Data-driven Construction of Symbolic Process Models for Reinforcement Learning

Erik Derner, Jiří Kubalík, and Robert Babuška

Presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), May 2018, Brisbane, QLD, Australia.

DOI: 10.1109/ICRA.2018.8461182
`https://ieeexplore.ieee.org/document/8461182`

IEEE Copyright Notice

BibTeX Record

```
@INPROCEEDINGS{Derner2018-ICRA,
author="Derner, Erik and Kubal{\'i}k, Ji{\v{r}}{\'i} and Babu{\v{s}}ka, Robert",
booktitle={2018 IEEE International Conference on Robotics and Automation (ICRA)},
title={Data-driven Construction of Symbolic Process Models for Reinforcement Learning},
year={2018},
volume={},
number={},
pages={1-8},
doi={10.1109/ICRA.2018.8461182},
issn={2577-087X},
month={May},}
```

Data-driven Construction of Symbolic Process Models for Reinforcement Learning

Erik Derner[1], Jiří Kubalík[2], and Robert Babuška[3]

Abstract— **Reinforcement learning (RL) is a suitable approach for controlling systems with unknown or time-varying dynamics. RL in principle does not require a model of the system, but before it learns an acceptable policy, it needs many unsuccessful trials, which real robots usually cannot withstand. It is well known that RL can be sped up and made safer by using models learned online. In this paper, we propose to use symbolic regression to construct compact, parsimonious models described by analytic equations, which are suitable for real-time robot control. Single node genetic programming (SNGP) is employed as a tool to automatically search for equations fitting the available data. We demonstrate the approach on two benchmark examples: a simulated mobile robot and the pendulum swing-up problem; the latter both in simulations and real-time experiments. The results show that through this approach we can find accurate models even for small batches of training data. Based on the symbolic model found, RL can control the system well.**

Index Terms— **Model learning for control, AI-based methods, symbolic regression, reinforcement learning, optimal control.**

I. Introduction

An RL agent optimizes its behavior by interacting with its environment in order to find an optimal policy which maximizes the long-term cumulative reward. Although RL can work in a completely model-free fashion, the absence of a model leads to slow convergence and therefore extensive learning times [1], [2], [3]. To speed up learning, various model-based methods have been proposed [4], [5], [6], [7], [8], [9]. Many model-learning approaches have been described in the literature: time-varying linear models [10], [11], Gaussian processes [4], [12] and other probabilistic models [13], deep neural networks [14], [15] or local linear regression [16], [17].

All the above approaches suffer from drawbacks induced by the use of the specific approximation technique, such as a large number of parameters (deep neural networks), local nature of the approximator (local linear regression), computational complexity (Gaussian process), etc.

In this article we propose another way to capture the system dynamics using a symbolic model constructed by means of symbolic regression (SR). Symbolic regression is based on genetic programming and it has been used in nonlinear data-driven modeling, often with quite impressive results [18], [19], [20], [21], [22]. Therefore, we hypothesize that also within RL, it will generate accurate, parsimonious models based on even very small training data sets.

To date, only a few works have been reported on the use of SR within the RL domain. For example, in [23] SR was used to construct a symbolic function, which serves as a proxy to the value function and from which a continuous policy can be derived. Another work dealing with value function discovery by means of GP is [24], where algebraic descriptions of the value function are obtained based on data sampled from the optimal value function. A multi-objective evolutionary algorithm was proposed in [25] to learn a value function capturing the user's preferences provided through regular interactions with the user. In [26] SR was used to construct a smooth analytic approximation of the policy based on data sampled from the interpolated policy. To our best knowledge, there have been no reports in the literature on the use of symbolic regression for building the process model in reinforcement learning. We argue that the effective use of symbolic regression for model learning is a valuable element missing from the current RL schemes and we demonstrate its usefulness.

The paper is organized as follows. Section II presents the necessary reinforcement learning prerequisites. The proposed method for finding symbolic process models is described in Section III and Section IV presents the results of experiments we performed in order to evaluate the method. Finally, Section V concludes the paper.

II. Preliminaries

The system for which an optimal control strategy is to be learnt is described in discrete time by the following nonlinear state-space model

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

This work was supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000470), by the Grant Agency of the Czech Republic (GAČR) with the grant no. 15-22731S titled "Symbolic Regression for Reinforcement Learning in Continuous Spaces", and by the Grant Agency of the Czech Technical University in Prague, grant no. SGS16/232/OHK3/3T/13. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

[1]Erik Derner is with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16636 Prague, Czech Republic and with the Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 16627 Prague, Czech Republic erik.derner@cvut.cz

[2]Jiří Kubalík is with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16636 Prague, Czech Republic jiri.kubalik@cvut.cz

[3]Robert Babuška is with Cognitive Robotics, Faculty of 3mE, Delft University of Technology, 2628 CD Delft, The Netherlands and with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, 16636 Prague, Czech Republic r.babuska@tudelft.nl

with $x_k, x_{k+1} \in \mathcal{X} \subset \mathbb{R}^n$ and $u_k \in \mathcal{U} \subset \mathbb{R}^m$. Note that the actual process can be stochastic (e.g., the sensor readings are corrupted by noise), but for the sake of RL control, we aim at constructing a deterministic model. The control goal is specified through a *reward function* which assigns a scalar reward $r_{k+1} \in \mathbb{R}$ to each state transition from x_k to x_{k+1}:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1}) \tag{2}$$

The reward function ρ is defined by the user and typically calculates the reward based on the distance of the current state to a given goal state that should be attained.

Based on model (1), we compute the optimal control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that in each state it selects a control action so that the cumulative discounted reward over time, called the return, is maximized:

$$R^\pi = E\left\{ \sum_{k=0}^{\infty} \gamma^k \rho\left(x_k, \pi(x_k), x_{k+1}\right) \right\} \tag{3}$$

Here $\gamma \in (0,1)$ is a discount factor and the initial state x_0 is drawn uniformly from the state space domain $\mathcal{X}$ or its subset. The return is captured by the value function $V^\pi : \mathcal{X} \to \mathbb{R}$ defined as:

$$V^\pi(x) = E\left\{ \sum_{k=0}^{\infty} \gamma^k \rho\left(x_k, \pi(x_k), x_{k+1}\right) \middle| x_0 = x \right\} \tag{4}$$

An approximation of the optimal V-function, denoted by $\hat{V}^*(x)$, can be computed by solving the Bellman optimality equation

$$\hat{V}^*(x) = \max_{u \in \mathcal{U}} \left[\rho\left(x, \pi(x), f(x,u)\right) + \gamma \hat{V}^*\left(f(x,u)\right) \right] \tag{5}$$

To simplify the notation, in the sequel, we drop the hat and the star superscript: $V(x)$ will therefore denote the approximated optimal V-function. Based on $V(x)$, the corresponding optimal control action is found as the one that maximizes the right-hand side of (5):

$$u = \underset{u' \in U}{\operatorname{argmax}} \left[\rho(x, u', f(x,u')) + \gamma V(f(x,u')) \right] \tag{6}$$

In this paper, the above equation is used online as the control policy and U is a set of discretized actions, so that the near-optimal action can be found by enumeration. To find the approximation of the optimal value function $V(x)$ we employ the fuzzy V-iteration algorithm [27].

The prerequisite for this model-based RL approach is the availability of the process model (1). In the following section, we propose an approach for constructing such a model by using symbolic regression.

III. Method

A. Model-based reinforcement learning scheme

The RL scheme we use is depicted in Fig. 1 and it consists of the following elements:

1) Low-level control loop with the RL controller. This controller can be either the policy (6) based on the V-function learnt offline using value iteration, or it can be any model-based RL algorithm whose parameters are adjusted online. Examples of such algorithms are the standard DYNA [28], [29] or more recent ones, such as MLAC [16].
2) Input-state samples of the form (u_k, x_k) are collected in the data buffer. In this paper we assume that the states are measured, but the method equally well applies to input-output models of the form $y_{k+1} = f(y_k, y_{k-1}, \ldots, u_k, u_{k-1}, \ldots)$, where the state is represented by the vector of past inputs and outputs.
3) Symbolic regression is periodically applied to the available data set, yielding a symbolic model of the process, described in the form of analytic equations. This model can be used in several ways: a) offline to (re)compute the value function and policy, b) online by the RL algorithm to assist the parameter update [16], c) online to generate transition samples in the parallel DYNA setting [29]. Note that also the data stored in the buffer can be used by the on-line RL algorithm using experience replay [14].

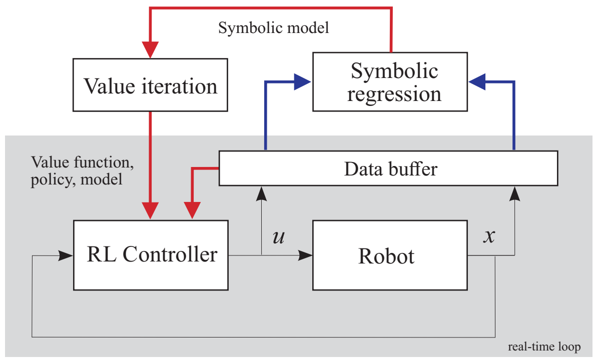


Fig. 1. Model-based reinforcement learning scheme: the RL control loop and the data logging in the buffer run in real time, symbolic regression and value iteration are computed offline in a parallel process.

In the sequel we describe in detail the application of symbolic regression to the process data collected online and the validation of the model obtained.

B. Data collection

Two different situations can be distinguished:

a) Initial model learning: when a control policy is not yet available, the system can be excited by a test signal in order to obtain a sufficiently rich data set. There is a large body of system-identification literature on designing suitable test signals, such as the generalized binary noise (GBN) sequence [30]. The important parameters to be selected are the input signal amplitude, the way the random signal is generated (e.g., the 'switching' probability) and the experiment duration. In the experiments section we detail the choice of these parameters for our examples.

b) Model learning under a given policy: as soon as an acceptable control policy has been obtained, the system can be operated such that it executes the task required. Data from trajectories obtained during the task execution can then be used to further improve the model. Depending on the task, some care has to be exercised when collecting such data. The information content of data collected under steady operating conditions is usually insufficient for model learning purposes. Therefore, also in this situation, the system can be deliberately excited with a test (exploration) signal added to the control input determined by the policy. This test signal typically has different characteristics (e.g., a lower amplitude) than when learning the initial model.

Each data set is divided into a training set and a validation set. The training set is used to evolve symbolic models and the validation set to rank the model according to their predictive power.

C. Symbolic regression

The system to be controlled is described by (1), where the state-transition function f is unknown and we approximate it by an analytic model constructed by using symbolic regression. Symbolic regression is a suitable technique for this task, as we do not have to assume any detailed a priori knowledge on the structure of the nonlinear model. The class of symbolic models is defined as:

$$f(x,u) = \sum_{i}^{n_f} \beta_i f_i(x,u) \tag{7}$$

where the nonlinear functions $f_i(x,u)$, called features, are constructed by means of genetic programming and n_f is the number of features (a user-defined parameter). The coefficients β_i are estimated by least squares.

To evolve the features f_i, we define a set of elementary functions whose combination is deemed sufficient to produce a precise model. These functions can be nested and combined using basic arithmetic operators, such as plus, minus and multiply. To avoid over-fitting, we control the complexity of the regression model by imposing a limit on the maximal depth of the evolved symbolic expressions.

The specific algorithm used in this work is a modified version of Single Node Genetic Programming (SNGP) [31], [32], which is a graph-based genetic programming method. For further details on the modifications we introduced, refer to [33]. SNGP evolves the symbolic model so that the mean-squared error calculated over the training data set is minimized.

IV. Experimental Results

We have carried out experiments with two non-linear control problems: a mobile robot and the under-actuated pendulum swing-up task.

A. Parameters

In all experiments, the SNGP population comprised 500 individuals and the evolution duration was set to 30000 generations. The operator and elementary function set was $\{*, +, -, \sin, \cos, \text{sign}\}$. The maximum depth of the evolved expressions was limited to 7 and the number of features n_f in regression models varied between 1 and 10.[1]

[1]For further information on the SNGP configuration and other implementation details, please refer to our code at the GitHub repository: `https://github.com/erik-derner/SR4RL`

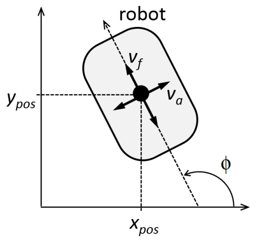
The sampling period we used in all experiments was $T_s = 0.05\,\text{s}$. In RL experiments we used $\gamma = 0.95$ and the value function was approximated by an RBF network with 49 Gaussian basis functions.

B. Complexity

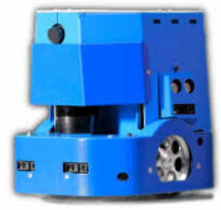
A single run of the SNGP algorithm with the configuration used for the experiments presented in this paper took between 10 seconds and 5 minutes on a standard desktop PC (the algorithm was tested on a computer with a quad-core processor Intel Core i7-4790 @ 3.6 GHz and 32 GB RAM). The computational complexity increases linearly with the size of the training data set.

C. Mobile Robot

1) System description: The state of a two-wheel mobile robot (see Fig. 2) is described by $x = [x_{pos}, y_{pos}, \phi]^\top$, where x_{pos} and y_{pos} are the position coordinates of the robot and ϕ is the robot's heading. The control input is $u = [v_f, v_a]^\top$, with v_f and v_a the desired forward and angular velocity of the robot, respectively.



(a)



(b)

Fig. 2. Mobile robot schematic (a) and photograph (b).

The continuous-time model of the mobile robot dynamics is:

$$\begin{aligned} \dot{x}_{pos} &= v_f \cos(\phi), \\ \dot{y}_{pos} &= v_f \sin(\phi), \\ \dot{\phi} &= v_a\,. \end{aligned} \tag{8}$$

2) Data sets: To verify that symbolic regression can recover the model equations from data, we have first generated noise-free input-state data by using the Euler method to simulate the above differential equations. The discrete-time approximation of (8) then becomes[2]:

$$\begin{aligned} x_{pos,k+1} &= x_{pos,k} + 0.05\, v_{f,k} \cos(\phi), \\ y_{pos,k+1} &= y_{pos,k} + 0.05\, v_{f,k} \sin(\phi), \\ \phi_{k+1} &= \phi_k + 0.05\, v_{a,k}\,. \end{aligned} \tag{9}$$

[2]Note that the value of $T_s = 0.05\,\text{s}$ is already inserted in the equations to facilitate the comparison with the symbolic models presented in the sequel.

The initial state x_0 and the control input u_k at each time step k are randomly chosen within the following limits:

$$\begin{aligned} x_{pos} &\in [-1,1]\,\mathrm{m}, \\ y_{pos} &\in [-1,1]\,\mathrm{m}, \\ \phi &\in [-\pi,\pi]\,\mathrm{rad}, \\ v_f &\in [-1,1]\,\mathrm{m\cdot s^{-1}}, \\ v_a &\in \left[-\frac{\pi}{2},\frac{\pi}{2}\right]\,\mathrm{rad\cdot s^{-1}}. \end{aligned} \tag{10}$$

We generate n_s training samples.

A validation data set was created in order to evaluate the performance of the symbolic models on data different from the training set. The validation data set entries were sampled on a regular grid with 11 points spanning evenly each state and action component domain, as defined by (10). These samples were stored together with the next states calculated by using the Euler approximation.

3) Experiment setup: In *Experiment 1*, symbolic process models were evolved using SNGP on training data sets of varying sizes. The purpose of this experiment was to test the ability of SNGP to recover from the data the process model described by a known state-transition function. Different combinations of the number of features n_f and the size of the training set n_s were tested. As the SNGP algorithm only allows modeling one output at a time, the algorithm was run independently for each of the state components $x_{pos,k+1}$, $y_{pos,k+1}$ and ϕ_{k+1}.

4) Results: The symbolic models found by SNGP were evaluated by calculating the median of the root mean squared error (RMSE) on the validation data set over 30 independent runs. The results are summarized in Table I. The cells with '–' in the configuration with 10 features and only 10 training samples indicate that a symbolic model could not be reliably found, as the least-squares problem was poorly conditioned.

An example of a symbolic process model found with the parameters $n_f = 2$ and $n_s = 100$ is:

$$\begin{aligned} \hat{x}_{pos,k+1} &= 1.0x_{pos,k} + 0.0499998879\,v_{f,k}\cos(\phi_k) \\ \hat{y}_{pos,k+1} &= 1.000000023\,y_{pos,k} + 0.0500000056\,v_{f,k}\sin(\phi_k)+ \\ &\quad + 0.0000000191 \\ \hat{\phi}_{k+1} &= 0.9999982931\,\phi_k + 0.0500000536\,v_{a,k}- \\ &\quad - 0.0000059844 \end{aligned} \tag{11}$$

The coefficients are rounded to 10 decimal places in order to demonstrate the order of magnitude of the symbolic model error w.r.t. the Euler approximation (9). The results show that even with a small training data set, a precise parsimonious symbolic process model can be found based on noise-free data.

In addition, the results show that the choice of the number of features n_f plays a significant role. In general, the RMSE decreases with increasing number of features, whereas the complexity naturally grows by adding more features to the final model. If the number of features is smaller than the underlying function, the results get substantially worse. This is caused by the fact that the least squares method can easily and accurately find the coefficients of the features, while the symbolic regression would have to develop the constants (parameters) by the evolution of the population, which is considerably more difficult. Therefore, it is advised to set the number of features equal to the expected number of terms in the underlying function, which might be known or easy to estimate in practice.

D. Pendulum Swing-up

1) System description: The pendulum system consists of a weight of mass m attached to an actuated link which rotates in the vertical plane, see Fig. 3(a). The state vector is $x = [\alpha, \dot{\alpha}]^\top$, where α is the angle and $\dot{\alpha}$ is the angular velocity of the link. The control input is the voltage u. The continuous-time model of the pendulum dynamics is:

$$\ddot{\alpha} = \frac{1}{J}\cdot\left(\frac{K}{R}u - mgl\sin(\alpha) - b\dot{\alpha} - \frac{K^2}{R}\dot{\alpha} - c\,\mathrm{sign}(\dot{\alpha})\right) \tag{12}$$

with $J = 1.7937\times10^{-4}\,\mathrm{kg\cdot m^2}$, $m = 0.055\,\mathrm{kg}$, $g = 9.81\,\mathrm{m\cdot s^{-2}}$, $l = 0.042\,\mathrm{m}$, $b = 1.94\times10^{-5}\,\mathrm{N\cdot m\cdot s\cdot rad^{-1}}$, $K = 0.0536\,\mathrm{N\cdot m\cdot A^{-1}}$, $R = 9.5\,\Omega$ and $c = 8.5\times10^{-4}\,\mathrm{kg\cdot m^2\cdot s^{-2}}$. The angle is $\alpha = 0$ or $\alpha = 2\pi$ for the pendulum pointing down and $\alpha = \pi$ for the pendulum pointing up.

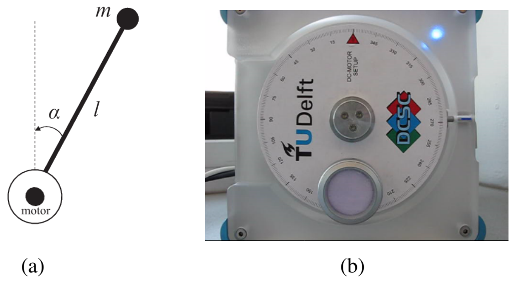


Fig. 3. Inverted pendulum schematic (a) and the real inverted pendulum system (b).

The reward function used in the RL experiments was defined as follows:

$$\begin{aligned} &\rho(x_k, u_k, x_{k+1}) = \\ &= -0.5|\alpha_r - \alpha_k| - 0.01|\dot{\alpha}_r - \dot{\alpha}_k| - 0.05|u_k|, \end{aligned} \tag{13}$$

where $[\alpha_r, \dot{\alpha}_r]^\top$ is a constant reference (goal) state.

2) Data sets: Both simulated and real measured data were used in the experiments with the Pendulum Swing-up system.

The *Euler data set* was generated using the Euler approximation of (12):

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + 0.05\,\dot{\alpha}_k, \\ \dot{\alpha}_{k+1} &= 0.9102924564\,\dot{\alpha}_k - 0.2369404025\,\mathrm{sign}(\dot{\alpha}_k)+ \\ &\quad + 1.5727561084\,u_k - 6.3168590065\,\sin(\alpha_k). \end{aligned} \tag{14}$$

The *Runge-Kutta data set* was created by integrating (12) by using the fourth-order Runge-Kutta method. The initial state was $\alpha = 0$, $\dot{\alpha} = 0$ and the control input was random in the range $u_k \in [-5, 5]\,\mathrm{V}$ for each time step k.

TABLE I

Comparison of symbolic process models for the mobile robot obtained in Experiment 1. The table shows the RMSE medians over 30 runs of the SNGP algorithm for varying number of features n_f and number of training samples n_s.

Variable	n_f	Number of training samples n_s						
		10	20	50	100	200	500	1000
x_{pos}	1	9.44×10^{-1}	1.77×10^{-1}	1.37×10^{-1}	9.98×10^{-2}	7.88×10^{-1}	3.25×10^{-2}	2.87×10^{-2}
	2	7.18×10^{-8}	6.44×10^{-8}	3.19×10^{-9}	2.05×10^{-9}	3.55×10^{-9}	5.93×10^{-9}	1.53×10^{-9}
	10	–	3.78×10^{-5}	2.29×10^{-7}	1.09×10^{-7}	1.45×10^{-7}	5.21×10^{-9}	2.68×10^{-9}
y_{pos}	1	9.11×10^{-1}	9.06×10^{-1}	4.34×10^{-1}	1.39×10^{-1}	1.74×10^{-1}	3.21×10^{-2}	3.16×10^{-2}
	2	3.44×10^{-1}	4.87×10^{-1}	1.81×10^{-8}	1.18×10^{-8}	4.09×10^{-9}	1.93×10^{-8}	2.39×10^{-8}
	10	–	4.48×10^{-4}	2.04×10^{-7}	4.11×10^{-7}	1.91×10^{-7}	1.60×10^{-8}	1.25×10^{-8}
ϕ	1	2.15×10^{0}	9.81×10^{-2}	2.60×10^{-2}	6.44×10^{-4}	6.57×10^{-5}	6.79×10^{-4}	5.55×10^{-3}
	2	1.07×10^{-7}	7.05×10^{-8}	1.36×10^{-8}	6.16×10^{-9}	3.78×10^{-8}	7.78×10^{-9}	5.16×10^{-8}
	10	–	5.35×10^{-6}	1.85×10^{-6}	2.09×10^{-6}	4.01×10^{-8}	6.00×10^{-9}	3.69×10^{-8}

The validation data set was created similarly as in Section IV-C.2. The samples were generated on a regular grid of $31 \times 31 \times 31$ points, spanning the state and action domain: $\alpha \in [-\pi, \pi]$ rad, $\dot{\alpha} \in [-40, 40]$ rad $\cdot$ s^{-1} and $u \in [-5, 5]$ V. In experiments where the Euler data set is used as the input for the SR algorithm, the next states for all samples in validation data sets were calculated using the Euler approximation too. Likewise, the fourth-order Runge-Kutta method was used to calculate the next states for all samples in the validation data sets that were used in experiments with the Runge-Kutta data sets.

The *real data sets* were measured on the real inverted pendulum system shown in Fig. 3(b). The *real random data set* was created initially by applying a uniformly distributed random control input u_k within the range $[-5, 5]$ V at each step k. The length of the random run was 100 steps (5 seconds). This data set is shown in Fig. 4.

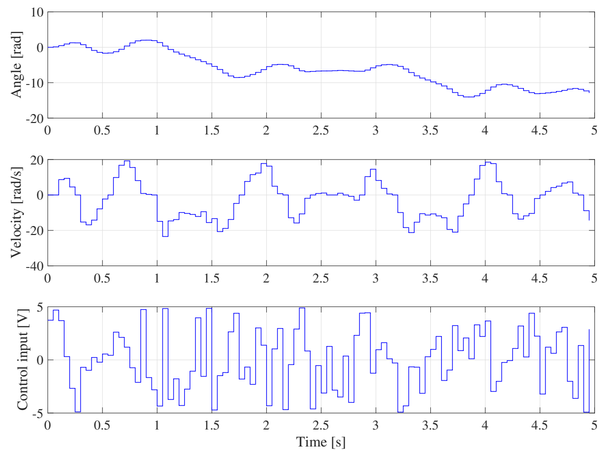


Fig. 4. Initial data set obtained on the real inverted pendulum system as a response to the random input shown in the bottom panel.

Finally, the *real policy data sets* were recorded while applying policy (6) to perform the swing-up task on the real system. The control goal was to stabilize the pendulum in the unstable equilibrium $x_r = [\alpha_r, \dot{\alpha}_r]^\top = [\pi, 0]^\top$, with the control input limited to the range $u \in [-2, 2]$ V. The available torque is insufficient to push the pendulum directly up from the majority of initial states. Therefore, from certain states (e.g., when the pendulum is pointing down), it needs to be swung back and forth to gather energy, prior to being stabilized.

All real data sets were split into training and validation subsets. Every third sample was used for the validation subset, the other samples formed the training subset.

In all experiments, the reported RMSE values are calculated on the respective validation data set.

3) Experiment setup: In *Experiment 2*, SNGP was run in order to test the ability of SR to generate precise models for the Pendulum Swing-up system using the Euler data set, similarly to Experiment 1 on the mobile robot.

Experiment 3 demonstrates how the symbolic process models are evolved using the Runge-Kutta simulation data set with noise. The states $x_n = [\alpha_n, \dot{\alpha}_n]^\top$ with Gaussian noise are defined as

$$\begin{aligned} \alpha_n &= \alpha + \pi\lambda r_{n,1}\,, \\ \dot{\alpha}_n &= \dot{\alpha} + 40\lambda r_{n,2}\,, \end{aligned} \tag{15}$$

where $r_{n,1}$, $r_{n,2}$ are random numbers drawn from a normal distribution with zero mean and a standard deviation of 1, the constant $\lambda \in \{0, 0.01, 0.05, 0.1\}$ controls the amount of noise and the constants π and 40 are chosen so that the added noise is approximately proportional to the range of each variable. The number of features n_f in the SNGP algorithm was set to 10 in order to facilitate the evolution of models capturing the more complex underlying function. This experiment tests the behavior of the method in environments with noisy measurements.

We conclude the experiments with *Experiment 4*, which shows the intended use of the method within RL. First, we constructed 30 symbolic models using the real random data set and then selected the best one based on its performance on the validation set. This model was employed to calculate the policy for the swing-up task (see Section II). We applied the policy to the real system in eight independent runs: four with exploration noise added to the control input and four without. The exploration noise was normally distributed with the standard deviation ranging from 0.2 to 0.5 V. These experiments yield data in important parts of the state space, around the trajectory to the goal state. All eight real policy data sets, each consisting of approximately 50 measurements, were added to the initial random data set. Using this extended data set, 30 refined symbolic process models were learned and the model with the lowest error on the validation set was chosen as the final refined model. Like in Experiment 3, the number of features was set to $n_f = 10$.

4) Results: In *Experiment 2*, the RMSE medians of the symbolic process models over 30 runs were calculated on the validation data set and the results for varying number of features n_f and varying number of samples n_s are summarized in Table II. The cells with '–' in the configuration with 10 features and 10 training samples denote the cases where the symbolic models were not reliably found, as the least squares estimation problem was poorly conditioned.

An example of a symbolic process model found with parameters $n_f = 2$ for α, $n_f = 4$ for $\dot{\alpha}$ and $n_s = 20$ is:

$$\begin{aligned} \hat{\alpha}_{k+1} &= \alpha_k + 0.05\,\dot{\alpha}_k - 0.0000000001, \\ \hat{\dot{\alpha}}_{k+1} &= 0.9102924745\,\dot{\alpha}_k - 0.2369403835\,\mathrm{sign}(\dot{\alpha}_k) + \\ &+ 1.5727561072\,u_k - 6.3168589936\,\sin(\alpha_k) + \\ &+ 0.0000000013. \end{aligned} \tag{16}$$

The error of the found model w.r.t. the Euler approximation (14) is very small. These results confirm that the proposed method can find reliable models even on small data sets. The conclusion we made in the case of Experiment 1 regarding the number of features holds here too.

The results for the Runge-Kutta data set with different levels of added noise in *Experiment 3* are shown in Table III. The algorithm was run 30 times and we report the RMSE median on the validation data set. The results show that the symbolic models are able to approximate the transition function well even on data with a reasonable amount of noise. The use of the Runge-Kutta data results in substantially more complicated models than the data generated with the Euler method. In contrast with Experiment 2, the number of training samples n_s influences substantially the performance. In this case, more training samples are needed to achieve better performance, mainly because the underlying model is much more complex.

In *Experiment 4*, we have shown that SR is able to find process models using data collected on the real system. Already after a short (5 s) interaction under the random input, a symbolic process model is found which is capable of performing the swing-up, see Fig. 6(a). The RMSE median of the initial models over 30 runs of the SNGP algorithm is 1.70×10^{-2} for α and 6.03×10^{-1} for $\dot{\alpha}$. Moreover, Fig. 6(b) shows that the performance of the model further improves after adding data collected while performing the swing-up task with the initial model. Fig. 5 captures a detailed comparison of the swing-up response with the initial and refined model. The histogram in Fig. 7 and a two-sample t-test with unpooled variance applied to the discounted return show that the performance improvement between the policy based on the initial and refined symbolic process model is statistically significant ($p = 2 \times 10^{-22}$). The RMSE median of the refined models over 30 runs of the SNGP algorithm is 1.16×10^{-2} for α and 3.35×10^{-1} for $\dot{\alpha}$. The refined model with coefficients rounded to five decimal places is:

$$\begin{aligned} \hat{\alpha}_{k+1} &= 0.99987\,\alpha_k + 0.04942\,\dot{\alpha}_k + 0.03358\,u_k - 0.00469\,\mathrm{sign}(\dot{\alpha}_k + \\ &+ 2.24017\,u_k) - 0.13979\,\sin(\alpha_k) + 0.00342\,\mathrm{sign}(\cos(u_k)\,(\dot{\alpha}_k + \\ &+ 0.58579)\,(\dot{\alpha}_k - 1.22508)) - 0.01208\,\cos(\cos(\alpha_k))\,\mathrm{sign}(\dot{\alpha}_k - \\ &- 0.84090) - 0.00271\,\cos(\alpha_k)\,(\dot{\alpha}_k - \sin(\cos(u_k) - 0.85154)) + \\ &+ 0.00291\,\cos(8.0\,\alpha_k)\,(\mathrm{sign}(\dot{\alpha}_k) - \cos(\alpha_k) + 1.31607) - \\ &- 0.00150\,\mathrm{sign}(\cos(\dot{\alpha}_k) - u_k + 0.87758)\,(2.24017\,u_k + \mathrm{sign}(\dot{\alpha}_k)) - \\ &- 0.00522, \\ \hat{\dot{\alpha}}_{k+1} &= 0.90686\,\dot{\alpha}_k - 7.53947\,u_k + 0.12185\,\sin(\sin(0.18629\,u_k)) + \\ &+ 0.17778\,\mathrm{sign}(\dot{\alpha}_k - \mathrm{sign}(u_k) + 0.47189) - 0.12185\,\sin(\cos(u_k)) + \\ &+ 0.92217\,\sin(\sin(\alpha_k)) - 5.64919\,\sin(\alpha_k) - 0.04695\,\mathrm{sign}(\cos(\dot{\alpha}_k + \\ &+ 1.0)) - 0.17778\,\cos(\sin(8.17576\,u_k)) + 0.00285\,u_k\,(u_k + \\ &+ 2998.41001) - 0.16883\,\cos(\sin(\sin(\alpha_k))\,\sin(u_k)\,(\dot{\alpha}_k - \\ &- 24.23152))\,\cos(\sin(\dot{\alpha}_k - \cos(\alpha_k) + 1.0)) - 0.11136\,\dot{\alpha}_k\,\cos(\alpha_k - \\ &- 18.84956) - 0.07441\,\cos((\cos(\alpha_k) - 4.13321)\,(\mathrm{sign}(u_k) - \alpha_k + \\ &+ 18.84956))\,(2.0\,\cos(\alpha_k) - 0.5) - 1.25205\,\sin(\alpha_k)\,(\mathrm{sign}(\sin(\alpha_k)) - \\ &- \sin(\alpha_k)) + 0.00222\,\sin(\sin(\alpha_k))\,(\dot{\alpha}_k - 24.23152) + \\ &+ 0.01109\,u_k\,\mathrm{sign}(\dot{\alpha}_k + \mathrm{sign}(u_k))\,(\cos(\sin(\alpha_k)) - \cos(\alpha_k) + 1.0)\,(\dot{\alpha}_k + \\ &+ u_k - \cos(\alpha_k) - \cos(u_k) + 1.0) + 0.26063. \end{aligned} \tag{17}$$

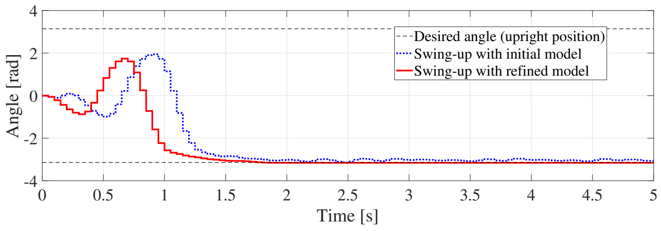


Fig. 5. Comparison of the real swing-up response with the initial model, learnt from the random data, and the refined model, learnt from the random data merged with additional data from eight real swing-up experiments.

We compared our modeling results with local linear regression (LLR) [34]. We selected the Runge-Kutta data set with 1000 samples and zero noise as a reference training set and the regular grid as a validation set (see Section IV-D.2 for details). The LLR memory contained 1000 samples and the number of nearest neighbors was set to 10. The RMSE achieved by LLR was 1.73×10^{-1} for α and 6.93×10^{0} for $\dot{\alpha}$. In both cases, our method achieved a better RMSE by at least one order of magnitude (6.11×10^{-3} for α and 5.04×10^{-1} for $\dot{\alpha}$).

We also compared the results of our method to a neural network. The network had one hidden layer, consisting of 40 neurons, and it was trained using the Levenberg-Marquardt algorithm. The number of neurons in the hidden layer was tuned by testing networks with 5 to 100 neurons and choosing the one that performed best on the validation data. The RMSE achieved on the aforementioned reference data set was 6.82×10^{-2} for α and 2.59×10^{0} for $\dot{\alpha}$. The performance of the neural network is inferior to our method.

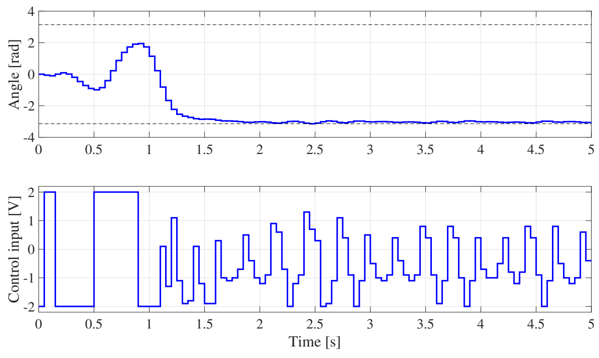
V. Conclusions

Symbolic regression is a suitable tool for constructing process models in reinforcement learning tasks. The advan-

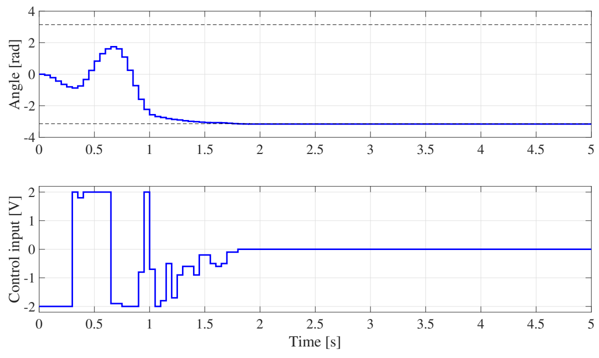
TABLE II

COMPARISON OF SYMBOLIC PROCESS MODELS FOR THE PENDULUM SWING-UP SYSTEM IN EXPERIMENT 2.THE TABLE SHOWS THE RMSE MEDIANS OVER 30 RUNS OF THE SNGP ALGORITHM FOR VARYING NUMBER OF FEATURES n_f AND VARYING NUMBER OF TRAINING SAMPLES n_s.

Variable	n_f	Number of training samples n_s						
		10	20	50	100	200	500	1000
α	1	2.63×10^{-1}	9.19×10^{-4}	3.80×10^{-4}	2.45×10^{-2}	2.28×10^{-3}	1.89×10^{-3}	2.38×10^{-3}
	2	9.76×10^{-8}	2.09×10^{-7}	2.39×10^{-7}	1.06×10^{-7}	1.82×10^{-9}	1.94×10^{-8}	4.60×10^{-9}
	10	–	5.03×10^{-9}	4.44×10^{-7}	4.41×10^{-9}	1.35×10^{-9}	8.45×10^{-10}	4.56×10^{-10}
$\dot{\alpha}$	1	7.55×10^{0}	7.97×10^{-1}	3.17×10^{-1}	2.51×10^{-1}	2.61×10^{-1}	2.34×10^{-1}	5.11×10^{-1}
	4	2.81×10^{-2}	1.12×10^{-6}	5.61×10^{-7}	1.19×10^{-6}	1.61×10^{-6}	8.17×10^{-7}	6.75×10^{-7}
	10	–	5.16×10^{-7}	1.83×10^{-7}	2.64×10^{-7}	4.40×10^{-7}	5.15×10^{-7}	2.66×10^{-6}



(a)



(b)

Fig. 6. A typical real swing-up experiment with the initial model (a) and refined model (b).

TABLE III

COMPARISON OF SYMBOLIC PROCESS MODELS FOR THE PENDULUM SWING-UP SYSTEM IN EXPERIMENT 3. THE TABLE SHOWS THE RMSE MEDIANS OVER 30 RUNS OF THE SNGP ALGORITHM DEPENDING ON THE STANDARD DEVIATION COEFFICIENT λ AND THE NUMBER OF TRAINING SAMPLES n_s.

Variable	λ	Number of training samples n_s		
		20	100	1000
α	0	9.58×10^{-2}	1.79×10^{-2}	6.11×10^{-3}
	0.01	3.95×10^{-1}	1.45×10^{-1}	2.80×10^{-2}
	0.05	1.15×10^{0}	4.89×10^{-1}	1.43×10^{-1}
	0.1	1.90×10^{0}	7.61×10^{-1}	3.55×10^{-1}
$\dot{\alpha}$	0	4.56×10^{0}	7.65×10^{-1}	5.04×10^{-1}
	0.01	4.22×10^{0}	2.28×10^{0}	8.13×10^{-1}
	0.05	7.89×10^{0}	6.07×10^{0}	3.14×10^{0}
	0.1	1.26×10^{1}	9.61×10^{0}	6.65×10^{0}

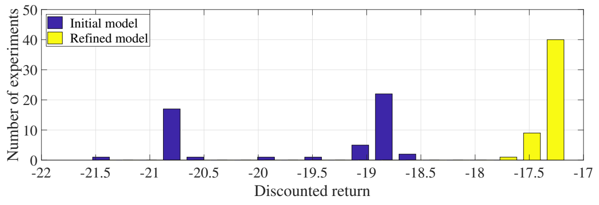


Fig. 7. Histograms of 50 real experiments with the initial model and refined model measured by the discounted return. The performance improvement is statistically significant ($p = 2 \times 10^{-22}$).

tage of symbolic models is that they are parsimonious and understandable for humans. Prior knowledge on the type of nonlinearities and model complexity can easily be included in the symbolic regression procedure.

Experimental validation on the pendulum swing-up task shows that already after a short interaction with the system (5 seconds), an initial symbolic process model is found, which not only accurately predicts the process behavior, but also serves as a reliable model for the design of an RL controller. The performance on the swing-up task significantly improves when adding samples from several runs with the swing-up policy found using the initial symbolic model.

We compared the performance of symbolic regression with two alternative methods: local linear regression and a feed-forward neural network. Both these alternative methods achieve performance in the order of magnitude worse than the symbolic regression. This is most likely due to the small amount of training data.

We propose several future extensions of this work. The main objective is to apply symbolic methods within the entire RL scheme, i.e., also for approximating the V-function, and also to use symbolic models in combination with actor-critic online RL.

In our future work, we will evaluate the performance of the proposed method on higher-dimensional problems. In some cases, especially when using many features, symbolic models tend to be unnecessarily complex. Therefore, we will investigate systematic reduction of symbolic models, especially for higher-order, multi-variable systems.

References

[1] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," *CoRR*, vol. abs/1603.00748, 2016.

[2] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2219–2225.

[3] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 579–610.

[4] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning (ICML)*, 2011, p. 465472.

[5] L. Kuvayev and R. S. Sutton, "Model-based reinforcement learning with an approximate, learned model," *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, pp. 101–105, 1996.

[6] J. Forbes and D. Andre, "Representations for learning control policies," in *Proc. 19th Int. Conf. Mach. Learn. Workshop Develop. Represent.*, 2002, pp. 7–14.

[7] T. Hester and P. Stone, "Intrinsically motivated model learning for developing curious robots," *Artif. Intell.*, vol. 247, no. C, pp. 170–186, June 2017.

[8] N. K. Jong and P. Stone, "Model-based function approximation in reinforcement learning," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '07. New York, NY, USA: ACM, 2007, pp. 95:1–95:8.

[9] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *SIGART Bull.*, vol. 2, no. 4, pp. 160–163, July 1991.

[10] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1071–1079.

[11] R. Lioutikov, A. Paraschos, J. Peters, and G. Neumann, "Sample-based information-theoretic stochastic optimal control," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 3896–3902.

[12] J. Boedecker, J. T. Springenberg, J. Wülfing, and M. Riedmiller, "Approximate real-time optimal control based on sparse gaussian process models," in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Dec 2014, pp. 1–8.

[13] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX: The 9th International Symposium on Experimental Robotics*, M. H. Ang and O. Khatib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 363–372.

[14] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Improved deep reinforcement learning for robotics through distribution-based experience retention," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[15] N. Heess, G. Wayne, D. Silver, T. P. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," *CoRR*, vol. abs/1510.09142, 2015.

[16] I. Grondman, M. Vaandrager, L. Buşoniu, R. Babuška, and E. Schuitema, "Efficient model learning methods for actor–critic control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 3, pp. 591–602, 2012.

[17] R. Lieck and M. Toussaint, "Temporally extended features in model-based reinforcement learning with partial observability," *Neurocomputing*, vol. 192, pp. 49–60, 2016.

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

[19] N. Staelens, D. Deschrijver, E. Vladislavleva, B. Vermeulen, T. Dhaene, and P. Demeester, "Constructing a No-Reference H. 264/AVC Bitstream-based Video Quality Metric using Genetic Programming-based Symbolic Regression," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 99, pp. 1–12, 2012.

[20] C. Brauer, "Using Eureqa in a Stock Day-Trading Application," 2012, Cypress Point Technologies, LLC.

[21] E. Vladislavleva, T. Friedrich, F. Neumann, and M. Wagner, "Predicting the energy output of wind farms based on weather data: Important variables and their correlation," *Renewable Energy*, vol. 50, pp. 236–243, 2013.

[22] J. Žegklitz and P. Pošík, "Linear combinations of features as leaf nodes in symbolic regression," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 145–146. [Online]. Available: http://doi.acm.org/10.1145/3067695.3076009

[23] E. Alibekov, J. Kubalík, and R. Babuška, "Symbolic method for deriving policy in reinforcement learning," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 2789–2795.

[24] M. Onderwater, S. Bhulai, and R. van der Mei, "Value function discovery in markov decision processes with evolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 9, pp. 1190–1201, Sept 2016.

[25] J. Branke, S. Greco, R. Sowiski, and P. Zielniewicz, "Learning value functions in interactive evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 88–102, Feb 2015.

[26] J. Kubalík, E. Alibekov, and R. Babuška, "Optimal control via reinforcement learning with symbolic policy approximation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4162 – 4167, 2017, 20th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317312594

[27] L. Buşoniu, D. Ernst, R. Babuška, and B. De Schutter, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, 2010.

[28] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[29] W. Caarls and E. Schuitema, "Parallel online temporal difference learning for motor control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1457–1468, July 2016.

[30] H. J. Tulleken, "Generalized binary noise test-signal concept for improved identification-experiment design," *Automatica*, vol. 26, no. 1, pp. 37 – 49, 1990.

[31] D. Jackson, "A new, node-focused model for genetic programming," in *Genetic Programming: 15th European Conference, EuroGP 2012, Málaga, Spain, April 11-13, 2012. Proceedings*, A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, Eds. Berlin, Heidelberg: Springer, 2012, pp. 49–60.

[32] ——, "Single node genetic programming on problems with side effects," in *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds. Berlin, Heidelberg: Springer, 2012, pp. 327–336.

[33] J. Kubalík, E. Derner, and R. Babuška, "Enhanced symbolic regression through local variable transformations," in *Proceedings of the 9th International Joint Conference on Computational Intelligence*, 2017, pp. 91–100.

[34] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 11–73, Feb 1997. [Online]. Available: https://doi.org/10.1023/A:1006559212014