Noname manuscript No.
(will be inserted by the editor)

Formal Non-fragile Verification of Step Response Requirements for Digital State-Feedback Control Systems

Thiago Cavalcante · Iury Bessa · Eddie Filho · Lucas Cordeiro

Received: date / Accepted: date

Abstract We describe and evaluate a novel approach to formally verify whether a digital control system meets specifications related to step-response parameters. In particular, we obtain a state feedback controller designed for a system represented by a state-space model. Then we analyze whether its required specifications regarding settling time and maximum overshoot are met, using both open- and closed-loop forms and considering finite word-length (FWL) effects for the latter. We developed our verification approaches inside DSVerifier, which is a verification tool that employs bounded (and unbounded) model checking based on satisfiability modulo theories. Thus, DSVerifier checks performance requirements of digital control systems considering fragility, such as round-off and numerical quantization errors. Our approaches were also evaluated over a set of standard control-system benchmarks extracted from the control literature. Experimental results show that DSVerifier can check settling-time and overshoot in control systems suffering from FWL effects, while other existing approaches routinely ignore those issues.

Keywords Formal verification · Digital control systems · Finite word length · Controller fragility

.

T. Cavalcante
Federal University of Amazonas, Brazil E-mail: thiagocavalcante@ufam.edu.br

I. Bessa
Federal University of Amazonas, Brazil E-mail: iurybessa@ufam.edu.br

E. Filho
TP Vision, Brazil E-mail: eddie.filho@tpv-tech.com

L. Cordeiro
University of Manchester, UK E-mail: lucas.cordeiro@manchester.ac.uk

1 Introduction

A digital control system consists of sensors, controlled systems, control algorithms, and actuators, which together seek to maintain the behavior of a plant's (controlled system) variables under control, *i.e.*, they ensure the desired transient- and steady-state responses [10]. The development of digital controllers is a crucial task in control engineering since they are routinely used for many different applications, which range from industrial plants to smart cities.

Digital control theory aims at preserving some properties based on discrete-time models, *e.g.*, stability and robustness [7], which are necessary for the correct operation of real plants through a digital controller. Controlling continuous systems using digital controllers raises problems typical of hybrid systems. Besides, round-off and quantization of measurements and coefficients in digital control systems, due to finite word length (FWL) implementations, can lead to overflow, limit-cycle oscillation, and poles and zeros sensitivity, which might cause system instability and performance degradation [18,2]. Indeed, that kind of sensitivity is called fragility [20] and its investigation is worth to ensure compliance with functional requirements, in real-world applications. In literature, some related studies consider fragility on control system implementations, *e.g.*, Istepanian and Whidborne [18] reported some of the main results regarding fragility analysis, in digital control systems.

In the past years, some initiatives using formal verification applied to dynamic control systems have been developed. For instance, Bessa *et al.* [2,4] presented a verification method to determine uncertain linear-system stability regarding digital controllers, taking into account implementation aspects. Wang *et al.* [31] extended traditional verification techniques for digital controller implementation, with the goal of addressing validation of robustness at both model and code level. Chaves *et al.* [3] explored a behavior-modeling approach and developed models for digital controllers implemented with fixed-point arithmetic, where effects, such as overflow and limit-cycle oscillations, were checked, which resulted in the correct operation or at least guidelines for system redesign.

Yordanov *et al.* [32] considered small perturbations in inputs of a dynamic system to synthesize a feedback control strategy for a discrete-time piecewise affine (PWA) system from a specification given as a linear temporal logic (LTL) formula over an arbitrary set of linear predicates over system's state variables. Lahijanian *et al.* [22] proposed model checking algorithms based on probabilistic computation tree logic. The authors check the state reachability and safety of discrete-time stochastic systems. Also, Sadraddini and Belta [27] used signal temporal logic (STL) formulas to specify properties on discrete-time positive monotone systems and then synthesize robust model predictive controllers, while guaranteeing that STL specifications are met. Nilsson *et al.* [24] applied formal methods to adaptive cruise control and performed the synthesis of controllers that are correct-by-construction.

Step-response parameters, *e.g.*, required settling-time and overshoot specifications [10], are commonly used for specifying control systems, in order to indicate that closed-loop entities are safe, concerning those performance specifications. Nonetheless, there exist a few related studies that tackle performance parameters with formal methods. In that sense, Gross *et al.* presented an approach for formalizing common requirements for spacecraft attitude-control systems, where settling-times are checked through hypothesis testing [13]. Additionally, Jin *et al.* proposed a framework based on formal methods, to capture requirements from closed-loop models of industrial-scale control systems, including settling-time and overshoot [19]. Indeed, given that disturbances or changes in inputs may lead to transient oscillations in regulated outputs, the latter must be in the settling-time region, *i.e.*, an area specifying the tolerated deviations from a given reference, within a specified settling-time, and overshoots are lower than a maximum required value [19]. It is worth mentioning that FWL effects have a significant impact on system behavior [18]; however, to the best of our knowledge, there exists no application of formal verification techniques to check non-fragile satisfaction of step-response specifications.

Thus, this work presents a formal technique for settling-time and overshoot verification, in digital control-systems and regarding realizations in fixed-point representations, which was implemented in a model checker named as DSVerifier. Indeed, the latter has been evolved and is now capable of checking many properties in digital systems, including digital filters, digital controllers, and closed-loop systems [6]. In summary, simulations with restricted numerical formats for a given system are run, while considering system-design requirements as properties to be verified by a formal method.

FWL effects and their influence on digital control-system stability were already investigated by Bessa *et al.* [2], while the present work tackles controller-fragility consequences regarding performance. Moreover, stability is not the only issue to be considered, given that performance aspects are also important. In fact, control systems are now being employed in many different areas, such as aerial vehicles [13] and biomedical engineering [16,30], whose behavior may be compromised by implementation issues. For instance, long settling-times have the potential to cause loss of stability or even result in structural damages in UAVs, while overshoot events may result in unexpectedly large amounts of medicament being administrated to patients, which must be avoided during design phases.

Contributions. This paper makes the following original contributions:

- We propose a novel approach for automatic verification of performance specifications (settling-time and overshoot) of digital linear time-invariant systems while considering system fragility in the fixed-point representation of state-feedback controllers. In particular, we use invariants [23] to check whether given settling-time specifications are met;
- Experimental results show that our approach can efficiently check settling-time and overshoot specifications, in control applications under FWL effects.

Outline. Section 2 presents related studies. Section 3, in turn, describes fundamental concepts about digital controllers, along with implementation aspects. In Section 4, the settling-time invariant for formal verification is presented. Section 5 is concerned with verifying non-fragile settling-time requirements, while section 6 covers the verification of non-fragile maximum overshoot requirements. Section 7 tackles the performed verification experiments and discusses the obtained results. Finally, Section 9 concludes this work and proposes future research topics.

2 Related Work

Currently, some studies address problems related to control-system verification, whether in continuous or discrete-time and also considering some requirements, such as stability; however, the focus of our work lies on discrete-time systems. Also, most studies deal with verification of different control-system requirements, while the present research tackles only verification of performance requirements, which promptly indicates its importance and novelty.

2.1 Verification of Closed-loop Control Systems

Bessa *et al.* [2] presented a verification methodology to formally determine uncertain stability of digital controllers, w.r.t. implementation aspects. Specifically, that verification approach was implemented in DSVerifier [17], which is a verification tool that uses bounded model checking (BMC) based on satisfiability modulo theories (SMT) to verify digital control systems while considering uncertainty, plants, and finite word length (FWL) effects on digital-controller implementations. As a consequence, it indeed checks the robust "non-fragile" stability of a given closed-loop system. That proposed verification methodology was evaluated with non-fragile control examples found in the corresponding literature, which showed it could predict fragility problems in robust controllers, regarding their stability and considering FWL effects. Nonetheless, that study concerns only stability verification, while performance aspects, which are of utmost importance when designing controllers, were left out.

Wang *et al.* [31] presented an approach to verify control-system robustness, at coding and modeling levels, which is based on invariant computation in discrete system dynamics. By using semi-definite programming (SDP) solvers, a Lyapunov-based function is synthesized, thereby capturing vector margins of closed-loop linear systems. Such a numerical invariant expressed through system state variables is compatible with code analysis and allows its validation in code artifacts. That automatic analysis extends verification techniques focused on controller implementations, thus addressing validation of robustness in both model and code levels. It was implemented in a tool that analyzes discrete single-input single-output (SISO) systems and generates super-approximations of phase and gain margins. Once again, this study employs verification through

a code-level approach, while considering effects caused by fixed-point arithmetic; however, it does not take into account performance aspects in controllers.

Gross *et al.* [13] proposed an approach to formalize common spacecraft attitude control system requirements, such as actuator limits, display error, reachability, deviation, settling time, rise time, and overshoot. Those authors used formal methods, more specifically model verification and hypothesis testing, to verify the performance requirements of spacecraft attitude control systems. That study evaluates such systems and verifies whether they meet requirements over the continuous-time domain. Although the initial intention was to determine whether attitude control designs for spacecraft met system requirements, discrete-time systems were not tackled. In particular, implementation aspects were not considered, and the resulting non-linear motion equations proved to be very complicated for solvers since only some aspects were analyzed; however, digital-controller performance aspects were considered, such as pointing accuracy, pointing range, slew rate, overshoot, and settling time. Indeed, although even settling-time and overshoot were considered, FWL effects were not taken into account, and the approach presented here is more generic.

Guéguen and Zaytoon [14] presented a basic overview regarding verification of hybrid systems and stated that, in such a context, the available approaches traditionally verify three kinds of properties: safety ones, which express unauthorized configurations to be avoided during system evolution; liveliness ones, which describe the probability or likelihood of certain necessary system evolutions; and timeliness ones, which express limits regarding temporal distance of certain characteristic events or evolutions. One may notably notice that they do not consider FWL effects or tackle implementation aspects, as done in the present article and constitute the core of the methodology proposed here.

Chaves *et al.* [5] describe a verification procedure for digital systems with uncertainties, based on software model checking and satisfiability modulo theories, which can check robust stability of closed-loop control systems concerning FWL effects. In particular, they describe verification algorithms to check for limit-cycle oscillations (LCOs), output quantization error, and robust non-fragile stability on common closed-loop associations of digital control systems (*i.e.*, series and feedback). In particular, their work takes into account FWL effects, which is great, given its importance when such systems are implemented in the real world; however, step-response performance requirements of digital control systems are not verified. In that sense, there is some complementarity between that work and the present one, which has already inspired future studies.

Araiza-Illan *et al.* [1] introduce a formal verification methodology for high-level properties of control systems, such as stability, feedback gain, and robustness, with theorem proving, using the Why3 tool [9]. The evaluated systems are represented as Simulink® models and three main validation steps were suggested: definition properties of interest over the signals in a model; automatic translation into Why3; and automatic property verification in Why3. Indeed, that approach is interesting for analyzing control systems; however, it does not

take into account FWL effects and performance state-feedback control-system properties.

As one may notice, the literature elements gathered here show the importance of the proposed methodology, as it fills in a gap regarding requirements and restricted implementations. Besides, the approach developed here has the potential to devise a new class of controller verifiers, which can validate implementations regarding design-requirement evaluation over final implementations, still in development phases.

2.2 Synthesis of Digital Control Systems

Yordanov *et al.* [32] presented a computational framework for automatic synthesis of a feedback control strategy for discrete-time systems; more precisely, piecewise affine (PWA) systems, which for short are defined by partitioning extended state-input spaces into polyhedral regions and associating each one with a different affine state update equation [8], from a given specification as a linear temporal logic (LTL) over an arbitrary set of linear predicates in system state variables. Such an approach consists of defining appropriate partitions for state and input spaces and then constructing a finite abstraction of the system, in the form of a control transition system. Then, by taking advantage of ideas and verification techniques of LTL models and Rabin games, the respective authors developed an algorithm to generate a control strategy for finite abstraction. The resulting approach ensures that a control strategy, generated for a finite control system, can be easily transformed into a control strategy for the first PWA system. Although proven to be correct, the overall solution is conservative and computationally expensive. Additionally, while succeeding in synthesizing controllers, it fails to consider relevant aspects in digital-controller designs, such as performance and effects of controller fragility, when implemented in microprocessors.

Sadraddini and Belta [26] developed a method to control discrete-time systems with constant parameters, which are initially unknown from LTL specifications. Those authors used notions of parametric and adaptive transition systems (non-deterministic) and formal methods tools to compute adaptive control strategies for finite systems. Although traditional adaptive-control strategies are not employed, an approach that is corrected by construction is used, which does not require a reference model and can handle a much more comprehensive range of systems and specifications. However, it does not take into account aspects of system performance nor considers step responses, as well as leaves aside implementation aspects, such as FWL effects. As the majority of applications of formal methods, their results suffer from high computational complexity; as discussed in this particular study, the number of states in resulting adaptive transition systems (ATSs) can be vast. Finally, the construction of finite quotients for infinite systems is computationally challenging.

2.3 Optimization of Controllers

Hassani and Lee [15] presented a generic multi-objective design paradigm that uses quantum particle swarm optimization (QPSO) to decide the optimal configuration of an LQR controller for a given problem, considering a set of competing objectives. There exist three main contributions introduced in that study: (1) the standard QPSO algorithm is reinforced with an informed initialization scheme based on simulated annealing and Gaussian neighborhood selection, (2) it is also augmented with a local search strategy that integrates the advantages of the memetic algorithm in conventional QPSO, and, finally, (3) it also introduces an aggregate dynamic weighting criterion that dynamically combines soft and hard constraints with control objectives, to provide a set of optimal Pareto solutions, which allows it to choose a target solution based on practical preferences. The mentioned study is very promising since it verifies some performance specifications, such as settling time and overshoot, in control systems. Apart from that, the entire analysis considers controllers in continuous-time. As it does not work directly with digital control systems, this approach does not take into account its implementation aspects in microprocessors. It is then unable to generate controllers that, when implemented in those platforms, behave as designed, since they may suffer from FWL effects. The present paper tackles some aspects that many of those studies already address; however, it also covers all of these aspects at the same time, such as analysis in discrete-time systems, performance requirements of control systems (settling time and overshoot), and implementation aspects (FWL effects).

One may also notice that the problem of digital-controller verification is pertinent since this process is of paramount importance in design phases. Verification of digital controllers is regarded as a complex problem and it is also difficult to be understood because many studies try to solve it in different ways and take into account different factors. The main advantages of the proposed methodology, when comparing with the other schemes presented here, can be listed as follows:

- It works with discrete-time control systems, which is effectively used when implementing on an embedded microprocessor platform;
- It verifies controllers and considers performance requirements (settling time and overshoot);
- Implementation aspects are considered, since FWL effects, for example, can lead to faults.

Table 1 present a brief comparison between the proposed method and the selected related studies. The performance-requirement support described in Table 1 considers settling time and overshoot. One may notice that, regarding the studies presented here, no one covers discrete-time systems. Besides, there is a lack of studies regarding the verification of performance requirements. Finally, another gap is related to considering aspects of controller implementations, more specifically, FWL effects, because a few of them indeed deal with such an issue.

Table 1: Related work comparison

Related works	**Discrete-time system support**	**Performance requirement support**	**Consider FWL effects**
Bessa et al. (2017)	X		X
Wang et al. (2016)	X		
Yordanov et al. (2012)	X		
Sadraddini and Belta (2017)	X		
Gross et al. (2017)		X	
Guéguen and Zaytoon (2004)	X		
Chaves *et al.* (2019)	X		X
Araiza-Illan *et al.* (2014)	X		
Hassani and Lee (2016)		X	
The proposed work	X	X	X

3 Preliminaries

3.1 Digital Dynamic Systems

Let Ω be an n-th order single-input single-output (SISO) linear time-invariant (LTI) digital dynamic state-feedback control system represented as

$$\Omega : \begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \\ u(k) = r(k) - Kx(k) \end{cases}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, $C \in \mathbb{R}^{1\times n}$, and $D \in \mathbb{R}$ are the state-space realization matrices (A, B, C, D) of Ω, $K \in \mathbb{R}^{1\times n}$ is a digital state-feedback controller, $u(k) \in \mathbb{R}$ is a control signal, $y(k) \in \mathbb{R}$ is a system's output signal, $r(k) \in \mathbb{R}$ is a control system's reference signal, and $x(k) \in \mathbb{R}^n$ is a state signal vector. Ω is an open-loop system, if there exists no feedback signal, *i.e.*, $K = \mathbf{0}$. The spectrum $\mathcal{S} = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ of Ω is the set of its eigenvalues [7], *i.e.,* λ_i, for $i = 1, ..., n$, are the roots of Ω's characteristic equation

$$\det(\bar{A}\lambda - I) = 0, \tag{2}$$

where I is the identity matrix and $\bar{A} = A - BK$. Fig. 1 illustrates the step response of a discrete system, where L_{upp} and L_{low} define the settling time region $\mathit{\Pi}$, and y_{ss}, M_{p}, k_{s} and k_{r} (cf. Section 5) are its steady-state value, maximum overshoot, settling time, *i.e.*, the time required for a signal to remain within $p\%$ of its final value, and the reach time, *i.e.*, the sample where the system's response reaches the settling-time region for the first time, respectively.

In addition, the output of a discrete system given by Franklin et al. [10] as

$$y(k) = CA^k x(0) + \sum_{m=0}^{k-1} (CA^{k-m-1}Bu(m)) + Du(k), \tag{3}$$

which can be rewritten as $y(k) = y_{\text{s}}(k) + y_{\text{h}}(k)$, where y_{s} and y_{h} are its particular and homogeneous solutions, respectively, and the former is computed when

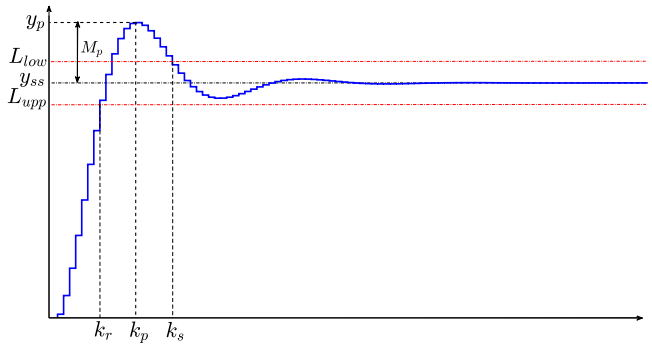


Figure 1: Step response of a discrete-time system.

time tends to infinite. If a system is stable, its exponential becomes very small at infinite and we only have what we call y_{ss}. As a consequence, it coincides with a particular solution, if the input is a unit step, *i.e.*, $u(k) = 1, \forall k \geq 0$. Finally, its solution is [10]

$$y(k) = y_{\text{ss}} + \sum_{i=1}^{n_i} \lambda_i^k \sum_{j=1}^{m_j} C_{ij} k^{j-1} \ , \tag{4}$$

where n_i is the number of non repeated poles, m_j is the multiplicity of the j-th pole, and C_{ij} is a constant for each member of the summation. Nonetheless, in Eq. (4), we have made a generalization, thus including eigenvalues with multiplicity greater than 1. Finally, y_{ss} is described as [10]

$$y_{\text{ss}} = C(I - A)^{-1}B + D. \tag{5}$$

We also want to find an invariant for settling-time verification. In particular, we aim to find a mapping ϕ of a given collection M, which consists of mathematical objects endowed with a fixed equivalence relation ρ, into another collection N of mathematical objects that is constant on the equivalence classes of M, concerning ρ (more precisely, that is an invariant of the equivalence relation ρ on M). If X is an object in M, then one can often say that $\phi(M)$ is an invariant of X [28].

3.2 FWL Effects in Digital State-Feedback Control Systems

We typically design control systems with high-precision poles, zeros, and variables (*e.g.*, double-precision floating-point arithmetic). However, when implemented, system parameters and signal variables are often represented with limited word-length (*e.g.*, fixed- or floating-point arithmetic), through hardware registers, which gives rise to FWL effects due to truncation and round-off errors, in control software [18].

The static state-feedback controller matrix K (see Eq. 1) of a digital system is heavily affected by FWL effects, which can compromise some system properties and must be handled during design phases. FWL effects on coefficients are described as

$$\mathcal{FWL}_{\langle I,F\rangle}[\cdot] : \mathbb{R}^{m\times n} \rightarrow \mathbb{R}_Q^{m\times n} \ , \tag{6}$$

where $\mathbb{R}_Q^{m\times n}$ is the discrete set of matrices $m \times n$ composed by elements of $\mathbb{R}^{m\times n}$, which can be represented in the fixed-point format $\langle I, F\rangle$, *i.e.*, I and F are the number of bits of integer and fractional parts, respectively.

As only K is digital and implemented with fixed-point arithmetic, whose coefficients are subject to FWL effects, a different n-th order digital state-space dynamic system with state feedback controller Ω_{FWL} then arises, whose control signal is $u(k) = r(k) - K_{\text{FWL}}x(k)$, where the matrix $K_{FWL} := \mathcal{FWL}_{\langle I;F\rangle}[K]$ is the controller matrix suffering from FWL effects. As a consequence, such FWL effects can influence step-response parameters of a system, which may then fail to meet design specifications.

3.3 Verifying Settling-Time and Maximum Overshoot

DSVerifier is a digital system verification tool that uses the Efficient SMT-Based Context-Bounded Model Checker (ESBMC) [11] as its main verification engine. It receives a digital system specification, computes the maximum and minimum representable numbers for a chosen FWL format, and then, during verification, checks whether all required parameters were correctly provided. DSVerifier uses BMC based on SMT and also implements an efficient k-induction proof rule [11]. Currently, DSVerifier supports stability, limit cycle, quantization error, settling-time, and overshoot, for closed-loop systems suffering from FWL effects, and, also, overflow and minimum phase, for open-loop ones [2]. The proposed method for verifying settling-times was implemented in DSVerifier, as illustrated in Fig. 2. The settling time t_s is the amount of time required for a signal to remain within 2% of its final value, for all future times, which is also sometimes defined as reaching 1% or 5% of that [10]

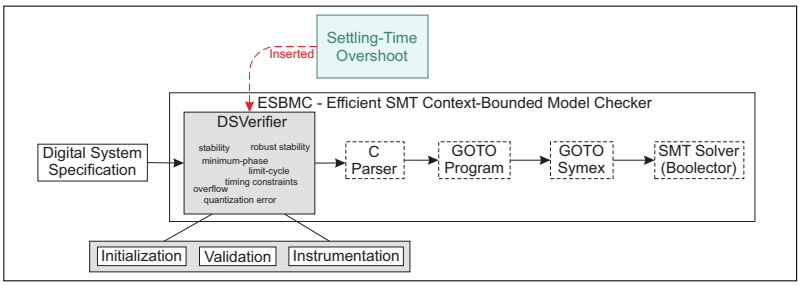


Figure 2: Settling-time and overshoot verification procedure implemented in DSVerifier.

On the one hand, we must use the following command-line, if DSVerifier is configured for checking settling-time: "`dsverifier specsFile.ss --property SETTLING_TIME`". In addition, a system and its required settling-time must be specified in file `specsFile.ss`, as shown in Fig. 3, where `tsr` is the required settling-time to be verified, `ts` is the adopted sampling time, and `p` is the settling-time region percentage. On the other hand, if we verify overshoot, the command-line would change to "`dsverifier specsFile.ss --property OVERSHOOT`", where we only need to define, in the specification file `specsFile.ss`, the parameter `POr`, which is the required percentage overshoot.

```
implementation ⟨16, 16⟩
range [1,1]
states = 4;
inputs = 1;
outputs = 1;
A = [-0.5,0.6,0.0,0.0;-0.6,-0.5,0.0,0.0;
B = [0.0;0.0;2.5;1.0]
C = [0.0,2.6,0.5,1.2]
D = [0.0]
x0 = [0.0;0.0;0.0;0.0]
inputs = [1.0]
tsr = 30.0;
POr = 10
ts = 0.5;
p = 5.0;
K = [0.0,12.6,4.5,1.2]
```

Figure 3: A digital-system specification for DSVerifier.

4 Settling-Time Invariant for Formal Verification

In this study, we aim to check whether a given digital control system meets a required settling-time and, as a consequence, we need to model its behavior. In that sense, let $S_1 = \{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_n|\}$ be the set of absolute values of eigenvalues of $\bar{A}$ in system Ω and $\overline{\lambda}$ is the spectral radius of $\bar{A}$, *i.e.*, $\overline{\lambda} = \max(S_1)$.

We can now define a heuristic function, which was developed in the present work and represents the output response of the system $\bar{\Omega}$ (heuristic system), as

$$\overline{y}(k) = y_{\text{ss}} + \overline{c}\overline{\lambda}^{k} \ , \tag{7}$$

where $\overline{c}$ is a constant that makes $\overline{y}(k)$ enter the settling-time region. Indeed, the heuristic function above uses the slowest eigenvalue $\overline{\lambda}$ of $\bar{A}$ (largest eigenvalue, in terms of absolute value), which is called spectral radius of $\bar{A}$, thus ensuring that $\bar{\Omega}$ always reaches the settling-time region after Ω.

It is worth noticing that Eq. (7) is similar to the response of a 1^{st} order system, where its particular solution is the constant y_{ss} and its homogeneous solution is the exponential with $\overline{\lambda}$ as base, in a power of sample k, as stated by Franklin, Powell, and Workman [10]. Therefore, we can say that this function

is an approximation of the original system by a 1st order one, as we will show in our experimental results described in Section 7.

Equation (7) holds by definition, as we propose a signal with that structure (see Eq. (4)), and, given that we use the spectral radius of A as exponential base, the behavior of that response ($\bar{y}(k)$) is always slower than $y(k)$, which is the base for settling-time analysis. Also, Eq. (7) is an exponential that converges to y_{ss}. As a result, we would be able to compute the instant $\hat{k}$, where a given response enters a settling-time region. $\hat{k}$ is a worst-case estimation and, if it is lower than a required settling-time, the verified system will be able to meet a settling-time requirement. As a result, we can use $\hat{k}$ as an invariant value, in this work, to help verify settling-time.

In summary, we can find a heuristic function based on the largest eigenvalue of Ω and then check the moment it enters a settling-time region because if that happens before the required settling-time, that is also true for Ω. To perform settling-time verification, we indicate a Π region employing a percentage p, usually 2% or 5%, so its upper (L_{upp}) and lower (L_{low}) limits are described as

$$L_{\text{upp}} = \left(1 + \frac{p}{100}\right) y_{\text{ss}} \tag{8}$$

and

$$L_{\text{low}} = \left(1 - \frac{p}{100}\right) y_{\text{ss}}, \tag{9}$$

respectively. The instant $\hat{k}$ when a system enters its settling-time region can be found by simply making (7) equal to (8), due to the form of the curve in (7), which results in

$$\hat{k} = \lceil \log_{\bar{\lambda}} \left(\frac{p}{100\bar{c}} y_{\text{ss}} \right) \rceil. \tag{10}$$

By approximating Ω (4) with $\bar{\Omega}$ (7), we can apply a point of Ω to $\bar{\Omega}$, in this case, the point where the maximum peak is located (k_p,y_{p}), in order to obtain

$$\bar{c} = \frac{y_{\text{p}} - y_{\text{ss}}}{\bar{\lambda}^{k_{\text{p}}}}. \tag{11}$$

The estimation of (k_p,y_{p}) will be further discussed along this work, specifically in Algorithm 3.

5 Verifying Non-fragile Settling-Time Requirements

Our methodology, which is illustrated in Fig. 4, provides settling-time verification in open- and closed-loop systems; however, the former do not present controllers and, consequently, FWL effects are not taken into account for them. Its first step consists in obtaining the necessary input parameters, *i.e.*, state-space matrices A, B, C, and D and a system's input u, which is followed by the

digital controller design, if a closed-loop system is considered (`Step 2`). Then, an FWL implementation is chosen, when in closed-loop (`Step 3`), which is followed by definition of required settling-time t_{sr}, percentage p of settling-time region, and sampling time T_{s} (`Step 4`). This way, we have the specification file (`SpecsFile.ss`) and we first verify system stability. After that, steps `A` to `E` are executed (if a resulting system is stable), *i.e.*, computation of FWL controller, if closed-loop is chosen (`Step A`), and steady-state value (y_{ss}), as described in Eq. (5) (`Step B`), estimation of the largest peak-values of the system's output (y_{p}) and sample k_{p} corresponding to y_{p} (`Step C`), and, finally, computation of $\overline{\lambda}$ (`Step D`), $\overline{c}$ and $\hat{k}$ (`Step E`).

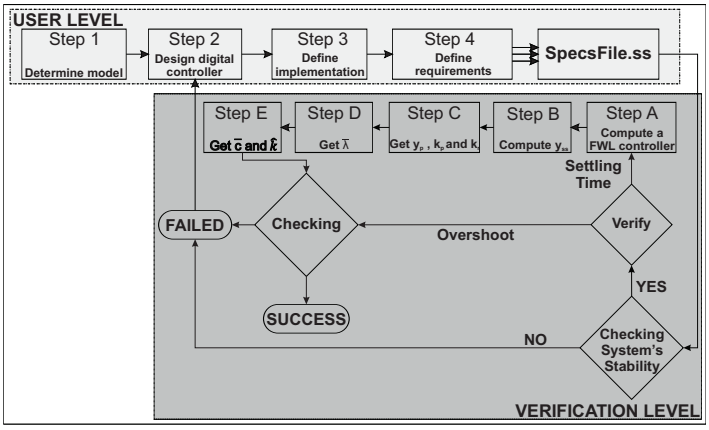


Figure 4: The proposed verification method for settling-time and overshoot.

The proposed procedure for verifying settling-time, which is described in Algorithm 1, consists of first checking if all system eigenvalues are real or y_{p} is in the settling-time region and, if this is true, $\hat{k}$ gets k_{r}, the latter being computed with Algorithm 2. Then, if k_{sr} is lower than $\hat{k}$, the verification fails and, if not, it is successful; we check if the estimated $\hat{k}$ is lower than the required settling-time $k_{\mathrm{sr}} = \frac{t_{\mathrm{sr}}}{T_{\mathrm{s}}}$, which can also be checked through $\hat{k} \times T_{\mathrm{s}} \leq t_{\mathrm{sr}}$ (as used in Algorithm 1). If that is the case, we can assure that the output signal is within Π (see Section 4) after t_{sr} and, as a consequence, the associated verification is already successful; otherwise, we still need to check if y remains within Π from k_{sr} to $\hat{k}$ and, if the latter is not true for any sample y, the resulting verification fails. Algorithm 3 describes a procedure for estimating y_{p} and k_{p}, which are used in `Step C` of our verification methodology (cf. Fig. 4). This algorithm was developed in a way to facilitate the verification of settling time in digital control systems by an invariant based on eigenvalue analysis. With the use of an invariant, we do not need to try to calculate the settling time directly, which would be computationally weak since we would calculate the output for each sample to find the settling time. However, we easily calculate $\hat{k}$

(10) to verify whether a given control system meets the required settling time by using that invariant in the verification process described in Algorithm 1.

Algorithm 1 Verify Settling Time

```
1: procedure checkSettlingTime()
2:     if y_p ⊂ Π then
3:         k̂ ← k_r
4:     else
5:         k ← t_sr / T_s
6:         k̂ ← ⌈log_λ̄ ( p / (100 c̄) y_ss )⌉ (Eq. 10)
7:         if k > k̂ then
8:             return Verification Successful
9:         else
10:            while k ≤ k̂ do
11:                if y(k) < L_low or y(k) > L_upp then
12:                    return Verification Failed
13:                increment k
14:            return Verification Successful
15:    if t_sr < k̂ × T_s then
16:        return Verification Failed
17:    else
18:        return Verification Successful
```

Algorithm 2 Procedure to find the instant k_{r}

```
1: procedure
2:     while y(k) ⊄ Π do
3:         increment k
4:     k_r ← k
```

To provide a deeper understanding of the proposed methodology, a simple example will be presented. One can suppose that we have a system

$$\Omega : \begin{cases} x(k+1) = \begin{bmatrix} 1.5 & 1.0 & 0.0 \\ 0.0 & 1.5 & 1.0 \\ 0.0 & 0.0 & 1.5 \end{bmatrix} x(k) + \begin{bmatrix} -0.4 \\ 2.5 \\ -0.8 \end{bmatrix} u(k) \\ y(k) = \begin{bmatrix} 0.0 & 2.6 & 0.0 \end{bmatrix} x(k) + [0] u(k) \\ u(k) = r(k) - Kx(k) \end{cases}, \tag{12}$$

where $r(k) = u_{-1}$ is the unit step, and we analyze it according to our approach. This way, `Step 1` is ready and `Step 2` is then considered, where a controller matrix K is not defined, due to open-loop operation. Consequently, an FWL implementation (`Step 3`) is not provided either. Next, the desired requirements are defined, that is, $t_{\mathrm{sr}} = 10s$, $p = 5$ and $T_{\mathrm{s}} = 0.5s$, in `Step 4`, through a specification file `SpecsFile.ss`.

Finally, given that open-loop operation was considered, this system is evaluated with "`dsverifier SpecsFile.ss --property SETTLING_TIME`". `Steps A` to `E` of the proposed methodology are automatically performed by DSVerifier, which considers an open-loop implementation, and, as a consequence, "

Algorithm 3 Estimation of y_{p} and k_{p}

```
1: procedure ESTIMATE_MAXPEAK_VALUE()
2:     y_p ← y(k)
3:     lastGrad ← 1
4:     loop
5:         if |y(k+1)| > |y(k)| then
6:             gradient = (gradient > 0)?(gradient + 1) : 1
7:             if |y(k+1)|! = |y(k)| then
8:                 firstGradSample = y(k+1)
9:                 firstGradSampleIdx = k+1
10:        else
11:            gradient = (gradient < 0)?(gradient − 1) : −1
12:        if (lastGrad > 0) and (gradient < 0) then
13:            if |firstGradSample| <= |k_p| then
14:                increment numBadPeaks
15:                if numBadPeaks > 2 then
16:                    exit(loop)
17:                else
18:                    y_p = firstGradSample
19:                    k_p = firstGradSampleIdx
20:            else if (grad > 10) and (|(y(k+1) − yss)/yss| < 0.001) then
21:                if |yss| > |y_p| then
22:                    y_p = yss
23:                    k_p = 0
24:                exit(loop)
25:        lastGrad = grad
26:        increment k
```

`Verification FAILED`" is returned, because the evaluated system is unstable, since its eigenvalues are greater than 1 ($\lambda = 1.5$, due to Eq. (2)). One may notice that the proposed method first checks whether a system is stable, between `Steps A` and `B`.

As the chosen system is unstable, we can go back to `Step 2` and design a controller to stabilize it and satisfy the required settling time, by using pole assignment [7]. In that case, we used $K = [-0.7351 \ -5.0045 \ -18.5371]$ in `SpecsFile.ss` and ran this experiment again, with "`--closed-loop`" and still without FWL effects (`--no-fwl`), which results in "`Verification SUCCESSFUL`", as can be seen in Fig. 5(a), where the step response does not leave the Π region between k_{sr} and $\hat{k}$.

Finally, this experiment can consider FWL effects. That is accomplished by using format $\langle 4, 4\rangle$, controller matrix $K_{FWL} = [-0.6875 \ -5.0 \ -18.5]$, and omitting `--no-fwl` (`Step A`), which results in "`Verification FAILED`", as shown in Fig. 5(b), where the system becomes unstable. It is worth noticing that gain K in `SpecsFile.ss` is not modified and K_{FWL} is internally computed by DSVerifier.

To evaluate other formats, $\langle 8, 8\rangle$ and $\langle 16, 16\rangle$ are considered, in `Step 2`. As a consequence, $K_{\mathrm{FWL}} = [-0.7344 \ -5.0039 \ -18.5352]$ with "`Verification FAILED`", because the step response left the settling-time region between k_{sr} and $\hat{k}$, and $K_{\mathrm{FWL}} = [-0.7351 \ -5.0045 \ -18.5370]$ with "`Verification SUCCESSFUL`", because the step response did not leave the settling-time region between k_{sr} and $\hat{k}$, are respectively obtained, as can be seen in Figures 5(c) and (d).

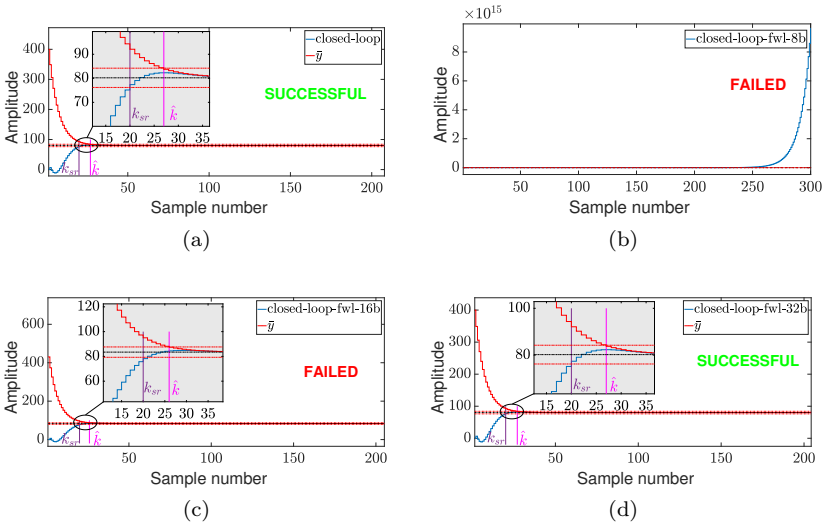


Figure 5: Settling-time verification for the example system, (a) without FWL effects and with formats (b) $\langle 4, 4\rangle$, (c) $\langle 8, 8\rangle$, and (d) $\langle 16, 16\rangle$.

As one may notice, the number of bits heavily influences settling-times in real implementations due to FWL effects. In addition, "`Verification FAILED`" was obtained for an intermediate format $\langle 8, 8\rangle$, while it was successful with $\langle 16, 16\rangle$. Indeed, given that the resulting settling-time is mostly dependent on the fastest natural response and its damping ratio [10], the interaction between them after quantization may cause such behavior, which further reinforces the use of the proposed methodology, during design phases.

Fig. 5(a) shows that $k_{\text{sr}} = t_{\text{sr}}/T_{\text{s}} = 20$ is less than the calculated $\hat{k} = 37$ and, as a consequence, for this required settling-time, the mentioned system is already in the settling time region, so that settling-time is doable in that case. Finally, the same reasoning is valid for Fig. 5(b), (c), and (d).

6 Verifying Non-fragile Maximum Overshoot Requirements

The maximum overshoot is the maximum transient value that exceeds y_{ss}, *i.e.*, $M_{\text{p}} = y_{\text{p}} - y_{\text{ss}}$. Then, the percentage overshoot is computed as follows

$$PO = 100 \times \frac{M_{\text{p}} = y_{\text{p}} - y_{\text{ss}}}{y_{\text{ss}}}, \tag{13}$$

where y_{p} can be obtained from Algorithm 3.

Fig. 4 illustrates the overshoot verification methodology step by step in DSVerifier. Firstly, we must determine the control system's model (`Step 1`) and design a controller, for closed-loop systems (`Step 2`). After that, we then

define a chosen FWL implementation (`Step 3`), and, in `Step 4`, the overshoot specification to be verified is informed. Next, `SpecsFile.ss` is created, which then enables the system's stability check and, finally, overshoot verification. The overshoot verification consists of checking whether the computed percentage overshoot (PO) is less than the required percentage overshoot (PO_{r}). If that is true, it returns "`Verification SUCCESSFUL`"; otherwise, "`Verification FAILED`" is output, as described in Algorithm 4.

Algorithm 4 Verify Overshoot

```
1: procedure CHECKOVERSHOOT(y_ss, PO_r)
2:     y_p ← estimate_maxPeak_value()
3:     PO ← (y_p − y_ss) / y_ss
4:     if PO > PO_r then
5:         return Verification Failed
6:     return Verification Successful
```

With the goal of deeper understanding regarding our overshoot verification approach, we can analyze the example illustrated in Eq. (12), by following the steps in Fig. 4. This way, `Step 1` is ready and `Step 2` is then considered, where a controller matrix K is not defined, due to open-loop operation. Consequently, an FWL implementation (`Step 3`) is not provided either. Next, the desired requirements are defined, such as $PO_{\mathrm{r}} = 9$, in `Step 4`, using a specification file `SpecsFile.ss`.

Given that open-loop was considered, this system is evaluated with the following command "`dsverifier SpecsFile.ss --property OVERSHOOT`" and, as a consequence, "`Verification FAILED`" is returned, because the evaluated system is unstable and, consequently, there exists no need to check overshoot. One may notice that the proposed method always checks whether a system is stable.

As the chosen system is unstable, we can go back to `Step 2` and design a controller aiming to stabilize it and satisfy the required percentage overshoot (PO_{r}), by using pole assignment [7]. In that case, we use the same controller matrix $K = [-0.7351 \ -5.0045 \ -18.5371]$ in `SpecsFile.ss`, as in the example explained in Section 5, since that controller was designed to satisfy both properties (settling-time and overshoot). Next, we run this experiment again, but now with option "`--closed-loop`" and still without FWL effects (`--no-fwl`), which results in `Verification SUCCESSFUL`, since DSVerifier returned $PO = 2.6228$ that is less than $PO_{\mathrm{r}} = 9$.

Finally, this experiment can consider FWL effects. That is accomplished by using format $\langle 4, 4\rangle$, controller matrix $K_{\mathrm{FWL}} = [-0.6875 \ -5.0 \ -18.5]$, and omitting `--no-fwl` (`Step A`), which results in "`Verification FAILED`", due to the fact that the system is still unstable.

To evaluate other formats, $\langle 8, 8\rangle$ and $\langle 16, 16\rangle$ were considered, in `Step 2`. As a consequence, $\langle 8, 8\rangle$ ($K_{\mathrm{FWL}} = [-0.7344 \ -5.0039 \ -18.5352]$) resulted in "`Verification SUCCESSFUL`", given that DSVerifier returned $PO = 1.6153$, which is lower than $PO_{\mathrm{r}} = 9$, and $\langle 16, 16\rangle$ ($K_{\mathrm{FWL}} = [-0.7351 \ -5.0045 \ -$

18.5370]) resulted in "`Verification SUCCESSFUL`", because our methodology provided $PO = 2.6253$, which is again lower than the required value.

One may notice that the number of bits heavily influences a controller design and, consequently, all properties tackled during design phases are also modified, due to changes in coefficients and computation results. In particular, settling-time and overshoot were affected in different ways, which means that if a property fails, the others may not, as can be seen for format $\langle 8, 8 \rangle$.

7 Experimental Evaluation

7.1 Description of the Benchmarks

In this section, the control-system benchmarks used to evaluate the proposed approach are described. In particular, the benchmarks used in this work are available at `https://goo.gl/fSwVVb` and provide information about state-space matrices A, B, and C, in Jordan form [7], controller matrix K, required settling-time t_{sr}, and required percentage overshoot PO_{r}. All benchmarks present state-space matrix $D = [0]$ and were chosen because they tackle different scenarios regarding eigenvalues of A. There exist systems of 2^{nd}, 3^{rd}, 4^{th} and 5^{th} order, with the same, different, real, and complex eigenvalues (cf. Section 3.1). Additionally, two of them are real-world examples: the 15th one represents an actual DC motor physical plant and the 16th one describes a magnetic levitation physical plant. Also, benchmark 17 is a 17^{th} order system based on the one provided by Tran, Nguyen, and Johnson [29], which was discretized. The benchmarks used in our study do not contain multiple-input multiple-output (MIMO) systems since the proposed methodology can check only SISO ones.

There exist 17 experiments in an open-loop configuration, 17 in closed-loop without FWL effects, and 51 in closed-loop with FWL effects, for 8, 16, and 32 bits, which amounts to 255 verification tasks for each property, that is, settling-time and overshoot.

Availability of Data and Tools. All tools, benchmarks, and results regarding evaluations are available at `https://goo.gl/fSwVVb`.

7.2 Experimental Objectives and Setup

By using the state-space models presented in Section 7.1, our evaluation has the following experimental goal (EG): *does our verification approach produce results that are confirmed outside our model* **(soundness)**? All experiments with DSVerifier v2.0.3 were conducted on an otherwise idle Intel Core i7-2600 3.40 GHz processor, with 32 GB of RAM and running Linux Mint OS. No time limit has been set.

7.3 Numerical Results

Table 2 shows settling-time and overshoot verification results for the benchmarks described in Section 7.1. In particular, experiments were conducted in three different types of system: open-loop, closed-loop without FWL effects, and closed-loop with FWL effects, in digital controllers with sampling time $T_s = 0.5s$. Basically, those experiments checked whether the required parameters (t_{sr} and PO_r), which are provided in our benchmarks at `https://goo.gl/fSwVVb`, were met by a given control system, according to the methodology described in Sections 5 and 6. Table 2 also shows columns with verification results for settling-time (ST) and overshoot (OS), including FWL effects ($\langle 4,4\rangle$, $\langle 8,8\rangle$ and $\langle 16,16\rangle$).

Benchmark			**Settling-Time Verification**					**Overshoot Verification**				
#	$t_{sr}(s)$	PO_r	*OL*	*CL*	$\langle 4,4\rangle$	$\langle 8,8\rangle$	$\langle 16,16\rangle$	*OL*	*CL*	$\langle 4,4\rangle$	$\langle 8,8\rangle$	$\langle 16,16\rangle$
1	2.5	5	F	S	S	S	S	F	S	S	S	S
2	3.5	5	F	S	F	S	S	F	S	F	S	S
3	4.5	30	F	S	S	F	S	S	S	S	S	S
4	5.5	20	F	S	F	F	S	F	S	F	S	S
5	3.0	5	F	S	S	S	S	F	S	S	S	S
6	5.0	30	F	S	F	S	S	F	S	F	S	S
7	10.0	4	F	S	F	F	S	F	S	F	F	S
8	1.5	8	F	S	F	F	S	F	S	F	S	S
9	2.0	8	F	S	S	S	S	F	S	S	S	S
10	5.0	30	F	S	F	S	S	S	S	S	S	S
11	10.0	18	F	S	F	F	S	F	S	F	F	S
12	8.0	10	F	S	F	S	S	F	S	F	S	S
13	10.0	9	F	S	F	F	S	F	S	F	S	S
14	10.0	2	F	S	S	F	S	F	S	S	F	S
15	6.5	5	F	S	S	S	S	S	S	S	S	S
16	10.5	8	F	S	F	F	S	F	S	F	F	S
17	27.0	1	S	S	F	F	S	F	S	F	F	S
S = Success, F = Fail, OL = open-loop, CL = closed-loop without FWL verification												

Table 2: Settling-time and overshoot verification results considering open- and closed-loop, the latter being with and without FWL formats.

In the experiments in closed-loop configuration and without FWL effects, one can notice that all of them produced `VERIFICATION SUCCESSFUL`, as expected because we designed suitable controllers, to satisfy both properties. These results reflect the fact that the original systems under verification meet the required settling-time t_{sr}, since the designed digital controllers respect the chosen specifications and use high precision. At the same time, the same occurs for the required percentage overshoot PO_r. However, regarding experiments in closed-loop configuration with FWL effects, one may notice that the number of bits in an FWL format has great influence in results (*e.g.*, benchmarks 11, for $\langle 4,4\rangle$ and $\langle 8,8\rangle$, and 14, for $\langle 8,8\rangle$).

It is interesting to notice that, in benchmark 4, the verification procedure failed for closed-loop with FWL effects, when using format $\langle 4,4\rangle$ and for both properties (ST and OS), as we can see in Fig. 6(a). Indeed, that happened since

the resulting system became unstable. However, when we look at the results for FWL effects with a precision of 16 bits ($\langle 8, 8\rangle$), the verification procedure only failed for settling-time, while produced `VERIFICATION SUCCESSFUL` for overshoot (its maximum overshoot is lower than the required), as one can notice in Fig. 6(b). In addition, for format $\langle 16, 16\rangle$, the proposed methodology produced `VERIFICATION SUCCESSFUL` for both properties. Specifically, regarding settling-time verification, the system output y remained in the settling-time region between samples k_{sr} and $\hat{k}$ and, for overshoot verification, $PO = 2.483$, which is lower than $PO_{\text{r}} = 20$, as one can notice in Fig. 6(c).

For the benchmarks used in this work, looking at Table 2, we can separate those results into 8 groups: $\mathcal{G}1 = \{1, 5, 9 \text{ and } 15\}$, $\mathcal{G}2 = \{2, 6 \text{ and } 12\}$, $\mathcal{G}3 = \{3\}$, $\mathcal{G}4 = \{4, 8, 13\}$, $\mathcal{G}5 = \{7, 11, 16\}$, $\mathcal{G}6 = \{10\}$, $\mathcal{G}7 = \{14\}$ and $\mathcal{G}8 = \{17\}$. $\mathcal{G}1$ is composed of systems that were not sensible to FWL effects, regarding any of the verified properties (ST and OS). $\mathcal{G}2$, in turn, is the group where its systems failed only for $\langle 4, 4\rangle$ FWL format, for both properties (ST and OS). Indeed, they can be regarded as traditional cases, where increasing the number of bits in a chosen representation produces correct results. $\mathcal{G}3$ consists of a system that failed only for settling-time with $\langle 4, 4\rangle$ FWL format, which shows some independence in the way the verified properties are affected by FWL effects. In $\mathcal{G}4$, those systems failed for both properties using FWL format $\langle 4, 4\rangle$ and just for settling-time, when applying $\langle 8, 8\rangle$. As one may notice, that reinforces independence between properties and also suggests that it may be easier to meet overshoot requirements than settling-time ones when dealing with precision issues, even with a reasonable number of bits allocated for a representation. $\mathcal{G}5$ is composed by systems that fail for both properties in FWL formats $\langle 4, 4\rangle$ and $\langle 8, 8\rangle$, which is solved only by applying the maximum number of bits considered in the experiments performed here. $\mathcal{G}6$ includes only benchmark 10, which fails for settling-time verification using FWL format $\langle 4, 4\rangle$. Also, in $\mathcal{G}7$, the proposed verification fails for both properties with FWL format $\langle 8, 8\rangle$. The latter is very interesting, given that it had already been successful for format $\langle 4, 4\rangle$. Finally, in $\mathcal{G}8$, open-loop only failed for overshoot; as a consequence, a controller was designed to meet the requirements for both settling-time and overshoot. It does not fail for both requirements, with no FWL effects; however, when FWL effects come into place, it fails for formats $\langle 4, 4\rangle$ and $\langle 8, 8\rangle$, for both requirements. Indeed, it reveals that step-response parameters are not simply and only affected by the number of bits, but also depend on interactions among other properties also influenced by word precision.

As a general conclusion regarding the experiments performed here, we were able to notice a trend towards failing when implementing with a lower number of bits, as can be seen for experiments with benchmarks 2, 4, 6, 7, 8, 10, 11, 12, 13, 14 and 17, which leads to the use of as many bits as allowed by a given architecture. However, there exist also experiments where failures were obtained for an intermediate format, which happened with benchmarks 3 and 14. Indeed, as already mentioned in Section 5, resulting settling-times and percentage overshoot depend on the interaction among quantized parameters, such as natural frequencies and damping ratio, which may produce such unexpected

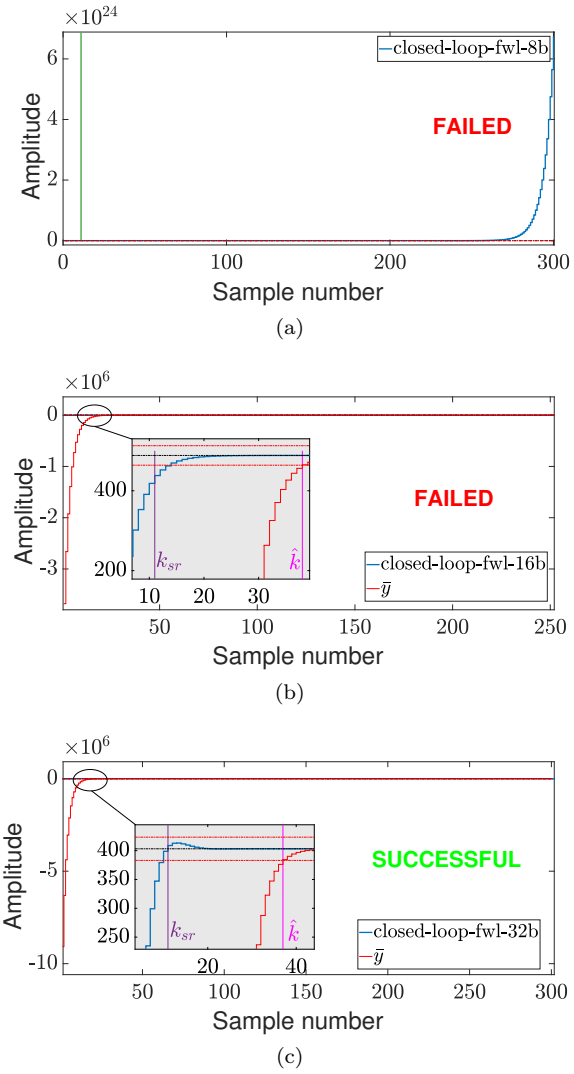


Figure 6: Settling-time verification for benchmark 4, in closed-loop with FWL formats (a) $\langle 4, 4\rangle$, (b) $\langle 8, 8\rangle$, and (c) $\langle 16, 16\rangle$.

results and claims for a verification methodology that takes it into account. Besides, no failure was noticed for the highest number of bits considered in the mentioned experiments, as one could expect, which further reinforces the intuition that a high number of bits is preferable.

There exists a last impressive result for settling-time verification: when a given system fails for this requirement, it does not mean it will fail for overshoot too, as we can see in Table 2 (groups $\mathcal{G}3$, $\mathcal{G}4$, and $\mathcal{G}6$). Indeed, given that they present some degree of decoupling, that is possible.

Finally, Table 3, in turn, shows how FWL effects affect benchmark 7's poles (eigenvalues) and, consequently, controller K. As the number of bits in the chosen FWL format is increased, the closer K_{FWL} gets to K, which also happens to the associated eigenvalues.

As one can see in Table 2, our methodology can verify systems implemented in a real platform (microcontrollers), by taking FWL effects into account in its controllers, to verify designed properties, which other existing approaches usually ignore. In summary, the presented results show that a system designed for respecting strictly defined properties, when tested with high precision, may fail, when FWL effects take place.

In summary, the results of this experimental assessment show that digital control systems may present fragile performance, concerning settling-time and overshoot. Although a greater number of bits often avoids this problem, our proposed formal settling-time and overshoot verification methodologies showed that, sometimes, an implementation with a lower number of bits does not violate those properties, which means that cheaper practical systems may be achieved.

Answering our main experimental question, we were indeed able to verify all results provided by our methodology implemented in DSVerifier. In particular, we have evaluated the same benchmarks used in this work in Matlab® and then graphically verified each behavior, as we can check online at `https://goo.gl/fSwVVb`. We performed this evaluation through resulting graphs and `m`-files.

	K	**Eigenvalues**
closed-loop	[14.8068 −5.2354 4.9214 −8.8323]	$0.8624 \pm 0.1155i$, 0.1496, 0.1353
closed-loop-fwl-8b	[14.7500 −5.1875 4.8750 −8.8125]	$0.2845 \pm 0.4363i$, 1.5730, −0.0421
closed-loop-fwl-16b	[14.8047 −5.2344 4.9180 −8.8320]	1.0530, 0.6038 0.2953, 0.0655
closed-loop-fwl-32b	[14.8068 −5.2354 4.9214 −8.8323]	$0.8624 \pm 0.1157i$, 0.1494, 0.1355

Table 3: Controller K and eigenvalues of benchmark 7, with FWL effects.

7.3.1 Case Study - Ball and Beam

An interesting application, with the potential to clarify some aspects of the proposed verification methods, is a ball-and-beam system, where a ball is allowed to roll along a beam, as illustrated in Figure 7. A lever arm is attached to one end of a beam, a servo gear to the other one, and, when the latter turns by an angle θ, the former changes the beam's angle by α, while gravity causes the ball to roll along with it. The model of this system is described as

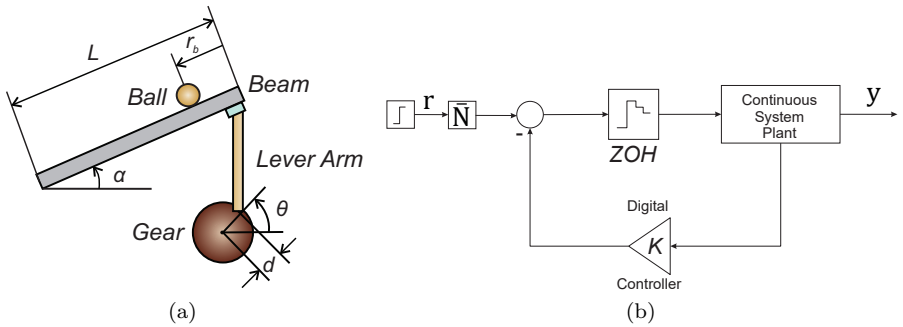
$$\Omega_{bb}\begin{cases}\dot{x} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & \frac{-mg}{\frac{J}{R^2}+m} & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix} x + \begin{bmatrix}0\\0\\0\\1\end{bmatrix} u\\ y = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} x\end{cases}, \tag{14}$$

where $x = \begin{bmatrix} r_b & \dot{r_b} & \alpha & \dot{\alpha}\end{bmatrix}^T$, $m = 0.11kg$, $R = 1.5cm$, $J = 9.99e^{-6}kg.cm^2$, and $r_b(cm)$ are the ball's mass, radius, moment of inertia, and position coordinate, respectively, $g = 980cm/s^2$ is the gravitational acceleration, and α is the beam's angle coordinate.

In this case study, our goal is to check if a derived digital control system behaves in the same way as its continuous-time counterpart. In particular, we will verify if the required settling-time t_{sr} satisfies the system given in (14), which must first be discretized, to be checked by DSVerifier. Besides, a discrete-time controller K must be designed, which is carried out by generating continuous-time system poles able to lead to t_{sr}. Finally, those are discretized with $z = e^{st}$ and used in the pole placement technique to create K. As a result, we obtain

$$\Omega_{bbd}\begin{cases}x(k+1) = \begin{bmatrix}1 & 0.1 & 3.5036 & 0.1168\\ 0 & 1 & 70.0714 & 3.5036\\ 0 & 0 & 1 & 0.1\\ 0 & 0 & 0 & 1\end{bmatrix} x + \begin{bmatrix}0.0029\\0.1168\\0.005\\0.1\end{bmatrix} u\\ y(k) = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} x\\ u(k) = r(k) - \begin{bmatrix}0.4616 & 0.2675 & 100.2593 & 14.2982\end{bmatrix} x\end{cases}$$

$x = \begin{bmatrix} r_b(k) & \dot{r_b}(k) & \alpha(k) & \dot{\alpha}(k)\end{bmatrix}^T \quad r(k) = 80u_{-1}(k).$



(a) (b)

Figure 7: Setup of ball and beam system.

In summary, a discrete-time static state-feedback controller with gain K is used for a continuous-time plant with zero-order holder (ZOH), as described by the block diagram illustrated in Figure 7, with $t_{sr} = 3$s. As a result, that system can now be checked by DSVerifier, whose results for closed-loop without FWL effects, with 8-, 16-, and 32-bit FWL formats, at different sampling times T_s, can be seen in Table 4. Finally, the mentioned system can be simulated to check the obtained results, which was done with $T_s = 0.1$s and Matlab®, as illustrated in Figure 7. Moreover, it is worth noticing that there is no stationary error, due to the gain for compensating the reference, which can be seen in the same figure. One may notice that the associated `m`-files can be found at `https://goo.gl/fSwVVb`. The obtained results are shown in Figure 8 and one may notice that they match what was reported by DSVerifier. Figures 8(a) and (d) show that $t_{sr} = 3$s is achievable for that system, *i.e.*, the output signal is restricted to the required settling-time region, as we can see in the zoomed stretch, if a format with 32 bits is used; however, shorter formats are unsuitable, as shown in Figures 8(b) and (c). Indeed, that was previously detected by DSVerifier, as one can see in Table 4.

T_s(s)	closed-loop	closed-loop-fwl-8b	closed-loop-fwl-16b	closed-loop-fwl-32b
0.1	S	F	F	S
0.15	S	F	F	S
0.2	S	F	F	S
0.25	S	F	S	S

Table 4: Use case results in DSVerifier, a ball-and-beam system.

We have used different sampling times, as reported in Table 4. One may notice further that, as long as we increase the sampling time, the chance to fail is smaller, as we observed in the simulation with closed-loop and FWL format $\langle 4, 4 \rangle$.

7.4 Threats to Validity

We have shown a favorable assessment of our method, over a set of digital-control systems (see online at `https://goo.gl/fSwVVb`), by considering stable and unstable eigenvalues in open- and closed-loop configurations, the latter being with and without FWL effects. Nonetheless, those benchmarks are limited within the scope of this paper and the proposed method's performance needs to be further assessed on a more extensive set of real-world systems.

Additionally, our approach for verification of settling-time and overshoot is only interesting for deterministic systems, from which we know all numerical values regarding controllers, in advance. This way, it is not prepared to work with systems with parametric uncertainties. Additionally, nonlinear systems are not addressed here. Our approaches were developed to help build LTI systems, which are indeed valid to get conclusions about the latter, for small perturbations.

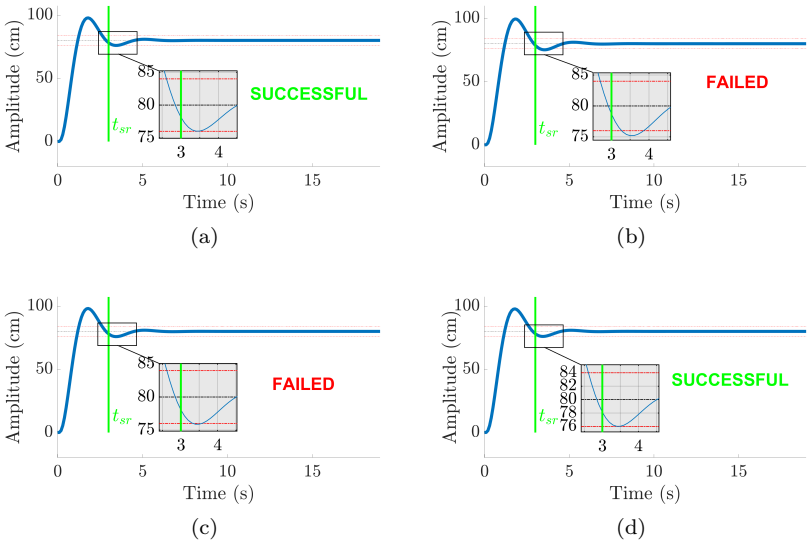


Figure 8: Analysis on the continuous-time plant at $T_s = 0.1$s, for closed-loop with no FWL (a) and FWL formats $\langle 4, 4\rangle$ (b), $\langle 8, 8\rangle$ (c), and $\langle 16, 16\rangle$ (d).

One may notice that our evaluation method focuses on discrete-time LTI systems, as can be seen throughout this text. As a consequence, our evaluation of discrete-time models is sound and related properties, such as superposition, hold as long as discretization is carried out properly. Hybrid systems are treated as discrete and the nonlinearities related to the interaction between continuous and discrete dynamics are not directly tackled: our focus is about the effects of digital implementations of dynamical controllers, in closed-loop performance. Our proposed method is not inherently linked to specific responses but checks outputs as systems properties instead, which consists of a more generic approach.

8 Brief Discussion Regarding Methodology Use, Contributions, Fragility and Parametric Errors

Currently, verification procedures for the design requirements of digital control systems do not take into account the influence of FWL effects on digital controllers. In practical scenarios, some developments present test phases, which usually tackle a system model with an arbitrary numerical format, to validate a future deployment. However, when a real implementation is carried out, tests on a target platform may review FWL effects and fragility issues, which then triggers a new phase for additional development or bug-fixing procedures. In that sense, the proposed work and other related studies available

in Section 2 devise a new paradigm for control-system design, which now includes a test phase during development and aims to minimize rework after implementing a given system in a real platform. Besides, the main obtained benefit is cost and time reduction in digital control-system development, due to minimization of additional development, bug-fixing, and even testing, given that the proposed methodology is fully automatic based on efficient model-checking procedures [12,21].

In particular, our settling time verification approach is different from traditional methods. In order to verify that a digital controller applied to a physical plant meets a specific required settling time, our methodology does not need to perform various successive iterations for computing the settling time of a system under consideration. Instead, it uses an invariant ($\hat{k}$) that is obtained through a heuristic function. This specific function is based on the spectral radius of a system, thus ensuring that it will enter the settling time region always after an original system's output response.

Here, we assumed that a discrete-time linear time-invariant (LTI) model of a system accurately represents its dynamics, *i.e.*, this study does not take into account the influence of parametric errors and variations (uncertainty). Indeed, its primary purpose is to investigate the effect of controller fragility, *i.e.*, variations on a controller's coefficients due to implementations restricted by fixed-point representations, in control system's performance specifications, *i.e.*, settling time and overshoot, and in compliance concerning step-response design requirements.

That being said, it is worth noticing that verification of robust fragility is also a very relevant topic, since parametric uncertainties may affect verification results and the fragility issue can modify robustness properties. Besides, such effects were already studied by Bessa *et al.* [2], for the problem of non-fragile robust stability of LTI systems. In that study, both parametric uncertainties and FWL effects were considered, when investigating the stability of closed-loop control systems with digital controllers implemented in fixed-point and represented through transfer functions; however, performance aspects were not considered. In the present manuscript, the problem of fragility for digital implementations is also tackled regarding state-space representations of LTI systems, which was not considered by Bessa *et al.* [2]. In particular, to the best of our knowledge, this article is the first work that investigates fragility regarding digital control system performance, in compliance with step-response design specifications.

Although a joint robustness and fragility investigation is not addressed here, given that it goes beyond the initial scope of this article, it should be investigated in further work. Such a result would enhance synthesis procedures and also verification steps of digital control systems, with the potential to produce extremely robust systems for practical applications, which could already be assured in development phases.

Finally, as one may notice, the proposed methodology is a novelty and, together with those other studies in Section 2, may constitute a toolbox for

a broader test phase regarding digital control-systems to anticipate problems related to restricted implementations and fragility.

9 Conclusions

New methods for verifying settling-time and overshoot requirements, in digital-control systems, are described and evaluated, by considering FWL effects in fixed-point digital controllers and taking into account the fragility problem concerning performance requirements. Prior studies proposed to verify digital-system properties, such as stability and limit-cycle, even considering FWL effects in implementations of digital controllers. However, the present work proposes performance-requirement (settling-time and maximum overshoot) verification algorithms implemented with a tool based on a formal method (DSVerifier).

The detailed experimental results (see Section 7.3) show that the proposed methods are efficient and effective when verifying fragility of closed-loop systems with or without FWL effects, in fixed-point digital controllers. Additionally, we have noticed that our methodology can be useful to check if a given required settling-time satisfies a given system, without explicitly calculating that, by using an invariant. Equation (7) is essential as an initial and fast check since it indicates an early settling-time region positioning. Also, if that does not hold, further verification is performed directly with output samples, as described in Algorithm 1. As a consequence, Equation (7) has the potential to reduce computation, by avoiding exhaustive exploration of system outputs.

For future work, we will introduce non-deterministic performance verification, to enable formal verification concerning parametric uncertainties. Additionally, given that we employ state-space representations, MIMO systems can already be tackled. More specifically, our methods are based on an analysis of the spectral radius of A so that it can be used with MIMO systems as well, but, at this moment, we did not prepare DSVerifier to support that kind of system. Besides, we will include robust and non-fragile performance verification and synthesis, thereby taking into account the influence of parametric errors and variations. Finally, further work will also tackle fragility investigation over specialized state-space realizations and inclusion of integrators into closed-loops to ensure step reference tracking [25].

References

1. Araiza-Illan, D., Eder, K., Richards, A.: Formal verification of control systems properties with theorem proving. In: 2014 UKACC International Conference on Control (CONTROL), pp. 244–249. IEEE (2014)
2. Bessa, I., Ismail, H., Palhares, R., Cordeiro, L., Chaves Filho, J.E.: Formal non-fragile stability verification of digital control systems with uncertainty. IEEE Trans. Comput. **66**(3), 545–552 (2017)
3. Chaves, L., Bessa, I.V., Ismail, H., dos Santos Frutuoso, A.B., Cordeiro, L., de Lima Filho, E.B.: Dsverifier-aided verification applied to attitude control software in unmanned

aerial vehicles. IEEE Transactions on Reliability **67**(4), 1420–1441 (2018). DOI 10.1109/TR.2018.2873260
4. Chaves, L.C., Bessa, I., Cordeiro, L.C., Kroening, D., de Lima Filho, E.B.: Verifying digital systems with MATLAB. In: Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 388–391. ACM (2017). DOI 10.1145/3092703.3098228
5. Chaves, L.C., Ismail, H.I., de Bessa, I.V., Cordeiro, L.C., de Lima Filho, E.B.: Verifying fragility in digital systems with uncertainties using dsverifier $v2.0$. Journal of Systems and Software **153**, 22–43 (2019). DOI 10.1016/j.jss.2019.03.015
6. Chaves, L.C., Ismail, H.I., Bessa, I.V., Cordeiro, L.C., de Lima Filho, E.B.: Verifying fragility in digital systems with uncertainties using dsverifier v2.0. Journal of Systems and Software **153**, 22–43 (2019). DOI 10.1016/j.jss.2019.03.015
7. Chen, C.T.: Linear system theory and design. Oxford University Press, Inc. (1995)
8. Christophersen, F.: Optimal control of constrained piecewise affine systems, vol. 359. Springer (2007)
9. Filliâtre, J.C., Paskevich, A.: Why3-where programs meet provers. In: European Symposium on Programming, pp. 125–128. Springer (2013)
10. Franklin, G.F., Powell, J.D., Workman, M.L.: Digital control of dynamic systems, vol. 3. Addison-wesley Menlo Park, CA (1998)
11. Gadelha, M.Y.R., Ismail, H.I., Cordeiro, L.C.: Handling loops in bounded model checking of C programs via k-induction. STTT **19**(1), 97–114 (2017). DOI 10.1007/s10009-015-0407-9
12. Gadelha, M.Y.R., Monteiro, F.R., Cordeiro, L.C., Nicole, D.A.: ESBMC v6.0: Verifying C programs using k-induction and invariant inference - (competition contribution). In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), *LNCS*, vol. 11429, pp. 209–213. Springer (2019)
13. Gross, K.H.: Formal specification and analysis approaches for spacecraft attitude control requirements. In: Aerospace Conference, pp. 1–11. IEEE (2017)
14. Guéguen, H., Zaytoon, J.: On the formal verification of hybrid systems. Control Engineering Practice **12**(10), 1253–1267 (2004)
15. Hassani, K., Lee, W.S.: Multi-objective design of state feedback controllers using reinforced quantum-behaved particle swarm optimization. Applied Soft Computing **41**, 66–76 (2016)
16. Ilyas, M., Khaqana, A., Iqbal, J., Riaz, R.A.: Regulation of hypnosis in propofol anesthesia administration based on non-linear control strategy. Brazilian Journal of Anesthesiology **67**(2), 122–130 (2017)
17. Ismail, H.I., Bessa, I.V., Cordeiro, L.C., de Lima Filho, E.B., Chaves Filho, J.E.: Dsverifier: A bounded model checking tool for digital systems. In: Model Checking Software, pp. 126–131. Springer (2015)
18. Istepanian, R., Whidborne, J.F.: Digital controller implementation and fragility: A modern perspective. Springer Science & Business Media (2012)
19. Jin, X., Donzé, A., Deshmukh, J.V., Seshia, S.A.: Mining requirements from closed-loop control models. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **34**(11), 1704–1717 (2015)
20. Keel, L.H., Bhattacharyya, S.P.: Robust, fragile, or optimal? IEEE Trans. Autom. Control **42**(8), 1098–1105 (1997). DOI 10.1109/9.618239
21. Kroening, D., Tautschnig, M.: CBMC - C bounded model checker - (competition contribution). In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), *LNCS*, vol. 8413, pp. 389–391. Springer (2014)
22. Lahijanian, M., Andersson, S.B., Belta, C.: Formal verification and synthesis for discrete-time stochastic systems. IEEE Trans. Autom. Control **60**(8), 2031–2045 (2015). DOI 10.1109/TAC.2015.2398883
23. Li, P., Yau, S.T.: A new conformal invariant and its applications to the willmore conjecture and the first eigenvalue of compact surfaces. Inventiones mathematicae **69**(2), 269–291 (1982)
24. Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A.D., Grizzle, J.W., Ozay, N., Peng, H., Tabuada, P.: Correct-by-construction adaptive cruise control: Two approaches. IEEE Trans. Contr. Sys. Techn. **24**(4), 1294–1307 (2016)

25. Rojas, A.J.: Step reference tracking in signal-to-noise ratio constrained feedback control. International Journal of Control, Automation and Systems **13**(5), 1131–1139 (2015). DOI 10.1007/s12555-013-9283-9. URL `https://doi.org/10.1007/s12555-013-9283-9`
26. Sadraddini, S., Belta, C.: Formal methods for adaptive control of dynamical systems. arXiv preprint arXiv:1703.07704 (2017)
27. Sadraddini, S., Belta, C.: Formal synthesis of control strategies for positive monotone systems. IEEE Trans. Autom. Control **64**(2), 480–495 (2019). DOI 10.1109/TAC.2018.2814631
28. Springer, T.A.: Invariant theory, vol. 585. Springer (2006)
29. Tran, H.D., Nguyen, L.V., Johnson, T.T.: Large-scale linear systems from order-reduction (benchmark proposal). In: 3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH), Vienna, Austria (2016)
30. Uemura, K., Kamiya, A., Hidaka, I., Kawada, T., Shimizu, S., Shishido, T., Yoshizawa, M., Sugimachi, M., Sunagawa, K.: Automated drug delivery system to control systemic arterial pressure, cardiac output, and left heart filling pressure in acute decompensated heart failure. Journal of Applied Physiology **100**(4), 1278–1286 (2006)
31. Wang, T.E., Garoche, P., Roux, P., Jobredeaux, R., Feron, E.: Formal analysis of robustness at model and code level. In: HSCC, pp. 125–134. ACM (2016)
32. Yordanov, B., Tumova, J., Cerna, I., Barnat, J., Belta, C.: Temporal logic control of discrete-time piecewise affine systems. IEEE Trans. Autom. Control **57**(6), 1491–1504 (2012)