CHALMERS
UNIVERSITY OF TECHNOLOGY

How Useful is Learning in Mitigating Mismatch Between Digital Twins and Physical Systems?

Citation for the original published paper (version of record):
Cronrath, C., Lennartson, B. (2024). How Useful is Learning in Mitigating Mismatch Between Digital Twins and Physical Systems?. IEEE Transactions on Automation Science and Engineering, 21(1): 758-770. http://dx.doi.org/10.1109/TASE.2022.3231386

How Useful is Learning in Mitigating Mismatch between Digital Twins and Physical Systems?

Constantin Cronrath, *Student Member, IEEE*, and Bengt Lennartson, *Fellow, IEEE*

***Abstract*—In the control of complex systems, we observe two diametrical trends: model-based control derived from digital twins, and model-free control through AI. There are also attempts to bridge the gap between the two by incorporating learning-based AI algorithms into digital twins to mitigate mismatches between the digital twin model and the physical system. One of the most straightforward approaches to this is direct input adaptation. In this paper, we ask whether it is useful to employ a generic learning algorithm in such a setting, and our conclusion is "not very". We denote an algorithm to be more useful than another algorithm based on three aspects: 1) it requires fewer data samples to reach a desired minimal performance, 2) it achieves better performance for a reasonable number of data samples, and 3) it accumulates less regret. In our evaluation, we randomly sample problems from an industrially relevant geometry assurance context and measure the aforementioned performance indicators of 16 different algorithms. Our conclusion is that blackbox optimization algorithms, designed to leverage specific properties of the problem, generally perform better than generic learning algorithms, once again finding that "there is no free lunch".**

***Note to Practitioners*—Digital twins have the potential to improve productivity and quality in complex systems such as manufacturing systems. Their impact on system performance hinges on the accuracy of their digital models around the system's operating points. Difficult to measure phenomena, such as wear and tear of equipment, however, may cause a mismatch between the digital twin model and the physical system. In this paper, we formalize this problem and compare 16 potential solution strategies under practical aspects. We argue that readily available off-the-shelf blackbox optimization algorithms may prove more useful for this problem, than more recent learning-based approaches. Specifically, gradient-based algorithms will perform best in systems with high-dimensional, continuous, and non-linear performance functions – even in the presence of white measurement noise.**

***Index Terms*—Smart manufacturing, cyber-physical systems, digital twins, learning (artificial intelligence), adaptive optimization, blackbox optimization.**

I. Introduction

THE digital twin concept refers to simulation models of such high fidelity that they nearly could be called exact copies (twins) of the physical space in the digital space. Through real-time sensor updates, these digital models are kept congruent with their physical counterpart at any point in time [1]–[3]. Digital twins are regarded as a core enabling technology of smart and autonomous manufacturing [4] to improve productivity, optimality, and efficiency [5], [6]. Indeed, optimization algorithms with sufficient computational resources may take advantage of the twin's accurate digital model to prescribe optimal control inputs for the physical system. We consider this emerging technology as the model-based approach to controlling complex systems.

Despite the large potential of digital twins, the concept is not without its challenges. Specifically, the "twinning" or "mirroring" property of the digital twin concept may be difficult to achieve in changing systems, and is, thus, often neglected in descriptions of digital twins in the literature [2], [7]. Model-system mismatches are commonly solved in the control and automation field through system identification [8]. This technique may too be applicable in the digital twin context. However, identifiability of the model parameters may be hindered by simulation model classes of still too low capacity to faithfully represent reality. Furthermore, observability may not be given (c.f. [3], [9], [10]), meaning some aspects of the physical system may not be measured by sensors, e.g. because of the unavailability of suitable measurement equipment. In such cases, the digital space will not mirror the physical space and the digital twin concept may, hence, not fully realise its potential. Yet, convergence [4], [9], [11] and evolution [5], [12], [13] of digital model and physical system are postulated key properties of digital twins. Without those properties, model-based optimality of smart and autonomous manufacturing is debatable.

Artificial intelligence (AI) and more specifically machine learning are frequently proposed as the solution to the previously described model-system mismatch problem, see [5], [9], [11], [14]. We refer to this as the model-free approach to controlling complex systems. Often, it remains rather ambiguous in the literature, what is meant by AI and machine learning, and how these methods may solve said problems. In this study, we consider an algorithm to be a learning-based algorithm, if: 1) it estimates a *persistent, global* model of the system's performance function from sampled data, 2) it uses this model to generate new candidate points to explore, and 3) it actively seeks to act optimally as measured against the system's performance function. Our definition is in line with the notion of a rational, intelligent agent in the field of AI and the reinforcement learning branch of machine learning [15].

An alternative solution approach is adaptive – or real-time – optimization, which deals with optimal control under model-

Manuscript received 23 December 2021; revised 15 August 2022; accepted 9 December 2022. This article was recommended for publication by Associate Editor W. Guo and Editor X. Xie upon evaluation of the reviewers' comments. This work was supported in part by the Swedish Foundation for Strategic Research through the Smart Assembly 4.0 Project within the Winquist Laboratory and in part by Vinnova under the ITEA3 "AITOC" (www.itea4.org/project/aitoc.html) Project. (Corresponding author: Constantin Cronrath.)

C. Cronrath, and B. Lennartson are with the Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden {cronrath, bengt.lennartson}@chalmers.se

system mismatches [16]. Chachuat, Srinivasan, and Bonvin [16] divide adaption strategies for real-time optimization into three categories: the two-step approach (repeated model parameter identification and optimization), the modifier approach (identifying an error model), and direct input adaptation. Direct input adaptation turns the optimization problem into a feedback control problem, in which the control inputs are optimized online. It is the category that lends itself most readily to applying generic learning-based algorithms and has been shown to be feasible for instance in [17]–[21]. Efficient use of each data sample is a key requirement in this online setting, since each sample may represent for instance a physical product with a tangible cost.

In this paper, we are interested in the mitigation of model-system mismatch through learning-based algorithms. The mismatch may arise from omitting a model calibration through system identification, or because the system identification procedure falls short of true convergence due to the aforementioned issues. Direct input adaptation is then a suitable remaining approach. Generic reinforcement learning [22] and blackbox optimization [23] algorithms may both readily be used in this context. Blackbox optimization is a mature field, which concerns the optimization of unknown functions that can only be probed through sampling. Blackbox optimization offers off-the-shelf solutions, which may prove beneficial for practitioners, and is thus a valid benchmark. Therefore, we ask:

> *How useful is learning when compensating for model mismatch in a digital twin?*

To answer this question, we compare 16 well-established learning-based and blackbox optimization algorithms on a problem of industrial relevance. More concretely, we base our experiments on a geometry assurance task in the assembly of automotive sheet metal parts, previously described in [24]. Here, adjustable locators of the assembly fixture position two or more sheet metal parts for subsequent joining through welding or clinching. The locators of the physical fixture may deviate from their digital twin due to wear and tear, damages, or effects of changing process parameters. We aim to optimize geometric quality by compensating for this system-twin mismatch through direct input adaptation. In that, we make the following contributions:

- we demonstrate the application of learning-based AI algorithms in the context of digital twins;
- we show how direct input adaptation can compensate for mismatches between physical and digital space;
- we provide comprehensive experimental results on 16 well-established algorithms (BOBYQA, Bayesian Optimization, Conjugate Gradients, COBYLA, DDPG, DIRECT-L, Differential Evolution, HOO, L-BFGS-B, MMA, Nelder-Mead, PRAXIS, Powell's Method, SLSQP, SPSA, and Subplex);
- we find that gradient-based blackbox optimization is better suited to compensate for system-twin mismatch of high-dimensional, continuous, and smooth performance functions than learning-based algorithms in the direct input adaptation setting.

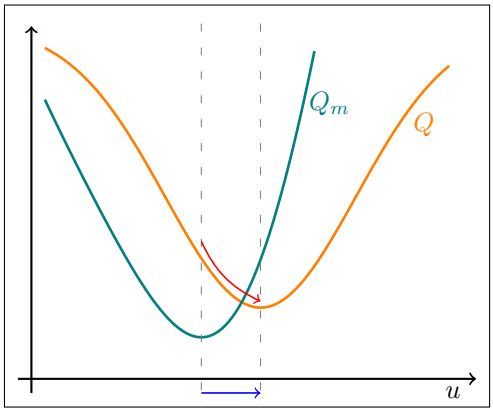


Fig. 1. An illustration of the problem under consideration. While Q_m is known and may be used for identifying the location of a minimum, disturbances, inaccuracies, or errors may cause this location to be shifted in the physical system. Using the minimizer of Q_m as a starting point, we are interested in searching efficiently for a minimum of the true but unknown Q.

In Sec. II, we specify our problem further by giving mathematical properties of the optimization problem, as well as by describing the system architecture, and the particular test cases for experiments. From that, we derive usefulness metrics in Sec. III and outline our test method. Sec. IV briefly introduces the algorithms used in the experiments. Results are reported in Sec. V, and in Sec. VI a discussion of these results is presented. Conclusions are documented in Sec. VII.

II. Problem Formulation

Following a concise problem formulation, the system architecture, our particular application cases, and their disturbance models will now be further explained in detail. Performance metrics for the evaluation of solution algorithms are discussed in Sec. III.

In short, we wish to compensate for system-twin mismatches through direct input adaptation in the large class of high-dimensional, nonlinear, and non-convex, but continuous and smooth static performance functions. While a model, or digital twin, of the system's performance function may exist, we assume it does not fully capture all aspects of reality, leading to a shift in the location of the minimal point. In this paper, we only consider shifts (or offsets) that are constant in nature, which we model as additive input disturbances. We assume the system-twin mismatch to be small enough such that the twin may be used to identify the neighborhood of the system's global optimum. Compensating the mismatch then takes the form of an online search for a local minimum. Fig. 1 illustrates the problem. We further consider the two cases, where measurements of the performance are 1) without error and 2) corrupted by white output noise.

A. System Architecture

Borrowing notation from the field of reinforcement learning [22], we denote the performance, or objective, function of the system with Q. Q may represent any such criterion as financial costs or measured quality defects of a control input u, defined over a system with possibly very large state

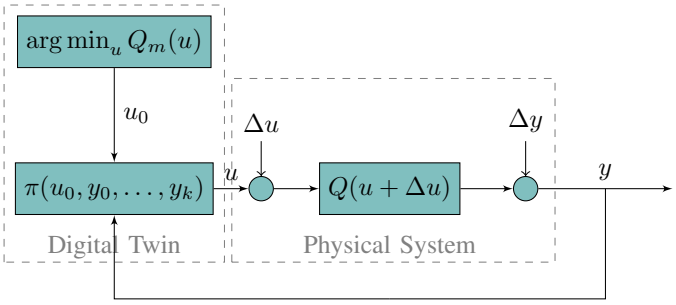


Fig. 2. A block diagram of the system under consideration. We are interested in control policies π capable of determining a high-dimensional control input u, that minimizes a performance or objective function Q of the physical system. A model, or digital twin, of this function is accessible, that may be queried for its minimizer u_0. While we assume this model to be sufficiently accurate in the neighborhood of the minimum, the model is assumed not to capture input disturbances Δu, and output disturbances Δy.

space. For simplicity, we neglect the explicit evolution of the system state and only observe the system's final performance under a control input u. For brevity, we may refer to this performance function Q as "the system". A model, or digital twin, of the true function Q is available and denoted with Q_m. Combined with a suitable optimization algorithm, this digital model Q_m is used to determine an initial control input u_0. However, due to inaccuracies of the model and additive input disturbances Δu, this initial control input u_0 is assumed to be *not* the minimizer u^* of the true performance function Q (see Fig. 1). Further, we consider also the case where the measurements y of Q are corrupted by additive white noise, that is $y = Q(u+\Delta u)+\Delta y$. In this context, we are interested in suitable control policies π that can adapt u_0 based on scalar measurements y to compensate for input disturbances Δu and slight, local model inaccuracies. Learning-based methods, such as reinforcement learning, or blackbox optimization may be such suitable control policies π. A corresponding block diagram of this setup is given in Fig. 2.

B. Case: Geometry Assurance System

More concretely, we base our experiments on a geometry assurance task in the assembly of automotive sheet metal parts. Here, two or more sheet metal parts are fixed in position for subsequent joining through spot welding or clinching. In that, the locators of the fixture may be individually adjusted to optimize the resulting geometric quality. A maladjusted locator can further contribute to nonlinear elastic deformations of the sheet metal parts in the joining process. Upon ejection from the fixture, the newly formed assembly springs back into its new unconstrained geometry. Its geometric quality can then be measured through e.g. scanning. We encapsulate this whole – highly nonlinear – process (positioning, clamping, joining, releasing, scanning) in $Q(u)$ as a function of locator adjustments u. Depending on the case, the dimension d of the control input space u varies between 12 and 20 in this study. Fig. 3 details our three application cases. The geometric quality $Q(u)$ of the finished assembly is computed by the commercial software tool *RD&T*[1] through a sequence of detailed finite element method calculations on geometry meshes of several thousand points. These points may be considered the system's internal state, captured by the performance measurement in $Q(u)$. In particular, $Q(u)$ describes the Root Square Mean Error (RSME) between the measured part geometry after assembly and its nominal geometry. Although $Q(u)$ has the ideal optimal target of 0.0, in practice, locator adjustments u alone may not be sufficient to drive the RSME to 0.0, which is why the optimal target may be non-zero. $Q(u)$ is highly non-linear and non-convex, and its high-dimensional optimization manifold has very many local optima in the search space $\mathbb{R}^d_{[-2.0,2.0]}$. This geometry assurance task is hence an interesting optimization problem, representing a larger class of industrially relevant problems.

In the previously described Fig. 2, we introduce two additive disturbances: an input disturbance Δu, and an output disturbance Δy. According to Wärmefjord, Söderberg, Lindau, Lindkvist, and Lorin [26], several factors affect the geometric quality of sheet metal assemblies. Of those, we can model with Δu: fixture deviations; clamping force and clamping stiffness deviations; and joining point, force, and tool variations. Moreover, the additive input disturbance may simulate mechanical deterioration, friction, or insufficient model calibration in general. With Δy we can capture: part geometry variations, material property variations, and generally any other disturbance propagating through the process Q into y.

For simplicity, we restrict ourselves here to constant input disturbances Δu and white noise output disturbances Δy. This is a reasonable limitation since it still encompasses a range of industrially relevant error cases, such as damaged fixtures or tools, that may not be represented in the digital model. Slowly changing input disturbances may also be regarded as a sequence of differing constant disturbances.

While our particular application cases are fairly specific, they still represent a much larger class of practically relevant problems. We wish to find the optimum of some unknown, high-dimensional, non-linear, non-convex, smooth, and continuous performance function Q defined for some system, for which we have a sufficiently good model Q_m that can be used for an initial guess but does not capture a variety of disturbances. To find the true optimum of Q, it is thus necessary to sample the system repeatedly at various points of the input space. The following section describes how we compare the usefulness of various algorithms for doing that.

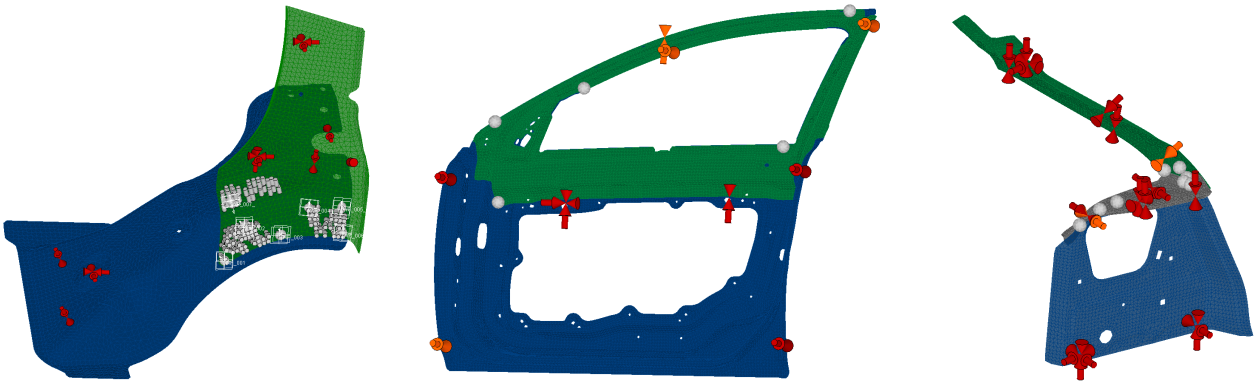
III. Hypothesis and Method

We wish to answer the research question: *How useful is learning when compensating for model mismatch in a digital twin?* For a meaningful answer, our notion of *usefulness* is first specified. Subsequently, we formulate a hypothesis and then introduce our experimental procedure.

A. Performance Metrics

We denote an algorithm to be more useful, than another algorithm, based on three aspects:

[1]Available from www.rdnt.se. RD&T's underlying theory and its usage as a digital twin are described for instance in [24]–[26].



(a) Tail, $d_u = 12$ (b) Door, $d_u = 19$ (c) Pillar, $d_u = 20$

Fig. 3. The three use cases for our experiments: (a) shows a sub-assembly of a car tail. Its two parts are positioned in space by twelve locators (shown in red). (b) shows the frame of a car door. Its two sheet metal parts are positioned in space by 19 locators. (c) shows a pillar of a car, consisting of three parts held in place by 20 locators. A digital twin computes the geometric quality after the assembly process (positioning, clamping, joining, releasing, scanning) based on several detailed Finite Element Method calculations, including elastic deformations and springback under the assembly forces. The final geometric quality may be influenced by adjusting each single locator along its axis. In this study, we induce a mismatch between the digital twin and the simulated system through constant, additive input disturbances and white-noise output disturbances. These disturbances may for instance represent wear and tear or process variations.

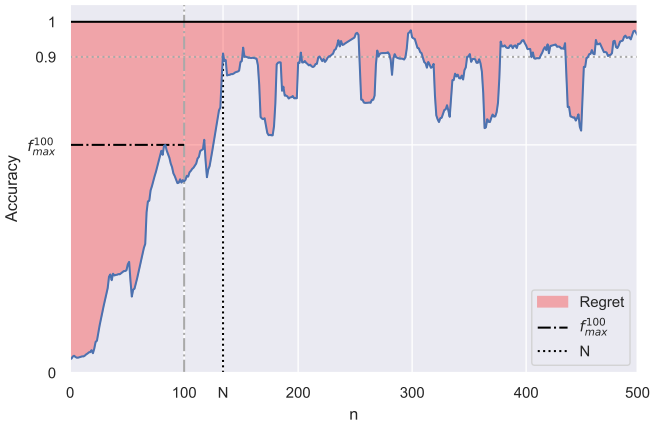


Fig. 4. An illustration of the applied usefulness metrics. The blue line depicts the progress of the adaption algorithm towards the optimal accuracy of 100%. We are interested in the required samples N to reach 90% accuracy, the maximum accuracy in the first 100 samples f^{100}_{max}, and the accumulated regret, if the optimal policy was known from the start (red area).

1) *sample efficiency* – it requires fewer data samples to reach a desired minimal performance,
2) *best in X* – it achieves a more accurate performance in a reasonable number of data samples, and
3) *regret* – it accumulates less regret when compared to the optimal policy.

All three metrics are illustrated in Fig. 4 and are further detailed in the following.

1) Sample efficiency: Since each sample represents a real physical product with associated costs in our case, we wish the sampling of the unknown performance function Q of the physical system to be as efficient as possible. That is, in the selection of the control policy π, algorithms, finding the true optimum in fewer samples, are preferred. To that end, we define – in accordance with [23] – the accuracy of the optimization at sample $k \in \{1, \dots, K\}$ as

$$f^k_{acc} = \frac{Q(u^0) - Q(u^k)}{Q(u^0) - Q(u^*)} \ , \tag{1}$$

where u^0 is the initial control input derived from the digital twin, and u^* is the true local optimum of the physical system. In practice, any accuracy close to 1.0 might be deemed sufficiently good, and thus we decide on

$$f^N_{acc} > 1 - \tau \tag{2}$$

as stopping criterion (with τ being the tolerance level, and N the value of k at which (2) becomes true). In our experiments, we set $\tau = 0.1$, meaning we consider the physical optimum to be found with sample N if 90% of the improvement potential was realized. Algorithm A is then deemed more useful than algorithm B, iff $N_A < N_B$.

2) Best in X: Besides being sample-efficient in finding the physical optimum, a good accuracy after a fixed number of samples may be useful. This may be for instance the case when a fixed calibration budget is granted, after which the best-found input is used for production. We set the sampling budget for this aspect arbitrarily to 100 parts and compare the algorithms based on

$$f^{100}_{max} = \max\{f^1_{acc}, f^2_{acc}, \dots, f^{100}_{acc}\} \ . \tag{3}$$

3) Regret: In addition, we prefer algorithms that interfere with the process to a lesser extent. This criterion is encapsulated in the metric of regret, expressed as

$$R = \sum_{k=1}^{K} \frac{Q(u^k) - Q(u^*)}{Q(u^0) - Q(u^*)} \ . \tag{4}$$

Regret is the sum of the differences between the algorithm's performance, seeking for the optimal input, and the performance if the optimal input was known already from the start. An algorithm that is sampling the physical system in many *"bad"* areas, potentially causes quality defects and

accumulates more regret. However, we may want to allow for a few quality defects if the optimal policy is identified quicker because of that. The measure of regret is capable of capturing this trade-off. In addition, we normalize the regret to enable comparison across test cases.

B. Hypothesis

With a clear understanding of the performance metrics used to determine usefulness, we now state our main hypothesis:

> It is useful to employ a learning-based algorithm when compensating for model mismatch in a digital twin.

We expect to see a learning-based algorithm outperform most, or all other algorithms, in one, or more, of the three previously discussed metrics of usefulness. If that is not the case, we are willing to reject our hypothesis.

C. Experimental Method

To test our hypothesis, we use a single-factor, fixed effect experimental design (c.f. [27]), for which we assume that model and system only differ by the unobserved input and output disturbances. This assumption allows for efficient experimentation on the digital model only. We create a representative (of the real-world) test set by randomly sampling part geometries, see Fig. 3, an input disturbance, and a standard deviation of the white-noise output disturbance for every second test. The sampling domains of the parameters of our application are listed in Tab. I. Randomization is introduced to reduce unwanted effects of the application parameters on our experimental results (c.f. [27]).

TABLE I
PARAMETERS OF OUR APPLICATION USED TO GENERATE A TEST SET FOR OUR EXPERIMENTS. THE DIMENSIONALITY d OF THE OPTIMIZATION PROBLEM IS GIVEN BY THE CASE, SEE FIG. 3, WITH $d \in \{12, 19, 20\}$.

Nuisance	Symbol	Domain
Case		$\{door, pillar, tail\}$
Part geometry index		$\{1, \ldots, 25\}$
Input disturbance	Δu	$\mathbb{R}^d_{[0.0,0.5]}$
White-noise output disturbance	$\sigma_{\Delta y}$	$\mathbb{R}_{[0.0,0.007]}$

For each case and each part within the corresponding assembly, geometry scans, consisting of several thousand measurement points, of 25 manufactured parts are available to us. Hence, we randomly choose one of 25^2 (25^3 for case *pillar*) possible assembly geometries for experiments. Next, we use a local optimization algorithm (SLSQP in our experiments) to identify u^* and Q^* of the physical system, which is emulated here by the digital model. The algorithm receives $u^0 = \mathbf{0}^d$ as the starting point and is run to full convergence.

For up to ten times, an additive input disturbance Δu is sampled uniformly from $\mathbb{R}^d_{[0,0.5]}$. We then check for local convexity by sampling Q along the euclidean distance between u^* and $u^* + \Delta u$. Based on this, we delimit our experiments to optimization problems that are locally convex. For sufficiently smooth functions Q and small input disturbances, this is a reasonable delimitation and circumvents the convergence to other local minima, which may corrupt our analysis. For stochastic optimization problems, a standard deviation $\sigma_{\Delta y}$ for the white-noise output disturbance Δy is uniformly sampled at the beginning of the experiments: $\sigma_{\Delta y} \in \mathbb{R}_{[0.0,0.007]}$. The upper bound on $\sigma_{\Delta y}$ is chosen arbitrarily to a small value, which represents about 5% of the average Q^* across all cases. During experiments, the measurements y are thus computed by $y = Q(u + \Delta u) + \mathcal{N}(0, \sigma_{\Delta y})$, where $\mathcal{N}(0, \sigma_{\Delta y})$ is a normal distribution with 0 mean and standard deviation $\sigma_{\Delta y}$. The input and output disturbances hence vary in size across experiments to mitigate potential systematic errors related to these problem parameters.

Once the optimization problem is defined accordingly, we let a range of learning-based and non-learning-based blackbox optimization algorithms (see Sec. IV) solve the problem. All algorithms have access to $u^0 = u^*$ but have no knowledge of Δu or Δy. An optimal policy is thus found if $\pi = u^* - \Delta u$. To find a policy close to that, we allow for up to $K = 500$ samples, since this seems to be a reasonable number for many practical applications. We record the previously described usefulness metrics and report them in Sec. V.

IV. ALGORITHMS

This section briefly introduces the algorithms used in our experiments. In total, 16 well-established algorithms are compared. We categorize the algorithms according to: global, local derivative-free, local gradient-based, and learning-based algorithms. This categorization is not completely accurate, since some learning-based algorithms are also global optimization algorithms. Furthermore, no differentiation in the categorization is made between stochastic and deterministic algorithms. All global and learning-based algorithms in this paper, as well as SPSA, are stochastic algorithms, which use random variables within their solution procedure. The selection of algorithms is discussed further in Sec. VI.

A. Global

Algorithms that do not sample points in close vicinity of the best-known point, but sample the whole search space more widely are denoted as global algorithms in this paper.

a) DIRECT-L: DIRECT-L is a locally biased version of the DIRECT (DIviding RECTangles) algorithm [28]. DIRECT successively divides the search space into smaller and smaller rectangles. Each rectangle is evaluated at its center. The size of the rectangle and the function value at the center determine which rectangles are further subdivided. DIRECT-L adapts the subdivision procedure to favor local exploration around good function values over exploration in sparsely sampled areas of the input space.

b) Differential Evolution: Differential Evolution [29] is an evolutionary algorithm that computes new input candidates from four other members of the population. If the candidate has a better or equal fitness value, it replaces one of the members. It is a stochastic direct search algorithm.

B. Local Gradient-Based

Algorithms that estimate the gradient of the objective function, for the purpose of minimizing it in some hill descending manner, are grouped into local gradient-based algorithms.

a) Conjugate Gradients: The conjugate gradient algorithm [30] minimizes a $d-$dimensional function sequentially along d linearly independent (conjugated) directions. To that end, the gradient of the function is approximated and a line search is performed in each of the d directions.

b) L-BFGS-B: The Limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm with Bound constraints (L-BFGS-B) [31] is a quasi-Newton method that approximates the Hessian of the objective function from first derivatives. First derivatives are estimated through two-point finite differences in our experiments. The limited memory variant of the BFGS algorithm represents the Hessian implicitly from past gradient differences. When extended with bound constraints, gradient projection is used to determine the active constraints set.

c) SLSQP: Sequential Least Squares Programming (SLSQP) [32] is a quasi-Newton method that replaces the quadratic programming formulation of the Newton direction with a non-negative least squares problem. As such, it minimizes repeatedly quadratic approximations of the objective function.

d) MMA: The Method of Moving Asymptotes (MMA) [33] uses conservative convex approximations by rational functions (defining the moving asymptotes) with quadratic penalty terms (enforcing a convex trust region). Conservatism is achieved by adapting the approximation, if the approximation predicts a better function value at a new candidate point than is sampled from the system. No line search is needed in MMA because of its conservatism.

e) SPSA: Simultaneous perturbation stochastic approximation (SPSA) [34], [35] estimates the gradient of the objective function from two data samples. Regardless of the problem's dimension, all input variables are perturbed randomly around the current mean estimate. The perturbation is usually generated from a symmetric Bernoulli distribution. Due to its stochastic nature, SPSA is suitable for objective functions that are corrupted by noise.

C. Local Derivative-Free

Algorithms that do not rely on gradient estimates, but search locally around the currently best-known sampling point are categorized as local derivative-free algorithms.

a) Powell: Powell's method [36] is a variation of the Conjugate Gradients algorithm discussed above, in which no derivatives are taken. Instead, the difference between the current and new best point becomes a conjugate direction and replaces one of the prior directions.

b) PRAXIS: The search directions in Powell's method may become linear dependent with a negative effect on convergence. PRAXIS [37] is a variation of Powell's method that regularly restores linear independence by setting the conjugate directions to be the principal axes of a quadratic approximation of the objective function.

c) Nelder-Mead: Nelder-Mead [38] is a direct search algorithm that constructs a simplex from $d+1$ sampled points (with d being the dimensionality of the problem). The centroid of the d-best points is used in determining a new sample point. The new point replaces the worst point in the simplex. If the new point is better than the best point so far, the simplex may be expanded. If it is worse than the second worse point so far, the simplex may be contracted. The algorithm stops when some termination criterion, such as a small standard deviation of the function values of the simplex, is met.

d) Subplex: Subplex [39] is a variant of Nelder-Mead, that uses Nelder-Mead to search a sequence of sub-spaces of the problem. Sub-spaces need to be orthogonal to each other and of lower dimensionality than the original search space. Subplex leverages the efficiency of Nelder-Mead in lower-dimensional problems to solve problems in higher dimensions.

e) COBYLA: Constrained Optimization BY Linear Approximation (COBYLA) [40] is an enhancement of Nelder-Mead that introduces explicit handling of constraints. Objective function and constraints are interpolated by linear approximation at the vertices of a simplex of $d+1$ points. COBYLA thus solves sequentially multiple linear programs. To account for approximation errors, the algorithm enforces a trust region on the new sample point.

f) BOBYQA: Bound Optimization BY Quadratic Approximation (BOBYQA) [41] uses a quadratic approximation of the objective function by fitting typically $2d+1$ points (with d being the dimensionality of the problem). A new candidate point is obtained by minimizing the quadratic approximation within a trust region and the box constraints. The new point replaces one of the $2d+1$ points. In updating the quadratic approximation, the Frobenius norm of the induced change of its second derivative is minimized.

D. Learning-Based

For an algorithm to be considered a learning-based algorithm in this study, it must: 1) estimate a *persistent, global* model of the system's performance function from sampled data, 2) use this model to generate new candidate points to explore, and 3) actively seek to act optimally as measured against the system's performance function.

a) Discretized Multi-Armed Bandit Algorithms: Our problem formulation is most similar to the multi-armed bandit problem in the learning literature. The multi-armed bandit is essentially a stateless reinforcement learning problem, in which the learner tries to find the arm (control input) that yields the same or higher pay-off in expectation than any other arm in a finite set of arms. To submit our continuous control problem to any of the available multi-armed bandit algorithms, one would need to discretize its input space (c.f. [42]). Due to the curse of dimensionality, however, this strategy is infeasible, because of the high dimensionality of the problem at hand. If one would discretize $\mathbb{R}^d_{[u^*-0.5,u^*+0.5]}$ in pieces of width 0.1 in each direction, one would get 10^d discrete arms to explore – with $d \in \{12, 19, 20\}$. The large class of multi-armed bandit algorithms is, thus, ruled out as infeasible for our high-dimensional, continuous problem.

b) HOO: Hierarchical Optimistic Optimization (HOO) [43] is a multi-armed bandit algorithm with an adaptive discretization mechanism. The algorithm uses a tree structure to incrementally estimate the mean of the objective function. The tree-structure partitions the input space finer and finer around potential optima, whereas areas of less interest remain partitioned coarsely. Similar to Bayesian optimization, HOO maintains an optimistic bound on the mean-objective function proportional to the partition size. The tree structure was deliberately chosen to enable also its application to continuous input spaces.

c) Bayesian Optimization: Bayesian Optimization [44] is a class of algorithms that employ Bayesian statistics to model a distribution over credible functions to be optimized. In the non-parametric version, Gaussian processes are used as model. In each iteration, the next sampling point is determined by optimizing an acquisition function that takes the mean and uncertainty of the distribution into account.

d) DDPG: Deep deterministic policy gradients (DDPG) [45] is a basic deep reinforcement learning algorithm of the actor-critic class. It maintains one neural network for the estimation of the Q-function and one network for the approximation of the mean of the continuous control policy. The input space is explored by means of correlated, additive white noise around the policy. Applied to the static bandit context, DDPG reduces to estimating the static Q-function through a neural network and taking gradient steps along it. Considering our special application context, we pre-train the policy network to output u^0 initially.

E. Implementation Details

In our experiments, we rely on the Python implementation of SciPy [46] for the algorithms: Differential Evolution, Conjugate Gradients, L-BFGS-B, SLSQP, Powell, Nelder-Mead, and COBYLA. For DIRECT-L, MMA, PRAXIS, Subplex, and BOBYQA, we use the implementation of nlopt [47]. Further, we use the scikit-optimize [48] implementation of Bayesian Optimization. The remaining ones, SPSA, HOO, and DDPG, were implemented by the authors. We keep all hyperparameters of the algorithms set to their proposed defaults, with the exception of the maximum number of function evaluations ($K = 500$) and – if possible – the bounds on u to $\mathbb{R}^d_{[u^*-0.5,u^*+0.5]}$. Furthermore, other hyperparameters of SPSA were set according to the rules outlined in [35]. Hyperparameters of HOO and DDPG were tuned by Bayesian Optimization.

V. Results

The central research question of this study is: *How useful is learning when compensating for model mismatch in a digital twin?* We denote an algorithm to be more useful than another algorithm, based on three aspects:

1) *sample efficiency* – it requires less data samples to reach a desired minimal performance (see (2)),
2) *best in* 100 – it achieves a more accurate performance in a reasonable number of data samples (see (3)), and
3) *regret* – it accumulates less regret when compared to the optimal policy (see (4)).

To that end, we sampled 198 problem instances from a geometry assurance context (see Sec. III) and solved those with the 16 algorithms discussed in Sec. IV. Our results indicate that it is *not* useful to employ a learning-based algorithm when compensating for model mismatch in digital twins under the direct input adaptation scheme. Instead, it appears to be most useful to employ a local gradient-based algorithm, such as SLSQP in deterministic settings or SPSA in stochastic settings. The details of our analysis are provided in the following.

A. Overview of Results

In total, we conducted 99 experiments for each of the deterministic and stochastic settings. For both settings, each application case (door, pillar, tail) makes up a third of the experiments. We consider nine randomly sampled and unique assemblies for *door*, eleven for *pillar*, and eleven for application case *tail* in our experiments. Tab. II summarizes the details of our experiments.

TABLE II
OVERVIEW OF CONDUCTED EXPERIMENTS. 198 EXPERIMENTS WERE CONDUCTED IN TOTAL, WHICH WERE EQUALLY DISTRIBUTED ACROSS ALL THREE APPLICATION CASES AND OUTPUT NOISE SETTINGS. THE NUMBER OF UNIQUE ASSEMBLIES AND THE AVERAGE NORM OF THE INPUT AND OUTPUT DISTURBANCES ARE GIVEN.

Case	Setting	#Exp.	Assemblies	$\|\overline{\Delta u}\|_2$	$\|\overline{\Delta y}\|_2$
Tail	det.	33	11	1.02	0.0
	sto.	33	8	1.02	3.87×10^{-3}
Door	det.	33	8	1.27	0.0
	sto.	33	9	1.23	3.83×10^{-3}
Pillar	det.	33	11	1.27	0.0
	sto.	33	8	1.27	3.59×10^{-3}

Tab. III lists the number of problems each algorithm could solve in the deterministic and the stochastic setting, as well as the 25%, 50%, and 75% quantiles of the number of data samples needed to solve the problems. We consider the problem to be solved when $f^N_{acc} > 0.9$ within 500 data samples. All algorithms were able to solve at least some of the deterministic problems, while most of the gradient-based algorithms (CG, L-BFGS-B, MMA, PRAXIS, and SLSQP) failed to solve any of the stochastic problems. COBYLA and Nelder-Mead only solved one stochastic problem each. In contrast, gradient-based SLSQP solved 97 out of 99 deterministic problems. In the stochastic setting, Bayesian Optimization and SPSA performed best with 87 out of 99 solved problems. Both algorithms also performed well in the deterministic case. Our usefulness criteria reflect these overall results.

B. Sample Efficiency

The first criterion to determine usefulness is the number of data samples needed to reach a desired minimal performance. We consider a problem instance to be solved by the respective algorithm at sample N, once the accuracy exceeds a desired level: $f^N_{acc} = \frac{Q(u^0)-Q(u^N)}{Q(u^0)-Q(u^*)} > 1-\tau$. In our analysis, we choose $\tau = 0.1$. This essentially means that we consider

TABLE III
NUMBER OF PROBLEMS SOLVED BY ALGORITHM IN THE DETERMINISTIC AND THE STOCHASTIC SETTING, AS WELL AS THE NUMBER OF DATA SAMPLES N NEEDED TO REACH $f_{acc}^N > 0.9$. 25%, 50%, AND 75% QUANTILES OF N ARE GIVEN. DASHES INDICATE MISSING NUMBERS DUE TO NO SOLUTION. THE BEST NUMBERS ARE PRINTED BOLD-FACED.

	deterministic				stochastic			
	#	N			#	N		
		25%	50%	75%		25%	50%	75%
DIRECT-L	9	–	–	–	12	–	–	–
Diff. Evolution	64	212	379	–	59	215	380	–
CG	94	104	156	231	–	–	–	–
L-BFGS-B	91	80	104	210	–	–	–	–
SLSQP	**95**	**55**	**67**	**110**	–	–	–	–
MMA	46	147	–	–	–	–	–	–
SPSA	91	87	174	337	**87**	**70**	**158**	**294**
Powell	78	187	249	410	68	221	332	–
PRAXIS	66	88	180	–	–	–	–	–
Nelder-Mead	31	455	–	–	1	–	–	–
Subplex	88	109	144	298	68	103	193	–
COBYLA	68	93	238	–	1	–	–	–
BOBYQA	11	–	–	–	12	–	–	–
HOO	29	387	–	–	35	322	–	–
Bayesian Opt.	83	93	134	343	**87**	74	169	346
DDPG	14	–	–	–	16	–	–	–

the problem to be solved once 90% of the improvement potential has been realized. For the purpose of comparing different algorithms, we rank all algorithms on each problem instance by the number of function evaluations N needed to achieve $f_{acc}^N > 1-\tau$. The algorithm with the smallest number of function evaluations is ranked first, the one with the most is ranked last.

Fig. 5a summarizes the ranking of each algorithm across all *deterministic* problem instances (no output disturbance: $\sigma_{\Delta y} = 0$). The boxplot orders the algorithms by median rank. We observe that local gradient-based optimization algorithms perform best on the deterministic problems. The algorithm with the best median ranking by far is SLSQP, meaning it solved the most problems in the least number of function evaluations. The other gradient-based local optimization algorithms are also ranked highly (L-BFGS-B – second, CG – fourth, SPSA – fifth), with the exception of MMA – ranked eleventh. The global optimization algorithms DIRECT-L and Differential Evolution as well as the learning-based algorithms HOO and DDPG rank in the lower half of the table. A notable outlier is Bayesian Optimization, which is both a global and learning-based algorithm and yet achieves rank three.

Fig. 5b repeats the analysis, which is described in the previous two paragraphs, for *stochastic* problem instances. In these problem instances, an output disturbance is added to model part variation and any other variations that propagate through the assembly process. The output disturbance is white-noise with zero mean and a randomly sampled standard deviation. The output disturbance seems to be especially detrimental to most of the local optimization algorithms. SLSQP, PRAXIS, MMA, L-BFGS-B, CG do not solve any of the stochastic problems while COBYLA and Nelder-Mead only solve 1/99 and thus rank last. Stochastic algorithms (SPSA, Bayesian Optimization, Diff. Evolution, HOO, DDPG, DIRECT-L) generally perform better in the stochastic setting than in the deterministic setting. SPSA performs best across all algorithms. The stochastic approximation of the gradient is by design capable of handling noisy function evaluations. SPSA is closely followed by Bayesian Optimization on rank two. The other learning-based algorithms HOO and DDPG achieve an overall rank of six and seven out of 16 algorithms.

Overall, we observe that local, gradient-based algorithms are most useful in terms of sample efficiency – SLSQP for deterministic problems, and SPSA for stochastic ones. The learning-based algorithms HOO and DDPG rank for both problem types in the lower middle field without a clear edge over the other algorithms. Bayesian Optimization ranking highly is in both settings a notable exception.

C. Best Result after 100 Samples

The second criterion to determine usefulness is the best-achieved result after 100 samples. To that end, we compute the maximum accuracy in the first 100 data samples. All algorithms are then ranked by maximum accuracy in the first 100 samples on each problem instance. The algorithm with the highest accuracy is ranked first, the one with the lowest is ranked last.

Fig. 5c depicts a boxplot of each algorithm's ranking over all *deterministic* problem instances (no output noise). Similar to the previous analysis, SLSQP ranks best across all deterministic problems. Accordingly, most global and learning-based optimization algorithms are found in the lower half of the figure (HOO, Diff. Evolution, DDPG, DIRECT-L).

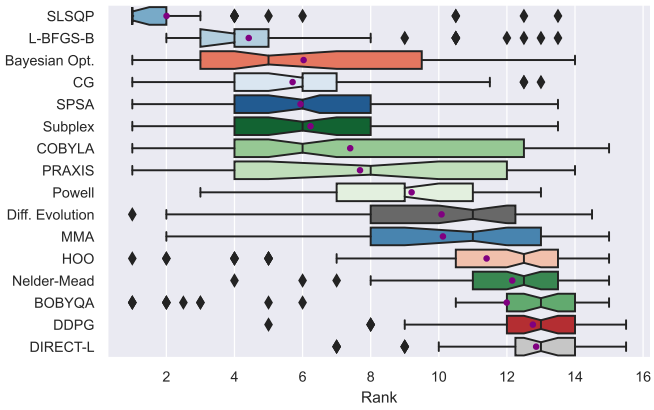
Fig. 5d depicts a boxplot of each algorithm's ranking over all *stochastic* problem instances (with output noise). Once again SLSQP fails in the stochastic setting and ranks last. SPSA and Bayesian Optimization rank best in this setting. Notably, SPSA enjoys a higher rank for the 75-percentile.

Overall, the results for the best in 100 metric are similar to the sample efficiency metric. In the deterministic setting, local algorithms dominate the upper half of the ranking. Gradient-based local algorithms perform below average in the stochastic setting. Bayesian Optimization and SPSA can both be considered outliers of their groups, performing well in both settings.

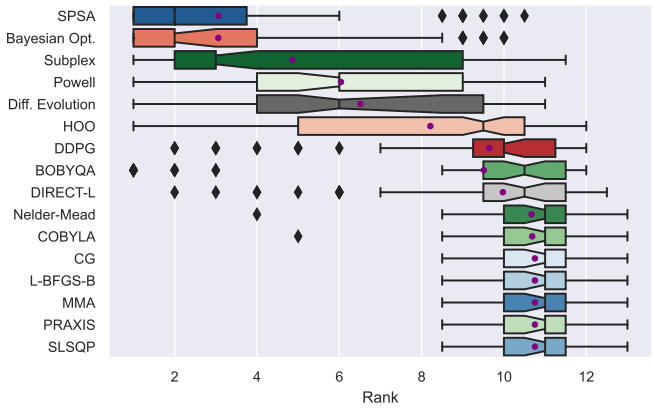
D. Regret

The third metric to determine usefulness is regret (see Sec. III). We compute regret as $R = \sum_{k=1}^{500} \frac{Q(u^k)-Q(u^*)}{Q(u^0)-Q(u^*)}$. All algorithms are then ranked by minimum regret on each problem instance. The algorithm with the least regret is ranked first, the one with the highest is ranked last.

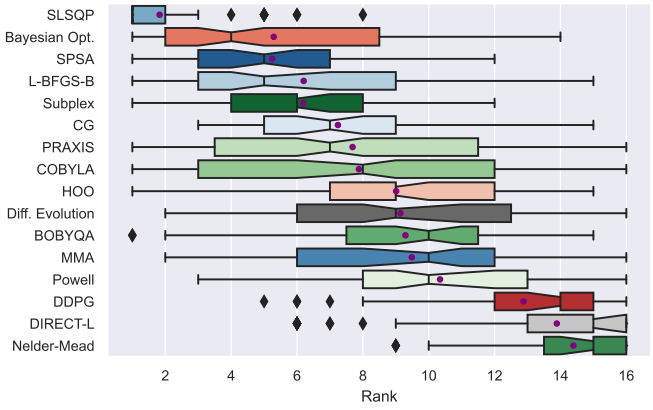
Fig. 5e depicts a boxplot of each algorithm's ranking over all *deterministic* problem instances (no output noise). Once again, the local optimization algorithms make up the upper half of the ranking. A notable difference to the previous two criteria is the placement of Bayesian Optimization on rank nine. This is due to its global search strategy, which keeps exploring areas with high uncertainty, even after inputs close to the optimum have been found.



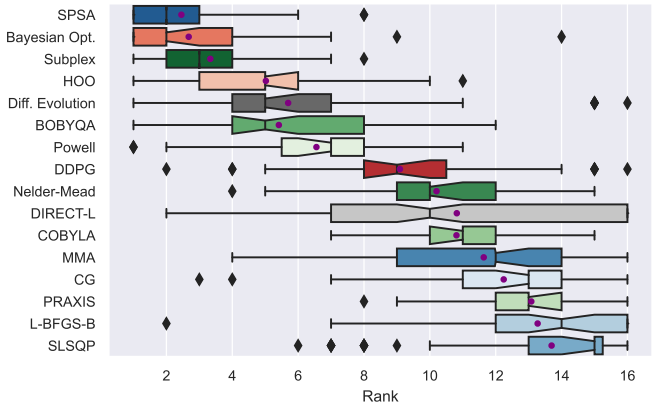
(a) Sample efficiency, $\sigma_{\Delta y} = 0.0$, $\tau = 0.1$



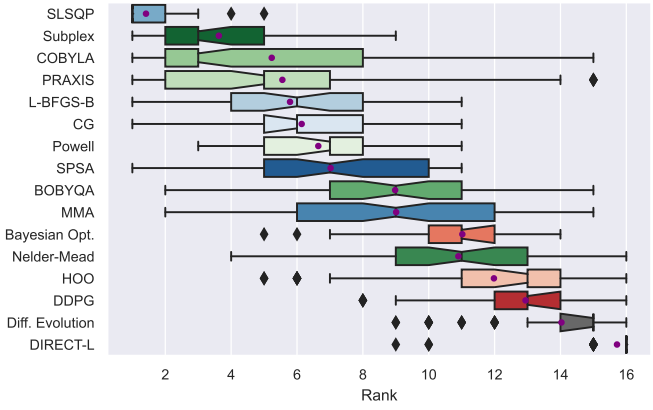
(b) Sample efficiency, $\sigma_{\Delta y} \in \mathbb{R}_{[0.0,0.007]}$, $\tau = 0.1$



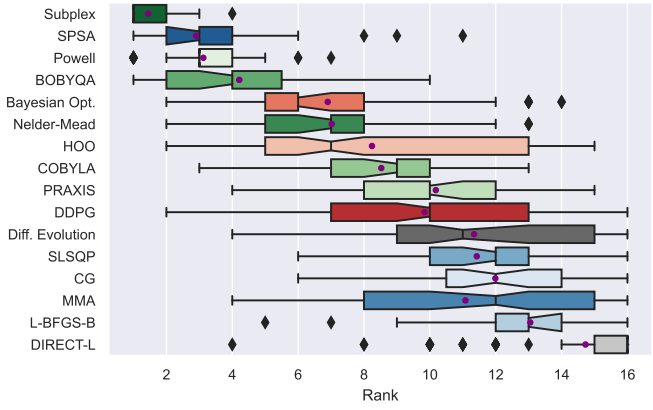
(c) Best in 100, $\sigma_{\Delta y} = 0.0$



(d) Best in 100, $\sigma_{\Delta y} \in \mathbb{R}_{[0.0,0.007]}$



(e) Regret, $\sigma_{\Delta y} = 0.0$



(f) Regret, $\sigma_{\Delta y} \in \mathbb{R}_{[0.0,0.007]}$

Fig. 5. Boxplots of the algorithms' rankings with respect to all three usefulness metrics. Rankings computed on 99 deterministic problems are displayed on the left, rankings computed on 99 stochastic problems (with additive white noise output disturbance) are displayed on the right. Colors indicate algorithm category according to Sec. IV (grey – global, blue – local gradient-based, green – local derivative-free, and red – learning-based). Algorithms are ordered by median. Notches in the box demarcate confidence intervals of the median calculated through bootstrapping. Purple dots indicate mean rank.

Fig. 5f depicts a boxplot of each algorithm's ranking over all *stochastic* problem instances (with output noise). The first three ranks are taken again by local algorithms (Subplex, SPSA, and Powell).

E. Meta Analysis

The 16 algorithms in this comparison were partially chosen for the differences in their key design principles, and partially for their similarity. Therefore, a number of observations can be made. The quasi-Newton methods L-BFGS-B and SLSQP, for instance, are highly competitive in the deterministic setting,

but fail, like most other gradient-based algorithms, in the presence of noise. Powell's method is a variant of the Conjugated Gradient (CG) algorithm, that does not rely on gradient estimates. Powell's method seems to benefit from that in the stochastic setting, whereas CG and PRAXIS – the successor to Powell's method – perform better in the deterministic setting. Subplex seems to be a clear improvement over its predecessor Nelder-Mead in both the deterministic and stochastic settings. In the deterministic setting, COBYLA also appears to be an improvement over Nelder-Mead. Several algorithms, furthermore, approximate the objective function as part of the optimization. L-BFGS-B, SLSQP, MMA, and BOBYQA approximate the objective locally by a convex function. This seems to be especially successful in the deterministic case. Bayesian Optimization and DDPG approximate the objective globally by continuous functions. HOO shares the Bayesian approach with Bayesian approximation but uses a discrete tree structure as an approximation that is more akin to the principle in DIRECT-L. We observe Bayesian Optimization performing best of these three algorithms. This may be due to its correct continuity assumption and the Bayesian approach to computing uncertainty in its estimates.

In summary, local algorithms are most useful for direct input adaption in all respects in the deterministic setting but perform mostly worse when output noise is present. However, the stochastic, local algorithm SPSA outranks almost all other algorithms on all criteria in the stochastic setting. Stochastic algorithms generally perform comparably better in the stochastic setting. This applies to the learning-based algorithms too. Yet, these algorithms can not be considered most useful in neither the stochastic nor the deterministic setting, even though Bayesian Optimization ranks highly except in terms of the regret metric.

VI. Discussion

To complete our presentation of the results, we briefly discuss various aspects of this study.

A. Sensitivity of Metrics

Two of the three usefulness metrics in this study depend on parameters set by the authors. The sample efficiency metric depends on the tolerance level τ, and the best in X metric depends on the sampling budget X. These parameters were set rather arbitrarily to $\tau = 0.1$ and $X = 100$. Higher or lower values may advantage or disadvantage one or the other algorithm. Regret, the third metric, is independent of user-selected parameters.

Tab. IV lists the ranking of the algorithms on deterministic problems with respect to sample efficiency using the same method as in Sec. V, but for different levels of the tolerance level τ. On the level of individual algorithms, the tolerance level affects the position in the ranking. SPSA, for instance, rises in rank the higher the tolerance level is. On the level of classes of algorithms, the ranking is less sensitive to the tolerance level. The upper half of the ranking consists mostly of local optimization algorithms, with the gradient-based ones claiming the highest positions. The only exceptions to that are Bayesian Optimization, which consistently ranks in the top five, and the entrance of Differential Evolution at a tolerance of $\tau = 0.4$. Besides that, the ranking is fairly consistent across algorithmic classes and tolerance levels. This extends to the stochastic setting and is the reason for the omission of a more detailed discussion of the sensitivity in that setting.

TABLE IV
Sensitivity of the sample efficiency ranking with regards to the tolerance level τ for the deterministic problem instances. The upper half of the ranking consists mostly of local optimization algorithms for all tolerance levels. Bayesian Optimization proves to be the exception to the rule.

Rank	$\tau = 0.05$	$\tau = 0.10$	$\tau = 0.20$	$\tau = 0.40$
1	SLSQP	SLSQP	SLSQP	SPSA
2	L-BFGS-B	L-BFGS-B	SPSA	SLSQP
3	CG	Bayesian Opt.	Bayesian Opt.	Bayesian Opt.
4	COBYLA	CG	L-BFGS-B	Subplex
5	Bayesian Opt.	SPSA	Subplex	Diff. Evolution
6	Subplex	Subplex	CG	BOBYQA
7	Powell	COBYLA	PRAXIS	L-BFGS-B
8	PRAXIS	PRAXIS	COBYLA	CG
9	SPSA	Powell	Powell	PRAXIS
10	Diff. Evolution	Diff. Evolution	Diff. Evolution	HOO
11	MMA	MMA	MMA	COBYLA
12	Nelder-Mead	HOO	HOO	Powell
13	HOO	Nelder-Mead	Nelder-Mead	MMA
14	DDPG	BOBYQA	BOBYQA	Nelder-Mead
15	DIRECT-L	DDPG	DIRECT-L	DIRECT-L
16	BOBYQA	DIRECT-L	DDPG	DDPG

Similarly, Tab. V gives the sensitivity of the second metric to the sampling budget for the deterministic problems. Stochastic algorithms SPSA, HOO, and Differential Evolution perform comparatively better under smaller sampling budgets. Besides that, we observe the same dominance of the local optimization algorithms across all values of the sampling budget. On the level of algorithmic classes, the ranking is also robust for different sampling budgets in the stochastic setting.

TABLE V
Sensitivity of the best result ranking with regards to the sampling budget X for the deterministic problem instances. Local optimization algorithms head the ranking with only a few exceptions for all sampling budgets.

Rank	$X = 25$	$X = 50$	$X = 100$	$X = 200$
1	SPSA	SLSQP	SLSQP	SLSQP
2	HOO	SPSA	Bayesian Opt.	L-BFGS-B
3	Bayesian Opt.	Bayesian Opt.	SPSA	CG
4	SLSQP	BOBYQA	L-BFGS-B	Bayesian Opt.
5	Subplex	Subplex	Subplex	Subplex
6	Diff. Evolution	HOO	CG	COBYLA
7	BOBYQA	CG	PRAXIS	SPSA
8	MMA	L-BFGS-B	COBYLA	PRAXIS
9	PRAXIS	PRAXIS	HOO	Powell
10	Powell	Diff. Evolution	Diff. Evolution	Diff. Evolution
11	COBYLA	MMA	BOBYQA	HOO
12	DDPG	COBYLA	MMA	MMA
13	Nelder-Mead	Powell	Powell	BOBYQA
14	CG	DDPG	DDPG	DDPG
15	DIRECT-L	Nelder-Mead	DIRECT-L	Nelder-Mead
16	L-BFGS-B	DIRECT-L	Nelder-Mead	DIRECT-L

B. Solving vs. Converging

The performance of the global and learning-based algorithms (DIRECT-L, Diff. Evolution, Bayesian Optimization,

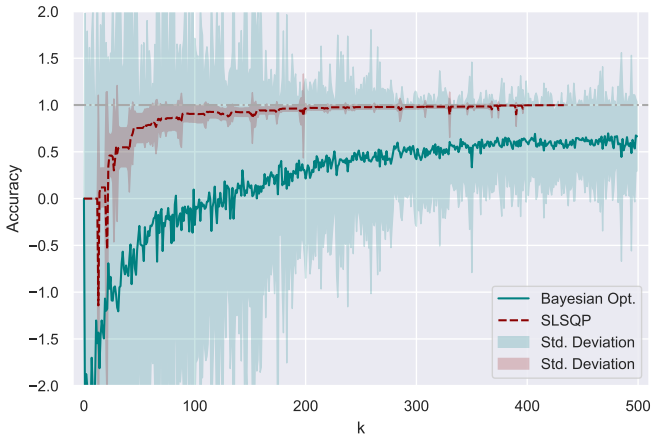


Fig. 6. Mean convergence plot of Bayesian Optimization and SLSQP over all deterministic problems. Shaded areas indicate the standard deviation. Bayesian Optimization has a higher standard deviation. For this reason, it may achieve high accuracy long before convergence.

HOO, and DDPG) in our experiments should be considered with caution. Although they are competitive in some settings, one has to differentiate between fulfilling our usefulness metrics and converging to the optimum. A global optimization algorithm may sample the system by chance close to the optimum after a few iterations, but continue to explore the input space nevertheless. For our usefulness metrics *sample efficiency* and *best in X*, we may regard this as having solved the problem in a few iterations or achieving high maximum accuracy. A local algorithm will improve rather monotonically on the accuracy measure. To illustrate this point, compare the mean and standard deviation of the accuracy over all deterministic experiments for Bayesian Optimization and SLSQP in Fig. 6. Bayesian Optimization has a much larger standard deviation than SLSQP throughout the whole experiment. Yet, we might consider the problem to be solved for our purposes long before Bayesian Optimization has converged. However, this difference is reflected in the usefulness as *low regret* metric and explains the comparatively lower ranking of Bayesian Optimization on this measure.

C. Hyperparameter Search

All algorithms except for HOO and DDPG were run with the default hyper-parameters values. For HOO and DDPG, no such default values exist. Instead, a hyperparameter search was performed on a surrogate model created from previously sampled data. The choice of hyperparameters significantly affected the performance of both algorithms. The results reported herein were obtained with the best hyperparameter settings found during the search. For a fairer comparison, the samples needed for hyperparameter tuning should be included also.

D. Curvature and Gradient-Based Algorithms

Local gradient-based optimization algorithms outperformed most other gradient-free algorithms in our experiments. Gradient-free algorithms, however, are advantageous if the optimization manifold has little to no curvature. One might argue, that in such application cases in which the manifold is mostly flat, gradient-free algorithms would outperform gradient-based algorithms, and, hence, one would reach a different conclusion. Two arguments speak against this. First, it is reasonable to assume that the manifold has curvature in the neighborhood of the optimum that was identified by using any (including gradient-free) optimization algorithm on the digital twin. For continuous and smooth manifolds, this assumption follows from the definition of an optimum. Second, even in cases of very wide and flat optima, gradient-based algorithms would not be significantly worse, since all algorithms would perform virtually equally well.

E. Choice of Algorithms

The algorithms tested in the study at hand are a small subset of all existing blackbox and learning-based algorithms. This subset was selected by the authors and may be considered a fixed factor experiment. Technically, our conclusions can *not* be generalized to the larger set of all existing blackbox and learning-based algorithms, for that reason. A different choice of algorithms may have resulted in a different conclusion. However, this is unlikely since our results indicate general trends. First, we observe that local algorithms are generally more useful than global algorithms in the direct input adaptation context. Second, we note that stochastic algorithms are more competitive in the setting with noise compared to the deterministic setting. A different selection of algorithms is likely to reproduce these trends.

F. Generalization to Other Applications

The results presented in this paper were obtained on a geometry assurance task in the assembly of sheet metal parts. Nevertheless, the results extend to other application domains, as long as their performance function Q is static, continuous, smooth, and of similar dimension. We would expect our usefulness ranking to change for applications with significantly differing control input dimensions, though. That is because the number of data samples needed for estimating a model or gradient often depends on the input dimension. A notable exception to this is the relatively high-ranking SPSA algorithm, which always requires only two samples and would likely strengthen our findings on higher dimensional applications. If an application is of similar dimension, our results would extend to convex problems too. This is due to the assumption made in this study that the optimization manifold is at least *locally* convex over the search space. This assumption follows as implication from the premise that the offline optimization of the digital twin identified a starting point reasonably close to a relatively good optimum in the system. Moreover, if the assumption holds, the starting point for the direct input adaptation may be supplied as well by means of any other model or expert. Although, performance improvements may be realized even without this assumption in other applications with similar properties.

Furthermore, we restricted our experiments to system-twin mismatches that are static and constant in nature. A system-twin mismatch that is *drifting* over timescales smaller than the timescale of the direct input adaptation may lead to different results in analyses similar to ours. Global and learning-based methods would most likely perform worse since those methods rely on the persistent accuracy of all sampled data. It would be less of an issue for local search methods, that use only a few samples to determine the next sampling point. However, our results may extend to *static* system-twin mismatches that can be described as linear or nonlinear continuous input our output disturbances. The effect of such disturbances would be a "warping" of the optimization manifold over the search space in u. As long as the local convexity and smoothness property of the optimization manifold is maintained, such a warping would most likely be without significant impact.

G. No Free Lunch in Optimization

Wolpert and Macready [49] proved mathematically that there is *"no free lunch"* in blackbox optimization. That is to say that all blackbox optimization algorithms perform on average equally well across all optimization problems. Comparing the performance of algorithms on a small subset of problems, as it is done in this study, seems problematic, therefore. The dominance of quasi-Newton algorithms on the deterministic problems and their complete failure in the presence of output noise is likely the most illustrative example of this no free lunch theorem in the paper at hand. However, Wolpert and Macready also emphasize the need for specialized blackbox algorithms, capable of leveraging properties of the specific problem class for faster convergence. Accordingly, we consider the empirical observation of this theorem as yet another reason to research adaptive optimization algorithms in the context of system-twin mismatch.

VII. Conclusion

Digital twins are regarded as key enablers for smart and autonomous systems. The core component of a digital twin is a digital model of such high accuracy that it can be considered to be the twin of its physical counterpart. Given such an accurate model and sufficient computational resources, virtually any off-line optimization algorithm could be used to derive optimal control inputs from the twin to improve the performance of the physical system. For very practical reasons, however, this accuracy may prove elusive, inducing a mismatch between the digital twin and the physical system. Hence, optimality of the control input derived from the digital twin may not be given. Machine learning and AI is often offered as a solution in the literature. In this paper, we juxtaposed the usefulness of 16 learning-based and blackbox optimization algorithms for the purpose of restoring optimal control under system-twin mismatch in high-dimensional, continuous, smooth, and non-linear static systems. To that end, we formulated the problem as an online optimization problem in the direct input adaptation scheme. Our results indicate that local gradient-based blackbox optimization algorithms outperform learning-based algorithms in terms of sample efficiency, accuracy within a limited sampling budget, and regret, even in the presence of measurement noise. Naturally, these results are conditional on the properties of the physical system under consideration. However, this paper highlights the need to extend the search for algorithms, which can restore optimal control in digital twin governed autonomous systems, beyond generic machine-learning algorithms. A repetition of the presented study for systems with different properties may be regarded as a continuation of this work.

References

[1] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," *National Aeronautics and Space Administration*, vol. 32, pp. 1–38, 2012.

[2] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[3] C. Cronrath, L. Ekström, and B. Lennartson, "Formal properties of the digital twin–implications for learning, optimization, and control," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 679–684.

[4] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.

[5] Z. Cunbo, J. Liu, and H. Xiong, "Digital twin-based smart production management and control framework for the complex product assembly shop-floor," *The International Journal of Advanced Manufacturing Technology*, vol. 96, no. 1-4, pp. 1149–1163, 2018.

[6] K. Ding, F. T. Chan, X. Zhang, G. Zhou, and F. Zhang, "Defining a digital twin-based cyber-physical production system for autonomous manufacturing in smart shop floors," *International Journal of Production Research*, vol. 57, no. 20, pp. 6315–6334, 2019.

[7] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108 952–108 971, 2020.

[8] L. Ljung, *System Identification : Theory for the User.*, 2nd ed. Prentice Hall, 1999.

[9] T. Fei, C. Jiangfeng, Q. Qinglin, M. Zhang, H. Zhang, and S. Fangyuan, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9-12, pp. 3563–3576, 2018.

[10] G. Zhou, C. Zhang, Z. Li, K. Ding, and C. Wang, "Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing," *International Journal of Production Research*, vol. 58, no. 4, pp. 1034–1051, 2020.

[11] F. Tao and M. Zhang, "Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20 418–20 427, 2017.

[12] K. M. Alam and A. El Saddik, "C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems," *IEEE access*, vol. 5, pp. 2050–2062, 2017.

[13] R. Rosen, J. Fischer, and S. Boschert, "Next generation digital twin: An ecosystem for mechatronic systems?" *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 265–270, 2019.

[14] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*. Springer, 2017, pp. 85–113.

[15] S. Russell, S. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[16] B. Chachuat, B. Srinivasan, and D. Bonvin, "Adaptation strategies for real-time optimization," *Computers & Chemical Engineering*, vol. 33, no. 10, pp. 1557–1567, 2009.

[17] E. Jorge, L. Brynte, C. Cronrath, O. Wigström, K. Bengtsson, E. Gustavsson, B. Lennartson, and M. Jirstrand, "Reinforcement learning in real-time geometry assurance," *Procedia CIRP*, vol. 72, pp. 1073–1078, 2018.

[18] C. Cronrath, A. R. Aderiani, and B. Lennartson, "Enhancing digital twins through reinforcement learning," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 293–298.

[19] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona, "Reinforcement learning for batch bioprocess optimization," *Computers & Chemical Engineering*, vol. 133, p. 106649, 2020.
[20] T. Savage, P. Petsagkourakis, D. Zhang, N. Shah, E. A. del Rio-Chanona *et al.*, "Data-driven optimization for process systems engineering applications," *Chemical Engineering Science*, p. 117135, 2021.
[21] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Optimization algorithms as robust feedback controllers," *arXiv preprint arXiv:2103.11329*, 2021.
[22] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.
[23] C. Audet and W. Hare, *Derivative-free and Blackbox Optimization*. Springer, 2017.
[24] R. Söderberg, K. Wärmefjord, J. S. Carlson, and L. Lindkvist, "Toward a digital twin for real-time geometry assurance in individualized production," *CIRP Annals*, vol. 66, no. 1, pp. 137–140, 2017.
[25] R. Bohlin, J. Hagmar, K. Bengtsson, L. Lindkvist, J. S. Carlson, and R. Söderberg, "Data flow and communication framework supporting digital twin for geometry assurance," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 58356. American Society of Mechanical Engineers, 2017, p. V002T02A110.
[26] K. Wärmefjord, R. Söderberg, B. Lindau, L. Lindkvist, and S. Lorin, "Joining in nonrigid variation simulation," *Computer-aided Technologies–Applications in Engineering and Medicine*, 2016.
[27] D. C. Montgomery, *Design and Analysis of Experiments*. John wiley & sons, 2017.
[28] J. M. Gablonsky and C. T. Kelley, "A locally-biased form of the DIRECT algorithm," *Journal of Global Optimization*, vol. 21, no. 1, pp. 27–37, 2001.
[29] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[30] J. Nocedal and S. J. Wright, "Conjugate gradient methods," *Numerical Optimization*, pp. 101–134, 2006.
[31] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.
[32] D. Kraft *et al.*, "A software package for sequential quadratic programming," DFVLR, Tech. Rep., 1988.
[33] K. Svanberg, "A class of globally convergent optimization methods based on conservative convex separable approximations," *SIAM Journal on Optimization*, pp. 555–573, 2002.
[34] J. C. Spall *et al.*, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.
[35] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, 1998.
[36] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, 1964.
[37] R. P. Brent, *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.
[38] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.
[39] T. H. Rowan, "Functional stability analysis of numerical algorithms," Ph.D. dissertation, The University of Texas at Austin, 1990.
[40] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*. Springer, 1994, pp. 51–67.
[41] ——, "The BOBYQA algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pp. 26–46, 2009.
[42] A. Slivkins, *Introduction to Multi-Armed Bandits*. Now Foundations and Trends, 2019.
[43] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits." *Journal of Machine Learning Research*, vol. 12, no. 5, 2011.
[44] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Springer, 2003.
[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[46] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
[47] S. G. Johnson, "The NLopt nonlinear-optimization package," 2014. [Online]. Available: http://github.com/stevengj/nlopt
[48] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi. (2018) Scikit-optimize. [Online]. Available: https://scikit-optimize.github.io/
[49] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Constantin Cronrath received his B.Sc. degree in Industrial Engineering from the University of Hamburg, Hamburg, Germany, in 2013 and the M.Sc. degree in Production Engineering from Chalmers University of Technology, Gothenburg, Sweden, in 2016. From 2015 to 2017, he was working as Data Scientist in Perth, Australia. Since 2017, he is pursuing the Ph.D. degree in Automation at Chalmers. His main areas of interest include digital twins, reinforcement learning, safety falsification and sustainable production.

Bengt Lennartson (Fellow, IEEE) was born in Gnosjö, Sweden, in 1956. He received the Ph.D. degree from the Chalmers University of Technology, Gothenburg, Sweden, in 1986. Since 1999, he has been a Professor of the Chair of Automation at the Department of Electrical Engineering, Chalmers University of Technology, where he was the Dean of Education from 2004 to 2007. Since 2005, he has been a Guest Professor at University West, Trollhättan, Sweden. He has (co)authored two books and more than 300 peer-reviewed articles in international journals and conferences. His main areas of interest include discrete-event systems, AI planning and learning, as well as sustainable production.

Prof. Lennartson was the General Chair of the 11th IEEE Conference on Automation Science and Engineering, CASE 2015, and the 9th International Workshop on Discrete Event Systems, WODES'08. He was an Associate Editor of Automatica from 2002 to 2005 and IEEE Transactions on Automation Science and Engineering from 2012 to 2015, and he is an IEEE Fellow for his contributions to hybrid and discrete event systems for automation and sustainable production.