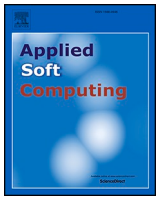
Applied Soft Computing Journal 167 (2024) 112236

Contents lists available at ScienceDirect

Applied Soft Computing

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



An experimental study on the application of reinforcement learning in injection molding in the spirit of Industry 4.0

Richárd Dominik Párizs [a], Dániel Török [a,b,*]

[a] Department of Polymer Engineering, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Műegyetem rkp. 3, Budapest H-1111, Hungary
[b] MTA-BME Lendület Lightweight Polymer Composites Research Group, Műegyetem rkp. 3, Budapest H-1111, Hungary

HIGHLIGHTS

- Reinforcement learning can be used to improve injection molding technology.
- Actor-critic algorithms are helpful in setting up an injection molding machine.
- The use of prior knowledge is necessary for self-learning machines.
- Injection molding simulations can be used as prior knowledge for self-learning.
- Our algorithm can compete with a professional through relevant prior knowledge.

ARTICLE INFO

Keywords:
Injection molding
Reinforcement learning
Actor-critic algorithm
Industry 4.0
Self-adjustment

ABSTRACT

The use of reinforcement learning in the injection molding process is a little-researched area in the era of Industry 4.0. The use of a smart decision-making algorithm is necessary for such a complex production method. Therefore, our research aims to extend the knowledge of the practical use of reinforcement learning in injection molding. In our study, we examined the effect of the parameters of the Actor-Critic algorithm to give a broader picture of the learning process. In addition, we show how to use simulation data, as prior knowledge, to set up the injection molding process for the production of an unknown part.

1. Introduction

Nowadays there are many articles on the subject of the fourth industrial revolution, also known as Industry 4.0 [1–3]. The idea behind the revolution is the reduction of human involvement in production processes by increasing the communication and connections between machines [4]. A way to achieve this is to teach the machines to learn and control themselves [5].

The plastics industry, as one of the most important manufacturing industry, also uses the concept of Industry 4.0. In some cases, the term refers to the collection and processing of data [6], or it relates to 3D printing because of the novelty of the new design and production method [7]. According to another perspective, 3D printing is part of Industry 4.0 due to the possibility of computer control [8]. Oleksy et al. [9] argue that improving productivity is an important aspect of Industry 4.0, therefore using integrated information systems is essential for plastics manufacturing companies. Yet another implementation of Industry 4.0 in polymer composite technologies is the application of robot-assisted processes [10]. Aminabadi et al. [11] consider the use of AI-controlled quality checks for injection molding a criterion of Industry 4.0.

Injection molding is a relatively complicated mass production process, in which several parameters can be adjusted, such as injection rate, holding pressure and mold temperature. [12,13]. Due to the complexity of the technology, mold design and the material, injection molding is still extensively researched [14,15]. In addition, injection molding can be used to produce parts of different sizes, which means customers can have quite different requirements. The quality of an injection molded

Abbreviations: ABS, Acrylonitrile butadiene styrene; CAD, Computer Aided Designing; ID, Injection molding dataset; SD, Simulation dataset.

* Corresponding author at: Department of Polymer Engineering, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Műegyetem rkp. 3, Budapest H-1111, Hungary.
E-mail address: torok@pt.bme.hu (D. Török).

https://doi.org/10.1016/j.asoc.2024.112236
Received 8 September 2023; Received in revised form 26 August 2024; Accepted 3 September 2024
Available online 14 September 2024

product can be defined with a number of parameters, such as surface roughness [16], geometrical accuracy [17], or the weight of the part [18].

It is not a coincidence that controlling injection molding and the mathematical modeling of the process are frequently researched topics. For example, Barghash and Alkaabneh [19] made regression models of part shrinkage and warpage by analyzing seven injection molding parameters. Kazmer and Westerdale [20] analyzed seven injection molding parameters with a factorial design, and from the analysis, they defined a model for quality control. Zhang et al. [21] made a model for controlling warpage based on principal component analysis. Mirigul Altan [22] used the Taguchi design, analysis of variance and a neural network to optimize the shrinkage of injection molded parts. It is important to mention injection molding simulation, as it is also used in many cases to optimize the process [23] or design the mold [24].

Besides modeling, researchers put a lot of effort into analyzing different control methods. Chen et al. [25] used strain gauges on the tie bars of the injection molding machine to control part weight. In their research, they varied the switchover point with a predefined step size and in addition to this, they used linear interpolation to fine-tune the quality of the product. Gordon et al. [26] suggested quality control using a multivariate in-mold sensor because the measurement data can predict the nonlinear behavior of tensile stress. To overcome the drawbacks of traditional PID controllers, Yang et al. [27] presented an improved PID controller that controls the injection speed more accurately. Tsai et al. [28] proposed an adaptive machine control strategy based on sensors in the injection molding machine, linear interpolation and neural network. Kumar et al. [29] presented a rule-based algorithm for avoiding failures during injection molding. In their study, Wang et al. [30] described the digital twin concept to control the injection molding technology. Su et al. [31] suggested a nozzle pressure and clamping force-based adaptive system to control the part weight for polymers with different melt flow properties. Naturally, the use of artificial neural networks on control processes is also researched, which can be used to intervene in the injection molding process immediately [32,33].

The term “artificial intelligence” is used relatively often and encompasses many methods, including machine learning [34]. Colut et al. [35] identified machine learning as one of the methods of Industry 4.0. Nian et al. [36] divided machine learning into four subgroups based on the learning process and data, namely: unsupervised learning, supervised learning, semi-supervised learning, and reinforcement learning.

In unsupervised learning, the input data is not labeled; therefore, the learning algorithms group the data based on the pattern of the input [37]. For injection molding, unsupervised learning can be useful for failure detection [38] or indexing the CAD model of molds in a database [39]. In supervised learning, the input data is labeled, so the learning algorithms try to define the relationship between the input and the label [40]. This method is also used for injection molding, thanks to the many types of learning algorithms it includes [41]. Supervised learning can be promising for product quality classification based on in-mold pressure measurement [42,43] or quality regression [44]. Labeling hundreds or thousands of data points could be exhausting and time-consuming. Therefore, in a mass production process such as injection molding, the use of semi-supervised learning can greatly increase efficiency, where the algorithm can learn from a mixture of labeled and unlabeled data [45].

In reinforcement learning, learning is based on the interaction between a decision maker (agent) and the environment, while the algorithm masters which actions or states bring it closer to the target state. During the process, the agent gets a reward from the environment and chooses an action that changes the state of the environment in order to maximize the reward. Because of this operation mechanism, reinforcement learning can be particularly suitable for regulation and control [46]. These methods are not as stable as traditional systems but work much better in new environments [47]. Conventional controllers, such as PID controllers, do not use predictions [48] and do not work well if there are sudden environmental changes [49]. One great disadvantage of reinforcement learning methods is that they require many training steps, but these algorithms can be trained offline [36].The number of studies on reinforcement learning has significantly increased in recent years [50]. However, it is interesting that this trend is not yet significant in the field of injection molding. On 13 June 2023, a search of the Web of Science for “injection molding” and “reinforcement learning” returned 10 articles. On the same day, the Scopus database returned 15 documents for the common intersection of the Title–Abstract–keyword of “injection molding” and “reinforcement learning”. After we filtered out the duplicates, there were 17 unique documents on the topic. Some of these articles, published in recent years, have explored exciting issues. Lee et al. [51] investigated the use of reinforcement learning for mold scheduling to satisfy the growing needs of customers. Li et al. [52] used Q-learning to control the injection molding process in case of actuator disturbance. Qin et al. [53] explored the usability of reinforcement learning to stabilize the injection molding process in the case of random process disturbances. In their study, Guo et al. [54] used the Actor-Critic algorithm to optimize ultra-high precision injection molded products based on simulation data and compared the results with a Fuzzy inference system and genetic algorithm.

Unlike the above-mentioned Actor-Critic algorithms, we propose the use of an Actor-Critic algorithm that uses state aggregation, temporal difference, and average reward. In addition, our algorithm uses a softmax policy, which makes it easier to discover even in new environments. As a result of these changes, our method is not limited by the accuracy of the model which describes the environment, as it is not required for the operation of our algorithm. Besides, we believe that injection molding should be tuned by optimizing each of the process phases separately, as they interact with each other. Therefore, in the present research, we show how to use our algorithm for optimizing the quality of a product during the holding phase.

The above-mentioned studies are essential to understanding the use of reinforcement learning in injection molding. However, these studies usually do not aim to explain how to set up the injection molding machine without prior knowledge. In addition, they usually only examine the effect of one or two algorithm parameters, which is insufficient for a reader to understand the application of reinforcement learning for injection molding. Therefore, in our research, we investigated the effect of several learning parameters for injection molding (state aggregation, number of possible actions, reward, etc.). Also, we examined the use of prior knowledge only from simulation data to set up the injection molding machine without preliminary experience of producing the actual part. In our research first, we describe the materials, methods, learning algorithm, and equipment used in our experiments. Here, we explain how the learning method and algorithm works, show the learning datasets and describe the process of analysis and evaluation. The results are then presented in two main parts. The first part consists of an analysis of the effect of algorithm parameters on the control. The second part describes the application of the algorithm to the injection molding technology, including the use and effect of prior knowledge.

2. Material and methods

2.1. Injection molding and measurement

To investigate the effects of learning parameters, we carried out injection molding experiments using an Arburg Allrounder 320 C 400–170 injection molding machine (Arburg GmbH+Co., Loßburg, Germany) and a two-cavity mold. The specimens were produced from acrylonitrile butadiene styrene (ABS), named Terluran GP-35 (INEOS Styrolution, Manchester, United Kingdom). The product was an 80 mm× 80 mm square tile-like part with a nominal height of 6 mm (Fig. 1). The mold was designed for teaching and research, and therefore the product has a complex geometry to illustrate a number of injection molding defects, such as weld lines and sink marks. After production, the weight of each

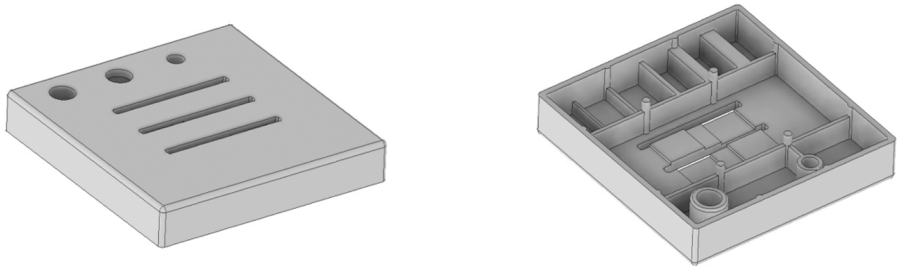


Fig. 1. The CAD model of the product: (a) top view (b) bottom view.

part was measured with an Ohaus Explorer analytical balance (OHAUS Europe GmbH, Uster, Switzerland). We used weight as a quality parameter for optimization and for the algorithm to learn. To investigate the use of prior knowledge, we made injection molding simulations of the same part with Moldex3D Studio 2022 (Moldex3D, Lecco, Italy) with the same setting combinations as for injection molding, and used the MATLAB R2021b platform to implement the Actor-Critic algorithm, carry out the learning process, and analyze the results. We used the Terluran GP-35 ABS material from the Moldex3D database for the simulation. The dimensions of the model used for simulation were the same as those used for the real injection molded model. The product has several features, such as a boss (with a thickness of 1 mm, outer diameter of 8.5 mm and a height of 6 mm), a small boss (with a thickness of 1 mm, 6 mm and a height of 4 mm), a hole (with a diameter of 6 mm), ribs (with thicknesses of 0.8, 1, 1.5, 3 and 4 mm, and heights of 2 and 4 mm), and wall thickness changes.

Our goal was to investigate if the Actor-Critic algorithm can be used to optimize the injection molding process. Injection molding is a cyclic technology with several steps and many process parameters which should be optimized. One of the most critical step is the holding phase, which has a great impact on the quality of the part (part weight, shrinkage, etc.). Therefore, we chose to optimize the holding phase. We first injection molded with several injection molding parameter combinations. The injection molding parameters used and not changed during production are shown in Table 1. The process parameters varied were holding time (varied between 0 s and 5 s in steps of 0.5 s) and holding pressure (varied between 0 bar and 1000 bar in steps of 100 bar). We injection molded one cycle with each setting combination except the setting combinations with 400, 700, and 1000 bar holding pressure, with which we produced six specimens each. We used these specimens to estimate the variance of part weight. Of course, the algorithm needs boundaries such as the maximum and minimum holding pressure and time, which can be set on the machine. We know these boundaries from the datasets in our experiments because we have no data outside the search space, but we should define them in other cases. The maximum holding pressure was 1000 bar, the maximum holding time was 5 s, and the minimum value for each was 0 bar and 0 s, respectively. If the algorithm chooses an action that would take it out of the boundaries, it does not change the holding pressure and holding time.

Table 1
The unchanged settings of the injection molding.

Process parameter	Value
Shot volume [cm^3]	30
Circumferential speed [m/min]	25
Back pressure [bar]	60
Decompression [cm^3]	3
Injection flow [cm^3/s]	30
Switch-over volume [cm^3]	9
Injection time [s]	1.04 +/- 0.13
Injection time limit [s]	3
Injection pressure [bar]	939 +/- 14
Injection pressure limit [bar]	1500
Clamping force [kN]	400
Cooling time [s]	10
Cycle time [s]	22,5
Melt temperature [°C]	220
Mold temperature [°C]	40

In our experiment, the Actor-Critic algorithm theoretically changes the injection molding parameters (holding pressure and holding time) during learning and observes the weight of the part produced using the dataset. We say "theoretically'' because we wanted to investigate the long-term performance of the algorithm, which, in our case, means thousands of injection molding cycles. This would require a lot of time, energy, and materials, which would be an unnecessary waste. Therefore, we used the results of the measurements and injection molding simulations and defined datasets where we simulated the learning process. For our experiments, we used two different datasets: one from injection molding and mass measurement, named injection molding dataset (ID), and another defined from the injection molding simulation, named simulation dataset (SD) (see Fig. 2). With these datasets, we interpolated the weight of the product for any holding pressure–holding time combination in the given range of measurement, therefore we were able to use smaller steps for the adjustment of settings during learning. With this method, we did not have to make thousands of injection moldings, and we were still able to use actual injection molding data (or data from injection molding simulation). Fig. 2 shows that injection molding and simulation give relatively different results due to the lack of optimization of the simulation. We did not optimize the simulation because, for a new product/mold, engineers work primarily from simulation software databases rather than measured material models. For this reason, our results show how useful such simple simulations can be for the learning algorithm.

2.2. Investigation of the effect of Actor-Critic parameters

2.2.1. The algorithm and its initial parameters

We aimed to use reinforcement learning to control the injection molding machine. It is clear from the literature that reinforcement learning can be used well in complex environments (like injection molding) and it can adapt to new environments. Another advantage is that it can also be taught with offline data. For these reasons, our method can adapt to adjusting the injection molding machine based on the (prior) knowledge obtained from simulation data.

Our first goal was to investigate the effect of the parameters of the algorithm on the learning process. For this, we used only the ID. We always used 0 bar holding pressure and 0 s holding time as a starting point of the learning process. This is the safest way to set the parameters if we do not have any information about the mold and the part. From the starting point, the algorithm choses an action according to the policy and changes the holding pressure and holding time. Then the algorithm picks the product weight from the ID due to the new setting combinations and updates its policy and value estimation. If there is no product weight for the new setting combination in the dataset, then the

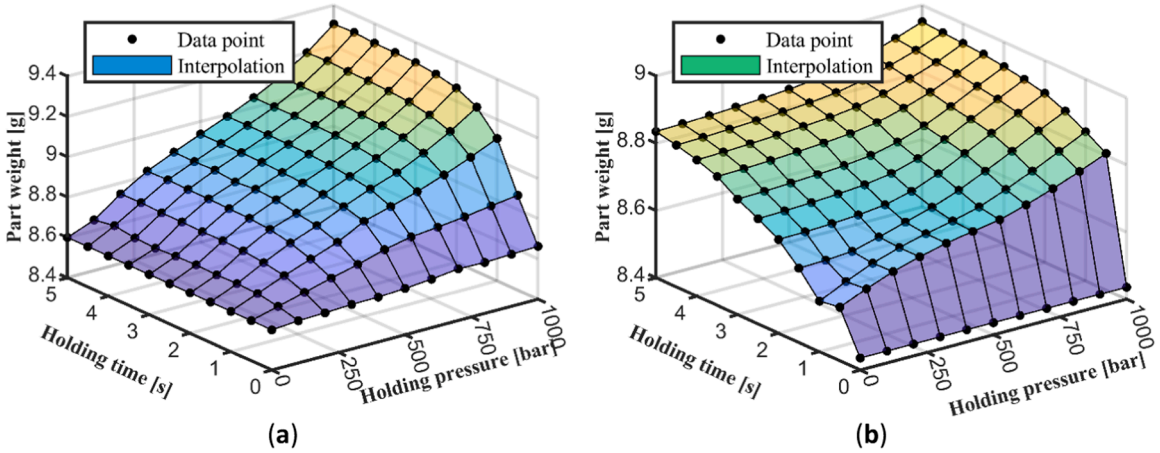


Fig. 2. (a) The injection molding dataset (ID) and (b) the simulation dataset (SD).

algorithm interpolates the weight of the product from the neighboring settings.

The pseudocode of our algorithm can be seen in Fig. 3. The algorithm uses two function estimations: one for softmax policy estimation ($\pi(a|s, \underline{\theta})$), and one for the value function estimation ($\hat{v}(s, \underline{w})$). These functions are estimated from the weight vectors ($\underline{\theta}$ and $\underline{w}$) and the feature vectors ($\underline{x_s}$ and $\underline{x_h}$) based on the actual state: s (the weight of the product) and the chosen action: a (i. e. how to change the technology). The weight vector of the value function stores a value for each aggregate state, which expresses the goodness of that state. The weight vector of the policy function contains a value for each aggregate state–action pair, which will determine the probability of choosing a given action in that state using the policy function. All values of the weight vectors ($\underline{\theta}$ and $\underline{w}$) are initially 0.5, i.e., all states and actions appear to be equally good.

For policy estimation, we used the softmax function. With this method, the algorithm is able to make greedy decisions after sufficient training but can improve this policy in new environments, adapting it to changes. The reward (R) is estimated from the difference between the target weight ($m_{goal} = 8.75g$) and the weight of the part produced. Our algorithm uses the average reward ($\bar{R}$), as this is a good measure of performance for continuous tasks or tasks with many steps. Since the algorithm learns through thousands of steps in our experiments, this metric is more useful than the discounted reward. We set the initial value of the average reward to 0, because it is the optimal value our algorithm wanted to achieve. The average reward is updated through learning because of the temporal difference error.

The algorithm needs a target state (s_{goal}) to get a reward (penalty) for deviating from it. In this case, the target state is a product with a weight of 8.75 g. However, for the algorithm to work, this state can be another indicator of product quality, as long as that indicator can be quantified in some way (for example, the shrinkage of the product).

1. Set the initial weights: $\underline{w}, \underline{\theta}$
2. Define the initial value function and policy function:
   $\hat{v}(s, \underline{w}) = \underline{x_s} \cdot \underline{w}$; $\pi(a|s, \underline{\theta}) = \exp\left(\underline{x_h}(s, a) \cdot \underline{\theta}\right) / \sum_{b \in \mathcal{A}} \exp(\underline{x_h}(s, b) \cdot \underline{\theta})$
3. Define the initial average reward function: $\bar{R} = 0$
4. Define the target value: s_{goal}
5. Define the learning rate values: $\alpha_R > 0, \alpha_w > 0, \alpha_\theta > 0$
6. Define the initial state based on injection molding settings: $s \in \mathcal{S}$
7. Decide whether the environment is deterministic or stochastic: σ
8. Define the end of learning $t_{end} = 4000$;
9. Repeat until $t = t_{end}$
10. $t \leftarrow t + 1$
11. $a \leftarrow \pi(\cdot|s, \underline{\theta})$
12. Observe s_{t+1} based on σ
13. Calculate R from s_{t+1} and s_{goal}
14. $\delta \leftarrow R - \bar{R} + \hat{v}(s_{t+1}, \underline{w}) - \hat{v}(s_t, \underline{w})$
15. $\bar{R} \leftarrow \bar{R} + \alpha_R \cdot \delta$
16. $w \leftarrow w + \alpha_w \cdot \delta \cdot \nabla \hat{v}(s, \underline{w})$
17. $\theta \leftarrow \theta + \alpha_\theta \cdot \delta \cdot \nabla \ln\left(\pi(a|s, \underline{\theta})\right)$
18. $s_t \leftarrow s_{t+1}$

Fig. 3. The pseudocode of our Actor-Critic algorithm.

The initial state is currently determined by the technological parameters. Apart from the machine settings, this state depends on the material (drying, homogeneity) and the precision of the machine. In our experiment, the initial settings are 0 bar holding pressure and 0 s holding time. We chose these initial settings because this is the safest combination of settings from a technological point of view, without any background knowledge. The initial state has a big impact on the outcome of learning, which is why an exploring start is often used in general. We did not use exploring start because we wanted the algorithm to learn to set up the machine as an expert would. We used the same initial state value for the learning scenarios to simplify the analysis.

In our experiment, we investigate the use of deterministic and stochastic environments. This can be adjusted with the parameter σ. If $\sigma = 0$, the environment is deterministic because there is no random noise on the output of injection molding. However, if $\sigma > 0$, the weight of the product will always get a random noise from a normal distribution with a mean of 0 and a standard deviation of σ.

In all cases, we defined the end of learning at the 4000th injection molding cycle (i. e. learning step) because, in most cases, this number of steps was enough for the algorithm to sufficiently converge to the target value. The algorithm could have been stopped earlier if it reached the target value, but we did not do so to allow the algorithm to discover and act as a continuous control. The pseudocode (Fig. 3) distinguishes three learning parameters (α_w, α_θ, α_R), but for simplicity, we used the same value for each of them in our experiment. These parameters determine the extent to which the temporal difference error should be accounted for when updating function weights ($\underline{\theta}$ and $\underline{w}$) and the average reward ($\bar{R}$).

In the learning process, one step corresponds to one injection molding cycle. After the initial setup, the algorithm chooses an action to perform, observes the quality of the product (i. e., the state) and then calculates the temporal difference error (δ) according to the derived reward. The algorithm then uses this error to update the weights of the functions associated with the former state and state–action pairs. The algorithm repeats this task until it reaches the 4000th step. Of course, this limit can be changed to suit the task, even as a continuous control task.

2.2.2. *Investigated parameters and the method of evaluation*

In our research, we analyzed how the input parameters affect the learning of the algorithm. We investigated the effect of several parameters (Table 2). The default values for each parameter are in bold.

To better understand the performance of the Actor-Critic algorithm

Table 2
The investigated algorithm parameters.

Algorithm parameter	Values/Methods
Magnitude of the reward	$R = -\lvert m_{actual} - m_{goal}\rvert$ **$R = -10 \bullet \lvert m_{actual} - m_{goal}\rvert$** $R = -100 \bullet \lvert m_{actual} - m_{goal}\rvert$
Way of state aggregation	**by-part weight** by-injection molding settings
Number of states	**10** 50 100
Magnitude of actions	**Holding pressure: +/−25 bar;Holding time: +/−0.25 s** Holding pressure: +/−50 bar Holding time: +/−0.5 s Holding pressure: +/−100 bar; Holding time: +/−1 s
Number of possible actions	**3 actions/settings** 5 actions/settings 7 actions/settings
Value of learning rate	α=0.01 α=0.05 **α=0.1**
Environment	**Deterministic** Stochastic

in different cases, we performed 100 learning scenarios for the chosen combinations of algorithm parameters. We illustrated the performance of the algorithm by the change in the difference between actual weight and the target weight (1) in each case, as this shows when the algorithm converges. However, plotting 100 scenarios is not easy because the data are already overlapping in the case of 2–3 scenarios (Fig. 4a). In addition, the choices made during the learning process can vary widely, which means great variance in the results. Therefore, we plotted the median value of difference from the target value on each learning step and the interquartile range from the 100 learning scenarios (Fig. 4b). This simplifies the representations of learning and makes it possible to show learning with different initial parameters simultaneously.

$$E^{-} = -\left|m_{actual} - m_{target}\right|, \tag{1}$$

where E^{-} is the negative error, m_{actual} is the weight of the product injection molded in the last cycle and m_{target} is the target weight.

2.3. Investigation of the applicability of the Actor-Critic algorithm

2.3.1. Method of applying prior knowledge

The use of a priori knowledge can speed up learning (and thus setting up the injection molding machine), which makes it easier to apply the algorithm in real-world situations. In our algorithm, prior knowledge means that the initial weight vectors ($\underline{\theta}$ and $\underline{w}$) are determined from previous learning scenarios (the initial parameters of the algorithms should be the same except $\underline{\theta}$ and $\underline{w}$) instead of using the initial weights, which were 0.5 for each element of the weight vectors (see 2.2.1). The new initial weights (prior knowledge) were the mean weight values after the 4000th learning step from the 100 different learning scenarios.

In the investigation of the use of prior knowledge, we used the ID. First, we tested whether prior knowledge can actually accelerate the learning process. To do this, we used data from injection molding only to generate prior knowledge (pre-learning) and subsequent application (post-learning). Pre-learning, in this case, means the form of learning described earlier, i.e., 100 different learning scenarios with over 4000 steps with given initial parameters. Post-learning is very similar, with the difference that the initial weight vectors ($\underline{\theta}$ and $\underline{w}$) are the values determined from pre-learning.

2.3.2. Using data from injection molding simulations

After we examined the application of prior knowledge, we investigated the use of prior knowledge from injection molding simulation. Since we run the simulations with the same setup combinations as actual injection molding, it can be assumed that learning from the simulation data will speed up setting up the machine when it is used in a real injection molding environment. Therefore, we used the search SD to generate the prior knowledge during pre-learning, and then used the ID for post-learning. In addition, we made learning scenarios where the starting point of post-learning was not the usual starting point (0 bar holding pressure and 0 s holding time) but 300 bar holding pressure and 3 s holding time. This setting combination produced product mass closest to the target product mass from the simulation results.

3. Results

3.1. The effect of algorithm parameters

In our first experiment, we investigated the effect of the key algorithm parameters, such as the magnitude of the reward, the way of state aggregations, the possible actions, etc. To do this, we simulated learning scenarios with the algorithm with different initial parameters and examined how the error (1) changes.

3.1.1. The magnitude of the reward

During learning, the algorithm compares the output of the actual state and the goal. In our case, the goal is to produce a part with a weight of 8.75 g. Therefore, a natural reward function might be the following (2):

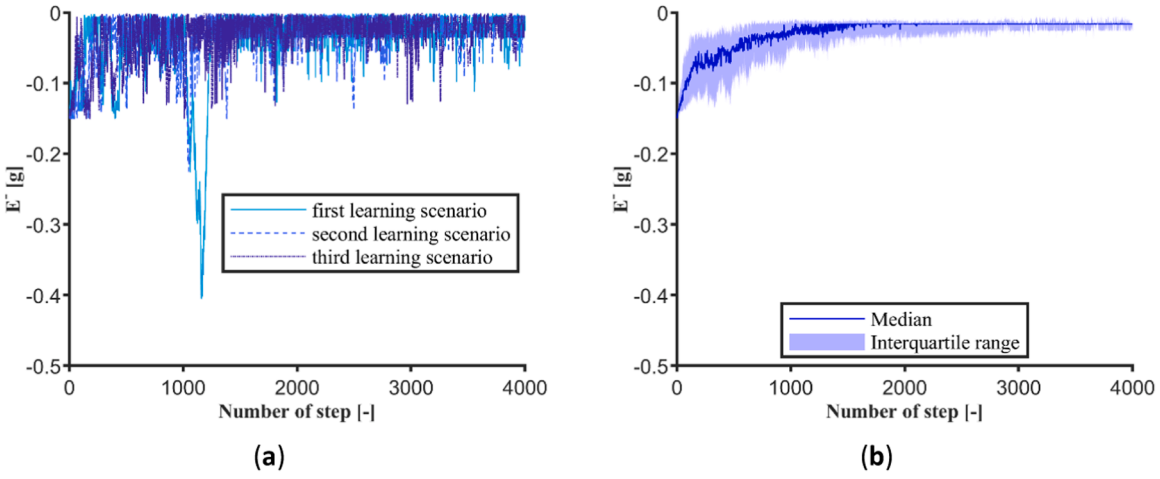


Fig. 4. (a) Three different learning scenarios with the same parameters. (b) 100 different learning scenarios with the same parameters shown with median and interquartile range.

$$R = -\left|m_{actual} - m_{goal}\right|, \tag{2}$$

where R is the reward, m_{actual} is the weight of the part produced with the actual setting combination, $m_{goal} = 8.75g$ is the goal of learning. However, the magnitude of the reward highly affects the temporal difference error, as does the weight vector of the policy function and value function. Therefore, we made learning scenarios with three different reward functions: with the absolute difference, ten times the absolute difference, and a hundred times the absolute difference. Fig. 5 shows the negative error from the target value during the 4000 learning steps. Each curve shows the results of 100 learning scenarios from the same starting point and with the same initial algorithm parameters except the highlighted parameter. The brightly colored line shows the median value, and the pale-colored area (with similar color) shows the interquartile range of the values. The results clearly indicate that rewards with greater magnitude have a significant influence on learning. Learning with (1) will converge too slowly to a local optimum which is clear from the great variance of the values. The reason behind this phenomenon is that the value of the reward is so small that it changes the weights (θ and $\underline{w}$) to a small extent. In practice, for a larger product, the weight of the part and its variation can be significantly more than for the actual task. Therefore, it is particularly important to look at the extent to which the reward, which seems to come naturally, helps the learning of the algorithm.

3.1.2. The effect of learning rate

The learning of the algorithm, therefore, depends on the size of the reward. However, the control of weights is not only through the reward, but also through another parameter, the learning rate. During learning, the Actor-Critic algorithm calculates the temporal difference error (δ) from the reward and the value function. After that step, the algorithm adjusts the average reward, value function, and policy function parameters with the temporal difference error weighted by the learning rate. Usually, the learning rates for these function parameters and average rewards are different. However, in our study, we use the same learning rate for each task. We made learning scenarios with three different learning rate values ($\alpha \in \{0.01,\ 0.05,\ 0.1\}$) as well. If we use too high a learning rate, we may change the parameters too much, and we will not find the optimum. The other extremum is when the learning rate values are too small, and it takes too many steps for the learning process to converge. This latter phenomenon can be seen in Fig. 6. The effect of learning rate and reward are similar. However, the learning rate, as opposed to the reward, is not information about the product in this case. For this reason, learning rates of the same value for products of different sizes may have similar effects, while the effect of reward could vary.

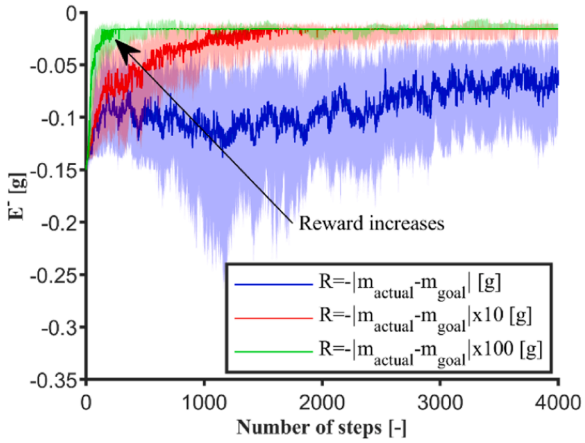


Fig. 5. The effect of the magnitude of the reward on learning.

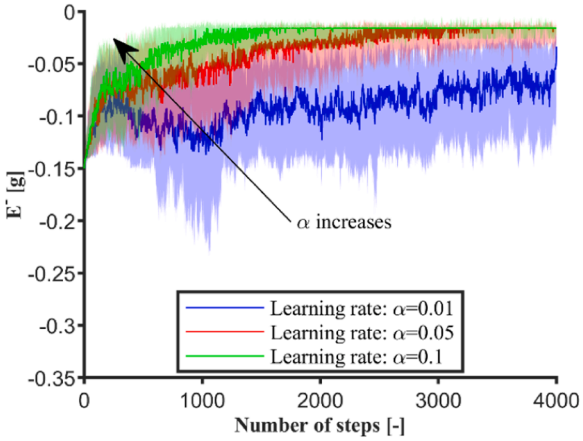


Fig. 6. The effect of the learning rate on learning.

3.1.3. The type of state aggregation

In our experiment, the Actor-Critic algorithm uses state aggregation to generalize between states that provide similar rewards or describe similar behavior. However, with state aggregation, the algorithm discriminates these states from all others and handles them differently. In Fig. 7, we showed two ways of state aggregation in our dataset. We can distinguish states based on the output of learning, in this case, the weight of the injection molded part. This can be seen in Fig. 7a, where each color represents a state with almost identical product weight, for 10 states. Another way is to distinguish states based on the injection molding parameters (Fig. 7b). With this method, the parts of the search space are considered a group with nearly identical holding pressures and holding times. In our case, we considered settings almost identical, which are within +25/-24.99 bar and +0.25/-0.2499 s of the highlighted settings. For example, the setting combination of 175 bar, 3.25 s belongs to the 150 bar, 3 s setting. Of course, this means way more states because we have 21×11 setting combinations. Therefore, we made a state aggregation with 231 states based on product weight.

The results of the two learning scenarios are very similar (Fig. 8). Each algorithm improves at the beginning, but later the variance of the collected reward increases. With state aggregation performed by settings, the median of the reward had a stable variance, as is the interquartile range. In the other case, the magnitude of variance is far more stochastic. The high variance in both cases is probably caused by too many states, which can make learning difficult. State aggregation by product weight results in a slight improvement after 2000 steps. This could mean learning from reward-dependent states is more manageable than from setting-dependent states. We think this is because the algorithm can handle state aggregation by product weight more easily than state aggregation by settings. The algorithm tries to find the relationship between the coordinates in the search space and the goodness of the product. State aggregation by product weight is, however, a natural indicator of the goodness of the product, and therefore, the algorithm finds the relationship somewhat more easily.

3.1.4. The number of states

We also investigated how the number of states with product weight–dependent state aggregation affects learning. In Fig. 9 we showed how differs the states if the number of discrete states increased from 10 to 50.

The results (Fig. 10) clearly indicate that learning is harder if the space is divided into several states. More states mean more places to discover, and with the increasing number of states, the number of function parameters in the weight vectors (w and θ) also increases. Therefore, if the number of initial function parameters is great (such as w and θ), exploration is all the more forced. This phenomenon could mean a slower convergence to the optimal values. From an injection molding

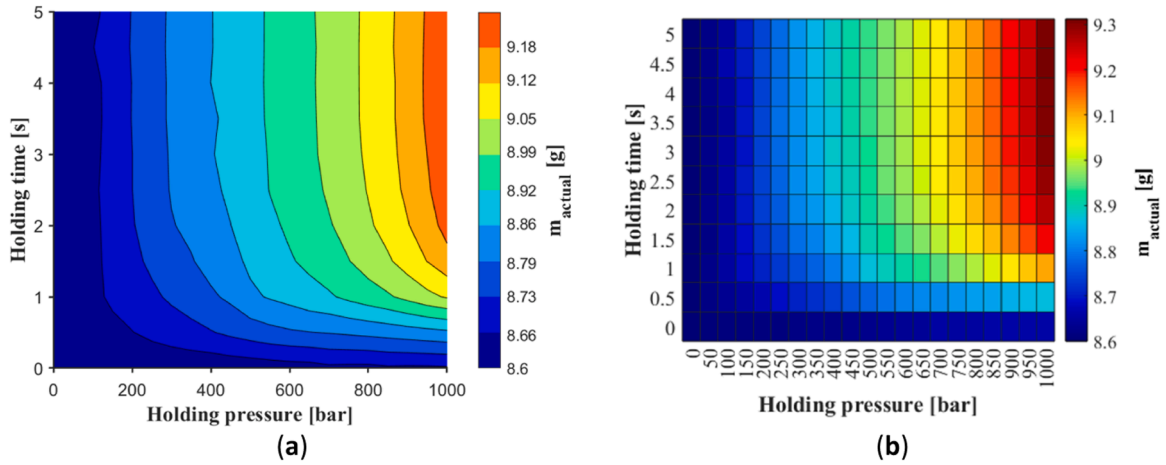


Fig. 7. State aggregation (a) by product weight with ten states (b) by injection molding settings.

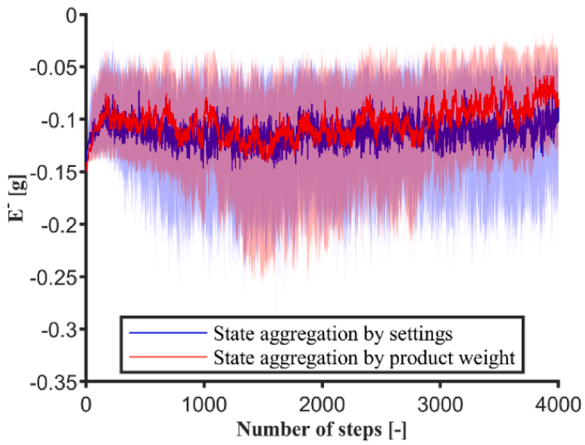


Fig. 8. The effect of the type of state aggregation.

perspective, this means that we need to look at many more cycles if the space is broken down into many parts. Of course, in industry, the number of states should be chosen according to the size of the product and its tolerance, but the aim should be to use as few states as possible without oversimplifying the search space.

3.1.5. *The magnitude of actions*

When an injection molding parameters are optimized, there is always the question of how much to change them. Therefore, knowing or at least suspecting how far the settings are from the optimal combination is essential. If we are far from the target, we need to use greater changes in the settings but close to the target, we should use smaller changes. It is also important to investigate the magnitude of actions with which the

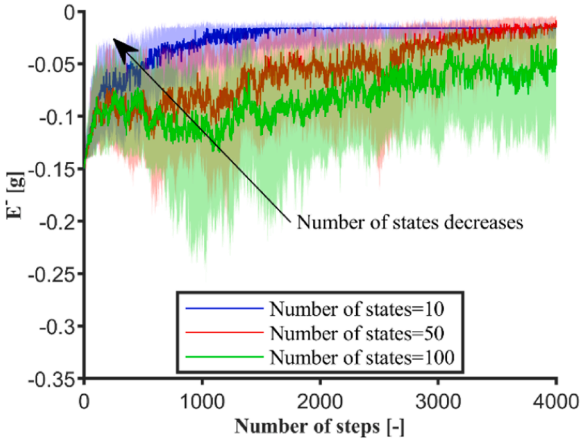


Fig. 10. The effect of the number of states.

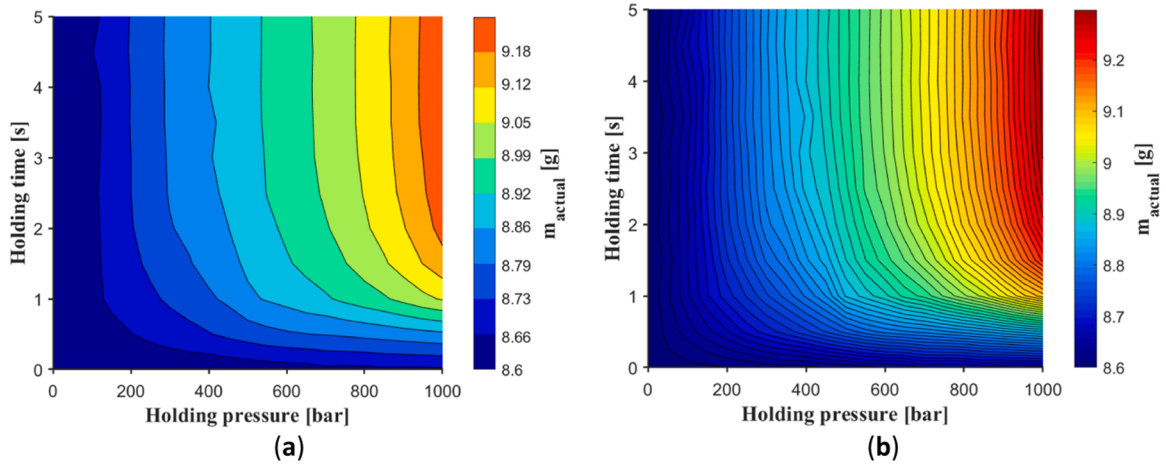


Fig. 9. State aggregation (a) with 10 states (b) with 50 states.

algorithm can find the target. We made learning scenarios where the algorithm had different magnitudes of action for each parameter. The actions are possibilities to change the state, i. e. the weight of the product. In our case, the algorithm needs two input arrays, which contain the possible changes of holding pressure and holding time. The final action will be selected from all combinations of each vector element due to the policy function. We used three action sets, and each set includes one holding pressure and holding time array, which will not change the setting on the injection molding machine (e. g. 0 bar and 0 s). The difference between the three sets (small, medium, and large action sizes) was the settings that changed the machine settings: the small action set had a holding pressure change of +/-25 bar and holding time change of +/-0.25 s, the medium action size set contained +/-50 bar and +/-0.5 s, and the large action size set had +/- 100 bar and +/-1 s holding pressure and holding time change, respectively (Fig. 11a).

Both the first and second sets reached an almost 0 reward, which means they found settings with which the desired product quality can be achieved (Fig. 11b). However, when the magnitude of the actions was 100 bar and 1 s, the algorithm could not reach this optimum. This phenomenon can be imagined as the algorithm jumping around the target value but not reach it.

3.1.6. The effect of the number of possible actions

We saw that when the magnitude of action is great, it can happen that the algorithm does not find the local optimum or just in certain scenarios. Another possibility is that more possible actions are used, which differ in magnitude. With this, the algorithm can learn that if it is far from the target, it should use greater steps than near the target. Therefore, in our study, we used three action sets with a different number of possible actions. The smallest action set (Fig. 12 a) was the small action size set (see 3.1.5.) that we used for most of our previous investigation. It has three possible steps for holding pressure (-25/0/+25 bar) and holding time (-0.25/0/+0.25 s). The second set (Fig. 12 b) contained two additional steps for each setting (+/-50 bar and +/0.5 s for holding pressure and time, respectively). The biggest action set (Fig. 12c) contained two more steps per setting (+/-100 bar and +/-1 s).

The results of these learning scenarios seem a little controversial (Fig. 13). One can expect that the algorithm would converge faster with a greater range of possible actions because it can choose actions with greater steps far from the optimum and smaller steps near the optimum. However, as well as for multiple states, the number of parameters increases for multiple actions. This means more exploration due to the starting values and many function parameters. The number of possible actions increases with the settings to adjust, which is a great challenge for a technology with as many input parameters as injection molding.

3.1.7. The effect of a stochastic environment

In the previous learning scenarios, we used a deterministic environment. Therefore, the output of injection molding (the weight of the product) was always the same for a given setting combination. However, this is not true for injection molding, due to the small inaccuracies of the various parts of the machine. We wanted to take into account this phenomenon, so we made six samples with three different holding pressures (400 bar, 700 bar, and 1000 bar) with all holding times. From these datasets, we calculated the standard deviation for each setting combination. The mean and median of the standard deviations were $\overline{\sigma} = 0.0053g$ and $\widetilde{\sigma} = 0.0048g$ respectively. Hence, we added a normally distributed random noise to each output result with a standard deviation of 0.005 g. With this change, we created a stochastic environment for learning. Fig. 14 a shows the results of learning from stochastic and deterministic environments. The convergence of both scenarios is similar, but the use of a stochastic environment shows a clear disadvantage. The variance of the median reward would not decrease to a fixed value even if there were a setting combination for injection molding (Fig. 14 b) which produces the part with the preferred weight. Due to this, the end of learning is somewhat more difficult to determine. However, during injection molding, the weight of the product will vary with the same settings due to the noise in the process and environment. Therefore, a stochastic environment should be used to simulate the real injection molding process better.

3.2. The application of reinforcement learning for injection molding

In our second experiment, we investigated how to use the Actor-Critic algorithm with prior knowledge for injection molding. The former experiments showed the effect of the algorithm parameters, but in each result, we saw that at least 250 steps were needed to achieve the desired quality. Therefore, in real-life applications, the algorithm would need at least 250 injection molding cycles at best to achieve the desired quality (8.75 +/-0.03 g). In practice, this is too many injection molding cycles, wasting a lot of material.

In industrial environments, injection molding machines are operated by technicians and/or engineers who may not have much information about a new mold/product but usually have a lot of experience with the technology. In other words, they have a lot of prior knowledge. In the previous experiments, the Actor-Critic algorithm had no specific prior knowledge/experience, as we used the same values for the weight of the value function and policy function of each state at the start of learning. The same is true for the policy function parameters. The value function estimates the goodness of a state, meaning how good it is for the algorithm to be in that state as a function of the objective. The policy function estimates the goodness of each state–action pair from which the

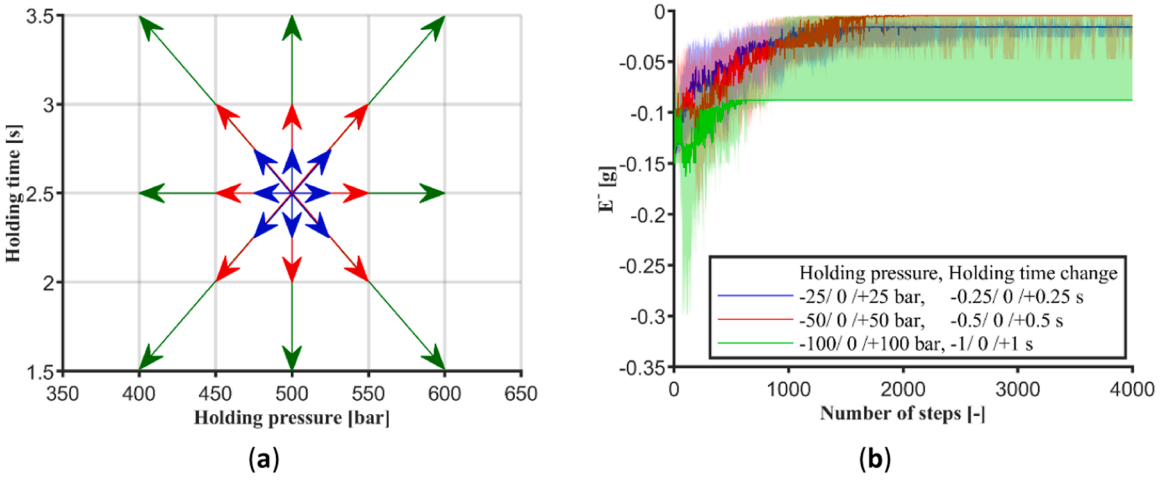


Fig. 11. (a) The possible steps with different magnitudes of actions in the search space. (b) the effect of the magnitude of actions.

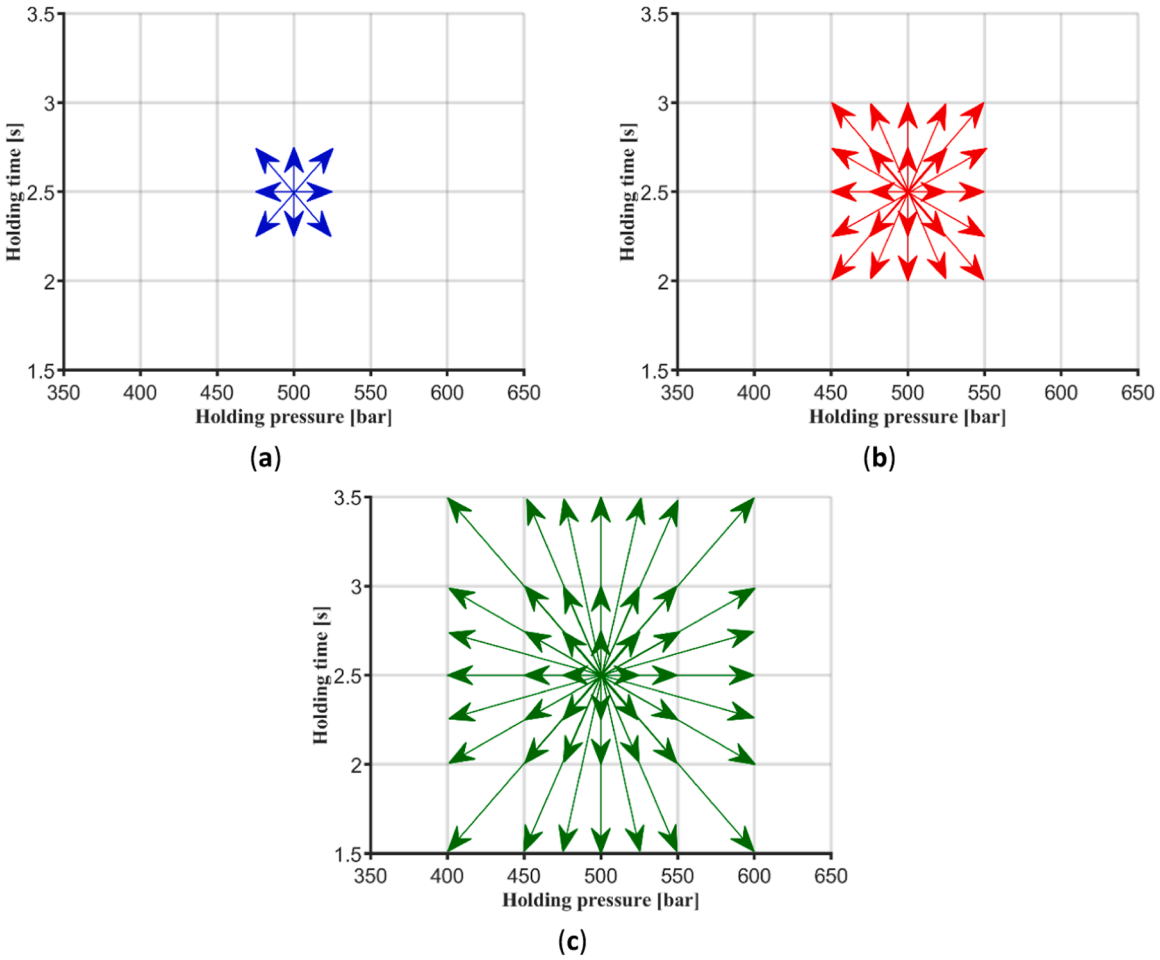


Fig. 12. (a) 9 possible actions (b) 25 possible actions (c) 49 possible actions.

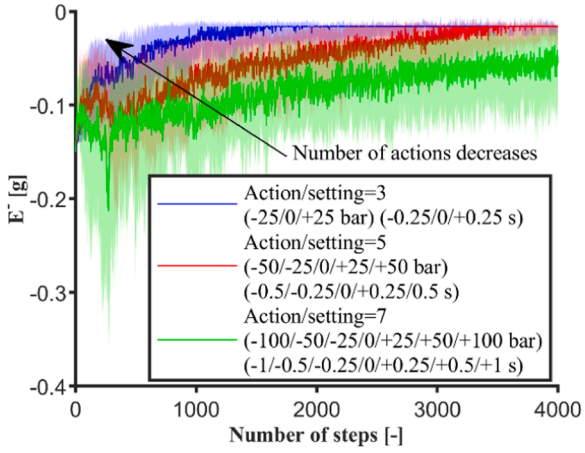


Fig. 13. The effect of the number of possible actions.

algorithm calculates the probability of each action for a given state. Therefore, we could provide the Actor-Critic algorithm with these function weights from a previous learning scenario, thus giving the algorithm a priori knowledge.

3.2.1. The effect of prior knowledge coming from injection molding experiments

First, we examined how the injection molding data can be used as prior knowledge. Of course, the weight vectors cannot be used directly from previous learning if the number of states or actions differs from the new learning scenarios because, in this case, the number of function weights is different (Fig. 15). Therefore, we used data from previous learning containing the same number of states and actions (we used the default algorithm settings from Table 2).

In Fig. 15 a, we can see state-value function weights for 10 states. Here the third state is the best for the algorithm. However, the value of the tenth state is, for some reason, greater than the surrounding states. We can see a similar phenomenon when there are 50 states (Fig. 15 b), as the first five and last five states appear to remain at the initial values. This is because the algorithm does not update the values of these states. This is because exploration did not cover these states in most cases or averaging smoothed them out.

To understand the effect of prior knowledge, we compared the learning scenarios with and without prior knowledge (Fig. 16 a). It is clear from the results that prior knowledge makes learning much faster. If we take the median curve, the algorithm crosses the acceptance limit after only 26 steps. It is important to take into account that this many steps are needed because the algorithm changed the injection molding parameters by 25 bar and 0.25 s (small action size). When we let the algorithm to choose from more actions (and thus from even greater actions), the algorithm will be faster, and the median curve reaches the limit from 17 steps (Fig. 16 b). This is a very important result because it shows the effectiveness of the Actor-Critic algorithm. Suppose we can use prior knowledge where the algorithm learned the use of many possible actions (and the convergence to the target value has been achieved in this prior knowledge). This can give us a significant advantage in the application because the algorithm can choose actions

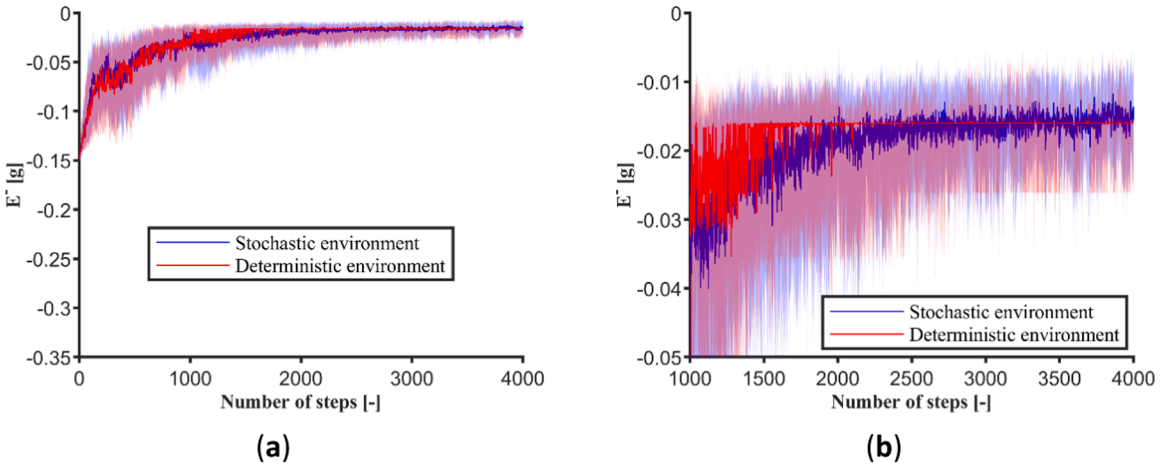


Fig. 14. The effect of different types of environments (a) during full learning. (b) at the convergence to the target.

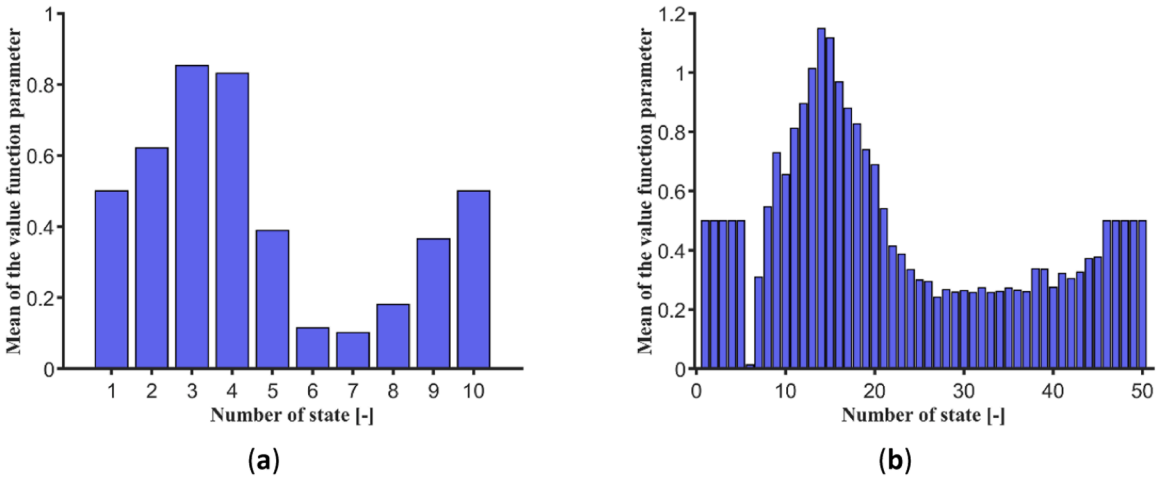


Fig. 15. The mean of the value function parameters after 4000 learning steps. (a) with 10 discrete states, (b) with 50 discrete states.

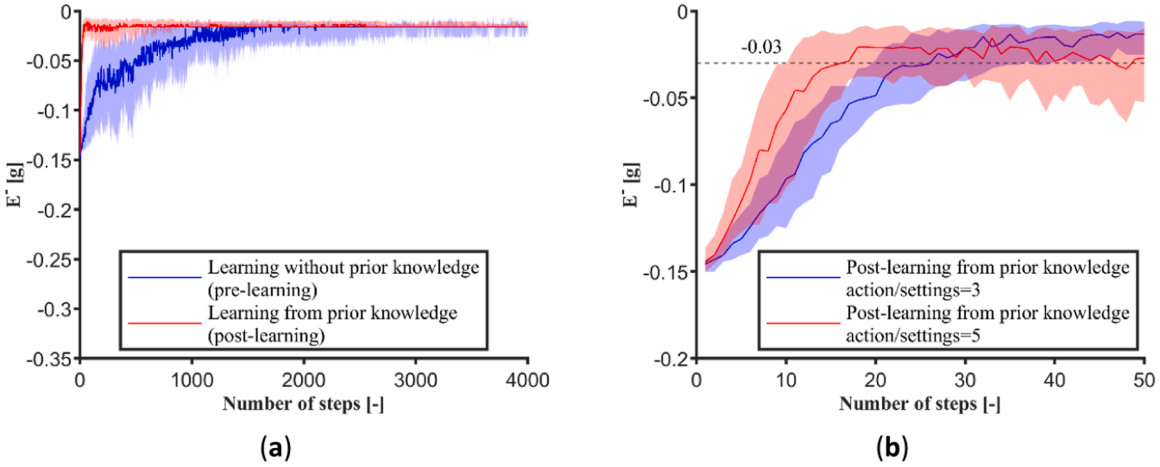


Fig. 16. (a) The effect of learning from prior knowledge (from injection molding) (b) the effect of possible actions on the post-learning.

far from the target, which makes larger steps toward the target.

3.2.2. *The effect of prior knowledge from simulation*

However, determining prior knowledge from injection molding is material- and energy-intensive. One way to do this is to experiment with many injection molding parameter combinations and make learning scenarios from the dataset (as we did in our analysis). The other way is to have the algorithm handle the machine and train it with real-time data until it produces stably in the optimal range. The results of the first experiment (3.1) show that this usually takes thousands of injection

molding cycles depending on state aggregation, actions, etc. Either way, both methods require a lot of time and material, which, under industrial conditions, do not always fit in with production.

Because of these drawbacks, we used a different approach. We collected prior knowledge from injection molding simulations. For this, we made injection molding simulations using the same combinations of injection molding settings. Of course, simulations will not capture real injection molding perfectly without the optimized simulation parameters (such as material properties, cooling system, and so on). However, our goal was not to adjust the simulation to reality but to get prior knowledge from roughly accurate simulations and give this to the Actor-Critic algorithm. If this were possible, the injection molding machine capacity and material waste could be minimized.

The probability distribution of part weight and the dataset from simulation (SD) and injection molding (ID) can be seen in Fig. 15. It is clear that there is a significant difference between the two datasets. The probability density function of simulation is narrower than the probability density function of injection molding (Fig. 15 a). In Fig. 15 b, we can see the reason for this. In simulations, the effect of holding pressure and holding time is smaller than in injection molding, and because of that, the change in part weight is smaller too. However, even though the magnitude of process parameters is smaller, the mechanism of change is similar; for example, we can see a saturation effect as a function of holding time. Interestingly, the saturation effect (due to holding time) appears in the simulations with 0 bar holding pressure, although no such effect is visible in reality. This saturation effect increases with holding pressure in the ID, but the same increase is significantly smaller in the SD. In our case, we wanted to use simplified simulations without the accurate simulation of mold deformation. In addition, we did not specify the cooling circuits of the mold or the coolant flowing in them in order to reduce the calculation time of these simulations. Other differences may be caused by the fact that we have not specified the thermal behavior of the material based on real measurements. Instead, we used the data provided the Moldex3D database. However, our aim was to make it possible to use a priori knowledge without these extra measurements.

If we want to use this method with different datasets (such as ID and SD), we can reconsider the method of state aggregation from two sides. On the one hand, the simulation space can be viewed as being driven by a similar mechanism on a different scale. This implies that we need a similar policy to reach the goal. For example, if we want to go from the origin [0,0] to coordinate [3,400], we should use the same policy in each dataset. Because of this line of thought, it would be logical to use state aggregation with rescaling (from the SD to the ID after pre-learning). We call this method dataset-dependent state aggregation in our research.

On the other hand, our goal is to produce a part with a specific part weight (8.75+/-0.03 g). Since the scale of the two datasets significantly differs (Fig. 17), the location of the optimal values will be different in each dataset. Therefore, for example, in simulation, the optimal values could be in the second state, but in the real-life injection molding dataset, these optimal values would be located in the fifth state. So, another way could be to use state aggregation independently of the datasets. Another method would be to use global state aggregation, where the weight of a given product is in the same state for both data sets. We call this method dataset-independent state aggregation in our research.

Therefore, we investigated the learning of prior knowledge (pre-learning) from the SD with both state aggregation, and then, the algorithm used this knowledge with the ID dataset (post-learning). Fig. 18 shows the results of the two post-learning scenarios (pre-learning reached stable production in both cases). The algorithm finds setting combinations with the needed part weight much faster with dataset-independent state aggregation. Of course, dataset-dependent state aggregation can be useful when the relative location of the goal is the same in each dataset. However, this requires relatively accurate simulations, the drawbacks of which have been described earlier.

3.2.3. *The effect of the initial starting point obtained from the simulation*

The learning process can be accelerated if the initial setting combination is changed for a setting combination possibly nearer the target combination. Up to this point, learning started in each scenario with the initial injection molding settings of a holding pressure of 0 bar and a

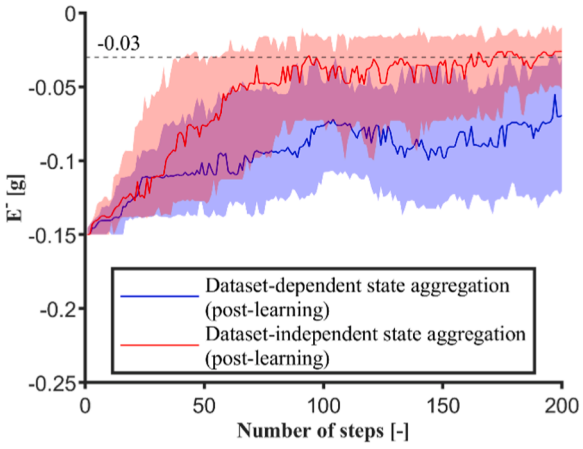


Fig. 18. The effect of dataset dependence of state aggregation on post-learning.

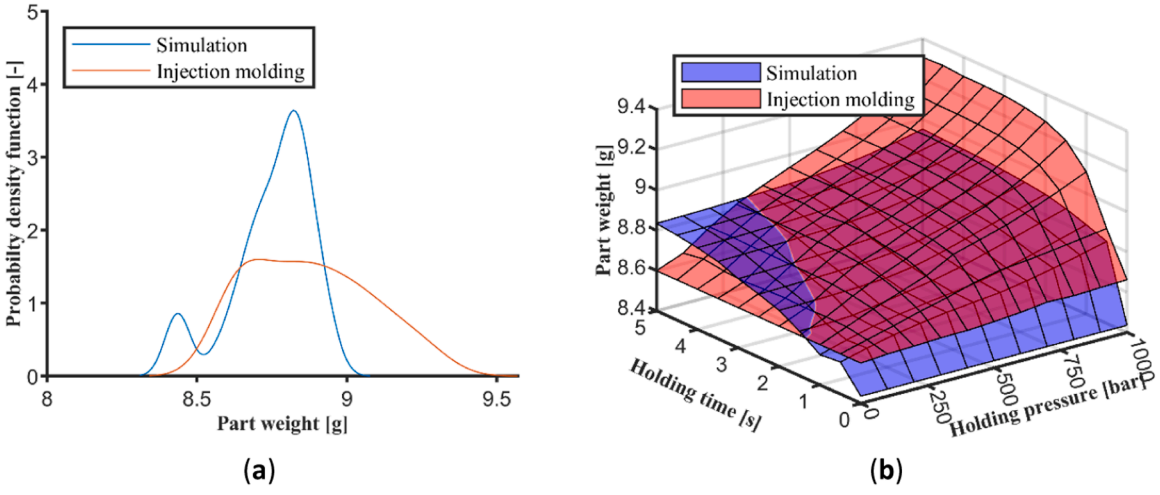


Fig. 17. Difference between the simulation and injection molding datasets. (a) probability density function (b) measured data in the search space.

holding time 0 s. However, from an injection molding simulation, we can get information about what setting combinations give the best results, and give those to the algorithm as initial values. Learning will look very different in this case. Fig. 19 a shows that the algorithm will start from a better state because the absolute error will be smaller at the first step. Therefore, the algorithm will produce acceptable parts with fewer steps (fewer injection molding cycles). However, it is important to highlight that the algorithm will make non-acceptable products after that because it adjusts its policy for the new search space. A good solution could be to stop the algorithm as soon it reaches the target value and produce parts with the same settings until production steps out of the tolerance due to some noise or anomaly, at which point the algorithm would be restarted. After the 8th setting adjustment, the algorithm already produces an acceptable product in more than 50 % of learning scenarios (Fig. 19 b).

3.2.4. Guidelines for adequate prior knowledge

Our results show that using prior knowledge significantly improves the performance of the algorithm. To obtain useful prior knowledge for the application, we should consider the following aspects.

In injection molding simulations, the geometry of the product must be accurately modeled to ensure that the filling of the product closely approximates filling in real injection molding. This may mean relatively little work compared to refining the simulation or measuring the behavior of the material, but it is important for the needed policy. For prior knowledge, having a clear picture about the search space is important. Therefore, a series of injection molding simulations are needed, so we recommend a dense mesh for the simulation. In our case, the mesh parameter was 1 mm for the given product. We made relatively simple simulations (the simulations ran for about 40 minutes on average). This is important because the simulation and real results will differ to some extent, but the amount of work invested remains relatively small.

After that, with the simulation results, pre-learning is needed. The number of steps during pre-learning should be long enough for the algorithm to stabilize and reach the target. The state aggregation used during pre-learning should also be used with post-learning. In this way, the corresponding elements of the weight vectors will refer to the same aggregated state during both learning processes. Our results confirmed that a global, dataset-independent state aggregation is more useful than dividing the datasets into a similar number of states. We recommend the use of multiple pre-learning scenarios (in our case, we did 100 learning scenarios) to ensure that the algorithm gathers enough experience due to its stochastic behavior. Of course, for the algorithm to work correctly, the number and value of the actions that the algorithm can select must be the same during pre- and post-learning.

Considering these guidelines, we believe that a prior knowledge suitable for learning can be obtained from the weight vectors available at the end of pre-learning. We also showed that the optimal technological settings found during pre-learning can also be interpreted as part of the prior knowledge.

4. Conclusion

In our study, we examined the usability of the Actor-Critic algorithm for injection molding. Our investigation is based on two parts. In the first part, we discussed how the algorithm parameters affect learning in this environment. We analyzed the effect of state aggregation, the reward function, learning rate, possible actions, and the impact of the stochastic environment.

The increase in learning rate and reward will make the learning of the algorithm faster. On the other hand, increasing the number of states reduces the speed of learning, as does increasing the number of actions, which increases the number of parameters to be discovered. We used a deterministic environment for most of our investigation, but showed that the algorithm will learn similarly with a stochastic environment. Of course, the reward will change in that case after each step. State aggregation by weight performed slightly better than state aggregation by technology, but the difference was relatively small. We concluded from the investigation that the practical use of an Actor-Critic algorithm needs improvement because the algorithm requires too many learning steps (e.g., injection molding cycles) and information to produce good parts.

Therefore, in the second half of the study, we discuss how prior knowledge can be used to improve the performance of this algorithm. We performed injection molding simulations to get prior knowledge about the process and used this knowledge for the Actor-Critic algorithm. As a result, the algorithm found the ideal settings more than 50 times out of practically 100 times after only 9 injection molding cycles. We proved that the Actor-Critic algorithm we used produces acceptable products much faster (with fewer injection molding cycles) if we use prior knowledge from simulation. With this method, the algorithm could compete with the work of a professional. This indicates that reinforcement learning and especially the Actor-Critic algorithm could set up the injection molding machine with the right initial settings. In the future, we will analyze the effect of the experimental design for prior knowledge and plan to validate our method by testing it in industrial applications.

Ethical approval

This article does not contain any studies with human participants or

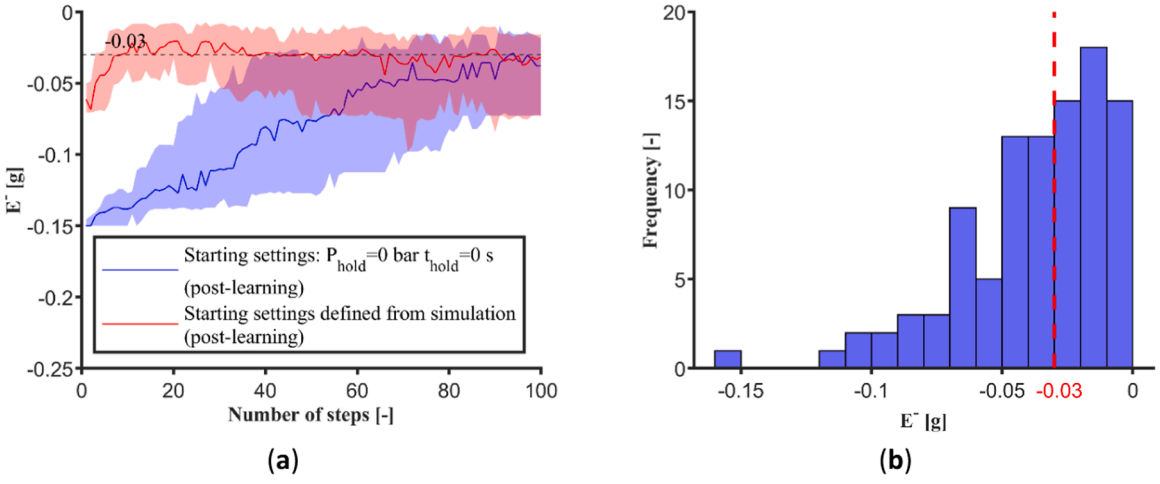


Fig. 19. (a) The effect of initial setting combinations for injection molding (b) the distribution of error from the target value after the 8th setting adjustment.

animals performed by any of the authors.

Funding

This work was supported by the National Research, Development and Innovation Office, Hungary [OTKA FK134336, OTKA FK138501]; National Laboratory for Renewable Energy has been implemented with the support provided by the Recovery and Resilience Facility of the European Union within the framework of Programme Széchenyi Plan Plus [Project no. RRF-2.3.1-21-2022-00009]; New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund [ÚNKP-22-3-II-BME-105].

CRediT authorship contribution statement

Daniel Török: Writing – review & editing, Supervision, Project administration, Funding acquisition. **Richárd Dominik Párizs:** Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of Competing Interest

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

Data availability

The data that has been used is confidential. Data will be made available on request.

Acknowledgement

The authors thank ARBURG HUNGÁRIA KFT. for the ARBURG Allrounder injection moulding machine, and TOOL-TEMP HUNGÁRIA KFT., LENZKES GMBH and PIOVAN HUNGARY KFT. for the accessories.

References

[1] A.C. Pereira, F. Romero, A review of the meanings and the implications of the Industry 4.0 concept, Procedia Manuf. 13 (2014) 1206–1214, https://doi.org/10.1016/j.promfg.2017.09.032).
[2] S. Vaidya, P. Ambad, S. Bhosle, Industry 4.0 – a glimpse, in: Procedia Manufacturing, **20**, 2018, pp. 233–238, https://doi.org/10.1016/j.promfg.2017.09.032).
[3] D. Guo, M. Li, Z. Lyu, K. Kang, W. Wu, R.Y. Zhong, G.Q. Huang, Synchroperation in industry 4.0 manufacturing, Int. J. Prod. Econ. 238 (2021) 108171, https://doi.org/10.1016/j.ijpe.2021.108171).
[4] M. Ghobakhloo, Industry 4.0, digitization, and opportunities for sustainability, J. Clean. Prod. 252 (2020) 119869, https://doi.org/10.1016/j.jclepro.2019.119869).
[5] H. Lasi, P. Fettke, H.G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Bus. Inf. Syst. Eng. 6 (2014) 239–242 (https://doi.org/10.1007/s12599-014-0334-4).
[6] T. Ageyeva, S. Horváth, J.G. Kovács, In-mold sensors for injection molding: on the way to industry 4.0, Sensors 19 (2019) 3551 (https://doi.org/10.3390/s19163551).
[7] A. Jandyal, I. Chaturvedi, I. Wazir, A. Raina, M.I.U. Haq, 3D printing – a review of processes, materials and applications in industry 4.0, Sustain. Oper. Comput. 3 (2022) 33–42, https://doi.org/10.1016/j.susoc.2021.09.004).
[8] M. Mehrpouya, A. Dehghanghadikolaei, B. Fotovvati, A. Vosooghnia, S. S. Emamian, A. Gisario, The potential of additive manufacturing in the smart factory industrial 4.0: a review, Appl. Sci. 9 (2019) 3865, https://doi.org/10.3390/app9183865).
[9] M. Oleksy, G. Budzik, A. Sanocka-Zajdel, A. Paszkiewicz, M. Bolanowski, R. Oliwa, Ł. Mazur, Industry 4.0 Part I. Selected applications in processing of polymer materials, Polymery 63 (2018) 531–535 (https://doi.org/10.14314/polimery.2018.7.7).
[10] H. Parmar, T. Khan, F. Tucci, R. Umer, P. Carlone, Advanced robotics and additive manufacturing of composites: towards a new era in Industry 4.0, Mater. Manuf. Process. 37 (5) (2022) 483–517, https://doi.org/10.1080/10426914.2020.1866195).
[11] S.S. Aminabadi, P. Tabatabai, A. Steiner, D.P. Gruber, W. Friesenbichler, C. Habersohn, G. Berger-Weber, Industry 4.0 in-line Ai quality control of plastic injection molded parts, Polymers 14 (17) (2022) 3551, https://doi.org/10.3390/polym14173551).
[12] V. Brnadić, T. Breški, Optimisation of mould design for injection moulding-numerical approach, Teh. čki Glas. 15 (2) (2021) 258–266, https://doi.org/10.31803/tg-20210531204548).
[13] T. Tábi, K. Pölöskei, The effect of processing parameters and calcium-stearate on the ejection process of injection molded poly (lactic acid) products, Period. Polytech. Mech. Eng. 66 (1) (2022) 17–25, https://doi.org/10.3311/PPme.18246).
[14] Ramini M., Agnelli S., Ramorino G.: Applications of shear heating parameter for injection molding process optimization of AEM rubber compounds, Express Polym. Lett., 16(4), 354-367 (2022). (https://doi.org/10.3144/expresspolymlett.2022.27).
[15] Minguez-Enkovaara L.F., Carrión-Vilches F.J., Avilés M.D., Bermúdez M.D.: Effect of graphene oxide and ionic liquid on the sliding wear and abrasion resistance of injection molded PMMA nanocomposites, Express Polym. Lett., 17(3), 237-251 (2023). (https://doi.org/10.3144/expresspolymlett.2023.18).
[16] Guo G.: Investigation on surface roughness of injection molded polypropylene parts with 3D optical metrology, Int. J. Interact. Des.Manuf., 16(1), 17-23 (2022). (https://doi.org/10.1007/s12008-021-00796-8).
[17] F.D. Santis, R. Pantani, V. Speranza, G. Titomanlio, Analysis of shrinkage development of a semicrystalline polymer during injection molding, Ind. Eng. Chem. Res. 49 (5) (2010) 2469–2476 (https://doi.org/10.1021/ie901316p).
[18] X. Zhou, Y. Zhang, T. Mao, H. Zhou, Monitoring and dynamic control of quality stability for injection molding process, J. Mater. Process. Technol. 249 (2017) 358–366, https://doi.org/10.1016/j.jmatprotec.2017.05.038).
[19] Barghash M.A., Alkaabneh F.A.: Shrinkage and warpage detailed analysis and optimization for the injection molding process using multistage experimental design, Qual. Eng., 26(3) 319-334 (2014). (https://doi.org/10.1080/08982112.2013.852679).
[20] D.O. Kazmer, S. Westerdale, A model-based methodology for on-line quality control, Int. J. Adv. Manuf. Technol. 42 (2009) 280–292, https://doi.org/10.1007/s00170-008-1592-4).
[21] S. Zhang, R. Dubay, M. Charest, A principal component analysis model-based predictive controller for controlling part warpage in plastic injection molding, Expert Syst. Appl. 42 (6) (2015) 2919–2927, https://doi.org/10.1016/j.eswa.2014.11.030).
[22] M. Altan, Reducing shrinkage in injection moldings via the Taguchi, ANOVA, and neural network methods, Mater. Des. 31 (1) (2010) 599–604 (https://doi.org/10.1016/j.matdes.2009.06.049).
[23] F. Finkeldey, J. Volke, J.C. Zarges, H.P. Heim, P. Wiederkehr, Learning quality characteristics for plastic injection molding processes using a combination of simulated and measured data, J. Manuf. Process. 60 (2020) 134–143, https://doi.org/10.1016/j.jmapro.2020.10.028).
[24] B. Zink, F. Szabó, I. Hatos, A. Suplicz, N.K. Kovács, H. Hargitai, T. Tábi, J. G. Kovács, Enhanced injection molding simulation of advanced injection molds, Polymers 9 (2) (2017) 77, https://doi.org/10.3390/polym9020077).
[25] J.Y. Chen, J.X. Zhuang, M.S. Huang, Monitoring, prediction and control of injection molding quality based on tie-bar elongation, J. Manuf. Process. 46 (2019) 159–169, https://doi.org/10.1016/j.jmapro.2019.09.005).
[26] G. Gordon, D.O. Kazmer, X. Tang, Z. Fan, R.X. Gao, Quality control using a multivariate injection molding sensor, Int. J. Adv. Manuf. Technol. 78 (2015) 1381–1391, https://doi.org/10.1007/s00170-014-6706-6).
[27] Yang A., Guo W., Han T., Zhao C., Zhou H., Cai J.: Feedback control of injection rate of the injection molding machine based on PID improved by unsaturated integral, Shock Vibr., 9960021 (2021). (https://doi.org/10.1155/2021/9960021).
[28] M.H. Tsai, J.C. Fan-Jiang, G.Y. Liou, F.J. Cheng, S.J. Hwang, H.S. Peng, H.Y. Chu, Development of an online quality control system for injection molding process, Polymers 14 (2020) 1607, https://doi.org/10.1155/2020/7023616).
[29] S. Kumar, H.S. Park, C.M. Lee, Data-driven smart control of injection molding process, CIRP J. Manuf. Sci. Technol. 31 (2020) 439–449, https://doi.org/10.1016/j.cirpj.2020.07.006).
[30] Z. Wang, W. Feng, J. Ye, J. Yang, C. Liu, A study on intelligent manufacturing industrial internet for injection molding industry based on digital twin, Complexity (2021) 8838914, https://doi.org/10.1155/2021/8838914).
[31] C.W. Su, W.J. Su, F.J. Cheng, G.Y. Liou, S.J. Hwang, H.S. Peng, H.Y. Chu, Optimization process parameters and adaptive quality monitoring injection molding process for materials with different viscosity, Polym. Test. 109 (2022) 107526, https://doi.org/10.1016/j.polymertesting.2022.107526).
[32] J.C. Chen, G. Guo, W.N. Wang, Artificial neural network-based online defect detection system with in-mold temperature and pressure sensors for high precision injection molding, Int. J. Adv. Manuf. Technol. 110 (2020) 2023–2033, https://doi.org/10.1007/s00170-020-06011-4).
[33] H.S. Park, D.X. Phuong, S. Kumar, AI based injection molding process for consistent product quality, Procedia Manuf. 28 (2019) 102–106, https://doi.org/10.1016/j.promfg.2018.12.017).
[34] P. Hamet, J. Tremblay, Artificial intelligence in medicine, Metabolism 69 (2017) S36–S40, https://doi.org/10.1016/j.metabol.2017.01.011).
[35] G. Culot, G. Nassimbeni, G. Orzes, M. Sartor, Behind the definition of Industry 4.0: analysis and open questions, Int. J. Prod. Econ. 226 (2020) 107617, https://doi.org/10.1016/j.ijpe.2020.107617).
[36] R. Nian, J. Liu, B. Huang, A review on reinforcement learning: introduction and applications in industrial process control, Comput. Chem. Eng. 139 (2020) 106886 (https://doi.org/10.1016/j.compchemeng.2020.106886).
[37] Duda R.O., Hart P.E., Stork D.G.: Pattern classification, Wiley-Interscience, ISBN: 0471056693 (2000).

[38] T. Mao, Y. Zhang, Y. Ruan, H. Gao, H. Zhou, D. Li, Feature learning and process monitoring of injection molding using convolution-deconvolution auto encoders, Comput. Chem. Eng. 118 (2018) 77–90 (https://doi.org/10.1016/j.compchemeng.2018.07.009).
[39] F. Guo, J. Liu, X. Zhou, H. Wang, Y. Zhang, D. Li, H. Zhou, An effective retrival method for 3D models in plastic injection molding for process reuse, Appl. Soft Comput. 101 (2021) 107034, https://doi.org/10.1016/j.asoc.2020.107034).
[40] A. Jaiswal, A.R. Babu, M.Z. Zadeh, D. Banerjee, F. Makedon, A survey on contrastive self-supervised learning, Technologies 9 (1) (2021) 2, https://doi.org/10.3390/technologies9010002).
[41] H. Jung, J. Jeon, D. Choi, J.Y. Park, Application of machine learning techniques in injection molding quality prediction: implications on sustainable manufacturing industry, Sustainability 13 (8) (2021) 4120, https://doi.org/10.3390/su13084120).
[42] K.C. Ke, M.S. Huang, Quality classification of injection-molded components by using quality indices, grading and machine learning, Polymers 13 (3) (2021) 353, https://doi.org/10.3390/polym13030353).
[43] R.D. Párizs, D. Török, T. Ageyeva, J.G. Kovács, Machine learning in injection molding: an industry 4.0 method of quailty prediction, Sensors 22 (7) (2022) 2704 (https://doi.org/10.3390/s22072704).
[44] Z. Song, S. Liu, X. Wang, Z. Hu, Optimization and prediction of volume shrinkage and warpage of injection-molded thin-walled parts based on neural network, Int. J. Adv. Manuf. Technol. 109 (2020) 755–769, https://doi.org/10.1007/s00170-020-05558-6).
[45] S. Lee, Y. Yun, S. Park, S. Oh, C. Lee, J. Jeong, Two phases anomaly detection based on clustering and visualization for plastic injection molding data, Procedia Comput. Sci. 201 (2022) 519–526, https://doi.org/10.1016/j.procs.2022.03.067).
[46] M. Tejedor, A.Z. Woldaregay, F. Godtliebsen, Reinforcement learning application in diabetes blood glucose control: a systematic review, Artif. Intell. Med. 104 (2020) 101836, https://doi.org/10.1016/j.artmed.2020.101836).
[47] H.I. Ugurlu, S. Kalkan, A. Saranli, Reinforcement learning versus conventional control for controlling a planar bi-rotor platform with tail appendage, J. Intell. Robot. Syst. 102 (2021) 1–17 (https://doi.org/10.1007/s10846-021-01412-3).
[48] Z. Wang, T. Hong, Reinforcement learning for building controls: the opportunities and challenges, Appl. Energy 269 (2020) 115036, https://doi.org/10.1016/j.apenergy.2020.115036).
[49] J.E. Sierra-Garcia, M. Santos, Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories, Expert Syst. 41 (2) (2024) e13076, https://doi.org/10.1111/exsy.13076).
[50] A.T.D. Perera, P. Kamalaruban, Applications of reinforcement learning in energy systems, Renew. Sustain. Energy Rev. 137 (2021) 110618, https://doi.org/10.1016/j.rser.2020.110618).
[51] S. Lee, Y. Cho, Y.H. Lee, Injection mold production sustainable scheduling using deep reinforcement learning, Sustainability 12 (20) (2020) 8718, https://doi.org/10.3390/su12208718).
[52] X. Li, Q. Luo, L. Wang, R. Zhang, F. Gao, Off-policy reinforcement learning-based novel model-free minmax fault-tolerant tracking control for industrial processes, J. Process Control 115 (2022) 145–156, https://doi.org/10.1016/j.jprocont.2022.05.006).
[53] Y. Qin, C. Zhao, F. Gao, An intelligent non-optimality self-recovery method based on reinforcement learning with small data in big data era, Chemom. Intell. Lab. Syst. 176 (2018) 89–100, https://doi.org/10.1016/j.chemolab.2018.03.010).
[54] F. Guo, X. Zhou, J. Liu, Y. Zhang, D. Li, H. Zhou, A reinforcement learning decision model for online process paramters optimization from offline data in injection molding, Appl. Soft Comput. 85 (2019) 105828, https://doi.org/10.1016/j.asoc.2019.105828).

Richárd Dominik Párizs obtained his B.Sc. and M.Sc. degree in Mechanical Engineering at the Budapest University of Technology and Economics (Hungary) in 2018 and 2020, respectively. He is currently a Ph.D. student at the Department of Polymer Engineering, Budapest University of Technology and Economics. His research interests is the use of machine learning for injection molding technology, mainly supervised learning and reinforcement learning.

Dániel Török is an assistant professor at the Department of Polymer Engineering, Faculty of Mechanical Engineering, Budapest University of Technology and Economics (Hungary), and a research fellow at MTA-BME Lendület Lightweight Polymer Composites Research Group, Hungarian Academy of Sciences (Hungary). His research interests include injection molding, polymer testing, polymer materials science, machine learning and image processing.