Mechanical Systems and Signal Processing 151 (2021) 107348

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

Controller design by symbolic regression

Kourosh Danai [a,*], William G. La Cava [b]

[a] Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, USA
[b] Institute for Biomedical Informatics, University of Pennsylvania, USA

ARTICLE INFO

Article history:
Received 27 June 2020
Received in revised form 7 September 2020
Accepted 3 October 2020

Keywords:
Symbolic Regression
Nonlinear Control
Structural Adaptation
Controller Design

ABSTRACT

A novel method of empirical controller design is introduced with the potential to produce exotic controller forms. The controllers in this method are derived by symbolic regression (SR) to be in equation form, hence, they are legible in how the controller output is computed as a function of loop variables. Because SR is computationally costly due to its extensive search of controller space, requiring evaluation of millions, if not billions, of candidate controllers, the candidate controllers cannot be evaluated in closed-loop due to the high cost of simulation associated with such evaluation. This paper offers a recourse to this closed-loop evaluation by allowing evaluations to be performed algebraically. To this end, a method of inverse solution is introduced that estimates the plant input for a desired plant output. This estimated plant input is then used as the target output for candidate controllers that can be readily evaluated algebraically based on the available time series of loop variables associated with the desired plant output. Unlike traditional control design which relies on closed-loop performance metrics to provide controller performance guarantees, the proposed open-loop approach sacrifices such guarantees in favor of new controller forms that it may yield. Therefore, the fidelity, as controllers, of candidate controllers need to be verified post-design. For this purpose, the candidate controllers are first evaluated as controllers in closed-loop simulation. Once verified by simulation, they need to be validated for closed-loop stability, as demonstrated for one of the studied cases.

1. Introduction

Controllers are usually defined for a formulaic plant model (i.e., represented by differential equations) and are analytic. When plants are linear or can be approximated by linear models, linear controllers of standard form (e.g., lead, lag, or state feedback) can be considered, with the controller parameters determined according to a design method, such as root locus, pole placement, or frequency response [1]. For nonlinear plants, nonlinear controllers may be considered and designed using a method such as feedback linearization, Lyapunov's direct method, sliding mode, or back-stepping [2–4]. The noted feature of analytic controllers is their legible form which makes them transparent in how their output is computed in relation to the controller inputs (see A for the intended definition of controller transparency).

Analytic control, however, has its limitations: (1) it requires an analytic plant model, (2) it needs expertise, especially for nonlinear controller design, and (3) its form is confined to established controller forms (e.g., state feedback). These limitations have motivated the application of empirical design methods, whereby both the controller structure and its parameters

* Corresponding author.
E-mail addresses: danai@umass.edu (K. Danai), lacava@upenn.edu (W.G. La Cava).

https://doi.org/10.1016/j.ymssp.2020.107348
0888-3270/© 2020 Elsevier Ltd. All rights reserved.

are derived through learning. Noted examples of such empirical methods are fuzzy control [5] and neuro-control [6–9]. The advantage of these controllers is that they can be developed without much control design expertise. Their disadvantage is their "black-box" form which offers little insight about their structure. The objective of this work is to rectify the black-box characteristic of empirically derived controllers, by deriving through learning empirical controllers that are in analytic form.

To produce legible controllers, we use symbolic regression (SR) [10–12] to derive controllers in analytic form (see B for a general overview of SR). Briefly, SR represents the time-series associated with loop variables such as the plant states as symbols and integrates them as blocks to form candidate equations that produce as output a time series close to the target output [13]. As such, these equations are transparent in their structure by informing how their output relates to their inputs. If these equations qualify as controllers, their transparency provides insight into their form, offering the possibility of discovering new controller forms. For instance, an equation found by SR (shown later in Table 1) defines the controller output u as $u = -3.6630e^3(\dot{x} + \dot{x}^2)$, where $\dot{x}$ denotes the velocity. Despite its empirical basis, the controller is transparent in how its output is computed. The objective of this paper is to make controller design by symbolic regression possible.

Ideally, the SR search, free of restrictions in form/structure, can be conducted by genetic programming (GP) for equations that tested as controllers generate closed-loop outputs close to a target output. However, SR is computationally demanding, requiring anywhere from millions to billions of evaluations. If the candidate controllers were to be evaluated for their performance in closed-loop, as traditionally conducted in empirical controller design, then each evaluation would require a time-consuming closed-loop simulation. But it is unfeasible to conduct so many simulation-based evaluations of candidate controllers during a genetic search, rendering closed-loop evaluation of candidate controllers impractical. Because of this computational impediment, the use in controls of evolutionary and/or genetic algorithms has been confined to parameter optimization [14,15] or search among a limited number of structural components [16–18], leaving unexplored the true benefit of SR in unrestricted search of controller forms.

An alternative to time-consuming closed-loop evaluation of candidate controllers by simulation is algebraic evaluation, which can be much more rapidly obtained. Such algebraic evaluation would be possible if the desired plant input were available to use as target for the candidate controller output. But, even with the availability of the desired plant input, and derivation of the candidate controllers by SR, it remains to be determined, as a fundamental hypothesis, whether such candidate controllers designed in open-loop would function as controllers in closed-loop. The paper examines this hypothesis by (1) offering a method of inverse solution to provide the plant input for a desired plant output, (2) deriving candidate controllers by SR, and (3) evaluating the performance of these candidate controllers in closed-loop. The above strategy is illustrated in Fig. 1.

As to the inverse solution, the first contribution of this work (termed "Input Estimation" in Fig. 1 and elsewhere in the paper), it is trivial for a linear plant since it can be computed using the discrete-time transfer operator of the plant as a difference equation. It incorporates the target output in the auto-regressive part so as to compute the plant input from the moving average part. For a nonlinear plant, however, the solution requires a second phase, which also relies on the linearized model of the plant. In the first phase, as for the linear plant, the approximate input to the linearized model of the plant is obtained with the desired output as the target. In the second phase, this input is adapted in a method akin to iterative learning control (ILC) [19,20] toward an input that renders the target output by the nonlinear plant.

Once the desired/target plant input becomes available, the potential controller can be constructed by SR to yield as its output the estimated plant input. SR uses the space of plant states and inputs as time-series to search for this potential controller (see Fig. 1). For SR, we use Epigenetic Linear Genetic Programming (ELGP) [21–24] which is developed to yield concise models. Equations that yield as output the target plant input (representing the inverse solution to the desired plant output) may qualify as controllers and will be tested via simulation in closed-loop.

We present the results from application of the proposed *SR-Based Controller Design (SRBCD)* method to three nonlinear plants. The results not only affirm the fundamental hypothesis of the study by confirming the efficacy of controllers generated by the SRBCD method, they provide valuable insight about the choice of variables to be included in SR to render an effective controller form for closed-loop operation. These results are, therefore, significant in not only validating the proposed method but also affirming the possibility of controller design in open-loop, whereby controllers can be designed with the objective of producing as output the desired plant input. Even though the proposed method of open-loop controller design, together with its input estimation, is examined and validated with controllers derived by SR, the open-loop design strategy, in and of itself, ought to be considered the main contribution of the present work. The justification for this claim is rooted in the paper's finding that controllers can be designed in open-loop to target the desired plant input. Such a strategy may be implemented, independent of SR, with any design strategy, including linear regression.

The paper is organized as follows. The input estimation method, which is an integral part of the proposed open-loop controller design strategy, is discussed in Section 2. A brief description of ELGP, which is an efficient method of search developed by the authors for creating concise dynamic models [23,24], is presented in Section 3. Section 4 presents the results from the application of the proposed method to three nonlinear plants, selected for the various challenges they pose to controller design. Some of the controllers presented in this section manifest the uniqueness of controller forms derived by SR, potentially unattainable by traditional design techniques. A necessary step to open-loop design, is stability analysis. An example of such analysis is provided in Section 5 for one of the studied cases via loop gain describing functions. Discussion of the results, future work, and conclusion are discussed in Sections 6–8, respectively.

Table 1
Sample controller equations found by ELGP for the active nonlinear suspension system.

$G_{c1}: u = -\dot{x}(5.3985e^4 x + 4.8533e^3\dot{x} + 463.59)$
$G_{c2}: u = 3.4242 - 4.5455e^4 x\dot{x} - 4.8780e^3(\dot{x} + 0.1459)\dot{x}$
$G_{c3}: u = -3.6630e^3(\dot{x} + \dot{x}^2)$
$G_{c4}: u = -3.39e^3\dot{x}$
$G_{c5}: u = 7.7156e^3 \sin\dot{x}(2x + \dot{x} + 0.2042)$

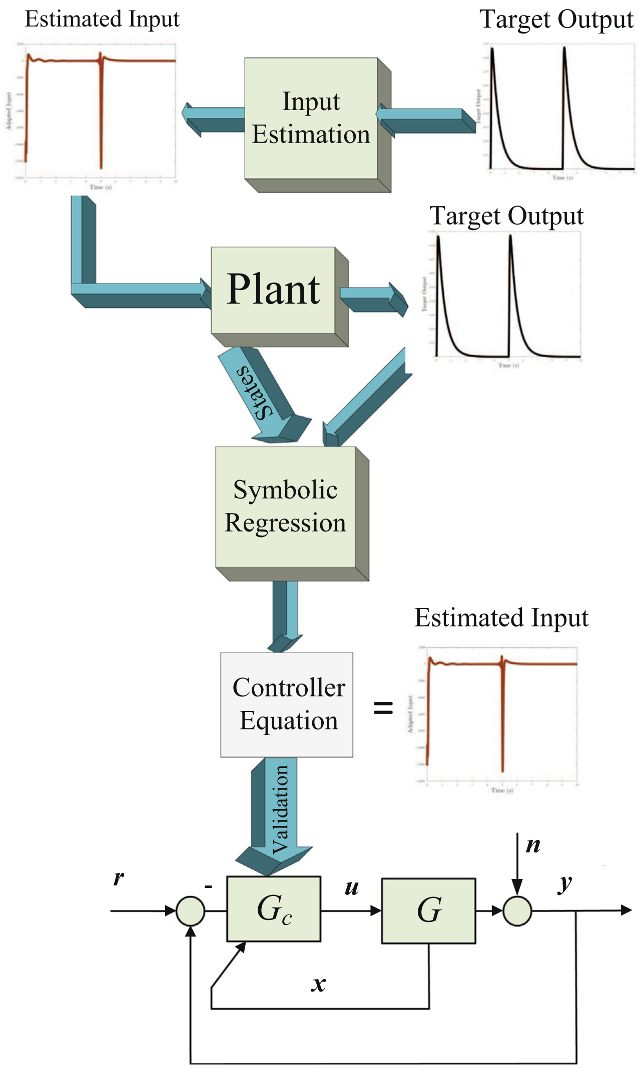

Fig. 1. The sequence of input estimation and controller construction in the proposed method. The plant input is first estimated for a desired output. This input is then used as the target of symbolic regression for finding candidate equations to be tested as controllers in closed-loop.

2. Plant input estimation

Estimation of plant input for a desired plant output, commonly referred to as inverse modeling [25], is generally hindered by non-minimum phase characteristics and nonlinearities [26]. We avoid here the construction of an inverse model [27] and focus instead on estimating the input (i.e., the inverse solution) alone, independent of the inverse model that would generate it. To this end, we rely on the discrete-time model of the plant as a difference equation.

Consider the transfer operator of the linear plant or of the linearized model of the nonlinear plant, as

$$\frac{y(k)}{u(k)} = \frac{B(q)}{A(q)} = \frac{b_1 q^{-1} + b_2 q^{-2} + \cdots b_m q^{-m}}{1 + a_1 q^{-1} + a_2 q^{-2} + \cdots a_n q^{-n}} \tag{1}$$

where q is the forward shift operator. Then, the above model can be used as a difference equation to estimate the input sequence for a desired/target output sequence $y_d(k),\ k = 1, \ldots, N$, as

$$\widehat{u}(k-m) = 1/b_m \left[y_d(k) + \sum_{i=1}^{n} a_i q^{-i} y_d(k) - \sum_{i=1}^{m-1} b_i q^{-i} \widehat{u}(k) \right] \tag{2}$$

By indexing k from $n+1$ to N, one at a time, one can obtain an estimate of the input $\widehat{u}(l)$ for $l = n+1-m, \ldots, N-m$, where $\widehat{u}(k) = 0\ \forall k < n+1-m$. It should also be noted that this method is only applicable to a forced response, since it assumes that any output y_d is caused by a corresponding input sequence. As such, y_d cannot be due to any initial conditions.

The application of this input estimation method to a linear plant is trivial as well as effective. Its utility in estimating the inputs to nonlinear plants, however, requires two phases, as shown in Fig. 2. In the first phase, as for the linear plant, an initial input is estimated based on a linearized model of the nonlinear plant. But given that this estimated input $\widehat{u}(l)$ would not render the target output by the nonlinear plant, one would need to adapt it toward the inverse solution $u^*(l)$ that renders the target output by the nonlinear plant. If one considers the current nonlinear plant output $y_j(k),\ k = 1, \ldots, N$ by some input $u_j(k),\ k = 1, \ldots, N$, as $y_j = f(u_j)$, and the target output y_d, as $y_d = f(u^*)$, and consider a first order approximation of the target output as

$$y_d \approx y_j + \frac{\partial f(u)}{\partial u} \delta u_j \approx y_j + \frac{\partial y}{\partial u} \delta u_j \tag{3}$$

where δu_j denotes the input error, then the error, $\epsilon_j(k) = y_d(k) - y_j(k)$, between the target and current output will have the form

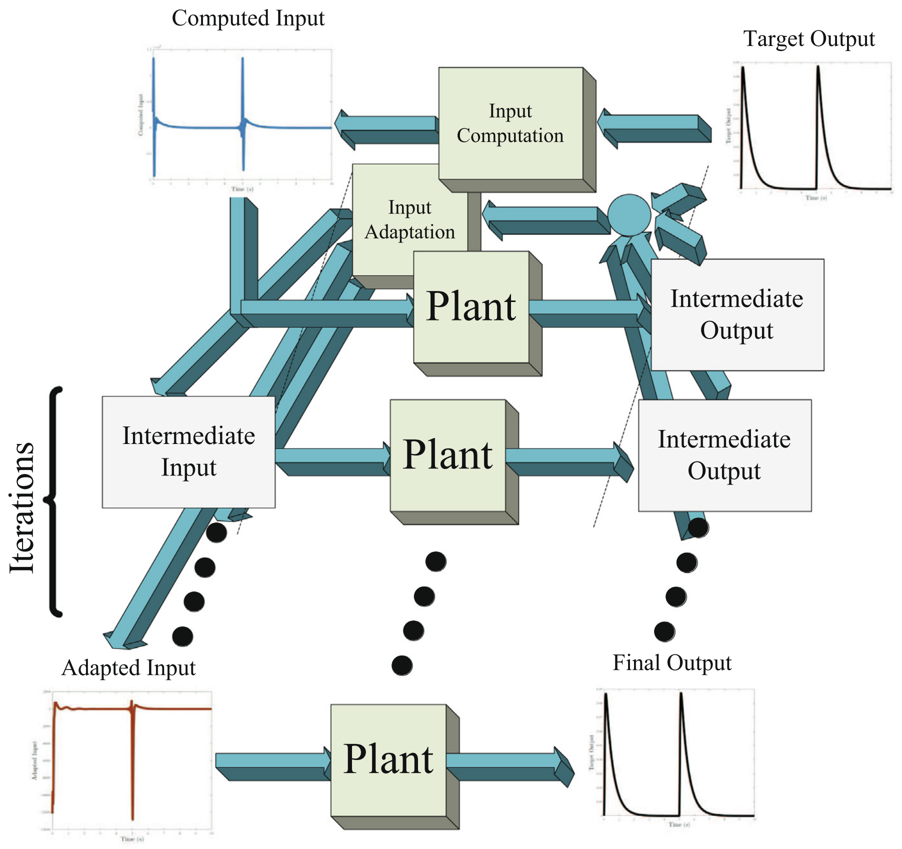

Fig. 2. Illustration of input estimation for a nonlinear plant in the proposed method. The input is first computed based on the linearized model of the plant. The output of the nonlinear plant, by this computed input, is then matched against the target output for computing the error. This error is then used for adapting the input according to Eq. (6). The new plant output is computed by this input and adaptation continues as in the previous step until the error has diminished according to a threshold.

$$\epsilon_j \approx \frac{\partial y}{\partial u}\delta u_j \tag{4}$$

Now if we consider the transfer operator in Eq. (1) as a valid approximation of $\partial y/\partial u$, then Eq. (4) can be defined as

$$\epsilon_j \approx \frac{B(q)}{A(q)}\delta u_j \tag{5}$$

and the estimation routine (2) can be used to estimate the input error δu_j, as

$$\widehat{\delta u}_j(k-m) = 1/b_m\left[\epsilon_j(k) + \sum_{i=1}^{n} a_i q^{-i}\epsilon_j(k) - \sum_{i=1}^{m-1} b_i q^{-i}\widehat{\delta u}_j(k)\right] \tag{6}$$

This input error estimate δu_j can then be used in the iterative input adaptation routine

$$\hat{u}_{j+1}(l) = \hat{u}_j(l) + k_p\widehat{\delta u}_j(l);\;\; k_p \in [0\;\; 0.2] \tag{7}$$

toward the desired output y_d until an error limit $\sum_{k=1}^{N}(\epsilon_j(k))^2 < C$ can be reached. The adaptation analogy here is that given the fidelity of the model in Eq. (5), used for updating δu in Eq. (6), $\lim_{j\to\infty} u_j = u^*$ according to Eq. (7), which translates to $\lim_{j\to\infty} y_j = f(u^*) = y_d$ or $\lim_{j\to\infty}\epsilon_j = 0$.

3. Symbolic regression by ELGP

In traditional applications of SR to dynamic modeling, analytical models are developed according to building blocks that comprise the state variables, inputs, constants (coefficients and exponents), and algebraic functions and operators. Symbolic regression generally uses genetic programming (GP) to search for the nonlinear differential equations that can fit the observations [10,28–30]. Classical GP [13] represents the equations by tree structures, although linear representations and others are also in use such as gene expression programming [31], cartesian GP [32], tree adjunct grammars [33], and grammatical evolution [34].

Traditional GP tends to generate overly complex or over-fit solutions, which are effectively "black box" [35]. A related but not identical problem is bloat, which is the tendency of programs to continue to grow in size with more generations, to the detriment of the search process [36]. We have developed Epigenetic Linear Genetic Programming (ELGP) [21–24] to efficiently form concise models. ELGP addresses the problem of model conciseness by regulating program expression such that accuracy and conciseness are optimized. Unlike the aforementioned methods, this regulation of program expression results in little to no computational overhead compared to traditional GP [21,22].

ELGP introduces several innovations to traditional GP-based symbolic regression. First, building on prior work [37,38], it uses a stack-based representation in lieu of tree-based representation, as illustrated in Fig. 3. This 'minimal-syntax' representation allows instructions to be silenced or activated in a genotype without invalidating the program's ability to execute; therefore, it eliminates syntactic invalidities due to changes to instructions and variables. Second, ELGP introduces epigenetic information into GP representation by including an on/off condition on each instruction in an individual's genotype. Programs are encoded as post-fix notation linear genotypes by an epigenotype with corresponding on/off values. When evaluated together, the expressed program (candidate equations); i.e., phenotype, is produced by ignoring the instructions that are turned off. This encoding not only maximizes the flexibility of representation, thereby simplifying the genetic operations, but also allows for more fine-tuning of genotype expression into phenotypes (i.e., candidate equations). Third, ELGP uses epigenetic hill climbing (EHC) as a local adaptation mechanism whereby the epigenotype is modified to improve the expressed equation. As such, the epigenetic layer introduces flexibility to the expression of a genotype by including unused expressions as a repository of local solutions to explore in the search space. Lastly, model structure improvements brought about by EHC

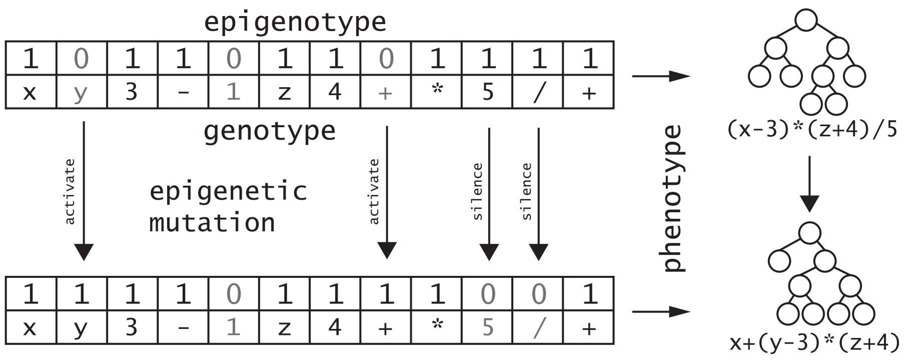

Fig. 3. Illustration of stack-based representation in ELGP and its process of generating phenotypes (i.e., candidate equations).

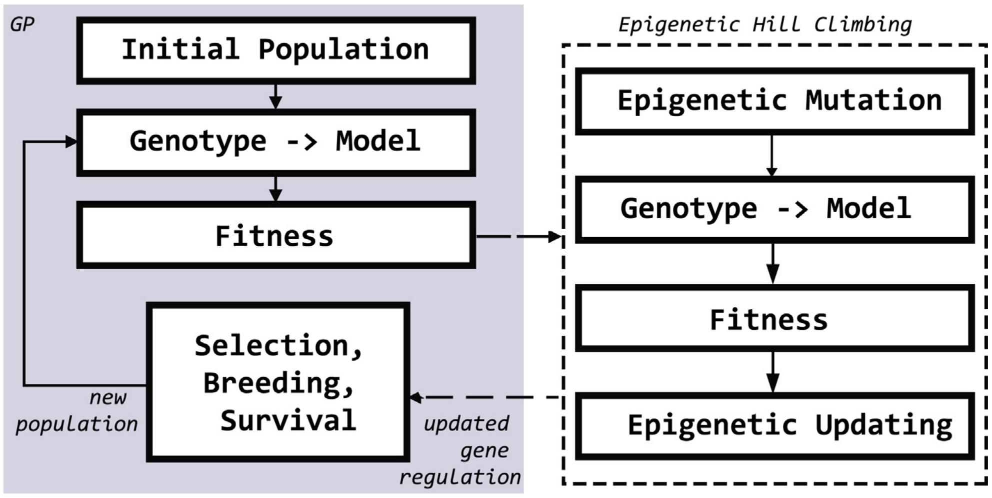

Fig. 4. Block diagram of ELGP. The typical GP steps are shown on the left. After fitness evaluation and before selection, the population undergoes an iteration of epigenetic hill climbing, represented by the block on the right.

are inherited such that the evolutionary process can benefit from local learning. Epigenetic activation and silencing is learned each generation using a stochastic hill climber and co-evolved with the corresponding genotypes.

The operation steps of ELGP are depicted in Fig. 4. Operation starts with randomly constructed programs (i.e., equations) that comprise an initial population. The outputs generated by these equations are evaluated for closeness to the target input within the training data. Depending on the variant of ELGP, the population then undergoes some form of epigenetic adaptation (e.g., epigenetic hill climbing). Afterwards, the population undergoes selection, recombination and mutation, as in standard GP, to produce an updated population. The process repeats until an adequate solution is produced.

The particular features of ELGP that benefit the current application are its efficiency and the conciseness of equations it develops compared to GP, as demonstrated in application to numerous system identification problems [23,24]. ELGP has consistently produced smaller models with better estimation performance for systems represented by nonlinear ordinary differential equations, and in system identification of real-world benchmark problems, industrial processes, population diversity models [23], and industrial wind turbines [24]. The mechanism responsible for its superior performance is shown to result from preservation of diversity in the equation forms during optimization, precipitated from the introduction of the epigenetic layer [21,22].

4. Study platforms

A salient feature of the SRBCD method is its capacity to present novel and exotic controller forms. As such, the potential value of SRBCD is in application to nonlinear plants, since it is unlikely to render controllers of superior performance to the linear controllers already available for linear plants. As such, we present here only the results from the SRBCD's application to nonlinear plants. Several controllers are developed for each of the three plants. Two of the derived controllers are regulators and one performs tracking. The first plant is a nonlinear active suspension system representing a minimum-phase system which requires regulation against road disturbances. The second plant is a linear system with a nonlinear valve that imposes nonlinearity on the input, for which a tracking controller is designed. The third plant is an inverted pendulum which is non-minimum phase and requires regulation. The plant models, target outputs, estimation of the plant inputs, the controllers constructed by ELGP, and the closed-loop performance of these controllers are presented in the corresponding subsections.

4.1. Case I: nonlinear active suspension

The first platform is a nonlinear active suspension as shown in Fig. 5. The controller applies the regulating force f_s to minimize the effect of road disturbance r. The equation of motion of this system is defined as

$$m_b\ddot{x}_b + b_s(\dot{x}_b - \dot{r})|\dot{x}_b - \dot{r}| + k_s(x_b - r)^{1.5} = f_s(t) \tag{8}$$

where both the spring and damper are nonlinear elements, with $m_b = 300$ kg, $b_s = 1000$ N s/m, and $k_s = 20,000$ N/m. This system is intended to test the SRBCD's effectiveness in regulation of nonlinear minimum-phase systems.

4.1.1. Case I: input estimation

The target output for the nonlinear plant is set to be the impulse response $(x(t)|_{r(t)=\delta(t)},)$ of a linear suspension of the form

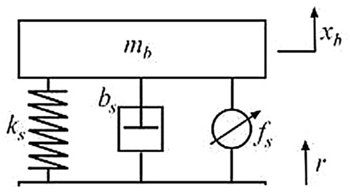

Fig. 5. Schematic of the active suspension system used as the first study platform.

$$m_b\ddot{x} + B_s\dot{x} + k_s x = B_s\dot{r} + k_s r \tag{9}$$

with a damping coefficient B_s ten times the damping coefficient of the nonlinear suspension ($B_s = 10b_s = 10,000$ N s/m). Using this target output in Eq. (2) for the first stage of input estimation, based on the linear model

$$\frac{y(k)}{u(k)} = \frac{0.4143e^{-7}q^{-1} + 0.4120e^{-7}q^{-2}}{1 - 1.9818q^{-1} + 0.9835q^{-2}}$$

which is the discretized transfer operator at the sampling frequency of 200 Hz of the standard transfer function

$$G(s) = \frac{1/m_b}{s^2 + (b_s/m_b)s + k_s/m_b}$$

produces the computed input for subsequent adaptation. The zero-phase low-pass filtered computed input is shown in the left plot of Fig. 6. Whereas this input would generate the target output by the above linear model, it is unsuitable for the nonlinear model. The much different output from the target output generated by the nonlinear plant with this input is shown in the right plot of Fig. 6 together with the target output.

It is clear from the pair of outputs in the right plot of Fig. 6 that the response of the nonlinear suspension is far from its target and adaptation is necessary for finding the correct input to the nonlinear plant. The adapted input (with $k_p = 0.05$ in Eq. (7) and the adaptation limit of $\sum_{k=1}^{N}|\epsilon(k)| < 50$ in Eq. (6)) is shown in the left plot of Fig. 7. This input produces the final output in the right plot of Fig. 7, shown together with the target output. The final and target outputs are very close, indicating the success of adaptation. For illustration purposes, shown in Fig. 8 is the change in $\sum_{k=1}^{N}|\epsilon_j(k)|$ during input adaptation.

4.1.2. Case I: controller forms

Next, ELGP was used to find an equation to define the adapted input (as target) in terms of the regressors. In this case, the only variables available were the position and velocity (x and $\dot{x}$) of the mass m_b in response to the adapted input and both

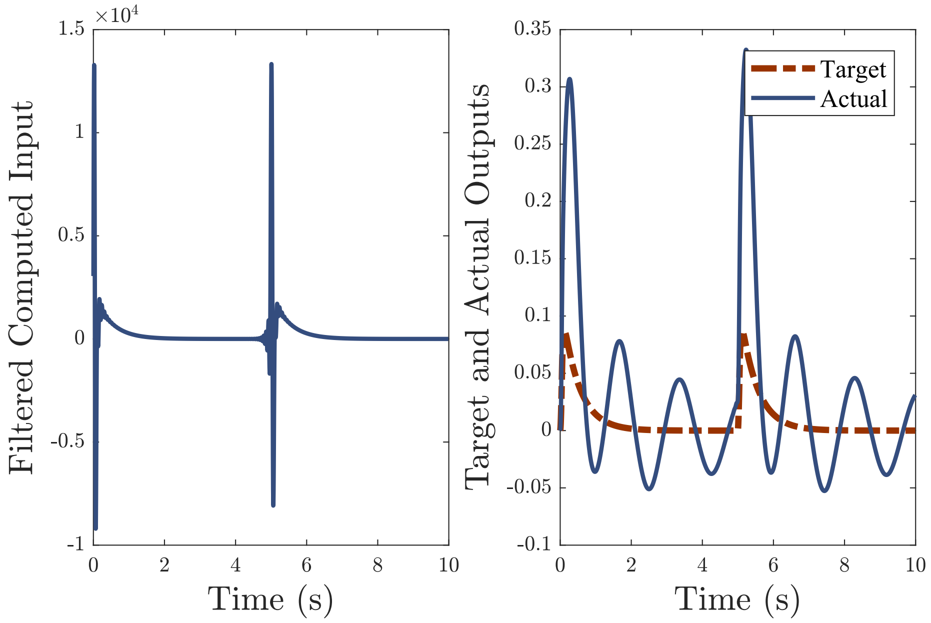

Fig. 6. Zero-phase low-pass filtered computed regulating input (left) and the output generated by the nonlinear suspension (right) in response to an impulse disturbance ($r(t) = \delta(t)$). Also shown on the right is the desired/target response for the nonlinear suspension.

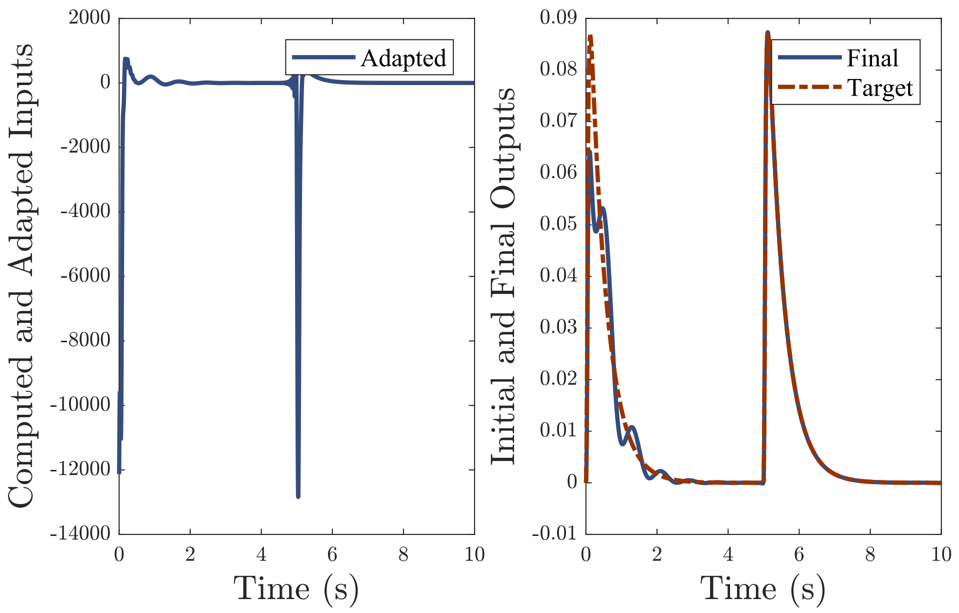

Fig. 7. The adapted input (left) and output of the nonlinear suspension after adaptation (right), shown with the target output.

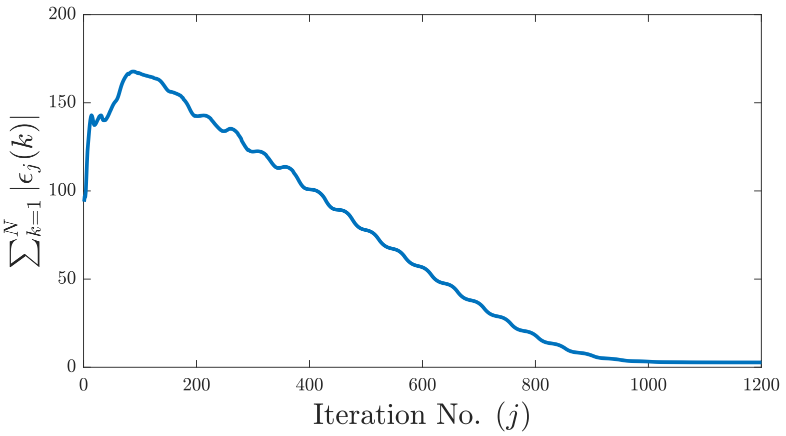

Fig. 8. Absolute sum of the error during input adaptation of the nonlinear suspension.

were used as regressors. The target input for regression and the regressors used by ELGP are shown in the left and right plots of Fig. 9, respectively. In general, necessary for equation generation is adequate representation of the shape features of the target (i.e., the estimated plant input) by the regressors. For example, the adapted plant input, shown again as the left plot of Fig. 9, contains spikes as well as waves. The spikes are represented in the mass velocity alone, but the waves are in both regressors, represented in the right plots of Fig. 9. Several nonlinear equations were obtained, shown in Table 1, which were validated in closed-loop as controllers. Two of them contain sinusoidal functions and some (e.g., G_{c3} and G_{c4} that include only one variable) have simpler forms than others.

Assuming that the regressors provide an adequate basis for the adapted input, there is a high likelihood that the equation found by ELGP will not provide a perfect match to the target input. This mismatch will, in turn, result in deviations of the actual controller inputs, from the regressors, when operated in closed-loop, causing further mismatch between the plant output and its intended target. It is because of these deviations that the functionality as closed-loop controllers of equations need to be investigated. Using a closed-loop model similar to the one depicted in the Simulink$^{\text{TM}}$ model of Fig. 10, the equations developed by ELGP were tested as controllers. The equations in Table 1 produce similar closed-loop plant inputs, despite their very different forms. For illustration purposes, the closed-loop inputs generated by the controllers in Table 1 are shown in the left plot of Fig. 11, with the corresponding plant outputs shown as the right plots of this figure. The results are both promising and revealing. They are promising in that equations developed by ELGP can function as controllers. They are revealing in that they do not perfectly match their counterparts in Fig. 7 in settling time. As discussed earlier, this

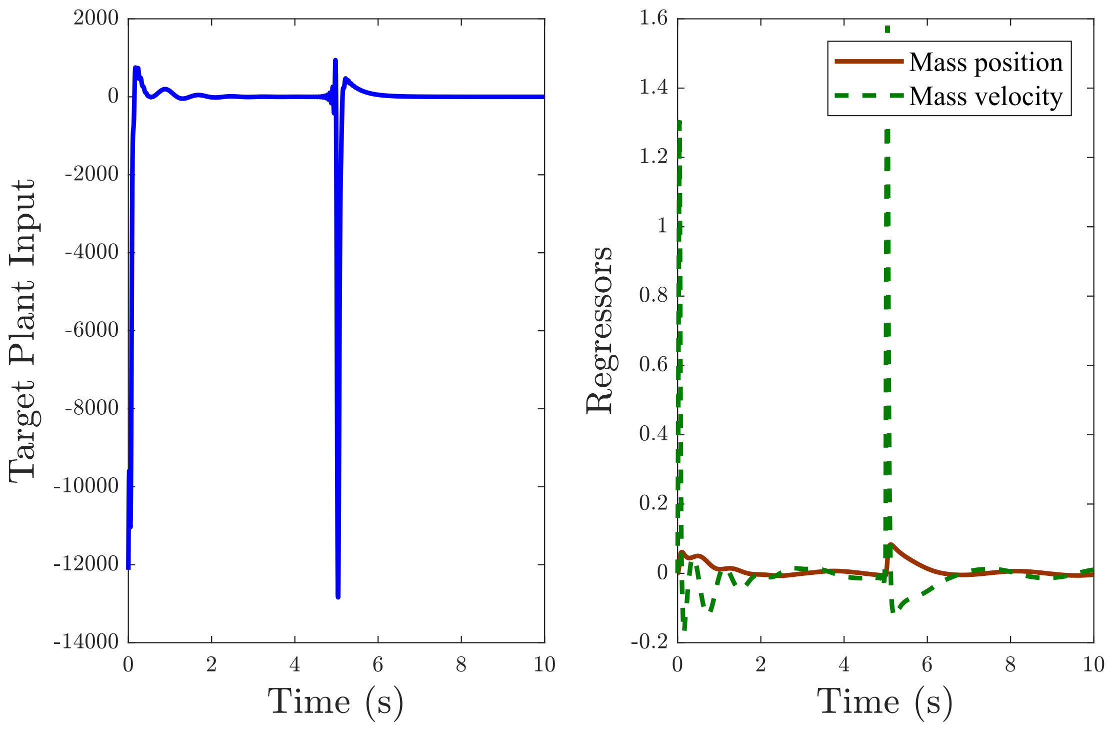

Fig. 9. Target input (left) and the variables used as regressors (right) by SR for finding the controller for the nonlinear suspension.

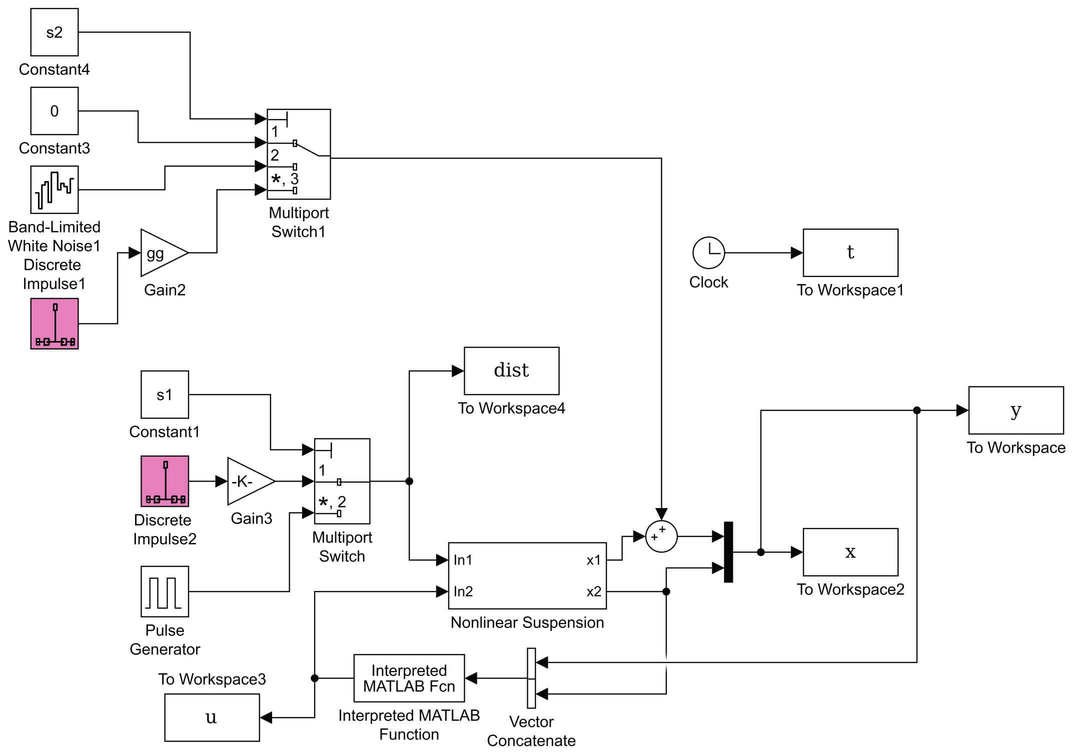

Fig. 10. Simulink^{TM} model for evaluating the controller obtained by SRBCD for the active suspension system.

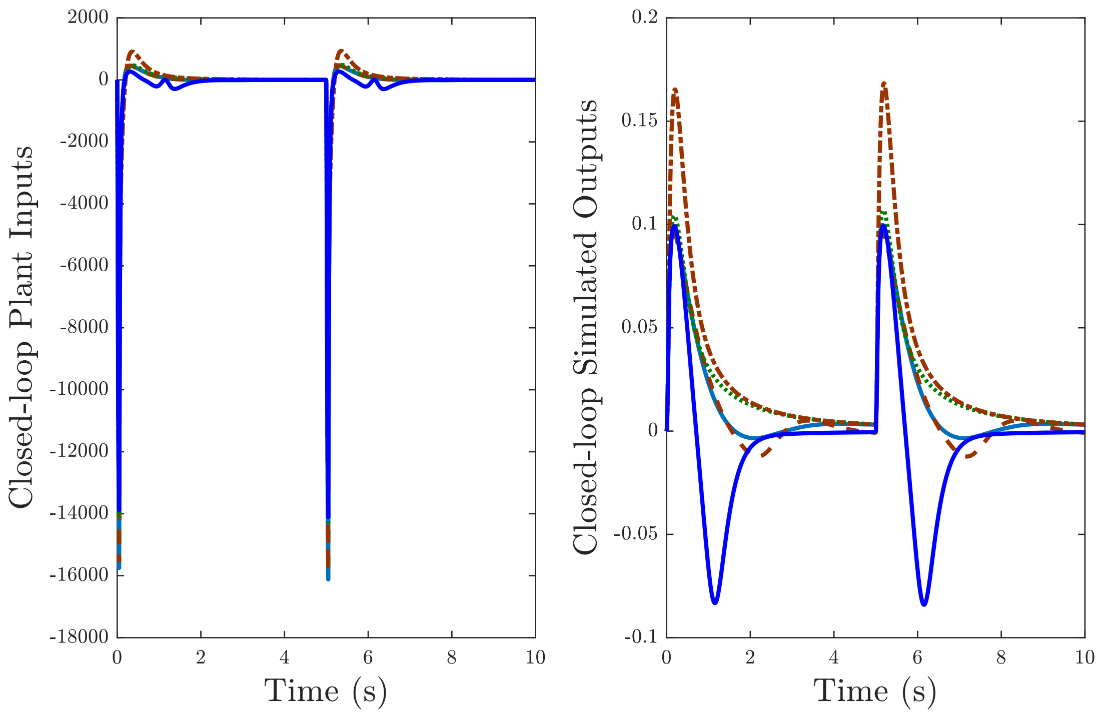

Fig. 11. Closed-loop simulated input (left) and output (right) of the nonlinear active suspension in response to impulse disturbances by the controllers in Table 1.

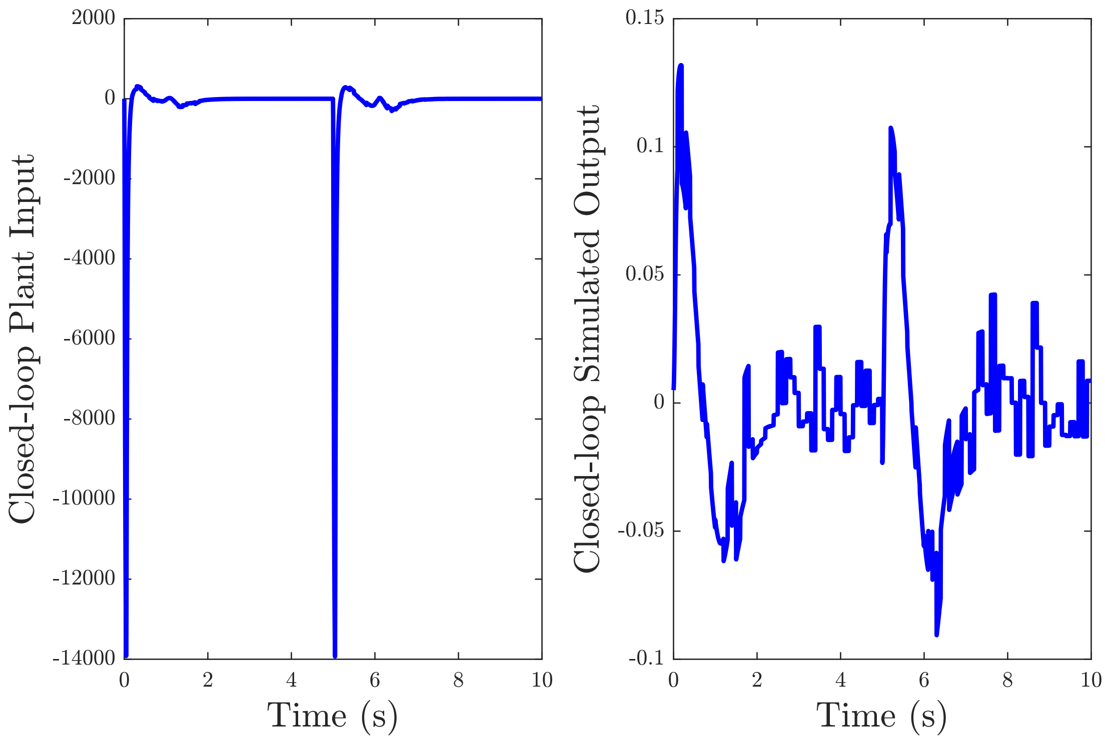

Fig. 12. Closed-loop simulated input (left) and output (right) of the nonlinear active suspension in response to impulse disturbances by the controller G_{c5} in Table 1 with added output noise.

expected deviation is due to not only the inability of ELGP equations in perfectly replicating the estimated plant input, but also the inevitable deviations of the controller inputs from those used in regression.

To test the noise rejection capacity of these controllers, band-limited white noise was added to the closed-loop output, as shown in Fig. 10. Shown in Fig. 12 are the closed-loop input (left) and output (right) of controller G_{c5} in Table 1 with the added output noise. The results indicate that the controller can operate with the added output noise.

4.2. Case II: linear system with a nonlinear valve

The second platform is the combination of a linear system and a nonlinear valve, adopted from [39]. The nonlinearity of the valve $f(u)$ and the transfer function of the linear system $G_0(s)$ are defined below,

$$f(u) = u^{1.35}, G_0(s) = \frac{1}{s^3 + 3s^2 + 3s + 1} \tag{10}$$

where the plant output $y(t)$ has the form

$$y(t) = \int_0^t f(u(\tau))g_0(t-\tau)\mathrm{d}\tau$$

with $g_0(t) = \mathcal{L}^{-1}\{G_0(s)\}$ being the impulse response function of the linear component.

To obtain the target output for this system, we first linearized the plant, then designed a PID (proportional + integral + derivative) controller for the linear system. The closed-loop response of this linear controlled system to a reference was designated as the target output of the nonlinear plant. The reference and target output are shown in the left and right plots of Fig. 13, respectively.

4.2.1. Case II: input estimation

As in the previous case, the target output was used to compute the plant input according to Eq. (2), using the linear model

$$\frac{y(k)}{u(k)} = \frac{1.3377e^{-4}q^{-1} + 2.6755e^{-4}q^{-2} + 1.3377e^{-4}q^{-3}}{1 - 2.7145q^{-1} + 2.4562q^{-2} - 0.7408q^{-3}}$$

obtained by discretizing at the sampling frequency of 10 Hz the linearized model of the plant

$$G(s) = \frac{0.6209}{s^3 + 3s^2 + 3s + 1}$$

The computed input is shown in the left plot of Fig. 14, with the corresponding output by the nonlinear plant (denoted as initial) shown in the right plot, together with the target output. It is clear, from the two outputs in the right plot of this figure, that the output generated with the computed input is far from its target, requiring adaptation of the plant input.

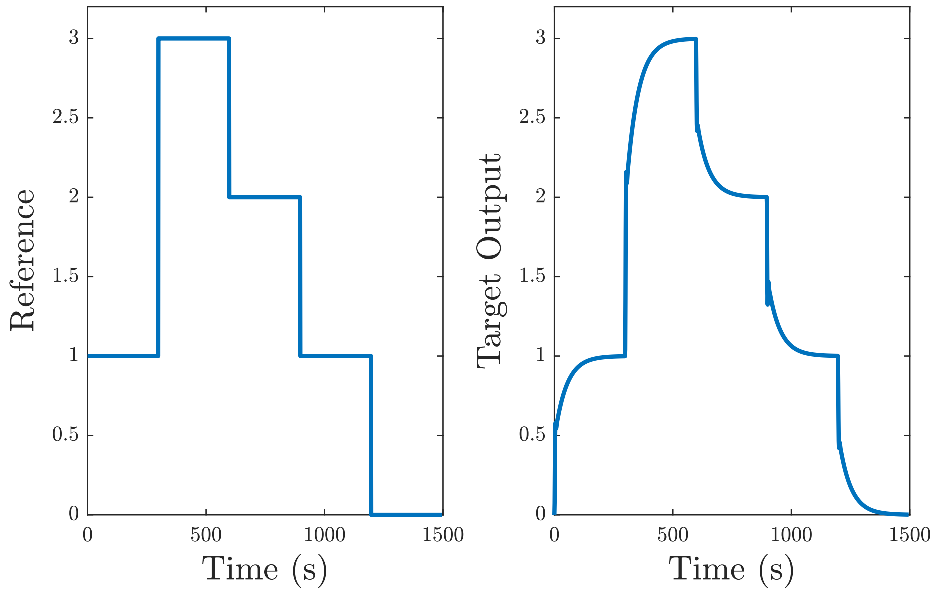

Fig. 13. The reference (left) used to generate the target output (right) for the nonlinear valve according to a linear closed-loop control system.

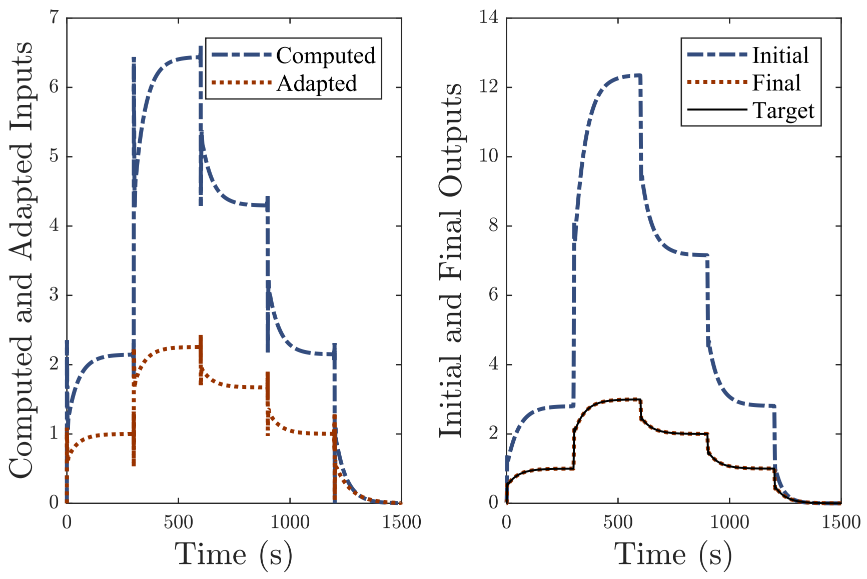

Fig. 14. The computed and adapted inputs (left) and the initial and final outputs of the nonlinear valve platform after adaptation (right), shown with the target output.

Table 2
Sample controller equations found for the nonlinear valve by ELGP for tracking.

$G_{c1}: u = 0.6165R - x_2 - \sin x_2 + 0.6165Re^{-R}(R + 0.4238) + 0.0589$
$G_{c2}: u = 0.6309R + 0.3619$
$G_{c3}: u = 0.3617 - (x_2 - 0.6311)(R - x_2)$
$G_{c4}: u = R\left(\left(Rx_2\left(x_2R^2 - R + x_2\right) + 0.3785\right)/(R - x_2) + 0.6247\right) - x_2 - x_2(R + 1.54)$
$G_{c5}: u = -(R + 0.5721)(x_2 - 0.6312)$

The adapted input according to Eq. (7) (with $k_p = 0.1$ in Eq. (7) and the adaptation limit of $\sum_{k=1}^{N}|\epsilon(k)| < 20$ in Eq. (6)) is shown in the left plot of Fig. 14, together with the computed input. The final output is shown in the right plot of Fig. 14, together with the initial and target outputs. The results indicate that the final output is virtually the same as the target output and that the adaptation mechanism is effective in finding the correct input for the plant.

4.2.2. Case II: controller forms

Controller forms were obtained for this system using as regressors, the reference input (R) and the second state of the plant (x_2), to exclude the output. The reason for excluding the output as a regressor, in this case, was that ELGP would invariably choose to define the equations in terms of the output alone, and independent of the reference, due to the better conformity of the desired plant input with the target output. However, the absence of a reference would leave the controller in regulation mode and insensitive to tracking. The controller forms found for this plant are shown in Table 2.

The closed-loop plant inputs and outputs using the equations in Table 2 as controllers are shown in the left and right plots of Fig. 15, respectively, together with the reference. Note that the reference sequence used in the test is different from the one used for input estimation and SR, to evaluate the controllers' capacity in tracking. Although the controllers respond to the reference, they seem to produce more agile outputs than the target output, having shorter transients. There is also some steady-state error at the magnitudes of 2 and 4, due to the static nature of the equations constructed by ELGP. Here, the deficiency of the controllers in producing the target output in its entirety is because of using the reference in exclusion of the plant output. The absence of the plant output as a regressor deprives the controller of the potentially desirable output transients.

It is likely that both deficiencies, namely the steady-state error and the lacking output dynamics, can be resolved by incorporating the integral (summation) and/or derivative of the tracking error as additional regressors in ELGP, so as to enable construction of dynamic equations (controllers). In order to test this hypothesis, dynamic controllers were constructed by including as regressors the tracking error as well as its integral and derivative. The controllers constructed by ELGP are shown in Table 3, and their performance is shown in Fig. 16. Except for G_{c1} which is a nonlinear controller, the others are

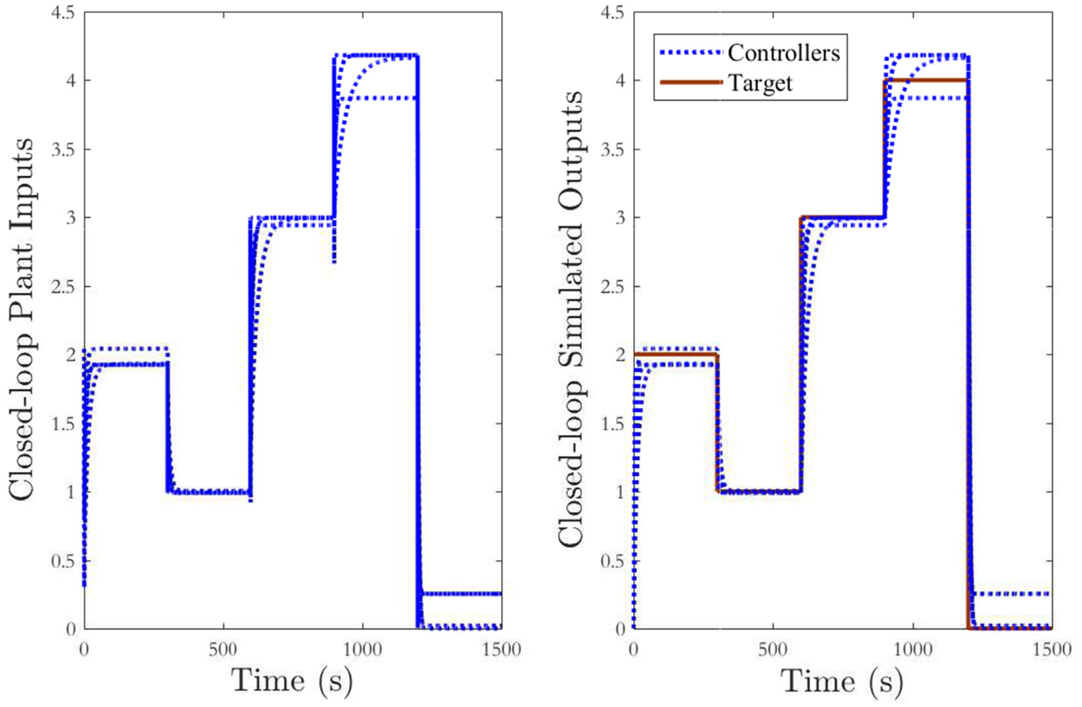

Fig. 15. Closed-loop simulated inputs (left) and outputs (right) of the nonlinear valve platform by the static controllers generated by SRBCD in Table 2.

Table 3
Sample controller equations found for the nonlinear valve by ELGP for tracking.

$G_{c1}: u = 0.6020\,\epsilon(t) + 0.0248\int\epsilon(t) + 0.0744\,\mathrm{d}\epsilon/\mathrm{d}t - 0.0248\sin^2\epsilon(t) - 0.0248\sin\epsilon(t) + 0.3319$
$G_{c2}: u = 0.0248\int\epsilon(t)\mathrm{d}t + 0.3211$
$G_{c3}: u = 0.5659\,\epsilon(t) + 0.0249\int\epsilon(t)\mathrm{d}t + 0.3281$
$G_{c4}: u = 0.5642\,\epsilon(t) + 0.0249\int\epsilon(t)\mathrm{d}t + 0.0663\,\mathrm{d}\epsilon/\mathrm{d}t + 0.3301$
$G_{c5}: u = 0.5678\,\epsilon(t) + 0.0249\int\epsilon(t)\mathrm{d}t + 0.0498\,\mathrm{d}\epsilon/\mathrm{d}t + 0.3288$

all linear controllers comprising proportional, integral and/or derivative components of the tracking error. As hypothesized, the closed-loop responses contain no steady-state error.

To evaluate the disturbance and noise rejection capacity of these controllers, impulse disturbances and noise were added to the output. The closed-loop plant input and output by controller G_{c2} in Table 2, as a sample, with disturbance and noise are shown as the left and right plots of Fig. 17, respectively. The results indicate the effectiveness of this controller in presence of both disturbance and output noise.

4.3. Case III: inverted pendulum

As a study platform, the noted feature of the inverted pendulum is its non-minimum phase characteristic. Furthermore, its regulation is challenged by the inability to accommodate initial conditions by the input estimation method. The model of the pendulum considered here has the form [3]

$$\dot{x}_1 = x_2, \dot{x}_2 = (\sin x_1 + u\cos x_1) \tag{11}$$

where x_1 represents its angular position and x_2 its angular velocity.

To circumvent modeling a desired plant output to an initial displacement, as a natural target output for a regulator, we consider a desired impulse response of the pendulum as the target output. To model this target output one could design a regulator and use the regulated response of the pendulum to the impulse as the target response of the plant. Alternatively, one could consider a reference model to generate the target output. Shown in Fig. 18 are four such target outputs. One target output is the closed-loop output of the pendulum with a linear controller of the form [3]

$$u = -3x_1 - 2x_2$$

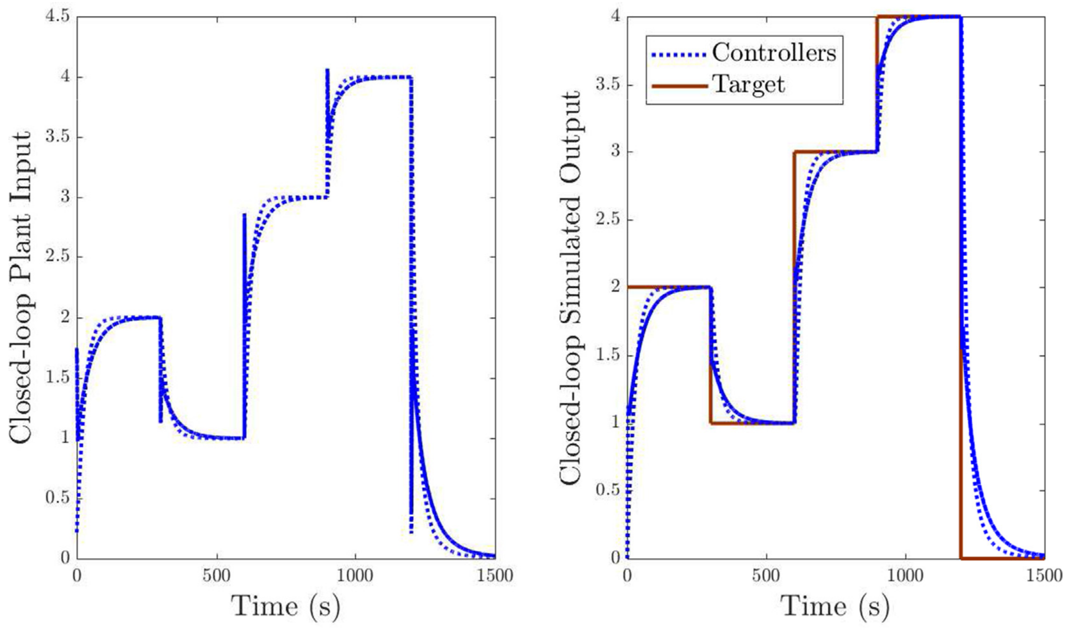

Fig. 16. Closed-loop simulated inputs (left) and outputs (right) of the nonlinear valve platform by the dynamic controllers generated by SRBCD in Table 3.

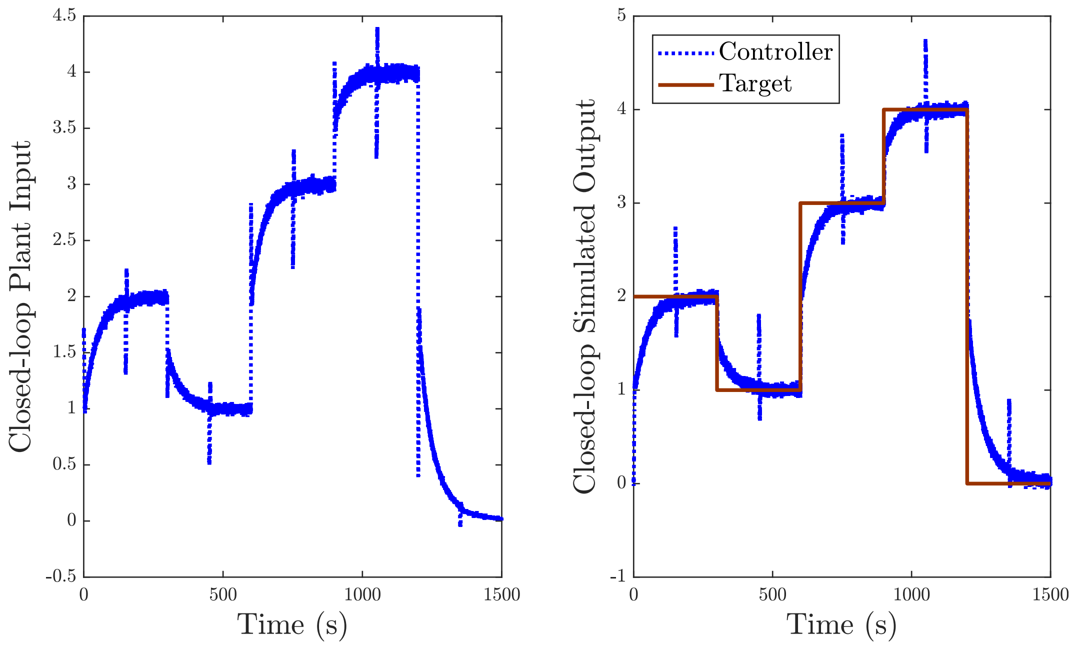

Fig. 17. Closed-loop simulated input (left) and output (right) of the nonlinear valve platform by controller G_{c2} in Table 3 in presence of impulse disturbances and output added noise.

which assignes the closed-loop poles of $-1 \pm j$. The second target output is the closed-loop response of the pendulum with a feedback linearized controller of the form [3]

$$u = \frac{1}{\cos x_1}[-\sin x_1 - 2(x_1 + x_2)]$$

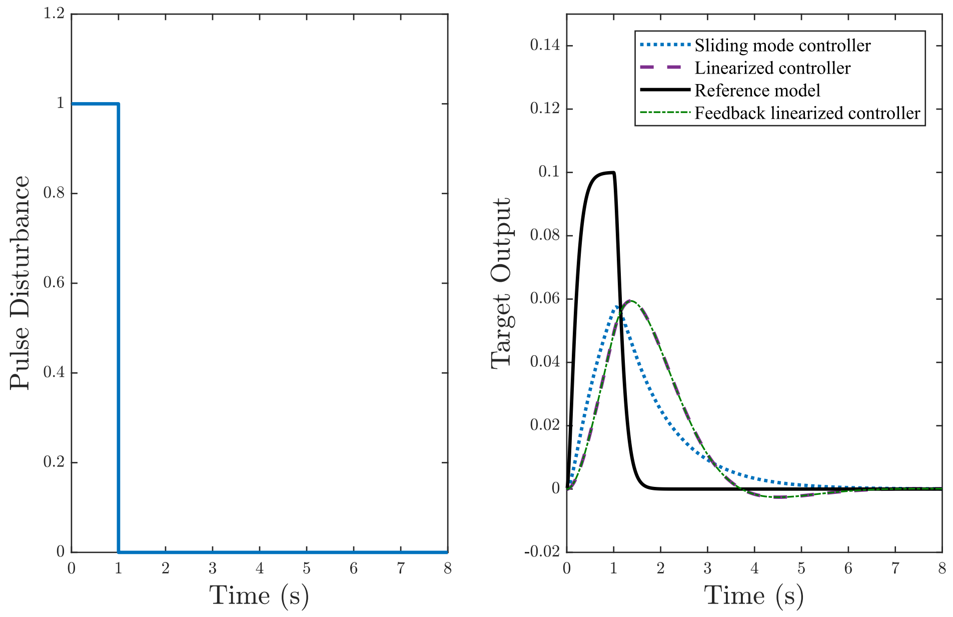

Fig. 18. The disturbance (left) used to generate the target output for the inverted pendulum according to a reference model and three controllers (right).

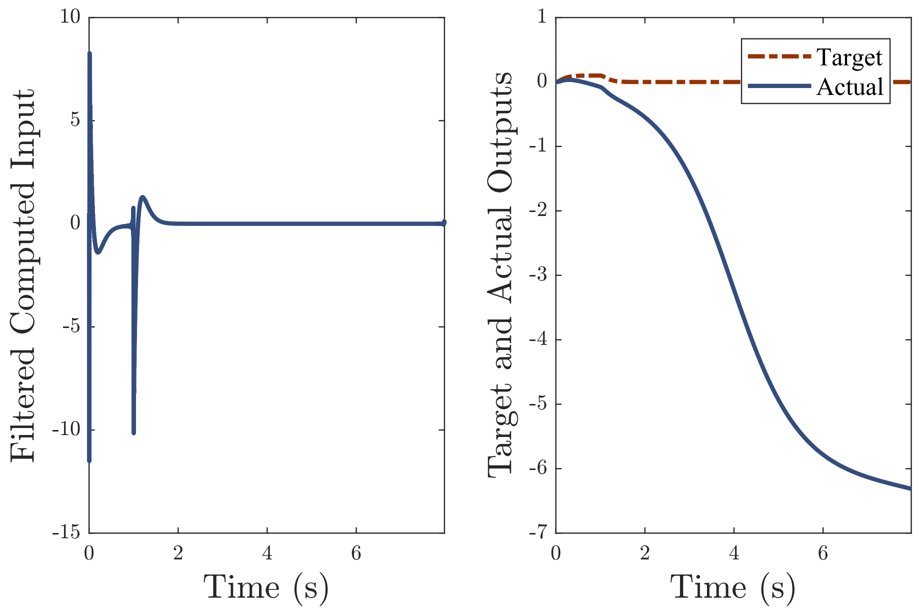

Fig. 19. The computed input (left) and the output it generates from the nonlinear inverted pendulum (right) relative to the target output.

The third target output is that of a sliding mode controller with the form [3]

$$u = -\beta(x)\,\mathrm{sat}\left(\frac{x_1 + x_2}{0.1}\right), \quad \beta(x) = 2\left|\frac{x_2}{\cos x_1}\right| + |\tan x_1| + 1$$

and the fourth target output is the response of a linear reference model with the transfer function

$$H(s) = \frac{10}{s^2 + 20s + 100}$$

The target outputs generated by these four different systems are shown in the right plot of Fig. 18, which are in response to the disturbance input shown in the left plot.

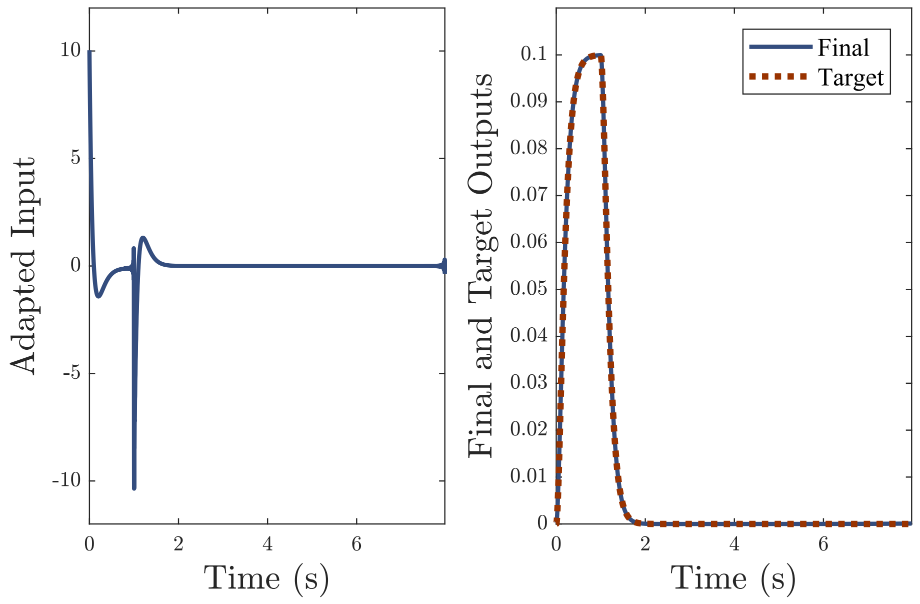

Fig. 20. The adapted input (left) and final output of the nonlinear inverted pendulum after adaptation (right), shown with the target output.

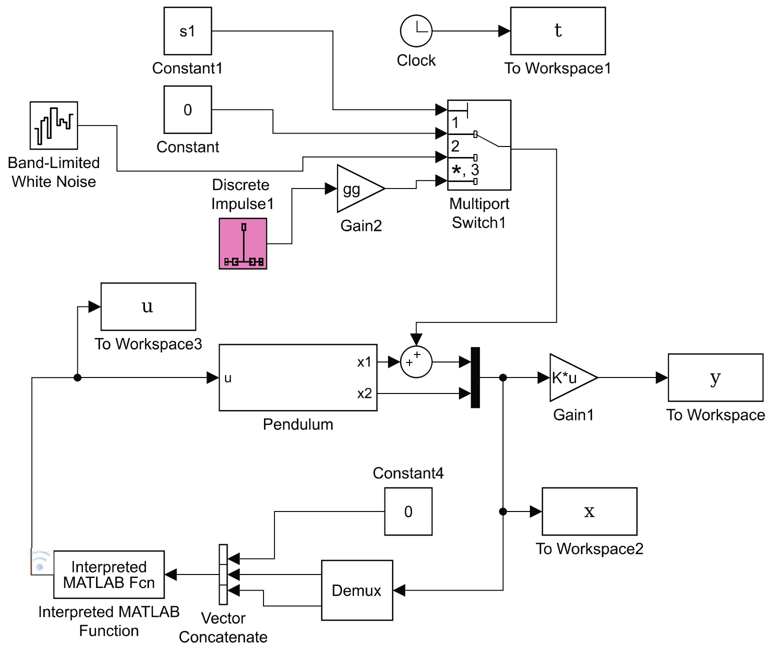

Fig. 21. Simulink^{TM} model masking the input disturbance to the controller by a zero constant for simulating the closed-loop response of the nonlinear inverted pendulum to an initial displacement.

Table 4
Sample controller equations found for the inverted pendulum by ELGP to mimic the response of the linear controller.

$G_{c1} : u = -35.63\sin(2x_1(t) - x_2(t)(4x_1(t) + x_2(t) - 0.34) + d(t)x_1(t) + (d(t) - (2.23\,d(t)x_1(t))(x_2(t) - 0.38))$
$G_{c2} : u = 28.61\,d(t) - 100x_1(t) - 19.85x_2(t) - 18.67\,d(t)^2$
$G_{c3} : u = 81.23\sin(0.25x_1(t) - 0.12\,d(t) + 0.25x_2(t) + \sin(x_1(t))$
$G_{c4} : u = 10d(t) - 101.06x_1(t) - 20x_2(t)$
$G_{c5} : u = d(t) - 10.34x_1(t) - 2x_2(t)$

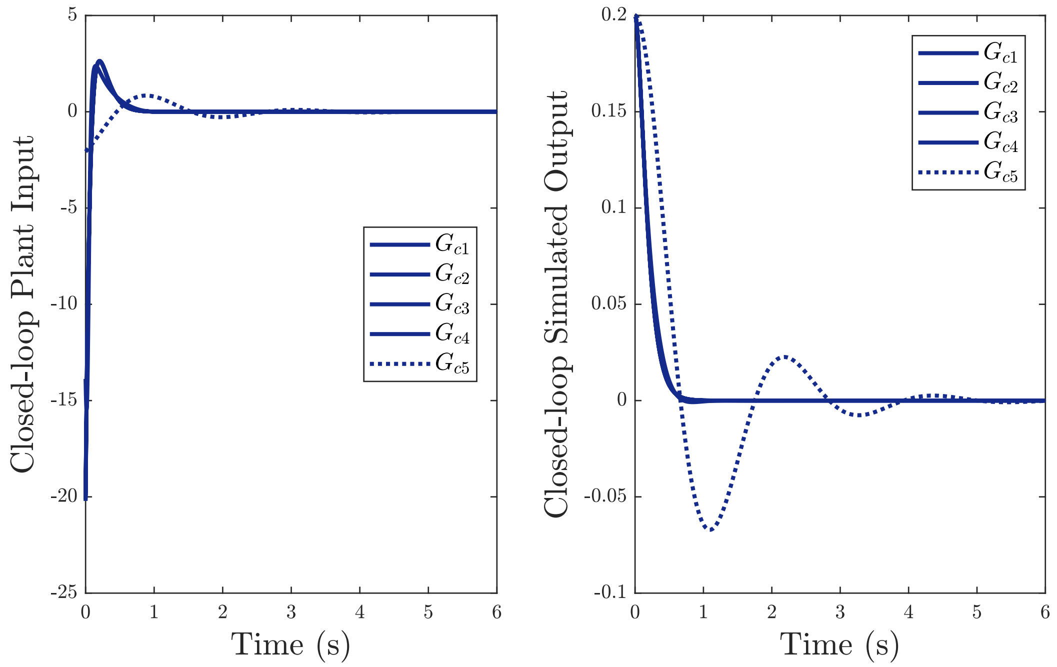

Fig. 22. Closed-loop simulated input (left) and output (right) of the nonlinear inverted pendulum by the controllers in Table 4 in response to an initial displacement.

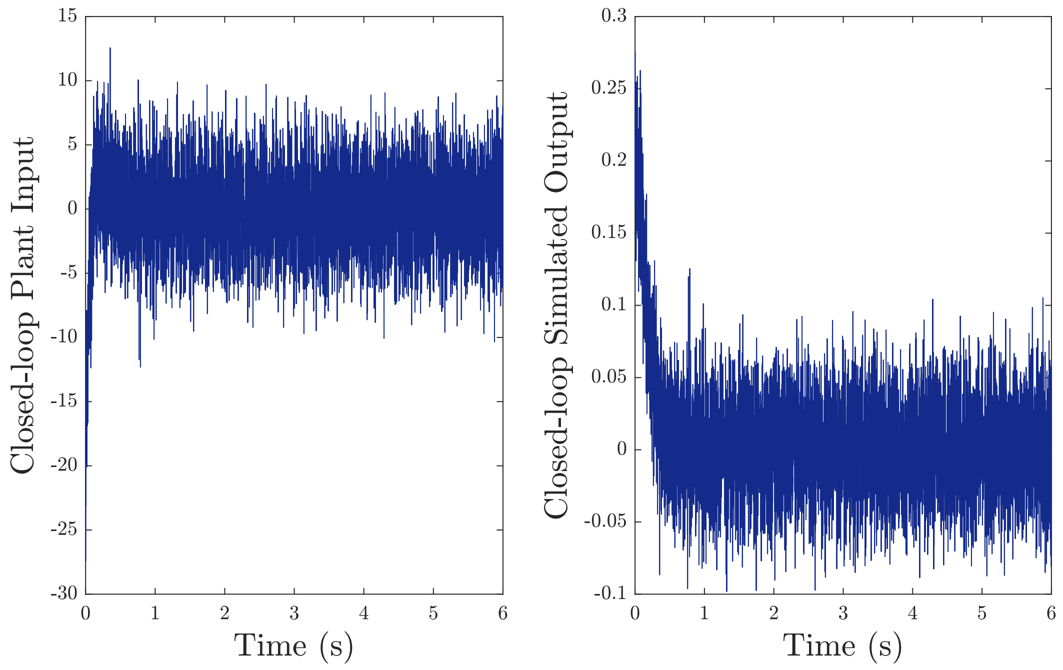

Fig. 23. Closed-loop simulated input (left) and output (right) of the nonlinear inverted pendulum by controller G_{c4} in Table 4 in response to an initial displacement with added output noise.

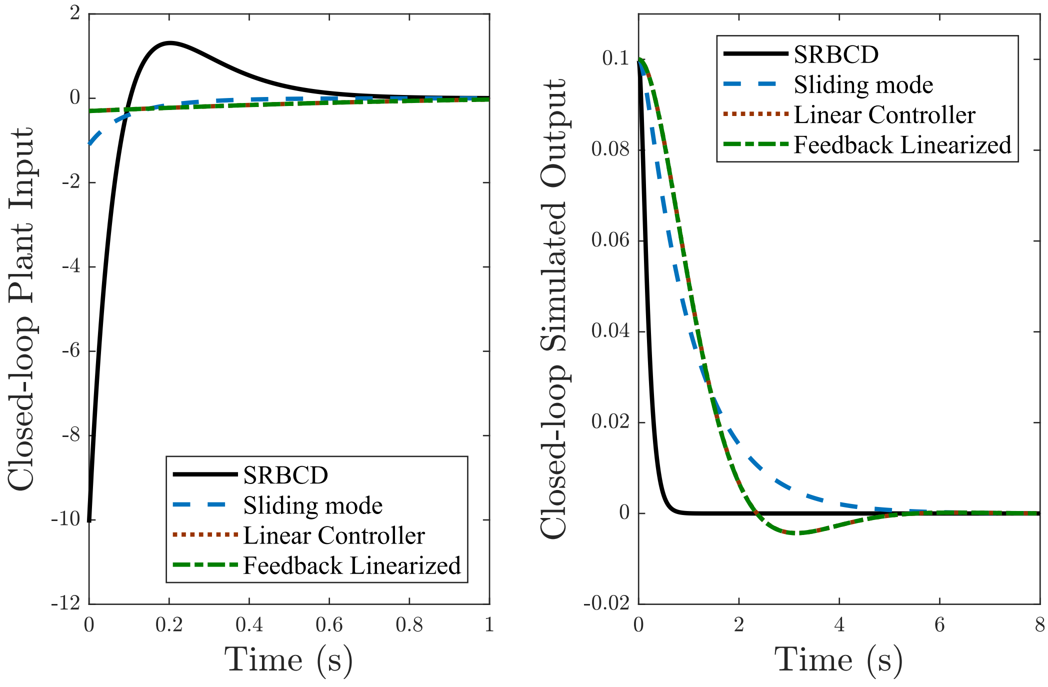

Fig. 24. Closed-loop simulated inputs (left) and outputs (right) of the nonlinear inverted pendulum by G_{c1} in Table 4 and three other controllers in response to an initial displacement.

4.3.1. Case III: input estimation

Using as target output the reference model output in Fig. 18, the input was computed according to Eq. (2) with the linear model

$$\frac{y(k)}{u(k)} = \frac{0.5e^{-7}q^{-1} + 0.5e^{-7}q^{-2}}{1 - 2q^{-1} + q^{-2}}$$

which is the discretized transfer operator at the sampling frequency of 1000 Hz of the transfer function

$$G(s) = \frac{1}{s^2 - 1}$$

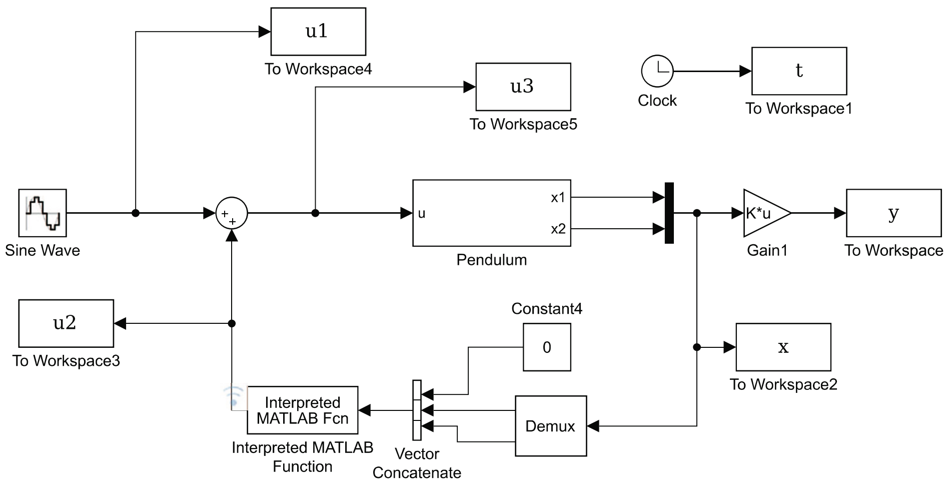

Fig. 25. Simulink$^{\text{TM}}$ model used to simulate the frequency response of the closed-loop system for the nonlinear inverted pendulum.

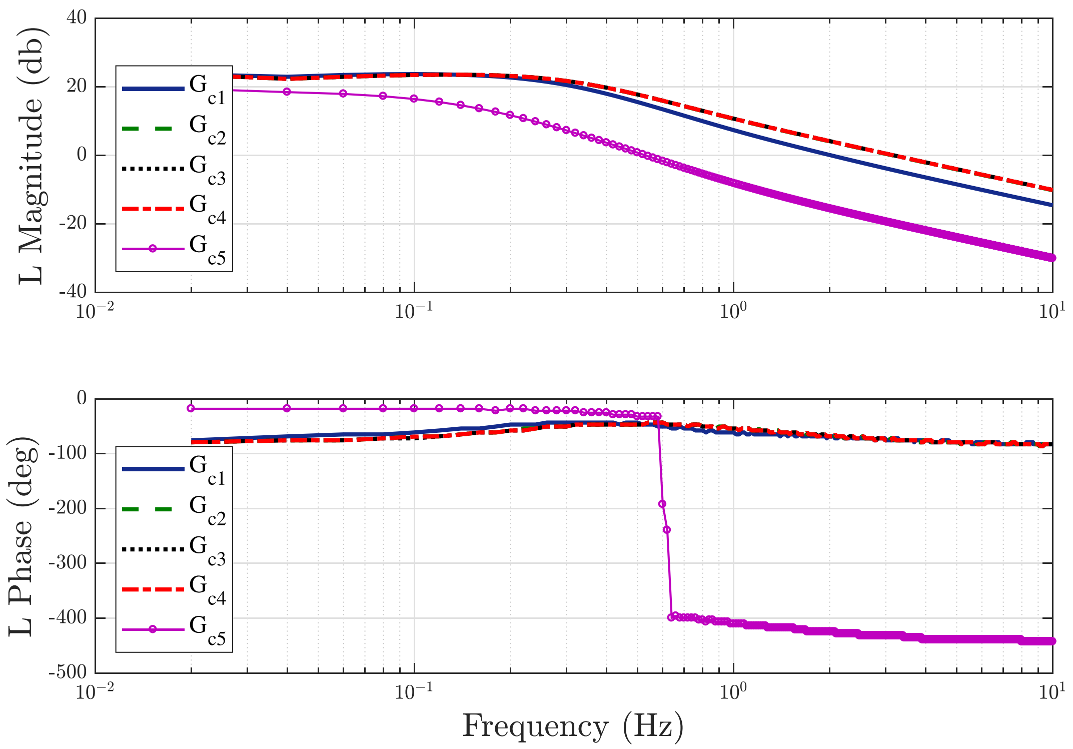

Fig. 26. Loop gain SIDFs of the inverted pendulum with the controllers in Table 4.

The computed input is shown in the left plot of Fig. 19. As expected, this input, which was obtained by a linearized model of the pendulum, produces an unstable response from the nonlinear model, as shown in the right plot of Fig. 19.

Adapting the computed input in Fig. 19 according to Eq. (7) (with $k_p = 0.05$ in Eq. (7) and the adaptation limit of $\sum_{k=1}^{N}|\epsilon(k)| < 8$ in Eq. (6)) produced the adapted input shown in the left plot of Fig. 20 with an output of the nonlinear pendulum very close to the target output, as shown in the right plot of this figure.

4.3.2. Case III: controller forms

Considering the impulse response as the target output poses an interesting dilemma. The impulse function needs to be represented as a regressor to symbolic regression to match the target output, yet it is unrealistic to have access to a disturbance in practice. The remedy adopted here is to use the impulse for formulating the controller equation, but to not provide that as an input (by using a zero value) to the controller in closed-loop, as depicted in the simulation model shown in Fig. 21. The logic here is that although the impulse is needed to model the plant input for the desired impulse response of the pendulum, it is the relationship with the other regressors that is essential for regulating the pendulum.

Using the adapted input in the left plot of Fig. 20 as the target of SR, and the disturbance $d(t)$ (shown in the left plot of Fig. 18), the position of the pendulum x_1 and its velocity x_2 as regressors, several equations were obtained by ELGP, as listed in Table 4. Three of the controllers: G_{c2}, G_{c4}, and G_{c5} are linear controllers, once one ignores the terms associated with the disturbance $d(t)$. In fact, two of the controllers: G_{c2} and G_{c4} are nearly identical in their coefficient values as well, as they appear to have much larger coefficients than the third linear controller G_{c5}.

The closed-loop plant inputs and outputs with the controllers in Table 4 in response to an initial displacement are shown in the left and right plots of Fig. 22, respectively. The effectiveness of the controllers in regulating the pendulum against the initial displacement gives credence to the logic of masking the impulse originally used for modeling the controllers. It is interesting to note that the first four controllers $G_{c1} - G_{c4}$ in Table 4 have almost identical performances, differing considerably from that of G_{c5}, which has much slower dynamics.

To evaluate the noise tolerance capacity of the controllers, band-limited white noise was added to the output, as shown in Fig. 21. The closed-loop plant input and output with the added output noise using controller G_{c4} in Table 4 are shown in Fig. 23. The results indicate the controller's ability to cope with output noise.

As a further evaluation of SRBCD, the closed-loop plant input and output of the pendulum to an initial displacement with controller G_{c1} in Table 4 are compared with their counterparts by the three controllers discussed earlier in Fig. 24. The results

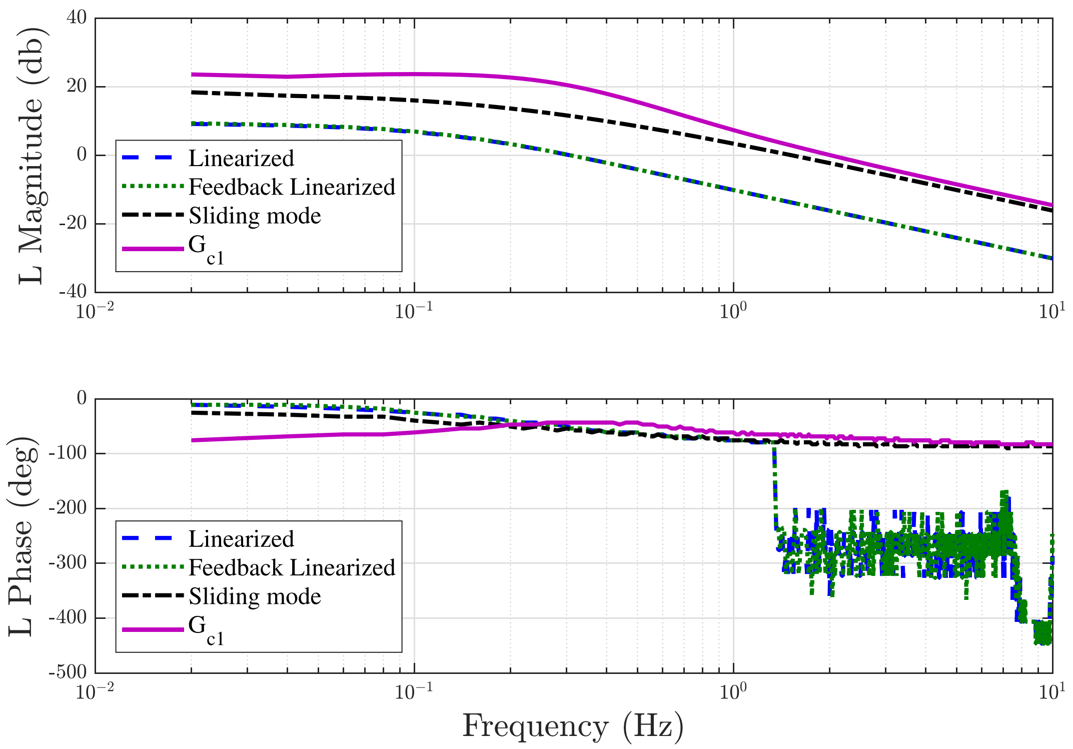

Fig. 27. Loop gain SIDFs of the inverted pendulum with controllers designed by linearization, feedback linearization, sliding mode, and the controller G_{c1} in Table 4.

show the faster response of the closed-loop system with the controller by SRBCD, as dictated by the faster response of the reference model (see Fig. 18) characterizing its performance.

5. Stability analysis

For illustration purposes, stability analysis is performed on the inverted pendulum for which several controllers are available for comparison. The input/output (IO) sinusoidal-input describing functions (SIDFs) (see D for further explanation) were obtained via simulation for the closed-loop systems comprising the controllers of the inverted pendulum in Table 4. To ensure steady state response, the last twelve cycles of each output were used to compute their Fourier transforms. The magnitude of each IO SIDF was then estimated as the ratio of the peak of the output and input Fourier transforms. For the IO SIDF phase, the peaks of the last twelve cycles of each input and output sinusoidal time series were located and the phase was obtained as the average of the phase difference between the IO peak locations.

To estimate the SIDFs approximation of the loop gain for each closed-loop system, a sinusoid was applied as input to the closed-loop system, and the controller output (u2) and the plant input ϵ (u3 $\epsilon = r - y$) were obtained from simulation, as shown in Fig. 25. The SIDF for the loop gain L was obtained between the plant input u3 and the controller output u2.

SIDFs were estimated for the loop gain L of the five closed-loop systems comprising the controllers in Table 4. The SIDF approximations are shown in Fig. 26, indicating the similarity of those associated with controllers G_{c1} - G_{c4}. The SIDF of L associated with G_{c5} is clearly the outlier, as is its closed-loop response in Figure 22. According to the loop gain SIDFs, all five systems are stable with the first four having infinite gain margins and phase margins of about 100°. The last controller G_{c5} provides a much smaller bandwidth (0.5 Hz vs. 20-30 Hz by others), and a very small gain margin.

The staibility of the controllers evaluated by the SIDFs in Fig. 26 would find better context when compared with the SIDFs of more traditional controllers. To this end, loop gain SIDFs were obtained for the three other (linearized, feedback linearized, and sliding mode) controllers available for the inverted pendulum [3]. The SIDF estimates of the loop gain L of the closed-loop systems comprising these controllers are shown in Fig. 27 together with that of the controller G_{c1} in Table 4. The L magnitude and phase of the sliding mode controller are are very similar to those of G_{c1}, having an infinite gain margin and a phase margin of about 90°. Both the linearized and feedback linearized controllers provide a much smaller bandwidth (as dictated by placement of the closed-loop poles in their design), providing a phase margin of about 120°, but a gain margin of 10 db.

6. Discussion

The two components of SRBCD are input estimation/adaptation and controller construction (see Appendix C for more details). Some of the issues observed during the implementation of these components are briefly discussed below.

6.1. Plant input estimation/adaptation

The input adaptation strategy proposed here is inspired by ILC [20] which uses a filter to contain the propagation of the error. Our estimation method, instead, uses the linearized model of the plant for input estimation/adaptation. Although the two methods differ in the underlying filter they use, they appear to have similar learning dynamics [19]. Some of the observed issues associated with input estimation are discussed below.

- **Difficulty with initial conditions.** The premise of computing the input according to Eq. (2) is that the input is the sole cause of the output. As such, the input estimation method presented here is unapplicable to target outputs that are affected by initial conditions. Even though we have avoided the use of initial conditions in input estimation (for instance, by using an external disturbance to generate the target output for the inverted pendulum), initial conditions cannot always be avoided. A case in point is when the model is linearized about a non-zero operating point., as in magnetic levitation [3], for example, where the operating point constitutes the initial condition for the model and a cause of the output. While we expect coordinate transformation to provide a remedy in such cases, we have not yet examined its applicability in SRBCD.
- **Choice of the target output.** To define any target output for the plant and find the input that would generate it seems too good to be true. Intractable outputs would often result in unreasonable inputs with either large magnitudes or shapes that are difficult to match by equations in terms of the available regressors. As such, a search for the suitable target output may need to be conducted before one is chosen.
- **Sensitivity to sampling frequency.** Although we have not performed a thorough investigation of the effect of sampling frequency, we have observed its adverse effect on input convergence when selected too low.
- **Sensitivity to the adaptation gain k_p.** The success of the adaptation routine (7), similar to nonlinear least squares that also relies on a first-order approximation, depends upon the convexity of the error surface with respect to the input. As such, the convergence rate of input adaptation is dependent on the correction size, as dictated by the magnitude of k_p. Although we have not performed a methodical study of its effect, we have found larger k_p values to derail input adaptation.

6.2. Controller construction

The equations by ELGP are in terms of a number of variables, obtained in open-loop, to match a target (plant input). Therefore, the ability of ELGP to define an equation for the plant input in terms of the regressors depends on not only the regressors but also the functions available to regression. Furthermore, those equations are successful as controllers which have as inputs the variables that efficiently guide the control objective; i.e., regulation or tracking. Some of our observations are discussed below.

- **Ability to duplicate the plant input.** The quality of equations generated by ELGP is evaluated by a fitness measure. Fitness is defined in our study in terms of shape conformity of each equation's output to its target (plant input), quantified by the correlation coefficient and the sum of the errors between them. There are cases, however, that despite a high fitness measure of, say, 0.98, one finds the said equation deficient in producing the target output. An example of such a case is shown in Fig. 28, where the difference between the equation's output and its target at the initial stage of the cycle makes the equation ineffective as a controller.
- **Choice of regressors.** The regressors, apart from their role as the basis of equations, ought to be suitable inputs to the controller in closed-loop. As such, it may be necessary to exclude similar-shape regressors that may end up replacing necessary closed-loop controller inputs. The case in point is the set of regressors selected for the nonlinear valve platform. There we chose to exclude the plant output as a regressor, because it happened to replace the reference input, due to its better agreement with the shape of the plant input. But not having the reference input available to the controller made it oblivious to it, hence, dysfunctional as a tracker. Aside from this rationale for selecting the regressors, there is a level of chance associated with the mix of regressors. For example, there are a total of five possible variables from the nonlinear valve platform, shown in the right plot of Fig. 29, that could be used as regressors for the target plant input, shown in the right plot of this figure. No valid controllers could be obtained from this mix of regressors, because ELGP found easiest to match the shape of the plant input with combinations of x_2 and x_3. But any equation independent of the reference is unsuitable as a tracking controller, so we had to test ELGP with different combinations of regressors to find the proper mix for delivering a tracking controller.

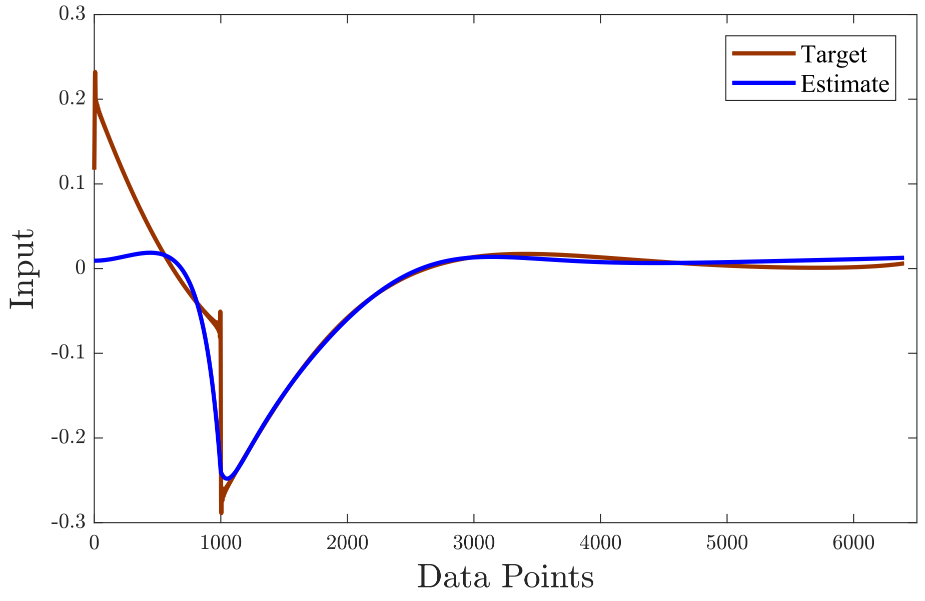

Fig. 28. Example of a deficient candidate controller output not adequately matching its target.

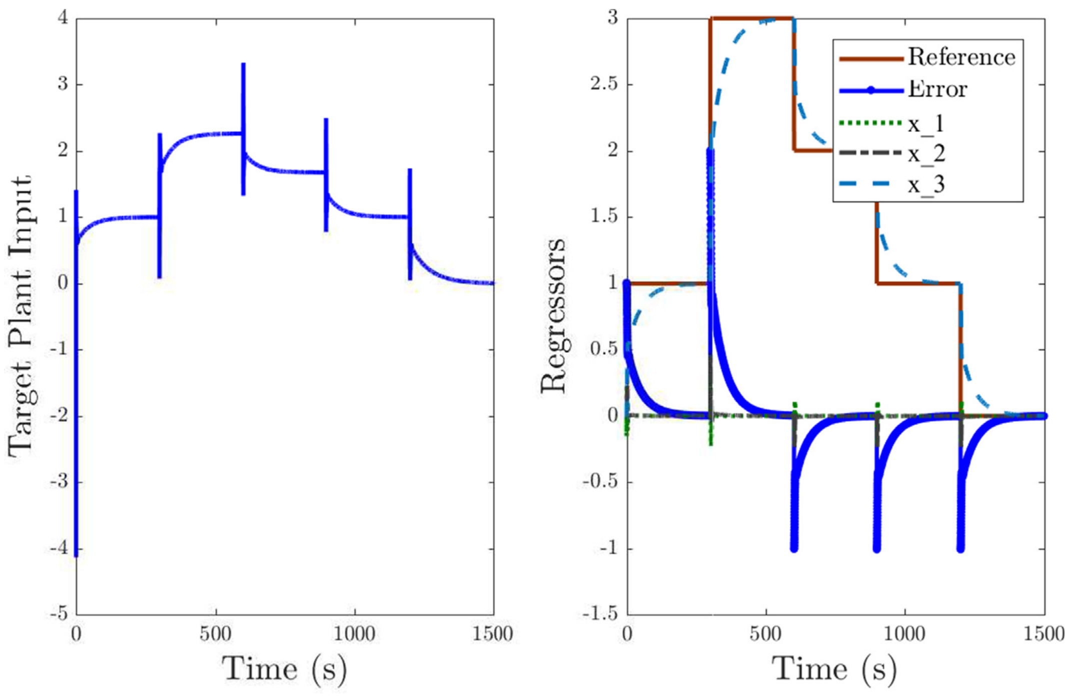

Fig. 29. All of the variables from the nonlinear valve platform (right) that could be used as regressors for generating an equation for the target plant input (left).

7. Future work

The proposed SRBCD method is in its early stages of development, so its results, although promising, cannot yet address all the issues of concern in a practical control method. Our thoughts about addressing some of these issues are expressed below.

- **Robustness analysis of the controllers.** The controllers produced by SRBCD are tested on plants having the same parameter values as during input estimation and controller construction. It behooves us, therefore, to evaluate the robustness of these controllers to parametric uncertainty and/or variability [25]. Given that the SRBCD-generated controllers are model-based, one could potentially analyze the stability of the closed-loop system analytically. The alternative approach that we are considering is robustness analysis through Monte Carlo simulation.
- **Performance in presence of output noise.** Although we did conduct in this paper a cursory evaluation of the controllers' performance in presence of added output noise, the evaluation is a bit idealistic by assuming the states different from the output to be measurable independent of noise. A more realistic evaluation is, therefore, needed wherein state observers are used and the effect of noise is mitigated by the application of low-pass filters.
- **Considering multiple targets for controller design.** Deriving a controller equation based on a single target is like training a car driver based on a single turning maneuver. Although it is remarkable and reassuring that such single-target equations can function as controllers and respond to inputs and disturbances for which they are not trained, it makes one wonder if their performance can be further improved when trained on multiple targets.
- **Application to MIMO systems.** The input estimation method was demonstrated for single-input single-output cases, so it is a natural next step to evaluate its applicability in multi-input multi-output cases.
- **Exploring the applicability of control tuning in adjusting the parameters of SRBCD-generated controllers.** As is observed from Tables 1–4, the controllers include very specific parameter values, making one wonder if the viability of these controllers is as dependent on these parameter values as it is on their form. To address this question, we plan to apply iterative feedback tuning [40–43] to explore the effect of parameter adaptation on the controllers' performance.

8. Conclusion

A novel method of empirical controller design is introduced with the potential to produce unique controller forms by symbolic regression (SR). To make SR applicable in controller design, time-consuming closed-loop evaluation of controller candidates is replaced by algebraic evaluation. For this purpose, the desired plant input is estimated and used as target for derivation of candidate controllers, testing, in effect, the following hypothesis: *Given the availability of the plant input that generates the desired plant output (i.e., inverse solution), whether the equation that generates this plant input could qualify as a controller; or stated differently, whether an equation developed to produce a desired plant input in terms of the loop variables can potentially function as a controller in closed-loop.* To test this hypothesis, the paper offers an input estimation/adaptation method to estimate the open-loop plant input for a desired plant output (inverse solution). It then uses ELGP for symbolic regression (SR) to produce concise equations for the plant input in terms of the plant states and other loop variables. It ultimately tests as controllers in closed-loop the produced equations by SR to evaluate their validity. The viability of the method and its underlying hypothesis are tested in application to three nonlinear plants.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The development of ELGP was partially supported by the NSF-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant No. 1068864).

Appendix A. Controller form and transparency

When controllers have a legible form they are called transparent, because they inform how their output is computed in terms of the controller inputs. For example, with $U(s)$ representing the Laplace transform of the controller output and $E(s)$ the Laplace transform of the controller input, if the controller is a lead compensator: $U(s)/E(s) = k_1(s+z)/(s+p),\ z < p$ or if it is in the form of proportional plus integral plus derivative (PID) controller: $U(s)/E(s) = k_p + k_i/s + k_d s$, it is clear what the order of the controller is, how its output $u(t)$ is computed as a function of its inputs, and how much weight is given to each one of its constituents. We refer to this clarity of structure as transparency. In contrast, one may consider a neuro-controller [6–9], as depicted in Fig. A1. It consists of numerous neurons and the connection weights between them. The controller outputs, therefore, cannot be distinctly associated with individual inputs, precluding understanding of the controller form and how its outputs are weighted in relation to its inputs.

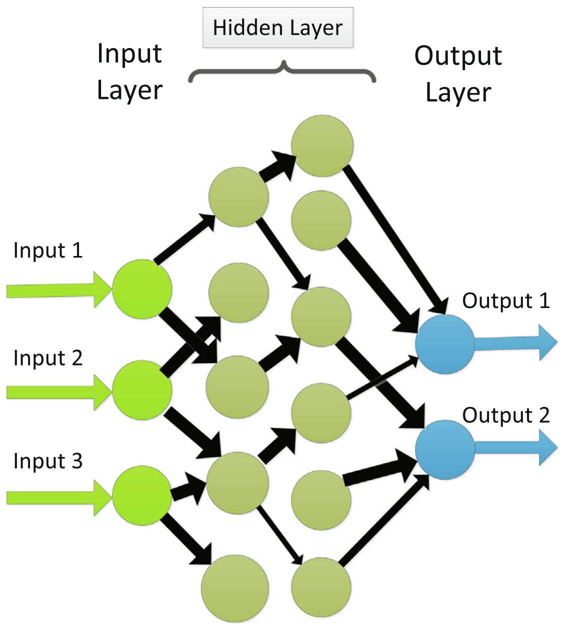

Fig. A1. A simple multi-layer neural network, with only a few of connections drawn.

Appendix B. A brief overview of symbolic regression

Symbolic regression (SR) is a form of regression that attempts to estimate a model's structure, as well as its parameters, starting with basic building blocks like mathematical functions. SR is typically conducted using genetic programming (GP) [44], which is a bio-inspired machine learning technique that constructs a population of candidate models from mathematical building blocks and then selects and varies these models over several generations before converging on a model that best fits the target (plant input, in this case). Due to an expansive search space that includes model structures in addition to parameters, SR can be computationally intensive in comparison to other empirical methods. SR attempts to solve the problem

$$\text{minimize } \mathrm{F}(\mathrm{M}_\theta, \mathcal{T}) \text{ subject to } \mathrm{M} \in \mathfrak{S} \tag{S1}$$

where M_θ is a model parameterized by θ from the space of possible models $\mathfrak{S}$, and F denotes a minimized fitness function over training examples $\mathcal{T}$. Given that it is impractical to exhaustively search $\mathfrak{S}$, the model found to minimize $\mathrm{F}(\mathrm{M}_\theta, \mathcal{T})$ may only be locally optimal, but it is assumed that such a model can nevertheless fulfill the purpose of adequately representing the process (controller, in this case), as depicted by the target. In the field of SR, stochastic optimization (i.e. metahueristic) techniques like GP have been successful [45]. These techniques are named for their incorporation of some randomness into the search process, thereby allowing the discontinuous changes needed for exploring equation forms. A number of other techniques fall under this umbrella as well, including hill climbing, simulated annealing, and particle swarm optimization [45].

GP is a population-based metaheuristic strategy that uses principles of biological evolution to define the ways in which candidate solutions are updated. Although evolutionary problem-solving strategies were discussed in the sixties [46] and seventies [47], the canonical GP used widely today was developed by John Koza [44] in early nineties (preceded by Cramer [48]), and many of his research practices continue to be popular. The overall process of SR is visualized in Fig. B1. In GP, model development is achieved using an evolutionary scheme in which a population of different computer programs are tested for their fitness according to one or several measures of their quality, and go through a process of Darwinian selection and variation to form the next iteration (generation) of candidate solutions. In SR the programs consist of building blocks (nodes) of equations, for example $\{+, -, *, /, \mathbf{x}, \mathbf{y}, \mathbf{z}\}$ where the variables $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ are represented by their numerical values (time-series, in our case). The blocks are then constructed into a graph, most commonly a tree. Candidate solutions in the population are probabilistically selected for survival each generation based on their fitness. Variation is introduced to the search process by mutation and recombination (i.e. crossover) of subprograms in chosen solutions each generation. The search is thus driven by selection and variation: the selection process allows for exploitation of promising solutions while variation mostly serves to explore the neighborhood of promising solutions. This generational process is repeated until an adequate solution is found.

GP and variants of it have been applied successfully for system identification in many fields, including climate modeling [35], robotics [49], wind turbine mechanics [50], physics [30], and biology [51]. It has become a popular method especially for finding nonlinear differential equation solutions [10,11,29,30], in addition to finding closed-form, analytical solutions to

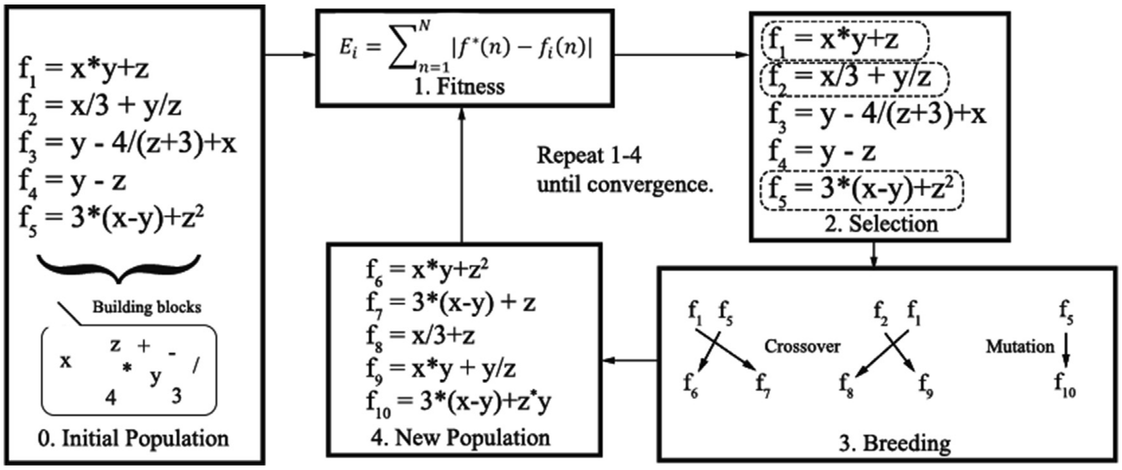

Fig. B1. Genetic Programming as applied to symbolic regression.

them [52,53]. Furthermore, GP has been used to successfully design systems and solve problems, and has produced human-competitive results in various fields, including antenna design [54], photonics [55], circuit design [56], game play [57], finite algebras [58], and quantum computer programming [59].

Appendix C. Implementation and algorithmic features

The design process by the SRBCD method, illustrated in Fig. 1, consists of two main steps: (1) input estimation, and (2) symbolic regression by ELGP. The necessary information for each step is as follows:

- **Input Estimation**

1. The target output that is computed, for example, as the impulse or step response of a linear system, or as the closed-loop response of the controlled linearized model.
2. A linearized model of the plant for input computation and its subsequent adaptation (see Fig. 2).
3. Open-loop simulation of the plant, for simulation-based implementation, or the plant and required instrumentation, for real-time acquisition of open-loop plant output for evaluation of the plant input after each adaptation iteration.
4. Specification of k_p and error threshold in Eq. (7).

- **Controller Construction**

1. The plant input obtained from the input estimation step, to be used as the target of symbolic regression.
2. The regressors in the form of time-series obtained from open-loop simulation of the plant subject to the desired/adapted plant input.
3. Selection of configuration parameters for ELGP, such as the number of generations, the size of population, fitness type, etc. The ELGP algorithm is available on-line at [60].

As far as computation time for each step, estimating the input for the inverted pendulum using Matlab™ and Simulink™ on a regular PC took 2671 adaptation steps and 594 s. ELGP took about 29 min and 4 s to evaluate 3 million equations over the course of 1000 generations for finding a candidate equation for the inverted pendulum.

Appendix D. A brief overview of sinusoidal-input describing functions

A describing function can approximate, under certain conditions, the frequency response of a nonlinear system [2]. The fundamental assumption of the describing function is that the higher harmonics generated by the nonlinear element from the sinusoidal input be filtered by the system linear elements, so that only the fundamental frequency contained in the input be considered for the analysis. As expected, both the magnitude of excitation and its frequency affect the output response, but more so the magnitude. For illustration purposes, shown in Fig. D1 are samples of the closed-loop output in response to a sinusoidal input. The output on the left, in response to a lower frequency excitation of the same magnitude as on the right, is nonharmonic, whereas the one on the right, in response to a higher frequency excitation, is harmonic. The response on the

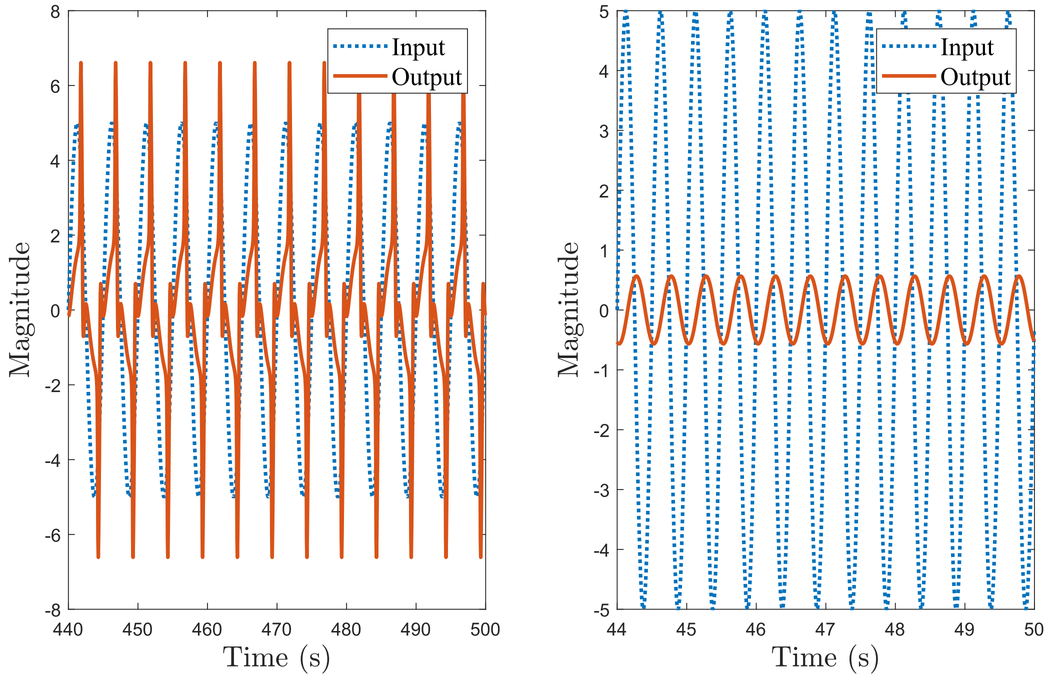

Fig. D1. Samples of outputs in response to a hormonic excitation. The left plot is a nonharmonic response disallowing the estimation of SIDFs.

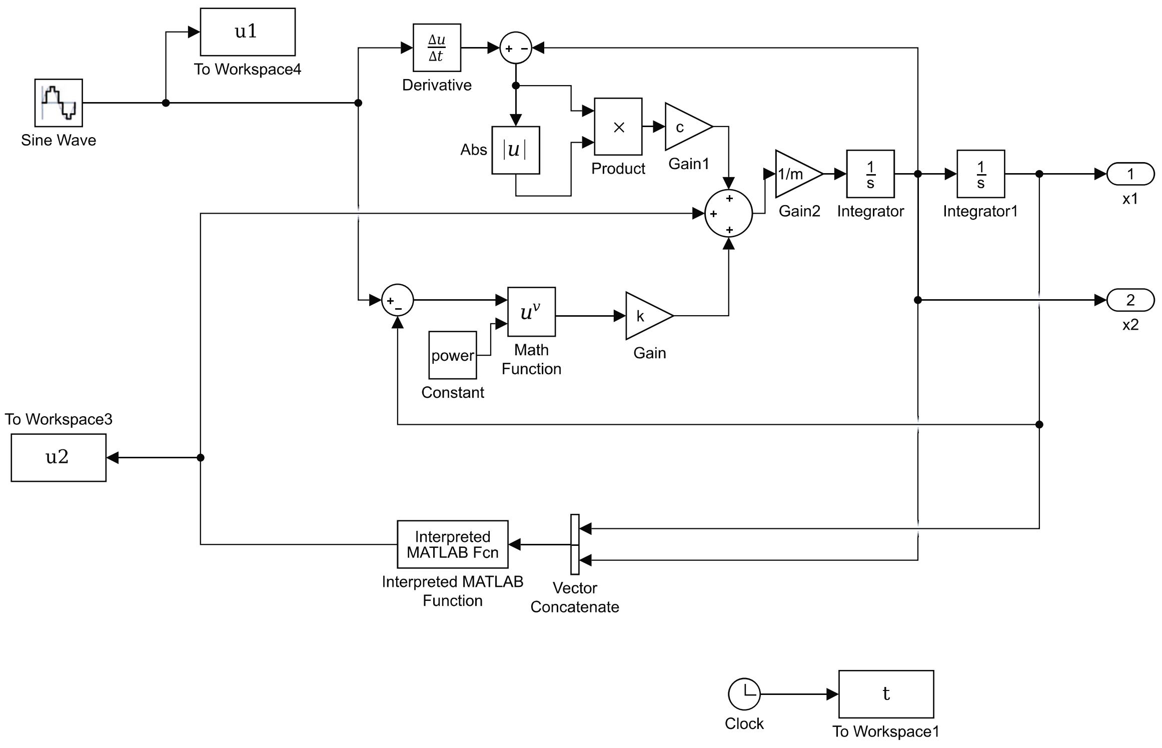

Fig. D2. Simulink^{TM} model for evaluating the controller obtained by SRBCD for the active suspension system.

left would become harmonic with a lower magnitude excitation, and the one on the right would become nonharmonic with a higher magnitude excitation. The SDIFs were obtained for excitation magnitudes that generated harmonic excitation at all frequencies.

The objective is to estimate describing functions that approximate the loop gain, L for frequency domain stability analysis [61]. However, it is not always possible to readily estimate the describing functions. An example of such a case is the nonlinear active suspension in Eq. (8). As depicted in the Simulink^{TM} model of Fig. D2, the ratio of no pair of variables available from the simulation (e.g., u1, u2, x1, and x2) would approximate the loop gain.

References

[1] G.F. Franklin, J.D. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems, seventh Edition., Pearson Education Inc, Upper Saddle River, New Jersey, USA, 2015.
[2] J.-J.E. Slotine, W. Li, Allied Nonlinear Control, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.
[3] H.K. Khalil, Nonlinear Control, Pearson Education Inc., 1 Lake St., Upper Saddle River, NJ 07458, 2015..
[4] M. Krstic, I. Kanellakopoulos, P. Kokotovic, Nonlinear and Adaptive Control Design, John Wiley and Sons, New York, NY, USA, 1995.
[5] S.S.F. Wata, R. Langari, D.P. Filev, Fuzzy Control: Synthesis and Analysis, first ed., John Wiley & Sons Inc, New York, NY, USA, 2000.
[6] K.S. Narendra, A.M. Annaswamy, Stable Adaptive Systems, Prentice Hall, 1989.
[7] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.
[8] B. Widrow, E. Walach, Adaptive Inverse Control, Prentice Hall PTR, Upper Saddle River, NJ 07458, 1996..
[9] T. Hrycej, Neurocontrol: Towards an Industrial Control Methodology, John Wiley & Sons, Inc., 605 Third Ave., New York, NY, 10158-0012, 1997..
[10] G.J. Gray, D.J. Murray-Smith, Y. Li, K.C. Sharman, T. Weinbrenner, Nonlinear model structure identification using genetic programming, Control Eng. Pract. 6 (11) (1998) 1341–1352, URL:http://www.sciencedirect.com/science/article/pii/S0967066198000872.
[11] H. Cao, L. Kang, Y. Chen, J. Yu, Evolutionary modeling of systems of ordinary differential equations with genetic programming, Genetic Program. Evol. Mach. 1 (4) (2000) 309–337, URL:http://link.springer.com/article/10.1023/A:1010013106294.
[12] J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proc. Nat. Acad. Sci. 104 (24) (2007) 9943–9948, URL:http://www.pnas.org/content/104/24/9943.short.
[13] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA USA, 1992.
[14] P.J. Fleming, R.C. Purshouse, Evolutionary algorithms in control systems engineering: a survey, Control Eng. Pract. 10 (2002) 1223–1241.
[15] G. Reynoso-Meza, X. Blasco, J. Sanchis, M. Martinez, Controller tuning using evolutionary multi-objective optimisation: current trends and applications, Control Eng. Pract. 28 (2014) 58–73.
[16] W.G.L. Cava, K. Sahare, K. Danai, Restructuring controllers to accommodate plant nonlinearities, ASME J. Dyn. Syst. Meas. Control 139 (081004) (2017) 1–10.
[17] A. Chipperfield, P. Fleming, Multiobjective gas turbine engine controller design using genetic algorithms, IEEE Trans. Ind. Electron. 43 (5) (1996) 583–587.
[18] S.A. Billings, Nonlinear System Identification, first ed., Wiley, Chichester, West Sussex, UK, 2013.
[19] H. Elci, R.W. Longman, M.Q. Phan, J.-N. Juang, R. Ugoletti, Simple learning control made practical by zero-phase filtering: applications to robotics, IEEE Trans. Circ. Syst. I Fundam. Theory Appl. 49 (6) (2002) 753–767.
[20] D.A. Bristow, M. Tharayil, A.G. Alleyne, A survey of iterative learning control, IEEE Control Syst. Mag. (2006) 96–114.
[21] W.G. La Cava, L. Spector, K. Danai, M.A. Lackner, Evolving differential equations with developmental linear genetic programming and epigenetic hill climbing, in: Proceedings of GECCO'14, Vancouver, BC, Canada, 2014..
[22] W.G. La Cava, T. Helmuth, L. Spector, K. Danai, Genetic programming with epigenetic local search, in: Proceedings of GECCO'15, Madrid, Spain, 2015..
[23] W.G. La Cava, K. Danai, L. Spector, Inference of campact nonlinear dynamic models by epigenetic local search, Eng. Appl. Artif. Intell. 55 (2016) 292–306.
[24] W.G. La Cava, K. Danai, L. Spector, P. Fleming, A. Wright, M. Lackner, Automatic identification of wind turbine models using evolutionary multiobjective optimization, Renew. Energy 87 (Part 2) (2015) 892–902.
[25] S. Devasia, Should model-based inverse inputs be used as feedforward under plant uncertainty, IEEE Trans. Autom. Control 47 (11) (2002) 1865–1871.
[26] S. Devasia, D. Chen, B. Paden, Nonlinear inversion-based output tracking, IEEE Trans. Autom. Control 41 (7) (1996) 930–942.
[27] N. Adhikary, C. Mahanta, Inverse dynamics based robust control method for position commanded servo actuators in robot manipulators, Control Eng. Pract. 66 (2017) 146–155.
[28] H. Cao, L. Kang, Y. Chen, J. Yu, Evolutionary modeling of systems of ordinary differential equations with genetic programming, Genet. Program. Evol. Mach. 1 (2000) 309–337.
[29] H. Iba, Inference of differential equation models by genetic programming, Inf. Sci. 178 (23) (2008) 4453–4468, https://doi.org/10.1016/j.ins.2008.07.029, URL:http://linkinghub.elsevier.com/retrieve/pii/S0020025508002909.
[30] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (5923) (2009) 81–85. URL:http://www.sciencemag.org/content/324/5923/81.short..
[31] C. Ferreira, Gene expression programming: a new adaptive algorithm for solving problems, Complex Syst. 13 (2) (2001) 87–129, arXiv:cs/0102027, URL:http://arxiv.org/abs/cs/0102027.
[32] J.F. Miller, P. Thomson, Cartesian genetic programming, in: Genetic Programming, Springer, 2000, p. 121–132. URL:http://link.springer.com/chapter/10.1007/978-3-540-46239-2_9.
[33] N.X. Hoai, R.I. McKay, D. Essam, R. Chau, Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02, vol. 2, IEEE, pp. 1326–1331. URL:http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1004435..
[34] M. O'Neil, C. Ryan, Grammatical evolution, in: Grammatical Evolution, Springer, pp. 33–47. URL:http://link.springer.com/chapter/10.1007/978-1-4615-0447-4_4..
[35] K. Stanislawska, K. Krawiec, Z.W. Kundzewicz, Modeling global temperature changes with genetic programming, Comput. Math. Appl. 64 (12) (2012) 3717–3728, https://doi.org/10.1016/j.camwa.2012.02.049, URL:http://www.sciencedirect.com/science/article/pii/S0898122112001745.
[36] S. Silva, E. Costa, Dynamic limits for bloat control: variations on size and depth, in: K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Eds.), Genetic and Evolutionary Computation – GECCO-2004, Part II, Vol. 3103 of Lecture Notes in Computer Science, Springer-Verlag, Seattle, WA, USA, 2004, pp. 666–677. doi:10.1007/b98645. URL:http://cisuc.dei.uc.pt/ecos/dlfile.php?fn=714_pub_31030666.pdf&idp=714..
[37] T. Perkis, Stack-based genetic programming, in: Proceedings of the 1994 IEEE World Congress on Computational Intelligence, vol. 1, IEEE Press, Orlando, Florida, USA, 1994, pp. 148–153. doi:10.1109/ICEC.1994.350025. URL:http://citeseer.ist.psu.edu/432690.html..
[38] L. Spector, A. Robinson, Genetic programming and autoconstructive evolution with the push programming language, Genet. Program. Evol. Mach. 3 (1) (2002) 7–40, https://doi.org/10.1023/A:1014538503543, URL:http://hampshire.edu/lspector/pubs/push-gpem-final.pdf.
[39] K.J. Astrom, B. Wittenmark, Adaptive Control, second ed., Dover Publications, Mineola, NY USA, 2008.
[40] H. Hjalmarsson, M. Gevers, S. Gunnarsson, O. Lequin, Iterative feedback tuning: theory and applications, IEEE Control Syst. Mag. 18 (4) (1998) 26–41.
[41] M. Gevers, A decade of progress in iterative process control design – from theory to practice, J. Process Control 12 (2002) 519–531.
[42] H. Hjalmarsson, Iterative feedback tuning – an overview, Int. J. Adapt. Control Signal Process. 16 (2002) 373–395.
[43] O. Lequin, M. Gevers, M. Mossberg, E. Bosmans, L. Triest, Iterative feedback tuning of pid parameters: comparison with classical tuning rules, Control Eng. Pract. 11 (9) (2003) 1023–1033.

[44] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press..
[45] S. Luke, Essentials of Metaheuristics, second ed..
[46] J. Von Neumann, A.W. Burks, et al, Theory of self-reproducing automata, IEEE Trans. Neural Networks 5 (1) (1966) 3–14.
[47] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, U Michigan Press, 1975..
[48] N.L. Cramer, A representation for the adaptive generation of simple sequential programs, in: Proceedings of the First International Conference on Genetic Algorithms, 1985, pp. 183–187.
[49] J. Bongard, H. Lipson, Nonlinear system identification using coevolution of models and tests, IEEE Trans. Evol. Comput. 9 (4) (2005) 361–384.
[50] W. La Cava, K. Danai, L. Spector, P. Fleming, A. Wright, and M. Lackner, Automatic identification of wind turbine models using evolutionary multiobjective optimization, Renew. Energy (2015). [Online]. Available: URL:http://www.sciencedirect.com/science/article/pii/S0960148115303475..
[51] M.D. Schmidt, H. Lipson, Automated modeling of stochastic reactions with large measurement time-gaps, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM, 2011, pp. 307–314. [Online]. Available: URL:http://dl.acm.org/citation.cfm?id=2001619..
[52] I.G. Tsoulos, I.E. Lagaris, Solving differential equations with genetic programming, Genet. Program. Evol. Mach. 7 (1) (2006) 33–54. [Online]. Available: URL:http://link.springer.com/article/10.1007/s10710-006-7009-y..
[53] T. Seaton, G. Brown, J.F. Miller, Analytic solutions to differential equations under graph-based genetic programming, in: A.I. Esparcia-Alcázar, A. Ekárt, S. Silva, S. Dignum, A.I. Uyar (Eds.), Genetic Programming, ser. Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2010, no. 6021, pp. 232–243. [Online]. Available: URL:http://link.springer.com/chapter/10.1007/978-3-642-12148-7_20..
[54] J.D. Lohn, G.S. Hornby, D.S. Linden, An evolved antenna for deployment on nasa's space technology 5 mission, in: Genetic Programming Theory and Practice II, Springer, 2005, pp. 301–315. [Online]. Available: URL:http://link.springer.com/chapter/10.1007/0-387-23254-0_18..
[55] S. Preble, M. Lipson, H. Lipson, Two-dimensional photonic crystals designed by evolutionary algorithms, Appl. Phys. Lett. 86 (6) (2005) 061111. [Online]. Available: URL:http://scitation.aip.org/content/aip/journal/apl/86/6/10.1063/1.1862783..
[56] J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, F. Dunlap, Automated synthesis of analog electrical circuits by means of genetic programming, IEEE Trans. Evol. Comput. 1 (2) (1997) 109–128. [Online]. Available: URL:http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=687879..
[57] A. Elyasaf, A. Hauptman, M. Sipper, GA-FreeCell: evolving solvers for the game of FreeCell, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM, 2011, pp. 1931–1938. [Online]. Available: URL:http://dl.acm.org/citation.cfm?id=2001836..
[58] L. Spector, D.M. Clark, I. Lindsay, B. Barr, J. Klein, Genetic programming for finite algebras, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, ACM, 2008, pp. 1291–1298. [Online]. Available: URL:http://dl.acm.org/citation.cfm?id=1389343..
[59] L. Spector, Automatic Quantum Computer Programming: a genetic programming approach. Springer, 2004, vol. 7. [Online]. Available: URL: http://books.google.com/books?hl=en&lr=&id=HzC58SW6qSQC&oi=fnd&pg=PR7&dq=spector+quantum+computer+programming&ots=PPdJ8Uzq54&sig=CoTjnhGgIl2z8tFh___KfeIXLyI.
[60] https://lacava.github.io/ellen/..
[61] James H. Taylor, Robust nonlinear control based on describing function methods, Proceedings of 1998 ASME IMECE, Dynamic Systems and Control Division, vol. 64, Anaheim, CA, Nov. 1998.