



10.24425/acs.2023.146960

*Archives of Control Sciences*  
Volume 33(LXIX), 2023  
No. 3, pages 565–587

# The buffered optimization methods for online transfer function identification employed on DEAP actuator

Jakub BERNAT  and Jakub KOŁOTA 

Identification plays an important role in relation to control objects and processes as it enables the control system to be properly tuned. The identification methods described in this paper use the Stochastic Gradient Descent algorithms, which have so far been successfully presented in machine learning. The article presents the results of the Adam and AMSGrad algorithms for online estimation of the Dielectric Electroactive Polymer actuator (DEAP) parameters. This work also aims to validate the learning by batch methodology, which allows to obtain faster convergence and more reliable parameter estimation. This approach is innovative in the field of identification of control systems. The research was supplemented with the analysis of the variable amplitude of the input signal. The dynamics of the DEAP parameter convergence depending on the normalization process was presented. Our research has shown how to effectively identify parameters with the use of innovative optimization methods. The results presented graphically confirm that this approach can be successfully applied in the field of control systems.

**Key words:** Stochastic Gradient Descent; ADAM; AMSGrad; DEAP; system identification

## 1. Introduction

In recent times, the rapid development of machine learning algorithms accelerated the development of new methods in many areas. As an example, the crucial part of learning algorithms is the optimization method which searches the most reliable parameters for the solution [12, 21]. The new developments in machine learning are attractive to implement in different areas of science [17, 27]. In our work, recent machine learning ideas are applied to solve the problem of online identification task in control systems [10]. One of the most popular optimization methods widely used in machine learning or neural networks is the Stochastic

---

Copyright © 2023. The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (CC BY-NC-ND 4.0 <https://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits use, distribution, and reproduction in any medium, provided that the article is properly cited, the use is non-commercial, and no modifications or adaptations are made

J. Bernat and J. Kołota (corresponding author, e-mail: [Jakub.Kolota@put.poznan.pl](mailto:Jakub.Kolota@put.poznan.pl)) are with Institute of Automatic Control and Robotics, Poznan University of Technology, Poznań, Poland.

Received 13.01.2023. Revised 22.06.2023.

Gradient Descent (SGD) approach. Stochastic gradient descent is an iterative method for optimizing an objective function. This method can be interpreted as a stochastic approximation of the descent gradient optimization as it replaces the gradient computed from the entire dataset with its estimate computed from a randomly selected subset of the data. Many improvements to the basic stochastic gradient descent algorithm have been proposed and used: AdaGrad (Adaptive Gradient Algorithm) [9, 18], RMSProp (Root Mean Square Propagation) [26], NADAM (Nesterov-Accelerated Adaptive Moment Estimation) [8], ADADELTA (extension of AdaGrad) [28], Adam (Adaptive Moment Estimation) [12, 13] or AMSGrad (extension of Adam) [20, 21]. The Adam algorithm can be explained as a combination of RMSProp and stochastic gradient descent with momentum. It combines the advantages of an adaptive gradient algorithm (AdaGrad) that maintains a per-parameter learning rate (that improves performance in case of problems with sparse gradients) and RMSProp in which parameter learning rates are adapted based on the average of recent magnitudes of the gradients [12]. AMSGrad is Adam's extension towards improving the convergence properties of the algorithm, avoiding large changes in learning speed for each input variable [21].

An interesting idea was also introduced in Reinforcement Learning, one of the most famous unsupervised learning methodologies [25]. Firstly, the RL techniques based on tabular or linear approximations [25]. Then, the more complex problems were analyzed by application of neural networks and deep learning [16, 19]. One of the important problems caused by applying deep neural networks is to obtain convergence and fast batch learning. The application of buffer memory which allows learning by batch is one of the crucial points targeting to success work [19]. In our work, we would like to use this methodology to obtain faster convergence and more reliable parameter estimation. This is a new approach in the problem of identification of control systems [10, 24].

In this work, the online identification problem is analyzed on the DEAP actuator which belongs to a new group of devices in control systems and robotics [7, 11]. The electroactive polymers are interesting materials which enable electrical to mechanical power conversion in a silent and soft way [4, 22, 23]. The actuator uses an electric field to generate the force, which is the main difference with traditional electromagnetic actuators. The recent developments show that this device has wide interesting features like fast movement [3] or wide range movement [22]. In compare to many smart materials like IPMC, they are practical applicable. The DEAP actuators are very interesting as the device testing of identification algorithms due to their nonlinear and double scaled time response. Furthermore, the problem of identification is based on the output feedback (not state feedback), which makes the problem more challenging.

In this work, the identification of the continuous transfer function is analyzed. The main goal of the work is to show that techniques known from machine learning are useful in the field of system identification. It will be shown that

new optimization techniques like Adam and AMSGrad more efficiently estimate parameters than typical methods. Furthermore, the application of replay buffer can improve the performance and efficiency of the estimator.

The plan of the work is as follows. Firstly, the introduction to the subject is shown. Secondly, the identification schema is described and parameter estimation performed by the optimization techniques. Next, the simulations with the transfer function are shown, which illustrates the main features of the described algorithms. Finally, the experiment with DEAP actuator is shown and the identification based on the acquired data is described.

## 2. Identification schema

In this work, the problem of identification of continuous transfer function is analyzed. The system is described by

$$G(s) = \frac{b_m s^m + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} = \frac{Y_p(s)}{U_p(s)}, \quad (1)$$

where  $U_p(s)$ ,  $Y_p(s)$  are the input and output respectively,  $b_m, \dots, b_0, a_{n-1}, \dots, a_0$  are transfer function coefficients,  $m$  is the degree of numerator and  $n$  is the order of the transfer function,  $n - m$  is the relative order of system. In the input-output system identification, the crucial step is to convert the transfer function to a linear in-parameter regressor. The standard way for the above function is known for Model Reference Adaptive Controller, described, for instance in the book [10]. The mentioned approach requires applying filter:

$$\Lambda(s) = s^n + \lambda_{n-1} s^{n-1} + \dots + \lambda_0 \quad (2)$$

to eliminate the derivatives. Therefore, the transfer function is represented as:

$$Y_p(s) = - \sum_{i=0}^{n-1} (a_i - \lambda_i) \frac{s^i}{\Lambda(s)} Y_p(s) + \sum_{i=0}^m b_i \frac{s^i}{\Lambda(s)} U_p(s). \quad (3)$$

Thus, the linear in-parameter regressor is obtained as:

$$Y_p(s) = \varphi_y^T(s)(a - \lambda) + \varphi_u^T(s)b = \varphi_{uy}^T(s)\theta_\lambda, \quad (4)$$

where  $a$ ,  $b$  and  $\lambda$  are vector of coefficients and  $\varphi_u$ ,  $\varphi_y$  are the regressor parts related with  $u_p$  and  $y_p$  respectively. The details of the signal definition are presented in the Appendix A.

In our work, we consider the hybrid schema of an identifier. This means that the transfer function is described in a continuous domain. However, the

identification is performed in discrete time. Our motivation for such choice is to keep the system description in a continuous domain which is more fitted for physical objects. However, the identification is usually computed in the processor unit.

In the discrete space, the output  $y_p(t_k)$  is considered as:

$$y_p(t_k) = \mathcal{L}^{-1} \{G(s)U_p(s)\} (t_k). \quad (5)$$

In practical applications, the above signal does not have to be calculated as long as it is measured. However, we would like to notify that it is the sampled output of the continuous system.

Signal  $\varphi_{uy}(t)$  is obtained by calculating filters with input  $u_p(t)$  and  $y_p(t)$  in discrete space. To make the system structure simple, the filters are firstly written in the continuous state space:

$$\begin{aligned} \dot{\omega}_y(t) &= F\omega_y(t) - l y_p(t), \\ \dot{\omega}_u(t) &= F\omega_u(t) + l u_p(t), \end{aligned} \quad (6)$$

where the state  $\omega_u$  and  $\omega_y$  creates the signal  $\varphi_{uy}(t) = [\omega_y(t) \ \omega_u(t)]$  (see Appendix A for definition of  $F$ ,  $l$ ). Then, to calculate the state in discrete space, the above filters are converted to a discrete system by a generalized bilinear transformation with coefficient 0.5 [29]:

$$\begin{aligned} \omega_y(t_{k+1}) &= F_d \omega_y(t_k) - l_d y_p(t_k), \\ \omega_u(t_{k+1}) &= F_d \omega_u(t_k) + l_d u_p(t_k). \end{aligned} \quad (7)$$

The estimation of  $\hat{y}_p(t_k)$  is given by:

$$\hat{y}_p(t_k) = \hat{\theta}_\lambda^T \varphi_{uy}(t_k). \quad (8)$$

It is worth to point out that even in the case of true parameters, the discretization causes that the regressor does not fit exactly:

$$y_p(t_k) - \theta_\lambda^T \varphi_{uy}(t_k) = \delta_d, \quad (9)$$

where  $\delta_d$  is small discretization error. It is in opposite to (4) where the discretization error does not exist. The signal  $\varphi_{uy}(t_k)$  is a virtual signal, because it does not exist in the physical device. Furthermore, we assume that  $\varphi_{uy}(s)$  is calculated in the discrete domain (in the processor unit), so it is not possible to obtain it as in (5). It is also worth to point out that the discretization error  $\delta_d$  depends on the sampling time, hence it is possible to decrease it by choosing a higher sampling rate.

In the simulation section, we will show that it is useful to perform the standardization of the signal  $y_p$  and  $\varphi_{uy}$ . Transforming the signals  $y_p$  and  $\varphi_{uy}$  to have zero mean and 1 as standard deviation will improve the transients and make the parameter transient less dependable on the signal range. Let us consider the regression written as:

$$y_p(t_k) = \sum_{i=1}^N \varphi_{uy,i}^T(t_k) \hat{\theta}_{\lambda,i} + \epsilon_p, \quad (10)$$

where  $N = n + m + 1$  is the number of unknown parameters. The signals are assumed to be working around working points, so their mean is about 0. Hence, only the range of the signal must be standardised. Let us consider the following transformation:

$$\frac{y_p(t_k)}{\sigma_y} = \sum_{i=1}^N \frac{\varphi_{uy,i}^T(t_k)}{\sigma_i} \hat{\theta}_i + \epsilon, \quad (11)$$

where  $\hat{\theta}_i = \frac{\sigma_i}{\sigma_y} \hat{\theta}_{i,\lambda}$  and  $\sigma_y, \sigma_i$  is the range or standard deviation of  $y$  and  $\varphi_{uy,k}$ . Finally, if the standardization is applied, the regressor is replaced by:

$$y(t_k) = \sum_{i=1}^N \varphi_i^T(t_k) \hat{\theta}_i + \epsilon = \varphi^T(t_k) \hat{\theta}(t_k) + \epsilon, \quad (12)$$

where  $y(t_k) = \frac{y_p(t_k)}{\sigma_y}$  and  $\varphi_k(t_k) = \frac{\varphi_{uy,k}(t_k)}{\sigma_k}$ .

In the identification schema, it is common to apply normalization to assure error signal boundness in the case of ill posed problems [10]. Therefore, in our work, the identification error with normalization is given by:

$$\epsilon_I(t_k) = \frac{y(t_k) - \hat{y}(t_k)}{m^2}, \quad (13)$$

$$m^2 = 1 + \varphi^T(t_k) \varphi(t_k).$$

It is worth to point out that the above normalization has a different target (ensuring the boundedness of signals in  $\mathcal{L}_\infty$  sense) than the standardization in machine learning.

In our work, the parameters are updated based on the same schema:

$$\hat{\theta}(t_k) = \hat{\theta}(t_{k-1}) + \Delta(t_k), \quad (14)$$

where  $\Delta(t_k)$  is calculated by the optimization algorithm discussed in the next section.

## 2.1. Optimization algorithms

In our work, two cost functions are considered. The first is the instantaneous cost function and the second is the integral cost function. Let us firstly analyze the optimizers based on the instantaneous cost function, defined by:

$$f_t(\hat{\theta}_t) = \frac{1}{2} \frac{(y_t - \hat{\theta}_t^T \varphi_t)^2}{1 + \varphi_t^T \varphi_t}. \quad (15)$$

where  $\hat{\theta}_t = \hat{\theta}(t_k)$  and  $\hat{\theta}_{t+1} = \hat{\theta}(t_{k+1})$  (the similar notation is applied in other functions in this section). The gradient of the function  $f_t(\hat{\theta}_t)$  is given by:

$$g_t = -\epsilon_{I,t} \varphi_t. \quad (16)$$

Due to the availability of gradient, it is possible to calculate the update rule  $\Delta_t$  based on gradient-based algorithms. The simplest possibility is to run the gradient descent method, which moves the parameters proportionally to gradient:

$$\Delta_t = -T_p \gamma g_t = T_p \gamma \epsilon_{I,t} \varphi_t, \quad (17)$$

where  $\gamma$  is the tunable gain. In the current state of machine learning literature, many tries have been done to improve the above basic algorithm.

Adaptive Movement Estimation (Adam) algorithm was described in 2015 and its value is an automatic adaptation with a separate learning rate for each parameter in the optimization problem [13]. This has allowed it to gain great popularity, especially in neural network applications, because instead of the classical stochastic gradient descent procedure to update network weights iterative based on training data. The algorithm has many advantages, which also perfectly fit into the processes of identifying parameters of executive objects in automatics and robotics. It is computationally efficient, straightforward to implement, little memory requirements, and the algorithm does well on online and nonstationary problems (for example, in a noisy environment). To our knowledge, no one has yet shown the results of using the Adam algorithm in identifying the parameters of DEAP actuators. The pseudocode of the Adam algorithm is presented in Algorithm 1 [13]. Adam computes individual adaptive learning rates for different parameters from estimates of the first  $m_t$  and the second  $v_t$  moments of the gradients. At the beginning of the algorithm, the gradients  $g_t$  are calculated for the current time step. Function  $f_t(\theta_t)$  defines stochastic objective function which is minimized with regard to its parameters  $\theta$  at the given timestep. The stochasticity comes from the evaluation of random learning batches. With  $\nabla_{\theta} f_t(\theta_{t-1})$  we denote the gradient, i.e. the vector of partial derivatives of  $f_t$  with regard to  $\theta_{t-1}$  parameters. In the identification algorithm, the gradient is expressed by (16). Next, the first and the second moments are updated using the gradients, the

squared gradients, and parameters  $\beta_1$  and  $\beta_2$  which control the exponential decay rates of moving averages of both gradients. In the next part of the algorithm, the first and the second moments are bias-corrected. Finally, the parameter values for the current iteration are calculated using  $\alpha$  which is the step size parameter, and small value  $\epsilon$  which ensures the method does not encounter a divide by zero error.

---

**Algorithm 1** The pseudocode of the Adam algorithm
 

---

- 1:  $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$
  - 2:  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
  - 3:  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
  - 4:  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
  - 5:  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
  - 6:  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
- 

A limitation of Adam is that it can increase the step size in some cases where it is undesirable. This can be detrimental to the overall performance of the algorithm [20]. AMSGrad which was presented in 2018, is an extension of the Adam's gradient method by improving the algorithm's convergence properties. It is characterized by avoiding large sudden changes in the learning speed for each input variable, which accelerates the optimization process [20]. AMSGrad maintains a maximum of the second moment vector and uses it to bias the correct gradient used to update the parameter, instead of the moment vector itself. The calculation of the gradient  $g_t$  and the update of the biased first and second moments are carried out in the same way as in the Adam method in Algorithm 1. The key differences in the AMSGrad pseudocode are presented in Algorithm 2.

---

**Algorithm 2** The pseudocode of the AMSGrad algorithm
 

---

- 1:  $\hat{v}_t \leftarrow \max(\hat{v}_{t-1}, v_t)$
  - 2:  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot m_t / (\sqrt{\hat{v}_t} + \epsilon)$
- 

Since it holds the maximum of all  $v_t$  in the current time step and uses this maximum value for normalizing the running average of the gradient, the AMSGrad results in a nonincreasing step size and avoids the pitfalls [20]. Opposite to gradient-based methods, the integral cost function [10] can be applied to search the parameters:

$$\begin{aligned}
 J_{\text{integral}}(\hat{\theta}) &= \frac{1}{2} \int_0^{t_c} e^{-\beta(t_c - \tau)} \epsilon_c^2(t_c, \tau) m^2(\tau) d\tau, \\
 \epsilon_c(t_c, \tau) &= \frac{y(\tau) - \hat{\theta}^T(t_c) \varphi(\tau)}{m^2(\tau)},
 \end{aligned} \tag{18}$$

where  $\beta > 0$  is a forgetting factor and  $t_c$  is current time in continuous domain. The main difference between the instantaneous and integral cost function is the scope of working. In the first case, only the current error is analyzed, which in the second case the previous errors are also taken into account. The above formulation assumes that all functions are continuous in time. In our work  $y$  and  $\varphi$ , which are defined in (12), are available only in discrete time points, the continuous solution of the above problem must be discretized. Taking into account the Euler forward method, we obtain the following expressions:

$$\begin{aligned}
 \hat{\theta}_{t+1} &= \hat{\theta}_t - T_p \Gamma (R_t \hat{\theta}_t + Q_t), \\
 R_{t+1} &= R_t + T_p \left( -\beta R_t + \frac{\varphi_t \varphi_t^T}{m^2} \right), \\
 Q_{t+1} &= Q_t + T_p \left( -\beta Q_t - \frac{y_t \varphi_t}{m^2} \right),
 \end{aligned} \tag{19}$$

where  $T_p$  is the sampling time and initial conditions are set  $R_0 = 0$  and  $Q_0 = 0$ .

## 2.2. Replay buffer experience

Recently, replay buffer experience enjoys a great success and is widely used especially in various deep reinforcement learning (RL) algorithms [1, 15]. The idea of this method is to use previous experience and sample data from it, instead of using only the latest sample. This methodology allows for faster convergence and more reliable estimation of parameters in the identification process. The algorithm implemented by the authors uses a fixed-size buffer (variable *buffer\_size*) with new data added to the end of the buffer, so that the oldest experience is pushed out of it. The size of the variable *batch\_size* defines the number of past samples that are randomized to fill the array and are then input arguments to the optimization process. It is important that the current sample is added each time at the end of the batch. Taking a batch of samples from the replay buffer for optimization is also known in the literature as experience replay [30]. In addition to making better use of the knowledge already acquired, it also allows breaking harmful correlations, and positively improves the convergence of optimization algorithms. The pseudocode of the replay buffer function used for the optimization algorithm is shown in Algorithm 3, where the input buffered parameters are a vector regressor  $\varphi$  of size  $N \times 1$  and a scalar displacement  $y$ , which are defined in (12). The initial value of the global variable *buffer\_index* is zero and the *Optimizer* function carries out the equation (14) according to the currently used optimization algorithm.



---

**Algorithm 3** The pseudocode of ReplayBuffer algorithm
 

---

**Require:**  $buffer\_size = const$   
 $batch\_size = const$   
 $buffer\_index = 0$

**Ensure:**  $\varphi$  of size  $N \times 1$  and a scalar displacement  $y$

$buffer_\varphi \leftarrow \varphi$   
 $buffer_y \leftarrow y$   
 $last\_index \leftarrow buffer\_index$   
 $buffer\_index \leftarrow buffer\_index + 1$

**if**  $buffer\_index = buffer\_size$  **then**  
 $buffer\_index = 0$   
 $current\_buffer\_size = buffer\_size$

**else**  
 $current\_buffer\_size = buffer\_index$

**end if**

**if**  $current\_buffer\_size > batch\_size$  **then**  
**random**  $batch\_size - 1$  indexes **from** 0 **to**  $current\_buffer\_size$   
**append**  $last\_index$   
**for**  $n$  **in range** ( $batch\_size$ )  
 $index \leftarrow indexes[n]$   
 $Optimizer(buffer_y[index], buffer_\varphi[index])$

**end for**

**else**  
 $Optimizer(buffer_y[last\_index], buffer_\varphi[last\_index])$

**end if**

---

### 3. Simulation analysis

In this section, the simulations showing the features of the identification algorithm are presented for numerical data and experimental data. Firstly, four simple examples are shown based on the numerical data generated by a simple transfer function. Therefore, the scope is focused on the main features of the presented algorithm. Secondly, the experimental data are analyzed to show the influence of all imperfections related to online identification in practical applications.

#### 3.1. Influence of buffering

In the presented example, the influence of the replay buffer is shown with the following transfer function:

$$G_1(s) = \frac{b_0}{s + a_0}, \quad (20)$$

where the parameters are set as  $b_0 = 1.25$  and  $a_0 = 0.5$ . The above transfer function is transformed to the regressor representation (8) with first order  $\Lambda(s) = s + 5$ . To generate data for identification, the input is a square waveform with

amplitude 1 and period 1.25 s and the system output is found by simulating the transfer function (20) for every step with size 1 ms. Then, the identification process is performed with the optimization algorithm AMSGrad (with gain  $\alpha = 5 \times 10^{-4}$ ) and Gradient (with gain 2). The algorithms are run with and without Replay Buffer algorithm 3. The parameters of Replay Buffer algorithm are as follows: the buffer size is 5000 and single batch is 20. The transients of the estimated parameters are presented in Figure 1. It is visible that buffering reduces the error of parameters for different algorithms. This is also visible in Table 1 where the performance index is found:

$$J_p = \sum_{k=0}^{N_s} e_p^2(t_k), \quad (21)$$

where  $p$  is the parameter of transfer function ( $a_i$  or  $b_i$ ),  $N_s$  is the number of samples in simulation,  $e_p(t_k) = \hat{p}(t_k) - p(t_k)$  is the error between the estimated  $\hat{p}(t_k)$  and true  $p$  value of parameter. In Table 1, it is visible that the performance index  $J_p$  is much lower for algorithms with buffering and the reduction is up to 50%. Furthermore, in the case of AMSGrad algorithm the replay buffer eliminates the overshoot for both parameters as it is shown in Fig. 1.

Table 1: The performance indexes of parameter error for algorithm with and without buffer

Name	$J_{a_0}$	$J_{b_0}$	$J_{\text{sum}}$
AMSGrad, with buffer	$2.14e + 03$	$6.40e + 01$	$2.20e + 03$
AMSGrad, no buffer	$5.19e + 03$	$5.89e + 02$	$5.77e + 03$
Gradient, with buffer	$9.15e + 02$	$3.87e + 02$	$1.30e + 03$
Gradient, no buffer	$8.41e + 02$	$1.64e + 03$	$2.48e + 03$

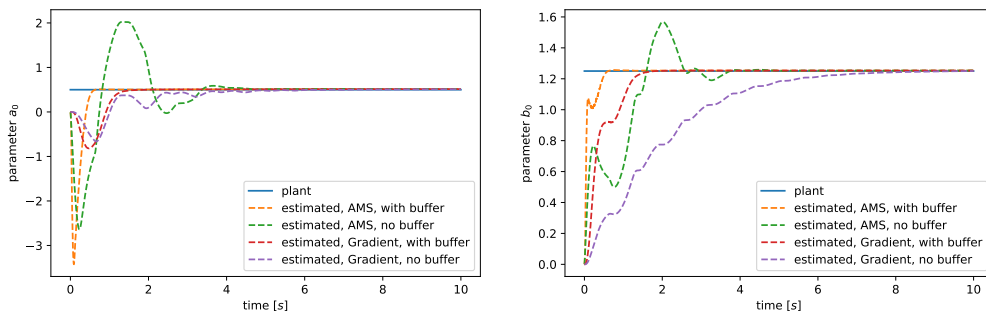


Figure 1: The example of system  $G_1(s)$  with and without buffer

### 3.2. Input signal amplitude

In this section, the influence of the input signal level on the identification process performance is analyzed. The transfer function  $G_1(s)$ , the filter  $\Lambda(s)$  and their parameters are the same as in the previous example. The analyzed optimizers are AMSGrad with  $\alpha = 2.5 \times 10^{-4}$  and Gradient with gain 2. The input signal has the same shape of the square waveform for all cases. However, different is the level of amplitude which is set to 10, 1 and 0.1 of the nominal signal. The data generation and system identification were performed for all cases and the results are shown in Figure 2. It is visible that the level of input signal has a small influence on the convergence of AMSGrad. However, Gradient optimizer is very sensitive on the level of amplitude, which is visible especially for 0.1. The same conclusion can be observed analysing the performance index (21) shown in Table 2 for all cases. The difference between Gradient and AMSGrad optimizer, which allows AMSGrad to be robust to the signal level, lies in the adaptive learning rate tuned by calculating first and second moment. It can be seen in Fig. 2 that application of AMSGrad not only improves the convergence of parameters but also it reduces the overshoot.

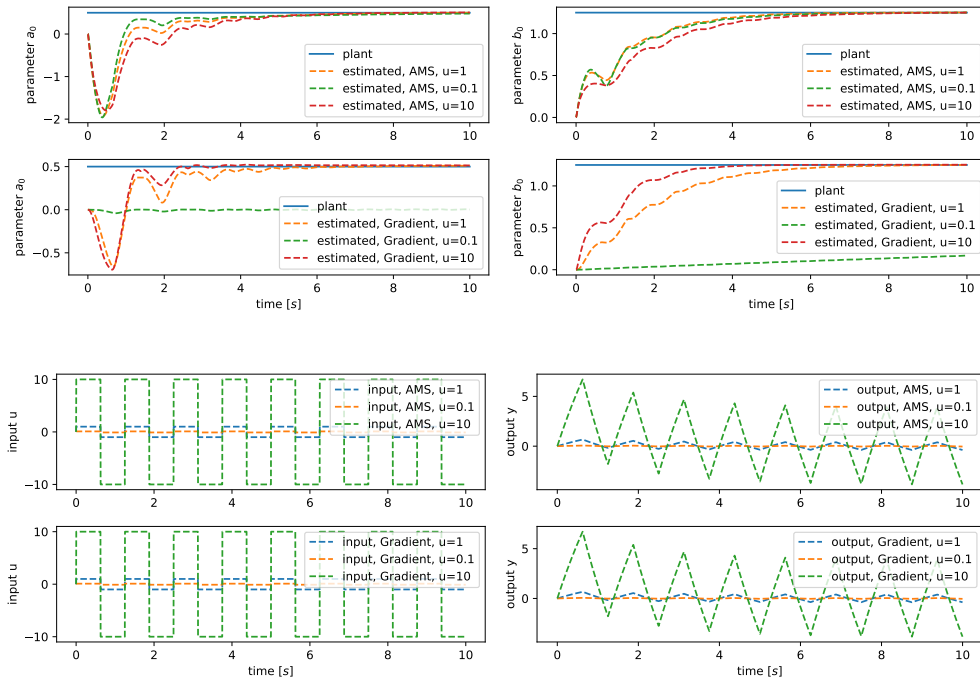


Figure 2: The influence of amplitude level of input signal on identification for different algorithms

Table 2: The performance indexes of parameter error for AMSGrad and Gradient algorithm for various input signal amplitude

Name	Amplitude	$J_{a_0}$	$J_{b_0}$	$J_{\text{sum}}$
AMSGrad	10	$4.05e + 03$	$1.28e + 03$	$5.33e + 03$
AMSGrad	1	$3.45e + 03$	$8.95e + 02$	$4.35e + 03$
AMSGrad	0.1	$2.96e + 03$	$9.26e + 02$	$3.89e + 03$
Gradient	10	$8.67e + 02$	$7.97e + 02$	$1.66e + 03$
Gradient	1	$8.41e + 02$	$1.64e + 03$	$2.48e + 03$
Gradient	0.1	$2.53e + 03$	$1.36e + 04$	$1.61e + 04$

### 3.3. Signal normalization

In the following example, the influence of signal normalization is shown by the analysis of the transfer function given by:

$$G_2(s) = \frac{b_1 s + b_0}{s^2 + a_1 s + a_0}, \quad (22)$$

where  $b_1 = 7.5$ ,  $b_0 = -12.5$ ,  $a_1 = 15$  and  $a_0 = 100$ . The transfer function is transformed to regressor representation (8) with first order  $\Lambda(s) = (s + 50)^2$ . The sampling time was set to 0.1 ms and input was square waved signal with amplitude 1 and period 5 s. Two cases were run, the first called ‘one’ with  $\sigma_y = \sigma_k = 1$  and the second called ‘adjusted’ with  $\sigma_y = 0.25$  and  $\sigma_1 = 2 \times 10^{-3}$ ,  $\sigma_2 = 4 \times 10^{-4}$ ,  $\sigma_3 = 6 \times 10^{-4}$ ,  $\sigma_4 = 1 \times 10^{-4}$ . In both cases the AMSGrad algorithm is used, however, in the case ‘one’ gain is set to  $\alpha = 2.5 \times 10^{-3}$  and in the case ‘adjusted’ gain is set to  $\alpha = 1 \times 10^{-5}$ . The transients of the estimated parameters are visible in Figure 3 for both cases. It is visible that even though the gain was reduced about two orders of magnitude, the performance of ‘adjusted’ case is better. It is visible that especially for parameter  $a_0$  the normalization improves the transients. Furthermore, the performance index is also better for the case with normalization as it is shown in Table 3. The improvement of performance index is about 57.68%.

Table 3: The performance indexes of parameter error with and without normalization

Performance index	one	adjusted
$J_{a_0}$	$1.85e + 09$	$3.15e + 08$
$J_{a_1}$	$2.08e + 07$	$7.78e + 06$
$J_{b_0}$	$1.38e + 07$	$5.30e + 06$
$J_{b_1}$	$3.41e + 06$	$4.81e + 06$
$J_{\text{sum}}$	$1.89e + 09$	$3.33e + 08$

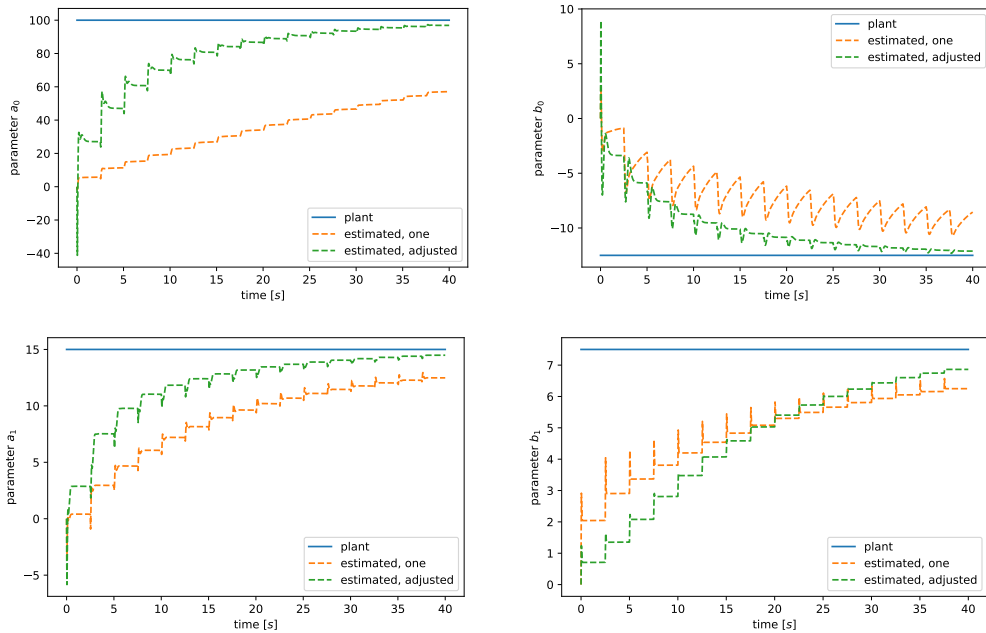


Figure 3: The influence of signal normalization on identification process

### 3.4. External bias

In this case, the influence of the external disturbance causing bias is shown. Let us consider the transfer function defined in the first example (20) and the filter  $\Lambda(s)$  is also the same. The above system is simulated with a square wave input (amplitude 1 and period 1.25 s). In the simulation the AMSGrad algorithm is used with gain  $\alpha = 0.5 \times 10^{-3}$  and sampling time is 1 ms. We assume that the output is biased with undesirable disturbances:

$$y_m(t) = y(t) + 0.004166 \times t \quad (23)$$

which has an influence on the system after a long time (for instance, for  $t = 60$  disturbance has value 0.25). We consider two cases: the first, in which the algorithm is not modified and the second, in which the regressor has an additional term:

$$y(t_k) = [\varphi^T(t_k) \ 1] \begin{bmatrix} \hat{\theta}_\lambda \\ \hat{\theta}_b \end{bmatrix}, \quad (24)$$

where  $\hat{\theta}_b$  is current bias estimation. In the regressor there is a term 1, but not  $t$ . It is due to the assumption that the bias is so slow that can be temporarily treated as constant. The advantage of the proposed modification is visible in Figure 4. It is visible that the extended regressor is capable to estimate the parameters

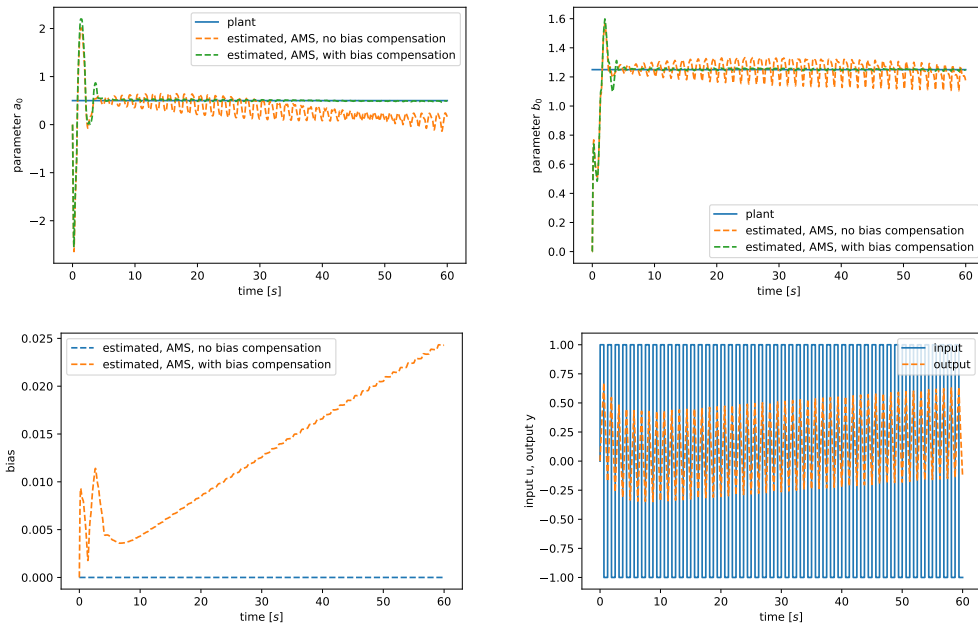


Figure 4: The example of biased system with and without correction

with lower error. This is also confirmed with the performance indexes shown in Table 4.

Table 4: The performance indexes of parameter error for AMSGrad with and without bias

Bias compensation	$J_{a_0}$	$J_{b_0}$	$J_{sum}$
with	$5.45e + 03$	$6.35e + 02$	$6.08e + 03$
without	$8.91e + 03$	$7.56e + 02$	$9.66e + 03$

## 4. Identification with experimental data

### 4.1. Laboratory kit and experiments

Dielectric Electroactive Polymers (DEAP) offer excellent performance, are light and flexible, therefore have many potential applications as dielectric actuators. These are electromechanical devices, and therefore exhibit an electromechanical coupling, transforming electrical energy into mechanical energy [2]. The optimization algorithms presented in this work were used in the process of identifying the parameters of the dielectric actuator made of 3M VHB tape which

dimensions are listed in Table 5. The actuator membrane is placed horizontally, and a mass is fixed at its center, thus creating a biasing force that causes the membrane to deform. A precise description of the phenomena occurring in the actuator and its model have been described by the authors in previous works [4–6, 14]. The laboratory kit used to carry out the experiments is shown in Figure 5. The distance was measured with a laser sensor (Micro-Epsilon optoNCDT ILD1320-10), and the voltage was applied by a high-voltage amplifier (TREK MODEL 10/10B-HS). The signals were processed by a data acquisition card (Inteco RT-DAC/US) connected to a computer. The research on the structure of the DEA model was carried out in the paper [4].

Table 5: Dimensions of dielectric electroactive polymer actuator

Parameter	Value	Unit
Pre-stretch tape thickness	1	mm
Post-stretch tape thickness	200	$\mu\text{m}$
Internal plate inner diameter	120	mm
Internal disc diameter	20	mm

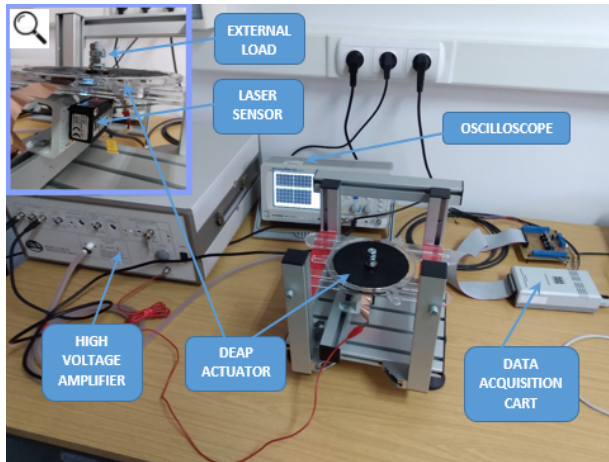


Figure 5: The laboratory set used in the identification process of DEAP actuator

In this work, the DEAP actuator is described by a series connection of the transfer function and a squared input  $u^2$  as it is presented in work [5]. It is an approximation of the general nonlinear model presented for instance in work [4]. The DEAP actuator transfer function is expressed as:

$$G_{DEAP}(s) = \frac{b_1s + b_0}{s^3 + a_2s + a_1s + a_0} = \frac{Y_p(s)}{U_p(s)}, \quad (25)$$

where  $Y_p(s)$  is the output  $y_p(t) = y_L(t) - y_R(t)$  and  $u_p(t) = u_A^2(t) - u_R^2(t)$ . The measured output of the laser is  $y_L(t)$  and the applied voltage by the power amplifier is  $u_A(t)$ . The signals  $y_R(t)$  and  $u_R(t)$  denotes the working point (equilibrium in steady state).

The transfer function parameters were identified by the least squares method for the estimation of the actuator displacement response to the given input voltage signal shown in Figure 6a. Figure 6b shows the comparison of the estimated value of the output with the value measured experimentally, while Figures 6c and 6d are the zoom versions for the selected time window. The identified parameters values are presented in Table 6. It is worth to point out that the DEAP transfer function has single stable zero, stable pole and stable complex poles. The imaginary part of the poles has a high value relative to the real part. This provides that the response is very low damped. It is also worth to point out that the dynamics has slow and fast parts. This causes that identification is a more challenging task.

Table 6: The parameters of transfer function of DEAP actuator

Parameter	$a_0$	$a_1$	$a_2$	$b_0$	$b_1$
Value	595.7	5891	3.292	7.722	51.47

In this section, the tests of online identification methods are performed on the data presented in Figure 6a, b. All algorithms (Gradient, Integral Cost, AMSGrad and Adam) are run with three configurations: without buffer, buffered with small gain, buffered with very small gain. In all cases the normalization is turned on with the following levels  $\sigma_y = 0.05$  and  $\sigma = [10^{-4} \ 10^{-4} \ 10^{-4} \ 10^{-6} \ 10^{-6} \ 10]$ . The last term denotes that bias is specified with coefficient 0.1 (not 1) and the other terms equalize the level of signals  $y$  and  $u$  transformed by filters  $\Lambda(s)$ . The size of buffer is 240000 elements. In the case of AMSGrad and Adam algorithm, the gain  $\alpha$  is equal to  $0.00005 \cdot g_n$  and in the case of Gradient and Integral Cost gain is  $0.05 \cdot g_n$ . It is worth to point out that the difference of gains between the algorithm is equalized by the sampling time which is applied in the Gradient and Integral Cost algorithm and not applied in AMSGrad and Adam algorithm. The transients of all estimated parameters are visible in Figure 8 and Figure 7. It is worth to point out that different dynamics of parameter estimation depends on the parameter. For instance, for all methods,  $b_1$  is quite slow while  $a_1$  is very fast. In the case of gradient algorithms, it has matter what signal is present in  $\varphi$  part. However, in general, the convergence of the gradient algorithm is the slowest. It is also visible that buffering has a crucial role on the transients and it significantly improves transients without making it more noisy.



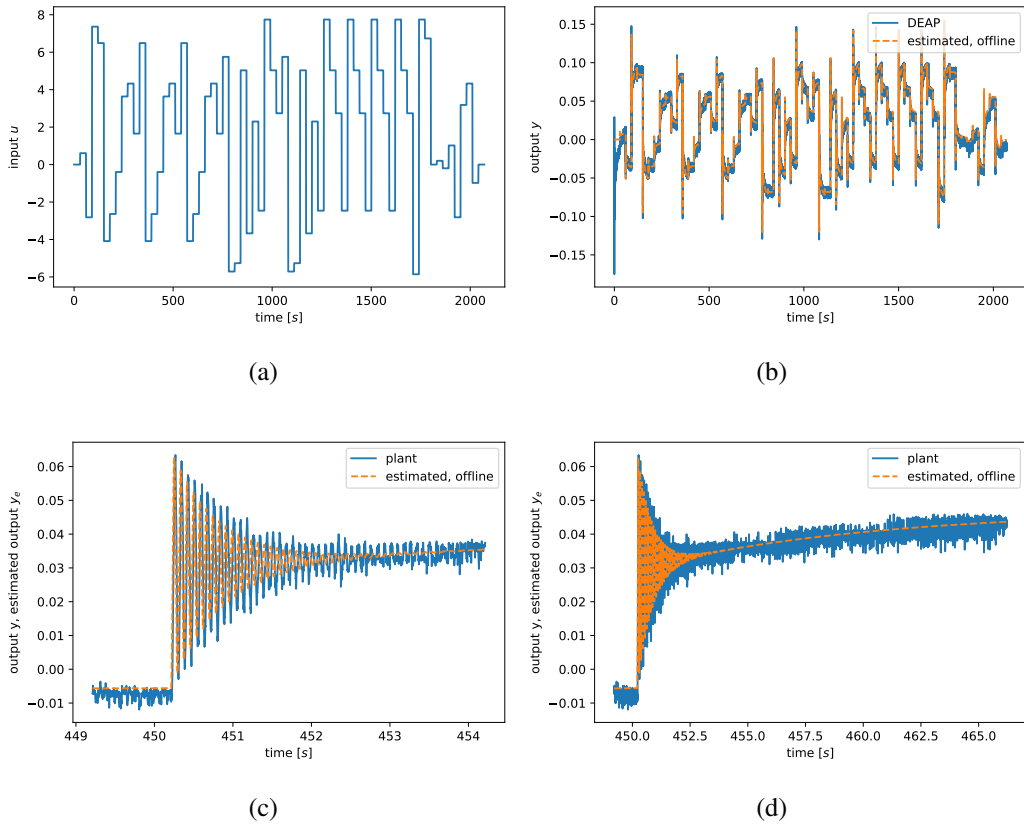


Figure 6: The input signal applied in DEAP identification (a). The compare of measured output with offline estimation with zooms (b, c, d)

## 5. Conclusion

The presented study describes the identification procedure for the system described by the continuous transfer function. The combination of standard methods known from system identification and machine learning algorithms allows to improve the convergence of estimated parameters. It was shown in the simulations that the application of replay buffer gives the possibility to reduce the performance indexes significantly. Furthermore, the algorithms based on the Stochastic Descent Gradient are less sensitive to the varying level of the signal amplitude. Finally, the proposed procedure works well for the experimental data obtained with Dielectric Electroactive Polymer actuator, which confirms that the presented method has a potential in practical applications. In the future works, the exploiting of the dynamics properties should be studied. For instance, the DEAP actuator has a slow and fast dynamics which could be identified separately.

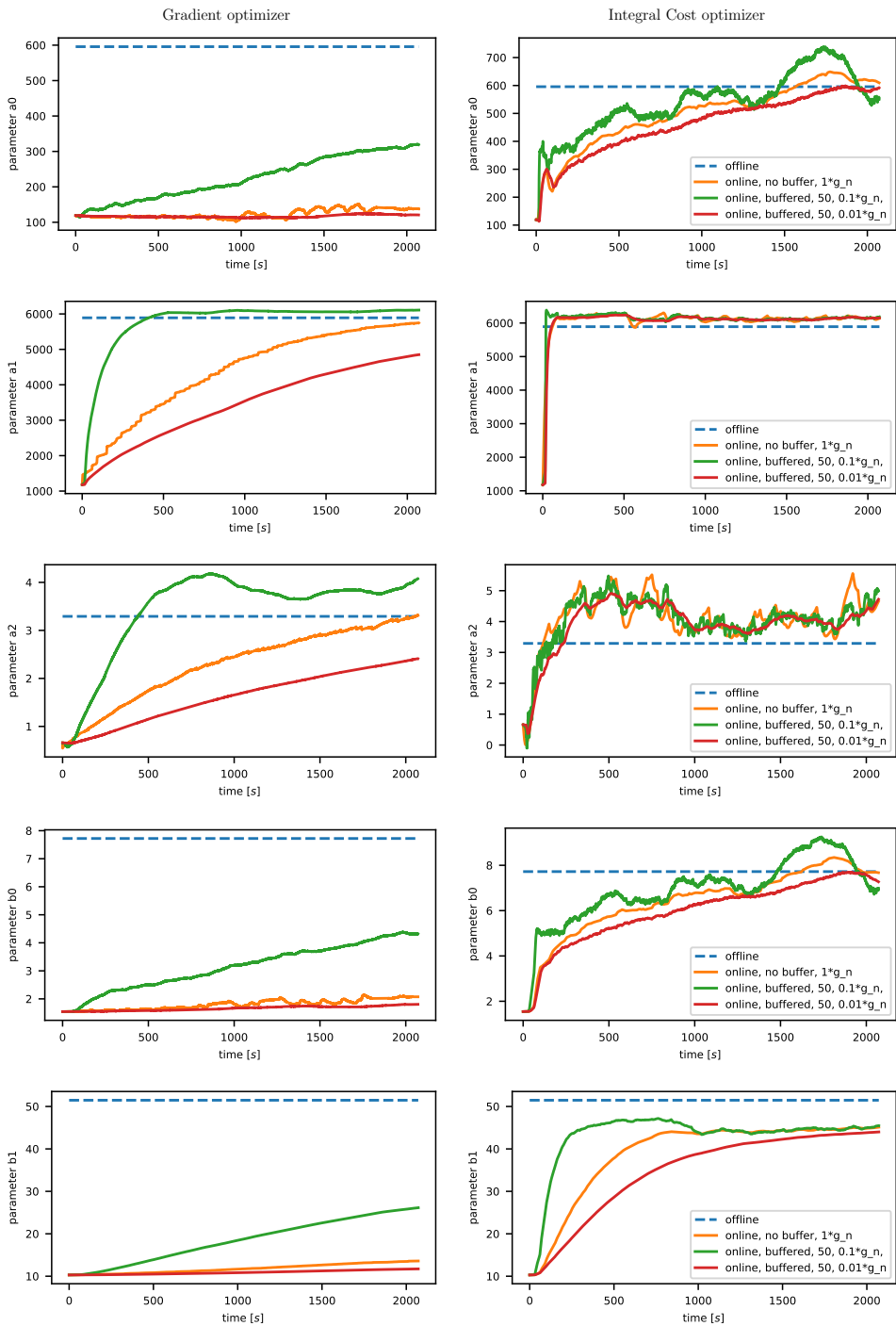


Figure 7: The transients of estimated parameters for DEAP experimental data for Gradient and Integral Cost optimizer

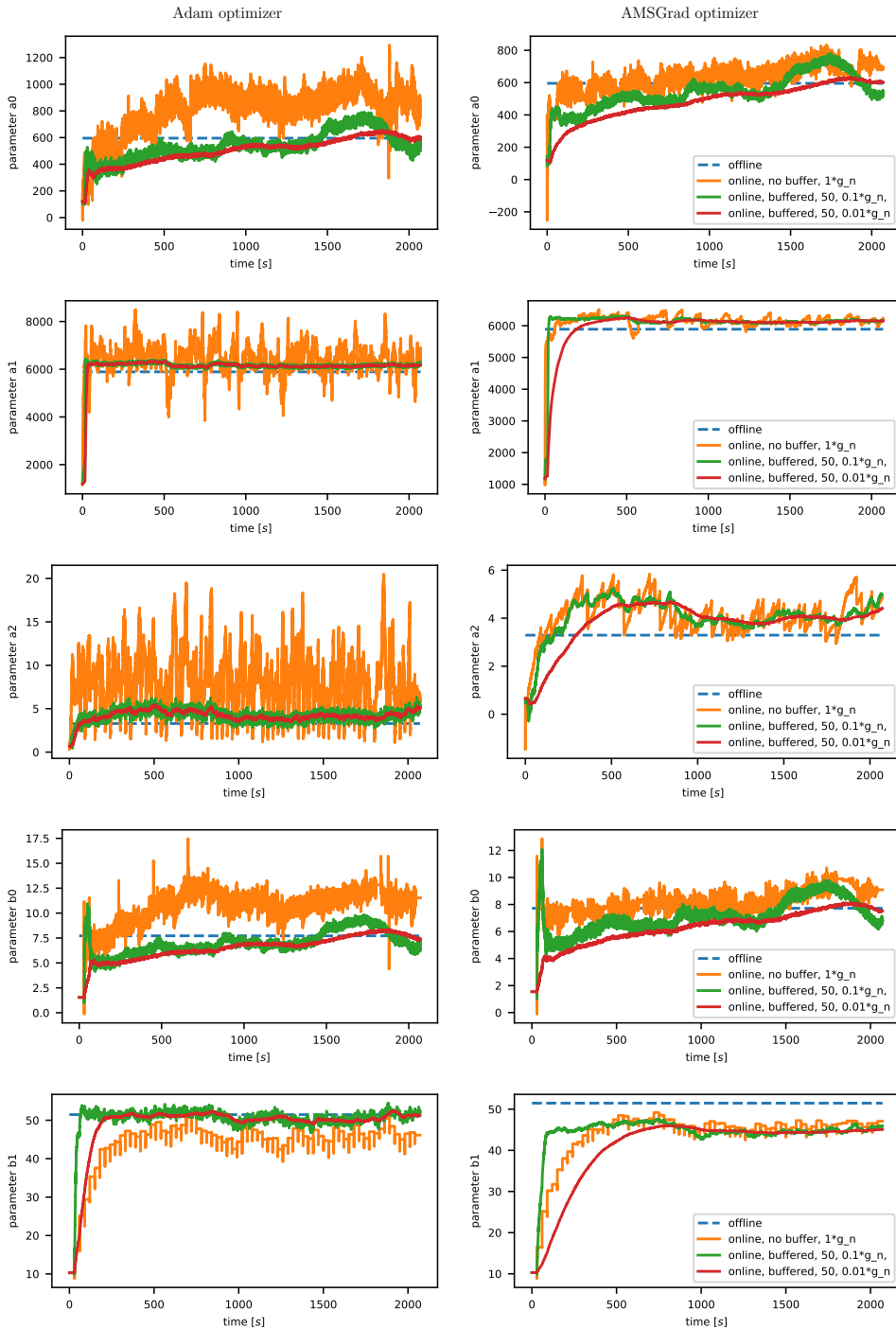


Figure 8: The transients of estimated parameters for DEAP experimental data for Adam and AMSGrad optimizer

## Funding

This work was supported by the statutory grant No. 0211/SBAD/0121.

### A. The variables in the continuous transfer function parametrization

$$\begin{aligned}\lambda &= [\lambda_{n-1} \ \dots \ \lambda_1 \ \lambda_0], \\ a &= [a_{n-1} \ \dots \ a_1 \ a_0], \\ b &= [b_m \ \dots \ b_1 \ b_0],\end{aligned}\tag{26}$$

$$\begin{aligned}\varphi_{y,i}(s) &= -\frac{s^i}{\Lambda(s)}Y(s), \\ \varphi_{u,i}(s) &= \frac{s^i}{\Lambda(s)}U(s),\end{aligned}\tag{27}$$

$$\begin{aligned}\theta_\lambda &= \theta_p + [-\lambda^T \ 0_{m+1}]^T, \\ \theta_p &= [a^T \ b^T]^T, \\ \varphi_{uy} &= [\varphi_y^T \ \varphi_u^T]^T, \\ \varphi_y &= \left[ -\frac{s^{n-1}}{\Lambda(s)}Y(s) \ \dots \ -\frac{s}{\Lambda(s)}Y(s) \ -\frac{1}{\Lambda(s)}Y(s) \right]^T, \\ \varphi_u &= \left[ \frac{s^m}{\Lambda(s)}U(s) \ \dots \ \frac{s}{\Lambda(s)}U(s) \ \frac{1}{\Lambda(s)}U(s) \ \dots \right]^T,\end{aligned}\tag{28}$$

$$\begin{aligned}F &= \begin{bmatrix} -\lambda_{n-1} & -\lambda_{n-2} & \dots & -\lambda_1 & -\lambda_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}, \\ l &= [1 \ 0 \ \dots \ 0]^T.\end{aligned}\tag{29}$$

## References

- [1] M. ANDRYCHOWICZ, F. WOLSKI, A. RAY, J. SCHNEIDER, R. FONG, P. WELINDER, B. MCGREW, J. TOBIN, P. ABBEEL and W. ZAREMBA: Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,

- S. Vishwanathan and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, **30** Curran Associates, Inc., 2017.
- [2] Y. BAR-COHEN: *Electroactive Polymer (EAP) Actuators as Artificial Muscles: Reality, Potential, and Challenges*. SPIE PRESS, Bellingham, Washington USA, 2nd. edition, 2004.
- [3] R. BAUMGARTNER, A. KOGLER, J.M. STADLBAUER, CH.CH. FOO, R. KALTSEIS, M. BAUMGARTNER, G. MAO, CH. KEPLINGER, S.J.A. KOH, N. ARNOLD, Z. SUO, M. KALTENBRUNNER and S. BAUER: A lesson from plants: High-speed soft robotic actuators. *Advanced Science*, **7**(5), (2020). DOI: [10.1002/advs.201903391](https://doi.org/10.1002/advs.201903391).
- [4] J. BERNAT, J. KOŁOTA and S. ROSSET: Identification of a nonlinear dielectric elastomer actuator based on the harmonic balance method. *IEEE/ASME Transactions on Mechatronics*, **26**(5), (2021). DOI: [10.1109/TMECH.2020.3044492](https://doi.org/10.1109/TMECH.2020.3044492).
- [5] J. BERNAT and J. KOŁOTA: A PI controller with a robust adaptive law for a dielectric electroactive polymer actuator. *Electronics*, **10**(11), (2021). DOI: [10.3390/electronics10111326](https://doi.org/10.3390/electronics10111326).
- [6] J. BERNAT and J. KOŁOTA: DEAP actuator composed of a soft pneumatic spring bias with pressure signal sensing. *Energies*, **14**(4), (2021). DOI: [10.3390/en14041189](https://doi.org/10.3390/en14041189).
- [7] F. CARPI, I. ANDERSON, S. BAUER, G. FREDIANI, G. GALLONE, M. GEI, C. GRAAF, C. JEAN-MISTRAL, W. KAAL, G. KOFOD, M. KOLLOSCH, R. KORNBLUH, B. LASSEN, M. MATYSEK, S. MICHEL, S. NOWAK, B. O'BRIEN, Q. PEI, R. PELRINE, B. RECHENBACH, S. ROSSET and H. SHEA: Standards for dielectric elastomer transducers. *Smart Materials and Structures*, **24**(10), (2015). DOI: [10.1088/0964-1726/24/10/105025](https://doi.org/10.1088/0964-1726/24/10/105025).
- [8] T. DOZAT: Incorporating Nesterov momentum into Adam. *International Conference on Learning Representations*, San Juan, Puerto Rico (2016). <https://api.semanticscholar.org/CorpusID:70293087>.
- [9] J.C. DUCHI, E. HAZAN and Y. SINGER: Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**(61), (2011). <https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf>.
- [10] P. IOANNOU: *Robust Adaptive Control*. Dover Publications, Inc., New York, 2003.

- [11] K.J. KIM and S. TADOKORO: *Electroactive Polymers for Robotic Applications: Artificial Muscles and Sensors*. Springer, London, 2007.
- [12] D.P. KINGMA and J. BA: Adam: A method for stochastic optimization. *Computing Research Repository (CoRR)*, abs/1412.6980, (2015). DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980).
- [13] D.P. KINGMA and J. BA: Adam: A method for stochastic optimization. *International Conference on Learning Representations*, San Diego, CA, USA (2015).
- [14] J. KOŁOTA: The FEM model of the pump made of dielectric electroactive polymer membrane. *Applied Sciences*, **10**(7), (2020). DOI: [10.3390/app10072283](https://doi.org/10.3390/app10072283).
- [15] T.P. LILICRAP, J.J. HUNT, A. PRITZEL, N. HEESS, T. EREZ, Y. TASSA, D. SILVER and D. WIERSTRA: Continuous control with deep reinforcement learning. *Computing Research Repository (CoRR)*, abs/1509.02971, (2015). DOI: [10.48550/arXiv.1509.02971](https://doi.org/10.48550/arXiv.1509.02971).
- [16] T.P. LILICRAP, J.J. HUNT, A. PRITZEL, N. HEESS, T. EREZ, Y. TASSA, D. SILVER, and D. WIERSTRA: Continuous control with deep reinforcement learning. (2016).
- [17] G. LIU and J. WANG: Dendrite net: A white-box module for classification, regression, and system identification. *IEEE Transactions on Cybernetics*, **52**(1), (2022), 13774–13787. DOI: [10.1109/TCYB.2021.3124328](https://doi.org/10.1109/TCYB.2021.3124328).
- [18] H.B. McMAHAN and M. STREETER: Adaptive bound optimization for on-line convex optimization. *Proceedings of the 23rd Annual Conference On Learning Theory*, Haifa, Israel, (2010), 244–256.
- [19] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A.A. RUSU, J. VENESS, M.G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A.K. FIDJELAND, G. OSTROVSKI, S. PETERSSEN, C. BEATTIE, A. SADIK, I. ANTONOGLU, H. KING, D. KUMARAN, D. WIERSTRA, S. LEGG, and D. HASSABIS: Human-level control through deep reinforcement learning. *Nature*, **518**(7540), (2015), 529–533. DOI: [10.1038/nature14236](https://doi.org/10.1038/nature14236).
- [20] S.J. REDDI, S. KALE and S. KUMAR: On the convergence of Adam and beyond. *International Conference on Learning Representations*, Vancouver, Canada, (2018).
- [21] S.J. REDDI, S. KALE and S. KUMAR: On the convergence of Adam and beyond. arXiv, (2019).

- [22] G. RIZZELLO, D. NASO, A. YORK and S. SEELECKE: Modeling, identification, and control of a dielectric electro-active polymer positioning system. *IEEE Transactions on Control Systems Technology*, **23**(2), (2015), 632–643. DOI: [10.1109/TCST.2014.2338356](https://doi.org/10.1109/TCST.2014.2338356).
- [23] S. ROSSET, O.A. ARAROMI, S. SCHLATTER and H.R. SHEA: Fabrication process of silicone-based dielectric elastomer actuators. *Journal of Visualized Experiments*, **108** (2016), 1–13. DOI: [10.3791/53423](https://doi.org/10.3791/53423).
- [24] T. SÖDERSTRÖM and P. STOICA: *System Identification*. Prentice-Hall, Inc., USA, 1988.
- [25] R.S. SUTTON and A.G. BARTO: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, London England, 2018.
- [26] T. TIELEMAN and G. HINTON: RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, **4**(2), (2012), 26–31.
- [27] J-W. YEH and S-F. SU: Efficient approach for RLS type learning in TSK neural fuzzy systems. *IEEE Transactions on Cybernetics*, **47**(9), (2017), 2343–2352. DOI: [10.1109/TCYB.2016.2638861](https://doi.org/10.1109/TCYB.2016.2638861).
- [28] M.D. ZEILER: ADADELTA: An adaptive learning rate method. *Computing Research Repository (CoRR)*, abs/1212.5701, (2012). DOI: [10.48550/arXiv.1212.5701](https://doi.org/10.48550/arXiv.1212.5701).
- [29] G. ZHANG, X. CHEN and T. CHEN: Digital redesign via the generalised bilinear transformation. *International Journal of Control*, **82**(4), (2009), 741–754. DOI: [10.1080/00207170802247728](https://doi.org/10.1080/00207170802247728).
- [30] S. ZHANG and R.S. SUTTON: A deeper look at experience replay. arXiv: 1712.01275, (2018). DOI: [10.48550/arXiv.1712.01275](https://doi.org/10.48550/arXiv.1712.01275).