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

No Author Given

No Institute Given

Abstract. This work presents a sound and automated approach to synthesise digital feedback control architectures for physical plants represented as linear, time invariant models. Models are encompassed by dynamical equations with inputs, evolving over a continuous state space describing the evolution of the physical variables of the system. The approach is divided in two main parts, brought together by a counter-example guided inductive synthesis (CEGIS) loop, as follows. First, we devise a static feedback controller that stabilises the system; then, we verify its potential safety by means of either of the following approaches: forward reachability analysis via unfolding of the dynamics, or invariant set generation via abstract acceleration; further, if the above step fails, we loop back and invoke a CEGIS call to seek alternative stabilising feedbacks, benefiting from the obtained counter-examples to safety. [Mention GA and Abstract Acceleration here? Alessandro: we are not using GA. We don't think that we should describe it here.]

GA? Genetic Algorithms? We don't use GA?

The approach is proven sound by accounting for errors due to the digitalisation of signals that are manipulated by the controller block. The proposed synthesis approaches and a developed tool are evaluated considering a set of intricate physical plant models taken from the digital control literature.

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

1 Introduction

Linear Time Invariant (LTI) control models represent a broad class of dynamical systems with significant impact in numerous application areas such as the life sciences, robotics, and engineering [2, 11]. Control synthesis for LTI is broadly investigated, however the use of digital control architectures adds new challenges due to the effects of finite-precision arithmetic, time discretization, and quantisation noise, which are typically introduced by Analogue-to-Digital (A/D) and Digital-to-Analogue (D/A) conversion. Whilst the research on digital control is well developed [2], automated and sound control synthesis is a challenging task.

Whilst there is a vast literature on validation of control systems, the subject of control synthesis has been somewhat restricted in the properties it explores.

Traditionally, this focuses mainly on stability [1, 23], or using switched models [21]. Reachability tools are also used when discretizing the state-space for verification and parametric synthesis [6].

In this work, we propose the synthesis of *safe* control algorithms for LTI models taking into consideration the continuous domain of the plant, alongside the discrete domain of the controller and the hybrid elements that relate them. Due to the complexity of such systems, in this particular work we focus on linear models and perform correct-by-design synthesis of safe digital controllers.

We describe and evaluate two approaches for synthesizing digital controllers for state-space physical plants. Both approaches consider a counterexample guided inductive synthesis (CEGIS) loop. In the first approach, we devise a digital controller that stabilizes the system; then, we verify its potential safety by unfolding the dynamics of the system and checking for a completeness threshold. In the second approach,...

we have to connect both paragraphs...

To this end, we formalise the notion of solution generalisation for Counter-Example Guided Inductive Synthesis (CEGIS), which makes use of abstraction refinement in order to achieve scalability. This strategy enables the inductive generalisation engine inside CEGIS to look for candidate solutions in a reduced solution space.

TODO: Add contributions (Cristina)

2 Related work

CEGIS Program synthesis is the problem of computing correct-by-design programs from high-level specifications, and algorithms for this problem have made substantial progress in recent years. One such approach [12] inductively synthesizes invariants to generate the desired programs.

Program synthesisers are an ideal fit for synthesis of controllers since the semantics of programs capture effects such as FWL precisely. In [20], the authors use CEGIS for the synthesis of switching controllers for stabilizing continuous-time plants with polynomial dynamics. The work extends to its application on affine systems, finding its major challenge in the hardness of solving linear arithmetic with the state-of-the-art SMT solvers. Since this approach uses switching states instead of linear dynamics in the digital controller, it entirely circumvents the FWL problem. It is also not suitable for the kind of control we seek to synthesize. Moreover, in [21] the same authors use a CEGIS-based approach for synthesizing continuous-time switching controllers that guarantee reach while stay properties of a closed loop system. The solution is based on synthesizing control Lyapunov functions for switched systems, that yield switching controllers with a guaranteed minimum dwell time in each mode.

In [1], Abate et al. focus on synthesising stabilizing controllers for continuous plants by exploiting advantage in bit-accurate verification of C programs to obtain a verifier for software-implemented digital control [4]. While they also make use of a CEGIS-based technique, their approach is restricted to synthesising controllers for hybrid closed-loop systems that are stable and only consider transfer function models. In contrast, in the current paper, we consider a state-space representation of the physical system, which allows us to synthesise controllers that make the closed-loop system *safe* as well as stable. Moreover, a key aspect of the second approach we evaluate is that we integrate abstraction refinement inside the CEGIS loop.

The tool Pessoa [**?**] is a MatLab toolbox that synthesises correct-by-design embedded control software. It is based on the principle of describing a physical system as an equivalent finite-state machine and computing reachability over the finite-state machine. Pessoa can synthesise safe embedded controller software for a range of properties based on a safety specification over state values, as well as specification on characteristic functions of the controller. However, Pessoa does not have native inclusion of the ADC/DAC quantization error needed for synthesising a fixed point digital controller.

We use a combination of a synthesis engine with a control verification tool that addresses the challenges presented here in the form of FWL effects and stability measures for LTI SISO controllers. We take the former from [7]. The latter is based on [3], but adapted to state-space representation and enhanced to include evaluation of quantization effects.

something about the acceleration verification engine

Discretization Effects The classical approach to control synthesis has often disregarded quantization as a minimal effect, thus proving unsound in the case of digital control systems. Modern techniques focus on different elements of the discretization effect, including delayed response [8], and Finite Word Length (FWL) semantics with the goal to either verify (e.g. [3]) or optimize (e.g. [16]) 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 [10], 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 [27] uses μ calculus to directly model the digital controller so that the selected parameters are stable by design. The analyses in [22, 26] rely on an invariant computation on the discrete system dynamics using Semi-Definite Programming (SDP). While the former uses 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 approach is taken by [17], where the verification of existing code is performed against a known model by extracting an LTI model of the code through symbolic execution. In order to account for rounding errors, an upper bound is introduced in the verification phase. The work in [18, 19] 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, [13] evaluates the dynamics of the quantization error and binds their trajectory to a known region over a finite period of time. This technique works for both linear and non-linear systems. This paper uses an approach that projects the maximum effect of the noise over an unbounded time and uses it as a 'buffer' region for the safety specification.

3 Preliminaries

3.1 State-space representation of physical systems

We consider models of physical plants expressed as ordinary differential equations (ODEs), with full state information (namely where 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 $x(0)$. Eq. (1) is soundly discretised in time (as shown in the Appendix) into

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

where $k \in \mathbb{N}$, and with initial state $x_0 = x(0)$. Later, we will deal with the issue of variable quantisation, as introduced by the ADC and DAC blocks.

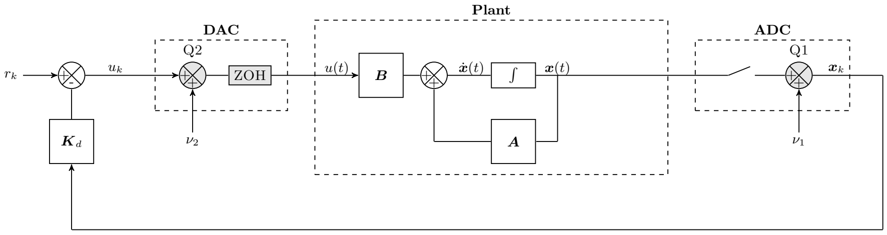


Fig. 1. Closed-loop digital control system

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 affect the properties and behaviours of the continuous process (the plant) of interest. We are interested in the synthesis of digital feedback, as implemented on modern FPGAs or ICs. The most basic feedback architecture is the state feedback, where the control action u_k (notice we work with the discretised signal) is computed by:

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

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

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

The gain matrix K can be set so that the closed-loop digitalised dynamics are shaped as desired, for instance according to a specific stability goal or around a specific dynamical behaviour [2]. As argued later, in this work, we will target more complex objectives, such as quantitative safety requirements that are not typical in the digital control literature.

3.3 Stability of closed-loop systems

Asymptotic stability boils down to Lyapunov stability and convergence to an equilibrium point . Lyapunov stability requires that model trajectories remain in a neighbourhood of the equilibrium point. In the case of closed-loop LTI models as in (4), the equilibrium point is the origin (if the reference is equal to zero), or an offset thereof. In this paper, we consider systems with a reference signal equal to zero.

add reference

A discrete-time system as (4) is asymptotically stable if and only if 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 value are less than one [2]. In this paper, we express the stability specification $\phi_{stability}$ in terms of a check known as Jury's criterion [9]. This condition boils down to an easy algebraic formula to select the entries of matrix K, so that the closed-loop dynamics are shaped as desired. A detailed description of this specification can be found in the Appendix A.

3.4 Safety specifications for dynamical systems

As mentioned earlier, in this work we are not bound to the synthesis of digital stabilising 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 in K) must ensure that the states never exist such requirement. Note that a stable, closed-loop system is not necessarily a safe system: indeed, the state values may leave the safe whilst they converge to the equilibrium, which is typical in the case of oscillatory dynamics.

In this work, the safety property represented by ϕ_{safety} is encoded 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 constraints on the input signal u_k, which we assume has given bounds $\overline{u}$ and $\underline{u}$, such that $\underline{u} \leq u_k \leq \overline{u}, \forall k \geq 0$. In practice, this means that the control input saturates in view of physical constraints.

3.5 Numeric representations and soundness

I've moved the following paragraph to this section, from that on safety specs.

explain and clarify notation.

The states values must be computed considering the digital controller error, i.e., the error introduced by the ADC and the digital controller. Thus, the states are computed using the following equations. We use $\mathcal{F}_{\langle I_c,F_c\rangle}$ to to denote numbers represented with the precision of the digital controller. All other numbers are represented with the precision of our plant model:

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

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 its states. The second is the numeric error introduced by the digital controller, which performs operations in the domain of fixed-point numbers. In this section we formally outline the errors introduced by fixed-point representation of numbers. In further sections we will differentiate between these two errors, by referring to them as the *plant error* and the *digital controller error* respectively.

notice this notation differs from the one used above

why R, comment on that.

Let $\mathbb{R}\langle I,F\rangle$ be 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:

the following maths is loose. fix as follows: define terms with their domain. define operation. apply operator $\mathcal{F}$ above. derive bound.

same as above.

why not? comment further.

1. **Truncation:** Let $\mathcal{F}\langle I,F\rangle(\cdot) : \mathbb{R} \to \mathbb{R}\langle I,F\rangle, \tilde{x} = \mathcal{F}\langle I,F\rangle(x) = x - \delta_T : \delta_T = x \,\%_{c_m}\, \tilde{x}$, where $\%_{c_m}$ is the modulus operation performed on the last bit. So δ_T is the truncation error and it will propagate across operations.
2. **Rounding:** The following errors appear in simple operations:
   (a) Addition/Subtraction: these operations only propagate errors coming from truncation of the operands. $c_1 \pm c_2 = c_3 + \delta_3 : |\delta_3| \leq |\delta_{T1}| + |\delta_{T2}|$
   (b) Multiplication: $c_1 * c_2 = c_3 + \delta_3 : |\delta_3| \leq |\delta_{T1} * c_2| + |\delta_{T2} * c_1| + c_m$
   (c) Division will not be applied in these implementations.
3. **Overflow:** The finite word length set a maximum representable number in the domain $(\pm(2^{I-1}+1-2^{-F}))$, which means that any number outside this range cannot be represented by the domain. Our tool currently assumes no saturation function and verifies that no state overflows.

Thus far we have presented a verification procedure for a digitalised plant in a continuous space (*ie* the range of the quantisers is still $\mathbb{R}$. The next step we must realise is that the controller exists in a digital program, hence its domain is $\mathbb{R}\langle I,F\rangle$. We have $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{x} \in \mathbb{R}^*$, $\boldsymbol{K}_d, \boldsymbol{u}, \hat{\boldsymbol{x}} = \mathcal{F}\langle I,F\rangle(\boldsymbol{x}) \in \mathbb{R}\langle I,F\rangle^*$ Where

what is next? a domain? define formally, no back-of-the-envelope mathsy stuff.

yes, please reformat the discussion below so that it fits the discussion above.

$*$ represents the appropriate dimension for each matrix/vector. Unfortunately this means that although the closed loop dynamics are in the reals, they suffer from the fixed word restrictions in their composing elements. Apart from the quantisation errors q_1, q_2 there are rounding errors in the calculation of u_k. These will introduce a new source of quantisation noise which we may denote $\boldsymbol{\nu}_3 \in [-q_3 \quad q_3]$. The error in $\boldsymbol{u}$ corresponds of the multiplication of $\boldsymbol{K}_d\hat{\boldsymbol{x}}$, hence $q_3 = \bar{\delta}_u = (|\boldsymbol{K}|_1 + 1)c_m$

4 CEGIS of Safe Controllers for LTI Systems with Solution Generalisation

In this section we outline the general architecture of the program synthesiser used in both approachesWe then give the details of the two methods of solution generalisation that we compare; the first checking safety by unwinding the transition relation to a completeness threshold; and the second employing abstract acceleration.

4.1 General architecture of the program synthesiser

The input specification provided to the program synthesizer is of the form $\exists \boldsymbol{P}.\forall \boldsymbol{x}.\sigma(\boldsymbol{x}, \boldsymbol{P})$ where $\boldsymbol{P}$ ranges over functions, $\boldsymbol{x}$ ranges over ground terms and σ is a quantifier-free formula. We interpret the ground terms over some finite domain $\mathcal{D}$.

now variables are vectorised, however previously the model variables aren't. please align notation. I do not like to use vectors unless essential, let's say it's a waste of ink in printouts.

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 $\boldsymbol{P}$ satisfying the specification $\sigma(\boldsymbol{x}, \boldsymbol{P})$ for all $\boldsymbol{x}$ in INPUTS (as opposed to all $\boldsymbol{x} \in \mathcal{D}$). If the synthesis phase succeeds in finding a witness $\boldsymbol{P}$, 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., $\boldsymbol{P}$ satisfies the specification $\sigma(\boldsymbol{x}, \boldsymbol{P})$ for all $\boldsymbol{x} \in \mathcal{D}$). 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.

It is worth noting that each iteration of the loop adds a new input to the finite set INPUTS that is used for synthesis. Given that the full set of inputs $\mathcal{D}$ is finite, this means that the refinement loop can only iterate a finite number of times.

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

The following paragraph has been already stated above. Let us clarify this is a useful repetition to recap things?

The synthesis problem we are trying to solve is the following: find a digital controller K (see Eq. 3) that makes the closed-loop system 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 and the reference signal to be set to zero.

When mapping back to the notation used for describing the general architecture of the program synthesiser, the controller K denotes P and (x_0, u_k) represents x. While the controller's precision $\langle I_c, F_c \rangle$ is given, we can vary the plant's precision $\langle I_p, F_p \rangle$ such that $\mathbb{R}\langle I_p, F_p \rangle \supseteq \mathbb{R}\langle I_c, F_c \rangle$.

4.3 Solution generalisation

Given the large solution space to be searched during the inductive synthesis phase, we parametrise the target language, which induces a lattice of progressively more expressive languages. This enables us to start by attempting to synthesise a program at the lowest point on this lattice and increase the parameters of the language until we reach a point at which the synthesis succeeds. Whenever we manage to synthesise a candidate solution at some lower point on the lattice, we must apply some form of solution generalisation to obtain a full solution (i.e. a solution in the original search space).

As well as giving us an automatic search procedure, this parametrisation greatly increases the efficiency of our system since languages low down the lattice are very easy to decide safety for (i.e. the validation oracle finds an answer fast). If a program can be synthesised in a low-complexity language, the whole procedure finishes much faster than if synthesis had been attempted in a high-complexity language.

We investigate two different approaches of integrating solution generalisation with CEGIS:

1. The first approach (described in Sec. 4.4) uses a multi-staged verification phase and performs solution generalisation based on the plant precision, the number of loop unrollings, and the sampling rate. Essentially, it finds a controller for an individual initial state and input with a bounded time horizon and generalises to all states, inputs, and time horizons.
2. The second approach (described in Sec 4.5) finds a controller for a continuous initial set of states and input set, over an abstraction of the continuous dynamics that conforms to witness proofs at specific times; it further generalises the solution over all times by refining the number of witnesses.

clarify use of abstract acceleration. add references.

4.4 CEGIS with multi-staged verification

The controller synthesis is described in Fig. 2. One important observation is that, for this approach, we verify and synthesise a controller over k time steps. We then compute a completeness threshold, $\overline{k}$, for this controller, and verify the controller correctness to $\overline{k}$ time steps. Essentially, $\overline{k}$ represents the number of iterations required to sufficiently unwind the closed-loop state-space system such

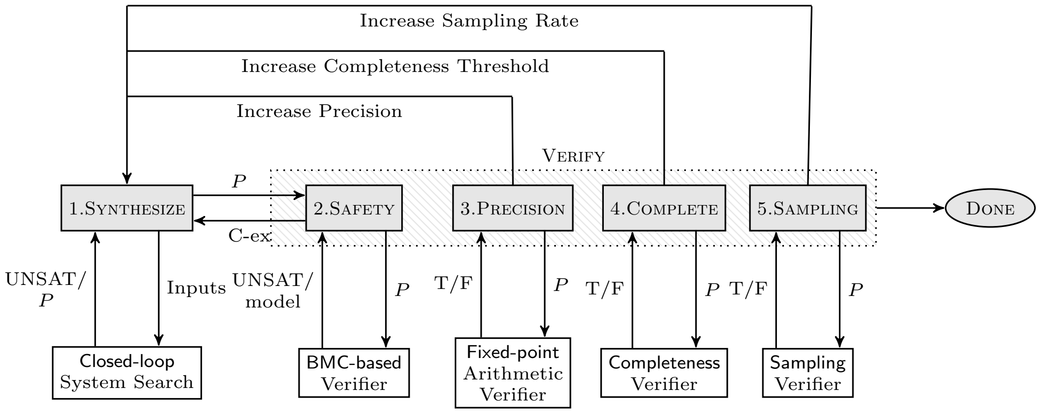


Fig. 2. CEGIS with multi-staged verification

that the boundaries are not violated for any $k>\overline{k}$ (i.e. it is sufficient to unwind the closed-loop state-space system up to $\overline{k}$ in order to ensure that ϕ_{safety} holds).

Next, we describe the different phases in Fig. 2 (blocks 1 to 5):

please eventually itemize next

1. The inductive synthesis phase (i.e. SYNTHESIZE) uses BMC to compute a candidate solution K that satisfies both the stability criteria detailed in Sec. 3.3 and the safety specification in Sec. 3.4. Regarding the latter, we synthesise a safe controller by unwinding the transition system k steps and picking a controller K and an initial state such that the states at each step do not violate the safety criteria. 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 all these assumptions are sound are performed by subsequent VERIFY stages.

```
function $safetyCheck()$
    assert($\underline{u} \leq u \leq \overline{u}$)
    while $i = 0; i < k; i++$ do
        $u = (plant\_typet)((controller\_typet)K * (controller\_typet)x)$
        $x = A * x + B * u$
        assert($\underline{x} \leq x \leq \overline{x}$ )
    end while
end function
```

2. The first VERIFY stage, SAFETY, checks that the candidate solution K is safe for all possible initial states, i.e., does not reach an unsafe state within k steps where again we assume k to be under the completeness threshold. After

unwinding the transition system corresponding to the previously synthesised controller k steps, we check that the safety specification holds for any initial state.

This is relatively computationally expensive if the bounds on the allowed initial states are large. However, we can show that, if the 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 number of states.

please will Alessandro write out the proof that checking only the corner cases is valid for the safety verification - EP

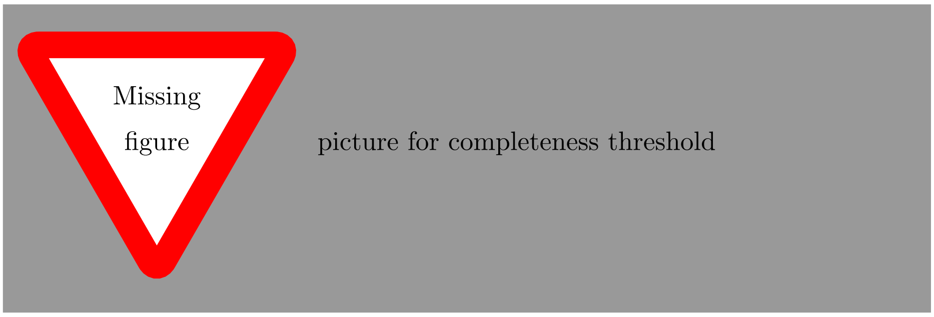
Proof. Consider the set of initial states, X_0, assumed to be convex since it's a box. Name v_i its vertices, where $i = 1, \ldots, 2^n$. Thus any point $x \in X_0$ can be expressed by convexity as $x = \sum_{i=1}^{2^n} \alpha_i v_i$, where $\sum_{i=1}^{2^n} \alpha_i = 1$. So then if $x_0 = x$, then $x_k = (A_d - B_d K)^k x = (A_d - B_d K)^k \sum_{i=1}^{2^n} \alpha_i v_i = \sum_{i=1}^{2^n} \alpha_i (A_d - B_d K)^k v_i = \sum_{i=1}^{2^n} \alpha_i x_k^i$, where I have denoted with x_k^i the trajectories obtained from the single vertices v_i.QED

3. The second VERIFY stage, PRECISION, restores soundness with respect to the plant's precision by using interval arithmetic [14] 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 that the boundaries are not violated for any $k' > k$. For this purpose, we compute the completeness threshold $\overline{k}$ for the current candidate K and check that $k \leq \overline{k}$.

A stable control system is known to have converging dynamics, i.e., the distance from the reference (in this case the origin) decreases over time.

please will Alessandro write the proof for the completeness threshold here - EP



Proof sketch. Consider spectrum of matrix $(A_d - B_d K)$ and select eigenvalue with smallest (non trivial) imaginary value. call such value ϑ. At every time interval $kT_s, (k+1)T_s$ (T_s being the sampling time), the dynamics projected on the corresponding eigenspace rotate of ϑT_s radiants. So surely taking k as the ceiling of $\frac{2\pi}{\vartheta T_s}$, after k steps we've completed a rotation. This is the CT.

5. The last VERIFY stage, SAMPLING, ensures that the current sampling rate is large enough by using an external checker (TODO: add more details here once we have them).

4.5 CEGIS with abstraction refinement

The verification of a hybrid continuous/discrete system over an unbounded time, requires the non-deterministic simulation of an infinite set of states, which is undecidable. Our initial approach creates an abstraction of this system by establishing bounds on the error between such a system and a discrete-time/continuous-space model. This model is still however, quite complex, hard to verify, and many of the states involved are inconsequential to the safety specification. Our second model, rather than look at a bisimulation in discrete time, explores a much higher abstraction, evaluating the unbounded time-continuous space reach tube as a set of inequalities that represent only the safety specification. We begin with a minimum set of linear inequalities describing the constraints caused by the input and the initial state during a single iteration, in addition to a stability specification defined by Jury's criteria, and try to synthesise a controller that meets these specs. A second abstraction that encompasses the accelerated continuous reach-tube over an unbounded time is used to verify that the controller meets the full safety specification. If this step fails, we find a counterexample iteration and initial state and create a new constraint to refine our abstraction. The advantage of this model is that it starts with a very simple description regardless of the complexity of the overall dynamics, and only expands to more complex models when a solution cannot be found.

specific notion that has not been formally define - do not use

TODO: Move as much as possible from Sec 4.5 to the appendix (e.g. discussion on abstract acceleration). Merge the rest in Sec 4.5.

Verification of a controllable system Let us recall the formulas used for single-input and single-output (SISO) systems in sections B.4 and ??. A controllable system with quantization will have the following dynamics:

eliminate vectors

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_t \boldsymbol{x}_k + \boldsymbol{B}_{tp} \boldsymbol{r}_k + \boldsymbol{B}_{tn}(\boldsymbol{\nu}_1 + \boldsymbol{\nu}_2 + \boldsymbol{\nu}_3) \tag{6}$$

$$\boldsymbol{A}_t = \boldsymbol{T}\boldsymbol{A}_d\boldsymbol{T}^{-1} \qquad \boldsymbol{B}_{tp} = \boldsymbol{T}\boldsymbol{B}_d \qquad \boldsymbol{B}_{tn} = \boldsymbol{T}\boldsymbol{B}_d \qquad \boldsymbol{T} = \boldsymbol{R}_{cf}\boldsymbol{R}^{-1}$$

$$\boldsymbol{A}_t = \boldsymbol{T}\boldsymbol{A}_d\boldsymbol{T}^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ k_n - a_n & k_{n-1} - a_{n-1} & k_{n-2} - a_{n-2} & \cdots & k_1 - a_1 \end{bmatrix}$$

The characteristic polynomial of the above matrix is:

$$z^n + \sum_{i=1}^{p} (a_i - k_i) z^{n-i}$$

For a stabilized closed-loop, which is necessary for our safety property, we need these roots to remain within the unit circle, which we verify using Jury's criteria. Notice that in our polynomials $c_0 = 1$ and we only use reals.

Once we have established convergence, we may examine the specification of the state and input spaces ($\phi_{safety} \wedge \phi_{input}$). Whilst there are other properties, we may include in the verification, we will solely focus on these for simplicity of the explanation. We assume that we do not want saturation in our system, hence the control loop has only one mode. We have also identified the quantization noise as a bounded non-deterministic input. The first step constraints

$$\boldsymbol{K} \models \boldsymbol{u}_k = -\boldsymbol{K}\boldsymbol{x}_k : \phi_{safety} \wedge \phi_{input} \tag{7}$$

The second step is evaluating the progression of the system over time to evaluate $\boldsymbol{x}_{k+1} \models \phi_{safety}$. For this step we require an initial set from which the system progresses. Obviously, this set $\boldsymbol{x}_0$ must satisfy the specification; otherwise, the system will be unsafe to begin with. We accept a specification in the form $\boldsymbol{E}\boldsymbol{T}\boldsymbol{x}_0 < \boldsymbol{f}$. The presence of $\boldsymbol{T}$ is because this will typically be given in the original state-space. Accelerating (6), we obtain:

$$\boldsymbol{x} = \boldsymbol{A}_t^k \boldsymbol{x}_0 + \sum_{i=0}^{k-1} \boldsymbol{A}_t^i \boldsymbol{B}_{tp} \boldsymbol{r}_i + \sum_{i=0}^{k-1} \boldsymbol{A}_t^i \boldsymbol{B}_{tn} (\boldsymbol{\nu}_1 + \boldsymbol{\nu}_2 + \boldsymbol{\nu}_3) : \boldsymbol{x}_0 \models \phi_{init} \tag{8}$$

which would still require us to verify the system for every k up to infinity. Instead we use abstract acceleration [5], which translates the formula into

$$\begin{aligned} \hat{X}^{\#} =& \mathcal{A}_t X_0 + \mathcal{B}_{tp} R + \mathcal{B}_{tn} N \\ X_0 =& \{\boldsymbol{x} : \boldsymbol{x} \models \phi_{init}\} \\ R =& \{r : \boldsymbol{E}_r < \boldsymbol{f}_r\} \\ N =& \left\{ \boldsymbol{\nu}_1 + \boldsymbol{\nu}_2 + \boldsymbol{\nu}_3 : \nu_1 \in \begin{bmatrix} -\frac{q1}{2} & \frac{q1}{2} \end{bmatrix} \wedge \nu_2 \in \begin{bmatrix} -\frac{q2}{2} & \frac{q2}{2} \end{bmatrix} \wedge \nu_3 \in [-q3 \;\; q3] \right\} \end{aligned} \tag{9}$$

where $\mathcal{A}_t = \bigcup_{k=1}^{\infty} \boldsymbol{A}^k, \mathcal{B}_{tp} = \bigcup_{k=1}^{\infty} \sum_{i=0}^{k} \boldsymbol{A}^i \boldsymbol{B}_{tp}, \mathcal{B}_{tn} = \bigcup_{k=1}^{\infty} \sum_{i=0}^{k} \boldsymbol{A}^i \boldsymbol{B}_{tn}$ are abstract matrices for the system in (6) and r is a non-deterministic parametric input (*ie* $\forall k > 0, r_{k+1} = r_k$) whereas the set N is composed of non-deterministic variable inputs.

Controller synthesis Now that we have explained the verification process, we may proceed to describe the synthesis procedure, which is illustrated in figure 3.

1. We start by preprocessing the dynamics. This can be computationally intensive and we don't want it to be part of the CEGIS loop.

   (a) Calculate $\boldsymbol{T}$ using equations (32)(33)(34)
   (b) Extract the Plant's characteristic polynomial coefficients from the last row of $P_a = (\boldsymbol{TAT}^-1)_{n*}$.
   (c) Calculate the noise set N from the quantizer resolutions and estimated round-off errors $(\frac{q_1}{2}, \frac{q_2}{2}, q_3)$
   (d) Calculate a set of initial bounds on K based on the input constraints.

$$\phi_{input} \Leftrightarrow u_k = r - Kx_k : x \in X^{\#} \wedge u_k \in [\underline{u} \quad \overline{u}]$$

2. Next we synthesise a candidate controller. Let $\mathcal{J}\langle I, F\rangle(P, \boldsymbol{T})$ be a program describing Jury's method.
   (a) Select a controller $\tilde{\boldsymbol{K}} \in \mathbb{R}\langle I, F\rangle^n$. This is achieved by solving a SAT formula $\mathcal{J}_K = \mathcal{J}\langle I, F\rangle(P_{a-k}, \boldsymbol{T}) \wedge \phi_{input}$ that satisfies the input constraint and Jury's criteria for $P_{a-k} = z^n + \sum_{i=1}^{n}(a_i - k_i)z^{n-i} : k_i \in \boldsymbol{K} \wedge \boldsymbol{K} = \tilde{\boldsymbol{K}}\boldsymbol{T}$.
   (b) 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 using abstract acceleration.
   (a) Calculate the abstract matrices $\mathcal{A}_t, \mathcal{B}_{tp}, \mathcal{B}_{tn}$ for the candidate closed loop solution.
   (b) Evaluate $\hat{X}^{\#} \models \Psi$. If the verification holds we have a solution, and exit the loop.
4. If the candidate solution is not valid we refine the abstraction used by the synthesiser
   (a) Find the constraints that invalidate the property. These are the rows of $\boldsymbol{E} \in \Psi$ that intersect/segment $\hat{X}^{\#}$. Transforming these rows into the eigenspace of $\boldsymbol{A}_t$, and applying backward acceleration to obtain a set of counterexamples for the eigenvalues, which we define as $\phi(\Lambda)$. This is a constraint in the spectrum (*ie* frequency response transfer function) of the closed loop dynamics.
   (b) Transform the soft constraints on the spectrum into soft constraints on the coefficients of the characteristic polynomial. $z^n + \sum_{i=1}^{n}[a_i - k_i]z^{n-i} = \prod_{i=1}^{n}(z - [\lambda_i]) : |[\lambda_i]| < 1 \wedge [\lambda_i] \models \phi_\Lambda$.
   (c) Pass the refined abstraction with the new contraints to the synthesiser, as well as a counterexample plant coefficients (P_a) and repeat the loop again. We note that the Abstraction refinement is a soft constraint on the proposed model which can increase or decrease the parameter space at any time, whereas the concrete counterexample passed to the synthesiser is monotonically decreasing. The first one improves speed of the tool whereas the second ensures convergence.

5 Experimental Evaluation

5.1 Description of the benchmarks

A set of state-space models for different classes of systems were extracted from the literature and employed for validating our automated synthesis methodology.

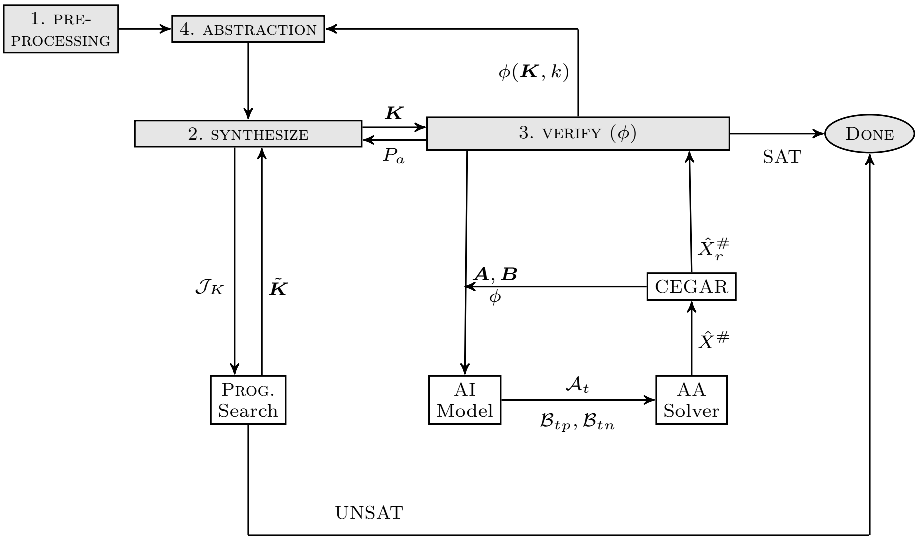


Fig. 3. CEGIS with abstraction refinement

All the systems are SISO models of the following form:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \\ y(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}u(t) \end{cases} \tag{10}$$

Table 1 summarizes all benchmarks extracted from the control system literature [11, 15, 24, 25]. *DC Motor Position and Rate* plants describes the angular position and velocity of a DC Motor, respectively, that coupled with wheels and cables provides translational motion. *Aircraft Pitch Dynamics* plant represents the air vehicle orientation and control based on the pitch dimension. *Ball Magnetic Levitation* plant describes a physical model to keep a metal ball hanged in air via a magnetic field . *Buck Converter* plant represents a power converter model that steps down voltage from the supply to the load while stepping up current. *Boost Converter* plant also represents a power converter model, but here it steps up voltage from the supply to the load while stepping down current. *Buck-boost Converter* plant represents a converter model that has an output voltage magnitude, which is either greater than or less than the input voltage magnitude.

Automotive Cruise System plant describes a physical model that represents the speed of a motor vehicle. *Helicopter Longitudinal Motion* plant provides the longitudinal motion model of a helicopter. *Inverted Pendulum* plant describes a pendulum model with its center of mass above its pivot point. *Magnetic Suspension* plant provides the physical model for which a given object is suspended via a magnetic field. *Magnetized Pointer* plant describes a physical model that is rotated through interaction with magnetic fields (typically employed in analog gauges and indicators). *1/4 Car Suspension* plant presents a physical model that connects a car to its wheels and allows relative motion between both parts. *Computer Tape Driver* plant describes a physical model to read and write data on a storage device in the computer. *USCG Cutter Tampa Heading Angle* plant presents a physical model for the yaw angle dynamics of a US warship USCG Cutter Tampa.

All benchmarks were discretized with different sample times using the approach described in the Appendix B. All experiments were performed considering $x_i^- = -1$ and $x_i^+ = 1$ and the inputs $u_k = 0, \forall k > 0$.

We have also conducted the experimental evaluation on a 12-core 2.40 GHz Intel Xeon E5-2440 with 96 GB of RAM and Linux OS. All times given are wall clock time in seconds as measured by the UNIX date command.

we have to describe the timeout...

5.2 Objectives

5.3 Results

5.4 Threats to validity

6 Extensions

6.1 LTI systems with output

LTI models in the form:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \quad : \quad \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{n\times m} \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \quad : \quad \boldsymbol{y} \in \mathbb{R}^o, \boldsymbol{C} \in \mathbb{R}^{o\times n}, \boldsymbol{D} \in \mathbb{R}^{o\times m} \end{aligned} \tag{11}$$

where $\dot{\boldsymbol{x}}(t)$ describes the state evolution equation; $\boldsymbol{y}(t)$ represents an instantaneous output equation; and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are matrices that fully specify a continuous plant. Eq. (1) is soundly discretized (as shown in the appendix) into

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}_d\boldsymbol{x}_k + \boldsymbol{D}_d\boldsymbol{u}_k. \end{aligned} \tag{12}$$

References

1. A. Abate, I. Bessa, D. Cattaruzza, L. C. Cordeiro, C. David, P. Kesseli, and D. Kroening. Sound and automated synthesis of digital stabilizing controllers for continuous plants. *Hybrid Systems: Computation and Control (to appear)*, 2017.
2. K. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Prentice Hall information and system sciences series. Prentice Hall, 1997.
3. I. Bessa, H. Ismail, L. Cordeiro, and J. Filho. Verification of fixed-point digital controllers using direct and delta forms realizations. *Design Autom. for Emb. Sys.*, 20(2):95–126, 2016.
4. 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.
5. D. Cattaruzza, A. Abate, P. Schrammel, and D. Kroening. Unbounded-time analysis of guarded LTI systems with inputs by abstract acceleration. volume 9291 of *LNCS*, pages 312–331. Springer, 2015.
6. A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Parameter synthesis with IC3. In *FMCAD*, pages 165–168, 2013.
7. 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.
8. 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.
9. S. Fadali and A. Visioli. *Digital Control Engineering: Analysis and Design*, volume 303 of *Electronics & Electrical*. Elsevier/Academic Press, 2009.
10. 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.
11. G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Pearson, 7th edition, 2015.
12. S. Itzhaky, S. Gulwani, N. Immerman, and M. Sagiv. A simple inductive synthesis methodology and its applications. In *ACM Sigplan Notices*, volume 45, pages 36–46. ACM, 2010.
13. D. Liberzon. Hybrid feedback stabilization of systems with quantized signals. *Automatica*, 39(9):1543–1554, 2003.
14. R. E. Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.
15. 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.
16. 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.
17. J. Park, M. Pajic, I. Lee, and O. Sokolsky. Scalable verification of linear controller software. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 662–679. Springer, 2016.
18. B. Picasso and A. Bicchi. Stabilization of lti systems with quantized state-quantized input static feedback. In *International Workshop on Hybrid Systems: Computation and Control*, pages 405–416. Springer, 2003.

19. B. Picasso, F. Gouaisbaut, and A. Bicchi. Construction of invariant and attractive sets for quantized-input linear systems. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 824–829. IEEE, 2002.
20. H. Ravanbakhsh and S. Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *54th IEEE Conference on Decision and Control, CDC*, pages 4232–4239, 2015.
21. H. Ravanbakhsh and S. Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *2016 International Conference on Embedded Software, EMSOFT 2016, Pittsburgh, Pennsylvania, USA, October 1-7, 2016*, pages 8:1–8:10, 2016.
22. P. Roux, R. Jobredeaux, and P. Garoche. Closed loop analysis of control command software. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC*, pages 108–117, 2015.
23. M. S. Sadabadi and D. Peaucelle. From static output feedback to structured robust static output feedback: A survey. *Annual Reviews in Control*, 2016.
24. R. H. G. Tan and L. Y. H. Hoo. Dc-dc converter modeling and simulation using state space approach. In *2015 IEEE Conference on Energy Conversion (CENCON)*, pages 42–47, Oct 2015.
25. C. M. University, U. of Michigan, and U. of Detroit Mercy. Control tutorials for MATLAB and SIMULINK.
26. T. E. Wang, P. Garoche, P. Roux, R. Jobredeaux, and E. Feron. Formal analysis of robustness at model and code level. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC*, pages 125–134, 2016.
27. 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 systems

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

$$P(z) = \det(zI_n - A_d + B_d K), \tag{13}$$

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, the system is stable. 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 $(a_0, a_1, ..., a_{N-1})$ of $P(z)$ for checking the validity of the stability property $\phi_{stability}$.

In particular, Jury stability test is already explained in the control system literature (*e.g.* [9]). This study, however, limits itself to explain the encoding of Jury's criterion for the formal synthesis purpose. For the stability test procedure, the following Jury matrix $M = [m_{ij}]_{(2N-2)\times N}$ is built from $P(z)$ coefficients:

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

where $V^{(k)} = [v_{ij}^{(k)}]_{2\times N}$ 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: $S(1) > 0$;
- R_2: $(-1)^N P(-1) > 0$;
- R_3: $|a_0| < a_N$;
- R_4: $m_{11} > 0 \iff m_{31} \wedge m_{51} \wedge \cdots \wedge m_{(2N-3)(1)}$.

The stability property $\phi_{stability}$ is then encoded as follows:

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

B LTI system Transformations and Equivalencies

B.1 Discretising 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).

$$\dot{x}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{w}(t). \tag{14}$$

with $\boldsymbol{w} \sim \mathcal{N}(0, Q)$.
Furthermore, a control system may have a derived output that is a linear combination of its states and inputs, which may restrict the observability of the state-space from the output space.

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t). \tag{15}$$

Discretization of an continuous dynamical system turns the ODE into a difference equation, assuming zero-order hold for the input $\boldsymbol{u}$ and continuous integration for the noise $\boldsymbol{w}$, to

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{u}_k + \boldsymbol{w}_k, &(16)\\ y_k &= \boldsymbol{C}_d\boldsymbol{x}_k + \boldsymbol{D}_d\boldsymbol{u}_k. &(17)\end{aligned}$$

with covariance for $\boldsymbol{w}_k \sim \mathcal{N}(0, Q_d)$, where

$$\begin{aligned} \boldsymbol{A}_d &= e^{\boldsymbol{A}T_s} = \mathcal{L}^{-1}(s\boldsymbol{I} - \boldsymbol{A})^{-1}{}_{t=T_s}, &(18)\\ \boldsymbol{B}_d &= \int_0^{T_s} e^{\boldsymbol{A}t} dt\ \boldsymbol{B} = \boldsymbol{A}^{-1}(\boldsymbol{A}_d - \boldsymbol{I})\boldsymbol{B}, &(19)\\ \boldsymbol{C}_d &= \boldsymbol{C}, &(20)\\ \boldsymbol{D}_d &= \boldsymbol{D}, &(21)\\ Q_d &= \int_0^{T_s} e^{\boldsymbol{A}t} Q e^{\boldsymbol{A}^T t} dt. &(22)\end{aligned}$$

and T_s is the sample time. Then

$$x(kT_s) = x_k \wedge y(kT_s) = y_k, \forall k.$$

Let $g_d(k) = \mathcal{G}(t, g(t), T)$ be a function that performs the discretization described above, where $g(t)$ represents the continuous dynamics and $g_d(k)$ the corresponding discrete dynamics. Let $G(s) = \mathcal{L}(g(t))$ and $G_d(z) = \mathcal{Z}(g_d(k))$ be the corresponding Laplace and Z-transforms of $g(t)$ and $g_d(k)$. Given this relation, we have

$$G_d(z) = G(z)|_{z=e^{sT}} : g_d(k) = \mathcal{G}(t, g(t), T) \wedge T < \frac{1}{.5 f_s},$$

where $.5f_s$ is the Nyquist frequency of $g(t)$. This last restriction is introduced to avoid the effects of aliasing which could cause "phantom poles" to appear otherwise. The eigenvalues of $\boldsymbol{A}_d$ corresponding to the poles of $G_d(z)$, and those of $\boldsymbol{A}$ corresponding to the poles of $G(s)$ are similarly related. $\hat{\lambda}_i = e^{-\lambda_i T} : \hat{\lambda}_i \in \sigma(\boldsymbol{A}_d) \wedge \lambda_i \in \sigma(\boldsymbol{A})$ where $\sigma(\cdot)$ is the spectrum of a matrix. The following remark is worth mentioning regarding the above.

Remark 1. The witnessed maximum amplitude of the discrete signals x_k, y_k may be smaller to that of $x(t), y(t)$ due to synchronism at fraction-frequency sampling. This means that reasoning about the state-space must consider the effects of this possibility as well of the case of maximal inputs from the continuous specification.

Since (16) is a bi-simulation of (1), we may use the semantics in section **??** to model continuous dynamical systems.

B.2 Overapproximating continuous space dynamics using discrete models

Let us recall equation 18. The discrete representation of the system dynamics discretized with sample time T_1 is ruled by the formula $\boldsymbol{A}_1 = e^{\boldsymbol{A}T_1}$. From matrix theory, we have

$$\boldsymbol{A}_1 = \boldsymbol{S}_1 \boldsymbol{J}_1 \boldsymbol{S}_1^{-1} = \boldsymbol{S}_1 e^{\boldsymbol{J}T_1} \boldsymbol{S}_1^{-1}. \tag{23}$$

Let $\boldsymbol{J}$ be a diagonal matrix, such that $\lambda_{1_i} = e^{\lambda_i T_1}, \forall \lambda \in \boldsymbol{J}$. Let us now take a different sample rate T_2, such that $\lambda_{2_i} = e^{\lambda_i T_2}, \forall \lambda \in \boldsymbol{J}$. We can then say that

$$\lambda_{2_i} = e^{\lambda_i T_1 T_2 / T_1} = \lambda_{1_i}^{T_2/T_1} \Rightarrow A_2 = A_1^{\frac{T_2}{T_1}} \tag{24}$$

We have also shown in the previous section that $x_k = A^n x_0$ is a witness of $x(t) : \dot{x}(t) = Ax(t)$ at time kT_s. Applying this notion to multiple sample times, we can state that $x_{k_i} = A_i^n x_0$ are a set of witnesses at times kT_i, and by extension the set $\{x = A_i^n x_0 : \forall n \in \mathbb{R}\}$ witnesses all states visited by the continuous system in time $t \in [0, \infty)$. This applies to any arbitrary sample time T_i.

B.3 Modelling quantization as noise

During any given ADC conversion, the continuous signal will be sampled in the real domain and transformed into a $\mathbb{R}\langle I_{adc}, F_{adc}\rangle$ value. this sampling uses a threshold which is defined by the less significant bit ($q_{adc} = 2^{-F_{adc}}$) of the ADC and some non-linearities of the circuitry. The overall conversion is

$$\mathcal{F}\langle I_{adc}, F_{adc}\rangle(y(t)) = y_k : y_k \in \left[y(t) - \frac{q_{adc}}{2} \quad y(t) + \frac{q_{adc}}{2} \right]$$

. If we denote the error in the conversion as $\nu_k = y_k - y(t) : t = nk$ then we may define some bounds for it $\nu_k \in [-\frac{q_{adc}}{2} \quad \frac{q_{adc}}{2}]$.

We will assume, for the purposes of this analysis, that the domain of the ADC is that of the digital controller (*ie* the quantizer includes any digital gain added in the code). The process of quantization in the DAC is similar except that it is going from $\mathbb{R}\langle I_{adc}, F_{adc}\rangle \rightarrow \mathbb{R}\langle I_{dac}, F_{dac}\rangle$. If these domains are the same ($I_{adc} = I_{dac}, F_{adc} = F_{dac}$), 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 infer that $\nu_1 \in [-\frac{q_1}{2}\ \ \frac{q_1}{2}], \nu_2 \in [-\frac{q_2}{2}\ \ \frac{q_2}{2}] : q_1 = q_{adc} \wedge q_2 = q_{dac}$. Note that 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:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\boldsymbol{x}_k + \boldsymbol{B}_d(\boldsymbol{u}_k + \boldsymbol{\nu}_{2k}) + \boldsymbol{w}_k, \tag{25}$$

$$y_k = \boldsymbol{C}_d\boldsymbol{x}_k + \boldsymbol{D}_d\boldsymbol{u}_k + \boldsymbol{\nu}_{1k}. \tag{26}$$

Which, when modelled into closed loop dynamics results in:

$$\boldsymbol{x}_{k+1} = (\boldsymbol{A}_d - \boldsymbol{B}_d\boldsymbol{K}_d\boldsymbol{C}_d)\boldsymbol{x}_k + \boldsymbol{B}_d\boldsymbol{\nu}_{2k} + \boldsymbol{w}_k, \tag{27}$$

$$y_k = \boldsymbol{C}_d\boldsymbol{x}_k + \boldsymbol{D}_d\boldsymbol{u}_k + \boldsymbol{\nu}_{1k}. \tag{28}$$

B.4 Controllable canonical form

Let us have a discrete time SISO system with an $n^{th}, m^{th} : n \geq m$ order armax model

$$y[k] = \sum_{i=1}^{n} a_i y[k-i] + \sum_{i=0}^{m} b_i u[k-i],$$

where y is the output of the system and u its input. The equivalent LTI dynamics of such a model are:

$$\begin{aligned}
\boldsymbol{x}_{k+1} =& \boldsymbol{A}_{cf}\boldsymbol{x}_k + \boldsymbol{B}_{cf}u_k : \boldsymbol{x}_k = [y_{k-n}\ \ \ldots y_{k-1}\ y_k]^T \\
y_k =& \boldsymbol{C}_{cf}\boldsymbol{x}_k + b_0 u_k \\
\boldsymbol{A}_{cf} =& \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix}, \boldsymbol{B}_{cf} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \\
\boldsymbol{C}_{cf} =& [b_n - a_n b_0\ b_{n-1} - a_{n-1}b_0 \cdots\ b_1 - a_1 b_0] : b_{i\in(m\ \ n]} = 0
\end{aligned} \tag{29}$$

where the coefficients a_i describe the characteristic polynomial of the system. This matrix shape is called a Controllable Canonical Form because the dynamics of the feedback are directly related to these coefficients. Specifically, given a feedback controller $\boldsymbol{K}$ for this system, the closed loop dynamics will have the shape

$$\boldsymbol{A}_{cf} + \boldsymbol{B}_{cf}\boldsymbol{K}_{cf} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ k_n - a_n & k_{n-1} - a_{n-1} & k_{n-2} - a_{n-2} & \cdots & k_1 - a_1 \end{bmatrix} \tag{30}$$

which means we can directly modify each coefficient by selecting a controller parameter. In the MIMO case, the Controllable Canonical Form has the following shape:

$$\boldsymbol{A}_{cf} = \begin{bmatrix} 0 & \boldsymbol{I}_{n-1} & 0 & \cdots & 0 \\ 0 & 0 & \boldsymbol{I}_{n-2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -\boldsymbol{a_n}\boldsymbol{M}_n & -\boldsymbol{a_{n-1}}\boldsymbol{M}_{n-1} & -\boldsymbol{a_{n-2}}\boldsymbol{M}_{n-2} & \cdots & -\boldsymbol{a_1}\boldsymbol{M}_1 \end{bmatrix}, \boldsymbol{B}_{cf} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \boldsymbol{M}_0 \end{bmatrix}$$

$$\boldsymbol{C}_{cf} = [\boldsymbol{\eta}_n \; \boldsymbol{\eta}_{n-1} \cdots \boldsymbol{\eta}_1] : \boldsymbol{I}_i, \boldsymbol{M}_i \in \mathbb{R}^{q_i \times q_i} \wedge \boldsymbol{a}_i \in \mathbb{R}^{q \times q_i} \wedge \boldsymbol{\eta}_i \in \mathbb{R}^o \wedge \boldsymbol{\eta}_{i \in (m \; n]} = 0$$

where q is the number of inputs, q_i the number of inputs with a transfer function of order of at least i, $\boldsymbol{I}_i$ an identity matrix, $\boldsymbol{M}_i$ an upper triangular matrix with 1s in its diagonal, $\boldsymbol{a_i}$ a matrix of coefficients of the i^th order on its diagonal, and o the number of outputs. Ideally we aim for $\boldsymbol{M}_i = \boldsymbol{I}_i, \forall i > 0$

Any LTI system can be transformed into a controllable canonical form by a matrix $\boldsymbol{T} : \boldsymbol{x} = \boldsymbol{T}\hat{\boldsymbol{x}}$ Then the dynamics of the controllable form are

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1} =& \boldsymbol{A}_{cf}\hat{\boldsymbol{x}}_k + \boldsymbol{B}_{cf}u_k : \boldsymbol{A}_{cf} = \boldsymbol{T}\boldsymbol{A}\boldsymbol{T}^{-1} \wedge \boldsymbol{B}_{cf} = \boldsymbol{T}\boldsymbol{B} \\ y_k =& \boldsymbol{C}_{cf}\hat{\boldsymbol{x}}_k + b_0 u_k : \boldsymbol{C}_{cf} = \boldsymbol{C}\boldsymbol{T}^{-1} \end{aligned} \tag{31}$$

where $\boldsymbol{T}$ can be calculated as follows:

Let

$$\boldsymbol{R} = [\boldsymbol{B} \; \boldsymbol{A}\boldsymbol{B} \; \boldsymbol{A}^2\boldsymbol{B} \ldots \boldsymbol{A}^{n-1}\boldsymbol{B}] \tag{32}$$

be the reachability matrix of the system, and

$$\boldsymbol{R}_{cf} = \begin{bmatrix} 1 & a_1 & a_2 & \ldots & a_{n-1} \\ 0 & 1 & a_1 & \ldots & a_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{33}$$

the reachability matrix of the canonical form. Then

$$\boldsymbol{T} = \boldsymbol{R}_{cf}\boldsymbol{R}^{-1} \tag{34}$$

Table 1. Benchmarks for State-Space Physical Plants.

System	**A**	**B**	**C**	**D**
DC Motor Position	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & -1.087 & 8587 \\ 0 & -9964 & -1.455 \times 10^6 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ -1.455 \times 10^6 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$	0
Aircraft Pitch Dynamics	$\begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$	0
Ball Magnetic Levitation	$\begin{bmatrix} 0 & 1 & 0 \\ 1752 & 0 & -34.07 \\ 0 & 0 & -38.27 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1.923 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$	0
Buck Converter	$\begin{bmatrix} 0 & -500 \\ 4545 & -1515 \end{bmatrix}$	$\begin{bmatrix} 125 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	0
Boost Converter	$\begin{bmatrix} 0 & -375 \\ 3409 & -1515 \end{bmatrix}$	$\begin{bmatrix} 500 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	0
Buck-boost Converter	$\begin{bmatrix} 0 & 375 \\ -3409 & -1515 \end{bmatrix}$	$\begin{bmatrix} 125 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	0
Automotive Cruise System	-0.05	0.03125	0.032	0
DC Motor Rate	$\begin{bmatrix} -1000 & -13 \\ -8 & 0 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 6.25 \end{bmatrix}$	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
USCG Cutter Tampa Heading Angle	$\begin{bmatrix} -0.271 & -0.05336 & 0 \\ 0.03125 & 0 & 0 \\ 0 & 0.01562 & 0 \end{bmatrix}$	$, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.5888 & -0.2562 \end{bmatrix}$	0