AIAA SciTech Forum
6-10 January 2020, Orlando, FL
AIAA Scitech 2020 Forum

10.2514/6.2020-2100

Safe Curriculum Learning for Optimal Flight Control of Unmanned Aerial Vehicles with Uncertain System Dynamics

T.S.C. Pollack [*], E. van Kampen [†]
Delft University of Technology, 2629HS Delft, The Netherlands

Reinforcement learning (RL) enables the autonomous formation of optimal, adaptive control laws for systems with complex, uncertain dynamics. This process generally requires a learning agent to directly interact with the system in an online fashion. However, if the system is safety-critical, such as an Unmanned Aerial Vehicle (UAV), learning may result in unsafe behavior. Moreover, irrespective of the safety aspect, learning optimal control policies from scratch can be inefficient and therefore time-consuming. In this research, the safe curriculum learning paradigm is proposed to address the problems of learning safety and efficiency simultaneously. Curriculum learning makes the process of learning more tractable, thereby allowing the intelligent agent to learn desired behavior more effectively. This is achieved by presenting the agent with a series of intermediate learning tasks, where the knowledge gained from earlier tasks is used to expedite learning in succeeding tasks of higher complexity. This framework is united with views from safe learning to ensure that safety constraints are adhered to during the learning curriculum. This principle is first investigated in the context of optimal regulation of a generic mass-spring-damper system using neural networks and is subsequently applied in the context of optimal attitude control of a quadrotor UAV with uncertain dynamics.

Nomenclature

C_D, C_L	=	Aerodynamic force coefficients	c	=	Damping coefficient, N/m s^{-1}
C_T	=	Thrust constant	$f(\boldsymbol{x})$	=	Internal state dynamics
C_β, C_{β_m}	=	Blade flapping constants	k	=	Spring coefficient, N/m
C_τ	=	Rotor torque constants	l	=	Rotor arm length, m
C_{τ_D}	=	Rotational drag coefficient	m	=	Input state dimension; mass, kg
$G(\boldsymbol{x})$	=	Control effectiveness matrix	n	=	State space dimension
H	=	Risk perception region	p, q, r	=	Roll, pitch and yaw rate, rad
$\boldsymbol{J}$	=	Inertia matrix, kg m^2	r	=	Observed utility
K	=	Number of policy evaluation iterations	$\boldsymbol{u}_t$	=	Input vector at time t
M	=	Markov decision process	u, v, w	=	Body speed components, m/s
N_c, N_a	=	Number of hidden nodes	$v(\boldsymbol{x})$	=	Control Lyapunov function
Q	=	State weighting matrix	$\boldsymbol{x}_t$	=	State vector at time t
R	=	Input weighting matrix	α	=	Angle of attack, rad
T_1, T_2, T_3, T_4	=	Rotor thrust, N	β	=	Sideslip angle, rad
$V_*(\boldsymbol{x})$	=	Optimal state value function	γ	=	Discount rate
$V_{\tilde{\pi}}(\boldsymbol{x})$	=	State value function	$\boldsymbol{\Delta}(\boldsymbol{x}, \boldsymbol{u})$	=	Bounding model
$W(\boldsymbol{x})$	=	Risk perception function	$\boldsymbol{\epsilon}$	=	Closeness reaching interval
$\mathcal{A}_{map}$	=	Knowledge mapping	λ	=	Step in curriculum sequence
$\tilde{\mathcal{F}}$	=	State transition function	$\pi(\boldsymbol{x})$	=	Control policy
$\mathbb{T}_{ij}$	=	Transformation matrix from frame j to i	$\tilde{\rho}$	=	Performance index
$\mathcal{U}$	=	Input space	ϕ, θ, ψ	=	Euler attitude angles, rad
$\mathcal{X}$	=	State space	$\omega_1, \omega_2, \omega_3, \omega_4$	=	Rotor rotational velocity, rad/s

[*]Graduate student, Control and Simulation Division, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, the Netherlands
[†]Assistant professor, Control and Simulation Division, Faculty of Aerospace Engineering, Kluyverweg 1, 2629HS Delft, the Netherlands

I. Introduction

Automatic flight control system (AFCS) design commonly relies on the availability of accurate model descriptions of the system dynamics, enabling common engineering practice such as computer-aided design [1]. This process calls for an iterative design cycle that typically consists of system identification based on first-order principles, simulation modeling, control law design, ground tests, and eventually flight tests. Consequently, control law design is largely model-based, which calls for extensive verification and validation activities to ensure that the design is coherent with the real, physical system. For newly developed aerial vehicles, the need for high-quality models calls for extensive engineering analysis and judgment that involves computer simulations, wind tunnel measurements, ground tests, and flight tests. This indicates that flight control system design is a complex process that may consume a considerable part of the total development time and costs of an aerial vehicle.

Modern control engineering has developed various techniques that allow for reduced model fidelity requirements. Robust control methods such as H_∞-control provide certain reliability guarantees in terms of closed-loop stability and performance in the face of model uncertainties and disturbances [2–4]. However, although these methods are generally very effective, they can also lead to conservatism in the control system design [5]. Other methods aim at reducing the need for a global model altogether, by adopting a sensor-based approach. These include the incremental counterparts of Nonlinear Dynamic Inversion (NDI) [6] and Backstepping (BS) [7], known as Incremental Nonlinear Dynamic Inversion (INDI) [5] and Incremental Backstepping (IBS) [8], respectively. These methods reduce the impact of model mismatch by directly feeding back acceleration measurements. As a result, the only model terms required for the control law are those related to the control effectiveness. However, this leads to new problems related to the synchronization of sensor measurements, especially when extensive filtering is required.

From this, a case can be postulated for fully autonomous flight control laws that shape themselves online on the real system. Reinforcement Learning (RL) [9–12] is a promising paradigm for generating optimal, adaptive control laws for systems with uncertain dynamics [13]. In the case of inexpensive, small unmanned aerial vehicles (UAVs) for example, RL could be adopted to learn optimal control of the system on-line from scratch. Alternatively, RL methods could be used to train a flight control system off-line based on a crude model description of the system dynamics, to be followed by an on-line fine-tuning phase. This may considerably reduce development time and costs, as knowledge about the system is generated autonomously by the controller itself.

Motivation

Despite its appealing benefits, reinforcement learning comes with its own challenges that currently limit its suitability for deployment on real safety-critical systems. First, large amounts of data are typically required during the learning phase. This is especially true for high-dimensional state and input spaces, which are typical for aerospace systems. Curriculum learning [14, 15] is a machine learning paradigm that specifically deals with the data dependency issue by making the learning process more tractable. This can for example be done by breaking down the complexity of a learning task and training the agent first on a range of related tasks that are significantly easier to learn. This concept exhibits strong analogies with human education, where students are presented with easier material first before moving on to more complex subjects. This principle can be leveraged for autonomous shaping of flight control laws, by subjecting a learning agent to an appropriate task curriculum that allows it to gradually expand its capabilities.

A second major challenge in RL is how to perform online training safely, i.e. how to let the agent learn autonomously without damaging itself or its surroundings. This idea has given rise to another major branch of RL research known as Safe Learning (SL) [16]. A large number of methods has been proposed in this field, including safety filters [17, 18], imposing initial domain knowledge [19], and action limiting [20]. However, little work has been performed that unite the concepts of curriculum learning and safe learning, whereas this may be of large benefit to the applicability of RL on safety-critical systems such as UAVs. Therefore, the contribution of this research is to demonstrate how effective learning curricula can be constructed for a reinforcement learning agent to learn optimal control of an aerial vehicle in a safe manner.

Related work

Although safe curriculum learning for flight control is still a largely unexplored research area, other (related) paradigms have been investigated that aim to address the said safety and efficiency issues that undermine online reinforcement learning applications. Recently, Helmer et al. [21] proposed the framework of flexible heuristic dynamic programming (HDP), and showed how decomposing a target task into a pair of successive tasks of lower complexity can expedite learning in the context of flight control. The concept of flexible function approximation can be regarded as a

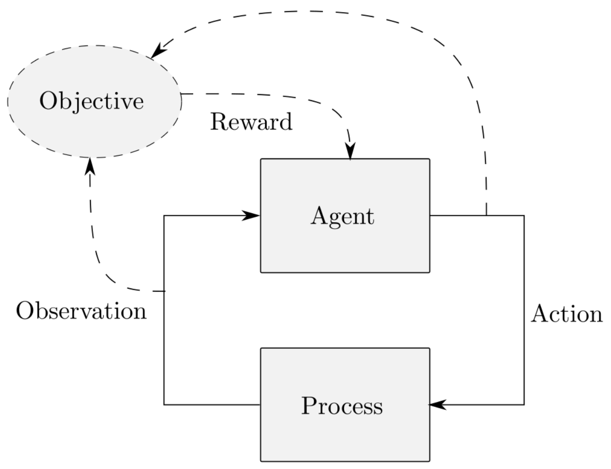


Fig. 1 The sequential decision-making problem. Adapted from [10].

form of transfer learning (TL), which is a branch of RL research that specifically deals with the question of how the large data dependency of the reinforcement learning paradigm can be reduced by transferring knowledge obtained from training on one task to another task in which learning has not yet taken place [22]. Transfer learning takes a key position in curriculum learning for RL [15]. Consequently, the study by Helmer et al. [21] can be regarded as work in curriculum learning, although the authors did not explicitly recognize it as such. Safety during learning was not considered.

Outline

This article is structured as follows. In Section II, the safe curriculum learning paradigm is substantiated by briefly covering the necessary background on reinforcement learning, safe learning, and curriculum learning. This is followed by a description of a safe curriculum approach to optimal control of discrete-time systems in Section III. The resulting framework is tested in a simple experiment based on a linear, unstable, cascaded mass-spring-damper (MSD) system in Section IV, where its performance is compared to other non-curriculum approaches. Subsequently, a flight control application focusing on optimal attitude control of a Parrot quadrotor Unmanned Aerial Vehicle (UAV) is described in Section V. Finally, conclusions and recommendations for further research are drawn in Section VI.

II. Fundamentals

Demonstrating how safe and effective machine learning curricula can be constructed requires an adequate substantiation of the safe curriculum learning paradigm. In this section, a general theoretical framework is formulated that describes in high-level terms the fundamental concepts and requirements for safe curriculum learning. To this end, Section II.A provides a very brief introduction to the fundamentals of reinforcement learning and adaptive/approximate dynamic programming. This is followed by a short description of curriculum learning in Section II.B, based on existing views reported in the literature and the authors' interpretations. In Section II.C, key concepts in the area of safe learning are discussed. In Section II.D, these views are projected on the curriculum learning paradigm to establish a safe curriculum learning framework.

A. Reinforcement Learning and Adaptive/Approximate Dynamic Programming

Generally, RL can be applied to solve sequential decision-making problems (SDMP) formulated as Markov Decision Processes (MDPs) in the absence of an explicit model description of the system [10]. SDMPs are often regarded as discrete-time formulations of the optimal control problem and can be visualized as shown in Figure 1. In this framework, a decision-making entity repeatedly interacts with a certain process or environment described by a state distribution function (dynamics) $\tilde{F} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \mapsto [0, 1]$, where it receives a scalar reward or utility $R_{t+1} = r$ after each transition based on a certain objective captured by a (possibly stochastic) reward function or performance index $\tilde{\rho} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \mapsto \mathbb{R}$ [10, 12]. Here, $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ represent the state and input spaces associated with the MDP. In brief, an MDP can be concisely referred to as a tuple $M : \langle \mathcal{X}, \mathcal{U}, \tilde{F}, \tilde{\rho}, \gamma \rangle$. In the field of artificial intelligence, the decision-making entity is referred to as the agent, whereas in control theory inputs are generated by a controller consisting of a set of control laws. RL methods can directly operate on data obtained from the system to learn the (near-)optimal decision sequence for the problem at hand [12]. Ultimately, the goal of the agent is to maximize or minimize the reward sequence from every state $X_t \in \mathcal{X}$, denoted as the return. In discrete-time, the return can be formulated as the algebraic sum of total collected utility, discounted at every time step by a discount rate $0 \leq \gamma \leq 1$ [10]:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

The goal of maximizing (or minimizing) the discounted return G_t can be attained by the agent through an appropriate sequence of control actions that together constitute an optimal trajectory. Such a decision sequence is captured by a policy (control law) $\tilde{\pi} : \mathcal{X} \mapsto \mathcal{U}$. The expected return by following a control policy $\tilde{\pi}(\boldsymbol{u}|\boldsymbol{x})$ from a given state $X_t = \boldsymbol{x}$ gives rise to the concept of value functions, defined as [10]:

$$V_{\tilde{\pi}}(\boldsymbol{x}) \doteq \mathbb{E}_{\tilde{\pi}}\left[G_t | X_t = \boldsymbol{x}\right] = \mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | X_t = \boldsymbol{x}\right] \tag{2}$$

This expression is often rewritten in recursive form to make the process of evaluating states and policies more tractable. This leads to the Bellman expectation equation [10]:

$$V_{\tilde{\pi}}(\boldsymbol{x}) = \mathbb{E}_{\tilde{\pi}}\left[R_{t+1} + \gamma V_{\tilde{\pi}}(X_{t+1}) | X_t = \boldsymbol{x}\right] \tag{3}$$

The value function of a given policy $\tilde{\pi}$ can be used to find new, better policies. The concept of value functions can also be applied to state-input pairs, which enables direct evaluation of any input $U_t = \boldsymbol{u}$ given the current state $X_t = \boldsymbol{x}$ and following the policy $\tilde{\pi}(\boldsymbol{u}|\boldsymbol{x})$ thereafter [10, 12]. The iterative process of value function prediction and policy improvement eventually converges to the optimal policy, which maximizes the expected return for every state $X_t \in \mathcal{X}$ [10]:

$$\pi_*(\boldsymbol{u}|\boldsymbol{x}) = 1 \iff \boldsymbol{u} = \arg\max_{\boldsymbol{u}} \mathbb{E}\left[R_{t+1} + \gamma V_*(X_{t+1}) | X_t = \boldsymbol{x}\right] \tag{4}$$

Once the optimal policy has been found, the MDP has been solved. In general, there are many different approaches to solving Equation 4. If the system dynamics $\tilde{F}$ and reward function $\tilde{\rho}$ are fully known, Dynamic Programming (DP) techniques such as Value Iteration (VI) and Policy Iteration (PI) can be used [10, 12]. In case a priori knowledge about the dynamics is not available, these methods are no longer applicable and one has to use data-driven RL techniques such as Monte Carlo (MC) learning or temporal-difference (TD) methods [10]. A central topic underlying these data-driven approaches is the careful trade-off between the exploitation of intermediate policies to further improve learned behavior, and exploration to ensure that all parts of the state space are visited sufficiently often. Within the context of optimal adaptive control, Adaptive/Approximate Dynamic Programming (ADP) [23, 24] techniques are often adopted, due to the general need to approximate the value function and policy. In this view, Adaptive Critic Designs (ACDs) [25, 26] are frequently used to enable data-driven identification of the optimal control policy.

B. Curriculum Learning

The ability of an RL algorithm to converge towards an optimal control policy for a given MDP is naturally related to the complexity of the task at hand [15]. Task complexity covers multiple aspects of a task, such as dimensionality of the state space $\mathcal{X} \subset \mathbb{R}^n$ and input space $\mathcal{U} \subset \mathbb{R}^m$, the nature of the dynamics $\tilde{F}$, or richness of the reward signal. However, a key insight is that task complexity does not exclusively define learning performance, but that there is a strong link with the learning abilities of the agent itself. This understanding gives rise to the idea of adjusting the complexity of a given learning task to the agent's competencies, for it to learn the desired behavior more effectively [15, 22]. In this way, a sequence of intermediate training distributions can be constructed to increase the tractability of the learning process. This concept is used frequently in studies on animal training, where it is known as 'shaping' [14, 27]. In machine learning, the shaping sequence is referred to as a learning curriculum [15].

The concept of curriculum learning appears in various forms in a range of different fields. A key contribution was made by Bengio et al. [14] in the area of supervised learning with deep neural networks. Here, the authors successfully demonstrated effective learning curricula for shape recognition and language modeling. The proposed framework is based on the idea that for non-convex optimization tasks, the order in which training samples are presented can be used to avoid local minima [14, 28]. It is argued that by presenting the learner with an appropriate sequence of samples, the

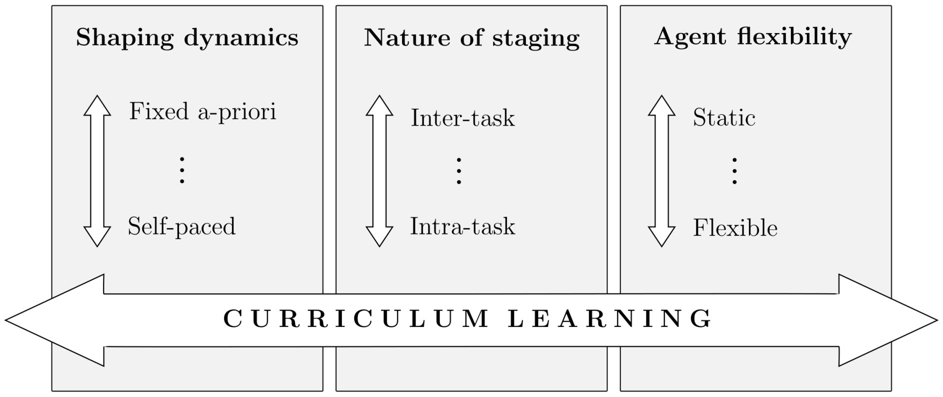


Fig. 2 Visual representation of the curriculum learning taxonomy

solution is drawn towards a region of the parameter space that is in the vicinity of the global optimum. This effectively results in a 'smoothing' of the non-convex optimization criterion. Additionally, the curriculum learning framework was also conceived in the field of transfer learning (TL) [15], which advocates that knowledge learned in one MDP can be applied in a different, but related MDP [22]. This concept can be extended to a learning sequence, where in each stage knowledge is transferred between a pair of tasks. This extends the motivation for curriculum learning beyond non-convex optimization problems. Moreover, it implies that the shaping aspect of a learning curriculum is not limited to the training distribution only, but also resides in the agent representation. This interpretation of curriculum learning can be recognized in the work by Helmer et al. [21] on flexible heuristic dynamic programming.

Although the motivations for applying curriculum learning for supervised learning or transfer learning may be different, they share a clear common philosophy. In the authors' view, the curriculum learning paradigm can be decomposed along three dimensions, as shown in Figure 2. The first element is referred to as the *shaping dynamics*. In general, a learning curriculum can be fully determined a priori [14], or may arise from online feedback of the capabilities of the agent during the learning process [28, 29]. Curriculum learning of the first form is based on a fixed sequence of training distributions selected heuristically by a domain expert. This is an intuitive way of curriculum design and allows a domain expert to embed prior knowledge in the learning process. However, such a ranking of training distributions can be difficult to provide a priori if it is not clear what determines the learning complexity. Moreover, feedback about learning progress is not used. This gives rise to the philosophy of letting a learning curriculum be generated dynamically by the learner itself [28] or some agent-specific teacher [30]. This concept is known as *self-paced* learning (SPL) [28, 29]. However, SPL suffers from other problems that need to be adequately addressed.

The second dimension will be called the *nature of staging*. The main categories considered here are designated as *inter-task* and *intra-task* learning, referring to the notion of learning within or across a (set of) MDP(s). In intra-task learning, the system state transition function (dynamics) of the target MDP $M : \langle \mathcal{X}, \mathcal{U}, \tilde{F}, \tilde{\rho}, \gamma \rangle$ is shared by any of the intermediate MDPs $M_\lambda : \langle \mathcal{X}_\lambda, \mathcal{U}_\lambda, \tilde{F}, \tilde{\rho}_\lambda, \gamma_\lambda \rangle$ that comprise the learning curriculum. This implies that complexity management is limited to strict subsets of the target state and input space and designating alternative forms of the performance index. In case the dimensionality of the state space is used as a shaping factor, an intra-task learning curriculum requires the dynamic modes associated with these states to be fully decoupled. This contrasts with inter-task learning, where the state transition function is allowed to change across the shaping sequence. Effective inter-task learning requires a high degree of correlation across tasks to ensure a positive learning effect.

The third aspect curriculum learning mentioned here relates to the flexibility of the agent domain, and will be referred to as the *agent flexibility*. In broad terms, one can make a distinction between static and dynamic agent representations. With dynamic agent representations, one can opt to apply a mapping or to switch altogether between different representations. This can relate to different types of information, such as the value function, the policy, or an internal model. Depending on the type of function approximator, dedicated techniques are required to achieve effective knowledge transfer. This is again a major topic in the field of transfer learning [22, 31].

C. Safe Learning

The application of RL in its basic form is troublesome if the process or system that is to be interacted with is safety-critical in nature. As argued by García and Fernández [16], the problem of safety may appear in both the exploitation and exploration phases of learning. The objective of maximizing or minimizing the expected discounted return may not be separable from occasional unsafe or risky transitions for some environments, whereas the random element that underlies the exploration phase may inadvertently lead the system to a dangerous state (i.e., one for which the probability of recovery equals zero). Therefore, dedicated measures are required [16, 18, 19, 32–36].

Independent of the nature of control policy, online autonomous interaction with safety-critical systems requires measures to keep the system state $\boldsymbol{x}_t$ bounded to a safe subset of the complete state space $\mathcal{X}_{safe} \subset \mathcal{X}\ \forall\ t \in [t_0, \infty\rangle$. In systems theory, a topic that captures this objective very effectively is known as Lyapunov stability [6, 37]. In Lyapunov stable systems, any trajectory starting from a state $\boldsymbol{x}_t$ remains bounded to the safe set $\mathcal{X}_{safe}$ if there exists a scalar continuous function known as a (control) Lyapunov function $v(\boldsymbol{x}_t)$ over the domain spanned by $\mathcal{X}_{safe}$. For systems that are stabilizable but do not possess Lyapunov stable internal dynamics, a stabilizing policy can be derived directly from a known control Lyapunov function (CLF) by rendering its derivative negative semi-definite. This is a very useful property that enables the design of safe control policies for both linear and nonlinear systems [6, 37].

Lyapunov stability in its basic form does not consider state- or input-constrained systems or systems with bounded uncertainties or disturbances. Nevertheless, the concept can also be extended to this class of systems. Dedicated CLFs can be found that prevent constraint violation or ensure robust stabilizability. For example, one can adopt Barrier Lyapunov functions [38, 39] or Robust CLFs [37] to design control policies that meet these objectives. This shows that, in theory, the issue of safety during learning can be completely resolved once an appropriate CLF has been obtained.

However, despite their appealing properties, Lyapunov functions may not be straightforward to find for general nonlinear systems [6]. If not available, there are no guarantees that the system state remains bounded to the safe subspace $\mathcal{X}_{safe}$. In the absence of Lyapunov functions, alternative techniques are required that provide guarantees on the existence of a control strategy that ensures boundedness of state- and input-constrained systems. A central topic here is known as ergodicity [34]. If the condition of ergodicity applies, it is with certainty that some trajectory exists between any arbitrary pair of states in the safe subspace $\mathcal{X}_{safe}$. Ensuring ergodicity requires an internal belief, or model, of the system dynamics, as well as sensing capabilities for detecting surrounding states [17, 34]. The former makes the process inherently model-based. Consequently, instead of starting with an initial admissible policy or initial value function in the form of a CLF, safety can be guaranteed by virtue of a prior model only. This model can feature bounded uncertainties, as long as the actual system dynamics are within these bounds [17, 18]. Moreover, the prior model may be updated online to enable more effective verification of the ergodicity condition.

Although the availability of an internal model is key to the ergodicity argument, the question remains how verification of ergodicity can be achieved. One approach is to find a global backup policy that maximizes the safe subspace $\mathcal{X}_{safe}$ under the belief [34]. From a control-theoretic perspective, it can be argued that the key principles underlying this concept show correspondences to robust stabilizability. Another method is to find local backup policies that render a given state to the vicinity of another state that has been visited before. This requires the online generation of trajectories based on the internal model. A heuristic approach to this technique is presented by Mannucci et al. as the Safety Handling Exploration with Risk Perception Algorithm (SHERPA) [17, 18].

D. Safe Curriculum Learning

Considering the curriculum learning paradigm, the fundamental concepts of (robust) Lyapunov stability and ergodicity retain a central position. Under minimum robustness requirements, a learning curriculum can be exploited to enhance learning safety in a similar manner as it improves learning efficiency. That is, for any intermediate MDP $M_\lambda : \langle \mathcal{X}_\lambda, \mathcal{U}_\lambda, \tilde{F}_\lambda, \tilde{\rho}_\lambda, \gamma_\lambda \rangle$ in the shaping sequence, boundedness of the system state $\boldsymbol{x}_t$ to a safe subset $\mathcal{X}^{(\lambda)}_{safe} \subset \mathcal{X}_\lambda\ \forall\ t \in [t_0, \infty\rangle$ can be guaranteed if a safe control policy $\pi^{(\lambda-1)}_{safe}(\boldsymbol{x}_t)$ can be safely re-used from a predecessor MDP $M_{\lambda-1} : \langle \mathcal{X}_{\lambda-1}, \mathcal{U}_{\lambda-1}, \tilde{F}_{\lambda-1}, \tilde{\rho}_{\lambda-1}, \gamma_{\lambda-1} \rangle$. Of central importance are the risk profile and system dynamics across the shaping sequence. For true safety, the complete risk profile should always be available at any stage in the curriculum. That is, if it is not specified a priori that $\left\{\mathcal{X}^{\lambda}_{safe}\right\}^{\complement} \neq \left\{\mathcal{X}^{\lambda-1}_{safe}\right\}^{\complement}$, safety cannot be guaranteed when solving M_λ.

The central concept can best be described from a control theoretic perspective. For example, an agent policy $\pi_{\lambda-1}(\boldsymbol{x}_t)$ that stabilizes the system $\boldsymbol{x}_{t+1} = f_{\lambda-1}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ in $\mathcal{X}^{\lambda-1}_{safe}$ can be safely re-used if the successor system $\boldsymbol{x}_{t+1} = f_\lambda(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is also Lyapunov stable in $\mathcal{X}^{\lambda}_{safe}$ under autonomy of (a transformed form of) $\pi_{\lambda-1}(\boldsymbol{x}_t)$. The question of whether a transformation of $\pi_{\lambda-1}(\boldsymbol{x}_t)$ is required depends on the relative topology and dimensionality of the state and input

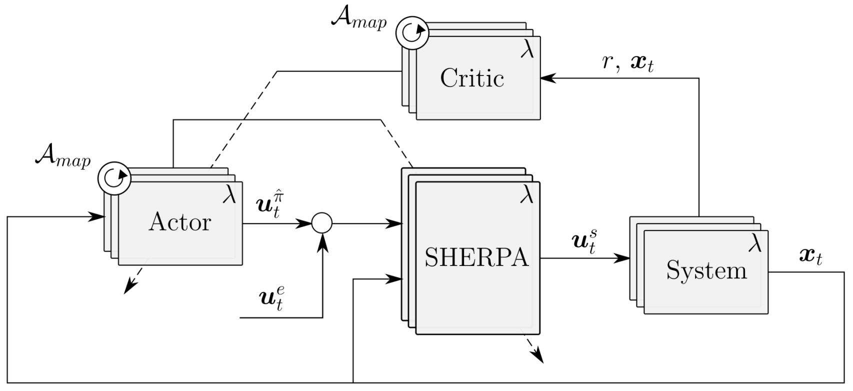


Fig. 3 Block diagram of the safe curriculum optimal control framework

spaces: in case $\mathcal{X}_\lambda \neq \mathcal{X}_{\lambda-1}$ or $\mathcal{U}_\lambda \neq \mathcal{U}_{\lambda-1}$, safe re-use of $\pi_{\lambda-1}(\boldsymbol{x}_t)$ requires an appropriate mapping $\mathcal{A}_{map}$ of $\pi_{\lambda-1} : \mathcal{X}_{\lambda-1} \mapsto \mathcal{U}_{\lambda-1}$ to $\pi_\lambda : \mathcal{X}_\lambda \mapsto \mathcal{U}_\lambda$ [22, 31, 40]. This directly relates to the agent flexibility aspect of safe curriculum learning. In general, construction of a safe $\mathcal{A}_{map}$ requires an additional source of domain knowledge.

The same elemental considerations apply to the verification of ergodicity. However, the internal model that is used for verification should be representative for every stage in the MDP. Theoretically, the availability of this model implies that safe learning in MDP M_λ can be fully achieved without a prior safe control policy $\pi_{safe}^{(\lambda-1)}(\boldsymbol{x}_t)$. However, this prior policy can still be exploited to expedite the search for ergodicity-preserving policies. This is particularly beneficial for online identification of control sequences between a given state and a state visited earlier if the subspace of ergodicity-preserving policies is relatively small compared to the total policy space. It can be argued that the latter serves as a measure of learning complexity in the context of safety. In this case, (a mapped variant of) $\pi_{safe}^{(\lambda-1)}(\boldsymbol{x}_t)$ can serve as a means to bias the search for backup policies. This again stems from the insight that for autonomous dynamic systems, ergodicity is very much in line with the concept of robust Lyapunov stability.

III. Methodology

In order to demonstrate the safe curriculum paradigm in the context of optimal control, several instruments are required. The schematic illustrated in Figure 3 shows a generic picture of the safe curriculum control framework considered in this study. A fixed learning curriculum consisting of a sequence of MDPs $M_\lambda : \langle \mathcal{X}_\lambda, \mathcal{U}_\lambda, f_\lambda, r_\lambda, \gamma_\lambda \rangle$ describing a collection of dynamic (sub)-systems and utility functions is considered. An intelligent flexible learning agent represented by a collection of feedforward neural network control policies takes a central role in the learning architecture. A collection of critic networks are used to approximate the value function. Because of the flexible nature of the agent and the changing dimensionality of $\mathcal{X}_\lambda$ and $\mathcal{U}_\lambda$ across the curriculum, an a priori designed mapping strategy $\mathcal{A}_{map}$ is adopted to transfer the information encoded in the actor and critic networks across successive learning stages. The learning architecture is completed by the SHERPA ergodicity-preserving safety filter [17, 18], which overrides any inputs under which the ergodicity condition cannot be maintained.

The approximate dynamic programming approach used to train the agent is presented in Section III.A, which is followed by a description of the neural-network-based actor-critic setup in Section III.B. The subject of task network mapping strategies is considered in Section III.C. Finally, a basic overview of the SHERPA safety filter is provided in Section III.D, together with a description of how the learning curriculum is exploited to improve its performance.

A. Learning Framework

In general, ADP schemes can be applied to find optimal control policies for a wide class of dynamical systems, including those that are time-varying, nonlinear, or stochastic in nature [23]. Here the scope will be limited to time-invariant, deterministic, control-affine, nonlinear systems of the following form:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) + G(\boldsymbol{x}_t)\boldsymbol{u}_t \tag{5}$$

The discrete-time formulation of the system dynamics makes the optimal control problem consistent with the digital control framework. Under the assumption that the system is Lipschitz continuous and that is it stabilizable, an optimal feedback policy $\hat{\pi}_*(\boldsymbol{x})$ can be found that minimizes the expected (discounted) utility accumulated over time [23, 41]. In this research, the scope will be limited to optimal regulation, which implies that the goal is to bring all states of the system to zero such that the performance index is optimized. However, the problem formulation can be readily extended to tracking control as well. In general, the performance index can take any shape, but it is often defined as a utility that is quadratic in the state and input variables [23, 42]:

$$r(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{x}_t^T Q \boldsymbol{x}_t + \boldsymbol{u}_t^T R \boldsymbol{u}_t \tag{6}$$

A similar utility can be specified based on observed outputs. Since $\lim_{t\to\infty} r(\boldsymbol{x}_t, \boldsymbol{u}_t) = 0$ under an admissible control policy, the following (undiscounted) definition of the value function applies [41, 42]:

$$V_\pi(\boldsymbol{x}) \doteq \sum_{t=0}^{\infty} r(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{7}$$

In this case, no discounting of the expected future return is required, i.e. $\gamma = 1$. Consequently, the optimal value function can be written as follows:

$$V_*(\boldsymbol{x}_t) = \min_{\boldsymbol{u}} \left[r(\boldsymbol{x}_t, \boldsymbol{u}_t) + V_*(\boldsymbol{x}_{t+1}) \right] \tag{8}$$

In optimal control theory and ADP, this equation is also known as the discrete-time Hamilton-Jacobi-Bellman (HJB) equation [41, 42]. Subsequently, the optimal control policy $\pi_*(\boldsymbol{x})$ can be derived directly from the discrete-time formulation of the system dynamics. Setting $\frac{\partial V_*(\boldsymbol{x}_t)}{\partial \boldsymbol{u}_t} = 0$ and expanding the partial derivative using the chain rule, the following is obtained [42, 43]:

$$\boldsymbol{u}^*(\boldsymbol{x}_t) = -\frac{1}{2} R^{-1} G^T(\boldsymbol{x}_t) \frac{\partial V_*(\boldsymbol{x}_{t+1})}{\partial \boldsymbol{x}_{t+1}} \tag{9}$$

The discrete-time HJB equation is solved online through Heuristic Dynamic Programming (HDP) based on Generalized Policy Iteration (GPI). In GPI, the policy evaluation step only consists of a finite number of K iterations towards the true value function $V_\pi(\boldsymbol{x})$ [42]. This makes it a compromise between Value Iteration (VI) and Policy Iteration (PI), for which $K = 1$ and $K \to \infty$, respectively. Whereas GPI is less sample-efficient compared to PI, this comes at the benefit of not needing an initial admissible control policy. Consequently, for $k = 0, 1, \ldots, K-1$, the policy evaluation step takes the following form:

$$\hat{V}_{\pi_j}^{(k+1)}(\boldsymbol{x}_t) = r(\boldsymbol{x}_t, \boldsymbol{u}_t) + \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_{t+1}) \tag{10}$$

Here, $\hat{V}_{\pi_j}^{(\bullet)}(\boldsymbol{x}_t)$ represents a compact approximation of the true intermediate value function in the form of a critic network. The policy update step follows directly from Equation 9:

$$\boldsymbol{u}^{(j+1)}(\boldsymbol{x}_t) = -\frac{1}{2} R^{-1} G^T(\boldsymbol{x}_t) \frac{\partial \hat{V}_{\pi_j}^{(K)}(\boldsymbol{x}_{t+1})}{\partial \boldsymbol{x}_{t+1}} \tag{11}$$

In principle, explicit evaluation of this equation yields the optimal control directly without the need for a separate actor network. However, this requires a prediction of the future state $\boldsymbol{x}_{t+1}$ based on a full model of the system dynamics. A separate representation of the control policy omits this requirement, as proposed by [43–45], but the policy update step is still dependent on complete knowledge of the input dynamics $G(\boldsymbol{x}_t)$. Although this term could be learned from online interaction as well [46, 47], in this work it is assumed that this term is available a priori.

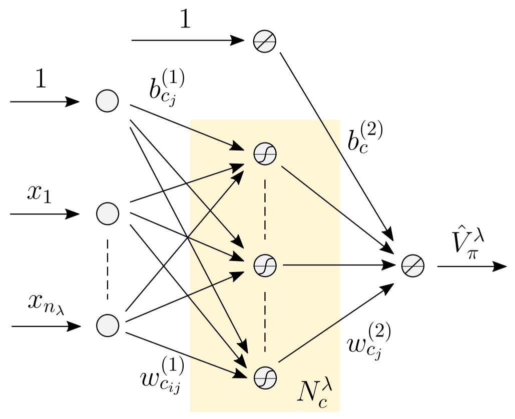
B. Actor-Critic Design and Training

The actor and critic have both been modeled as two-layer feedforward artificial neural networks (ANNs) with a single hidden layer and additional bias nodes. As illustrated in Figure 4a, the critic takes the system state $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^{n_\lambda}$ and returns an approximate estimate of the value function $\hat{V}_\pi^{(k)}(\boldsymbol{x}_t)$ based on a forward pass through the network:

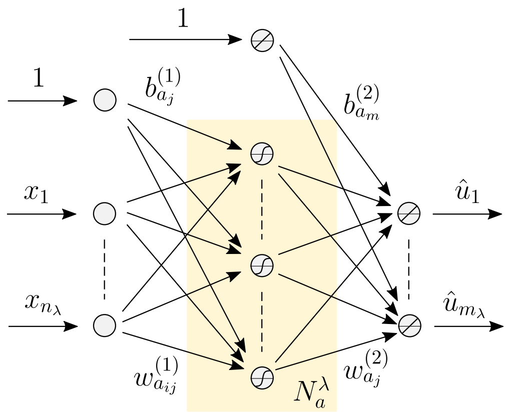
$$\hat{V}_\pi^{(k)}(\boldsymbol{x}_t) = \sum_{j=1}^{N_c^\lambda} w_{c_j}^{(2)}(k) \phi_{c_j} \left(\sum_{i=1}^{n_\lambda} w_{c_{ij}}^{(1)}(k) x_i + b_{c_j}^{(1)}(k) \right) + b_c^{(2)}(k) \tag{12}$$

Similarly, the actor network maps the system state $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^{n_\lambda}$ towards the greedy control input $\pi(\boldsymbol{x}_t) \in \mathcal{U} \subset \mathbb{R}^{m_\lambda}$:

$$\hat{u}_m^{(k)}(\boldsymbol{x}_t) = \sum_{j=1}^{N_a^\lambda} w_{a_{mj}}^{(2)}(k) \phi_{a_j} \left(\sum_{i=1}^{n_\lambda} w_{a_{ij}}^{(1)}(k) x_i + b_{a_j}^{(1)}(k) \right) + b_{a_m}^{(2)}(k) \tag{13}$$



(a) Critic



(b) Actor

Fig. 4 Generic neural network architectures at learning stage λ

The generic architecture of the actor network is also visualized in Figure 4b. For both the actor and critic networks, hyperbolic tangent functions $\phi_{c_j}(\bullet) = \phi_{a_j}(\bullet) = \tanh(\bullet)$ have been selected as the activation functions in the hidden layer. This type of functions is very suitable for approximating the value function and policy, thanks to their global character and the fact that they are continuously differentiable. The critic is trained through bootstrapping by minimizing the Bellmann or temporal-difference (TD) error [42, 45]:

$$e_c^{(k)}(\boldsymbol{x}_t) = \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_t) - r(\boldsymbol{x}_t, \boldsymbol{u}_t) - \hat{V}_{\pi_j}^{(k-1)}(\boldsymbol{x}_{t+1}) \tag{14}$$

Adjustment of the critic weights is achieved through backpropagation based on the Levenberg-Marquadt gradient descent algorithm, which generally results in much faster convergence compared to first-order optimization methods [23]. The loss function is taken as the quadratic TD-error:

$$E_c^{(k)}(\boldsymbol{x}_t) = \tfrac{1}{2} e_c^{(k)}(\boldsymbol{x}_t)^2 \tag{15}$$

The gradient for each weight can be computed analytically as follows:

$$\frac{\partial E_c^{(k)}(\boldsymbol{x}_t)}{\partial w_{c_j}^{(2)}(k)} = \frac{\partial E_c^{(k)}(\boldsymbol{x}_t)}{\partial e_c^{(k)}(\boldsymbol{x}_t)} \frac{\partial e_c^{(k)}(\boldsymbol{x}_t)}{\partial \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_t)} \frac{\partial \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_t)}{\partial w_{c_j}^{(2)}(k)} = e_c^{(k)}(\boldsymbol{x}_t) \tanh\left(\sum_{i=1}^{n_\lambda} w_{c_{ij}}^{(1)}(k) x_i + b_{c_j}^{(1)}(k)\right) \tag{16}$$

$$\frac{\partial E_c^{(k)}(\boldsymbol{x}_t)}{\partial w_{c_{ij}}^{(1)}(k)} = \frac{\partial E_c^{(k)}(\boldsymbol{x}_t)}{\partial e_c^{(k)}(\boldsymbol{x}_t)} \frac{\partial e_c^{(k)}(\boldsymbol{x}_t)}{\partial \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_t)} \frac{\partial \hat{V}_{\pi_j}^{(k)}(\boldsymbol{x}_t)}{\partial \phi_{c_j}(k)} \frac{\partial \phi_{c_j}(k)}{\partial w_{c_{ij}}^{(1)}(k)} = e_c^{(k)}(\boldsymbol{x}_t) w_{c_j}^{(2)}(k) \left(1 - \tanh^2\left(\sum_{i=1}^{n_\lambda} w_{c_{ij}}^{(1)}(k) x_i + b_{c_j}^{(1)}(k)\right)\right) x_i \tag{17}$$

A similar derivation applies to the bias terms. The actor network is trained to approximate the greedy input from Equation 11, as proposed by [23, 45]

$$\boldsymbol{e}_a^{(j+1)}(\boldsymbol{x}_t) = \boldsymbol{u}^{(j+1)}(\boldsymbol{x}_t) - \hat{\boldsymbol{u}}^{(j+1)}(\boldsymbol{x}_t) \tag{18}$$

where:

$$\boldsymbol{u}^{(j+1)}(\boldsymbol{x}_t) = -\frac{1}{2} R^{-1} G^T(\boldsymbol{x}_t) W_c^{(2)}(K) \operatorname{diag}\left(1 - \tanh^2\left(W_c^{(1)}(K)\boldsymbol{x}_{t+1} + \boldsymbol{b}_c^{(1)}(K)\right)\right) W_c^{(1)}(K) \tag{19}$$

with $W_c^{(1)}(K)$, $W_c^{(2)}(K)$, and $\boldsymbol{b}_c^{(1)}(K)$ representing the network weights and bias from policy evaluation step K in matrix and vector form, respectively. The loss function is defined as the square of this difference:

$$E_a^{(j+1)}(\boldsymbol{x}_t) = \tfrac{1}{2} \boldsymbol{e}_a^{(j+1)}(\boldsymbol{x}_t)^2 \tag{20}$$

The gradients can be derived analogously as for the critic. Training is performed in batch format for every intermediate policy $\pi_j(\boldsymbol{x}_t)$ based on a collection of visited states $\mathcal{X}_{visit_{\pi_j}}^\lambda \subset \mathcal{X}_{safe}^\lambda \subset \mathcal{X}^\lambda$. This is done after online interaction with the system. This implies that training is not performed online, but in-between online exploitation cycles.

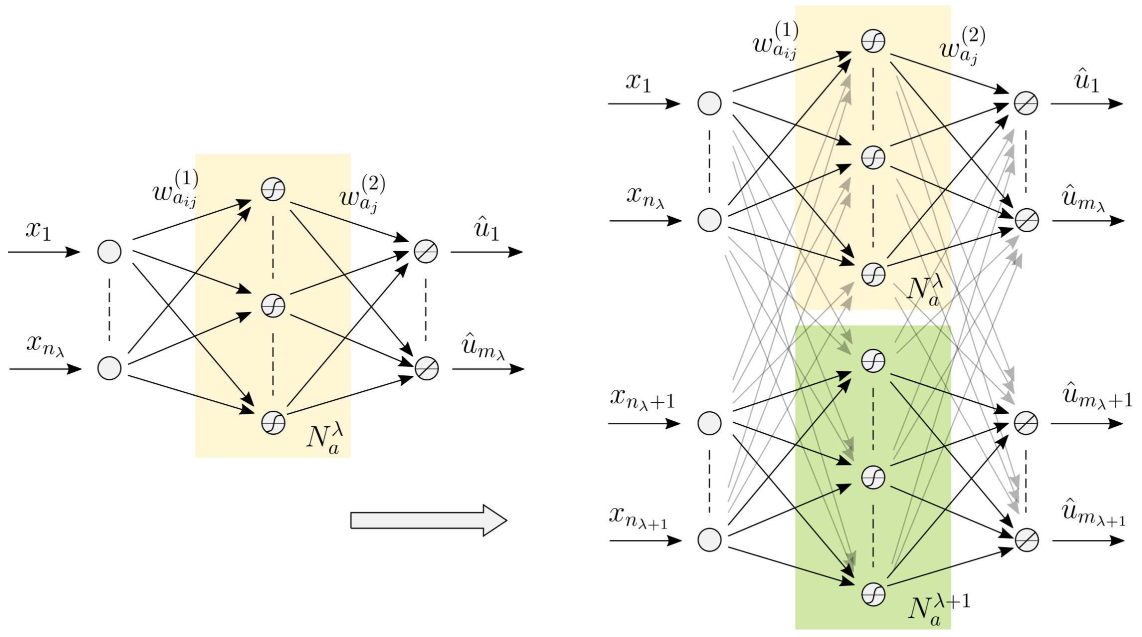


Fig. 5 Greedy network mapping strategy for the actor network; bias nodes not shown for clarity

C. Task Network Mappings

The actor and critic networks represent valuable knowledge that is to be built upon as the learning curriculum progresses. Depending on the relative topology, dimensionality, and semantics of the related MDPs, appropriate mappings $\mathcal{A}_{map}$ need to be applied to transfer this knowledge across successive learning stages. This concept takes a central position in the field of transfer learning, and was introduced by Taylor et al. [31] in the domain of reinforcement learning. Inspired by this work, task network mappings are adopted in the proposed safe curriculum control setup to map the ANN weights and biases between successive representations of the actor and the critic.

As explained in Section II.D, a primary objective for these mappings is that the dynamic system is stable under autonomy of the mapped policy $\hat{\pi}_0^{(\lambda)}(\boldsymbol{x_t})$. This implies that from the perspective of safety, any mapping can be applied that ensures the absence of unstable modes. In the case of naturally stable systems, this means that the zero mapping is sufficient to satisfy this condition. A secondary objective is that learning efficiency should be enhanced, which opens many possibilities for designing $\mathcal{A}_{map}$ [21, 31]. An intuitive approach is to directly copy weights to semantically similar network links, which is designated as *greedy* mapping. For example, in the case of the actor, this is in line with the idea that similar policies should be adopted for any new states $\boldsymbol{x}_t' = \text{proj}_{\mathcal{X}^{\lambda+1}\backslash\mathcal{X}^{\lambda}}(\boldsymbol{x}_t)$ that are governed by similar dynamics as any states counteracted before. For the hidden-layer weight matrix $W_a^{(1)}(\bullet)$, this implies that the following operation is applied:

$$\begin{bmatrix} w^{(1)}_{a11} & \cdots & w^{(1)}_{a1n^\lambda} \\ \vdots & \ddots & \vdots \\ w^{(1)}_{aN_a^\lambda 1} & \cdots & w^{(1)}_{aN_a^\lambda n^\lambda} \end{bmatrix} \mapsto \begin{bmatrix} w^{(1)}_{a11} & \cdots & w^{(1)}_{a1n^\lambda} & & & \\ \vdots & \ddots & \vdots & & & \\ w^{(1)}_{aN_a^\lambda 1} & \cdots & w^{(1)}_{aN_a^\lambda n^\lambda} & & & \\ & & & w^{(1)}_{a(N_a^\lambda+1)(n^\lambda+1)} & \cdots & w^{(1)}_{a(N_a^\lambda+1)n^{\lambda+1}} \\ & & & \vdots & \ddots & \vdots \\ & & & w^{(1)}_{aN_a^{\lambda+1}(n^\lambda+1)} & \cdots & w^{(1)}_{aN_a^{\lambda+1}n^{\lambda+1}} \end{bmatrix} \tag{21}$$

This process is also visualized by Figure 5 for further illustration. The same approach is also used for the output layer weights $W_a^{(2)}(\bullet)$ and the bias terms. For the critic network, the process is identical. The greedy mapping strategy works well for the experiments discussed in Sections IV and V. However, adequate design of $\mathcal{A}_{map}$ is highly problem-dependent, which implies that other strategies are generally required for other types of curricula. This requires some domain knowledge, but this is commonly available when facing an optimal control problem.

D. Safety Mechanism

The quadratic design of the utility function specified by Equation 6 implies that safety is not encoded in the bare learning framework. To this end, an external local ergodicity-preserving safety filter is adopted, which omits the need for global safe backup policies to be available a priori. Under the authority of this filter, not only remains the system state bounded under unstable agent policies, but the use of safe online exploration/exploitation cycles also becomes a possibility. To keep learning throughout the safe subset $\mathcal{X}^{\lambda}_{safe} \subset \mathcal{X}^{\lambda}$, random exogenous exploration inputs $\boldsymbol{u}^{e}_{t}$ are applied on an occasional basis to excite the dynamics of the system. By virtue of the safety filter, it is ensured that these exploration inputs do not drive the system state outside of the safe set. As explained in Section II.C, this requires some form of (approximate) predictive model to be available. Also, any transitions generated under the authority of the safety filter are off-policy, and can therefore not be used for training in the on-policy HDP learning scheme.

The safe curriculum control architecture proposed in this study adopts the Safety Handling Exploration with Risk Perception Algorithm (SHERPA), which was developed by Mannucci et al. [17, 18] in the context of UAV exploration. SHERPA is essentially a heuristic search algorithm, which reduces the knowledge required for safe exploration to an overestimate of the system dynamics and experience gained online by the agent. It is based on the assumption that fatal events are related to fatal states instead of actions, giving rise to the concept of Fatal State Space (FSS) [18]:

$$\mathcal{X}_{fatal} = \left\{\boldsymbol{x'} | \forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{u} \in \mathcal{U}, \tau = \left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x'}\right) \in \mathcal{T}_{fatal}\right\} \tag{22}$$

with $\mathcal{T}_{fatal}$ the set of transitions that lead to fatal occurrences. The risk modes are assumed to be known a priori and collectively form the Restricted State Space (RSS). The intersection of the RSS with the FSS gives rise to the Restricted Fatal State Space (RFSS):

$$RFSS = RSS \cap \mathcal{X}_{fatal} \tag{23}$$

The RFSS is unknown a priori. However, it is assumed that the algorithm can perceive how close the current system state is with respect to the nearest fatal state. This is referred to as the risk perception capability of the algorithm. Risk perception is defined as a function $W(\boldsymbol{x})$ that returns 1 if risk is detected in a limited perception region H. If no risk is detected, all states in the region H are added to a belief of the safe state space $\tilde{\mathcal{X}}_{safe} \subseteq \mathcal{X}_{safe}$. In this way, the safe exploration area is gradually expanded. The framework is complemented by the concept of Lead-to-Fatal (LTF) states, which are not part of the FSS, but will evolve into the FSS with certainty [18]. The set of LTF states is defined as [18]:

$$L = \{\boldsymbol{x} | \forall \boldsymbol{u}(\boldsymbol{x}(t_0), t),\ \exists t : \boldsymbol{\sigma}(\boldsymbol{x}(t_0), \boldsymbol{u}(t), t) \in FSS\} \tag{24}$$

With $\boldsymbol{\sigma}(\boldsymbol{x}(t_0), \boldsymbol{u}(t), t)$ representing a state-trajectory between time t and t_0. The key idea is that from any state $\boldsymbol{x}_t$, the agent must be able to return to a state $\boldsymbol{x}_p$ visited before that lies within $\tilde{\mathcal{X}}_{safe} \subseteq \mathcal{X}_{safe}$, i.e. it must avoid both fatal states as well as lead-to-fatal states. The predictive capabilities of the SHERPA algorithm are based on an overestimate of the system dynamics, referred to as an internal bounding model. Given the total set of transitions $\mathcal{T}$ governed by the dynamics of the system described by Equation 5, the bounding model $\boldsymbol{\Delta}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ ensures the following [18]:

$$\forall \tau = \left\{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x'}\right\} \in \mathcal{T} : \boldsymbol{x'} \in \boldsymbol{\Delta}(\boldsymbol{x}, \boldsymbol{u}) \tag{25}$$

With this approach, if all trajectories τ predicted by the bounding model lie within $\tilde{\mathcal{X}}_{safe} \subseteq \mathcal{X}_{safe}$ and do not include any LTF states, safety of the system can be guaranteed with probability one. The SHERPA algorithm integrates all of these aspects in its search for control backups, which enables the agent to explore indefinitely. If the predicted distribution of states following a proposed input $[\boldsymbol{x}_{t+1}] = \boldsymbol{\Delta}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is completely in the known safe state space, safe return to the neighborhood of a state $\boldsymbol{x}_p$ visited earlier is guaranteed if a control backup sequence $\boldsymbol{U}_b(t)$ can be found for $[\boldsymbol{x}_{k+1}]$ [18]. A strictly feasible control sequence $\boldsymbol{U}_b = \{\boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t+m}\}$ is a control backup if [18],

$$\forall i \in \{1, 2, \ldots, m\}, [\boldsymbol{x}_{t+i}] \subset \tilde{\mathcal{X}}_{safe_t} \tag{26}$$

and $[\boldsymbol{x}_{k+t}] \cap L = \emptyset$, i.e.,

$$\forall \boldsymbol{x}_{t+m} \in [\boldsymbol{x}_{t+m}],\ \exists p \leq t : (\boldsymbol{x}_{t+m} - \boldsymbol{x}_p) \in [\boldsymbol{\epsilon}] \tag{27}$$

meaning that the bounded uncertainty cannot exceed a reaching interval $[\boldsymbol{\epsilon}]$. This reaching interval forms the closeness condition that must be satisfied by the backup sequence and must be chosen heuristically.

Following the insights from Section II.D, the search for $\boldsymbol{U}_b(t)$ can be expedited by operating on the the bounding model $\boldsymbol{\Delta}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ in a closed-loop form, i.e. under the authority of the agent policy. Using the task network mapping

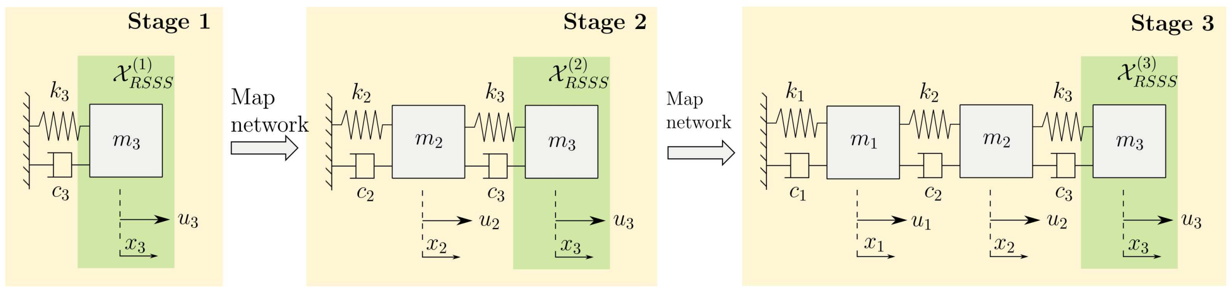


Fig. 6 Inter-task curriculum setup for the cascaded mass-spring-damper optimal regulation experiment

presented in Section III.C , the learning curriculum serves to reinforce this aspect of the SHERPA algorithm. The bounding model is then adapted with every policy update and learning stage, and can therefore be denoted as $\boldsymbol{\Delta}^{\lambda}_{\pi_j}(\boldsymbol{x}_t, \boldsymbol{u}_t)$.

SHERPA has been developed based on Interval Analysis (IA) techniques [17, 18]. However, the rapid growth of inclusion intervals under even moderate uncertainties generally leads to conservative trajectory predictions. Therefore, the proposed safe curriculum control framework takes a different approach based on model sampling. This implies the condition expressed by Equation 25 can only be met probabilistically, where the probability of adequate bounding model prediction is directly related to the parametric extremity of the actual system with respect to the total model space. This aspect is taken into account in the experiments discussed in Sections IV and V.

IV. Fundamental Experiments

This section presents the findings from a study based on a cascaded linear mass-spring-damper (MSD) system that has been performed to investigate the proposed safe curriculum control framework. The advantage of this setup is that validation of the results is possible using LQR theory and that its complexity can be altered in a straightforward manner. For example, one can adjust the dimensionality of the (observable) state-space and modify the number of actuators to be controlled by the learning agent. By focusing on this simple system first, valuable experience with the framework can be obtained before proceeding to quadrotor implementation in Section V. The experimental setup is briefly described in Section IV.A and is followed by a presentation of the results in Section IV.B.

A. Experimental Setup

The experimental setup is illustrated in Figure 6. The target system consists of three masses m_i that are connected along a single dimension via springs with constants k_i and dampers with coefficients c_i. A fixed inter-task learning curriculum has been designed to make the target learning task more tractable, with the dimensionality of the state and action spaces serving as the primary shaping factors. At any stage in the learning curriculum, the actor and critic networks* are scaled accordingly using the greedy task network mapping strategy from Section III.C.

An overview of the parameters defining the MSD sequence is given in Table 1. As a result of the negative value of k_3, the system is inherently unstable. Each mass features a unique actuator, which implies that the system is fully controllable. For the utility function, the following structure applies:

$$r(\boldsymbol{x}^{(\lambda)}_t, \boldsymbol{u}^{(\lambda)}_t) = 25 \left(\mathbf{x}^{(\lambda)}_t\right)^T \mathbf{x}^{(\lambda)}_t + \left(\boldsymbol{u}^{(\lambda)}_t\right)^T \boldsymbol{u}^{(\lambda)}_t \tag{28}$$

where $\mathbf{x}^{(\lambda)}_t \subset \boldsymbol{x}^{(\lambda)}_t$ represents the position vector. Safety resides in the fact that the outer mass m_3 must remain within a region of ± 1 m around its equilibrium point, which is indicated as $\mathcal{X}^{(\lambda)}_{RSSS} = RSS \cap \mathcal{X}^{(\lambda)}_{safe}$. The SHERPA safety filter is equipped with a bounding model of the system that features a $\pm 25\%$ parametric uncertainty range for $\{k_1, k_2, k_3\}$ and $\{c_1, c_2, c_3\}$. The available input authority for backups and exogenous exploration inputs is limited to ± 10 N.

Proper configuration of the safe curriculum control setup involves tuning a relatively large number of hyperparameters. These relate primarily to exploration, the ANN training scheme, and the SHERPA safety filter. Online interaction times have been set to $\{120, 180, 240\}$ seconds for every policy exploitation cycle, during which random exogenous exploration

*For this kind of LQR problem, less complex learning strategies based on least-squares regression are generally better suited compared to techniques that adopt gradient descent [42, 44, 48]. This follows from the fact that the optimal value function is quadratic.

Table 1 Cascaded mass-spring-damper system characteristics

Element #	m [kg]	k [N/m]	c [N / ms^{-1}]
1	0.6	5	4
2	1.2	7	2
3	0.8	−3	3

inputs $\boldsymbol{u}_t^e$ are applied intermittently. Since data may be scarce in some regions of the state space, especially for learning stages 2 and 3, the number of training epochs is kept at a relatively low value of 10 for every update step in the GPI HDP loop. This prevents the ANN training algorithm from overfitting and destroying the greedy actor-critic relationship. For the policy evaluation step, the number of iterations has been set to $K = 3$. For the actor and critic networks, the number of hidden nodes is set to $\{32, 64, 96\}$. This results in considerable approximation power, although the learning task could also be solved using a much smaller number of nodes. These settings were seen to work sufficiently in practice. Initialization of the network weights and biases is performed on a range between $[-0.01, 0.01]$, which ensures that the actor network approximates the zero policy if learning has not taken place.

For SHERPA, the risk perception limits have been set to ±0.2 m. This implies that the algorithm will only detect the boundary of $\mathcal{X}_{RSSS}^{(3)}$ once the position of the outer mass element comes within a 0.2 m distance. The reaching intervals ϵ_x and $\epsilon_{\dot{x}}$ for position and velocity have been set at 0.1 m and 0.2 m/s, respectively. For the probabilistic bounding model, the number of model samples has been set to 10. The maximum number of iterations for selecting a control action and finding a control backup sequence has been set to 5 and 10, respectively. These settings were selected primarily to keep the algorithm's computational complexity at a minimum[†].

For tuning the backup search itself, some domain knowledge is required. For this experiment, the minimum and maximum lengths of the backup sequence have been set to 0.2 and 0.6 seconds, respectively. The backup consists of a sequence of input bands that keep the control input constant over multiple time steps, which in this case amounts to 0.2 seconds. In this way, the likelihood that an effective control sequence is identified is increased. Finally, a state will only be added to the internal memory of visited steps every 0.3 seconds to prevent excessive memory requirements.

B. Results and Discussion

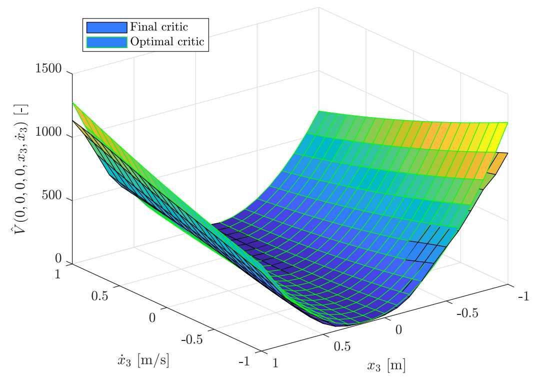
To draw sound conclusions on the effectiveness of the safe curriculum learning paradigm, the results of three learning strategies are examined. First, a benchmark case is considered without the use of a learning curriculum or safety filter. In this case, learning takes place directly in the stage 3 target task, and there is nothing that prevents the system from violating the imposed safety constraint. Exploration is achieved through random initialization of the system in $\mathcal{X}_{safe}^{(3)}$, which makes the learning problem episodic in nature. Figure 7 visualizes typical cross-sections of the actor and critic networks at the end of learning. These show that the optimal value function and policy that follow from the solution of the discrete-time Algebraic Ricatti Equation (ARE) are approximated quite well, except in the regions where x_3 is in the vicinity of the fatal state space and the velocity vector points towards its direction. This is an intuitive result, as this situation will only rarely (if ever) be encountered under a stable policy.

Figure 8 shows a few learning statistics in terms of the normalized Root Mean Square Error (RMSE) with respect to the optimal control policy and the number of constraint violations. The normalized RMSE is obtained using the Root Mean Square (RMS) control input from the optimal control policy:

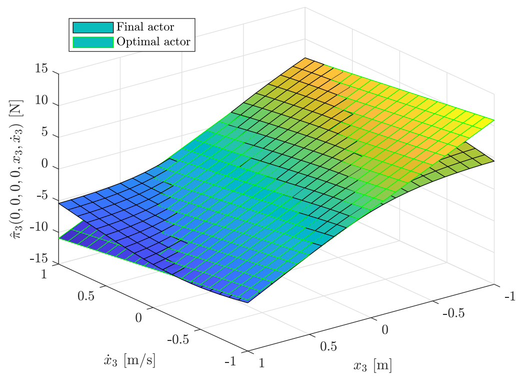
$$\overline{RMSE} = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{u}^*(\boldsymbol{x}_i) - \hat{\boldsymbol{u}}(\boldsymbol{x}_i))^2\right)} \Big/ \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{u}^*(\boldsymbol{x}_i))^2} \tag{29}$$

At the start of learning, the normalized RMSE is always equal to 1 as a result of the zero policy. This value reduces with every policy iteration, as shown in Figure 8a. As the actor policy converges towards the optimal control, it is able to stabilize the system. This is reflected in the low number of constraint violations in Figure 8b at the end of learning. However, prior to this point, a significant number of violations is incurred.

[†]Note that the proposed algorithm does not run in real-time.

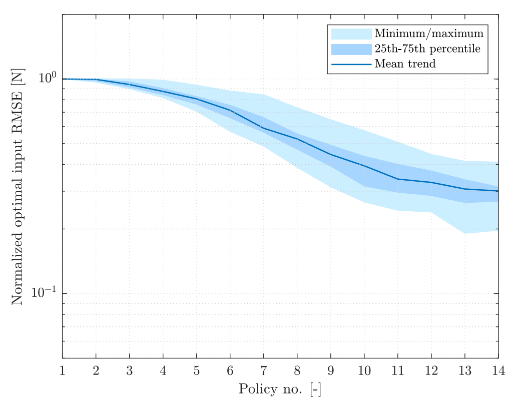


(a) Value function

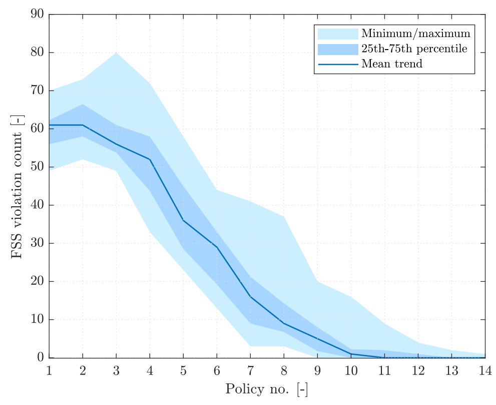


(b) Policy

Fig. 7 Example actor and critic outputs at the end of learning; non-curriculum approach, no safety filter

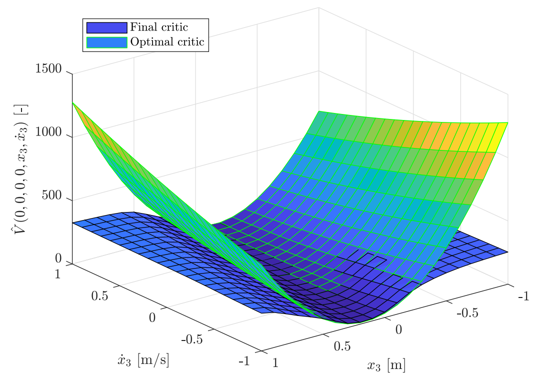


(a) Normalized policy RMSE

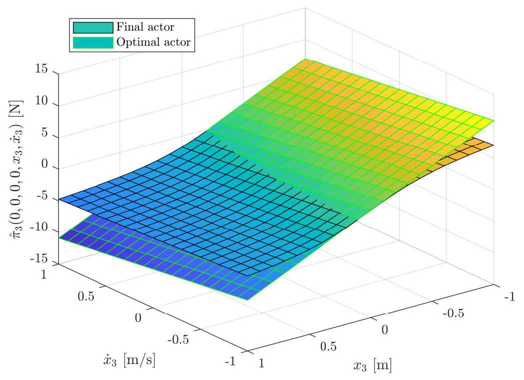


(b) Constraint violations

Fig. 8 Learning statistics for 25 independent learning trials; non-curriculum approach, no safety filter



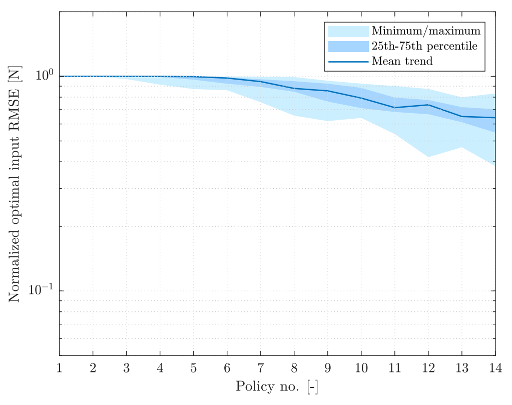
(a) Value function



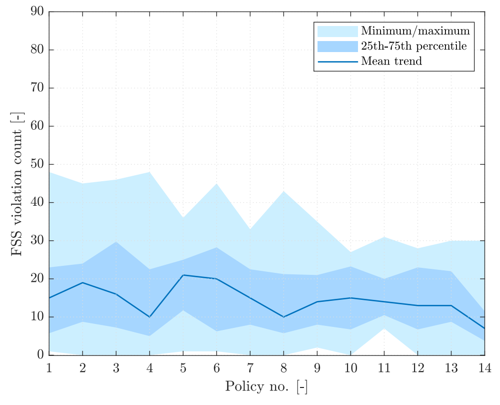
(b) Policy

Fig. 9 Example actor and critic outputs at the end of learning; non-curriculum approach, SHERPA enabled

To examine the impact of the SHERPA safety filter on the learning performance, a second experiment is performed where SHERPA is enabled in the control loop without the use of a learning curriculum. Typical outputs of the actor and

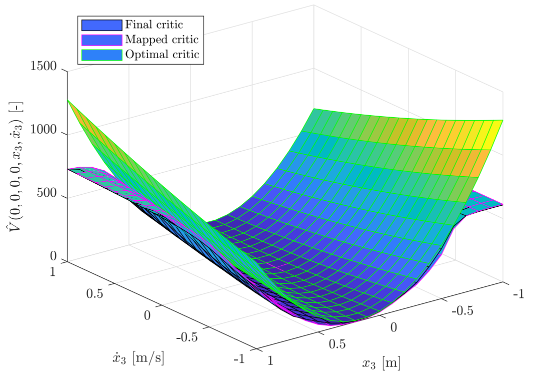


(a) Normalized policy RMSE

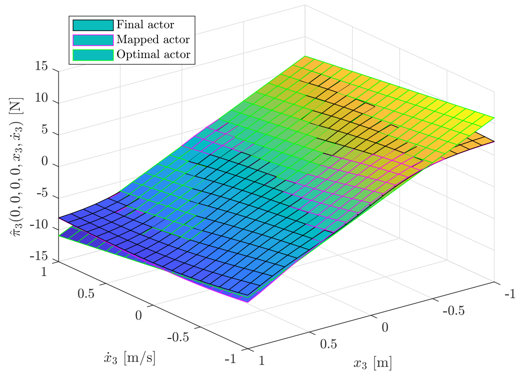


(b) Constraint violations

Fig. 10 Learning statistics for 25 independent learning trials; non-curriculum approach, SHERPA enabled

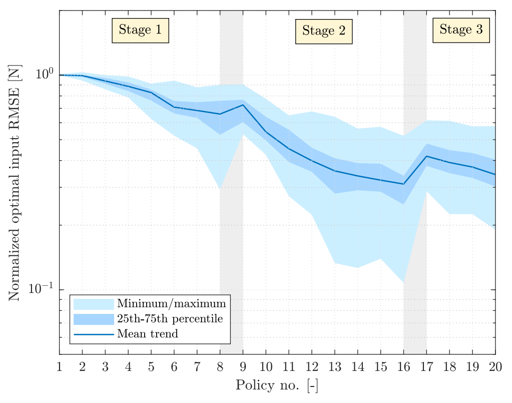


(a) Value function

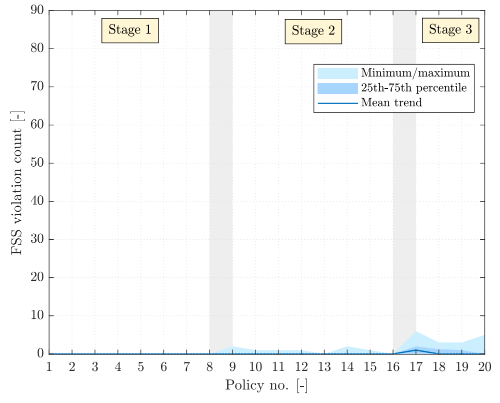


(b) Policy

Fig. 11 Example actor and critic outputs at the end of learning; safe curriculum learning using SHERPA



(a) Normalized policy RMSE



(b) Constraint violations

Fig. 12 Learning statistics for 25 independent learning trials; safe curriculum learning using SHERPA

critic networks are visualized in Figure 9. The learning statistics are shown in Figure 10. These results indicate that the learning pace is reduced compared to the non-safe approach. Although the number of constraint violations is smaller, especially in the early learning phase, SHERPA is often not capable of identifying feasible control backups within the limited number of iterations available. This is a direct consequence of the fact that the subspace of ergodicity-preserving policies is relatively small compared to the total policy space.

The final experiment aims to investigate the extent to which the learning curriculum can expedite the search for ergodicity-preserving policies, and quantify the effect on learning efficiency. Figure 11 illustrates the actor and critic network outputs at the end of learning, together with the optimal and mapped value functions and policies. This shows that the mapped control policy is already a good approximation of the optimal policy. As a result, both the system itself as well as the bounding model $\boldsymbol{\Delta}^{(3)}_{\pi_0}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ are stable under the transformed predecessor policy. The impact on learning efficiency and safety is illustrated in Figure 12. These results show that the number of constraint violations is reduced considerably compared to the prior two experiments. In terms of learning efficiency, similar normalized RMSE levels are reached as for the non-safe, non-curriculum approach in the first experiment, although the spread is larger. Note that the particular selection of online interaction times, combined with the total number of policy updates, results in the same number of collected samples for all experiments.

V. Flight Control Application

The results obtained from the cascaded linear mass-spring-damper experiments illustrate the potential of safe curriculum learning in expediting learning efficiency and safety in the context of optimal control. This yields an adequate basis for demonstrating the paradigm for autonomous learning of optimal control laws in the context of flight control. In this section, an experiment is described for optimal attitude regulation of a nonlinear quadrotor model in both pitch and roll axes. This experiment serves primarily as a proof-of-concept. A description of the system dynamics including additional simplifications made is given first in Subsection V.A, which is followed by an overview of the learning framework in V.B. The results are presented in Section V.C.

A. Quadrotor Simulation Model

The system used for this experiment is modeled after a Parrot AR 2.0 UAV and has been of use in earlier RL research as well [21, 49]. Rotational and translational control of the UAV is achieved by changing the rotational velocity of the individual motors. For the translational dynamics, the Earth frame serves as an inertial reference frame with the gravity vector acting perpendicular to the surface. Following Newton's second law and focusing on the body-fixed frame of reference, this implies that the governing equations of motion take the following form:

$$\dot{\boldsymbol{V}}^b_E = \frac{1}{m}\left(\boldsymbol{F}^b_F + \boldsymbol{F}^b_P\right) + \boldsymbol{g}^b - \boldsymbol{\omega}^b_{b/E} \times \boldsymbol{V}^b_E \tag{30}$$

where $\boldsymbol{V}^b_E = \begin{bmatrix} u & v & w \end{bmatrix}^T$ represents the velocity vector, $\boldsymbol{\omega}^b_{b/E} = \begin{bmatrix} p & q & r \end{bmatrix}^T$ the angular velocity vector, $\boldsymbol{g}^b$ the gravitational acceleration vector, and $\boldsymbol{F}^b_F$ and $\boldsymbol{F}^b_P$ the forces associated with the vehicle frame and rotors, respectively. The exogenous forces generated by the rotors largely dominate state-dependent aerodynamic forces. The steady-state rotor thrust in hover scales quadratically with rotational velocity according to momentum theory [49, 50]:

$$T_P = \sum_{i=1}^{4} T_i = \sum_{i=1}^{4} C_T \omega_i^2 \tag{31}$$

where C_T is a lumped positive scalar. In this experiment, the simplification is made that thrust levels follow this static relationship in any condition, and are therefore independent of effective rotor speed. Moreover, the assumption of perfect motors is made, which implies that the rotors reach a desired rotational velocity instantaneously. Blade flapping is incorporated for both the longitudinal and lateral directions, and also follows from a static relationship:

$$X_{bf} = -\sum_{i=1}^{4} T_i \sin(C_\beta v_{P_i}) \qquad\qquad Y_{bf} = \sum_{i=1}^{4} T_i \sin(-C_\beta u_{P_i}) \tag{32}$$

with C_β serving as the tilt constant, and u_{P_i} and v_{P_i} representing the u- and v-components of the local rotor speed. This yields the following expression for $\boldsymbol{F}^b_P$:

$$\boldsymbol{F}^b_P = \begin{bmatrix} X_{bf} & Y_{bf} & -T_P \end{bmatrix}^T \tag{33}$$

The airframe itself generates lift and drag forces in the aerodynamic frame of reference:

$$\boldsymbol{F}_F^b = \mathbb{T}_{ab}^{-1}(\alpha,\beta)\boldsymbol{F}_F^a = \mathbb{T}_{ab}^{-1}(\alpha,\beta)\bar{q}S\begin{bmatrix}-C_D & 0 & -C_L\end{bmatrix}^T \tag{34}$$

The rotational equations of motion are governed by the Euler equations:

$$\dot{\boldsymbol{\omega}}_{b/E}^b = \boldsymbol{J}^{-1}\left(\boldsymbol{M}_F^b + \boldsymbol{M}_P^b - \boldsymbol{\omega}_{b/E}^b \times \boldsymbol{J}\boldsymbol{\omega}_{b/E}^b\right) \tag{35}$$

where $\boldsymbol{M}_F^b$ and $\boldsymbol{M}_P^b$ are the moments associated with the vehicle frame and rotors, respectively, and $\boldsymbol{J}$ represents the diagonal inertia matrix. The moments generated by the rotors are again dominant, and are given as follows [49]:

$$\boldsymbol{M}_P^b = \begin{bmatrix} 0 & -l & 0 & l \\ l & 0 & -l & 0 \\ -C_\tau & C_\tau & -C_\tau & C_\tau \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} + \begin{bmatrix} L_{bf} \\ M_{bf} \\ 0 \end{bmatrix} \tag{36}$$

where C_τ is the rotor torque constant and l represents the rotor arm length. The terms related to blade flapping take the following form:

$$L_{bf} = -C_{\beta_m}\sum_{i=1}^{4} C_\beta v_{P_i} \qquad\qquad M_{bf} = C_{\beta_m}\sum_{i=1}^{4} C_\beta u_{P_i} \tag{37}$$

where C_{β_m} is a fixed scalar. The moment contribution by the vehicle airframe is very minor, and only appears as a rotational drag term:

$$\boldsymbol{M}_F^b = \begin{bmatrix}0 & 0 & \frac{1}{2}C_{\tau_D}\rho r^2\end{bmatrix} \tag{38}$$

Finally, the vehicle attitude and position are governed by the following equations:

$$\dot{\boldsymbol{\Phi}} = \begin{bmatrix}\dot{\phi} \\ \dot{\theta} \\ \dot{\psi}\end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi] \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad\qquad \dot{\mathbf{x}}_E^E = \mathbb{T}_{bE}^{-1}(\phi,\theta)\boldsymbol{V}_E^b \tag{39}$$

In the model, the attitude dynamics are simulated using quaternions to prevent the occurrence of singularities [49]. It is assumed that perfect sensors are available, with zero bias and noise, and that no external disturbances are acting on the system. The equations of motion are transformed to discrete-time by applying the forward Euler method and a small simulation time step of 0.005 seconds.

B. Learning Framework

A fixed two-stage intra-task learning curriculum is established, in which learning is limited to the longitudinal dynamics only in the first stage, and is extended to the lateral axis in the subsequent stage. The observed outputs and the inputs available to the agent during these phases are as follows:

$$\boldsymbol{y}_t^{(1)} = \begin{bmatrix}u & w & q & \theta\end{bmatrix}^T \qquad\qquad \boldsymbol{u}_t^{a(1)} = \begin{bmatrix}M_P\end{bmatrix}^T \tag{40}$$

$$\boldsymbol{y}_t^{(2)} = \begin{bmatrix}u & v & w & p & q & \phi & \theta\end{bmatrix}^T \qquad\qquad \boldsymbol{u}_t^{a(2)} = \begin{bmatrix}L_P & M_P\end{bmatrix}^T \tag{41}$$

The greedy task network mapping described in Section III.C is used to bias learning in the lateral axis. The inputs are taken as the moments generated through differential rotor thrust, and are allocated to rotor speed using the relations:

$$\begin{bmatrix}\Delta T_1 \\ \Delta T_2 \\ \Delta T_3 \\ \Delta T_4\end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2l} \\ -\frac{1}{2l} & 0 \\ 0 & -\frac{1}{2l} \\ \frac{1}{2l} & 0 \end{bmatrix}\begin{bmatrix} L_p \\ M_p \end{bmatrix} \qquad\qquad \omega_i = \sqrt{C_T^{-1}(T_{i_0} + \Delta T_i)} \tag{42}$$

Table 2 Quadrotor UAV safe curriculum learning hyperparameters

Parameter	**Value**	**Parameter**	**Value**
Policy evaluation iteration size K	5 [-]	Attitude, speed risk perception range	10 [deg], 1.0 [m/s]
Exploration/backup authority limits	±0.2 [Nm]	Reaching intervals $\epsilon_{u,v}$, $\epsilon_{p,q}$, $\epsilon_{\phi,\theta}$	0.5 [m/s], 10 [deg/s], 5 [deg]
Actor/critic number of hidden nodes	{32,64} [-]	Model uncertainty ΔC_β, ΔC_{β_m}	± 0.25 [-]
Number of training epochs	20 [-]	Backup horizon range	[0.15, 0.90] [sec]
Network weight initialization range	[-0.01, 0.01]	Number of control input iterations	3 [-]
Number of bounding model samples	2 [-]	Number of backup iterations	5 [-]

where T_{i_0} represents the trim thrust in hovering flight. By allocating the agent inputs in this way, the continuous-time moment effectiveness matrix simply equals the inverted inertia matrix.

Although the inactive state dimensions are to a large extent decoupled from the agent-controlled dynamics at any stage in the learning curriculum, their inherently unstable nature requires separate measures to keep the total system state bounded. To this end, a proportional multi-variable feedback control law is used as an external supervisor:

$$\boldsymbol{u}_t^{s(\lambda)} = K_s^{(\lambda)} \begin{bmatrix} p & q & r & \theta & \psi \end{bmatrix}^T \tag{43}$$

Although the supervisor is designed to stabilize the UAV in hover condition, its performance is far from optimal. In the first stage, the supervisor has both the roll and yaw axes under its authority, whereas it only controls yaw in the second stage. Although it never takes control of the pitch channel, it can be used to initialize the actor network such that the initial policy is stable around the hover condition. This implies that a large degree of safety is incorporated already at the onset of learning. For the target task utility function, the following structure is adopted:

$$r(\boldsymbol{y}_t^{(2)}, \boldsymbol{u}_t^{a(2)}) = \begin{bmatrix} u & v & \phi & \theta \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \begin{bmatrix} u \\ v \\ \phi \\ \theta \end{bmatrix} + \begin{bmatrix} L_P & N_P \end{bmatrix} \begin{bmatrix} I_{xx} & 0 \\ 0 & I_{yy} \end{bmatrix} \begin{bmatrix} L_P \\ N_P \end{bmatrix} \tag{44}$$

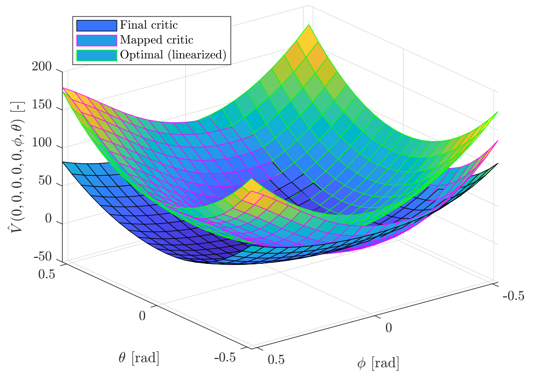
Although optimal control of attitude is the primary goal, it has been observed that the learning process becomes more tractable by incorporating additional penalty terms for velocity. For the first curriculum stage, the utility function is reduced to the longitudinal terms only.

Regarding the safety aspect of the learning task, two risk modes are established that define the safe subset $\mathcal{X}_{safe}^{(\lambda)} \subset \mathcal{X}^{(\lambda)}$. The first mode limits the attitude angle to ± 30 degrees, whereas the second mode specifies that the body speeds u and v should not exceed 3 m/s. Accordingly, the risk perception limits of the SHERPA safety filter have been set to 10 degrees and 1 m/s, respectively. For the internal bounding model, a reduced description of the nonlinear quadrotor dynamics is adopted which only includes the dominant dynamics related to exogenous rotor thrust and blade flapping. For the blade flapping contribution, parametric uncertainties of ±25% are used for the C_β and C_{β_m} terms. As these parameters can be lumped into a single term $C_\beta C_{\beta_m}$ for the rotational dynamics, as illustrated by equations 37, only two model samples corresponding to the lumped infimum and supremum are adopted. Since the internal bounding model $\boldsymbol{\Delta}_{\pi_j}^{\lambda}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ operates on the full state and input space, it also contains an internal representation of the supervisor.

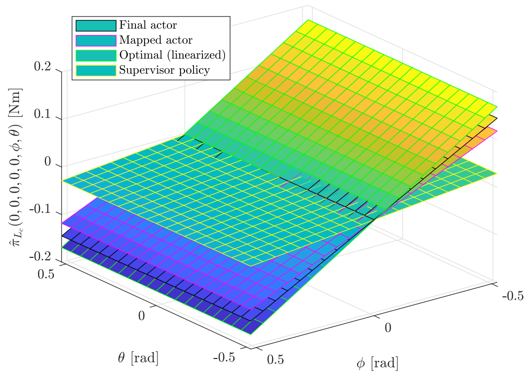
Finally, appropriate values need to be selected for the various hyperparameters associated with learning and the SHERPA safety filter. These are summarized in Table 2. These hyperparameters generally have a significant impact on learning performance, and obtaining adequate settings can be a tedious task.

C. Results and Discussion

Figure 13 visualizes the cross-sections of the critic and actor outputs as a function of attitude for the nonlinear target task after completion of the learning curriculum. To be able to evaluate learning performance, the optimal control policy and corresponding value function obtained by solving the discrete-time ARE for the quadrotor dynamics



(a) Value function

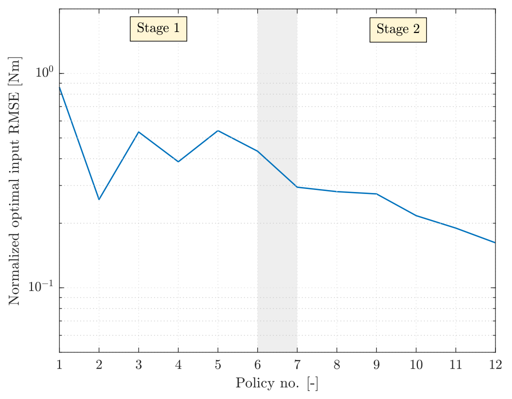


(b) Lateral policy

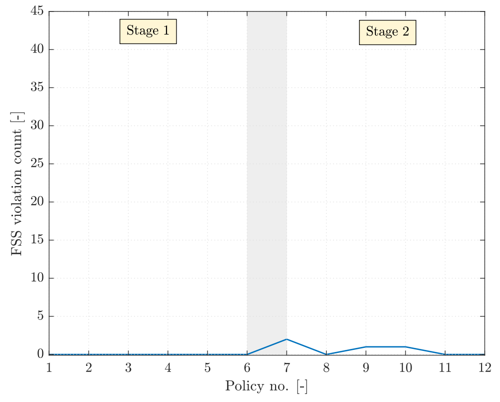
Fig. 13 Example actor and critic outputs as a function of pitch and roll Euler angles at the end of learning, quadrotor safe learning curriculum under SHERPA authority; the optimal outputs are based on the system linearized around the hover condition

linearized around the hover condition are shown as well. This shows that the optimal solution is approximated reasonably well by both the mapped and final actor-critic networks. The learning statistics corresponding to the safe curriculum demonstration run are shown in Figure 14. To give an indication of the favorable properties of the safe curriculum learning paradigm, Figure 15 shows the same information when learning takes place directly in the target task.

The observation is made that the curriculum approach results in lower normalized RMSE levels with fewer constraint violations. Despite the fact that any of the intermediate control policies are stable and therefore in principle safe for both experiments, the bounding model uncertainty complicates the identification of safe backup sequences. This is especially problematic for the target task, where uncertainties propagate in both the pitch and roll dimensions. Under the low-gain supervisor control law, these uncertainties result in a relatively large variation in the closed-loop bounding model response, which further reduces the likelihood of finding a backup control sequence that satisfies the closeness condition. By adopting a curriculum approach, the model uncertainty impact is better controlled, as variations in the first learning stage remain confined to the pitch dimension only. Moreover, the control policy obtained from the first stage is relatively high-gain, which limits the impact of model variations.

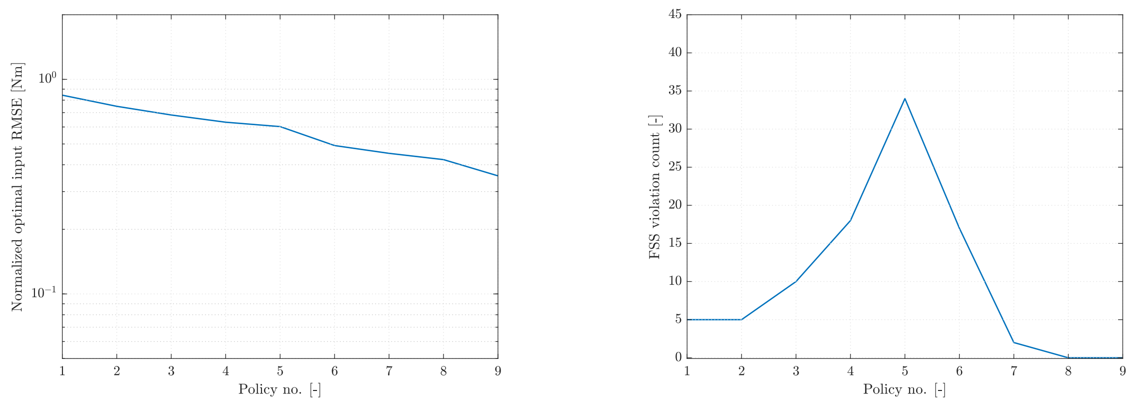


(a) Normalized policy RMSE



(b) Constraint violations

Fig. 14 Nonlinear quadrotor demonstration learning statistics, safe learning curriculum under SHERPA authority; normalized policy RMSE is determined using the optimal policy for the system linearized around the hover condition



(a) Normalized policy RMSE **(b) Constraint violations**

Fig. 15 Nonlinear quadrotor demonstration run learning statistics, non-curriculum approach under SHERPA authority; normalized policy RMSE is determined using the optimal policy for the system linearized around the hover condition

VI. Conclusions and Recommendations

This paper has demonstrated the safe curriculum learning paradigm to autonomously learn optimal control laws for systems with parametric uncertainties in a safe and efficient online manner. Under minimum robustness requirements, curriculum learning can be exploited to simultaneously enhance safety and efficiency during learning by transferring safe agent policies across successive stages in an MDP sequence. A neural-network-based actor-critic curriculum learning scheme supported by the SHERPA ergodicity-preserving safety filter has been proposed to investigate the framework in the context of general nonlinear, time-invariant systems with known input dynamics and partially uncertain internal dynamics. Using the concept of a-priori designed task network mappings, the transfer of neural networks weights across successive representations of the actor and the critic in the learning sequence is adopted to enhance learning efficiency and expedite the search for ergodicity-preserving backup control sequences. The latter is achieved by having SHERPA operate on internal closed-loop bounding models under the authority of the actor policy.

The safe curriculum framework has been demonstrated in a simple inter-task learning experiment based on a linear, unstable, cascaded mass-spring-damper system. The results show simultaneous improvement of learning efficiency and safety compared to non-curriculum approaches, which is a direct consequence of the fact that the SHERPA safety filter is able to exploit the stabilizing and uncertainty-limiting properties of the actor policy in the most demanding stages of the learning curriculum under the proposed network mapping strategy. Subsequent proof-of-concept of the framework in the context of optimal pitch and roll attitude control of a quadrotor UAV has resulted in similar observations, with the search for backup policies being more effective under the curriculum approach as a result of reduced model uncertainty propagation.

The results from this paper show that the informed application of the safe curriculum paradigm has the potential to advance the applicability of online reinforcement learning techniques for safety-critical systems such as UAVs. However, there are many challenges and directions for improvement that need to be addressed in future research. First, the level of a priori knowledge required should be further reduced to have a true advantage over the traditional flight control design cycle in terms of model-dependency. This could for example be achieved by applying self-paced and model-learning techniques. The former incorporates feedback from the agent learning progress, and may therefore result in more effective learning sequences. Online model-learning could be investigated to achieve reliable bounding model contractions, which may allow for higher initial model uncertainties and eventually reduced uncertainty propagation as the model is adapted using online system information. A curriculum approach could be considered here as well.

Second, safety and learning could be further integrated by introducing feedback from the SHERPA safety filter to the learning agent. Third, the curriculum framework could be further explored by quantifying the effects of non-observable dynamics and developing minimum conditions for positive knowledge transfer for improving learning efficiency. Finally, further research is needed to reduce the computational complexity of the algorithm and minimize the effects of hyperparameters. Since considerable effort must be spent to obtain adequate settings for neural network training and the safety filter, informed guidelines should be developed before the framework can be used in real applications.

References

[1] Stevens, B., Lewis, F., and Johnson, E., *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, Wiley, 2015.

[2] Khargonekar, P. P., Petersen, I. R., and Zhou, K., "Robust stabilization of uncertain linear systems: quadratic stabilizability and H^∞ control theory," *IEEE Transactions on Automatic Control*, Vol. 35, No. 3, 1990, pp. 356–361. doi:10.1109/9.50357.

[3] Veillette, R. J., Medanic, J. B., and Perkins, W. R., "Design of reliable control systems," *IEEE Transactions on Automatic Control*, Vol. 37, No. 3, 1992, pp. 290–304. doi:10.1109/9.119629.

[4] Guang-Hong Yang, Lam, J., and Jianliang Wang, "Reliable H_∞ control for affine nonlinear systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 8, 1998, pp. 1112–1117. doi:10.1109/9.704984.

[5] Sieberling, S., Chu, Q. P., and Mulder, J. A., "Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 6, 2010, pp. 1732–1742. doi:10.2514/1.49978.

[6] Slotine, J., and Li, W., *Applied nonlinear control*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[7] Sonneveldt, L., Chu, Q., and Mulder, J., "Nonlinear Flight Control Design Using Constrained Adaptive Backstepping," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 322–336. doi:10.2514/1.25834.

[8] Acquatella, P., van Kampen, E., and Chu, Q., "Incremental backstepping for robust nonlinear flight control," *EuroGNC 2013, 2nd CEAS Specialist Conference on Guidance, Navigation & Control*, 2013.

[9] Kaelbling, L., Littman, M., and Moore, A., "Reinforcement Learning: A Survey," *Journal of artificial intelligence research*, Vol. 4, 1996, pp. 237–285.

[10] Sutton, R., and Barto, A., *Reinforcement learning: An introduction*, MIT press, Cambridge, MA, 1998.

[11] Szepesvári, C., "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, Morgan & Claypool Publishers, 2010.

[12] Busoniu, L., Babuska, R., Schutter, D., and Ernst, D., *Reinforcement Learning and Dynamic Programming Using Function Approximators*, Automation and Control Engineering, CRC Press, Inc., 2010.

[13] Sutton, R. S., Barto, A. G., and Williams, R. J., "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine*, Vol. 12, No. 2, 1992, pp. 19–22. doi:10.1109/37.126844.

[14] Bengio, Y., Louradour, J., Collobert, R., and Weston, J., "Curriculum Learning," *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 41–48. doi:10.1145/1553374.1553380.

[15] Taylor, M. E., "Assisting Transfer-Enabled Machine Learning Algorithms: Leveraging Human Knowledge for Curriculum Design," *Agents that Learn from Human Teachers, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-01, Stanford, California, USA, March 23-25, 2009*, 2009, pp. 141–143.

[16] García, J., and Fernández, F., "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, Vol. 16, No. 42, 2015, pp. 1437–1480.

[17] Mannucci, T., Van Kampen, E., de Visser, C. C., and Chu, Q. P., "SHERPA: a safe exploration algorithm for Reinforcement Learning controllers," *AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum*, 2015. doi:10.2514/6.2015-1757.

[18] Mannucci, T., van Kampen, E., de Visser, C., and Chu, Q., "Safe Exploration Algorithms for Reinforcement Learning Controllers," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 4, 2018, pp. 1069–1081. doi:10.1109/TNNLS.2017.2654539.

[19] Perkins, T., and Barto, A., "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, Vol. 3, 2003, pp. 803–832. doi:10.1162/jmlr.2003.3.4-5.803.

[20] Held, D., McCarthy, Z., Zhang, M., Shentu, F., and Abbeel, P., "Probabilistically safe policy transfer," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5798–5805. doi:10.1109/ICRA.2017.7989680.

[21] Helmer, A., de Visser, C. C., and van Kampen, E., "Flexible Heuristic Dynamic Programming for Reinforcement Learning in Quad-Rotors," *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, 2018. doi:10.2514/6.2018-2134.

[22] Taylor, M., and Stone, P., "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, Vol. 10, No. 1, 2009, pp. 1633–1685.

[23] Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H., *Adaptive dynamic programming with applications in optimal control*, Springer, 2017.

[24] Wang, F., Zhang, H., and Liu, D., "Adaptive Dynamic Programming: An Introduction," *IEEE Computational Intelligence Magazine*, Vol. 4, No. 2, 2009, pp. 39–47. doi:10.1109/MCI.2009.932261.

[25] Werbos, P., "Approximate dynamic programming for realtime control and neural modelling," *Handbook of intelligent control: neural, fuzzy and adaptive approaches*, 1992, pp. 493–525.

[26] Prokhorov, D. V., and Wunsch, D. C., "Adaptive critic designs," *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, 1997, pp. 997–1007. doi:10.1109/72.623201.

[27] Skinner, B. F., "Reinforcement today." *American Psychologist*, Vol. 13, No. 3, 1958, pp. 94–99. doi:10.1037/h0049039.

[28] Kumar, M. P., Packer, B., and Koller, D., "Self-Paced Learning for Latent Variable Models," *Advances in Neural Information Processing Systems 23*, 2010, pp. 1189–1197.

[29] Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., and Hauptmann, A., "Self-Paced Learning with Diversity," *Advances in Neural Information Processing Systems 27*, 2014, pp. 2078–2086.

[30] Narvekar, S., Sinapov, J., and Stone, P., "Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning," *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 2536–2542. doi:10.24963/ijcai.2017/353.

[31] Taylor, M. E., Stone, P., and Liu, Y., "Transfer Learning via Inter-Task Mappings for Temporal Difference Learning," *Journal of Machine Learning Research*, Vol. 8, No. 1, 2007, pp. 2125–2167.

[32] Rosenstein, M. T., and Barto, A. G., "Reinforcement learning with supervision by a stable controller," *Proceedings of the 2004 American Control Conference*, Vol. 5, 2004, pp. 4517–4522 vol.5. doi:10.23919/ACC.2004.1384022.

[33] Hans, A., Schneegaß, D., Schäfer, A. M., and Udluft, S., "Safe Exploration for Reinforcement Learning," *Proceedings of the 16th European Symposium on Artificial Neural Networks*, 2008, pp. 143–148.

[34] Moldovan, T. M., and Abbeel, P., "Safe Exploration in Markov Decision Processes," *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 1451–1458.

[35] Akametalu, A. K., Fisac, J. F., Gillula, J. H., Kaynama, S., Zeilinger, M. N., and Tomlin, C. J., "Reachability-based safe learning with Gaussian processes," *53rd IEEE Conference on Decision and Control*, 2014, pp. 1424–1431. doi: 10.1109/CDC.2014.7039601.

[36] Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A., "Safe model-based reinforcement learning with stability guarantees," *Proceedings of Neural Information Processing Systems*, 2017, pp. 908–918.

[37] Freeman, R., and Kokotovic, P. V., *Robust nonlinear control design: state-space and Lyapunov techniques*, Springer Science & Business Media, 2008.

[38] Wieland, P., and Allgöwer, F., "Constructive Safety using Control Barrier Functions," *IFAC Proceedings Volumes*, Vol. 40, No. 12, 2007, pp. 462 – 467. doi:https://doi.org/10.3182/20070822-3-ZA-2920.00076, 7th IFAC Symposium on Nonlinear Control Systems.

[39] Tee, K. P., Ge, S. S., and Tay, E. H., "Barrier Lyapunov Functions for the control of output-constrained nonlinear systems," *Automatica*, Vol. 45, No. 4, 2009, pp. 918 – 927. doi:https://doi.org/10.1016/j.automatica.2008.11.017.

[40] Lazaric, A., *Transfer in Reinforcement Learning: A Framework and a Survey*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 143–173. doi:10.1007/978-3-642-27645-3_5.

[41] Kiumarsi, B., Vamvoudakis, K. G., Modares, H., and Lewis, F. L., "Optimal and Autonomous Control Using Reinforcement Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 6, 2018, pp. 2042–2062. doi:10.1109/TNNLS.2017.2773458.

[42] Lewis, F. L., and Vrabie, D., "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, Vol. 9, No. 3, 2009, pp. 32–50. doi:10.1109/MCAS.2009.933854.

[43] Al-Tamimi, A., Lewis, F. L., and Abu-Khalaf, M., "Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 38, No. 4, 2008, pp. 943–949. doi:10.1109/TSMCB.2008.926614.

[44] Kiumarsi-Khomartash, B., Lewis, F. L., Naghibi-Sistani, M., and Karimpour, A., "Optimal tracking control for linear discrete-time systems using reinforcement learning," *52nd IEEE Conference on Decision and Control*, 2013, pp. 3845–3850. doi:10.1109/CDC.2013.6760476.

[45] Kiumarsi, B., and Lewis, F. L., "Actor–Critic-Based Optimal Tracking for Partially Unknown Nonlinear Discrete-Time Systems," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 1, 2015, pp. 140–151. doi:10.1109/TNNLS.2014.2358227.

[46] van Kampen, E., Chu, Q. P., and Mulder, J. A., "Continuous Adaptive Critic Flight Control Aided with Approximated Plant Dynamics," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006. doi:10.2514/6.2006-6429.

[47] Grondman, I., Vaandrager, M., Busoniu, L., Babuska, R., and Schuitema, E., "Efficient Model Learning Methods for Actor–Critic Control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 42, No. 3, 2012, pp. 591–602. doi:10.1109/TSMCB.2011.2170565.

[48] Kiumarsi, B., Lewis, F. L., Naghibi-Sistani, M., and Karimpour, A., "Optimal Tracking Control of Unknown Discrete-Time Linear Systems Using Input-Output Measured Data," *IEEE Transactions on Cybernetics*, Vol. 45, No. 12, 2015, pp. 2770–2779. doi:10.1109/TCYB.2014.2384016.

[49] Molenkamp, D., van Kampen, E., de Visser, C. C., and Chu, Q. P., "Intelligent Controller Selection for Aggressive Quadrotor Manoeuvring," *AIAA Information Systems-AIAA Infotech @ Aerospace*, 2017. doi:10.2514/6.2017-1068.

[50] Mahony, R., Kumar, V., and Corke, P., "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics Automation Magazine*, Vol. 19, No. 3, 2012, pp. 20–32. doi:10.1109/MRA.2012.2206474.