Hindawi
Complexity
Volume 2021, Article ID 6617309, 12 pages
https://doi.org/10.1155/2021/6617309

Research Article

Control of Magnetic Manipulator Using Reinforcement Learning Based on Incrementally Adapted Local Linear Models

Martin Brablc,[1] **Jan Žegklitz**,[2] **Robert Grepl**,[1] **and Robert Babuška**[2,3]

[1]*Institute of Solid Mechanics, Mechatronics and Biomechanics, Faculty of Mechanical Engineering, Brno University of Technology, Brno 616 69, Czech Republic*
[2]*Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University, Prague, Prague 16 636, Czech Republic*
[3]*Cognitive Robotics, Delft University of Technology, Delft 2628 CD, Netherlands*

Correspondence should be addressed to Martin Brablc; martin.brablc@vutbr.cz

Received 11 December 2020; Revised 11 November 2021; Accepted 16 November 2021; Published 20 December 2021

Academic Editor: Aydin Azizi

Copyright © 2021 Martin Brablc et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Reinforcement learning (RL) agents can learn to control a nonlinear system without using a model of the system. However, having a model brings benefits, mainly in terms of a reduced number of unsuccessful trials before achieving acceptable control performance. Several modelling approaches have been used in the RL domain, such as neural networks, local linear regression, or Gaussian processes. In this article, we focus on techniques that have not been used much so far: symbolic regression (SR), based on genetic programming and local modelling. Using measured data, symbolic regression yields a nonlinear, continuous-time analytic model. We benchmark two state-of-the-art methods, SNGP (single-node genetic programming) and MGGP (multigene genetic programming), against a standard incremental local regression method called RFWR (receptive field weighted regression). We have introduced modifications to the RFWR algorithm to better suit the low-dimensional continuous-time systems we are mostly dealing with. The benchmark is a nonlinear, dynamic magnetic manipulation system. The results show that using the RL framework and a suitable approximation method, it is possible to design a stable controller of such a complex system without the necessity of any haphazard learning. While all of the approximation methods were successful, MGGP achieved the best results at the cost of higher computational complexity. Index Terms–AI-based methods, local linear regression, nonlinear systems, magnetic manipulation, model learning for control, optimal control, reinforcement learning, symbolic regression.

1. Introduction

A reinforcement learning (RL) agent interacts with the system to be controlled by measuring its states and applying actions according to a policy so that a given goal state is attained. The policy is iteratively adapted in such a way that the agent receives the highest possible cumulative reward, which is a scalar value accumulated over trajectories in the system's state space. The reward associated with each transition in the state space is described by a predefined value function.

Existing RL algorithms can be divided into critic-only, actor-only, and actor-critic variants. The critic-only variants optimize the value function (V-function) that is then used to derive the policy; the actor-only variants work directly on the policy optimization without any need for a value function; and actor-critic variants optimize both functions simultaneously. An example of the actor-only RL variant, often called Q-learning, can be found in [1] and that of the actor-critic variant in [2].

From a different point of view, RL algorithms can be also divided into model-based and model-free variants. Examples of both approaches can be found in [3, 4]. The model-based variants include a model representation of the system to be controlled and can be pretrained in simulation (offline) and then updated when controlling the actual system (online). Model-free methods learn online exclusively through trial and error. Both variants have their specific advantages and disadvantages. We can often find remarks about the model-free approach requiring much more data,

especially in high-dimensional cases [1]. In this paper, we employ the model-based, critic-only variant without any online training so that we can compare different modelling approaches.

We focus on two promising categories of approximation algorithms: genetic programming and local linear regression. Our aim is to contribute to the methodology of choosing the optimal out of dozens of existing modelling algorithms when presented with a specific RL task. This problem arises not only in connection with the RL framework (see [5]) but in modelling of a dynamical system in general [6, 7].

Genetic algorithms (GA) and their many variations are well established as a tool for modelling or parameter estimation of dynamical systems [8, 9]. However, genetic programming as a modelling approach used within RL is relatively new and promises good results with high-dimensional systems where other approaches fail. It creates a continuous-time, globally nonlinear model described by an analytical equation built of combinations of predefined functions [10]. As it is common with genetic optimization algorithms, these methods tend to be computationally demanding. On the other hand, local regression is a well-established modelling approach for model-based RF agents where the model is composed of local linear models, offering fast and computationally cheap approximation. There are several variants of local modelling methods; comprehensive examples of grid-based local linear model structure and data-based local linear regression (LLR) are described in [11, 12], respectively. Even though the use of local regression techniques within RL has been researched in the past, it was mainly based on simple, memory-based approximation methods such as the LLR, which is thoroughly described and examined in [13, 14], and more complex incremental methods such as the receptive field weighted regression (RFWR) [15, 16] or locally weighted projection regression (LWPR) [17] were omitted, with the exception of [18], where the RFWR algorithm was used as a critic approximator. The RFWR and LWPR methods provide significant benefits in lower memory use and higher stability by employing optimization-based (RFWR) or statistical (LWPR) methods to discover the optimal distribution of the local models' areas of validity, that is, the receptive fields.

It is important to benchmark the modelling methods because of the large number of existing approaches, which aim at similar tasks, while there are no simple guidelines on the method choice. Also, the presented algorithms are not yet well established within the RL domain. Finally, studying control algorithms for magnetic manipulation systems has importance on its own because of its application in many industrial fields (medical applications, magnetic levitation systems, etc.), thus leading to the two separate aims of this paper: exploring control algorithms suitable for control of precise magnetic manipulator systems and benchmarking different modelling approaches.

When dealing with real magnetic manipulator systems, we also need to address practical issues that are often neglected in simulations, that is, nonlinearities such as actuator dead zones, saturations, Coulomb friction, signal delays, and so on. These present significant obstacles then implementing the control algorithm on a real system. In some cases, the dead zone and saturation problem can be addressed by nonlinear or adaptive control laws. For example, [19] shows an approach using fuzzy control with Gaussian membership functions, which is in practice similar to the RFWR method, and [20] describes a gain-scheduling adaptive approach to deal with internal system bounds. Using RL to find a control law for a nonlinear system also has the advantage that it can often deal with such disturbances on its own through the optimization process; for example, only a limited range of the actor outputs may be limited, which is the approach utilized in this paper.

In this paper, we also present minor adjustments to the RFWR algorithm in Section 3, proposed to lower the computational complexity while preserving stability when working with low-dimensional problems.

2. Methods

2.1. Magnetic Manipulator. Genetic programming was already applied to nonlinear systems like an inverted pendulum or a collaborative robot [2, 10, 21]. To further investigate the approximation capabilities of these methods, we use a different system–a magnetic manipulator (Magman). This system consists of four coils that are independently operated by separate current controllers and a steel ball that can move freely over the coils; see Figure 1. To ensure that the ball moves only in the measured direction with limits on the edges, it is placed in a groove with 10 mm in size. In this case, we decided to limit the system to the first two coils only, as a system with four inputs is much more complex in terms of the RL computational complexity, while it does not enrich the system with different nonlinearities as it only spatially repeats the same kind of nonlinear behaviour.

The steel ball can be positioned by properly controlling the current and thereby the magnetic force of the coils. The magnetic force a coil exerts on the ball is highly dependent on the distance of the ball from the coil's centre, which introduces a significant nonmonotonic nonlinearity [22–24].

All experiments and simulation were scripted in MATLAB. The coil currents are controlled by stabilized current source modules, which communicate that MATLAB through a USB/RS232 transceiver using the virtual COM port (VCP) protocol on Windows OS. As the ball position is measured with a laser sensor with analog (voltage) output, the Humusoft MF634 IO card was used to measure the signal in real time from the MATLAB environment with a sampling period of 5 ms. Even though the Window OS is not an RTOS, with this sampling frequency, the period jitter is negligible (below 0.1%), and thus, the system can be considered real time.

Table 1 lists the parameters of the magnetic manipulator we use in our experiments. With the task being a precise positioning of an object in a magnetic field, similar concepts can be found in many real-world applications, for example, maglev, microrobots, contactless stirring of chemicals, and so on.

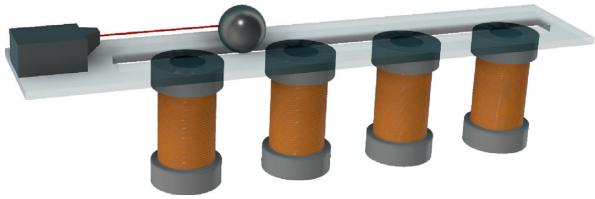

Figure 1: A schematic drawing of the magnetic manipulator system.

Table 1: Magnetic manipulator parameters.

Parameter	Value	Unit
Ball mass	53	g
Ball diameter	20	mm
Distance between edge coils and the ball position limit	20	mm
Distance between coils	25	mm
Maximal coil current	0.6	A
Sampling period	0.005	s

Approximate equations of motion inferred using the first principle method can be found in [24]. The system parameters were either measured directly or estimated using MATLAB parameter estimation toolbox based on measured data.

Generally, the system can be described by a continuous-time, nonlinear state-space model as follows:

$$\dot{x} = f(x, u), \tag{1}$$

where $\mathbf{x} = [x, \dot{x}]^\top$ is the state vector composed of the position x and velocity $\dot{x}$ of the ball, forming the continuous system state space $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^2$; $\dot{x}$ in $\dot{\mathcal{X}} \subset \mathbb{R}^2$ is the state vector derivative; and $\mathbf{u} = [u_1, u_2]^\top$ is the input (action) vector composed of the coil currents. u_1, u_2 form the system input space $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^2$. The nonlinear vector function $f: \mathcal{X} \times \mathcal{U} \longrightarrow \dot{\mathcal{X}}$ thus describes the system dynamics.

In this paper, by modelling the system, we mean approximating the underlying real function f using various methods, which all build upon experimentally measured input-output data. Each data point is formed by corresponding assumed inputs and outputs of the function f – $(\dot{\mathbf{x}}_k, \mathbf{x}_k, \mathbf{u}_k)$. In practice, these data points are corrupted by noise and other disturbances that are assumed to be with zero means.

2.2. SNGP. Single-node genetic programming (SNGP) is a graph-based genetic programming algorithm evolving a population organized as an ordered linear array of interlinked individuals, each representing a single program node [2, 10, 21]. Generally, symbolic regression algorithms try to find a model in the form of an analytical expression for a given data set by forming and evolving the expression out of elemental functions and operations. In our case, the algorithm is based on the assumption that the nonlinear function f in (1) can be efficiently approximated by the following equation:

$$\dot{x} = f(x, u) = \sum_{i,j}^{n_f, n_s} \beta_i f_i\left(x_j, u_j\right), \tag{2}$$

where the nonlinear function f_i, called the feature, is developed by means of genetic programming with n_f being the maximum number of features, n_s number of states, and the coefficients β_i estimated by the least-squares method. The features are constructed from a list of elementary functions that are assumed to be able to produce the required fitting approximation of the presented data. The features can be combined by common operators or nested, but the maximal depth of the expression is limited to avoid overfitting. The symbolic model is evolved so that the mean-squared error over the training data is minimized.

2.3. MGGP. The second GP algorithm we used is called multigene genetic programming (MGGP). As opposed to SNGP, it combines the features defined also by 2 into tree-like structured expressions called genes. The final expression is formed by a linear combination of these genes, which act as the individual features in equation (2). The parameters of this top-level linear combination are again estimated through least squares. Further details about the algorithm can be found in [25]. The actual MGGP implementation we used is extended with linear combinations of features [26] that enable the algorithm to find affine transformations of the feature space via a backpropagation-like technique, thus making it easier for the driving genetic programming algorithm to approximate the nonlinearities.

2.4. Receptive Field Weighted Regression. Receptive field weighted regression (RFWR) is an incremental approximation method that creates a set of local linear models and the corresponding Gaussian basis functions called the receptive fields and gradually updates them to fit the input-output data. The set of local linear models is updated with new data points (called the query points) using a weighted variant of the recursive least squares (RLS) method and the basis functions are updated through a gradient search with the help of heuristic decision rules. It can continually improve the set of models while still providing the best estimation of the approximated function at each query point based on the previously provided data. The original algorithm, first presented in [15, 16], which is the basis we build upon, can be best described by the following pseudocode:

(1) For each new query point $(\dot{\mathbf{x}}_k, \mathbf{x}_k, \mathbf{u}_k)$
(2) For each existing local model
(3) Calculate model weight w according to (4)
(4) If $w >$ activation limit w_{act}
(5) Update model parameters using RLS according to (6) and (7)
(6) Update the corresponding receptive field using (12) and (14)
(7) End
(8) End

(9) If no model was activated

(10) Place a new model at the query point using (15)

(11) Else if two or more models were activated with weight $w >$ pruning limit w_{prun}

(12) Prune the model with the smaller receptive field

(13) End

(14) Calculate the model output as a weighted average of the activated local models

(15) End

Usually, the receptive field activation limit is set as $w_{\text{act}} = 0.001$. This parameter represents the weight limit for a local model to be updated according to the new data and to be included in the output estimation through a weighted average with another activated model. The pruning limit is usually set as $w_{\text{prun}} = 0.7$, which represents the highest acceptable overlap of neighbouring receptive fields.

The RFWR variant described in this paper follows the main outline of the original algorithm [15] with several adjustments and improvements for the sake of stability and computational complexity for low-dimensional problems. This mainly concerns the rules for adding new local models, adjusting their receptive fields, and generalizing the algorithm in a way that the receptive fields are placed and optimized in a lower number of dimensions than the order of the models. This is especially useful in cases when the nonlinearities are significant mainly in one or two dimensions of the state space of the system. This algorithm, in its original implementation, is successfully being used to approximate inverse models of nonlinear systems to be used as a feedforward compensator [27, 28]. Figure 2 shows an example approximation of a complex univariate nonlinear function by the RFWR algorithm.

Each of the local models is represented by a parameter vector $b = [b_1, b_2, \ldots, b_n]^\top$. With the input vector (a query point) $\mathbf{X_q} = [x_1, x_2, \ldots, x_n]^\top$, the output y_q is calculated by

$$y_q = X_q^\top b. \tag{3}$$

The weight w of a local model at a query point X_q is determined by its Gaussian receptive field as follows:

$$w(X) = e^{-1/2\left(X_q - c\right)^\top C^{-1}\left(X_q - c\right)}, \tag{4}$$

with $\mathbf{c} = [c_1, c_2, \ldots, c_n]$ the vector of model centre coordinates and C^{-1} the distance inducing matrix of the basis function (receptive field). The overall output is then calculated as a weighted average of the outputs of the activated local models.

The output estimate of the set of local models and their receptive fields is calculated by the following equation:

$$\widehat{y} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i y_i. \tag{5}$$

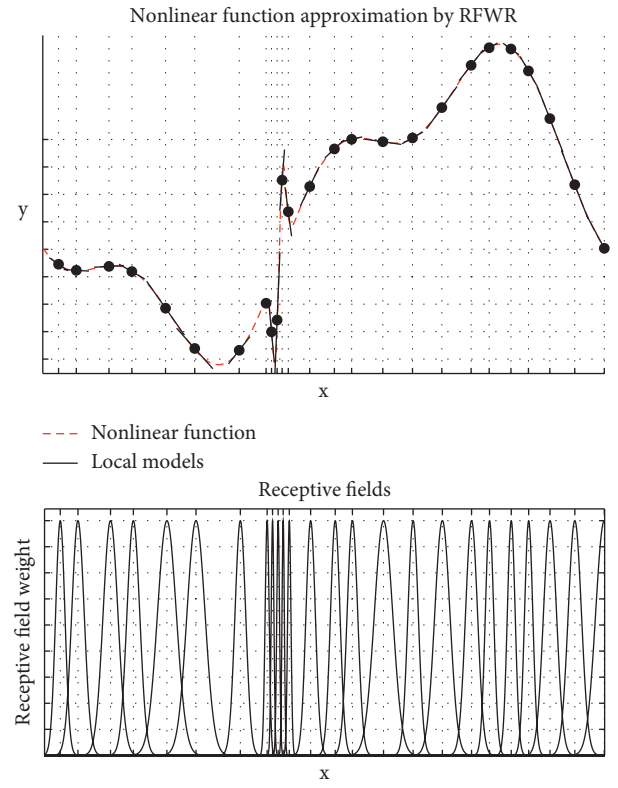

Figure 2: Example of RFWR approximation of a complex nonlinear function and the distribution of receptive fields.

We modified the original RFWR algorithm described in [15] to be used for low-dimensional problems. These modifications consist of the following:

(1) Precise placement of new receptive fields that takes into account the location and dimensions of the existing surrounding receptive fields (see section 3.3)

(2) Heuristic rules for stable updating of the receptive fields (see section 3.1)

(3) Receptive fields can be distributed along a lower number of dimensions than the dimension of the data space (see section 3.2)

2.4.1. Updating parameters of Local Models. When a new query point is acquired, the activated local models are updated using the recursive least-squares algorithm (RLS) according to

$$\begin{aligned} P^{n+1} &= \frac{1}{\lambda}\left(P^n - \frac{P^n X_q^T X_q P^n}{\lambda/w + X_q P^n X_q^T}\right), \\ b^{n+1} &= b^n + \mathrm{w}P^{n+1} X_q^T e, \\ e &= y_q - X_q b, \end{aligned} \tag{6}$$

where P is the covariance matrix of the estimate, λ is a forgetting parameter, and y_q is the acquired output for the actual system state X_q called the query point. The covariance matrix P needs is usually initialized as a diagonal matrix.

2.4.2. Updating dimensions of Basis Functions. To avoid calculating the matrix inversion in (4) for every local model, an upper triangular matrix M is used instead of C. Because of symmetry and positive definiteness, these matrices relate according to

$$C^{-1} = M^T M. \tag{7}$$

To update the receptive field, we update M using a gradient-descent optimization

$$M^{i+1} = M^i - \alpha \frac{\partial J(M)}{\partial M}, \tag{8}$$

of the cost function J as follows:

$$J = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(y_q - y_i\right)^2, \tag{9}$$

where w_i is the activated receptive field weight, y_i is the estimated output of the respective model at the query point $(y_q; X_q)$, and n is the number of local models. The parameter α is the gradient optimization step size. As the calculation of the cost function J according to (9) is computationally very complex, we simplified the optimization algorithm through a set of heuristic decision rules and implemented the optimization as follows:

$$M^{i+1} = M^i - \alpha p \frac{\partial w(M)}{\partial M}$$

$$\frac{\partial w(M)}{\partial M} = \frac{\partial\left(e^{-1/2\left(X_q - c\right)^\top M^T M\left(X_q - c\right)}\right)}{\partial M} =$$

$$= -\left(X_q - c\right)^\top M\left(X_q - c\right) e^{-\frac{1}{2}\left(X_q - c\right)^\top M^T M\left(X_q - c\right)}. \tag{10}$$

This implementation introduces a parameter p, which is an expression of a simple heuristic to decide whether the value of a basis function (weight) at the actual query point should be increased or decreased. This enables to stop updating the distance inducing matrix when a precision criterion is met and to limit the maximal number of local models to avoid overfitting.

Parameter p can be determined by various decision rules. A simple yet effective set, which was used in this research, can be created by using a long-term (cumulated over time) MSE of a particular model according to the data points, which can be described by

$$p = \begin{cases} -1, & \text{if MSE} > \text{MSE}_{\text{lim}} \\ 1, & \text{if MSE} < \text{MSE}_{\text{lim}} \end{cases} \tag{11}$$

2.4.3. Adding New Local Models. During the optimization process, it is possible that no model exceeds the activation limit w_{act}. In such a case, a new local model with a receptive field is added to the approximation set. The centre of the receptive field is automatically placed at the actual query point, and the model parameters are initialized to fit the measured output of the approximated system. What needs to be determined is the area in the state space that should be covered by the newly created receptive field. The original algorithm uses a default diagonal distance inducing matrix for every local model. However, an optimal distance inducing matrix can be determined. Intuitively, the new receptive field should cover the gap between the already existing models. The distance inducing matrix should be initialized as a diagonal matrix with parameters that ensure that the new receptive field does not overlap with any existing one more than a preset limit. In our case, the limit was set to $0.5w_{\text{prun}}$. Since this would be a complex optimization task not suitable for real-time calculation, we simplified the criterion so that the maximal overlapping weight of two models is analyzed only over the line segment connecting their centres. In that case, the distance parameter for initializing the distance inducing matrix can be determined by the following equation.where $v_i = c_n - c_i$ is a vector between the new centre c_n and the centre of a neighbouring receptive field c_i. A two-dimensional example is shown in Figure 3. This method yields a better estimate of the distance inducing matrix of the new receptive field than the fixed initial dimension matrix in the original algorithm as it requires fewer iterations to stabilize and to cover the gap between neighbouring receptive fields.

$$d_i = -\frac{2\,\log\left(w_{\text{prun}}/2\right)}{\left|v_i\right| - \sqrt{-2\,\log\left(w_{\text{prun}}/2\right)/v_i M_i^T M_i v_i}}, \tag{12}$$

The distance parameter d_i has to be calculated for every existing local model, and the minimal distance d_{min} is used to initialize the distance inducing matrix according to

$$M_n^0 = I\sqrt{d_{\text{min}}}, \tag{13}$$

where I is a unity matrix of the corresponding order.

In the specific case of the magnetic manipulator, the inputs of the local models would correspond to (x, u) and the output to $\dot{x}$.

2.5. Reinforcement Learning. Consider the following discrete deterministic state-space model of a system to be controlled:

$$x_{k+1} = f\left(x_k, u_k\right), \tag{14}$$

where $k \in \mathbb{Z}$ denotes discrete time instants, $x_k, x_{k+1} \in \mathcal{X} \subset \mathbb{R}^n$ is the state vector, and $u_k \in \mathcal{U} \subset \mathbb{R}^m$ is the input vector. An RL agent learns to control the system so that it achieves the maximal cumulated reward on a trajectory from the initial state to the desired state [10]. At each state transition, as described by (14), the agent receives a scalar reward according to

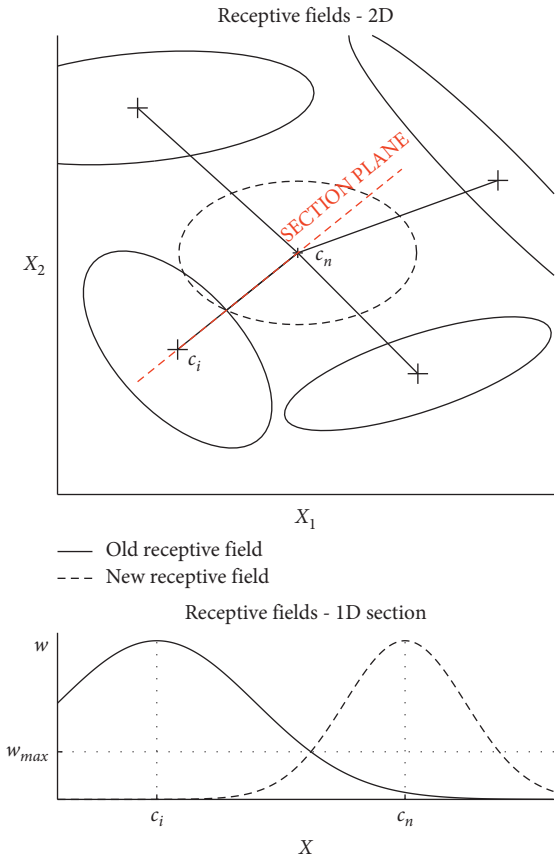

FIGURE 3: Demonstration of adding a new reference field to the set of RFWR local models and matching its dimensions to fit the surrounding reference fields.

$$r = \rho(x_k, u_k, x_{k-1}). \tag{15}$$

The reward function ρ is usually based on the distance of the current state to the goal state. The optimal control law, called the policy, π: $\mathcal{X} \longrightarrow \mathcal{U}$ is determined as follows such that it maximizes the cumulative reward, called the return:

$$R^{\pi} = E\left\{\sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k), x_{k+1})\right\}, \tag{16}$$

where $\gamma \in (0, 1)$ is the discount factor and the initial state x_0 is selected from the state space domain $\mathcal{X}$. The return for any permissible initial state x is captured by the value function V: $\mathcal{X} \longrightarrow \mathbb{R}$ defined as follows:

$$V(x) = E\left\{\sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k), x_{k+1})\right\}, \quad x_0 = x. \tag{17}$$

An approximation of the optimal V-function $\widehat{V}(x)$ can be found by solving the Bellman equation as follows:

$$\widehat{V}(x) = \max_{u \in \mathcal{U}} [\rho(x, \pi(x), f(x, u)) + \gamma \widehat{V}(f(x, u))]. \tag{18}$$

The optimal action can be found as the action that steers the system to a state with maximal value [21]. This corresponds to maximization of the right-hand side of (18):

$$u = \arg\max_{u' \in U} [\rho(x, u', f(x, u')) + \gamma V(f(x, u'))]. \tag{19}$$

3. Experimental Results

We prepared training and validation I/O data sets measured on the magnetic manipulator with random input signals as a list of data points in the form $(\dot{x}_k, x_k, u_k)$. The random input signals (coil currents) were generated in the way that only one coil was active at a time that eliminated possible electromagnetic interactions between them (the coil current was controlled by an HW-based current feedback controller module rendering the transient times negligible). Figure 4 shows an example of a training data set.

Even though the ball's position measurement is very precise, it still contains significant noise. For that reason, the time-domain derivatives of the position (velocity and acceleration) needed for the dynamic model approximation were determined using the Savitzky–Golay filter, which is an FIR filter based on least-squares polynomial approximation able to perform numerical differentiation while filtering the noise simultaneously [29, 30].

Especially for the RFWR implementation, it is important to note that the system's nonlinearity is mainly significant along the position of the ball and the system can be seen as linear in parameters along the other dimensions (acceleration and velocity). In this case, the general model (1) can be rearranged as follows:

$$\ddot{x} = f(x, u) + b_0 \operatorname{sign}(\dot{x}) + b_1 \dot{x} + b_2 \dot{x}^2, \tag{20}$$

where the function $f(x, u)$ represents the significant nonlinearity suitable for local approximation, the term $b_0 \operatorname{sign}(\dot{x})$ represents a simple model of dry friction, the term $b_1 \dot{x}$ represents the viscous friction, and the last term $b_2 \dot{x}^2$ models nonlinear damping caused by electromagnetic induction influencing the steel ball while moving rapidly through a magnetic field. Despite being nonlinear, all of the terms are linear in their parameters and can be modelled globally, which means that the local models share parameters b_0 through b_2.

The term $b_0 \operatorname{sign}(\dot{x})$ in the (20) is quite important in practical situations where Coulomb friction is not negligible. The sign function is often being used to approximate the effects of Coulomb friction wherever there is no significant stiction (difference between static and dynamic friction effects). There are better approximations for simulation purposes, for example, the sigmoid function; however, most of them are not linear in parameters and thus not applicable for RLS parameter estimation.

The same data set was presented to all of the approximation methods (RFWR, SNGP, and MGGP). Due to the stochastic nature of the two algorithms based on genetic programming, the same process was repeated with different pseudorandom seeds. Overall, 30 runs for SNGP and MGGP and 1 run for RFWR were made. Table 2 shows the summary of the MSE results.

Since the MSE of the models with respect to the training set is not sufficient to decide which models are better, two

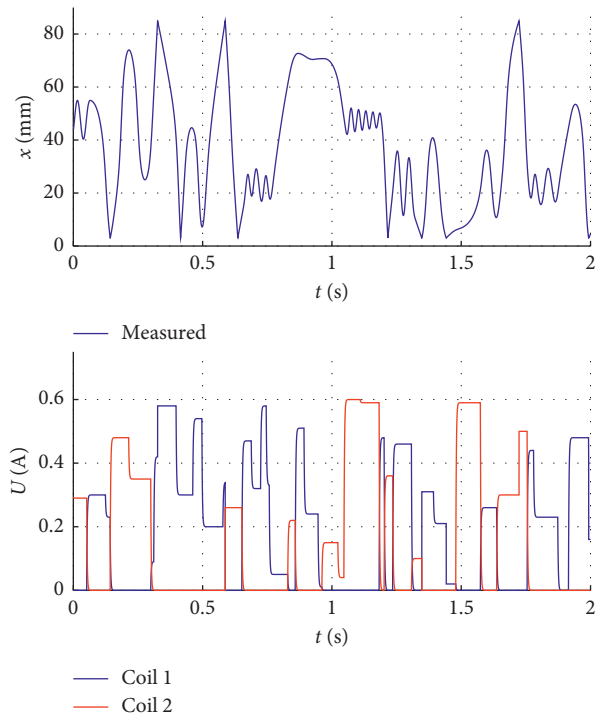

Figure 4: Random input signal (generalized binary noise with random coil switching) with only one coil active at a time for training and validation sets generation.

Table 2: Statistical results of the best models.

Model	Training MSE	5-SAP validation MSE	Control MSE
SNGP 3	$6.53 \cdot 10^{-2}$	$5.71 \cdot 10^{-12}$	$4.02 \cdot 10^{-5}$
SNGP 7	$6.78 \cdot 10^{-2}$	$5.77 \cdot 10^{-12}$	$2.46 \cdot 10^{-4}$
SNGP 10	$5.89 \cdot 10^{-2}$	$5.36 \cdot 10^{-12}$	$4.23 \cdot 10^{-5}$
SNGP 13	$7.28 \cdot 10^{-2}$	$6.37 \cdot 10^{-12}$	$4.28 \cdot 10^{-5}$
SNGP 17	$7.48 \cdot 10^{-2}$	$5.84 \cdot 10^{-12}$	$1.28 \cdot 10^{-4}$
SNGP 18	$7.23 \cdot 10^{-2}$	$5.79 \cdot 10^{-12}$	$2.06 \cdot 10^{-4}$
SNGP 21	$7.65 \cdot 10^{-2}$	$6.41 \cdot 10^{-12}$	$3.40 \cdot 10^{-3}$
SNGP 27	$5.32 \cdot 10^{-2}$	$6.57 \cdot 10^{-12}$	$1.16 \cdot 10^{-4}$
SNGP 29	$5.85 \cdot 10^{-2}$	$6.19 \cdot 10^{-12}$	$9.60 \cdot 10^{-5}$
SNGP 30	$6.93 \cdot 10^{-2}$	$5.99 \cdot 10^{-12}$	$3.75 \cdot 10^{-5}$
MGGP 5	$5.94 \cdot 10^{-2}$	$5.49 \cdot 10^{-12}$	$6.34 \cdot 10^{-5}$
MGGP 8	$4.87 \cdot 10^{-2}$	$5.13 \cdot 10^{-12}$	$4.14 \cdot 10^{-5}$
MGGP 9	$5.69 \cdot 10^{-2}$	$6.18 \cdot 10^{-12}$	$3.80 \cdot 10^{-5}$
MGGP 13	$6.59 \cdot 10^{-2}$	$5.67 \cdot 10^{-12}$	$4.13 \cdot 10^{-5}$
MGGP 16	$5.96 \cdot 10^{-2}$	$5.29 \cdot 10^{-12}$	$6.59 \cdot 10^{-5}$
MGGP 17	$5.99 \cdot 10^{-2}$	$5.08 \cdot 10^{-12}$	$5.46 \cdot 10^{-5}$
MGGP 18	$7.35 \cdot 10^{-2}$	$6.28 \cdot 10^{-12}$	$1.50 \cdot 10^{-4}$
MGGP 21	$5.60 \cdot 10^{-2}$	$5.92 \cdot 10^{-12}$	$6.74 \cdot 10^{-5}$
MGGP 22	$5.55 \cdot 10^{-2}$	$5.74 \cdot 10^{-12}$	$8.84 \cdot 10^{-5}$
MGGP 23	$6.00 \cdot 10^{-2}$	$5.82 \cdot 10^{-12}$	$3.56 \cdot 10^{-5}$
RFWR	$6.25 \cdot 10^{-2}$	$1.24 \cdot 10^{-11}$	$6.57 \cdot 10^{-5}$

separate data sets were measured: one was used for the training of the models and one for validation. However, since the magnetic manipulator is not open-loop stable, the common open-loop validation is not suitable, as every model diverges quickly even though the parameters may be close to ideal due to errors introduced by numerical integration. Therefore, the models were validated in several steps-ahead prediction mode. As we also suspected that only one-sample-ahead validation could be influenced by the remaining noise in the measured signals, we validated the model for 1, 3, 5, 10, 50, 100, and 250 samples ahead. We used the five-step-ahead prediction (5-SAP) as a baseline for selecting the best models for further experiments. The reason to choose five samples is based on an experimentally

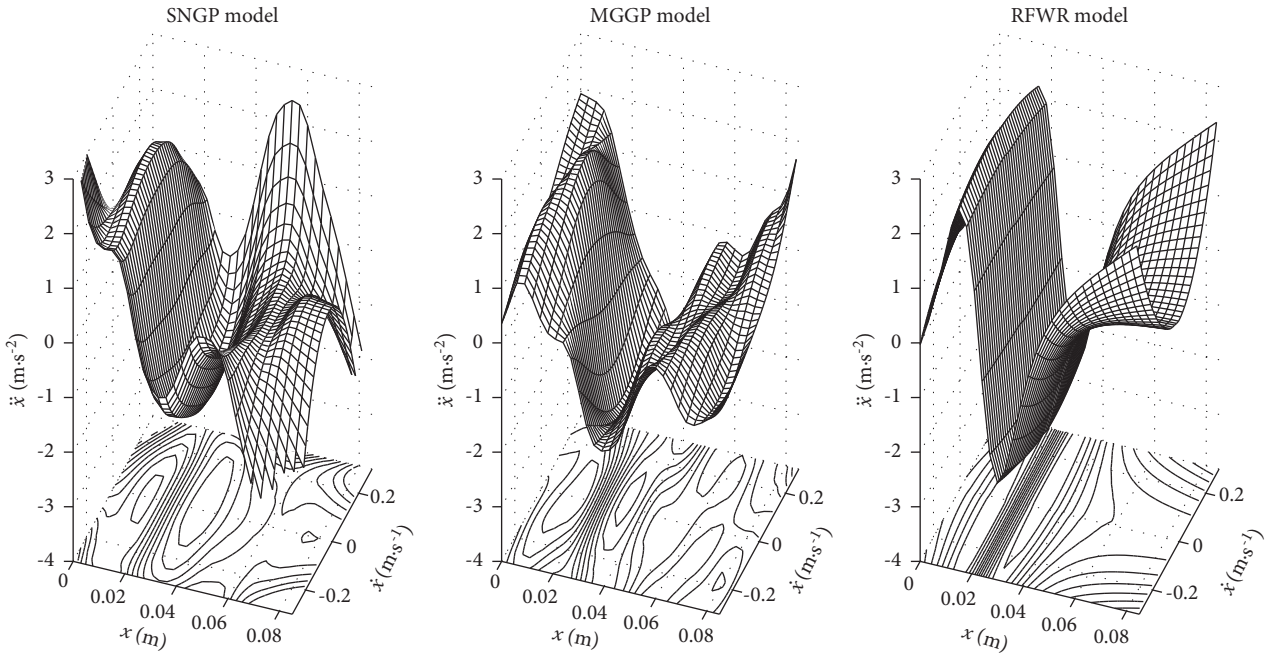

Figure 5: Comparison of approximated dynamic models for the SNGP, MGGP, and RFWR method, best-fitting models from each category. The 3D visualization is shown for constant input $u = [0, 0.6]^T$.

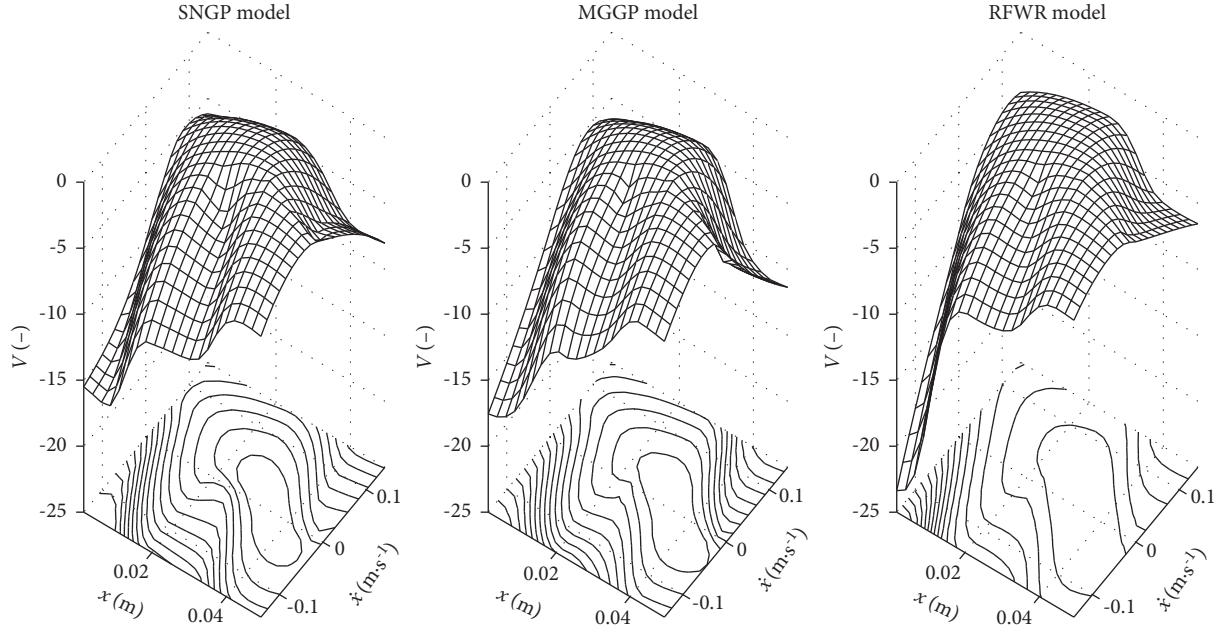

Figure 6: Comparison of V-functions based on the best fitting models from each of the approximation methods: SNGP, MGGP, and RFWR.

validated assumption that shorter intervals do not show the model's imprecision and longer intervals cause even very precise models to diverge randomly.

The n-step-ahead prediction validation is based on a moving frame of n consecutive data points, where the first data point is applied as the initial condition for numerical integration (using the ode45 solver) of the dynamical model being tested. When the simulation reaches the n-th step, an MSE residual is calculated between the corresponding data point and the model prediction. The resulting model validation metric is then calculated as the sum of residuals over each of the prediction frames.

Figure 5 shows examples of the models constructed by each algorithm. As the transition model of the system is four-dimensional, for visualization purposes, the figures show a two-dimensional situation for the input vector set to $u = [0, 0.6]^T$. This corresponds to the situation when the first coil is turned off and the current through the second coil is

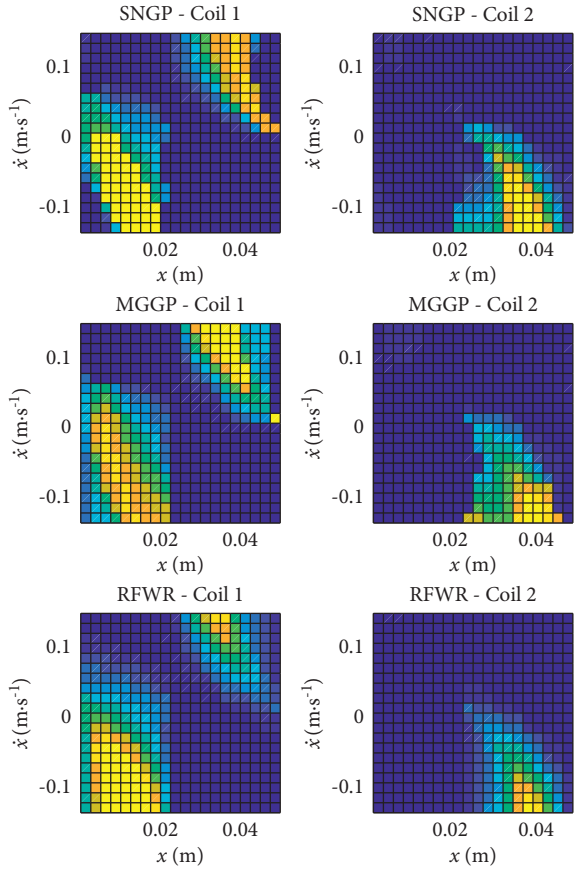

Figure 7: Comparison of policies based on the best fitting models of the magnetic manipulator from each of the approximation methods.

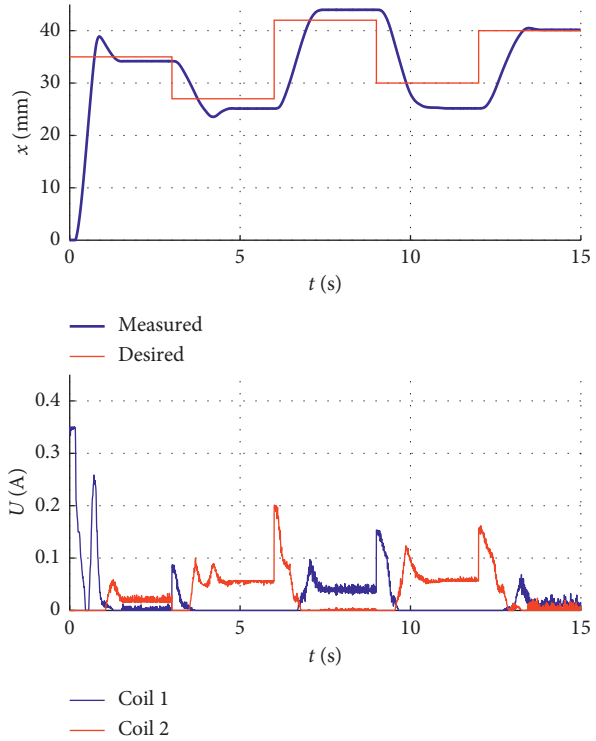

Figure 8: Example of the control performance for a controller based on the SNGP 10 model.

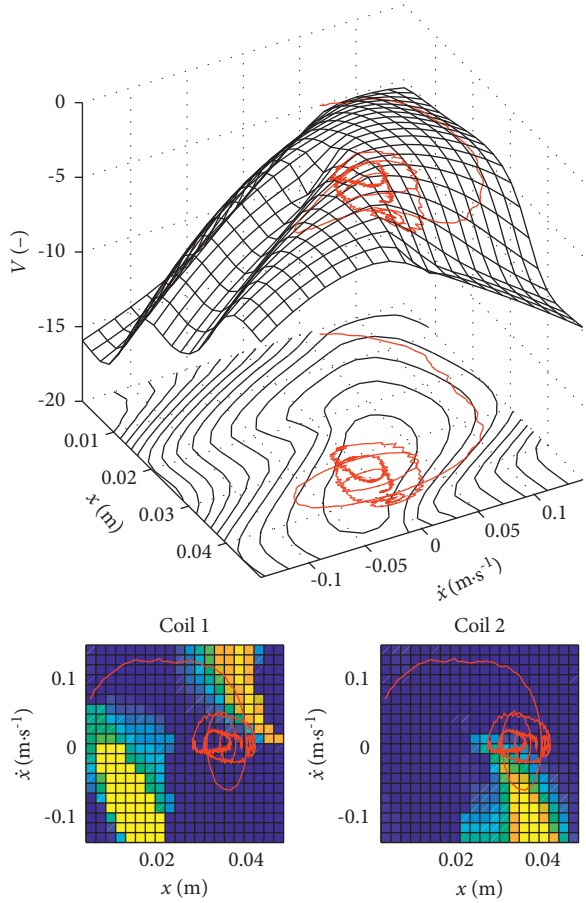

Figure 9: Example of the control sequence trajectory plotted over the policy and the value function based on the model SNGP 10.

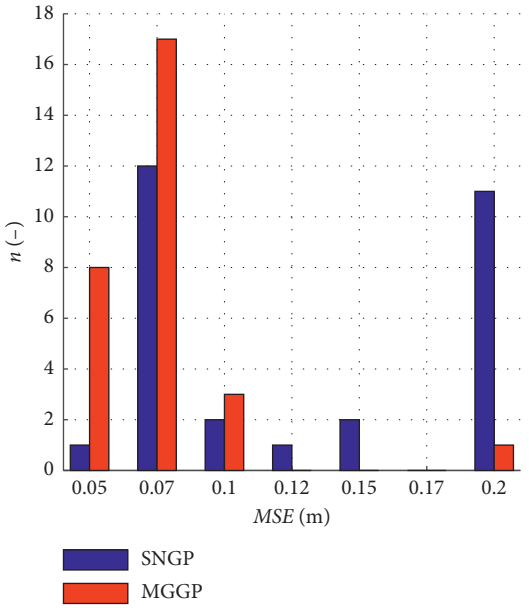

Figure 10: Histogram dividing the tested models into several categories according to their MSE.

set to the maximal value. The plotted functions were cropped in the parts of the state space that are not reachable by the system (typically high velocities close to the edge of the frame).

Based on the 5-SAP validation results, 21 best models (10x SNGP, 10x MGGP, and 1x RFWR) were chosen to be used for RL control of the magnetic manipulator. Based on these models, we calculated the approximations of the optimal V-functions using the fuzzy value iteration [10, 21]. Furthermore, we used equation (19) to calculate the corresponding policies. Figure 6 shows examples of the value functions that resulted in the policies shown in Figure 7.

All the policies were tested on the actual magnetic manipulator. A sequence of five consecutive goal states was chosen as a goal state trajectory, and a corresponding V-function and policy were calculated for each one. During the actual control process, a new policy is used every time the goal state changes. Figure 8 shows an example of the control experiments with an RL controller based on the model SNGP 10. Furthermore, Figure 9 shows the ball's trajectory in the state space plotted over the V-function and the policy.

To conduct all of the experiments, we measured the ball's position using a laser distance sensor, and a PCIe I/O card Humusoft MF634 was used to capture the sensor's analog output signal. The coil currents were driven by a custom dual-channel current controller. Both of the devices are operated from MATLAB.

3.1. Results. We compared the performance of various models of a complex nonlinear system created with three different approximation methods, two of which were based on genetic programming and the third was based on a modified local linear approximation algorithm (RFWR). Based on these models, we used a model-based critic-only RL agent to control the system and validate the results.

Table 2 shows the resulting MSE values from the estimation, validation, and control processes.

Most of the models selected for the actual control experiments were successful in achieving stable control, although they differ in precision. The histogram in Figure 10 shows the number of models for the two GP-based algorithms (SNGP and MGGP), which fall into several MSE categories. The MSE describes the control precision as the mean-squared error between the goal and the actual trajectory of the closed-loop controlled system.

4. Conclusions

First of all, the results show that it is possible to construct even such a complex nonlinear system using the RL framework. The results are not significant in terms of control precision, which depends highly on the specific system, amount of experimental data, and many other factors. An important achievement is a fact that all of the modelling algorithms demonstrated in this paper provide a viable alternative to the commonly used methods while being much less computationally expensive (in the case of RFWR) or much more user-friendly (in the cases of SNGP and MGGP). Also, it was proven that the commonly used RL framework may be built even on tom of imperfect models.

It is clear from the results that the methods that at least partially depend on random number generation (SNGP and MGGP) need to be run repeatedly in search for the best solution, which clearly outperforms the result of the local approximation method (RFWR). On the other hand, the RFWR method requires significantly less computational power than the GP-based methods. Also, it seems that both the SNGP and the MGGP are able to find similarly precise models with the MGGP having a higher probability of converging to the best solution. It is interesting to note that models with better training or validation fit are not always better for control, as can be seen in Table 2. All of the methods provide a useful tool to be used within the reinforcement learning framework with the main advantage of the GP-based approximation method being a form of output (analytical expression) that is understandable and readable and whose complexity is controllable through intuitive parameters. Considering the simplifications made during the simulations and experiments and a relative imprecision during the control processes, there is space for future research in modifying these methods to be suitable for higher-dimensional systems, implementing GPU, handling specific nonlinearities (friction, hysteresis, etc.), and using them also for an approximation of the V-functions and policies.

Data Availability

Research data in the form of simulation and experimental results and MATLAB files are available from the corresponding (first) author upon request (martin.brablc@vutbr.cz).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proceedings of the Proc. 33nd Int. Conf. Mach. Learn. New York, USA: JMLR.org*, pp. 2829—–2838, ACM, New York NY USA, 19 June 2016, https://arxiv.org/abs/1603.00748.

[2] E. Alibekov, J. Kubalik, and R. Babuska, "Symbolic method for deriving policy in reinforcement learning," in *Proceedings of the 2016 IEEE 55th Conf. Decis. Control*, pp. 2789–2795, IEEE, Las Vegas, NV, USA, 12 December 2016, https://ieeexplore.ieee.org/document/7798684/.

[3] I. Koryakovskiy, M. Kudruss, R. Babuška et al., "Benchmarking model-free and model-based optimal control," *Robotics and Autonomous Systems*, vol. 92, pp. 81–90, 2017, https://linkinghub.elsevier.com/retrieve/pii/S0921889016301592.

[4] M. P. Deisenroth and C. E. Rasmussen, "PILCO: a model-based and data-efficient approach to policy search," in *Proceedings of the Proc. 28th Int. Conf. Int. Conf. Mach. Learn*, pp. 465—–472, Omnipress, Bellevue Washington USA, 28 June 2011, https://dl.acm.org/citation.cfm?id=3104482.3104541.

[5] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the Proc. 33rd Int. Conf. Int. Conf. Mach. Learn*, vol. 48, pp. 1329–1338, JMLR.org, New York, USA, 19 June 2016, https://dl.acm.org/citation.cfm?id=3045390.3045531http://arxiv.org/abs/1604.06778.

[6] J. Peters and S. Schaal, "Policy gradient methods for Robotics," in *Proceedings of the 2006 IEEE/RSJ Int. Conf. Intell. Robot. Syst*, pp. 2219–2225, IEEE, Beijing, China, 9 October 2006, https://ieeexplore.ieee.org/document/4058714/.

[7] A. Azizi, F. Entessari, K. G. Osgouie, and A. R. Rashnoodi, "Introducing neural networks as a computational intelligent technique," *Applied Mechanics and Materials*, vol. 464, p. 11, 2013.

[8] A. Azizi, "Applications of artificial intelligence techniques to enhance sustainability of industry 4.0: design of an artificial neural network model as dynamic behavior optimizer of robotic arms," *Complexity*, vol. 2020, p. 3, 2020.

[9] *Applications of Artificial Intelligence Techniques in Industry 4.0*, Springer, Singapore, 2019.

[10] 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, https://linkinghub.elsevier.com/retrieve/pii/S2405896317312594.

[11] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, 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, https://ieeexplore.ieee.org/document/6096441/.

[12] I. Grondman, L. Busoniu, and R. Babuska, "Model learning actor-critic algorithms: performance evaluation in a motion control task," in *Proceedings of the 2012 IEEE 51st IEEE Conf. Decis. Control*, pp. 5272–5277, IEEE, Maui, HI, USA, 10 December 2012, https://ieeexplore.ieee.org/document/6426427/.

[13] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Lazy Learning*, vol. 11, no. 1-5, pp. 75–113, 1997, https://doi-org.ezproxy.lib.vutbr.cz/10.1023/A:1006511328852.

[14] L. Buşoniu, B. De Schutter, and R. Babuška, *Approximate Dynamic Programming and Reinforcement Learning*, pp. 3–44, Springer, Berlin, Heidelberg, 2010, https://link.springer.com/10.1007/978-3-642-11688-9_1.

[15] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, nov 1998, https://www.mitpressjournals.org/doi/10.1162/089976698300016963.

[16] J. Su, J. Wang, and Y. Xi, "Incremental learning with balanced update on receptive fields for multi-sensor data fusion," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 659–665, feb 2004, https://ieeexplore.ieee.org/document/1262536/.

[17] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical learning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 55–69, 2002, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4359172.

[18] D.-H. Lee and J.-J. Lee, "Incremental receptive field weighted actor-critic," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 62–71, feb 2013, https://ieeexplore.ieee.org/document/6246695/.

[19] M. Latifinavid, A. Donder, and E. ilhan Konukseven, "High-performance parallel hexapod-robotic light abrasive grinding using real-time tool deflection compensation and constant resultant force control," *International Journal of Advanced Manufacturing Technology*, vol. 96, p. 6, 2018.

[20] T. Yang, N. Sun, Y. Fang, X. Xin, and H. Chen, "New adaptive control methods for *n*-link robot manipulators with online gravity compensation: design and experiments," *IEEE Transactions on Industrial Electronics*, vol. 69, pp. 1–2022.

[21] E. Alibekov, J. Kubalík, and R. Babuška, "Policy derivation methods for critic-only reinforcement learning in continuous spaces," *Engineering Applications of Artificial Intelligence*, vol. 69, pp. 178–187, 2018, https://www.sciencedirect.com/science/article/pii/S0952197617302993?via%3Dihub.

[22] Z. Hurak and J. Zemanek, "Feedback linearization approach to distributed feedback manipulation," in *Proceedings of the 2012 Am. Control Conf.*, pp. 991–996, IEEE, Montreal, QC, Canada, 27 June 2012, https://ieeexplore.ieee.org/document/6315262/.

[23] J. Zemánek, S. Čelikovský, and Z. Hurák, "Time-optimal control for bilinear nonnegative-in-control systems: application to magnetic manipulation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 16032–16039, 2017, https://linkinghub.elsevier.com/retrieve/pii/S2405896317325430.

[24] J.-W. Damsteeg, S. P. Nageshrao, and R. Babuska, "Model-based real-time control of a magnetic manipulator system," in *Proceedings of the 2017 IEEE 56th Annu. Conf. Decis. Control*, pp. 3277–3282, IEEE, Melbourne, VIC, Australia, 12 December 2017, https://ieeexplore.ieee.org/document/8264140/.

[25] M. Hinchliffe, H. Hiden, B. McKay, M. Willis, M. Tham, and G. Barton, "Modelling chemical process systems using a multi-gene genetic programming algorithm – BibSonomy," in *Proceedings of the Late Break. Pap. Genet. Program. 1996 Conf.*, J. R. Koza, Ed., pp. 56—–65, Stanford Bookstore, Stamford, CA, USA, 28 July 1996, https://www.bibsonomy.org/bibtex/2ba500b4ed22826a3b171019d4a172229/brazovayeye.

[26] J. Žegklitz and P. Pošík, "Linear combinations of features as leaf nodes in symbolic regression," in *Proceedings of the Proc. Genet. Evol. Comput. Conf. Companion - GECCO '17*, pp. 145-146, ACM Press, New York, USA, 15 July 2017, https://dl.acm.org/citation.cfm?doid=3067695.3076009.

[27] R. Grepl, V. Sova, and J. Chalupa, "Adaptive control of electro-mechanical actuator using receptive field weighted regression," *Advanced Mechatronics Solutions*, vol. 393, pp. 621–626, 2016, https://link.springer.com/10.1007/978-3-319-23923-1_87.

[28] R. Grepl and B. Lee, "Modeling, parameter estimation and nonlinear control of automotive electronic throttle using a Rapid-Control Prototyping technique," *International Journal of Automotive Technology*, vol. 11, no. 4, pp. 601–610, aug 2010, https://link.springer.com/10.1007/s12239-010-0072-7.

[29] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, jul 1964, https://pubs.acs.org/doi/abs/10.1021/ac60214a047.

[30] M. Brablc, V. Sova, and R. Grepl, "Adaptive feedforward controller for a DC motor drive based on inverse dynamic model with recursive least squares parameter estimation," in *Proceedings of the Proc. 2016 17th Int. Conf. Mechatronics - Mechatronika, ME*, D. Maga and T. Brezina, Eds., pp. 146–150, IEEE, Prague, Czech Republic, 7 December 2016, https://www.scopus.com/inward/record.url?eid=2-s2.0-85015277645&partnerID=MN8TOARS.