Simulation-Based Design of Industry-Size Control Systems With Formal Quality Guarantees

Marco Esposito, Alberto Leva, *Member, IEEE*, Toni Mancini, Leonardo Picchiami, and Enrico Tronci

***Abstract*—Realistic industrial systems typically need to be modeled as hybrid systems consisting of hundreds (easily *thousands*) of nonlinear differential algebraic equations (DAEs). The size of such models is one of the major obstacles to overcome when developing automated design methods for industrial control systems. In this article, we present a scenario-based approach that, by exploiting the synergies among simulation, black-box optimization, and statistical model checking, allows us to automate the design of *quality-guaranteed* industry-size control systems, i.e., control systems for which a user-specified statistical guarantee on correctness holds over the possible operational scenarios. We show the effectiveness of our approach through a Modelica model consisting of a hybrid nonlinear DAE system with 1276 equations, 492 of which are nontrivial, containing 152 continuous state variables and 38 discrete ones, plus 7 algorithm blocks. Our experiments show that within a few hours of computation on an off-the-shelf workstation, we can find quality-guaranteed solutions (with very tight quality guarantees) to our design problem. We also compute an entire discretized Pareto front for such a large system over two conflicting key performance indicators.**

***Index Terms*—Black-box optimization (BBO), industrial control systems, scenario-based design of control systems, simulation-based design of quality-guaranteed (QG) control systems, statistical model checking (SMC).**

Received 14 October 2024; accepted 2 January 2025. Date of publication 14 February 2025; date of current version 21 April 2025. This work was supported in part by the INdAM "GNCS Project 2024," in part by Sapienza University under Grant RG12117A8B393BDC, Grant RM120172B9F35634, Grant RG1221816C948265, and Grant RG123188B482D2D9, in part by the Lazio POR FESR under Grant E84G20000150006 and Grant F83G17000830007, in part by the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5—Call for tender No. 3277 of 30 December 2021 of the Italian Ministry of University and Research funded by the European Union—NextGenerationEU, project under Grant ECS 0000024 Rome Technopole, Concession Decree No. 1051 of 23 June 2022 adopted by the Italian Ministry of University and Research under Grant CUP B83C22002820006, Rome Technopole. Paper no. TII-24-5385. *(All authors contributed equally to this work.) (Corresponding author: Toni Mancini.)*

Marco Esposito, Toni Mancini, Leonardo Picchiami, and Enrico Tronci are with the Department of Computer Science, Sapienza University of Rome, 00185 Roma, Italy (e-mail: esposito@di.uniroma1.it; tmancini@di.uniroma1.it; picchiami@di.uniroma1.it; tronci@di.uniroma1.it).

Alberto Leva is with the Department of Electronics, Information and Bioengineering, Polytechnic of Milan, 20133 Milano, Italy (e-mail: alberto.leva@polimi.it).

Digital Object Identifier 10.1109/TII.2025.3528556

I. Introduction

SIMULATION models are a fundamental component of the digital twins nowadays employed for designing industrial controls. In applications of engineering interest, such models are large and complex. They typically contain differential algebraic equation (DAE) systems with many equations, easily thousands, and very often such continuous dynamics coexists with *discrete* dynamics and variables, possibly giving rise to discontinuities in the system and in general resulting in *hybrid* DAE models. Furthermore, in many cases, some (dynamic) components of the system (e.g., logic controls) are specified directly as algorithms.

Hence, at the complexity and detail level required for fine-grained optimization, industrial control systems are frequently not amenable to mathematical approaches based on closed-form solutions for the (hybrid) DAEs that describe the system dynamics.

As a result, such large control systems can only be analyzed by means of simulation, and even this path is far from trivial to follow, since it may require specialized hardware if computational speed is at a premium (see, e.g., real-time simulators such as those by OPAL-RT Technologies Inc.). Thus, and not surprisingly, while plenty of well-established control design methods are available for linear systems, this is not equally the case for hybrid control systems *of industrial size and complexity*, because of the characteristics outlined above. In fact, virtually all design methods for hybrid systems tend to focus on quite small-size problems, i.e., on systems with just a few (hardly dozens) equations in both the continuous and the discrete parts, see e.g., [1]. Although such methods are suitable to handle *parts* of a complex control system (and, hence, they do possess industrial validity), they cannot deal with the system as a whole, which is typically too large and complex for them.

In this article, we offer a contribution to the model-based engineering of *large-size* industrial controls, by exploiting the only possibility available for systems of the complexity mentioned above, i.e., simulation. More specifically, we present a scenario-based approach [2] that—building on computationally efficient simulation, black-box optimization (BBO) [3], [4], [5], and statistical model checking (SMC) [6], [7]— allows one to automate the design of control systems whose dynamics is defined through thousands of nonlinear hybrid DAEs.

Since we want our technique to provide *formal guarantees* on the quality of the solutions found [quality-guaranteed (QG) controls], we pose and solve the system design problem as a *feasibility* problem. The reason is that formal guarantees for feasibility can be provided even when only a simulation model for the addressed system is available, whilst providing equal guarantees for *global optimality* in our setting—where for example no convexity property can be assumed—is in general computationally undecidable. Also, note that an optimization problem can be transformed into a *sequence* of feasibility ones, in which bounds are made tighter and tighter till the absence of feasible solutions indicates that the optimum was approximated with the desired precision.

A. Contributions

Our main contributions are as follows.

1) We present a simulation-based technique to design a Quality-Guaranteed (QG) control system. The technique takes the following input:
   a) a black-box model—in the setting of interest, a simulation program—for the system under design;
   b) a set of bounds for the key performance indicators (KPIs) of interest;
   c) a set of decision variables, no matter their nature;
   d) a distribution function for the set of plausible operational scenarios for the system under design;
   e) two real numbers $\varepsilon, \delta \in (0, 1)$.

   The technique is realized as an algorithm that, upon termination, returns a set of values for the above decision variables such that, with probability at least $(1 - \delta)$, the probability of occurrence of a scenario where even a single KPI violates its bound(s) is less than ε.
2) We show that by running our algorithm with different values for the upper bounds of the system KPIs, we can compute a discretized *Pareto front* for the system. This enables the designer to evaluate different tradeoffs among the system KPIs of interest.
3) We present an implementation of our approach using the following:
   a) Modelica as a modeling language for the system and its KPIs;
   b) OpenModelica as a simulation engine;
   c) NOMAD [3], [4] as a BBO solver;
   d) a suitable adaptation of the SMC algorithm in [7].
4) We present an evaluation of our approach to a meaningful case study: a Modelica model for a heating station. Our case study is a hybrid nonlinear DAE system with 1276 equations, 492 of which are nontrivial (i.e., not related to component interconnection), containing 152 continuous state variables and 38 discrete ones, plus 7 algorithm blocks. We use two conflicting KPIs: the average power (AP) demand (to favor solutions that save energy) and the average violation of the required temperature profile (to favor solutions that closely follow that temperature profile). Our experimental results show that, within a few hours of computation on an off-the-shelf workstation, we can find a solution to our design problem, and how we can effectively compute an entire discretized Pareto front for such a large system.
5) We show that the classical approach (see, e.g., [8]) of computing the system parameters considering just the *nominal* scenario (non-QG controls) leads to a significantly suboptimal behavior when evaluated on *off-nominal* (yet *plausible*) scenarios. This shows the benefits of our scenario-based approach providing formal quality guarantees.

B. Related Work

The approach closest to ours is scenario-based system design, e.g., [2], [9], [10], [11]. Scenario-based approaches compute an optimal solution to the design problem by solving a constrained optimization problem, whose constraints stem from the *plausible* system operational scenarios.

With respect to the (mainstream) scenario-based setting, we provide contributions along two directions. First, we do not require an explicit mathematical expression for the function to be optimized or for the constraints to be satisfied. Instead, we just use a simulator to compute such values and BBO methods to find candidate solutions. Second, while the typical scenario-based approach computes the number N of scenarios beforehand, based on the desired statistical confidence δ and error probability ε (e.g., see [2, eq. (2)], where β is used for our δ), we compute the number of scenarios at run time using SMC methods (namely, [7]) that adaptively compute N based on the constraint violation probability distribution function estimated during the SMC process. As shown in [7], doing so typically requires a smaller number of samples than approaches computing N beforehand.

The synergistic use of simulation, BBO, and SMC allows us to extend the scenario-based approach to industrial systems whose models contain thousands of DAEs (see Section VI). To the best of authors' knowledge, such systems are out of reach for any previously published technique. Note, in this respect that our approach requires to cast the control design problem as a feasibility problem, whereas classical scenario-based approaches (e.g., [2], [9], [10], [11]) can address a control design problem formulated as an optimization problem.

BBO and SMC have already been used in [12] for the optimal fault-tolerant positioning of sensors, ensuring the satisfaction of requirements for all operational scenarios. However, that approach cannot be used for simulation-based control system design, since it cannot accommodate simulation of control systems.

Black-box approaches to system verification through SMC (e.g., [6]) have been widely investigated. For example, SMC has been used for formal verification of hybrid systems in [1], [6], [13], [14], [15], and [16]. In [17] (and citations thereof), system-level formal verification of cyber-physical systems is conducted by optimizing simulation campaigns for all operational scenarios of interest. Those approaches can be used for system verification, but not for system design, which is our problem here.

As for simulation-based approaches, we note that model predictive control (MPC) is widely used to design complex

systems. MPC forecasts the future behavior of the system by means of a model of the controlled plant [18], [19], [20], be this based on first principles [21], [22], on learning techniques [23], [24], or hybrid [25]. The same idea can be exploited by using plant descriptions composed of data without the necessity of a dynamic model in the sense of the systems theory, resulting in the data-based control approach [26], [27]. Attempts to integrate the two main approaches just mentioned can also be found, see, e.g., [28], as each has potential and pitfalls. The main obstacles that MPC has to overcome are the mathematical and/or computational complexity of the cost functions to minimize, and the system size (in terms of equations or data amount) to handle. The former has led to the development of the so-called economic model predictive control (EMPC) framework [29], while the latter is classically tackled by model decomposition or reduction [30], [31]. (E)MPC approaches are typically implemented through a *receding horizon* schema. Thus, they can only be used if enough time is available to compute the future behavior of the plant. If a control law that is fast to compute at run time is sought, then (E)MPC cannot be used. Scenario-based approaches instead address exactly the case where a fast control law is sought. Thus, (E)MPC and scenario-based approaches address different design needs.

Summing up, to the best of authors' knowledge, no methods exist that support the design of industrial systems of realistic size, guaranteeing that, with a given statistical confidence, the probability that an operational scenario violating the system requirements materializes is smaller than a given threshold (*quality guarantee*). By exploiting the synergies between simulation, BBO, and SMC, we propose a scenario-based approach capable of achieving such a goal.

C. Summary

The rest of this article is organized as follows. Section II provides background to make the paper self-contained. Section III formally defines the problem we address and provides an algorithm to solve it. Section IV describes an efficient implementation for the algorithm in Section III. Section V describes the case study we use to evaluate our approach. Section VI shows our experimental results. Finally, Section VII concludes this article.

II. Background and Notation

Throughout this article, we denote with $\mathbb{Z}$ and $\mathbb{R}$ the sets of, respectively, integer and real numbers.

If x is a vector of n components, then x_i denotes its ith component. Thus, $x = [x_1, \ldots, x_n]$. If a and b are n-dimensional vectors, then $a \leq b$ stands for $\forall i \in \{1, \ldots, n\}$ $(a_i \leq b_i)$. As usual, 0 denotes the n-dimensional vector whose components are all 0.

An n-dimensional *discrete* (*continuous*) time *signal* is a function from an interval $\mathcal{I} \subseteq \mathbb{Z}$ ($\mathcal{I} \subseteq \mathbb{R}$) representing time to $\mathbb{R}^n$. If u is a signal over interval $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$ is an interval, then we denote with $u|_{\mathcal{J}}$ the *restriction* of u to $\mathcal{J}$.

If X is a random variable, we write $x \in X$ to mean that x is a realization of X.

Given a positive real number T, we denote with $H(T)$ (*hold operator*) the function taking as input an n-dimensional discrete time signal d and returning as output a piece-wise constant continuous time signal. Namely (writing $H(T, d, t)$ for $H(T, d)(t)$): $H(T, d, t) = d(\lfloor \frac{t}{T} \rfloor)$.

A. Black-Box Optimization

A BBO problem is a tuple (J, G) where: $J : \mathbb{R}^n \to \mathbb{R}$ (objective function) and $G : \mathbb{R}^n \to \mathbb{R}^m$ (constraints violation function), with $n, m \geq 1$. A solution (if any) to the BBO problem (J, G) is real vector $z \in \mathbb{R}^n$ such that: 1) $G(z) \leq 0$ (all constraints are satisfied); 2) $\forall x \in \mathbb{R}^n \; [(G(x) \leq 0) \to (J(z) \leq J(x))]$ (the objective function is minimized).

A *feasibility problem* is a BBO problem where J is identically 0, i.e., $(0, G)$. We denote with (G) a feasibility problem.

We will use the term *BBO oracle* to denote any function that takes a BBO problem and returns a solution to it, if one such solution exists, and $\perp$ otherwise. Determining the existence of a solution to a feasibility problem is computationally *undecidable*, and so is a fortiori existence of a solution to a BBO problem. Thus, there exists no algorithm implementing a BBO oracle (but such *complete* algorithms exist in case the BBO problem at hand satisfies strong-enough assumptions, e.g., convexity). However, even in the general case, there are many algorithms and solvers that, given enough computation time, can come as close as we need to a BBO oracle. Examples are NOMAD [3], [4] and Nevergrad [5].

For the sake of clarity, through this article, we will use BBO oracles to define our methods and approach neatly, and then will explain how an actual BBO solver (NOMAD in our case) can be used to implement the proposed algorithms.

More specifically, all BBO solvers have stopping criteria to guarantee termination. Upon termination, the solver may or may not return a solution. If it does not return a solution, we may *not* rule out the possibility that one exists. On the other hand, when a solution is returned, the solver guarantees that it is feasible, but, in general, no formal guarantee is provided about its global optimality. Since we will be using BBO solvers to solve feasibility problems, we will be safe. That is, we may fail to find the sought solution when one actually exists, but we will never return a value that is not feasible.

B. Statistical Model Checking

SMC (see, e.g., [6] for an overview) is a set of methods and algorithms to verify that, with a given *statistical confidence*, a system satisfies given requirements. We will use Monte Carlo-based SMC to compute an upper bound to the probability that a given function is not identically 0. To that end, we will use the following theorem.

Theorem 1: Let X be an n-dimensional real-valued random variable, $f : \mathbb{R}^n \to \{0, 1\}$, and $\varepsilon, \delta \in (0, 1)$. Then, an algorithm exists such that:

1) it terminates;
2) it returns a finite set (of *counterexamples*) $C \subset \mathbb{R}^n$ such that $\forall \omega \in C \; f(\omega) = 1$;

3) if $C = \emptyset$, then, with probability at least $(1-\delta)$ we have that $\Pr(f(X) = 1) \leq \varepsilon$.

Proof: Let Z be a Bernoulli random variable with unknown parameter γ, defined as: $Z = f(X)$. Thus, $\gamma = \Pr(f(X) = 1)$.

Our sought algorithm builds on [7, Algorithm 1]. The latter iteratively refines, via repeated independent and identically distributed Monte Carlo samples $\omega^{(1)}, \omega^{(2)}, \ldots$ of X, an estimation $\hat{\gamma}$ of γ, obtained as the sample average of $f(\omega^{(1)}), f(\omega^{(2)}), \ldots$.

The algorithm in [7] relies on a termination condition (based on statistical results proved in [7, Sec. 5]) that, when reached, entails that $\Pr(|\hat{\gamma} - \gamma| > \varepsilon) \leq \delta$.

Our sought algorithm is obtained by simply modifying the algorithm in [7] so that all found counterexamples (i.e., all samples $\omega^{(k)}$ such that $f(\omega^{(k)}) = 1$) are collected in set C. The thesis then follows from the proof of correctness in [7, Th. 5.3], observing that, when $C = \emptyset, \hat{\gamma} = 0$ and so $\gamma = \Pr(f(X) = 1) \leq \varepsilon$ with probability at least $(1-\delta)$. ■

When X is clear from the context, we denote with $\mathrm{SMC}(\varepsilon, \delta, f)$ the output C of the algorithm in the proof of Theorem 1.

III. Problem Statement and Solution Algorithm

In the following, we define our control design problem (see Section III-A) along with a solution algorithm for it (see Section III-B). Section IV describes our implementation.

A. Problem Statement

We model our *system under design* (typically, an industrial plant) as a dynamical system with only *uncontrollable* inputs, since our system comprises both the controlled and controlling subsystems.

Definition 1: A closed-loop system (CLS) $\mathcal{S}$ is a tuple (g, T, W, D, z), where:

1) *Stochastic parameters:* W is an r-dimensional real-valued random variable. We denote with w a realization of W. W models system parameters that are not under the control of the system designer, for example, uncertainties in some of the system parameters or the *plausible* initial states of the system.
2) *Uncontrollable inputs:* D is a q-dimensional discrete-time real-valued stochastic process (e.g., random noise). We denote with d a realization of D. D models uncontrollable inputs from the environment, e.g., *disturbances*, *faults*, inputs from other systems or users.
3) *Design parameters:* Vector $z \in \mathbb{R}^l$ models system parameters that the designer can choose.
4) *Output function:* For each continuous time point $t \in \mathbb{R}$, the output $y(t)$ of our system is computed through function g as follows:

$$y(t) = g\left(w, H(T,d)|_{[0,t)}, z, t\right) \tag{1}$$

where $H(T,d)|_{[0,t)}$ is the restriction of $H(T,d)$ to time interval $[0,t)$.

Positive real value T defines the holding time for the discrete values from $d \in D$.

Value $y(t) \in \mathbb{R}^m$ is an m-dimensional real-valued vector. Since here we are only interested in evaluating the system KPIs, in our setting $y(t)$ will be the value of such KPIs at time t when the system design parameters are set to z.

A realization (w, d) for the pair (W, D) will be named *operational scenario* (or just *scenario* in the following).

We are interested in evaluating the value of the system KPIs at the end of a simulation. So, let T_h be our time horizon, we are interested in evaluating $g(w, H(T,d)|_{[0,T_h)}, z, T_h)$. Since, in our setting, T and T_h are fixed once and for all, by abuse of language, in the following, we write $g(w, d, z)$ for $g(w, H(T,d)|_{[0,T_h)}, z, T_h)$. Our goal is to select the system design parameters z so that, for all *plausible* (i.e., that can actually materialize) operational scenarios, we have $y(T_h) \leq 0$, that is $g(w, d, z) \leq 0$. In the following, we formalize such a requirement.

Definition 2: A *QG control design problem* is a tuple $(\mathcal{S}, \varepsilon, \delta)$ where $\mathcal{S}$ is a CLS as in Definition 1, and $\varepsilon, \delta \in (0,1)$. A solution to such a problem is a value $z \in \mathbb{R}^l$ such that, with probability at least $(1-\delta)$, we have: $\Pr[\neg(g(W, D, z) \leq 0)] \leq \varepsilon$.

In other words, a QG control design problem asks us to select system parameters z so that with statistical confidence at least $(1-\delta)$, the probability that an operational scenario $(w, d) \in (W, D)$ violating the system requirements materializes is at most ε.

In our setting, we cannot rely on any hypothesis on the distribution probability of the parameters W, the stochastic process D modeling uncontrollable inputs, and the system dynamics g. In such a general setting, the main challenge to overcome when solving a QG problem is showing that a given candidate solution is *robust enough* to withstand, with the desired probability, all operational scenarios.

We address such a challenge by first identifying an algorithm that can correctly solve our problem from a mathematical perspective (this is done in Section III-B) and then devising an efficient engineering of the proposed algorithm so that we can actually use our algorithm on industrial size problems (this is done in Section IV).

B. Solution Algorithm

Our algorithm for solving the QG control design problem in Definition 2 is shown as Algorithm 1. Theorem 2 states its correctness.

Theorem 2: Let $(\mathcal{S}, \varepsilon, \delta)$ be as in Definition 2. Upon termination, Algorithm 1 returns z such that:

1) if $z = \perp$, then the problem has no solution;
2) otherwise, $z \in \mathbb{R}^l$ is a solution to it.

Proof: Note that $H(T,d)|_{[0,t)}$ depends only on a finite number of values of d, namely those at time points before t.

Thus, since horizon T_h is finite, a scenario (w, d) can be finitely represented by encoding d as a finite vector of values.

Let S be a set of scenarios. In the following, we write $g(S, z) \leq 0$ for $\forall (w, d) \in S\ [g(w, d, z) \leq 0]$.

Furthermore, we denote with $g(S)$ the function such that $g(S)(z) = g(S, z)$, and with the lambda expression $\lambda wd.[g(w, d, z) \leq 0]$ the function taking as argument a scenario (w, d) and returning 1 if $g(w, d, z) \leq 0$ and 0 otherwise.

Algorithm 1: Algorithm for QG Control Design.

require: $\varepsilon, \delta \in (0,1)$
ensure: $z \in \mathbb{R}^l$ or $z = \perp$
1: BBO $\leftarrow$ a BBO oracle $\triangleright$ Section II-A
2: $S \leftarrow$ a nonempty set of scenarios (w, d)
3: $C \leftarrow \emptyset$; $\triangleright$ Counterexamples
4: **repeat**
5: $S \leftarrow S \cup C$;
6: $z \leftarrow \text{BBO}(g(S))$; $\triangleright$ Section II-A
7: **if** $z = \perp$ **then**
8: **return** $\perp$; $\triangleright$ BBO problem infeasible
9: $C \leftarrow \text{SMC}(\varepsilon, \delta, \lambda wd.[g(w,d,z) \leq 0])$; $\triangleright$ Section II-B
10: **until** $C = \emptyset$;
11: **return** z;

As an initial step (line 2), Algorithm 1 inserts into set S a number of scenarios. Any positive number will do. This affects performance, but not correctness.

From the current set S, in line 6, we compute a design solution (if any) $z \in \mathbb{R}^l$, by solving the feasibility problem $g(S)$, as described in Section II-A.

If $g(S)$ does not have a solution, (i.e., the BBO oracle returns $\perp$), we terminate and conclude that no design parameter z exists that satisfies the given constraints on the KPIs for all scenarios in S.

During the SMC step (line 9), by using the algorithm in Theorem 1 in Section II-B, we check whether our computed solution z satisfies the required formal quality guarantees. Namely, we check that, with probability at least $1 - \delta$, the probability that a scenario (w, d) such that $\neg(g(w,d,z) \leq 0)$ materializes is at most ε.

If the set C of counterexample scenarios computed in line 9 is empty we are done, and our thesis follows from Theorem 1. Otherwise, we add the counterexample scenarios in C to our set of scenarios and start a new algorithm iteration from line 5. ■

Remark 1: The presence of the stochastic parameter W in Definition 1 makes the CLS input a nonergodic stochastic process. This implies that it is not enough to assess KPIs (i.e., $g(S, z)$) using a single, *long enough* simulation. We must also evaluate our system on a *large enough* number of realizations (scenarios). The role of SMC in Algorithm 1 is exactly to evaluate when enough scenarios have been considered to attain the sought guarantee about the solution quality.

IV. Algorithm Design

In this section, we describe how we engineered Algorithm 1.

A. BBO Solver, Line 1

Since there exists no program implementing what, in Section II-A, we termed *BBO oracle*, we use NOMAD [3], [4] as a *BBO solver*. This is safe, since we are solving a feasibility problem. Indeed, NOMAD might fail to return a solution when one exists, but it will never return a value $z \in \mathbb{R}^l$ that is not a feasible solution.

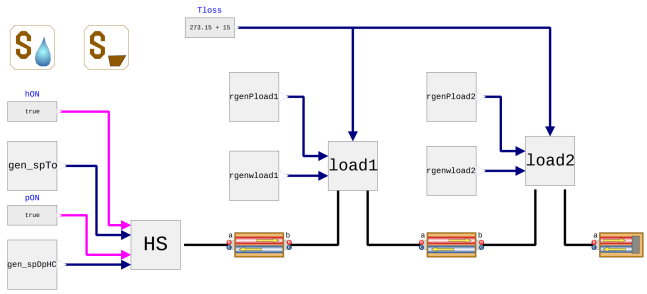


Fig. 1. Case study: overall model.

B. Initial Scenarios, Line 2

From the proof of Theorem 2, we know that any nonempty initial set S of scenarios yields a correct algorithm. However, the choice of S may have a critical impact on the algorithm performance. Indeed, choosing a *too small* set S could yield the BBO step in line 6 to find a solution z that would likely not withstand other, unseen scenarios. This would make more likely the subsequent SMC phase fail, and this, in turn, by producing a nonempty set C of counterexample scenarios (line 9), would likely yield a high number of iterations of the repeat-until loop. On the other hand, choosing a *too large* S would likely require a significant computational effort for all such scenarios to be simulated at each step of the BBO solving routine (line 6).

Our algorithm depends on hyperparameter n_0, which defines the number of initial scenarios to be randomly sampled and stored into set S. Experiments show (see Section VI-C) that a good value for n_0 is the number of available CPU cores.

C. BBO Step, Line 6

As already stated, our idealized BBO oracle is safely replaced by the BBO solver NOMAD.

D. SMC Step, Line 9

We follow the approach of Section II-B with an important caveat to boost performance. From the proof of Theorem 1, it follows that any choice for the number of returned counterexample scenarios (set C) will preserve soundness of the overall algorithm. However, the actual choice of how many counterexamples to return might impact the overall performance.

To this end, we equip our SMC algorithm with a second hyperparameter n_1, and terminate each SMC step either by declaring the candidate solution feasible (case $C = \emptyset$), or immediately after having collected the first n_1 counterexamples. Thus, the value of n_1 actually defines an *early SMC termination policy* that can be disabled by setting n_1 to $+\infty$.

In Section VI-C, we show that a suitable early SMC termination policy is critical, as it greatly affects the completion time of each SMC step, the number of iterations of the main loop, and the completion time of each BBO solver run, and that a good value for n_1 is, as for n_0, the number of available CPU cores.

E. Termination

In principle, we may discover counterexample scenarios, i.e., scenarios violating our requirements, at each SMC step,

and this would result in an infinite sequence of BBO and SMC phases. This is especially true in case of design errors in the plant, which hinder the existence of a satisfactory controller.

To guarantee termination in all cases, along the lines of iterative improvement AI methods, our engineered algorithm is equipped with a user-specified time budget, after which it stops declaring that no solution was found.

V. Case Study

We evaluate our approach on an industrial case study of realistic size, namely the design of control parameters for a heat network modeled with the Modelica language.

The model (see Fig. 1 for a high-level view and the online repository associated to this article[1] for full code) is entirely built on first-principle equations, namely dynamic balances of mass, energy, and momentum. Zero-dimensional fully mixed models are used for "short" components like heaters and valves. One-dimensional elements (most notably, pipes) are spatially discretized according to the finite-volume approach, comprising nonlinear relationships between flows and pressure drops as well as to describe heat transfer (e.g., the Dittus–Bölter correlation). Controllers are represented as digital algorithms executed at constant rates.

As a result, the model comprises a highly nonlinear hybrid DAE system, with 1276 equations, 492 of which are nontrivial (i.e., not related to component interconnection), containing 152 continuous state variables and 38 discrete ones, plus 7 algorithm blocks. Pseudorandom generators are used to produce stochastic load profiles on top of an average daily behavior, so as to subject the system to boundary conditions with a realistic variability over time.

Notwithstanding its complexity, far within the reach of modern simulation tools but quite remarkable when talking about BBO, the observed simulation time is good: with OpenModelica 1.21.0, configured to use the DASSL variable-step DAE solver, an entire simulation run completes on average within 5.65 s (stddev 1.18 s), on an off-the-shelf workstation (CPU AMD EPYC(R) 7301 @2.2 GHz, 256 GB RAM).

For such a system, we wish to compute values for *design parameters* $z \in \mathbb{R}^8$ (Definition 1) modeling set points for time, temperature, and pressure of switches between different control strategies, satisfying two conflicting requirements: 1) the AP must be below a given bound b_{AP}; 2) the average return temperature violation (ARTV) must be below a given bound b_{ARTV}.

AP at time $t > 0$, $\text{AP}(t)$, is $[t^{-1} \int_0^t \text{p}(\tau)\text{d}\tau]$, where $\text{p}(t)$ is the system's power requirement at time t. ARTV at time $t > 0$, $\text{ARTV}(t)$, is $[t^{-1} \int_0^t \text{RTV}(\tau)\text{d}\tau]$, where $\text{RTV}(t) = \max\{0, 30 - \text{RT}(t)\}$ is the return temperature violation at time t, i.e., how much the return temperature of the heating station, $\text{RT}(t)$, is below the target value of 30 °C.

We set our *stochastic uncontrollable input process* D as white noise modeling uncertainties in the load profiles, and *stochastic parameter* w as a heat transfer coefficient (*gammaloss* in the Modelica model) used to model the loss of heat from the heat exchanger in a substation. Time horizon is set to $T_h = 3$ days.

Thus, from Definition 2, we have $g(w, d, z) = (\text{AP}(T_h), \text{ARTV}(T_h)) - (b_{\text{AP}}, b_{\text{ARTV}}) \leq 0$. By changing $b = (b_{\text{AP}}, b_{\text{ARTV}})$, we can tighten (decrease b) or relax (increase b) our requirements. Accordingly, for short, we will regard b as our system design requirements.

[1][Online]. Available: https://github.com/RAISE-Sapienza/qg-design-tii-2025

VI. Experiments

We experimentally evaluate our approach by computing QG controls for our case study in Section V, and aim at answering the following three questions.

1) What is the quality of the designed controls (in terms of bounds $b = (b_{\text{AP}}, b_{\text{ARTV}})$ to our two KPIs), and how they compare to non-QG controls, i.e., controls computed based on a single scenario only, defining the *nominal* system inputs (see Section VI-A).
2) What is the impact, on the overall performance, of SMC (and its error and confidence thresholds ε and δ) to formally guarantee the quality of each candidate control (see Section VI-B).
3) What is the impact on the overall performance of the number of initial scenarios and of the SMC early termination policy (parameters n_0 and n_1, Section VI-C).

Each experiment was conducted over the 32 cores of an off-the-shelf workstation (AMD EPYC(R) 7301 CPU @2.2 GHz, 256 GB of RAM), by relying on an executable system simulator generated via OpenModelica v1.21.0, and on NOMAD v4.3.1 as BBO solver.

A. Quality of Computed QG Controls

For these experiments, we considered the tightest SMC thresholds among those considered in Section VI-B (i.e., $\varepsilon = \delta = 1 \times 10^{-3}$) and the values for the number of initial scenarios (parameter $n_0 = 32$) and for SMC early termination policy (parameter $n_1 = 32$) that, on average, show the most predictable (usually the best or near-best) computation time, as discussed in Section VI-C (see those forthcoming sections for an in-depth analysis).

As a baseline against which to compare the quality of the designed QG controls, we considered a *nominal* input scenario defined (see Definition 1) by using the nominal (i.e., average) value for w and by turning off noise d superimposed to the nominal load profile. We then designed controls that satisfy various bounds for our KPIs under such nominal scenario (*non*-QG controls). Fig. 2 shows a discretized Pareto front (grey dots) of the successfully computed non-QG controls over a grid for the bounds of our two conflicting KPIs. Pareto optimality of such points (on the grid) has been inferred by observing the failure of our approach (within a generous timeout of 24 h) to compute correct (non-QG) controls in the points denoted by grey x's in the figure.

Fig. 2 also shows the discretized Pareto front (blue dots) of the successfully computed QG controls. Obviously, non-QG controls dominate QG controls, since satisfying only the nominal scenario is a much more easily attainable goal for a control

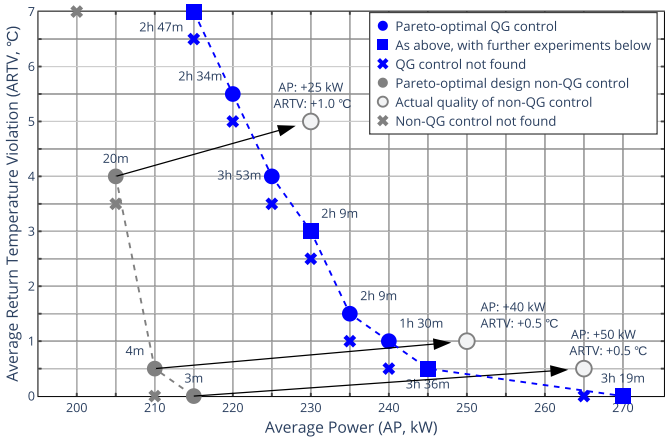


Fig. 2. Discretized Pareto fronts of QG (blue) and non-QG (grey) controls ($\varepsilon = \delta = 1 \times 10^{-3}$); computation time; actual quality of non-QG controls.

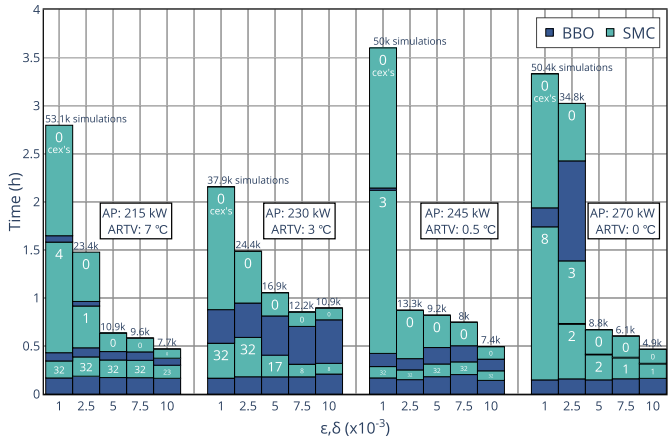


Fig. 3. Impact of SMC on performance: breakdown of computation time (BBO and SMC steps in each algorithm iteration) for QG controls denoted by blue squares in Fig. 2 for various values of ε and δ.

system. Also computation times are higher, jumping from 3 to 20 min as of non-QG controls to a few hours when quality guarantee is requested.

However, such more easily computable non-QG controls easily *fail* when the system is fed with non-nominal scenarios: indeed, in Fig. 2, for each Pareto optimal non-QG control (grey dot), we show (light-grey bordered dots) the *minimum* values (on the grid) for the KPI bounds that it *actually* satisfies, when verified by SMC on non-nominal scenarios (again $\varepsilon = \delta = 1 \times 10^{-3}$). This clarifies that such non-QG controls introduce a substantial *design error*, and may expose the system to clearly suboptimal performance and Pareto nonoptimality. For example, an error on AP KPI bound indeed witnesses a substantially wrong prediction of the power consumed by the system (as high as 50 kW i.e., $+23.25\%$, for the rightmost non-QG control in the figure), and directly results in a non-negligible financial loss. We also point out that the outcome of this kind of design errors is *unpredictable*, thus, not easily manageable at design time.

B. Impact of SMC on Performance

Fig. 3 shows an in-depth analysis of the time needed to compute the four Pareto-optimal QG controls denoted with blue squares in Fig. 2 (evenly spaced throughout the Pareto front and representative of all the others).

Namely, for each such control, the figure shows the breakdown of the computation time among the BBO and the SMC steps in each algorithm iteration (starting from the bottom), where we kept $n_0 = n_1 = 32$ as in Section VI-A, but varied the values of the two SMC parameters ε and δ (for succinctness, we always kept $\varepsilon = \delta$, whilst an analysis encompassing also $\varepsilon \neq \delta$ is available in the online repository associated to this article[1]). The heights of bar stacks associated to $\varepsilon = \delta = 1 \times 10^{-3}$ correspond to the computation times shown in Fig. 2.

The number on top of each bar is the overall number of scenarios simulated by our algorithm during the design of that control, whilst the number within each bar referring to an SMC step denotes the number of counterexample scenarios found during that step, and acquired, as additional constraints, into the next BBO step.

From the figure, it clearly emerges that the time devoted to the guarantee assessment of candidate solutions (SMC steps) largely *dominates* the overall computation time, especially when the tightest guarantees are sought ($\varepsilon = \delta = 1 \times 10^{-3}$), in which cases a higher number of simulations is required and a higher number of counterexample scenarios is also often generated at each SMC step (although always upper bounded by $n_1 = 32$). Conversely, the figure shows that the overall computation time can be drastically reduced by 2–3 times, and easily to less than 1 hour, by requiring *slightly looser quality guarantees* (i.e., just slightly larger values for ε and δ).

C. Impact of the Number of Initial Scenarios and of the SMC Early Termination Policy on Performance

Fig. 4 shows, in the same style as Fig. 3, the time needed to compute the same four Pareto-optimal QG controls (those denoted by blue squares in Fig. 2, and again split into the BBO and SMC steps of each algorithm iteration, starting from the bottom), where we kept $\varepsilon = \delta = 1 \times 10^{-3}$ as in Fig. 2, but varied the values of n_0 (number of randomly sampled initial scenarios, Section IV-B) and n_1 (early SMC termination policy, Section IV-D). Here, we used a generous timeout of 24 h to forcibly terminate some runs.

The figure shows that the *most predictable* computation times (which are often the best, or near-best) are achieved with $n_0 = n_1 = 32$. These are the values we actually fixed in the experiments of Figs. 2 and 3.

The observed behavior can be explained by observing that 32 is the number of CPU cores of our workstation. Hence, these values are likely to enable an efficient execution of the multicore BBO solving process, with *one parallel thread per initial scenario*, and a *moderate increase* (throughout the various iterations of the algorithm) of the number of scenarios to be considered as additional constraints (at most one additional scenario per algorithm iteration per CPU core). At the same

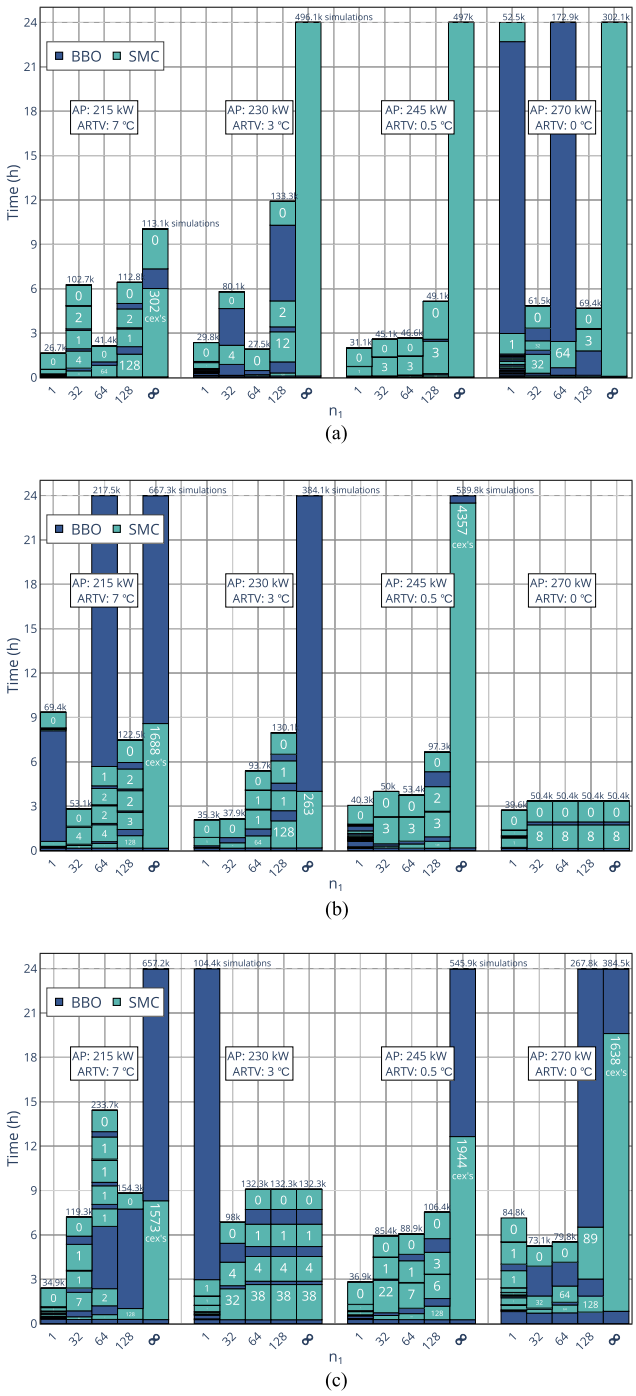


Fig. 4. Impact of the number of initial scenarios (n_0) and of the early SMC termination policy (n_1) on performance: breakdown of computation time (BBO and SMC phases along each algorithm iteration) for QG controls denoted by blue squares in Fig. 2 for various values of n_0 and n_1. Timeout: 24 h. (a) $n_0 = 1$. (b) $n_0 = 32$. (c) $n_0 = 64$.

time, such a value for n_1 enables the early termination of each SMC step, whose time share is typically the highest (as shown in Fig. 4).

Indeed, higher values for n_1 might severely increase both the completion time of each SMC step, and that of the forthcoming BBO steps, as the BBO solver could be overwhelmed with too many additional constraints to handle (often thousands when $n_1 = \infty$). Conversely, smaller values for n_1 could yield too few counterexample scenarios to be added into the problem solved during the forthcoming BBO steps. This, in turn, makes more likely that the candidate solutions found by the solver in the next iterations will be rejected during verification. All this would likely result in more iterations (see, in particular, the bars of Fig. 4 referring to $n_1 = 1$).

Analogous issues might occur, although with lower severity, when initializing the algorithm with a single ($n_0 = 1$) or too many scenarios ($n_0 = 64$).

VII. Conclusion

In this article, we presented a scenario-based approach that, by exploiting the synergies among simulation, BBO, and SMC, allows us to automate the design of QG industrial size control systems, i.e., control systems for which a user-specified statistical guarantee on correctness holds over the possible operational scenarios.

We showed experimentally that, within a few hours of computation on an off-the-shelf workstation, we can compute QG controls (with very tight quality guarantees) for a dynamic Modelica model of high fidelity and industrial size and complexity. Also, we successfully computed an entire discretized Pareto front of optimal QG controls over two conflicting KPIs.

To the best of authors' knowledge, our approach is the only one viable when no closed form is available for the KPIs of the system under design. However, if a closed-form expression is available for the system KPIs, then, obviously, white-box scenario-based approaches (e.g., [2], [9], [10], [11]) should be preferred, as they will usually be computationally more efficient (see, e.g., [32, Sec. 10.6], which shows how a straightforward white-box method—gradient descent—is typically more efficient than many of the state-of-the-art BBO methods). Accordingly, our approach should be used only when white-box scenario-based methods are hindered because no closed-form expression is available for the system KPIs.

Acknowledgment

The authors are would like to thank anonymous reviewers for their comments and suggestions, which have improved the quality of this article.

References

[1] J. Lunze and F. Lamnabhi-Lagarrigue, *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

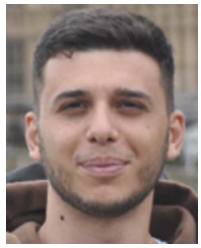
[2] M. Campi, A. Carè, and S. Garatti, "The scenario approach: A tool at the service of data-driven decision making," *Annu. Rev. Control*, vol. 52, pp. 1–17, 2021.

[3] C. Audet, S. Le Digabel, V. Montplaisir, and C. Tribes, "Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm," *ACM Trans. Math. Softw.*, vol. 48, no. 3, pp. 1–22, 2022.

[4] S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm," *ACM Trans. Math. Softw.*, vol. 37, no. 4, pp. 1–15, 2011.

[5] P. Bennet, C. Doerr, A. Moreau, J. Rapin, F. Teytaud, and O. Teytaud, "Nevergrad: Black-box optimization platform," *ACM SIGEVOlution*, vol. 14, no. 1, pp. 8–15, 2021.

[6] G. Agha and K. Palmskog, "A survey of statistical model checking," *ACM Trans. Model. Comput. Simul.*, vol. 28, no. 1, pp. 6:1–6:39, 2018.
[7] C. Jegourel, J. Sun, and J. Dong, "Sequential schemes for frequentist estimation of properties in statistical model checking," *ACM Trans. Model. Comput. Simul.*, vol. 29, no. 4, pp. 1–22, 2019.
[8] D. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Belmont, MA, USA: Athena Scientific, 2017.
[9] S. Garatti, A. Carè, and M. Campi, "Complexity is an effective observable to tune early stopping in scenario optimization," *IEEE Trans. Autom. Control*, vol. 68, no. 2, pp. 928–942, Feb. 2022.
[10] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 742–753, May 2006.
[11] M. Campi, S. Garatti, and F. Ramponi, "A general scenario theory for nonconvex optimization and decision making," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4067–4078, Dec. 2018.
[12] M. Esposito, T. Mancini, and E. Tronci, "Optimizing fault-tolerant quality-guaranteed sensor deployments for UAV localization in critical areas via computational geometry," *IEEE Trans. Syst., Man Cybern.: Syst.*, vol. 54, no. 3, pp. 1515–1526, Mar. 2024.
[13] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani, "Automated verification and synthesis of stochastic hybrid systems: A survey," *Automatica*, vol. 146, 2022, Art. no. 110617.
[14] R. Lal and P. Prabhakar, "Counterexample guided abstraction refinement for polyhedral probabilistic hybrid systems," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, pp. 1–23, 2019.
[15] B. Xue, M. Zhang, A. Easwaran, and Q. Li, "PAC model checking of black-box continuous-time dynamical systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3944–3955, Nov. 2020.
[16] B. Weng, L. Capito, U. Ozguner, and K. Redmill, "A formal characterization of black-box system safety performance with scenario sampling," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 199–206, Jan. 2022.
[17] T. Mancini, I. Melatti, and E. Tronci, "Optimising highly-parallel simulation-based verification of cyber-physical systems," *IEEE Trans. Softw. Eng.*, vol. 49, no. 9, pp. 4443–4455, Sep. 2023.
[18] C. García, D. Prett, and M. Morari, "Model predictive control: Theory and practice–A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
[19] S. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003.
[20] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *Int. J. Adv. Manuf. Technol.*, vol. 117, no. 5/6, pp. 1327–1349, 2021.
[21] M. Gräber, C. Kirches, J. Schlöder, and W. Tegethoff, "Nonlinear model predictive control of a vapor compression cycle based on first principle models," *IFAC Proc. Volumes*, vol. 45, no. 2, pp. 258–263, 2012.
[22] T. Mejdell, H. Kvamsdal, S. Hauger, F. Gjertsen, F. Tobiesen, and M. Hillestad, "Demonstration of non-linear model predictive control for optimal flexible operation of a CO_2 capture plant," *Int. J. Greenhouse Greenhouse Control*, vol. 117, 2022, Art. no. 103645.
[23] L. Hewing, K. Wabersich, M. Menner, and M. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 3, pp. 269–296, 2020.
[24] M. Rashedi, H. Khodabandehlou, M. Demers, T. Wang, and C. Garvin, "Model predictive controller design for bioprocesses based on machine learning algorithms," *IFAC-PapersOnLine*, vol. 55, no. 7, pp. 45–50, 2022.
[25] D. Kim, J. Lee, S. Do, P. Mago, K. Lee, and H. Cho, "Energy modeling and model predictive control for HVAC in buildings: A review of current research trends," *Energies*, vol. 15, no. 19, 2022, Art. no. 7231.
[26] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Inf. Sci.*, vol. 235, pp. 3–35, 2013.
[27] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 657–667, Jan. 2014.
[28] J.-W. Huang and J.-W. Gao, "How could data integrate with control? A review on data-based control strategy," *Int. J: Dyn. Control*, vol. 8, no. 4, pp. 1189–1199, 2020.
[29] M. Ellis, H. Durand, and P. Christofides, "A tutorial review of economic model predictive control methods," *J. Process Control*, vol. 24, no. 8, pp. 1156–1178, 2014.
[30] M. Katebi and M. Johnson, "Predictive control design for large-scale systems," *Automatica*, vol. 33, no. 3, pp. 421–425, 1997.
[31] S. Hovland, J. Gravdahl, and K. Willcox, "Explicit model predictive control for large-scale systems via model reduction," *J. Guid., Control, Dyn.*, vol. 31, no. 4, pp. 918–926, 2008.
[32] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Berlin, Germany: Springer, 2017.



Marco Esposito received the master's and Ph.D. degrees in computer science from Sapienza University, Rome, Italy, in 2018 and 2022, respectively.

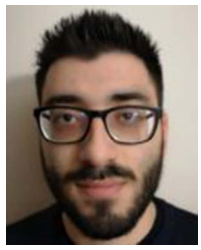
He is currently a Postdoctoral Research Associate with the Department of Computer Science, Sapienza University of Rome, Rome, Italy. His research interests comprise: artificial intelligence, formal verification, cyber-physical systems, control software synthesis, systems biology.

Alberto Leva (Member, IEEE) received the master's degree in electronic engineering from the Polytechnic of Milan, Milan, Italy, in 1989.

He is currently a Full Professor with the Department of Electronics, Information and Bioengineering, Polytechnic of Milan. His research interests comprise: control and control-based design of computing systems and networks, object-oriented modeling, simulation and control, automatic tuning of industrial controls, event-based control, and control education methods and tools.

Toni Mancini received the master's and Ph.D. degrees in computer engineering from Sapienza University, Rome, Italy, in 2001 and 2005, respectively.

He is currently a Full professor with the Department of Computer Science, Sapienza University of Rome, Rome, Italy. His research interests comprise: artificial intelligence, formal verification, cyber-physical systems, control software synthesis, systems biology, smart grids.



Leonardo Picchiami received the master's degree in computer science from Sapienza University, Rome, Italy, in 2021. He is currently working toward the Ph.D. degree in computer science with the Sapienza University of Rome, Rome, Italy.

His research interests comprise: artificial intelligence, formal verification, cyber-physical systems, control software synthesis, systems biology.

Enrico Tronci received the master's degree in electrical engineering from Sapienza University, Rome, Italy, in 1987, and the Ph.D. degree in applied mathematics from Carnegie Mellon University, Pittsburgh, PA, USA, in 1991.

He is currently a Full Professor with the Department of Computer Science, Sapienza University of Rome, Rome, Italy. His research interests comprise: formal verification, model checking, system level formal verification, hybrid systems, embedded systems, cyber-physical systems, control software synthesis, smart grids, systems biology.