2022 30th Mediterranean Conference on Control and Automation (MED)
June 28 - July 1, 2022. Athens, Greece

2022 30th Mediterranean Conference on Control and Automation (MED) | 978-1-6654-0673-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/MED54222.2022.9837155

On Asymptotic Stability of Nonlinear Systems with Deep Reinforcement Learning Controllers

Gioacchino Manfredi, Luca De Cicco, and Saverio Mascolo

***Abstract*— Controlling systems with learning-based control strategies is attracting the interest of the research community due to the advantages that machine learning offers, such as the possibility of controlling nonlinear systems that would be hard to control with conventional techniques, the possibility of controlling systems whose model is not available and so on. Reinforcement Learning (RL) and Deep Neural Networks (DNN) can be merged to obtain Deep Reinforcement Learning (DRL) control strategies. Yet, such new approaches are implemented only in few real-world applications since classical DRL control policies cannot guarantee asymptotic stability, which is a key requirement to guarantee safety. In this work, we propose a framework that, after extracting the DRL control policy, tries to synthesise a Lyapunov function that certifies the asymptotic stability of the system controlled with such a policy. We also show that our framework paves the way for safety guarantees that are often necessary when deriving a control policy. Results show that Lyapunov functions can be synthesised for the considered benchmark systems, thus ensuring asymptotic stability. Furthermore, the corresponding regions of attraction prove the quality of DRL control policies wrt other state-of-the-art learning-based controls.**

***Index Terms*— Lyapunov stability, Regions of attraction, Deep Reinforcement Learning, Actor-Critic**

I. Introduction

Deep Reinforcement Learning (DRL) methods aim at learning control policies through interaction with an environment. However, one of the prominent reasons that hinders the adoption of such learning-based approaches in real control applications is safety [1]. Despite the promising performance DRL control policies can achieve, most of the systems employed in reality are safety-critical, usually because of their interaction with human beings and/or with equipment that could be damaged when unsuitable control actions are taken by the control policy.

Therefore, it is important to ensure the learning process produces *safe control policies*. In general, a state is considered safe if system trajectories are bounded within a region and eventually converge asymptotically to the equilibrium point under a given control policy. One of the downsides of DRL methods applied to nonlinear continuous time systems is that there is no guarantee that the learned control policy always stabilises the system as prescribed. This is due to the fact that, during the training phase, the DRL algorithm cannot exhaustively explore all possible states since they are infinite. As a consequence, when the trained control policy is actually deployed, it could occur that some unexplored states are visited, in which case the system dynamics may diverge.

One commonly employed approach is to linearise the system dynamics around an equilibrium point and to use the theory of linear systems to compute a linear feedback controller (f.i., using a Linear Quadratic Regulator) that guarantees stability in a small neighbourhood of the equilibrium point where the linear approximation remains valid. Lyapunov methods are adopted to derive controllers valid outside the small neighbourhood of the equilibrium point. One popular application of these methods rely on polynomial approximations of the system dynamics and seeks sum-of-squares (SOS) polynomials as Lyapunov functions using semidefinite programming (SDP) [2], [3]. Nevertheless, we end up again with an approximation of the system and therefore a restriction on the control.

In this work, the goal is to check whether the control policy obtained with a DRL algorithm makes the system asymptotically stable. In particular, we focus on deriving stability certificates to prove the asymptotic stability of nonlinear systems at an equilibrium point in the presence of *deterministic control policies* obtained with DRL. When a control policy is derived with a DRL technique to control a continuous-time system without any approximation of its dynamics, there is no guarantee that such a policy is always able to successfully control the system. This controller can be plugged into the nonlinear model of the system and then a *Lyapunov Neural Network* (LNN) can be implemented to synthesise a Lyapunov function, which provides a stability certificate for the system. In our framework, a DRL module derives the best feedback control possible given the system dynamics and a reward function. Concerning the Lyapunov function synthesis, we adopt a *Learner-Verifier* approach, which has been proposed recently [4], [5]. In particular, the learner trains an LNN over a set of sample states with stochastic gradient descent to tune proper parameters for a Lyapunov candidate in such a way that a specific cost–representative of the violation of the Lyapunov conditions–is minimised. The verifier takes the DDPG control policy and the Lyapunov candidate from the learner and verifies whether the candidate is actually a Lyapunov function. If it is not the case, the verifier adds a certain number of counterexamples to the set of sample states and feeds them back to the learner so that the procedure iterates until a Lyapunov function is found or a maximum number of iterations is reached. It is important to point out that in general it is not easy to verify the Lyapunov conditions and that for this reason we employ a sound decision procedure using Satisfiability Modulo Theory

This work has been partially supported by the Italian Ministry of Universities and Research through the MAIA project (ARS01_00353). The authors are with the Dipartimento di Ingegneria Elettrica e dell'Informazione at Politecnico di Bari, Via Orabona 4, 70125, Bari, Italy Emails: name.surname@poliba.it.

(SMT) [6] called δ-complete decision procedure [7]. It provides guarantees of correctness of the Lyapunov function found in terms of fulfilment of the Lyapunov conditions. This paper makes the following contributions: (i) we provide a learning-based methodology to prove asymptotic–and not only practical–stability at the equilibrium point of a nonlinear system with a deterministic DRL control policy; (ii) we compute the region of attraction of a system at an equilibrium point with a deterministic DRL feedback control policy, thus bounding the trajectories of the system when the initial state lies inside the region; (iii) we show that a DRL feedback controller can ensure wider regions of attraction compared to other state-of-the-art controllers.

II. Related work

In [8], an approach to learn safety certificates for nonlinear discrete-time closed-loop dynamical systems is presented. In essence, sampling methods are used to verify the Lyapunov conditions and a manual design of the neural network is employed to derive the Lyapunov function given a controller. This method, along with [9], [10], [11], does not provide formal numerical soundness. Our approach first derives a control policy using a DRL algorithm and then relies on a generic feed-forward network to obtain a Lyapunov function with a *Learner-Verifier* method, which is formally sound because it is based on SMT solvers. Mehrjou *et al.* [12] provide an improvement of the algorithm in [8] to enlarge the region of attraction only of a specific class of systems.

An interesting approach is presented in [4], in which the authors leverage feed-forward neural networks and SMT solvers to synthesise both the control and the Lyapunov function. However, they guarantee Lagrange–or practical–stability, i.e., stability is not guaranteed in a neighbourhood of the equilibrium point. Conversely, in this work we aim at guaranteeing full asymptotic stability and compare our results with the aforementioned work to show that we are still able to synthesise Lyapunov functions and that the controller obtained with DRL techniques ensures even larger regions of attraction. Abate *et al.* [5] propose a method for the formal synthesis of LNNs to ensure full asymptotic stability of autonomous nonlinear systems. A polynomial Lyapunov function is derived and comparisons with other approaches prove the efficiency of the method in terms of computation time needed to synthesise a Lyapunov function. The focus of this work is mainly computation time efficiency for autonomous nonlinear systems without analysing the regions of attraction. Our method is based on the computation of a DRL control policy to prove its suitability in terms of asymptotic stability and on the analysis of the regions of attraction compared to other methodologies.

III. Preliminaries

Before describing the procedure to obtain a deterministic DRL controller and to synthesise a Lyapunov function that certifies its stability, we provide in the following the relevant preliminary background and notation used in this work.

Lyapunov stability theory. Consider the continuous-time time-invariant nonlinear system:

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

where $f : \mathscr{D} \to \mathbb{R}^n$ is a Lipschitz-continuous vector field and $\mathscr{D} \subseteq \mathbb{R}^n$ is the domain of the system containing all the states $x(t)$ and such that $0 \in \mathscr{D}$. The continuous function $u : \mathscr{D} \to \mathbb{R}^p$ is the static state-feedback control policy to be learned, thus we can write $\dot{x} = f(x, u(x))$. Throughout this work, we will consider w.l.o.g. that $x = 0$ is an equilibrium point for system (1).

According to a well-known theorem in [13], let $f(x, u(x))$ be a locally Lipschitz continuous vector filed with an equilibrium point at the origin. Let $V : \mathscr{D} \to \mathbb{R}$ be a continuously differentiable function such that $V(0) = 0$, $V(x) > 0$ $\forall x \in \mathscr{D} \setminus \{0\}$ and $\dot{V}(x) < 0$ $\forall x \in \mathscr{D} \setminus \{0\}$. Then, the origin is an asymptotically stable equilibrium point of the system and V is called a Lyapunov function.

Notice that $\dot{V}(x)$ represents the Lie derivative of $V(x)$ over the vector field $f(x, u(x))$ and is defined as follows:

$$\dot{V}(x) = \nabla_x V(x) \cdot f(x, u(x)) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x, u(x)). \tag{2}$$

In general, it is not easy to find a Lyapunov function and there exists no general method to construct Lyapunov functions [14]. These functions are important tools to estimate the regions of attraction at an equilibrium point of a general nonlinear dynamical system [13]. Suppose system (1) is asymptotically stable at the origin under the control law $u(x)$ and let $V(x)$ be a Lyapunov function for system (1) in $\mathscr{D}$. A region of attraction $\mathcal{S}$ is an invariant subset of $\mathscr{D}$ that contains the origin, i.e., if the initial state of the system belongs to $\mathcal{S}$, then the system trajectories always stay inside $\mathcal{S}$. All level surfaces of $V(x)$ contained in $\mathscr{D}$ are regions of attraction for the system, i.e., for some $c > 0$, $\mathcal{S} = \{x \in \mathbb{R}^n | V(x) \leq c\} \subseteq \mathscr{D}$.

Neural Networks. Neural networks are employed in a wide range of applications. They have been used also to enhance Reinforcement Learning (RL) agents to derive policies for the control of continuous-time systems. To this end, Deep Reinforcement Learning (DRL) allows us to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces. In this work, we will consider two DRL algorithms with deterministic control policy: *Deep Deterministic Policy Gradient* (DDPG) [15] and *Twin Delayed Deep Deterministic Policy Gradient* (TD3) [16]. The DDPG algorithm is one of the most widely used learning technique applied to control systems. It employs an off-policy Actor-Critic architecture to learn control policies in continuous action spaces. In a nutshell, the actor chooses an action to perform on the environment and observes the reward and the new state. Such a transition is stored in the *replay buffer*. The critic measures how good the action chosen by the Actor was through the Q-function. Then, it samples a number of transitions from the replay buffer to minimise a loss function and to update

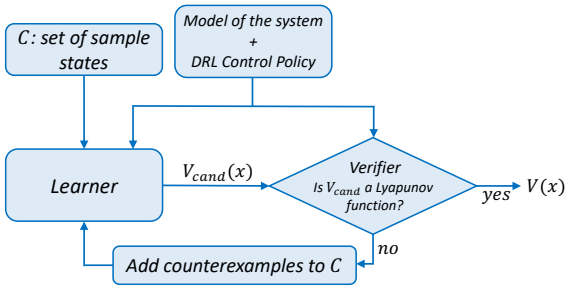

Fig. 1: Proposed architecture

the actor policy. The TD3 algorithm improves the DDPG by introducing two Q-functions instead of just one and uses the smallest of the two Q-values to compute the loss function.

Neural networks are also efficient regressors. This is a particularly useful property that allows us to approximate Lyapunov functions, usually called Lyapunov Neural Networks (LNNs). In this work, we implement a feed-forward neural network as a LNN whose input has the dimension of the state space of the system, say n, the output layer has dimension 1, and in between there are m hidden layers with $h_1, h_2, \ldots, h_m$ neurons. Supposing fully connected layers, the weights are contained in $m+1$ matrices as follows: $W_1 \in \mathbb{R}^{h_1 \times n}$ for those weights from the input to the first hidden layer, $W_2 \in \mathbb{R}^{h_2 \times h_1}$ and so on until W_{m+1}. It is possible to have also an additive bias in each hidden neuron, therefore $B_1 \in \mathbb{R}^{h_1}$, $B_2 \in \mathbb{R}^{h_2}$ and so on. Every layer can have an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ applied to each neuron. Hence, if x_0 is the input state, it results that:

$$l_1 = \sigma_0(W_1 x_0), \tag{3}$$

$$l_i = \sigma_i(W_i l_{i-1}), \;\; i = 2, \ldots, m, \tag{4}$$

where σ_i is computed element-wise and l_i is the output of the i-th layer. The output of the feed-forward network represents the value of the Lyapunov function in the input state:

$$V(x_0) = \sigma_{m+1}(W_{m+1} l_m). \tag{5}$$

Once this network is trained to meet the Lyapunov conditions, we end up with an LNN that represents a Lyapunov function $V : \mathscr{D} \to \mathbb{R}$ proving the asymptotic stability of the controlled system at the equilibrium point in the domain $\mathscr{D}$.

IV. The Proposed Architecture

In this section, we present an architecture that extracts a DRL control policy and proves–if it is the case–full asymptotic stability of the system through the synthesis of a Lyapunov function.

Following a counterexample-guided approach such as the one employed in [5], Fig. 1 shows the proposed architecture composed of three main modules: the *model of the system with the DRL Control Policy*, the *Learner* and the *Verifier*.

DRL Control Policy: it employs the model of the system and is in general based on a DRL algorithm that outputs the control policy $u(x)$ after a training phase. Notice that, in order to extract the control policy as a function of the states, the DRL algorithm needs to be deterministic, i.e., the output of the algorithm must provide an action when an input state is presented rather than an average value and a standard deviation of the action, which would imply stochasticity.

Learner: it trains a neural network on the basis of an input set $\mathcal{C}$ containing a certain number of sample states. The trained network represents a candidate LNN whose output given a state is the value of the candidate Lyapunov function in that state. The neural network is trained by minimising a cost function using steepest gradient descent. The cost function must necessarily take into account the Lyapunov conditions.We formulate such a cost as in [4]:

$$L(x) = \frac{1}{N}\sum_{1}^{N}(\max(-V(x_i), 0) + \max(0, \dot{V}(x_i))) + V(0)^2, \tag{6}$$

where x_i is the i-th sample of the state and is such that $x_i \in \mathcal{C}$ with $|\mathcal{C}| = N$. Notice that for each sample, the cost is 0 when $V(x) > 0$, i.e. when the first Lyapunov condition is met, and $\dot{V}(x) < 0$, i.e. when the second Lyapunov condition is met. If $V(x) < 0$, then the first condition is violated and such a violation is penalised of a term equal to $-V(x)$. The same can be said for the condition $\dot{V}(x) < 0$. The average over all the samples plus $V(0)^2$ represents our cost, which is minimised using the steepest gradient descent. Obviously, when $V(x)$ is a Lyapunov function, $L(x) = 0$, although the contrary is not true. In fact, it could happen that there are some states not included in $\mathcal{C}$ in which the candidate function violates the conditions. It is at this point that the *Verifier* checks if there are some states that violate the Lyapunov conditions and adds them to $\mathcal{C}$.

Verifier: it makes use of a solver to find–if it is the case–states that violate the Lyapunov conditions wrt a candidate Lyapunov function. Satisfiability Modulo Theory (SMT) solvers are powerful tools to check the satisfiability of first-order logic formulae. Such solvers are formally sound and therefore provide guarantees that are equivalent to giving analytical proofs. For this reason, we employ SMT solvers to check candidate Lyapunov functions. As already said, a Lyapunov function has to meet the following conditions:

$$V(0) = 0 \wedge V(x) > 0 \wedge \dot{V}(x) < 0, \;\; \forall x \in \mathscr{D} \backslash \{0\}. \tag{7}$$

In order to make use of SMT solvers, we derive the so called *dual falsification problem*, which is needed in formal verification. This problem is simply the negation of (7),

$$\exists x \in \mathscr{D} \backslash \{0\} : V(x) \leq 0 \vee \dot{V}(x) \geq 0. \tag{8}$$

If the falsification condition is true for some nonzero x in $\mathscr{D}$, then $V(x)$ is not a Lyapunov function since there is at least one state in the domain that violates the Lyapunov conditions. On the contrary, if it is false, it means that $V(x)$

is a Lyapunov function in $\mathscr{D}$. To solve problem (8), it is necessary to globally minimise non-convex functions such as Lie derivatives, which is an NP-hard problem. For this reason, the *Verifier* relies on an open-source SMT solver for nonlinear formulas over the reals called dReal [17]. This solver provides formally sound solutions to problem (8), i.e., if a solution exists, it is always found. Such a guarantee comes with the δ-*completeness* property. For the sake of clarity, let us provide the following definitions [7].

Definition 1 (SMT problem)**.** An SMT problem is the problem of determining whether an SMT formula, which is a first-order logic formula, is satisfiable.

Definition 2 (δ-completeness)**.** Let φ be an SMT formula and δ a positive rational number, a decision procedure $\mathcal{P}$ is δ-complete if it either determines that φ is not satisfiable or that the δ-*weakening* of φ is satisfiable.

The δ-*weakening* of φ is a numerical relaxation of the original formula. In our case, a δ-complete procedure $\mathcal{P}$ that verifies the formula in (8) is what we need because if a formula is satisfiable, then its δ-relaxation is always satisfiable. On the contrary, if the δ-relaxation of the formula is satisfiable, then it could be that either the formula itself is satisfiable or it is not satisfiable. Therefore, we can guarantee the check on the Lyapunov conditions because if the δ-relaxation of (8) is not satisfiable, then also (8) itself is not satisfiable, thus implying that $V(x)$ satisfies the Lyapunov conditions and is a Lyapunov function. However, it could happen that the δ-relaxation of (8) is satisfiable. This provides no guarantee on (8) and the procedure gives counterexamples that are solutions of the relaxed formula. If the not-relaxed formula is not satisfiable, then the counterexamples obtained are just spurious solutions that do not pose any issue since this would translate in the synthesis of a more conservative candidate Lyapunov function.

V. Results

In this section we provide experimental results of the proposed approach gathered on two test cases considering the most widely implemented deterministic DRL algorithms: DDPG and TD3. In the *Learner* we set up a feed-forward neural network with one hidden layer while in the *Verifier* we use dReal [17] as the SMT solver.

Case 1: Inverted Pendulum. In the first test case we consider the problem of controlling an inverted pendulum whose control goal is to balance it in the upright position with zero angle and zero angular velocity. The control variable is the torque that can be applied to the hinge of the pendulum whereas the state is represented by the angle, measured in radians, and the angular velocity, measured in radians per second. Finally, we impose a domain $\mathscr{D} = \{x = (\theta, \dot{\theta}) \in \mathbb{R}^2 | \, ||x||_2 \leq 6\}$.

Let us start by considering a DDPG control policy. As a first step, we train the Actor and Critic until the cost function is minimised and the control goal is achieved. To the purpose, we considered an Actor with the two states as inputs, no hidden layers, and one output with linear activation. In other words, we considered a controller of the form $u(x) = Kx$, thus obtaining: $u(\theta, \dot{\theta}) = -1.2616117\,\theta - 0.29633152\,\dot{\theta}$.

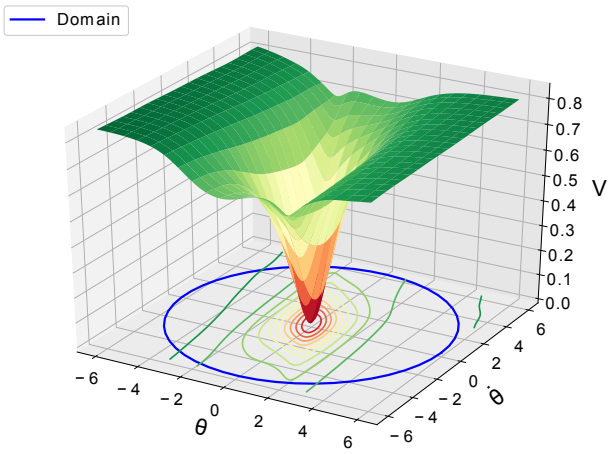

Fig. 2: Lyapunov function for Inverted Pendulum with DDPG control

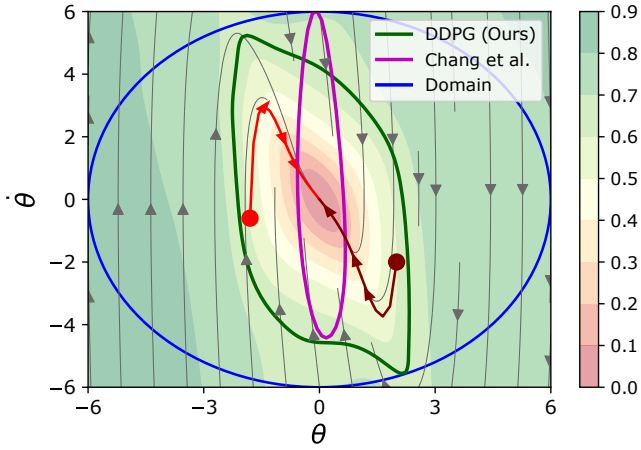

Fig. 3: Region of Attraction for Inverted Pendulum with DDPG control

At this point, a set $\mathcal{C}$ of 500 random states sampled from the domain is created and, on the basis of such a set and of the model of the system, the *Learner* derives a candidate Lyapunov function that is passed to the *Verifier*. The latter uses the SMT solver to check if the Lyapunov conditions are verified. In the case they are not, the solver augments $\mathcal{C}$ with some states that violate the Lyapunov conditions in such a way that, at the next iteration, the *Learner* can adjust the Lyapunov candidate function to satisfy the conditions. Notice that, according to what we have said in Section IV, it could happen that the *Learner* finds a Lyapunov function but the *Verifier* does not recognise it due to the δ-completeness property of the SMT solver. In this case, the counterexamples added to $\mathcal{C}$ are spurious data that are not detrimental to the *Learner*: they can help it find a more suitable Lyapunov function that, eventually, will pass the *Verifier*'s check.

Fig. 2 shows the Lyapunov function $V(\theta, \dot{\theta})$ found by the *Learner* and valid inside $\mathscr{D}$. Qualitatively, one can see that

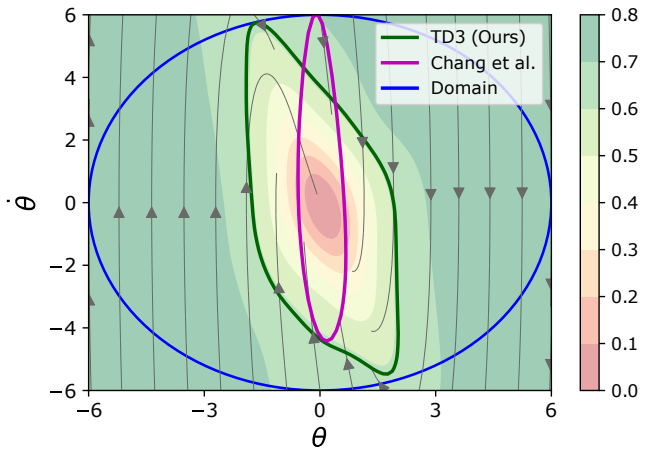

Fig. 4: Region of Attraction for Inverted Pendulum with TD3 control

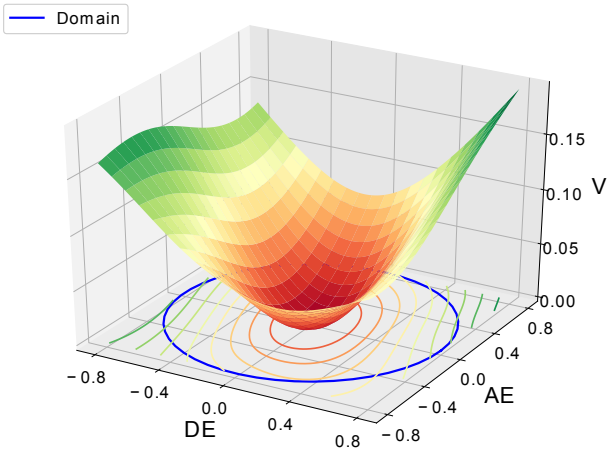

Fig. 5: Lyapunov function for Bicycle circle tracking with DDPG control

this function is 0 only when $(\theta, \dot{\theta}) = (0, 0)$ and is strictly positive elsewhere in the domain (blue circle in the figure). The *Verifier* formally ensures this along with the condition on the gradient. At this point we have the guarantee that the DDPG control policy asymptotically stabilises the nonlinear system also when unexplored states are visited.

At this point, we can consider the widest region of attraction in the domain to have an idea of the stabilising performance guaranteed by the controller. Fig. 3 outlines the region of attraction obtained with the synthesised Lyapunov function (green curve) and the phase portrait of the closed-loop system (grey arrows) showing that the equilibrium point in zero is a stable point. For a better understanding, if we pick two random initial states (large dots in the figure) within the region of attraction and let the system evolve, the trajectories will never leave the region until convergence. Referring to Fig. 3, if the initial state lies outside the highlighted green region, then the system trajectories can evolve arbitrarily far before converging to zero. This represents an important insight towards tackling safety issues concerning DRL control strategies. In fact, safety is the main reason why such approaches meet few practical applications. Therefore, synthesising a Lyapunov function for the DDPG controller in the case of an inverted pendulum not only guarantees asymptotic stability of the controlled system, but also allows us to find a region of attraction, which–provided that the initial state is in it–ensures that no state outside it is visited, thus preventing the system from reaching possible unsafe states.

Fig. 3 also shows the region of attraction found with the controller developed by Chang *et al.* [4] for the same example (magenta curve). This linear controller is synthesised by the same neural network that generates the candidate Lyapunov function. Even if the goal of the present work is different, let us make some considerations on the kind of controllers found. From the figure, it is clear that the DDPG controller guarantees a much wider region of attraction wrt to the other controller. In other words, it would prove a more performing control. Moreover, the control in [4] guarantees in practice Lagrange stability, thus excluding a neighbourhood of the equilibrium point when synthesising the Lyapunov function. On the other hand, our analysis ensures full stability at the equilibrium point yet with a wider region of attraction.

In the same way, we evaluated the TD3 control policy, obtaining essentially similar results. In particular, after finding the Lyapunov function, which is omitted here for space constraints but which has a similar shape as in Fig. 2, we show the corresponding region of attraction in Fig. 4. Also in this case, the TD3 controller is able to guarantee full stability and still a wider region of attraction compared to the case described in the literature.

Case 2: Bicycle circle tracking. In this example, we consider the case of a bicycle, running at constant speed, that has to track a unit circle. The control variable is the steering angle of the front wheel whereas the state variables are represented by the Distance Error (DE), i.e., the distance measured in meters between the rear wheel and the circle, and the Angle Error (AE), i.e., the difference measured in radians between the angle of the rear wheel and the angle of the tangent to the circle in the closest point to the wheel (see [4] for more details). The control goal is to steer the state to zero. In this case, we set a domain $\mathscr{D} = \{x = (DE, AE) \in \mathbb{R}^2 | \, ||x||_2 \leq 0.8\}$.

Let us first evaluate the DDPG algorithm. Notice that in this case we set up the Actor with an input layer of dimension 2, two hidden layers with 256 neurons and an output layer of dimension 1, all with linear activation. After training a control policy able to stabilise the system in the testing phase, we leverage the framework displayed in Fig. 1 to try to synthesise a Lyapunov function that proves asymptotic stability of the controlled system. The *Learner* managed to find a Lyapunov function, which is displayed in Fig. 5. From its level sets, we are able to find a region of attraction for the system (green curve) as shown in Fig. 6.

Finally, we carried out the same analysis for the TD3 algorithm. A Lypunov function–similar to the DDPG case–was successfully found. Fig. 7 shows the region of attraction

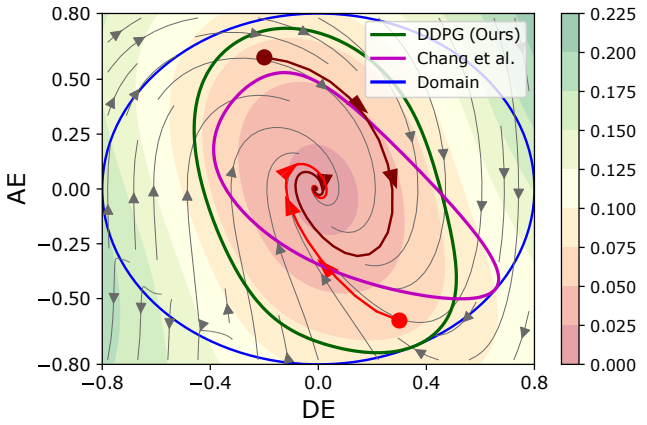

Fig. 6: Region of attraction for Bicycle circle tracking with DDPG control

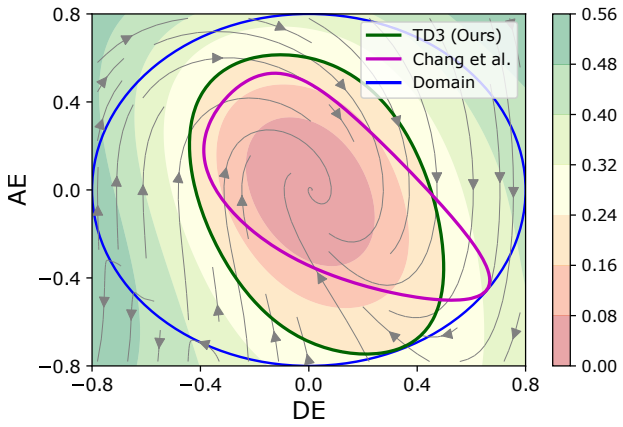

Fig. 7: Region of attraction for Bicycle circle tracking with TD3 control

from which the same conclusions as the DDPG can be drawn.

In each of the considered benchmark control systems, the region of attraction obtained with a DRL controller is also way wider than the region derived with classical controllers such as LQR or SOS (see [4] for a visual comparison).

As already explained, in general, if no Lyapunov function is found, then one cannot establish any conclusion about the system's stability. It could happen that the LNN either finds a Lyapunov function or reaches a maximum number of iterations. The latter case simply implies the network was not able to find a Lyapunov function but it could exist. Moreover, the more the system is complex, the wider the networks employed in the DDPG or TD3 to control it. However, as long as the activation functions on the neurons are linear, the resulting controller is linear or affine in the state and therefore the LNN can be easily deployed. In the case of nonlinear activation functions, the time complexity of the proposed procedure can considerably increase. The time complexity also depends on the complexity of the Lyapunov function we try to synthesise. In this work, we considered linear or affine controllers and a LNN with one hidden layer, 64 neurons and a *tanh* activation function. In the simulations, the time needed for the synthesis ranges from 132 to 431 seconds when using Google Colab. In any case, this is an offline procedure, whose goal is to certify the asymptotic stability of the controlled system without taking into account the time required to accomplish such a task.

VI. Conclusions

This work proposes a methodology that can be employed to prove asymptotic stability of nonlinear systems controlled via deterministic DRL algorithms. In our analysis, we consider two such algorithms: DDPG and TD3. After their training phase, we extract the control policy and implement a *Learner-Verifier* approach to synthesise a Lyapunov function with the help of SMT solvers. The results obtained on two benchmark control systems show the effectiveness of our approach, ensuring not only asymptotic stability of DRL algorithms but also important considerations on safety. An interesting future research direction could be testing the framework on real control systems, but also developing an equivalent approach for stochastic DRL algorithms.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[2] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

[3] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Control applications of sum of squares programming," in *Positive Polynomials in Control*. Springer, 2005, pp. 3–22.

[4] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," *arXiv preprint arXiv:2005.00611*, 2020.

[5] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[6] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of model checking*. Springer, 2018, pp. 305–343.

[7] S. Gao, J. Avigad, and E. M. Clarke, "δ-complete decision procedures for satisfiability over the reals," in *International Joint Conference on Automated Reasoning*. Springer, 2012, pp. 286–300.

[8] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*, 2018, pp. 466–476.

[9] M. Mittal, M. Gallieri, A. Quaglino, S. S. M. Salehian, and J. Koutník, "Neural lyapunov model predictive control," *arXiv preprint arXiv:2002.10451*, 2020.

[10] N. Noroozi, P. Karimaghaee, F. Safaei, and H. Javadi, "Generation of lyapunov functions by neural networks," in *Proceedings of the World Congress on Engineering*, vol. 2008, 2008.

[11] V. Petridis and S. Petridis, "Construction of neural network based lyapunov functions," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 5059–5065.

[12] A. Mehrjou, M. Ghavamzadeh, and B. Schölkopf, "Neural lyapunov redesign," *arXiv preprint arXiv:2006.03947*, 2020.

[13] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, the book can be consulted by contacting: PH-AID: Wallet, Lionel. [Online]. Available: https://cds.cern.ch/record/1173048

[14] P. Giesl and S. Hafstein, "Review on computational methods for lyapunov functions," *Discrete & Continuous Dynamical Systems-B*, vol. 20, no. 8, p. 2291, 2015.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[17] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *International conference on automated deduction*. Springer, 2013, pp. 208–214.