Automated Formal Synthesis of Digital Controllers for State-Space Physical Plants

Alessandro Abate, Iury Bessa, Dario Cattaruzza, Lucas Cordeiro, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen

Abstract. We present a sound and automated approach to synthesize safe digital feedback controllers for physical plants represented as linear, time invariant models. Models are given as dynamical equations with inputs, evolving over a continuous state space and accounting for errors due to the digitalization of signals by the controller. Our approach has two stages, leveraging counterexample guided inductive synthesis (CEGIS) and reachability analysis. CEGIS synthesizes a static feedback controller that stabilizes the system under restrictions given by the safety of the reach space. Safety is verified either via BMC or abstract acceleration; if the verification step fails, we refine the controller by generalizing the counterexample. We synthesize stable and safe controllers for intricate physical plant models from the digital control literature.

Keywords: State-space dynamical models of physical systems; Digital controllers; Analogue-to-digital converters; Time sampling; quantization; Fixed-point arithmetic; CEGIS; safety requirements.

1 Introduction

Linear Time Invariant (LTI) models represent a broad class of dynamical systems with significant impact in numerous application areas as in the life sciences, robotics, and engineering [3,13]. Synthesis of controllers for LTI models is broadly investigated, however the use of digital control architectures adds new challenges due to the effects of finite-precision arithmetic, time discretization, and quantization noise, which are typically introduced by Analogue-to-Digital (ADC) and Digital-to-Analogue (DAC) conversion. While research on digital control is well developed [3], automated and sound control synthesis is challenging, particularly when the synthesis objective goes beyond classical stability. There are recent methods for verifying more general reachability properties of a given controller [14,15]. However, these methods have not been generalized to control synthesis. We remark that a synthesis algorithm that guarantees satability does not ensure safety: the system might transitively visit an unsafe state that results in an unrecoverable failure.

We propose a novel algorithm for the synthesis of control algorithms for LTI models that are guaranteed to be both stable and safe, considering both the continuous dynamics of the plant and the finite-precision discrete dynamics of the controller, as well as the hybrid elements that connect them. Due to the complexity of such systems, we focus on linear models with known implementation features

(e.g., number of bits, fixed-point arithmetic). We expect a safety requirement given as a reachability property. Safety requirements are frequently overlooked in conventional feedback control synthesis, but play an important role in systems engineering.

We give two alternative approaches for synthesizing digital controllers for state-space physical plants, both based on CounterExample Guided Inductive Synthesis (CEGIS) [31]. We prove their soundness by quantifying errors caused digitalization and quantization effects that arise when time/value-discrete digital controller interacts with the continuous plant.

The first approach uses a *naïve* technique that starts by devising a digital controller that stabilizes the system while remaining safe for a pre-selected time horizon and a single initial state; then, it verifies unbounded safety by unfolding the dynamics of the system, considering the full hyper-cube of initial states, and checking a *completeness threshold*, i.e., the number of iterations required to sufficiently unwind the closed-loop state-space system such that the boundaries are not violated for any larger number of iterations. As it requires explicit unfolding up to the completeness threshold, this approach is computationally expensive.

Instead of explicitly unfolding the dynamics, *the second approach* employs *abstract acceleration* [7] to implicitly evaluate all possible progressions of the system simultaneously. Additionally, the second approach uses *abstraction refinement*, enabling us to always start with a very simple description regardless of the complexity of the overall dynamics, and only expand to more complex models when a solution cannot be found.

We provide experimental results showing that both approaches are able to efficiently synthesize stable and safe controllers for a set of intricate physical plant models taken from the digital control literature: the median run-time for our benchmark set is 7.9 seconds, and most controllers can be synthesized in less than 17.2 seconds. We further show that, in a direct comparison, the abstraction-based approach (second) lowers the median run-time of our benchmarks by a factor of seven over the first approach based on the unfolding of the dynamics.

Contributions

1. We compute state-feedback controllers that are provably stable and in addition guaranteee a given safety property. Existing methods for controller synthesis rely on transfer function representations, which are insufficient to prove safety requirements.
2. We give two novel algorithms: the first, naive one, relies on an unfolding of the dynamics up to a completeness threshold, while the second one is abstraction-based and leverages abstraction refinement and acceleration to improve scalability while retaining soundness. Both approaches provide sound synthesis of state-feedback systems and consider the various sources of imprecision in the implementation of the control algorithm and in the modeling of plant.

3. We develop a model for different sources of quantization errors and their effect on reachability properties. We give bounds that ensure the safety of our controllers in a hybrid continuous-digital domain.

2 Related work

CEGIS Program synthesis is the problem of computing correct-by-design programs from high-level specifications; algorithms for this problem have made substantial progress in recent years. One such approach [18] inductively synthesizes invariants to generate the desired programs.

Program synthesizers are an ideal fit for synthesis of controllers, since the semantics of programs capture the effects of finite-precision arithmetic precisely. In [28], the authors use CEGIS for the synthesis of switching controllers for stabilizing continuous-time plants with polynomial dynamics. The work extends to affine systems, but is limited by the capacity of the state-of-the-art SMT solvers for solving linear arithmetic. Since this approach uses switching models instead of linear dynamics for the digital controller, it avoids problems related to finite precision problem, but potentially suffers from state-space explosion. Moreover, in [29] the same authors use a CEGIS-based approach for synthesizing continuous-time switching controllers that guarantee *reach-while-stay* properties of closed-loop systems, i.e., properties that specify a set of goal states and safe states (constrained reachability). This solution is based on synthesizing control Lyapunov functions for switched systems that yield switching controllers with a guaranteed minimum dwell time in each mode. However, both approaches are not suitable for the kind of control we seek to synthesize.

The work in [2] synthesizes stabilizing controllers for continuous plants given as transfer functions by exploiting bit-accurate verification of software-implemented digital controllers [5, 17]. While this work also uses CEGIS, their approach is restricted to digital controllers for stable closed-loop systems given as transfer function models. By contrast, in the current paper we consider a state-space representation of the physical system, which has known advantages over the transfer function representation [4]: (i) it generalizes to multivariate systems (i.e., with multiple inputs and outputs); (ii) gives the state-to-input relationships, offering an internal model of the system; and (iii) allows to synthesize control systems with guarantees on the internal dynamics, e.g., to synthesize controllers that make the closed-loop system *safe*. Our work focuses on the *safety* of internal states, which is usually overlooked in the literature. Moreover, our work integrates an abstraction/refinement step inside the main CEGIS loop.

The tool Pessoa [21] synthesizes correct-by-design embedded control software in a Matlab toolbox. It is based on the abstraction of a physical system to an equivalent finite-state machine and on the computation of reachability properties over the finite-state machine. Based on this safety specification, Pessoa can synthesize embedded controller software for a range of properties. The embedded controller software can be more complicated than the state-feedback control we synthesized, and the properties available cover more detail. However,

Pessoa does not have consider the quantization error introduced by Digital-to-Analogue/Analogue-to-Digital conversion needed for synthesizing fixed-point digital controllers.

Discretization Effects The classical approach to control synthesis has often disregarded quantization, thus sacrificing soundness. Modern techniques focus on different aspects of discretization, including delayed response [10] and Finite Word Length (FWL) semantics, with the goal either to verify (e.g., [9]) or to optimize (e.g., [24]) given implementations.

There are two different problems that arise from FWL semantics. The first is the error in the dynamics caused by the inability to represent the exact state of the physical system, while the second relates to rounding errors during computation. In [12], a stability measure based on the error of the digital dynamics ensures that the deviation introduced by FWL does not make the digital system unstable. A more recent approach [34] uses μ-calculus to directly model the digital controller so that the selected parameters are stable by design. The analyses in [30, 33] rely on an invariant computation on the discrete system dynamics using Semi-Definite Programming (SDP). While the former uses bounded-input and bounded-output (BIBO) properties to determine stability, the latter uses Lyapunov-based quadratic invariants. In both cases, the SDP solver uses floating-point arithmetic and soundness is checked by bounding the error. An alternative is [25], where the verification of given control code is performed against a known model by extracting an LTI model of the code by symbolic execution. To account for rounding errors, an upper bound is introduced in the verification phase. The work in [26, 27] introduces attractive invariant sets as a mechanism to bound the quantization error effect on stabilization as an invariant set that always converges toward the controllable set. In a similar fashion, [20] evaluates the dynamics of the quantization error and bounds its trajectory to a known region over a finite period of time. This technique works for both linear and non-linear systems.

3 Preliminaries

3.1 State-space representation of physical systems

We consider models of physical plants expressed as ordinary differential equations (ODEs), which we assume to be controllable and under full state information (i.e., we have access to all the model variables):

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, \tag{1}$$

where $t \in \mathbb{R}_0^+$, where A and B are matrices that fully specify the continuous plant, and with initial states set as $x(0)$. While we ideally would like to work on the continuous-time plant, in this work Eq. (1) is soundly discretized in time (as shown in Appendix B) into

$$x_{k+1} = A_d x_k + B_d u_k. \tag{2}$$

where $k \in \mathbb{N}$ and with initial state $x_0 = x(0)$. A_d and B_d denote the matrices that describe the discretized plant dynamics and A and B the continuous plant dynamics. Later, we will address the issue of variable quantization, as introduced by the ADC/DAC conversion blocks (Fig. 1).

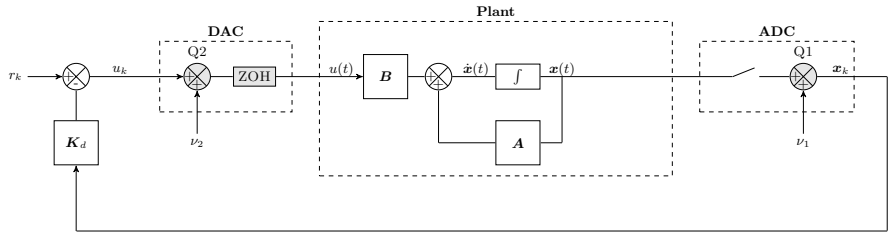

Fig. 1. Closed-loop digital control system

3.2 Controller synthesis via state feedback

Models (1) and (2) depend on external non-determinism in the form of input signals $u(t), u_k$, respectively. Feedback architectures can be employed to manipulate the properties and behaviors of the continuous process (the plant). We are interested in the synthesis of digital feedback control algorithm, as implemented on modern Field-Programmable Gate Arrays or Digital Signal Processors. The most basic feedback architecture is the state feedback, where the control action u_k (notice we work with the discretized signal) is computed by:

$$u_k = r_k - Kx_k. \tag{3}$$

Here $K \in \mathbb{R}^{m \times n}$ is a state-feedback gain matrix, and r_k is a reference signal (again digital). The closed-loop model then takes the form

$$x_{k+1} = (A_d - B_d K)x_k + B_d r_k. \tag{4}$$

The gain matrix K can be set so that the closed-loop discrete dynamics are shaped as desired, for instance according to a specific stability goal or around a specific dynamical behavior [3]. As argued later in this work, we will target more complex objectives, such as quantitative safety requirements, which are not typical in the digital control literature. Further, we will embrace the digital nature of the controller, which manipulates quantities with finite precision.

3.3 Stability of Closed-loop Systems

Asymptotic stability implies convergence to an equilibrium point from any states in its neighborhood. This stability concept is stronger than the Lyapunov and

BIBO stability researched by previous work [2, 29, 30, 33]. In the case of linear systems as in (4), the equilibrium point for zero-input response is always the origin. Here, we consider systems with a zero reference signal.

A discrete-time LTI system as (4) is asymptotically stable iff all the roots of the characteristic polynomial (i.e., the eigenvalues of the closed-loop matrix $A_d - B_d K$) are inside the unity circle in the complex plane, i.e., iff their absolute values are less than one [3]. In this paper, we express the stability specification $\phi_{stability}$ in terms of a check known as Jury's criterion [11]: this is an easy algebraic formula to select the entries of matrix K, so that the closed-loop dynamics are shaped as desired (see Appendix A for details).

3.4 Safety specifications for dynamical systems

We are not limited to the synthesis of digital stabilizing controllers – a well known task in the literature on digital control system – but target safety requirements with an overall approach that is sound and automated. More specifically, we require that the closed-loop system (4) meets given safety specifications. A safety specification raises a requirement on the states of the model, such that the feedback controller (namely the choice of the gains matrix K) must ensure that the states never exit such requirement. Note that a stable, closed-loop system is not necessarily a safe system: indeed, the state values may leave the safe space whilst they converge to the equilibrium, which is typical in the case of oscillatory dynamics. However, a safe system will always be stable. In this work, the safety property is modeled as follows:

$$\phi_{safety} \iff \forall k \geq 0. \bigwedge_{i=1}^{n} \underline{x_i} \leq x_{i,k} \leq \overline{x_i}, \tag{5}$$

where $\underline{x_i}$ and $\overline{x_i}$ are lower and upper bounds for the i-th coordinate x_i of state $x \in \mathbb{R}^n$ at the k-th instant, respectively. This means that the states will always be within the n-dimensional hyper-box.

Furthermore, it is practically relevant to consider the constraints ϕ_{init} on the input signal u_k and ϕ_{input} on the initial states x_0, which we assume have given bounds: $\phi_{init} = \forall k. \underline{u} \leq u_k \leq \overline{u}$, $\phi_{input} = \bigwedge_{i=1}^{n} \underline{x_{i,0}} \leq x_{i,0} \leq \overline{x_{i,0}}$. In the first case, this means that the control input saturates in view of physical constraints.

3.5 Numeric representations and soundness

The models we consider have two sources of error due to numeric representation. The first is the numeric error introduced by the fixed-point numbers employed to model the plant, i.e., to represent the plant dynamics A_d, B_d and x_k. The second is the quantization error introduced by the digital controller, which performs operations on fixed-point numbers. In this section we outline the notation for the fixed-point representation of numbers, and briefly describe the errors introduced. A formal discussion is in Appendix B.2.

Let $\mathcal{F}_{\langle I,F\rangle}(x)$ denote a real number x represented in a fixed point domain, with I bits representing the integer part and F bits representing the decimal part. The smallest number that can be represented in this domain is $c_m = 2^{-F}$. Any mathematical operations performed at the precision $\mathcal{F}_{\langle I,F\rangle}(x)$ will introduce errors, for which an upper bound can be given [6].

We will use $\mathcal{F}_{\langle I_c,F_c\rangle}(x)$ to denote a real number x represented at the fixed-point precision of the controller, and $\mathcal{F}_{\langle I_p,F_p\rangle}(x)$ to denote a real number x represented at the fixed-point precision of the plant model ($I_p \geq I_c \wedge F_p \geq F_c$). Thus any mathematical operations in the digital controller will be in the range of $\mathcal{F}_{\langle I_c,F_c\rangle}$, and all other calculations will be carried out in the range of $\mathcal{F}_{\langle I_p,F_p\rangle}$. The physical plant operates in the reals, which means our verification phase must account for calculation errors in the model.

Effect on safety specification and stability Let us first consider the effect of the quantization errors on safety. Within the controller, state values are manipulated at low precision, alongside the vector multiplication Kx. The inputs are computed using the following equation:

$$u_k = -(\mathcal{F}_{\langle I_c,F_c\rangle}(K) \cdot \mathcal{F}_{\langle I_c,F_c\rangle}(x_k)).$$

This induces two types of the errors detailed above: first, the truncation error due to representing x_k as $\mathcal{F}\langle I_c, F_c\rangle(x_k)$; and second, the rounding error introduced by the multiplication operation. We represent these errors as non-deterministic noise.

An additional error is caused by the representation of the plant dynamics, namely

$$x_{k+1} = \mathcal{F}_{\langle I_p,F_p\rangle}(A_d)\mathcal{F}_{\langle I_p,F_p\rangle}(x_k) + \mathcal{F}_{\langle I_p,F_p\rangle}(B_d)\mathcal{F}_{\langle I_p,F_p\rangle}(u_k).$$

We address this last error by use of interval arithmetic [22] in the verification phase.

Previous studies [19] show that the FWL affects the poles and zeros positions, degrading the closed-loop dynamics, causing steady-state errors (see Appendix B for details) and eventually destabilizing the system [5]. However, since in this paper we require stability only as a precursor to safety, it is sufficient to check that the (perturbed, noisy) model converges to a neighborhood of the equilibrium within the safe set.

In the following, we shall disregard these steady-state errors (caused by FWL effects) when the stability is ensured by synthesis, and then verify its safety accounting for the finite-precision errors.

4 CEGIS of Safe Controllers for LTI Systems

In this section, we describe our technique for synthesizing safe digital feedback controllers using CEGIS. For this purpose, we first provide the synthesizer's general architecture, followed by describing our two approaches to synthesizing

safe controllers: the first one is a baseline approach that relies on a naive explicit unfolding of the transition relation, whereas the second uses abstraction to implicitly evaluate all possible progressions of the system simultaneously.

4.1 General architecture of the program synthesizer

The input specification provided to the program synthesizer is of the form $\exists P. \forall x. \sigma(x, P)$, where P ranges over functions, x ranges over ground terms, and σ is a quantifier-free formula. We interpret the ground terms over some finite domain $\mathcal{D}$. The design of our synthesizer consists of two phases, an inductive synthesis phase and a validation phase, which interact via a finite set of test vectors INPUTS that is updated incrementally. Given the aforementioned specification σ, the inductive synthesis procedure tries to find an existential witness P satisfying the specification $\sigma(x, P)$ for all x in INPUTS (as opposed to all $x \in \mathcal{D}$). If the synthesis phase succeeds in finding a witness P, this witness is a candidate solution to the full synthesis formula. We pass this candidate solution to the validation phase, which checks whether it is a full solution (i.e., P satisfies the specification $\sigma(x, P)$ for all $x \in \mathcal{D}$). If this is the case, then the algorithm terminates. Otherwise, additional information is provided to the inductive synthesis phase in the form of a new counterexample that is added to the INPUTS set and the loop iterates again. More details about the general architecture of the synthesizer can be found in [8].

4.2 Synthesis problem: statement (recap) and connection to program synthesis

At this point, we recall the the synthesis problem that we solve in this work: we seek a digital feedback controller K (see Eq. 3) that makes the closed-loop plant model safe for initial state x_0, reference signal r_k and input u_k as defined in Sec. 3.4. We consider non-deterministic initial states within a specified range, the reference signal to be set to zero, saturation on the inputs, and account for digitalization and quantization errors introduced by the controller.

When mapping back to the notation used for describing the general architecture of the program synthesizer, the controller K denotes P and (x_0, u_k) represents x.

4.3 Naive Approach: CEGIS with Multi-staged Verification

An overview of the controller synthesis is given in Fig. 2. One important observation is that, for this approach, we verify and synthesize a controller over k time steps. We then compute a completeness threshold, $\overline{k}$, for this controller, and verify correctness for $\overline{k}$ time steps. Essentially, $\overline{k}$ is the number of iterations required to sufficiently unwind the closed-loop state-space system such that the boundaries are not violated for any $k > \overline{k}$.

Lemma 1. *There exists a $\overline{k}$ such that it is sufficient to unwind the closed-loop state-space system up to $\overline{k}$ in order to ensure that ϕ_{safety} holds.*

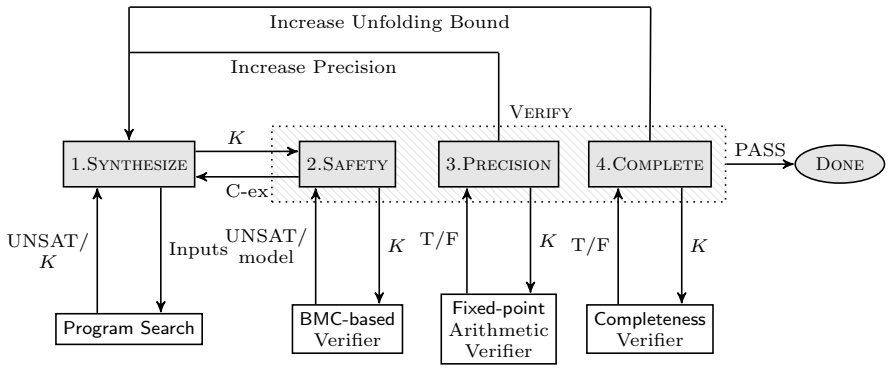

Fig. 2. CEGIS with multi-staged verification

Proof. A stable control system is known to have converging dynamics. Assume the closed-loop matrix eigenvalues are not repeated (which is sensible to do, since we select them). The distance of the trajectory from the reference point (origin) decreases over time within subspaces related to real-valued eigenvalues; however, this in general not the case when dealing with complex eigenvalues. Consider the closed-loop matrix, which updates the states every discrete time step, and select the eigenvalue ϑ with smallest (non-trivial) imaginary value. At every consecutive time steps $kT_s, (k+1)T_s$, the dynamics projected on the corresponding eigenspace rotate of ϑT_s radiants. Thus, taking $k \leq \overline{k}$ as the ceiling of $\frac{2\pi}{\vartheta T_s}$, after $k \leq \overline{k}$ steps we have completed a full rotation, which results in a point closer to the origin, and it is the completeness threshold.

Next, we describe the different phases in Fig. 2 (blocks 1 to 4) in detail.

1. The inductive synthesis phase (i.e., SYNTHESIZE) uses BMC to compute a candidate solution K that satisfies both the stability criteria (Sec. 3.3) and the safety specification (Sec. 3.4). To synthesize a controller that satisfies the stability criteria, we require that a computed characteristic polynomial satisfies Jury's criteria. The details of this calculation can be found in the Appendix.
   Regarding the second requirement, we synthesize a safe controller by unfolding the transition system k steps and by picking a controller K and a single initial state, such that the states at each step do not violate the safety criteria. That is, we ask the bounded model checker if there exists a K that is safe for at least one x_0 in our set of all possible initial states. This is sound if the current k is below the completeness threshold, which we assume to be true in the SYNTHESIZE phase. We also assume some precision $\langle I_p, F_p \rangle$ for the plant and a sampling rate. The checks that these assumptions hold are performed by subsequent VERIFY stages.

```
1: function safetyCheck()
2:     assert(u ≤ u ≤ ū)
3:     set x_0 to be a vertex state, e.g., [x_0, x_0]
4:     for (c = 0; c < 2^{Num_States}; c++) do
5:         while i = 0; i < k; i++ do
6:             u = (plant_typet)((controller_typet)K * (controller_typet)x)
7:             x = A * x + B * u
8:             assert(x ≤ x ≤ x̄ )
9:         end while
10:        set x_0 to be a new vertex state
11:    end for
12: end function
```

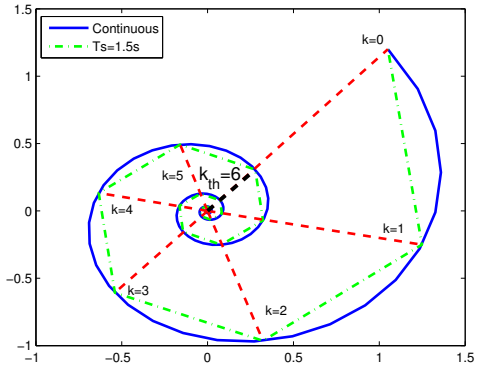

Fig. 3. Completeness threshold for multi-staged verification

2. The first VERIFY stage, SAFETY, checks that candidate solution K, which we synthesized to be safe for at least one initial state, is safe for *all* possible initial states, i.e., does not reach an unsafe state within k steps where we assume k to be under the completeness threshold. After unfolding the transition system corresponding to the previously synthesized controller k steps, we check that the safety specification holds for any initial state.
3. The second VERIFY stage, PRECISION, restores soundness with respect to the plant's precision by using interval arithmetic [22] to validate the operations performed by the previous stage.
4. The third VERIFY stage, COMPLETE, checks that the current k is large enough to ensure safety for any $k'>k$. Here, we compute the completeness threshold $\overline{k}$ for the current candidate K and check that $k \leq \overline{k}$. This is done according to the argument given above and illustrated in Fig. 3.

Checking that the safety specification holds for any initial state can be computationally expensive if the bounds on the allowed initial states are large.

Lemma 2. *If a controller is safe for each of the corner cases of our hypercube of allowed initial states, it is safe for any initial state in the hypercube.*

Thus we only need to check 2^n initial states, where n is the dimension of the state space (number of continuous variables).

Proof. Consider the set of initial states, X_0, which we assume to be convex since it's a hypercube. Name v_i its vertexes, where $i = 1, \ldots, 2^n$. Thus any point $x \in X_0$ can be expressed by convexity as $x = \sum_{i=1}^{2^n} \alpha_i v_i$, where $\sum_{i=1}^{2^n} \alpha_i = 1$. Then if $x_0 = x$, we obtain

$$x_k = (A_d - B_d K)^k x = (A_d - B_d K)^k \sum_{i=1}^{2^n} \alpha_i v_i = \sum_{i=1}^{2^n} \alpha_i (A_d - B_d K)^k v_i = \sum_{i=1}^{2^n} \alpha_i x_k^i,$$

where x_k^i denotes the trajectories obtained from the single vertex v_i. We conclude that any k-step trajectory is encompassed, within a convex set, by those generated from the vertices.

Illustrative Example We illustrate our approach with an example, extracted from the literature [13]. We start with a candidate solution with all coefficients zero, $K = [0\ 0\ 0]^T$, and a precision of $I_p = 13$, $F_p = 3$. In the first VERIFY stage, the SAFETY check finds the counterexample $x_0 = [-0.5\ 0.5\ 0.5]$. After adding the new counterexample to its sets of INPUTS, SYNTHESIZE finds the candidate solution $K = [0\ 0\ 0.00048828125]^T$, which prompts the SAFETY verifier to return $x_0 = [-0.5 - 0.5 - 0.5]$ as the new counterexample.

In the subsequent iteration, the synthesizer is unable to find further suitable candidates and it returns UNSAT, meaning that the current precision is insufficient. Consequently, we increase the precision to $I_p = 17$, $F_p = 7$. Since the previous counterexamples were obtained at lower precision, we remove them from the set of counterexamples. Back in the SYNTHESIZE phase, we re-start the process with a candidate solution with all coefficients 0. Next, the SAFETY verification stage provides the first counterexample at higher precision, $x_0 = [-0.5\ 0.5\ 0.5]$ and SYNTHESIZE finds $K = [0\ 0.01171875\ 0.015625]^T$ as a candidate that eliminates this counterexample. However, this candidate triggers the counterexample $x_0 = [0.5\ -0.5\ -0.5]$ found again by the SAFETY verification stage. In the next iteration, we get the candidate $K = [0\ 0\ -0.015625]$, followed by the counterexample $x_0 = [0.5\ 0.5\ 0.5]$. Finally, SYNTHESIZE finds the candidate $K = [0.01171875\ -0.013671875\ -0.013671875]^T$, which is validated as a final solution by all verification stages.

4.4 Abstraction-based CEGIS

The naive approach described in Sec. 4.3 synthesizes a controller for an individual initial state and input with a bounded time horizon and, subsequently, it generalizes it to all reachable states, inputs, and time horizons during the verification phase. Essentially, this approach relies on the explicit simulation over a bounded time horizon of individual states that form part of an uncountable space and tries to generalize it for an infinite space over an infinite time horizon.

Conversely, in this section, we find a controller for a continuous initial set of states and set of inputs, over an abstraction of the continuous dynamics that conforms to witness proofs at specific times. Moreover, this approach uses abstraction refinement enabling us to always start with a very simple description regardless of the complexity of the overall dynamics, and only expand to more complex models when a solution cannot be found.

The CEGIS loop for this approach is illustrated in Fig. 4.

1. We start by doing some preprocessing:
   (a) Compute the characteristic polynomial of the matrix $(A_d - B_d K)$ as $P_a(z) = z^n + \sum_{i=1}^{p} (a_i - k_i) z^{n-i}$.
   (b) Calculate the noise set N from the quantizer resolutions and estimated round-off errors:
   $$N = \left\{ \nu_1 + \nu_2 + \nu_3 : \nu_1 \in \left[-\frac{q_1}{2} \quad \frac{q_1}{2} \right] \wedge \nu_2 \in \left[-\frac{q_2}{2} \quad \frac{q_2}{2} \right] \wedge \nu_3 \in [-q_3 \quad q_3] \right\}$$
   where q_1 is the error introduced by the truncation by the ADC, q_2 is the error introduced by the DAC and q_3 is the maximum truncation and rounding error in $u_k = -K \cdot \mathcal{F}_{\langle I_c, F_c \rangle}(x_k)$ as discussed in Section 3.5. More details on how to model quantization as noise are given in Appendix B.3.
   (c) Calculate a set of initial bounds on K, ϕ_{init}^K, based on the input constraints. Note that these bounds will be used by the SYNTHESIZE phase to reduce the size of the solution space.
   $$(\phi_{init} \wedge \phi_{input} \wedge u_k = -Kx_k) \Rightarrow \phi_{init}^K$$
2. In the SYNTHESIZE phase, we synthesize a candidate controller $K \in \mathbb{R}\langle I_c, F_c \rangle^n$ that satisfies $\phi_{stability} \wedge \phi_{safety} \wedge \phi_{init}^K$ by invoking a SAT solver. If there is no candidate solution we return UNSAT and exit the loop.
3. Once we have a candidate solution, we perform a safety verification of the progression of the system from ϕ_{init} over time, $x_{k+1} \models \phi_{safety}$. In order to compute the progression of point x_0 at iteration k, we accelerate the dynamics of the closed-loop system and obtain:
   $$x = (A_d - B_d K)^k x_0 + \sum_{i=0}^{k-1} (A_d - B_d K)^i B_n (\nu_1 + \nu_2 + \nu_3) : B_n = [1 \; 1 \cdots \; 1]^T \tag{6}$$
   As this still requires us to verify the system for every k up to infinity, we use abstract acceleration again to obtain the reach-tube, i.e., the set of all reachable states at all times given an initial set ϕ_{init}:
   $$\begin{aligned} \hat{X}^{\#} &= \mathcal{A} X_0 + \mathcal{B}_n N \\ X_0 &= \{ x : x \models \phi_{init} \} \end{aligned} \tag{7}$$
   where $\mathcal{A} = \bigcup_{k=1}^{\infty} (A_d - B_d K) t^k, \mathcal{B}_n = \bigcup_{k=1}^{\infty} \sum_{i=0}^{k} (A_d - B_d K)^i B_n$ are abstract matrices for the closed-loop system [7], whereas the set N is non-deterministically chosen.

We next evaluate $\hat{X}^{\#} \models \phi_{safety}$. If the verification holds we have a solution, and exit the loop. Otherwise, we find a counterexample iteration k and corresponding initial point x_0 for which the property does not hold, which we use to locally refine the abstraction. When the abstraction cannot be further refined, we provide them to the ABSTRACT phase.

4. If we reach the ABSTRACT phase, it means that the candidate solution is not valid, in which case we must refine the abstraction used by the synthesizer.
   (a) Find the constraints that invalidate the property as a set of counterexamples for the eigenvalues, which we define as ϕ_Λ. This is a constraint in the spectrum i.e., transfer function) of the closed loop dynamics.
   (b) We use ϕ_Λ to further constrain the characteristic polynomial $z^n + \sum_{i=1}^{n}(a_i - k_i)z^{n-i} = \prod_{i=1}^{n}(z - \lambda_i) : |\lambda_i| < 1 \wedge \lambda_i \models \phi_\Lambda$. These constraints correspond to specific iterations for which the system may be unsafe.
   (c) Pass the refined abstraction $\phi(K)$ with the new constraints and the list of iterations k to the SYNTHESIZE phase.

Illustrative Example Let us consider the Helicopter example in Table 2 with discretized dynamics

$$A_d = \begin{bmatrix} 2.6207 & -1.1793 & 0.65705 \\ 2 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}, \quad B_d = \begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix}$$

Using the initial state bounds $\underline{x_0} = -0.9$ and $\overline{x_0} = 0.9$, the input bounds $\underline{u} = -10$ and $\overline{u} = 10$, and safety specifications $\underline{x_i} = -0.92$ and $\overline{x_i} = 0.92$, the SYNTHESIZE phase in Fig. 4 generates an initial candidate controller $K = [0.24609375\ -0.125\ 0.1484375]$. This candidate is stable, but the VERIFY phase finds it to be unsafe and returns a list of iterations with an initial state that fails the safety specification $(k, x_0) \in \{(2, [0.9\ -0.9\ 0.9]), (3, [0.9\ -0.9\ -0.9])\}$. This allows the ABSTRACT phase to create a new safety specification that considers these iterations for these initial states to constrain the solution space. This refinement allows SYNTHESIZE to find a new controller $K = [0.23828125\ -0.17578125\ 0.109375]$, which this time passes the verification phase, resulting in a safe system.

5 Experimental Evaluation

5.1 Description of the Benchmarks

A set of state-space models for different classes of systems was extracted from the literature [1, 13, 23, 32] and employed for validating our methodology.

DC Motor Rate plants describes the angular velocity of a DC Motor, respectively. The *Automotive Cruise System* plant represents the speed of a motor vehicle. The *Helicopter Longitudinal Motion* plant provides the longitudinal motion model of a helicopter. The *Inverted Pendulum* plant describes a pendulum model with its center of mass above its pivot point. The *Magnetic Suspension*

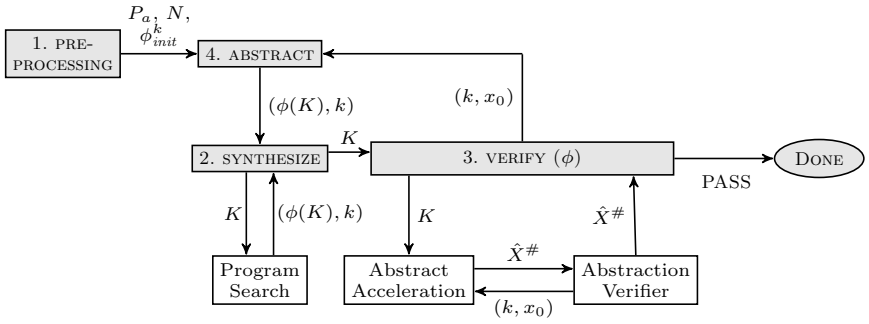

Fig. 4. Abstraction-based CEGIS

plant provides a physical model for which a given object is suspended via magnetic field. The *Magnetized Pointer* plant describes a physical model employed in analogue gauges and indicators that is rotated through interaction with magnetic fields. The *1/4 Car Suspension* plant presents a physical model that connects a car to its wheels and allows relative motion between both parts. The *Computer Tape Driver* plant describes a system to read and write data on a storage device in the computer.

Section C in the appendix summarizes all our benchmarks, which are SISO models (Section 3). The Inverted Pendulum appears to be a two output system, but it was treated as two SISO systems during the experiments. For the sake of simplicity all the state measurements are assumed available. In practice, state estimators should be designed in order to obtain an estimate of the states from the single output. Further work can include state observers synthesis.

All benchmarks were discretized with different sample times using the approach described in Appendix B. All experiments were performed considering $\underline{x_i} = -1$ and $\overline{x_i} = 1$ and the inputs $u_k = 0, \forall k > 0$. We have conducted the experimental evaluation on a 12-core 2.40 GHz Intel Xeon E5-2440 with 96 GB of RAM and Linux OS. We used the Linux *times* command to measure CPU time used for each benchmark. The runtime was limited to one hour per benchmark.

5.2 Objectives

Using the state-space models given in Section 5.1, our experimental evaluation aims to answer two research questions:

RQ1 **(CEGIS)** do the multi-staged verification and abstraction-based CEGIS generate a FWL digital controller in a reasonable amount of time?

RQ2 **(sanity check)** are the synthesized controllers sound and can their stability and safety be confirmed outside of our model?

5.3 Results

We give the results in Table 1. Here *Benchmark* is the name of the respective benchmark, $\mathcal{F}_{\langle I_p,F_p\rangle}$ is the fixed-point precision used to model the plant, while *Time* is the total time required to synthesize a stable controller for the given plant, for CEGIS with multi-staged verification as well as abstraction-based CEGIS. Timeouts are indicated by ✗. The precision for the controller, $\mathcal{F}_{\langle I_c,F_c\rangle}$, is chosen to be $I_c = 8$, $F_c = 8$.

For the majority of our benchmarks, we observe that the abstraction-based back-end is faster than the basic multi-staged verification approach. However, it fails to find solutions to three benchmarks, as compared to two with the multi-staged back-end, where the numbers given in the discretization matrices are too small. In direct comparison, the abstraction-based approach is on average able to find a solution in approximately 35% of the time required using the multi-staged back-end, and has a median run-time 1.15 s, which is seven times smaller than the multi-staged approach. The two back-ends complement each other in benchmark coverage and together solve all benchmarks in the set.

#	Benchmark	Multi-staged		Abstraction	
		$\mathcal{F}_{\langle I_p,F_p\rangle}$	Time	$\mathcal{F}_{\langle I_p,F_p\rangle}$	Time
1	Cruise Control	8,16	6.88 s	16,16	0.24 s
2	DC Motor	8,16	8.10 s	20,20	5.38 s
3	Helicopter	✗	✗	16,16	2.12 s
4	Inverted Pendulum	8,16	7.81 s	16,16	0.39 s
5	Magnetic Pointer	✗	✗	28,28	132.85 s
6	Magnetic Suspension	12,20	8.73 s	32,32	7.85 s
7	Pendulum	8,16	9.01 s	20,20	1.15 s
8	Suspension	12,20	49.41 s	✗	✗
9	Tape Driver	8,16	9.18 s	✗	✗
10	Satellite	8,16	8.98 s	✗	✗

Table 1. Experimental results

The median run-time for our benchmark set is 7.9 s. Overall, the average synthesis time amounts to approximately 17.2 s. We consider these times short enough to be of practical use to control engineers, and thus affirm RQ1.

There are several cases for which the system may fail to find a controller. For the naive approach, the completeness threshold may be too large, thus causing a timeout. On the other hand, the abstraction-based approach may require a very precise abstraction, resulting in too many refinements and, consequently, timeout. Yet another source of incompleteness is the inability of the SYNTHESIZE phase to use a large enough precision for the plant model.

The synthesized controllers were confirmed to be stable outside of our model representation using MATLAB, positively answering RQ2. A link to the full exper-

imental environment, including scripts to reproduce the results, all benchmarks and the tool, is provided in the footnote.[1]

5.4 Threats to validity

Benchmark selection: We report an assessment of both our approaches over a diverse set of real-world benchmarks. Nevertheless, this set of benchmarks is limited within the scope of this paper and the performance may not generalize to other benchmarks.
Plant precision and discretization heuristic: Our algorithm to select suitable FWL word widths to model the plant behavior employs a heuristic based on user-provided controller word-width specifications. For the first approach, where we may need to increase the precision of the time discretization, we select the time steps for the discretizations on a similar basis. This works sufficiently well for our benchmarks, but performance may suffer in some cases, for example if the completeness threshold is high.
Abstraction on other properties: The performance gain from abstract acceleration may not hold for more complex properties than safety, for instance "eventually always reaches a certain safe set".

6 Conclusion

We have presented two automated approaches to synthesize digital state-feedback controllers that ensure both stability and safety over the state-space representation. The first approach relies on explicit unfolding of the closed-loop model dynamics up to a completeness threshold, whilst the second one applies abstraction refinement and acceleration to increase speed whilst retaining soundness. Both approaches are novel within the control literature: they give a fully automated synthesis method that is algorithmically and numerically sound, considering various error sources in the implementation of the digital control algorithm and in the computational modeling of plant dynamics. Our experimental results show that both approaches are able to synthesize stable controllers for most benchmarks within a reasonable amount of time fully automatically. In particular, both approaches complement each other and together solve all benchmarks, which have been derived from the control literature.

Future work will focus the extension of these approaches to the continuous-time case, to models with output-based control architectures (hinging on the use of observers), and to considering performance requirements (alongside safety) in the synthesis of digital controllers.

[1] `https://drive.google.com/open?id=0ByIexo3Z5N91eDludEdWdWxfV1E`

References

1. Control tutorials for MATLAB and SIMULINK. `http://ctms.engin.umich.edu/`.
2. 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 *Hybrid Systems: Computation and Control (HSCC)*. ACM, 2017.
3. K. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Prentice Hall information and system sciences series. Prentice Hall, 1997.
4. K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA, 2008.
5. I. Bessa, H. Ismail, R. Palhares, L. Cordeiro, and J. E. C. Filho. Formal non-fragile stability verification of digital control systems with uncertainty. *IEEE Transactions on Computers (to appear)*, 2016.
6. M. Brain, C. Tinelli, P. Rümmer, and T. Wahl. An automatable formal semantics for IEEE-754 floating-point arithmetic. In *ARITH*, pages 160–167. IEEE, 2015.
7. D. Cattaruzza, A. Abate, P. Schrammel, and D. Kroening. Unbounded-time analysis of guarded LTI systems with inputs by abstract acceleration. In *22nd International Symposium on Static Analysis*, volume 9291 of *LNCS*, pages 312–331. Springer, 2015.
8. C. David, D. Kroening, and M. Lewis. Using program synthesis for program analysis. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-20)*, LNCS, pages 483–498. Springer, 2015.
9. I. V. de Bessa, H. Ismail, L. C. Cordeiro, and J. E. C. Filho. Verification of fixed-point digital controllers using direct and delta forms realizations. *Design Autom. for Emb. Sys.*, 20(2):95–126, 2016.
10. P. S. Duggirala and M. Viswanathan. Analyzing real time linear control systems using software verification. In *IEEE Real-Time Systems Symposium*, pages 216–226, Dec 2015.
11. S. Fadali and A. Visioli. *Digital Control Engineering: Analysis and Design*, volume 303 of *Electronics & Electrical*. Elsevier/Academic Press, 2009.
12. I. J. Fialho and T. T. Georgiou. On stability and performance of sampled-data systems subject to wordlength constraint. *IEEE Transactions on Automatic Control*, 39(12):2476–2481, 1994.
13. G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Pearson, 7th edition, 2015.
14. G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.
15. G. Frehse, C. L. Guernic, A. Donzé, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.
16. R. A. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
17. H. Ismail, I. Bessa, L. C. Cordeiro, E. B. de Lima Filho, and J. E. C. Filho. DSVerifier: A bounded model checking tool for digital systems. In *Model Checking Software (SPIN)*, volume 9232, pages 126–131. LNCS, 2015.
18. S. Itzhaky, S. Gulwani, N. Immerman, and M. Sagiv. A simple inductive synthesis methodology and its applications. In *OOPSLA*, pages 36–46. ACM, 2010.
19. G. Li. On pole and zero sensitivity of linear systems. *IEEE Trans. on Circuits and Systems–I: Fundamental Theory and Applications*, 44(7):583–590, 1997.

20. D. Liberzon. Hybrid feedback stabilization of systems with quantized signals. *Automatica*, 39(9):1543–1554, 2003.
21. M. Mazo, Jr., A. Davitian, and P. Tabuada. PESSOA: A tool for embedded controller synthesis. In *Computer Aided Verification (CAV)*, volume 6174 of *LNCS*, pages 566–569. Springer, 2010.
22. R. E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.
23. V. A. Oliveira, E. F. Costa, and J. B. Vargas. Digital implementation of a magnetic suspension control system for laboratory experiments. *IEEE Transactions on Education*, 42(4):315–322, Nov 1999.
24. A. K. Oudjida, N. Chaillet, A. Liacha, M. L. Berrandjia, and M. Hamerlain. Design of high-speed and low-power finite-word-length PID controllers. *Control Theory and Technology*, 12(1):68–83, 2014.
25. J. Park, M. Pajic, I. Lee, and O. Sokolsky. Scalable verification of linear controller software. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS, pages 662–679. Springer, 2016.
26. B. Picasso and A. Bicchi. Stabilization of LTI systems with quantized state-quantized input static feedback. In *6th International Workshop on Hybrid Systems: Computation and Control*, pages 405–416. Springer, 2003.
27. B. Picasso, F. Gouaisbaut, and A. Bicchi. Construction of invariant and attractive sets for quantized-input linear systems. In *41st Conference on Decision and Control, CDC*, volume 1, pages 824–829. IEEE, 2002.
28. H. Ravanbakhsh and S. Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *Conference on Decision and Control (CDC)*, pages 4232–4239, 2015.
29. H. Ravanbakhsh and S. Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *EMSOFT*, pages 8:1–8:10. ACM, 2016.
30. P. Roux, R. Jobredeaux, and P. Garoche. Closed loop analysis of control command software. In *HSCC*, pages 108–117. ACM, 2015.
31. A. Solar-Lezama. Program sketching. *STTT*, 15(5-6):475–495, 2013.
32. R. H. G. Tan and L. Y. H. Hoo. DC-DC converter modeling and simulation using state space approach. In *IEEE Conference on Energy Conversion, CENCON*, pages 42–47, Oct 2015.
33. T. E. Wang, P. Garoche, P. Roux, R. Jobredeaux, and E. Feron. Formal analysis of robustness at model and code level. In *HSCC*, pages 125–134. ACM, 2016.
34. J. Wu, G. Li, S. Chen, and J. Chu. Robust finite word length controller design. *Automatica*, 45(12):2850–2856, 2009.

A Stability of closed-loop models

A.1 Encoding of Jury's criterion

For a closed-loop system as (4), the characteristic polynomial $P(z)$ should be computed as follows:

$$P(z) = \det(zI_n - A_d + B_d K)\,, \tag{8}$$

where I_n is the n-th order identity matrix.

Jury's criterion provides necessary and sufficient conditions for the eigenvalues to be within the unit circle and, consequently, for the system is stable [11]. Given a polynomial $P(z)$ of the form

$$P(z) = a_0 z^N + a_1 z^{N-1} + ...a_{N-1} z + a_N = 0,\;\; a_0 \neq 0\,,$$

Jury's criterion explores solely the coefficients $a_0, a_1, \ldots, a_{N-1}$ of $P(z)$ for checking stability, which we capture as property $\phi_{stability}$, as detailed below.

This study limits itself to explain the encoding of Jury's criterion for formal synthesis. For the stability test procedure, the following matrix $M = [m_{ij}]_{(2N-2)\times N}$ is built from the coefficients of $P(z)$:

$$M = \begin{bmatrix} V^{(0)} \\ V^{(1)} \\ \vdots \\ V^{(N-2)} \end{bmatrix},$$

where $V^{(k)} = [v_{ij}^{(k)}]_{2\times N}$ is such that

$$v_{ij}^{(0)} = \begin{cases} a_{j-1}, & \text{if } i = 1 \\ v_{(1)(N-j+1)}^{(0)}, & \text{if } i = 2 \end{cases}, \text{ and}$$

$$v_{ij}^{(k)} = \begin{cases} 0, & \text{if } j > n-k \\ v_{1j}^{(k-1)} - v_{2j}^{(k-1)} \cdot \frac{v_{11}^{(k-1)}}{v_{21}^{(k-1)}}, & \text{if } j \leq n-k \text{ and } i = 1 \\ v_{(1)(N-j+1)}^{(k)}, & \text{if } j \leq n-k \text{ and } i = 2 \end{cases},$$

where $k \in \mathbb{Z}$, such that $0 < k < N-2$. $P(z)$ is the characteristic polynomial of a stable system if and only if the following four propositions hold:

- R_1: $P(1) > 0$;
- R_2: $(-1)^N P(-1) > 0$;
- R_3: $|a_0| < a_N$;
- R_4: $m_{11} > 0 \iff (m_{31} > 0) \wedge (m_{51} > 0) \wedge \ldots \wedge (m_{(2N-3)(1)} > 0)$.

The stability property $\phi_{stability}$ is encoded as the following formula:

$$\phi_{stability} \iff (R_1 \wedge R_2 \wedge R_3 \wedge R_4).$$

A.2 Stability of Closed-loop Systems with Fixed-point Controller Error

The proof above relies on the fact that the relationship between states and next states is defined by $x_{k+1} = (A_d - B_d K) \cdot x_k$, all computed at infinite precision. When we employ a FWL digital controller, the operation becomes:

$$x_{k+1} = A_d \cdot x_k - (\mathcal{F}_{\langle I_c, F_c \rangle}(K) \cdot \mathcal{F}_{\langle I_c, F_c \rangle}(x_k)).$$
$$x_{k+1} = (A_d - B_d K) \cdot x_k + B_d K \delta$$

where δ is the maximum error that can be introduced by the FWL controller in one step, i.e., by reading the states values once and multiplying by K once. We derive the closed form expression for x_n as follows:

$$\begin{aligned}
x_1 &= (A_d - B_d K)x_0 + B_d K\delta \\
x_2 &= (A_d - B_d K)((A_d - B_d K)x_0 + B_d K\delta) + B_d K\delta \\
&= (A_d - B_d K)^2 x_0 + (A_d - B_d K)B_d K\delta + B_d K\delta \\
x_3 &= (A_d - B_d K)^3 x_0 + (A_d - B_d K)^2 B_d K\delta + (A_d - B_d K)B_d K\delta + B_d K\delta \\
x_n &= (A_d - B_d K)^n x_0 + (A_d - B_d K)^{n-1} B_d K\delta + ... + (A_d - B_d K)^1 B_d K\delta + B_d K\delta \\
&= (A_d - B_d K)^n x_0 + \sum_{i=0}^{i=n-1} (A_d - B_d K)^i B_d k\delta
\end{aligned}$$

The definition of asymptotic stability is that the system converges to a reference signal, in this case we use no reference signal so an asymptotically stable system will converge to the origin. We know that the original system with an infinite-precision controller is stable, because we have synthesized it to meet Jury's criterion. Hence, $(A_d - B_d K)^n \cdot x_0$ must converge to zero.

To show that a fixed-point implementation of the same controller will converge to a finite value, we need to show that the infinite sum converges. A known result is that a power series of matrices converges [16], iff the eigenvalues of the matrix are less than 1, as follows: $\sum_{i=0}^{\infty} T^i = (I - T)^{-1}$, where I is the identity matrix and T is a square matrix. Thus, our system will converge to the value

$$0 + (I - A_d + B_d K)^{-1} B_d k\delta\,.$$

As a result, if the value $(I - A_d + B_d K)^{-1} B_d k\delta$ is within the safe space, then the synthesized fixed-point controller results in a safe closed-loop model. The convergence to a finite value, however, will not make it asymptomatically stable.

B LTI system Transformations and Equivalences

B.1 Time discretization of continuous-space dynamics

A dynamical system is a system in which a function describes the progression of a state over time. In a continuous domain with linear dynamics, it is described by a first-order Ordinary Differential Equation (ODE).

$$\dot{x}(t) = Ax(t) + Bu(t)\,. \tag{9}$$

Discretization of an continuous dynamical system turns the ODE into a difference equation, assuming zero-order hold for the input u, to

$$x_{k+1} = A_d x_k + B_d u_k \,, \tag{10}$$

where

$$A_d = e^{AT_s} = \mathcal{L}^{-1}(sI - A)^{-1}{}_{t=T_s} \tag{11}$$

$$B_d = \int_0^{T_s} e^{At} dt \; B = A^{-1}(A_d - I)B \,, \tag{12}$$

and T_s is the sample time. Then $x(kT_s) = x_k \forall k \in \mathbb{N}$. The eigenvalues of A_d are related to those of A: $\hat{\lambda}_i = e^{-\lambda_i T_s}$, where $\hat{\lambda}_i \in \sigma(A_d) \wedge \lambda_i \in \sigma(A)$ where $\sigma(\cdot)$ is the spectrum of a matrix.

B.2 Errors due to numerical representation

We have used $\mathcal{F}_{\langle I,F\rangle}(x)$ denote a real number x represented in a fixed point domain, with I bits representing the integer part and F bits representing the decimal part. The smallest number that can be represented in this domain is $c_m = 2^{-F}$. The following approximation errors will arise in mathematical operations and representation:

1. **Truncation:** Let x be a real number, and $\mathcal{F}_{\langle I,F\rangle}(x)$ be the same number represented in a fixed-point domain as above. Then $\mathcal{F}_{\langle I,F\rangle}(x) = x - \delta_T$ where the error $\delta_T = x \; \%_{c_m} \; \tilde{x}$, and $\%_{c_m}$ is the modulus operation performed on the last bit.
   Thus, δ_T is the truncation error and it will propagate across operations.
2. **Rounding:** The following errors appear in basic operations. Let c_1, c_2 and c_3 be real numbers, and δ_{T1} and δ_{T2} be the truncation errors caused by representing c_1 and c_2 in the fixed-point domain as above.
   (a) Addition/Subtraction: these operations only propagate errors coming from truncation of the operands, namely $\mathcal{F}_{\langle I,F\rangle}(c_1) \pm \mathcal{F}_{\langle I,F\rangle}(c_2) = c_3 + \delta_3$ with $|\delta_3| \leq |\delta_{T1}| + |\delta_{T2}|$.
   (b) Multiplication: $\mathcal{F}_{\langle I,F\rangle}(c_1) \cdot \mathcal{F}_{\langle I,F\rangle}(c_2) = c_3 + \delta_3$ with $|\delta_3| \leq |\delta_{T1} \cdot \mathcal{F}_{\langle I,F\rangle}(c_2)| + |\delta_{T2} \cdot \mathcal{F}_{\langle I,F\rangle}(c_1)| + c_m$, where $c_m = 2^{-F}$ as above.
   (c) Division: the operations performed by our controllers in the FWL domain do not include division. However, we do use division in computations at the precision of the plant. Here the error depends on whether the divisor is greater or smaller than the dividend. $\mathcal{F}_{\langle I,F\rangle}(c_1) / \mathcal{F}_{\langle I,F\rangle}(c_2) = c_3 + \delta_{T3}$ where δ_{T3} is $(\delta_{T2} \cdot c_1 - \delta_{T1} \cdot c_2)/(\delta_{T2}^2 - \delta_{T2} c_2)$,
3. **Overflow:** The maximum size of a real number x that can be represented in a fixed point domain as $\mathcal{F}_{\langle I,F\rangle}(x)$ is $\pm(2^{I-1} + 1 - 2^{-F})$. Numbers outside this range cannot be represented by the domain. We check that overflow does not occur.

B.3 Modelling quantization as noise

During any given ADC conversion, the continuous signal will be sampled in the real domain and transformed by $\mathcal{F}\langle I_c, F_c\rangle(x)$ (assuming the ADC discretization is the same as the digital implementation). This sampling uses a threshold which is defined by the less significant bit ($q_c = 2^{-F_c}$) of the ADC and some non-linearities of the circuitry. The overall conversion is

$$\mathcal{F}\langle I_c, F_c\rangle(y(t)) = y_k : y_k \in \left[y(t) - \frac{q_c}{2} \quad y(t) + \frac{q_c}{2}\right] .$$

If we denote the error in the conversion by $\nu_k = y_k - y(t)$ where $t = nk$, and n is the sampling time and k the number of steps, then we may define some bounds for it $\nu_k \in [-\frac{q_c}{2} \quad \frac{q_c}{2}]$.

We will assume, for the purposes of this analysis, that the domain of the ADC is that of the digital controller (i.e, the quantizer includes any digital gain added in the code). The process of quantization in the DAC is similar except that it is calculating $\mathcal{F}\langle I_{dac}, F_{dac}\rangle(\mathcal{F}\langle I_c, F_c\rangle(x))$. If these domains are the same ($I_c = I_{dac}$, $F_c = F_{dac}$), or if the DAC resolution in higher than the ADCs, then the DAC quantization error is equal to zero. From the above equations we can now define the ADC and DAC quantization noises $\nu_{1k} \in [-\frac{q_1}{2} \quad \frac{q_1}{2}]$ and $\nu_{2k} \in [-\frac{q_2}{2} \quad \frac{q_2}{2}]$ where $q_1 = q_c \wedge q_2 = q_{dac}$. This is illustrated in Fig. 3.2 where Q_1 is the quantizer of the ADC and Q_2 the quantizer for the DAC. These bounds hold irrespective of whether the noise is correlated, hence we may use them to over-approximate the effect of the noise on the state space progression over time. The resulting dynamics are

$$x_{k+1} = A_d x_k + B_d(u_k + \nu_{2k}) \tag{13}$$

$$u_k = -K x_k + \nu_{1k} , \tag{14}$$

which result in the following closed-loop dynamics:

$$x_{k+1} = (A_d - B_d K_d)x_k + B_d \nu_{2k} + \nu_{1k} . \tag{15}$$

C Benchmarks for State-Space Physical Plants

We have derived a number of benchmarks from the control literature, which we report below. As discussed, in this paper, we disregard the presence of output signals, that is the effect of matrices $\mathbf{C}, \mathbf{D}$, and use direct state-feedback.

System	A	B	C	D
Automotive Cruise System	-0.05	0.03125	0.032	0
DC Motor Rate	$\begin{bmatrix} -1000 & -104 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 200 \end{bmatrix}$	0
Helicopter Longitudinal Motion	$\begin{bmatrix} -0.42 & 0.0168 & -0.396 \\ 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$	$\begin{bmatrix} 16 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.6125 & -0.6125 & 15.44 \end{bmatrix}$	0
Pendulum	$\begin{bmatrix} 0 & -2.453 \\ 4 & 0 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -1.225 & -2.15 \end{bmatrix}$	9.8
Inverted Pendulum	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 & -0.75 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
Magnetic Suspension	$\begin{bmatrix} 0 & 31.25 \\ 32 & 0 \end{bmatrix}$	$\begin{bmatrix} 8 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 9.766 \end{bmatrix}$	0
Magnetic Pointer	$\begin{bmatrix} -0.271 & -0.05336 & 0 \\ 0.03125 & 0 & 0 \\ 0 & 0.01562 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.5888 & -0.2562 \end{bmatrix}$	0
1/4 Car Suspension	$\begin{bmatrix} -516.1 & -222.1 & -79.75 & -33.06 \\ 256 & 0 & 0 & 0 \\ 0 & 64 & 0 & 0 \\ 0 & 0 & 32 & 0 \end{bmatrix}$	$\begin{bmatrix} 8 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 9.969 & 4.133 \end{bmatrix}$	0
Computer Tape Driver	$\begin{bmatrix} -1760 & -976.6 & -457.8 \\ 1024 & 0 & 0 \\ 0 & 512 & 0 \end{bmatrix}$	$\begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1.5 & 2.051 & 1.144 \end{bmatrix}$	0

Table 2. Benchmarks for State-Space Physical Plants