experiments shown in Section 5. The first-order continuization of the linear case used here is

$$\dot{x}_{pc} = \begin{bmatrix} A_p x_p + B_{pc} f_c(x_p + \delta_p, \tilde{x}_c + \delta_c) \\ \frac{1}{T}(f_c(x_p + \delta_p, \tilde{x}_c + \delta_c) - \tilde{x}_c - \delta_c) \end{bmatrix} \tag{136}$$

$$= \begin{bmatrix} A_p x_p + B_{pc}\left(B_{cp}(x_p + \delta_p) + A_c(\tilde{x}_c + \delta_c)\right) \\ \frac{1}{T}(B_{cp}(x_p + \delta_p) + (A_c - I)(\tilde{x}_c + \delta_c)) \end{bmatrix} \tag{137}$$

$$= \begin{bmatrix} A_p + B_{pc}B_{cp} & B_{pc}A_c & B_{pc}B_{cp} & B_{pc}A_c \\ \frac{1}{T}B_{cp} & \frac{1}{T}(A_c - I) & \frac{1}{T}B_{cp} & \frac{1}{T}(A_c - I) \end{bmatrix} \begin{bmatrix} x_p \\ \tilde{x}_c \\ \delta_p \\ \delta_c \end{bmatrix} \tag{138}$$

$$= f_{pc}(x_{pc}, \delta_{pc}). \tag{139}$$

Intervals are used for the sets X and Δ. Analogous to Remark 4.4, $\underline{X} = \square X'_{pc}$ is used. The bloating ϵ in Theorem 4.7 is chosen as $\epsilon = 10^{-10}$. However, soundness is not strictly guaranteed because SpaceEx uses unsound numerics for the scenario used here.

The analysis is executed as follows:

1. The iteration starts with the empty set $\underline{\Delta} = \{\}$ as initial guess.
2. Repeat up to five times:
   - If the condition of Theorem 4.7 is not satisfied, enlarge $\underline{\Delta}$ by $\underline{\Delta} = 2\square\Delta'$.
   - If it is satisfied, the analysis was successful. To reduce excess growth, shrink $\underline{\Delta}$ to $\underline{\Delta} = \Delta'$ and re-run the analysis per Theorem 4.7, which will succeed by construction. Return the resulting bounds.
3. If the maximum number of iterations was reached, return "Continuization failed".

For the randomized simulations with PySim, the automaton was adapted to PySim's subtly different semantics analogous to [9, p. 198 f.]. For SpaceEx, the used settings are `scenario = stc`, `directions = oct` (octagonal set approximation) and `set-aggregation = "none"` (no set aggregation at discrete transitions).