Two-Stage Input Signal Handling in Symbolic Regression

Isabella Rudengren, Torbjörn Wigren and Niklas Wahlström

Abstract— **Symbolic regression methods are relatively new tools in the area of system identification. However, few of these methods include input signal handling. The paper therefore proposes a two-stage algorithm that combines a prediction error method for identification of nonlinear dynamic systems with input signals and a symbolic regression method. The approach treats the input signal as an additional state. This enables the use of any symbolic regression method capable of identifying right-hand sides of autonomous ordinary differential equations. An evaluation using an experimental data set shows that the method performs well by generating analytical expressions similar to the physical equations describing the systems.**

I. Introduction

System identification of dynamic systems is important for understanding and controlling systems in our world, such as biological and industrial processes. These systems are often nonlinear, making their modeling more challenging, which is why system identification of nonlinear systems remains an active field of research.

Several different classes of nonlinear system identification methods have emerged from this field of research. Nonlinear grey-box identification [1] refers to estimation of a set of parameters in a model obtained by other means, e.g., by using physical and chemical laws. Grey-box methods thus often provide algebraic structure with analytical expressions. Parametric nonlinear black-box identification on the other hand, identifies general parameterized model structures like block-oriented models [2]–[5], nonlinear difference equation models [6], or nonlinear state-space models [6], [7], using steepest descent search [3], prediction error techniques [8], or Bayesian sequential Monte-Carlo methods [7], [9], [10]. Nonlinear system identification is also related to supervised machine learning [11], [12].

Another method with increasing interest and highly applicable for nonlinear system identification problems is symbolic regression (SR). This method searches the model space of analytical expressions to find the optimal one describing a system using measured data from the system at hand, which can be relatively small in size [13]. Thus, it simultaneously finds the structure and the parameter values of the analytical expression, often with very little prior information regarding the structure of the equations, but also provides interpretable results.

The work is financially supported by the Swedish Research Council (VR) via the project Physics-informed machine learning (registration number: 2021-04321).

I Rudengren, T. Wigren and N. Wahlström are with the Department of Information Technology, Uppsala University, SE-75105 Uppsala, Sweden. {isabella.rudengren, torbjorn.wigren, niklas.wahlstrom}@it.uu.se.

SR methods are based on several different approaches. One common approach is using genetic programming, such as in PySR [13], Eureqa [14] and Operon [15]. However, different deep learning approaches have also become more common through the increased popularity of the technique [16], [17].

Currently, there exist several SR methods capable of identifying autonomous nonlinear dynamic systems. Extensions of a neural network-based SR method EQL [18] and of the probabilistic grammar-based method ProGED [19] are two examples. PySINDy [20], [21], utilizing linear sparse regression of a set of candidate basis functions, and previously mentioned Eureqa [14] also support identification of these types of systems.

However, considering SR methods supporting nonlinear exogenous systems, there exists only a limited number of methods. Previously mentioned PySINDy [20], [21] and Eureqa [14] are two examples, e.g., Eureqa was used for identification of a robotic manipulator and for controller design [22]. However, PySINDy is limited to sparse linear combinations of basis functions, and Eureqa is no longer an open-source model but commercial software. The vast majority of SR methods and software are limited to autonomous nonlinear dynamic systems. This restriction is particularly severe in engineering, where input signals and feedback strategies are the main topics.

The main contribution of the paper addresses the need for symbolic regression methods for nonlinear dynamic systems with input signals by a novel two-stage algorithm. The method enables the handling of exogenous systems by existing algorithms and software for autonomous systems. It consists of a black-box identification algorithm for nonlinear dynamic systems with input signals and a symbolic regression algorithm. The black-box algorithm enables modeling of dynamic systems, since it identifies an n^{th} order state-space ordinary differential equation (ODE) describing the system. However, since the black-box system identification method only provides a polynomially parameterized right-hand side (RHS) of a differential equation, an SR algorithm is used to analyze the RHS to find more efficient analytical expressions. Experiment with live data shows that the SR algorithm is able to find established analytical expressions that describe the system from the polynomial ODE.

The rest of the paper is organized as follows. In section II, the algorithm of the new method and its implementation are presented. The results on live data are presented in section III, and these are further discussed in section IV. In section V, the conclusions of the paper are presented, and future work is discussed.

II. Method

The proposed method is a two-stage algorithm illustrated in Fig. 1. It consists of a black-box nonlinear system identification method followed by an SR method, with a data generation step in between. The nonlinear system identification method identifies an n^{th} order state-space ODE describing the system, where the time derivative of the n^{th} state vector component is expressed with a parametrized polynomial expression. This polynomial expression is determined by the algorithm using measured data of the system.

Using the identified system obtained from Stage 1 of the algorithm, simulated data is generated. Based on the simulated data set, an SR method is then used to identify the static RHS of the n^{th} ODE component and thus find a more efficient expression for the RHS. This results in a final model for the system. The key aspect here is that the proposed method enables handling the input signal as an additional state when the SR method is run.

A. Algorithm

In the following section, the two algorithms in the two-stage method and the intermediate data generation step are briefly defined. For more detailed descriptions of the individual algorithms, the reader is referred to the references. Lastly, a summary of the complete algorithm is presented.

1) Black-box Identification of a Nonlinear ODE: Stage 1 of the two-stage algorithm is a method for identification of nonlinear dynamic systems with input signals using a prediction error identification algorithm [23]. For the description of the method, define

$$\boldsymbol{u}(t) = \begin{bmatrix} u_1(t) & \dots & u_1^{(n_1)}(t) & \dots & u_k(t) & \dots & u_k^{(n_k)}(t) \end{bmatrix}^\top \tag{1}$$

as the measured input vector where $u_k^{(n_k)}(t)$ is the n_k^{th} time-derivative of $u_k(t)$. Also, let the state vector be defined as

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) & \dots & x_n(t) \end{bmatrix}^\top, \tag{2}$$

and the output vector as

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1(t) & \dots & y_p(t) \end{bmatrix}^\top. \tag{3}$$

Since the system is assumed to be described by a nonlinear ODE, it can be modeled according to the following n^{th} order state-space model,

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \vdots \\ x_n(t) \\ f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}) \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} y_1(t) \\ \vdots \\ y_p(t) \end{bmatrix} = \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{p1} & \dots & c_{pn} \end{bmatrix} \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}. \tag{5}$$

The last row in (4) is defined by a polynomial parametrization, $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta})$. It is a sum of all different monomials of the components in the state vector $\boldsymbol{x}(t)$ and the components in the input vector $\boldsymbol{u}(t)$, given that the allowed maximum polynomial degree for each component is specified. This function is the key quantity determined by the algorithm. Note that the selected model structure condenses the effect of the input signal into a single RHS ODE state component. According to [23], [24], the parametrization can be written in the form (here omitting the time dependency due to readability)

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}) = & \sum_{i_{x_1}=0}^{I_{x_1}} \dots \sum_{i_{x_n}=0}^{I_{x_n}} \sum_{i_{u_1}=0}^{I_{u_1}} \dots \sum_{i_{u_1^{(n_1)}}=0}^{I_{u_1^{(j_1)}}} \\ & \dots \sum_{i_{u_k}=0}^{I_{u_k}} \dots \sum_{i_{u_k^{(n_k)}}=0}^{I_{u_k^{(j_k)}}} \theta_{i_{x_1} \dots i_{x_n} i_{u_1} \dots i_{u_1^{(n_1)}} \dots i_{u_k} \dots i_{u_k^{(n_k)}}} \\ & \cdot x_1^{i_{x_1}} \dots x_n^{i_{x_n}} \cdot u_1^{i_{u_1}} \dots (u_1^{(n_1)})^{i_{u_1^{(n_1)}}} \dots u_k^{i_{u_k}} \\ & \dots (u_k^{(n_k)})^{i_{u_k^{(n_k)}}} = \boldsymbol{\phi}^\top(\boldsymbol{x}, \boldsymbol{u}) \boldsymbol{\theta}, \end{aligned} \tag{6}$$

where I_z is the maximum polynomial degree of variable z. As seen in (6), the parametrization can also be written as the scalar product between the parameter vector $\boldsymbol{\theta}$ and the vector $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u})$ defined as

$$\begin{aligned} \boldsymbol{\theta} = \Big[& \theta_{0\dots0} \ \dots \ \theta_{0\dots I_{u_k^{(n_k)}}} \ \theta_{0\dots010} \\ & \dots \ \theta_{0\dots01 I_{u_k^{(n_k)}}} \ \dots \ \theta_{0\dots0 I_{u_k^{(n_k-1)}} 0} \\ & \dots \ \theta_{0\dots0 I_{u_k^{(n_k-1)}} I_{u_k^{(n_k)}}} \ \dots \ \theta_{I_{x_1}\dots I_{u_k^{(n_k)}}} \Big]^\top, \end{aligned} \tag{7}$$

$$\begin{aligned} \boldsymbol{\phi} = \Big[& 1 \ \dots \ (u_k^{(n_k)})^{I_{u_k^{(n_k)}}} \ u_k^{(n_k-1)} \ \dots \ u_k^{(n_k-1)} (u_k^{(n_k)})^{I_{u_k^{(n_k)}}} \\ & \dots \ (u_k^{(n_k-1)})^{I_{u_k^{(n_k-1)}}} \ \dots \ (u_k^{(n_k-1)})^{I_{u_k^{(n_k-1)}}} (u_k^{(n_k)})^{I_{u_k^{(n_k)}}} \\ & \dots \ \left(x_1^{I_{x_1}} \ \dots \ x_n^{I_{x_n}} u_1^{I_{u_1}} \ \dots \ (u_k^{(n_k)})^{I_{u_k^{(n_k)}}} \right) \Big]^\top. \end{aligned} \tag{8}$$

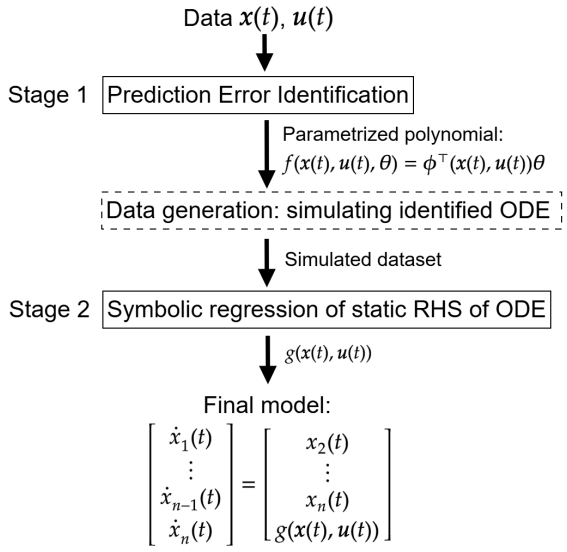

Fig. 1. A flowchart showing the two-stage algorithm with input and output from each of the two algorithms and the data generation stage in between.

This model describes the dynamics of the system (4).

Since the algorithm assumes discrete time measurements of the system of interest, the ODE model is discretized with an Euler forward discretization scheme to obtain

$$\begin{bmatrix} x_1(t+T_s,\boldsymbol{\theta}) \\ \vdots \\ x_{n-1}(t+T_s,\boldsymbol{\theta}) \\ x_n(t+T_s,\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} x_1(t,\boldsymbol{\theta}) \\ \vdots \\ x_{n-1}(t,\boldsymbol{\theta}) \\ x_n(t,\boldsymbol{\theta}) \end{bmatrix} + T_s \begin{bmatrix} x_2(t,\boldsymbol{\theta}) \\ \vdots \\ x_n(t,\boldsymbol{\theta}) \\ \boldsymbol{\phi}^T(\boldsymbol{x}(t,\boldsymbol{\theta}),\boldsymbol{u}(t))\boldsymbol{\theta} \end{bmatrix} \tag{9}$$

$$\boldsymbol{y}(t,\boldsymbol{\theta}) = \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{p1} & \dots & c_{pn} \end{bmatrix} \begin{bmatrix} x_1(t,\boldsymbol{\theta}) \\ \vdots \\ x_n(t,\boldsymbol{\theta}) \end{bmatrix} = \boldsymbol{C}\boldsymbol{x}(t,\boldsymbol{\theta}), \tag{10}$$

where T_s denotes the sampling period. The measurement model (10) is hence assumed to be linear. The gradient of the output predictor with respect to the parameter vector is used in the updates in the algorithm. It is calculated from the product of the $\boldsymbol{C}$-matrix and the derivative of the state vector $\boldsymbol{x}(t,\boldsymbol{\theta})$ with respect to the parameter vector $\boldsymbol{\theta}$ of the discretized ODE model (see [23], [24] for details).

The criterion minimized in the prediction error identification algorithm, using the stochastic Gauss-Newton method, is

$$V(\boldsymbol{\theta}) = \frac{1}{2}\,\mathbb{E}\left[\boldsymbol{\varepsilon}^\top(t,\boldsymbol{\theta})\boldsymbol{\Lambda}(t,\boldsymbol{\theta})^{-1}\boldsymbol{\varepsilon}(t,\boldsymbol{\theta}) + \log\det(\boldsymbol{\Lambda}(t,\boldsymbol{\theta}))\right], \tag{11}$$

where the prediction error is given by

$$\boldsymbol{\varepsilon}(t,\boldsymbol{\theta}) = \boldsymbol{y}_m(t) - \boldsymbol{y}(t,\boldsymbol{\theta}), \tag{12}$$

where $\boldsymbol{y}_m(t)$ is the measured output. The expectation operator is denoted $\mathbb{E}$, and $\boldsymbol{\Lambda}(t,\boldsymbol{\theta})$ is the covariance matrix of the prediction error. To monitor the stability of the identification method, the algorithm of [23] contains a projection algorithm based on the linearized system matrix.

Remark 1: The implicit function theorem can be used to motivate that (4) and (9) locally describe the dynamics of ODEs with a generalized RHS [24]. In addition, the canonical form of (9) avoids problems caused by overparameterization.

Remark 2: The two-stage method can now be motivated further, noting that physical engineering models are often described by coupled ODE:s where the derivative of each state variable may correspond to a non-trivial RHS component. The method proposed for Stage 1 below instead computes a model from input-output data that is in canonical form, with only the last state variable having a non-trivial polynomial right-hand side, parameterized in states and inputs. This representation allows the input to be treated as a known variable, facilitating the generation of the state derivatives of the left-hand side with one-dimensional polynomial evaluation. The SR method in Stage 2 requires such an evaluation of the time derivative of the state. The black-box model thus provides this through the parametrized polynomial expression.

2) Data Generation based on the Identified ODE: Based on the identified system obtained from the nonlinear identification method in Stage 1, the states are simulated using an Euler forward method using the same initial conditions, input signal and time sampling instances as the data. The identified polynomial expression in (6) is then evaluated for all simulated state-input data points, i.e., $f_j = f(\boldsymbol{x}_j(t), \boldsymbol{u}_j(t))$ is generated for all time steps $j = 1, \dots, N$ where N is the total number of time sampling instances. This is necessary for the second stage of the algorithm, where an SR method is used to find more efficient analytical expressions of the static RHS of the ODE.

3) Symbolic Regression on the Static RHS of the ODE: The second part of the algorithm consists of the previously mentioned SR algorithm, PySR [13]. PySR is an open-source symbolic regression machine learning model made for scientific discovery. The method is a multi-population evolutionary algorithm utilizing genetic algorithms to find a Pareto front of optimal analytic expressions describing a given data set. The analytic expressions are constructed based on a set of allowed operators as well as constraints on how these operators are combined, both given by the user. The populations then evolve by creating new candidate expressions based on the fittest expression in the individual populations, using, e.g., mutations, simplifications, optimizations and crossovers. Migration of different candidate expressions between populations also takes place.

For SR methods in general, the candidate expressions should minimize prediction error and model complexity simultaneously, since the expressions should be both accurate and simple. For PySR, this is achieved by defining the loss for a candidate expression as a combination of the predictive loss and another loss term taking into account the frequency and recency of the expression compared to other expressions within its complexity class in its population.

The algorithm presents a list consisting of the best expression for each complexity level along with their loss and score, ordered from lowest complexity to highest complexity. Since PySR operates by finding the best candidate expression balancing complexity and loss, the score metric used is defined as

$$\text{score} = -\frac{\log\left(l(E)_i\right) - \log\left(l(E)_{i-1}\right)}{C(E)_i - C(E)_{i-1}}, \tag{13}$$

where $l(E)_i$ and $C(E)_i$ are the loss and complexity, respectively, for expression E in row i in the ordered list of candidate expressions with respect to complexity [25]. Thus, the score is a metric based on comparisons between changes in loss relative to the change in complexity between adjacent candidate expressions in the list of the best candidate expression for each complexity level. A candidate expression receives a high score if the loss decreases significantly while the complexity slightly increases.

Stage 2 of the algorithm finds the best analytical expression for the static RHS of the ODE based on data generated in the previous step. This expression, obtained from the SR method, is here denoted $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$. By substituting $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta})$ with $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$ in the canonical state-space model in (4) obtained from Stage 1, the following

final model is obtained,

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \vdots \\ x_n(t) \\ g(\boldsymbol{x}(t), \boldsymbol{u}(t)) \end{bmatrix}. \tag{14}$$

4) Algorithm Summary: A short summary of the complete two-stage algorithm is given here:

The prerequisite consists of a data set with measurements of the output of a nonlinear dynamic system together with an input signal.

- **Stage 1**: a polynomial parametrization, $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta})$, of the ODE in (4) is identified based on the data set using the prediction error method.
- A data set for the SR part of the algorithm is created based on the state-space model identified in Stage 1. It consists of values of the state vector, the input vector and the evaluated expression of $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta})$ in the points of the data set.
- **Stage 2**: The static RHS of the ODE, $f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta})$, is identified based on the created data set using the SR software, PySR. The expression $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$ is obtained.

The final model is obtained, defined by (14) where $g(\boldsymbol{x}(t), \boldsymbol{u}(t))$ is given by the identified expression in Stage 2.

B. Implementation

Stage 1 of the two-stage algorithm is implemented using [26]. Given a data set, the algorithm is first set up. The maximum polynomial degree of each variable, I_z, and which polynomial terms should be disregarded are set, as well as the initialization of the parameter vector $\boldsymbol{\theta}$. The algorithm then computes the values of $\boldsymbol{\theta}$, thereby minimizing (11), while also checking the stability during the calculations. This results in an identified expression for the parametrized polynomial function (6) in the n^{th} row of the system of ODEs, and thereby an identified state-space model for the system, cf. (4). The simulated data based on the identified system is manually transferred to Python for Stage 2 of the algorithm.

Stage 2 of the method uses PySR (v1.5.9) [13]. In PySR, a symbolic regression model is set up by defining a set of hyperparameters regarding the search space and the objective. The data given to the model consists of state vector values, input signal values and a vector consisting of evaluations of the parametric expression in (6) for these values of $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$. The generated data is subsampled to approximately 1000 data points to minimize computational complexity without losing essential information about the system [25]. The algorithm generates candidate expressions according to the algorithm described in section II-A.3 for a given number of iterations. The three highest-scoring and the best candidate expressions are displayed here.

III. Results

To illustrate the performance of the proposed method, the ODE describing a single-tank system with a free outlet is identified using live data. The ODE [23] describing the system is,

$$\begin{cases} \dot{x}_1(t) = -\frac{a_1\sqrt{2g}}{A_1}\sqrt{x_1(t)} + \frac{k}{A_1}u(t) + w(t) \\ y(t) = x_1(t) + e(t). \end{cases} \tag{15}$$

Here $x_1(t)$ is the water level in the upper tank, $u(t)$ is the input signal to the system, i.e., the voltage applied to the pump, and $w(t)$ and $e(t)$ denote a system noise and a measurement disturbance, respectively. The gravitational acceleration constant is denoted g, and $\frac{a_1}{A_1}$ and $\frac{k}{A_1}$ denote model parameters dependent on the area of the tank, the area of the tank outlet and a conversion factor, k, from the applied voltage to the input flow. The values of these parameters are unknown for this data set.

The data set used was obtained from [27]. It consists of measurements from two cascaded tanks driven by a pump, where both tanks have free outlets. Only the data for the upper tank was used here. The data set was generated using a pseudorandom binary sequence (PRBS) multiplied by a uniform amplitude distribution as an input signal with a clock period of $30\,\text{s}$ and a sampling period of $5.0\,\text{s}$. In total, the data set consists of 2500 data points.

For Stage 1, the maximum polynomial degrees were defined as $[I_{x_1}, I_u] = [3, 1]$. In addition, all mixed terms (e.g., $x_1(t)u(t)$) were excluded due to difficulties in obtaining a stable solution in Stage 1 of the algorithm when including the mixed terms. The parameter vector was initialized as

$$\boldsymbol{\theta}_o = \begin{bmatrix} 0.2 & 0.6 & -0.3 & 0.0 & 0.0 \end{bmatrix}^\top, \tag{16}$$

for

$$\boldsymbol{\phi} = \begin{bmatrix} 1 & u(t) & x_1(t) & x_1^2(t) & x_1^3(t) \end{bmatrix}^\top, \tag{17}$$

which resulted in the identified expression for the polynomial parametrization seen in Table I. A comparison of the simulated output from this identified system, $y_{\text{sim1}}(t)$, and the true output from the data set, $y_{\text{data}}(t)$, can be seen in Fig. 2. The simulated output coincides well with the true output, though one can note deviations, especially around time $2000\,\text{s}$. This might imply that the full behavior of the system is not captured by the parametrized model; however, the model is considered good enough to use for Stage 2 of the algorithm.

The PySR model for this experiment was set up with $\{+, -, *, /, \text{sqrt}\}$ as the allowed operators. The model used the default L2-distance loss function, and due to the limited size of the true expression, the maximum complexity of the analytic expressions was set to `maxsize=20`. To add a small punishment to more complex expressions, `parsimony=0.0001`, and also `weight_optimize=0.001` were used to increase the frequency of optimization of candidate expressions. The experiment was run for `niterations=800` iterations. The three highest-scoring final candidate expressions and the best expression according to PySR are presented in Table I (omitting time dependency notation).

The highest-scoring expression and the best expression are both polynomial expressions similar to the polynomial

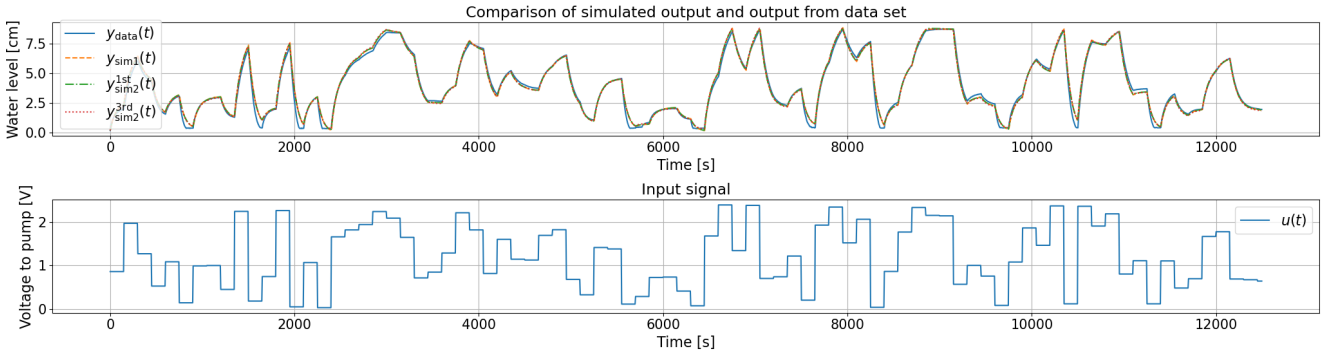

Fig. 2. Comparison of output from data set, $y_{\text{data}}(t)$, and simulated output from identified system after Stage 1, $y_{\text{sim1}}(t)$, and Stage 2, $y_{\text{sim2}}(t)$, for the tank system. The trajectories for the highest-scoring and third-highest-scoring expressions from Table I are included. The input signal $u(t)$, which is a PRBS multiplied by a uniform amplitude distribution, is also included in the figure.

expression obtained from Stage 1, and they are of no interest. The third highest-scoring expression is relatively close to the true expression. It contains a u-term, a bias term and a $\sqrt{x_1 + 0.638}$-term, which is relatively close to the true $\sqrt{x_1}$-term, since the range of x_1-values ($[0.0, 8.8]$) is mostly larger than the constant, 0.638. The second highest-scoring expression contains an $x_1\sqrt{x_1}$-term and an x_1-term, which is not consistent with the true expression.

The trajectories obtained from simulating the identified systems of the highest-scoring expression and third-highest-scoring expression in Table I are visualized in Fig. 2. Both trajectories coincide well with the trajectory obtained after Stage 1, and they only deviate slightly from each other. However, the deviation from the data is larger.

IV. Discussion

The results in section III show that the exact true form of the physical expression for the single-tank system is not included among the highest-scoring candidates. However, one candidate expression has a structure similar to the true expression, and considering the range of the state vector, one could argue that this is a candidate expression similar to the true one. Also, considering this is a live data set, which therefore presumably contains more uncertainty, the resulting trajectories in Fig. 2 coincide well with the data. Existing inaccuracies and deviations might originate from modeling errors in model (15) and inaccuracies in the identified expression obtained after Stage 1 due to difficulties with the stability of the method. It is also worth noting that the algorithm identified the polynomial expression obtained in Stage 1.

The proposed two-stage algorithm can hence identify nonlinear dynamic systems with input; thus, this is a new algorithm for handling systems of ODEs with input signals using SR methods. In addition, the proposed method is flexible since the individual stages of the algorithm can be exchanged with similar methods. However, the nonlinear identification method should generate a system of coupled ODEs and a function for the time derivative of the state vector to enable evaluation of this derivative. The SR method should be able to identify static functions, which several already existing SR methods can. Furthermore, this proposed algorithm is a good alternative to other SR algorithms for exogenous systems, such as Eureqa [14] and PySINDy [20], [21], when wanting non-commercial software with more flexibility than linear combinations of basis functions.

One can question whether a correct identification would have required the true expression to be chosen as the best expression according to PySR. However, the definition of the best equation implies that more complex expressions tend to get chosen depending on the choice of maximum complexity, since the expressions with higher complexity tend to get a lower loss compared to expressions with lower complexity. This does not necessarily mean that a more complex expression is a better model of the system, since it might be an overfit to the data. Therefore, the choice of maximum complexity remains an important factor.

It is also important to note that for a 1^{st} order state-space ODE, as for the simple-tank system, where identifying a system of coupled ODEs is not needed, an alternative approach would be to only use the SR method. However, this requires a numerical approximation of the time derivative of the state since this property often cannot be exactly measured. For noisy experimental data, numerical methods can result in poor approximations of the time derivative. This proposed method can thus be advantageous in this case since it may provide a better approximation of the time derivative for noisy data. It could be interesting to analyze this further with future experiments.

V. Conclusions and Future Work

The paper proposes a new two-stage algorithm for identification of nonlinear dynamic systems with input signals using SR methods. The method combines a nonlinear system identification prediction error method and an SR method. It was shown to successfully identify a system from experimental data, and it offers, to some extent, flexibility in exactly what methods are used in the two stages. The method is therefore applicable to low-order models and grey-box modeling since it requires some prior knowledge of the model structure.

The paper also aims at highlighting the need for symbolic regression methods that can handle nonlinear dynamic sys-

TABLE I

LIST OF SELECTED FINAL CANDIDATE EXPRESSIONS FOR THE TANK SYSTEM WITH CORRESPONDING LOSS AND SCORE. THE THREE HIGHEST-SCORING EXPRESSIONS AND THE BEST EXPRESSION (BOLD EXPRESSION) ACCORDING TO PYSR ARE INCLUDED. THE NOTATION FOR TIME DEPENDENCY IS OMITTED HERE DUE TO READABILITY.

	Equation $f(\boldsymbol{x}, \boldsymbol{u})$	Loss	Score
True	$-\frac{a_1\sqrt{2g}}{A_1}\sqrt{x_1} + \frac{k}{A_1}u$	–	–
Stage 1	$-0.0009 + 0.0512u - 0.0191x_1 + 0.0009x_1^2 - 0.00002x_1^3$	–	–
Stage 2	$u0.0512 + x_1\left(x_1\left(x_1\left(-2.18 \cdot 10^{-5}\right) - -0.000943\right) - 0.0191\right) - 0.000936$	$1.96 \cdot 10^{-13}$	9.05
	$0.0512u + x_1\left(0.00338\sqrt{x_1} - 0.0226\right)$	$1.88 \cdot 10^{-8}$	3.04
	$0.0510u - 0.0510\sqrt{x_1 + 0.638} + 0.0465$	$7.87 \cdot 10^{-7}$	2.67
	$\boldsymbol{u0.0512 + x_1\left(x_1\left((x_1 + 0.342)\left(-2.18 \cdot 10^{-5}\right) - -0.000950\right) - 0.0191\right) - 0.000936}$	$4.37 \cdot 10^{-14}$	0.752

tems with input signals. To the best of our knowledge, only a few methods exist; however, there is a need for methods that directly handle these kinds of systems and that output scientific, interpretable expressions.

Future work could involve further evaluation of the proposed two-stage algorithm by investigating the range of problems the method can handle and defining its limitations. It could be done by testing the method on other simulated and experimental data sets of systems with varying prior knowledge. Also, experimenting with the choice of method in Stage 1 of the algorithm could be interesting. The development of an SR method directly handling nonlinear dynamic systems with input signals would also be desirable, but it requires more work. A further study on the criteria that balance complexity and accuracy is also a possible future direction.

REFERENCES

[1] T. Bohlin, "A case study of grey box identification," *Automatica*, vol. 30, no. 2, p. 307–318, 1994.

[2] S. A. Billings and S. Y. Fakhouri, "Identification of systems containing linear dynamic and static nonlinear elements," *Automatica*, vol. 18, no. 1, p. 15–26, 1982.

[3] K. Åström, "The adaptive nonlinear modeler", in *Research Reports TFRT-3178*. Lund, Sweden: Department of Automatic Control, Lund Institute of Technology (LTH), 1985.

[4] P. Stoica and T. Söderström, "Instrumental-variable methods for identification of Hammerstein systems," *Int. J. Control*, vol. 35, no. 3, pp. 459–476, 1982.

[5] S. A. Billings and S. Y. Fakhouri, "Identification of nonlinear systems using the Wiener model," *Electron. Lett.*, vol. 13, pp. 502–504, 1977.

[6] S. Chen and S. A. Billings, "Representation of non-linear systems: the NARMAX model," *Int. J. Control*, vol. 49, no. 3, pp. 1012–1032, 1989.

[7] T. Schön and F. Gustafsson, "Particle filters for system identification of state-space models linear in either parameters or states," in *Proc. SYSID*, Rotterdam, The Netherlands, Aug. 2003, pp. 1251–1256.

[8] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, N.J: PTR Prentice Hall, 1999.

[9] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge, UK: Cambridge Univ. Press, 2013.

[10] A. Wigren, J. Wågberg, F. Lindsten, A. G. Wills, and T. B. Schön, "Nonlinear system identification: Learning while respecting physical models using a sequential Monte Carlo method," *IEEE Control Syst. Mag.*, vol. 42, no. 1, pp. 75–102, 2022.

[11] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning - A First Course for Engineers and Scientists*. Cambridge, UK: Cambridge Univ. Press, 2022.

[12] S. J. D. Prince, *Understanding Deep Learning*. Boston, MA: MIT Press, 2023.

[13] M. Cranmer, "Interpretable machine learning for science with PySR and SymbolicRegression.jl," 2023, *arXiv:2305.01582*.

[14] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[15] B. Burlacu, G. Kronberger, and M. Kommenda, "Operon C++: An efficient genetic programming framework for symbolic regression," in *Proc. 2020 Genetic Evol. Comput. Conf. Companion*, ACM, New York, NY, USA, 2020, p. 1562–1570.

[16] G. Martius and C. H. Lampert, "Extrapolation and learning equations," 2016, *arXiv:1610.02995*.

[17] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, and M. Soljačić, "Integration of neural network-based symbolic regression in deep learning for scientific discovery," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 9, pp. 4166–4177, 2021.

[18] S. Sahoo, C. Lampert, and G. Martius, "Learning equations for extrapolation and control," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 10–15, 2018, pp. 4442–4450.

[19] N. Omejc, B. Gec, J. Brence, L. Todorovski, and S. Džeroski, "Probabilistic grammars for modeling dynamical systems from coarse, noisy, and partial data," *Mach. Learn.*, vol. 113, no. 10, pp. 7689–7721, 2024.

[20] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "PySINDy: A python package for the sparse identification of nonlinear dynamical systems from data," *J. Open Source Softw.*, vol. 5, no. 49, p. 2104, 2020.

[21] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, "PySINDy: A comprehensive python package for robust sparse system identification," *J. Open Source Softw.*, vol. 7, no. 69, p. 3994, 2022.

[22] Z. Zhang and Z. Chen, "Modeling and control of robotic manipulators based on symbolic regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2440–2450, 2023.

[23] T. Wigren, "Recursive prediction error identification and scaling of non-linear state space models using a restricted black box parameterization," *Automatica*, vol. 42, no. 1, pp. 159–168, 2006.

[24] ——, "Recursive prediction error identification of nonlinear state space models," Uppsala University, Uppsala, Sweden, Tech. Rep. Dept. Inf. Technol. 004-2004, 2004. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:75835/FULLTEXT01.pdf

[25] M. Cranmer. "API Reference." PySR Discover Mathematical Laws from Data. Accessed: Oct. 27, 2025. [Online]. Available: https://ai.damtp.cam.ac.uk/pysr/v1.5.9/api

[26] T. Wigren, "Matlab software for recursive identification and scaling using a structured nonlinear black-box model - revision 7," Uppsala University, Uppsala, Sweden, Tech. Rep. Dept. Inf. Technol. 2021-008, Dec. 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1706386/SOFTWARE01.zip

[27] ——, "Input-output data sets for development and benchmarking in nonlinear identification," Uppsala University, Uppsala, Sweden, Tech. Rep. Dept. Inf. Technol. 2010-020, Aug. 2010. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:379738/SOFTWARE01.zip