Model-free Adaptive Optimal Control of Episodic Fixed-horizon Manufacturing Processes Using Reinforcement Learning

Johannes Dornheim*, Norbert Link, and Peter Gumbsch

Abstract: A self-learning optimal control algorithm for episodic fixed-horizon manufacturing processes with time-discrete control actions is proposed and evaluated on a simulated deep drawing process. The control model is built during consecutive process executions under optimal control via reinforcement learning, using the measured product quality as a reward after each process execution. Prior model formulation, which is required by algorithms from model predictive control and approximate dynamic programming, is therefore obsolete. This avoids several difficulties namely in system identification, accurate modeling, and runtime complexity, that arise when dealing with processes subject to nonlinear dynamics and stochastic influences. Instead of using the pre-created process and observation models, value-function-based reinforcement learning algorithms build functions of expected future reward, which are used to derive optimal process control decisions. The expectation functions are learned online, by interacting with the process. The proposed algorithm takes stochastic variations of the process conditions into account and is able to cope with partial observability. A Q-learning-based method for adaptive optimal control of partially observable episodic fixed-horizon manufacturing processes is developed and studied. The resulting algorithm is instantiated and evaluated by applying it to a simulated stochastic optimal control problem in metal sheet deep drawing.

Keywords: Adaptive optimal control, manufacturing process optimization, model-free optimal control, reinforcement learning.

1. INTRODUCTION

Series production of parts is a repetition of processes, transforming each part from an initial state to some desired end state. Each process is executed by a finite sequence of - generally irreversible - processing steps. Examples are processes involving plastic deformation and subtractive or additive manufacturing processes. The process optimization is formulated as a Markov decision process with finite horizon. The control strategy (policy) can be refined between subsequent process executions, which are called episodes in reinforcement learning [1]. This allows for the adaptation of the policy to changing process conditions. The proposed reinforcement learning approach does not require an explicit process model and is therefore highly adaptive.

The episodic fixed-horizon manufacturing processes considered here can be regarded as nonlinear stochastic processes. The episode length (horizon) is determined externally and each episode consists of $T$ irreversible control steps. Our example process used throughout this paper, deep drawing, is depicted in Fig. 1. Based on the measured quality of the process episode result, costs are assigned and transferred to the agent by a reward signal $R_T$ at the end of each execution. The control effort can be reflected by intermediate costs. The goal is to find a control policy that minimizes the cost and thereby optimizes the process performance regarding the resulting product quality and the process efficiency. The processes considered here are not fully observable. Current sensor observations are not sufficient for deriving optimal control decisions, therefore historic information has to be taken into account. In addition, no prior information, (like reference trajectories, a process model or an observation model) is given and the optimal control has to be learned during execution.

We use the optimal control of a deep drawing process as application example and for method evaluation. In deep drawing, a planar metal sheet (blank) is pushed by a punch...
into the hollow inner part of a die, causing the metal sheet to assume the desired form. The material flow is regulated by blank holders, which are pressing the blank against the outer part of the die with a prescribed blank holder force. Apart from fixed value parameters, like the initial blank shape and thickness variation, the temporal variation of the blank holder force is crucial for the resulting process quality. The influence of space and time variation schemes of blank holder forces on the process results is examined e.g., in [2], [3] and [4]. The process state, consisting of the strain-stress field of the blank and the friction, is not directly measurable. Instead, only some observables are accessible. The goal of optimal control in the application example is to produce cups with optimal mechanical strength while using as little material as possible. The blank holder force is optimized, while the punch acts with a predefined speed.

For a given combination of process model, observation model, and cost function, the optimal control problem can be solved by model-based optimal control methods. Offline optimal control is reached by dynamic programming [5]. Dynamic programming is subject to the so-called curse of dimensionality in high-dimensional state spaces, leading to difficulties in terms of the required sample size and the computational complexity. In the case of continuous state spaces, the application of dynamic programming requires a state space discretization. These problems are addressed in the field of approximate dynamic programming, combining dynamic programming with function approximation [6].

A family of methods for online optimal control that is often applied to industrial processes is model predictive control (MPC). An extensive overview of MPC and implementation examples for industrial processes can be found in [7]. Like approximate dynamic programming, MPC requires a process model, usually determined by linear system identification methods. While MPC with linear models has been explored extensively, nonlinear MPC is an active field of research [8]. The online computational costs in MPC depend on the performance of the prediction model and can be reduced by offline pre-calculations as in the explicit MPC method [9], or by using artificial neural networks to approximate the prediction model [10, 11].

The application of model-based optimal control methods is limited by the quality of the underlying process model. The identification of highly nonlinear processes from experimental data is very resource and time-consuming. Alternatively, numerical simulation methods, like the finite element method (FEM), can be used for process identification. In [12], approximate dynamic programming methods are applied for optimal control of FEM-simulated deep drawing processes. In [13], methods from nonlinear MPC in combination with FEM calculations are used for optimal control of a glass-forming process. However, accurate simulation models of nonlinear manufacturing processes are usually computationally demanding and are thus rarely used in recent work. Simulation efforts lead to high offline costs when using approximate dynamic programming, and extensive online costs when using MPC.

Due to the difficulties inherent in model-based optimal control, we pursue an adaptive model-free optimal control method. Adaptive online learning of optimal control policies without the need for a priori given models is accomplished by model-free reinforcement learning (RL) [1]. Reinforcement learning is an area of machine learning. Like dynamic programming, it operates on Markov decision processes and solves the Bellman equation. Unlike standard dynamic programming, the Bellman equation is solved online. Recent prominent applications of RL, in combination with methods from deep learning and graph search, are human-level video game control [14] and the superhuman performance in the game of Go [15]. RL in the context of model-based and model-free adaptive optimal control with continuous actuator values is often referred to as adaptive dynamic programming [16].

Fig. 1. Deep drawing optimal control as an episodic fixed-horizon process. For every control step \( t \), the optimal control task is to determine the control action \( u_t \) that maximizes the expected episode reward \( R_T \), based on the observations \( o_0, ..., o_{T-1} \) in the current episode and on data from previous episodes. (source, left image: commons.wikimedia.org).
ations of RL to tracking control problems and regulation control problems are surveyed in [17]. In [18], an adaptive dynamic programming algorithm, based on a nonlinear state-space model, is proposed for optimal control of wave energy converters. Several model-free adaptive approaches for optimal control in the context of infinite-horizon processes can be found in the literature. Recent examples are [19], where an RL approach is applied to optimal setpoint control of linear chemical systems, and [20], where a similar method is applied to a more complex polymerization reaction system. Applications of RL methods can also be found in optimal operational control, where a two-layered control structure is used to combine setpoint tracking on the device layer and the optimization with respect to operational objectives on an operational layer. Recent examples include [21] and [22]. In [21], Q-learning is used on both layers for model-free optimal operational control of a linearized thickening process. In [22], a PI controller on the device layer is combined with an actor-critic RL approach on the operational layer to achieve optimal operational control of nonlinear flotation processes without the requirement of a dedicated operational model.

Instead of solving the optimal control problem by using a given process model, model-free optimal control methods optimize the control policy online, while interacting with the process. Due to online learning, model-free methods for optimal control are inherently adaptive, whereas, according to [23], approaches for adaptive MPC focus on robustness and tend to be conservative. The absence of a process model in RL also results in online performance advantages over MPC and in the ability to handle nonlinear systems. The horizon of costs taken into account in RL is usually unbounded, while in MPC it is bounded by the chosen prediction horizon. One advantage of MPC over RL, however, is the ability to handle state constraints and feasibility criteria. MPC (unlike RL) has a mature theory of robustness and stability. These relations between MPC and RL are studied in depth in [23].

Regarding the size of the state space and the action space, RL suffers from the same limitations as dynamic programming. The basis of many RL algorithms is, therefore, the use of function approximation, to estimate the expected value functions based on experience data, stored in a so-called replay memory. Algorithms following this scheme are neural fitted Q-iteration [24] and the more recent deep Q-learning [14]. While in neural fitted Q-iteration artificial neural networks are retrained from scratch regularly, in deep Q-learning the artificial neural networks are iteratively redefined based on mini-batch samples uniformly drawn from the replay memory, enabling the computationally efficient use of deep learning methods.

Model predictive control, approximate dynamic programming, and reinforcement learning determine the optimal control action based on the current process state. Manufacturing processes are often only partially observable, the quantities measured in the current state do not unambiguously describe the state with respect to the optimization problem. Observation models are therefore used in model-based optimal control to reconstruct the current state from the observation and control history. If representative data of observables and corresponding state values can be drawn (as in [12]), an observation model can be learned from these data. The partially observable Markov decision process (POMDP) framework can be used to model partially observable environments [25]. Alternatively, sequence models such as recurrent neural networks (RNNs) are used in RL to derive surrogate states from the observation and control history ([25–27]).

Research on model-based optimal blank holder force control can be found in the publications of Senn et al. ([12]) and Endelt et al. ([28, 29]). In [12], approximate dynamic programming is used for the calculation of an optimal blank holder force controller based on a process model, which is learned on FEM-simulation results. A proportional optimal feedback control loop, optimizing blank holder forces regarding a reference blank draw-in, is presented in [28]. In [29], the in-process control is embedded in an outer iterative learning control loop. The iterative learning control transfers information between deep drawing executions for linear adaption of non-stationary process behavior.

The main contributions of this paper are: (a) The proposal of an adaptive optimal control concept, using an enhanced model-free RL algorithm for nonlinear, episodic, fixed-horizon manufacturing processes with a delayed cost signal. The algorithm (neural fitted Q-iteration [24]) has been enhanced, to make efficient use of the episodic fixed-horizon process structure and to deal with partial observability. (b) The identification of a class of manufacturing processes, to which the algorithm is directly applicable. (c) An extensive evaluation and parameter study on an interactive, partially observable deep drawing process, which is numerically simulated using the FEM.

The paper is structured as follows: Section 2 introduces the methodical background. In Section 3, the problem of adaptive optimal control of episodic fixed-horizon manufacturing processes is described and specified. Section 4 presents the proposed reinforcement learning approach. In Section 5, the evaluation environment, including the simulated process, is described, and in Section 6, we discuss quantitative and qualitative evaluation results.

2. BACKGROUND

2.1. Markov decision processes

The framework of Markov Decision Processes (MDP) is used to model time-discrete decision optimization problems and constitutes the formal basis of algorithms from
dynamic programming and reinforcement learning. An MDP is described by a 5-tuple \((X, U, P, R, \gamma)\), where \(X\) is the set of states \(x\), \(U\) is the set of control actions \(u\), \(P_a(x, x')\) is the probability of a transition to \(x'\) when applying the control action \(u\) in state \(x\), \(R_a(x, x')\) is a reward signal given for the transition from state \(x\) to \(x'\) due to action \(u\), and \(\gamma\) is a discount factor representing the relative weight decrease of future rewards. By definition an MDP satisfies the Markov property: The probability of any upcoming state \(x'\) depends only on the current state \(x\) and on action \(u\) and, given \(x\) and \(u\), is conditionally independent of previous states and actions. Hence,

\[
p(x_{t+1}, R_{t+1} \mid x_t, u_t, \ldots, x_t, u_t) = p(x_{t+1}, R_{t+1} \mid x_t, u_t),
\]

(1)

where \(x_t\) is the system state, \(R_t\) the given reward, and \(u_t\) the control action applied in control step \(t\). The goal of decision optimization algorithms is to solve a given MDP by finding an optimal policy \(\pi^* : X \rightarrow U\), a mapping from states to actions with the property that, when following \(\pi^*\) starting in the state \(x\), the expected discounted future reward \(V(x)\) is maximized.

2.2. Optimal control and the Bellman equation

Consider the nonlinear time-discrete system

\[
x' = f(x, u, w).
\]

(2)

According to the MDP definition in 2.1, \(x\) is a vector of system state variables, \(u\) is the control vector. \(x'\) denotes the system state following \(x\) under \(u\) and \(w\), where \(w \sim W\) are stochastic process conditions.

An optimal control problem for the nonlinear system \(f\), for a defined local cost function \(C(x, u, x')\), hidden conditions \(w\) and for \(\gamma\)-discounted future costs, can be solved by calculating the solution \(V^*\) of the Bellman equation (3), also named cost-to-go function in dynamic programming.

\[
V^*(x) = \min_{w \sim W} \mathbb{E}_{x \sim X} \left[ C(x, u, x') + \gamma V^*(x') \right].
\]

(3)

When (3) is solved, the optimal policy \(\pi^*(x)\) can be extracted by \(\arg\min_{w \sim W} \mathbb{E}_{x \sim X} [V^*(x')]\) for a given process model \(f\).

In this paper, following the standard notation of reinforcement learning, the system \(f\) and the optimization problem are modeled as an MDP as introduced in the previous section. In MDPs, instead of the cost function \(C\), a reward function \(R\) is used, leading to a maximization problem instead of the minimization in (3). The Bellman equation is then given by

\[
V^*(x) = \max_{u \in U} \mathbb{E}_{P} \left[ R_a(x, x') + \gamma V^*(x') \right],
\]

(4)

where the probability of \(x'\) is given by the transition probability function \(P_a(x, x')\), capturing stochastic process conditions \(w\).

2.3. Q-learning

The objective of Q-learning [30] is to find the optimal Q-function

\[
Q^*(x, u) = \mathbb{E}_P \left[ R_a(x, x') + \gamma \max_{u' \in U} Q^*(x', u') \right].
\]

(5)

Unlike (4), the Q-function is defined on state and action tuples \((x, u)\). By taking actions into account, the Q-function implicitly captures the system dynamics, and no additional system model is needed for optimal control. Once the optimal Q-function has been found, the optimal control policy \(\pi^*\) is given by

\[
\pi^*(x) = \arg\max_{w \sim W} Q(x, u).
\]

(6)

In Q-learning-based algorithms, \(Q^*\) is found by constantly updating a \(Q^*\)-approximation \(Q\), using experience tuples \((x, u, x', R)\) and a given learning rate \(\alpha \in [0, 1]\), while interacting with the process in an explorative manner

\[
Q'(x, u) \leftarrow (1 - \alpha)Q(x, u) + \alpha \left( R + \gamma \max_{u' \in U} Q(x', u') \right).
\]

(7)

If the action space \(U\) and the state space \(X\) are finite, convergence to \(Q^*\) (and, hence, to the optimal control policy) is guaranteed when using Q-learning, as long as all actions from \(U\) are repeatedly sampled for all states from \(X\) [30]. However, convergence may fail if a Q-function approximation is used instead of the tabular Q-function.

3. PROBLEM DESCRIPTION

In this paper, we consider sequentially executed fixed-horizon manufacturing processes. The processing of a single workpiece involves a constant number of discrete control steps and can be modeled as a Markov Decision Process (MDP), where for every policy and every given start state, a terminal state is reached at control step \(T\). Each processing may be subject to slightly different conditions (e.g., the initial workpiece, lubrication, tool wear). In reinforcement learning, the control of a fixed-horizon MDP from the start to the terminal state (here: single workpiece processing) is denoted as an episode, and tasks with repeated execution of episodes are denoted as episodic tasks. Hereafter, the term episode is used in reinforcement learning contexts, and the phrase process execution in manufacturing contexts.

Most processes, such as forming or additive manufacturing, are irreversible. The related control decisions are leading to disjoint sub-state-spaces \(X_t\) depending on the control step \(t\). The terminal state \(x_T\) is assessed, and the final product quality is quantified by a reward function. At each control step during the process execution, a negative reward can be assigned according to cost arising from the
execution of a dedicated action in the present state. For deterministic processes, where the process dynamics are solely dependent on the control parameters $u$, the structure of the fixed-horizon processing MDP is represented by a tree graph with the starting state as root vertex and the terminal states as leaf vertices. An exemplary MDP tree is depicted on the right-hand side of Fig. 2, relating to the optimal control of the blank holder force in deep drawing. In the optimal control cases considered in this paper, the process dynamics during an individual process execution are dependent on stochastic per-episode process conditions. In this scenario, the underlying MDP equals a collection of such trees, each representing a particular process condition.

3.1. Deep Drawing, Blank Holder Force Optimization

A controlled deep drawing process is used as a proof of concept, where the optimal variation of the blank-holder force is determined at processing time. As emphasized in Section 1, the result of the deep drawing process is strongly dependent on choosing appropriate blank holder force trajectories.

An interactive surrogate process, based on the ABAQUS FEM solver is used to simulate the process behavior for method development and evaluation. The current process state $x$ is simulated for the given control action $u$, the previous process state and process conditions $w$, which are randomly sampled from a given distribution. Based on $x$, the current reward $R$ is derived by a reward function, and observable values $o$ are derived by a sensor model. The $(R, o)$ tuple is finally reported to the control agent. A detailed description of the simulated environment is provided in Section 5.

4. MODEL-FREE OPTIMAL CONTROL APPROACH

For model-free and adaptive optimal control of fixed horizon manufacturing processes (FHMP), a Q-learning based algorithm (FHMP-Q-control) is proposed. The following listing outlines the FHMP-Q-Control algorithm, which is described in detail afterward.

```
FHMP-Q-CONTROL($\alpha$, $\epsilon_0$, $\lambda$, $k$, $T$, $\#episodes$)
1 for $t = 1$ to $T$
2 // initialize replay_memory,
3 // initialize $Q$, weights $\theta$
4 for $i = 1$ to $\#episodes$
5 $o_0 \leftarrow$ Env.observe()
6 $\bar{x} \leftarrow [o_0]$
7 for $t = 1$ to $T$
8 $u_t \leftarrow \pi_e(\bar{x})$ // Eq. (8)
9 $(o_t, R) \leftarrow$ Env.execute($u_t$)
10 $\bar{x}' \leftarrow \text{concat}(\bar{x}, o_t, u_t)$
11 // insert $(\bar{x}, u_t, \bar{x}', R)$ into replay_memory,
12 $\bar{x} \leftarrow \bar{x}'$
13 if $i \mod k = 0$
14 for $t = T$ to $1$
15 // extract $(X, y)$ from replay_memory,
16 // update weights $\theta$ according to Eqs. (9) to (11)
```

The FHMP-Q-control is applied to an optimal control task for $\#episodes$ episodes with horizon $T$. The environment is represented in the listing by an observation function $\text{Env.observe}()$ and an execution-function
**Env.execute(u).** The learning rate α determines the degree of influence of new information on the Q-Function update (7). The exploration rate ε and the decay rate λ are exploration-specific parameters, explained below.

Control actions are derived from an explorative policy πε and executed (lines 8, 9). Surrogate state descriptions ̄x are built from the observable values o, and control actions u (line 10, see Section 4.1). Transition data ( ̄x, u, ̄x′) and rewards R are stored in a replay memory (line 11). The replay memory is used for periodic retraining of the Q-function approximation every k episodes (lines 13 to 16, see Section 4.2).

During learning, an ε-greedy policy is used, acting randomly in a decayed ε-fraction of control actions, where ε = ε0e−λt. In all other cases the current approximation of the Q-function is used, to exploit the current knowledge about the environment. The explorative policy πε is given by

\[ \piε( ̄x) = \begin{cases} \arg\max_{u \in U} Qt( ̄x, u, Θt), & \text{if } z > ε0e^{-λt}, \\ \text{random } u_t, & \text{otherwise}, \end{cases} \tag{8} \]

where z is randomly sampled from the uniform distribution U(0, 1).

### 4.1. Partial observability

For real-world manufacturing processes, the observation oτ of the process state xτ is often restricted to indirect and noisy information from sensors. Q-learning relies on the formulation of the problem as a Markov decision process. If a process is partially observable [31], the Markov property (1) does not hold with respect to the observations.

In our approach, we use the full process information, provided during the episode, by concatenating the current observable and action history into a surrogate state vector ̄x. The dimension of ̄x is dependent on the current control step t ∈ [0, 1, ..., T], according to dim( ̄x) = t · (dim(o) + dim(u)).

### 4.2. Q-function approximation

The incorporation of function approximation into Q-learning allows for a generalization of the Q-function over the (X, U)-space. Thus it becomes possible to transfer information about local Q-values to newly observed states x and unexecuted control actions u. Approximating the Q-function, therefore, increases the learning speed in general and, furthermore, allows for an estimated representation of the Q-function in cases with a continuous state space.

We use feedforward neural networks as function approximation models, which are retrained every k episodes, based on data from a so-called replay memory. The replay memory consists of an increasing set of experience tuples ( ̄x, u, ̄x′, R). Due to the non-stationary sampling process (following πε), the use of standard Q-learning in combination with iteratively trained function approximation can result in a complete loss of experience (catastrophic forgetting) [32]. Using a replay memory for periodic retraining of Q-function approximation models has been shown to be more stable and data-efficient [24].

As described in Section 4.1, the dimension of the surrogate state ̄x is dependent on t, and multiple artificial neural networks Qt( ̄x, u, Θt), with weight parameters Θt, have to be used. For each tuple ( ̄x, u, R) in the replay memory, the Q-values are updated before retraining according to (7). The update is based on the estimated Q-values for the current control step t, Qt( ̄x, u, Θt), and the Q-function approximation for t + 1, Qt+1( ̄x′, u′, Θt+1), where Θt are weight values for control step t from the previous training iteration, and Θt+1 are weight values already updated in the current training iteration. The training target y for a given replay memory sample then is

\[ y = (1 - α)Qt( ̄x, u, Θt) + α[R + γ \max_{u′ \in U} Q_{t+1}( ̄x′, u′, Θ_{t+1})]. \tag{9} \]

The per-example loss used for training is the squared error

\[ L[(x, u), y, Θt] = (y - Q_t(x, u, Θt))^2. \tag{10} \]

The proposed update mechanism works backward in the control steps t and incorporates the already updated parameters for Qt+1 when updating Qt. Thereby, the propagation of reward information is accelerated.

At the beginning of an episode, the observable values only depend on the externally determined initial state, which in our case does not reflect any process influence. The Q-function Q0 therefore only depends on the control action u0. By assuming that the stored transition samples are representative regarding the transition probability function p( ̄x1|u0), the expected future reward for the first control step u0 (here denoted as Q0) is defined by

\[ Q_0(u_0) = E[R + γ \max_{u \in U} Q_t( ̄x_1, u)], \tag{11} \]

and is calculated directly by using the replay memory and the approximation model for Q1.

### 5. EVALUATION ENVIRONMENT

The presented approach is evaluated by applying it to a simulated deep drawing process, where the blank holder force is optimized. A controlled evaluation environment is implemented in python. The digital twin of a deep drawing process is based on FEM-simulations and used for the online process control assessment. The simulation model, as depicted on the left-hand side of Fig. 2, simplifies the deep drawing process, assuming rotational symmetry and isotropic material behavior. Due to its simplicity, the FEM model is very efficient with respect to computing time.
is published on GitHub. The python environment, together with the simulation model, is provided on GitHub\(^1\).

The ABAQUSS FEM explicit solver is used for this work. For the adaptive optimal control setting, the simulation has to be flexible, allowing for observables to be derived from the simulation state, and control variables to be set during process execution. Extensive use of the ABAQUSS scripting interface [33] in combination with analysis restarts (confer [34], Chapter 9.1.1) is made in order to obtain an efficient, flexible and reusable simulation for the adaptive optimal control setting.

5.1. FEM model

The rotationally symmetric FEM model, visualized on the left-hand side of Fig. 2, consists of three rigid parts interacting with the deformable blank. The circular blank has a thickness of 2.5 mm and a diameter of 40 mm. An elastic-plastic material model is used for the blank, which models the properties of Fe-28Mn-9Al-0.8C steel [35]. The punch pushes the blank into the die with constant speed and forms a cup with a depth of 25 mm. Blank holder force values can be set at the beginning of each of five consecutive control steps, where the blank holder force changes linearly in each control step from its current value to the value given at the start of the step. The target values can be chosen from the set \{20 kN, 40 kN, ..., 140 kN\} in each control step. The initial blank holder force is 0 kN.

5.2. Process disturbances

Stochastic process disturbances and observation noise are added to the simulated deep drawing process described in Section 5.1 to create a more realistic digital twin. Process disturbances are introduced by stochastic contact friction. The friction coefficient is modeled as a beta-distributed random variable, sampled independently for each deep drawing process execution (episode). The distribution parameters are chosen to be \(\alpha = 1.75, \beta = 5\). The distribution is rescaled to the range \([0, 0.14]\). For better reusability of simulation results, the friction coefficient is discretized into 10 bins of equal size. The resulting distribution is discrete and defined over 10 equidistant values from the interval \([0.014, 0.14]\) with a mode of 0.028.

5.3. Observables

The current process state \(x\) is dependent on the previous state, the control action \(u\) and the current friction coefficient. Neither the state \(x\) nor the friction coefficient is accessible by the agent. The underlying beta-distribution is not known either. Instead, three observable values \(o\), depicted in Fig. 2, are provided in each control step:

1) The current stamp force \(F_{\text{stamp}}\).
2) The current blank infeed in the horizontal-direction \(p_{\text{oxy}}\).
3) The current offset of the blank-holder in the vertical-direction \(p_{\text{oy}}\).

The measurement noise of these observables is assumed to be additive and to follow a normal distribution, where the standard deviation is chosen to be 1% of the respective value range for \(F_{\text{stamp}}\) and \(p_{\text{ox}}\), and 0.5% of the value range for \(p_{\text{ox}}\). This corresponds to common measurement noise characteristics encountered in analog sensors.

5.4. Reward function

The reward function can comprise local costs (related to the state and action of each single processing step) and global costs (assigned to the process result or final state). In the case of zero or constant local cost, the optimization is determined by the reward assigned to the terminal state, which is assessed by a quality control station. For our evaluation, we consider this case. The process goal is to produce a cup with optimal mechanical strength while using as little material as possible. The mechanical strength is represented by low internal stresses and sufficient material thickness. The reward function is the relevance-weighted harmonic mean of three reward terms \((r_a, r_p, r_c)\), representing these quantities.

FEM elements \(e\) of the blank are structured in the form of a discretization grid with \(m\) rows and \(n\) columns, depicted in Fig. 2. For the reward calculation, three \(m \times n\) matrices \((S, H^\psi, D^\chi)\) of element-wise simulation results are used. \(S\) is the mean von Mises Stress of element \(e_{ij}\), \(H^\psi\) is the height of element \(e_{ij}\) (in vertical \(\psi\)-direction) and \(D^\chi\) is the horizontal \(\chi\)-displacement of element \(e_{ij}\) between control step 0 and \(T\). Using these matrices, the reward is composed of the following three cost terms based on the simulation results: The \(L_2^2\)-norm of the vectorized von Mises stress matrix

\[
c_{\text{v}}(x_T) = \| \text{vec}(S) \|_2, \tag{12}\]

the blank thickness at the thinnest point

\[
c_{\text{h}}(x_T) = -\min(h), \tag{13}\]

where \(h\) is a vector of column-wise summed heights \(h_i = \sum_j H^\psi_{ij}\), and the summed displacements of elements in the last column, representing material consumption during drawing

\[
c_{\text{c}}(x_T) = \sum_i D^\chi_{in}. \tag{14}\]
The cost terms \( c_v \) are negated and scaled, with \( c_v, v \in \{ a, b, c \} \), resulting in approximately equally balanced reward function terms

\[
 r_v(x_T) = 10 \cdot \left( 1 - \frac{c_v(x_T) - c_v^{\text{min}}}{c_v^{\text{max}} - c_v^{\text{min}}} \right), \tag{15}
\]

where the extrema \( c_v^{\text{min}} \) and \( c_v^{\text{max}} \) are empirically determined per cost term, using 100 data tuples from randomly sampled terminal states. The final reward is defined by

\[
 R(x_T) = \begin{cases} 
 H(x_T, W), & \text{if } \forall i \in \{ a, b, c \} : r_v(x_T) \geq 0, \\
 0, & \text{otherwise}, 
\end{cases} \tag{16}
\]

where \( H \) is the weighted harmonic mean

\[
 H(x_T, W) = \frac{\sum_v w_v}{\sum v r_v(x_T)}. \tag{17}
\]

The harmonic mean was chosen to give preference to process results with balanced properties regarding the reward terms. The weights of the harmonic mean can be used to control the influence of cost terms to the overall reward. Equal weighting is used for our evaluation.

5.5. Q-function representation

The Q-function, as introduced in Section 4.2, is represented by a set of feedforward artificial neural networks, which are adapted via the backpropagation algorithm. The optimization uses the sum-of-squared-error loss function combined with a \( L^2 \) weight regularization term. The limited-memory approximation to the Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm is used for optimization because it has been shown to be very stable, fast converging and a good choice for learning neural networks when only a small amount of data is available [36]. Rectified linear unit functions (ReLU) are used as activation functions in the hidden layers. Two hidden layers are used for all networks, consisting of 10 units each for \( Q_1 \) and 50 neurons each for \( Q_2, Q_3 \) and \( Q_4 \), respectively.

6. RESULTS

The approach presented in Section 4 was investigated with a set of experiments, executed with the simulated deep drawing process described in Section 5.1. The proposed model-free adaptive optimal control algorithm is compared to non-adaptive model-based methods from model predictive control and from approximate dynamic programming. The non-adaptive methods are based on a given perfect process model, which does not capture stochastic process influences. This is reflected by a static friction coefficient of 0.028 in the simulation model. The non-adaptive methods are represented by a blank holder force trajectory which is optimal for the restricted simulation model and was determined by a full search over the trajectory space. The expected baseline reward of 5.13, which is used for the evaluation plots, is the expectation value of the reward reached when the determined trajectory is applied in the stochastic setting.

The experiments are conducted with the neural network hyperparameters defined in Section 5.5 and, unless otherwise stated, the following reinforcement learning parameters: learning rate \( \alpha = 0.7 \), exploration rate \( \epsilon_0 = 0.3 \) and \( \epsilon_0 \)-decay \( \lambda = 10^{-3} \). The model retraining interval \( k \) was chosen to be 50 process executions (episodes). The first Q-function training is carried out after 50 random episodes.

Fig. 3 shows the results of the reinforcement learning control for 10 independent experimental batches of 1000 deep drawing episodes each. The expected reward depends on the given friction distribution and denotes the mean reward that would be reached by a non-explorative policy by exploiting the current Q-function. The expected reward reached by the reinforcement learning agent is calculated for each of the 10 independent batches.

In Fig. 3, the mean expected reward (blue solid line) is plotted together with the range between the 10th percentile and the 90th percentile with linear interpolation (shaded area). The baseline (dashed gray line) is the expected reward for non-adaptive methods, determined as described above.

The performance of the Q-function approximators is evaluated by 5-fold cross-validation during each retraining phase. In Fig. 4 the coefficient of determination (R² Score), resulting from cross-validation, is plotted over the sequence of episodes. For \( t \in \{2, 3, 4\} \), the R²-Score

![Fig. 3. Expected reward for the RL approach during optimization (blue) and expected reward for the baseline approach (grey, dashed).](image-url)
is increasing over time as expected. For the first control step \((t = 1)\), however, a decrease of the \(R^2\)-Score over time is observed. No hyperparameters leading to a different convergence behavior for \(t = 1\) were found during optimization. Results from experiments, visualized in Fig. 5, where the current friction-coefficient can be accessed by the agent, indicate that the decrease is related to the partial observability setting (see Section 4.1). A deeper analysis of the data shows that, due to the early process stage and measurement noise, the information content about the friction coefficient in the observable values is very low for \(t = 1\). This causes the \(Q_1\)-function to rely mainly on the previous action \(u_0\), which is part of \(\bar{x}\), and the planned action \(u_1\). In the first episodes, high exploration rates and low quality \(Q\)-functions are leading to very stochastic control decisions, and consequently to almost equally distributed values of \(u_0\) and \(u_1\) in the replay memory. In later episodes, optimal control decisions \(u_0^*\) and \(u_1^*\) (\(u^* = \arg\max_{u \in U} Q(x, u)\)) are increasingly dominant in the replay memory. Due to the low information content in the observable values, \(u_0^*\) and \(u_1^*\) are independent of the friction coefficient. The advantage of the \(Q_1\) model over the simple average on the replay memory is decreasing due to this dominance. Hence, the \(R^2\) score is shrinking. In the fully observable case (Fig. 5), \(u_0^*\) and \(u_1^*\) are dependent on the friction coefficient, causing the \(R^2\) score of the \(Q_1\) model to increase over time.

In order to reveal the effect of the learning parameter variation, experiments were conducted with varying learning rates \(\alpha\) and exploration rates \(\epsilon_0\) to explore their influence on the quality of the online control, reflected by the distribution of the resulting reward. For each parameter combination, 10 independent experimental batches were executed under control of the respectively parameterized reinforcement learning algorithm, each with 2500 deep drawing episodes.

The control quality decreases slightly with increasing exploration rate. The mean reward \(\mu\) achieved in the first 250 episodes varies between \(\mu = 4.69\) for \(\epsilon_0 = 0.4\) and \(\mu = 5.03\) for \(\epsilon_0 = 0.1\). This is due to the negative effect of explorative behavior on the short-term outcome of the process. In later episodes, the control is more robust regarding the actuator noise. The mean reward achieved in the episodes 1250 to 1500 varies between \(\mu = 5.75\) for \(\epsilon_0 = 0.4\) and \(\mu = 5.86\) for \(\epsilon_0 = 0.1\). The mean reward achieved in the last 250 episodes varies between \(\mu = 5.89\) for \(\epsilon_0 = 0.4\) and \(\mu = 5.95\) for \(\epsilon_0 = 0.1\). In the deep drawing process considered, even low exploration rates lead to fast overall convergence, which is not necessarily the case in more complex applications. For the given problem, the measured performance slightly increases with an increasing learning rate. The mean \(\mu\) and standard deviation \(\sigma\) of the reward achieved in the last 250 deep drawing episodes, with a constant exploration rate \(\epsilon_0 = 0.3\), are \((\mu = 5.85, \sigma = 0.69)\) for \(\alpha = 0.3\), \((\mu = 5.87, \sigma = 0.63)\) for \(\alpha = 0.5\), \((\mu = 5.89, \sigma = 0.78)\) for \(\alpha = 0.7\), and \((\mu = 5.90, \sigma = 0.69)\) for \(\alpha = 0.9\), respectively.

Besides learning parameter variation, experiments were made to investigate the effect of the partial process observability. In Fig. 6, the rewards achieved by reinforcement learning-based control of 10 independent executions in three different observability scenarios are visualized. The resulting reward distribution for each scenario is visu-
alized by box plots, each representing a bin of 500 subsequent episodes. In the fully observable scenario (left), the agent has access to the current friction coefficient and thus perfect information about the process state. The partially observable scenario (middle) is equivalent to the scenario described in Section 4.1 and is used in the previously described experiments. The “blind” agent (right) has no access to the process state, and its optimal control is based on the knowledge about previous control actions and rewards only. In the latter scenario, the mean control performance after 1000 episodes is comparable to the baseline.

7. DISCUSSION AND FUTURE WORK

It has been shown how reinforcement learning-based methods can be applied for adaptive optimal control of episodic fixed-horizon manufacturing processes with varying process conditions. A model-free Q-learning-based algorithm has been proposed, which enables the adaptivity to varying process conditions through learning to modify the $Q$-function accordingly. A class of episodic fixed-horizon manufacturing processes has been described, to which the proposed algorithm can be applied. The approach has been instantiated for, and evaluated with, the task of blank holder force optimal control in a deep drawing process. The optimal control goal was the optimization of the internal stresses, of the wall thickness and of the material efficiency for the resulting workpiece. The deep drawing processes used to evaluate the approach were simulated via FEM. The experimental processes were executed in an automatic virtual laboratory environment, which was used to vary the process conditions stochastically and to induce measurement noise. The virtual laboratory is currently specialized to the deep drawing context but will be generalized and published in future work for experimentation and evaluation of optimal control agents on all types of FEM-simulated manufacturing processes.

Contrary to model-based approaches, no prior model is needed when using RL for model-free optimal control. It is therefore applicable in cases where accurate process models are not feasible or not fast enough for online prediction. When applied online, the approach is able to self-optimize, according to the cost function. The approach is able to adapt to instance-specific process conditions and non-stationary outer conditions. In a fully blind scenario, with no additional sensory information, the proposed algorithm reaches the results of non-adaptive model-based methods from model predictive control and approximate dynamic programming in the example use case. The disadvantage of model-free learning approaches, including the proposed reinforcement learning (RL) approach, is the dependence on data gathered during the optimization (exploration). Exploration leads to suboptimal behavior of the control agent in order to learn about the process and to improve future behavior. When used for optimal control of manufacturing processes, RL can lead to increased product rejection rates, which decrease during the learning process. Optimal manufacturing process control with RL is, therefore, most efficient in applications with high production figures.

Extending the proposed approach with methods from transfer learning or multiobjective reinforcement learning could enable information transfer between various process instances, differing e.g., in the process conditions or the production goal, and could thereby lead to more efficient learning, and consequently to decreased rejection rates, and a wider field of application.

We use feedforward neural networks for $Q$-function approximation. Due to the fixed input dimension of feedforward neural networks and the variable dimension of $\bar{x}$, a dedicated approximation model for each control step is required. This complicates the learning process in cases with higher numbers of control steps. To overcome this limitation, recurrent neural networks can be used to map the observable and action history to a fixed dimension surrogate state vector.

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT press, 2018.

[2] S. Tommerup and B. Endelt, “Experimental verification of a deep drawing tool system for adaptive blank holder pres-
sure distribution,” *Journal of Materials Processing Technology*, vol. 212, no. 11, pp. 2529-2540, 2012.

[3] C. P. Singh and G. Agnihotri, “Study of deep drawing process parameters: A review,” *International Journal of Scientific and Research Publications*, vol. 5, no. 1, pp. 2250-3153, 2015.

[4] A. Widi and A. Mosallam, “Some aspects of blankholder force schemes in deep drawing process.” *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 24, no. 1, pp. 315-323, 2007.

[5] R. Bellman, “Dynamic programming.” *Science*, vol. 153, no. 3731, pp. 34-37, 1966.

[6] W. B. Powell, *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

[7] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer, 2007.

[8] L. Grüne and J. Pannek, “Nonlinear model predictive control,” *Nonlinear Model Predictive Control*, Springer, 2017.

[9] A. Alessio and A. Bemporad, “A survey on explicit model predictive control,” in *Nonlinear Model Predictive Control*, pp. 345-369, Springer, 2009.

[10] J. Saint-Donat, N. Bhat, and T. J. McAvoy, “Neural net based model predictive control.” *International Journal of Control*, vol. 54, no. 6, pp. 1453-1468, 1991.

[11] B. M. Åkesson and H. T. Toivonen, “A neural network model predictive controller,” *Journal of Process Control*, vol. 16, no. 9, pp. 937-946, 2006.

[12] M. Senn, N. Link, J. Pollak, and J. H. Lee, “Reducing the computational effort of optimal model controllers for continuous state spaces by using incremental learning and post-decision state formulations.” *Journal of Process Control*, vol. 24, no. 3, pp. 133-143, 2014.

[13] T. Bernard and E. E. Moghaddam, “Nonlinear model predictive control of a glass forming process based on a finite element model.” *Proceedings of the 2006 IEEE International Conference on Control Applications*, pp. 960-965, 2006.

[14] V. Mnih, K. Vukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning.” *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, “Mastering the game of go without human knowledge.” *Nature*, vol. 550, no. 7676, p. 354, 2017.

[16] F.-y. Wang, H. Zhang, and D. Liu, “Adaptive dynamic programming: an introduction.” *IEEE computational intelligence magazine*, vol. 4, no. 2, pp. 39-47, 2009.

[17] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, “Optimal and autonomous control using reinforcement learning: A survey.” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042-2062, 2018.

[18] J. Na, B. Wang, G. Li, S. Zhan, and W. He, “Nonlinear constrained optimal control of wave energy converters with adaptive dynamic programming.” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 10, pp. 7904-7915, Oct. 2019.

[19] S. P. K. Spielberg, R. B. Gopaluni, and P. D. Loewen, “Deep reinforcement learning approaches for process control,” in 6th *International Symposium on Advanced Control of Industrial Processes*, pp. 201-206, IEEE, 2017.

[20] Y. Ma, W. Zhu, M. G. Benton, and J. Romagnoli, “Continuous control of a polymerization system with deep reinforcement learning.” *Journal of Process Control*, vol. 75, pp. 40-47, 2019.

[21] W. Xue, J. Fan, V. G. Lopez, J. Li, Y. Jiang, T. Chai, and F. L. Lewis, “New methods for optimal operational control of industrial processes using reinforcement learning on two time-scales,” *IEEE Transactions on Industrial Informatics*, 2019. DOI: 10.1109/TII.2019.2912018.

[22] Y. Jiang, J. Fan, T. Chai, and F. L. Lewis, “Dual-rate operational optimal control for flotation industrial process with unknown operational model.” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4587-4599, 2019.

[23] D. Görge, “Relations between model predictive control and reinforcement learning.” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4920-4928, 2017.

[24] M. Riedmiller, “Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method.” in *European Conference on Machine Learning*, pp. 317-328, Springer, 2005.

[25] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based pomdp solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1-51, 2013.

[26] L.-J. Lin and T. M. Mitchell, “Reinforcement learning with hidden states.” in *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pp. 271-280, The MIT Press, 1993.

[27] B. Bakker, “Reinforcement learning with long short-term memory.” in *Advances in Neural Information Processing Systems*, vol. 14, pp. 1475-1482, 2002.

[28] B. Endelt, S. Tommerup, and J. Danckert, “A novel feedback control system - controlling the material flow in deep drawing using distributed blank-holder force.” *Journal of Materials Processing Technology*, vol. 213, no. 1, pp. 36-50, 2013.

[29] B. Endelt, “Design strategy for optimal iterative learning control applied on a deep drawing process.” *The International Journal of Advanced Manufacturing Technology*, vol. 88, no. 1-4, pp. 3-18, 2017.

[30] C. J. C. Hawkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[31] K. J. Astrom, “Optimal control of markov processes with incomplete state information.” *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174-205, 1965.

[32] R. M. French, “Catastrophic forgetting in connectionist networks.” *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128-135, 1999.
Johannes Dornheim, Norbert Link, and Peter Gumbsch

[33] Dassault Systèmes. Abaqus 6.14 Scripting User’s Guide. Dassault Systèmes, 2014.

[34] Dassault Systèmes. Abaqus 6.14 Analysis User’s Guide: Analysis, vol. II. Dassault Systèmes, 2014.

[35] J. D. Yoo, S. W. Hwang, and K.-T. Park, “Factors influencing the tensile behavior of a Fe-28Mn-9Al-0.8C steel,” Materials Science and Engineering: A, vol. 508, no. 1-2, pp. 234-240, 2009.

[36] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” Proc. of 28th International Conference on Machine Learning, pp. 265-272, Omnipress, 2011.

Johannes Dornheim is a member of the Intelligent Systems Research Group at the Karlsruhe University of Applied Sciences (KA-UAS) and a Ph.D. candidate at the Karlsruhe Institute of Technology (KIT). He received his M.S. degree in computer science from the KA-UAS in 2014. His research interests include the development and application of self learning adaptive control algorithms in the fields of manufacturing control and material science.

Norbert Link is a professor at the computer science faculty and founder of the Intelligent Systems Research Group of the Karlsruhe University of Applied Sciences (KA-UAS). He received his Ph.D. degree from the Karlsruhe Institute of Technology (KIT) in 1990. Afterwards, he conducted research in computer vision and machine learning at the KA-UAS until 1997. From 1997 to 2000, he was a head of the department for Recognition and Diagnostic Systems of the Fraunhofer IITB research institute. His current research interests include the development and application of machine learning algorithms in the fields of intelligent manufacturing and material science.

Peter Gumbsch is a professor for mechanics of materials at the Karlsruhe Institute of Technology (KIT) and head of the Fraunhofer Institute for Mechanics of Materials IWM. He studied physics and received his doctoral degree in chemistry from the University of Stuttgart in 1991. After a Post-Doc at Imperial College in London and the University of Oxford, he became a group leader at the Max-Planck-Institute for Metals Research in Stuttgart. His research focuses on modeling and simulation of materials, in particular multiscale modeling approaches with current emphasis on the investigation of friction and wear processes.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.