Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-37 (2022) 339–344

Curriculum-based reinforcement learning for path tracking in an underactuated nonholonomic system ⋆

Prashanth Chivkula * **Colin Rodwell** *
Phanindra Tallapragada *

* *Department of Mechanical Engineering, Clemson University (e-mail: ptallap@clemson.edu)*

Abstract: Underactuated mechanical systems with nonholonomic constraints find applications in bioinspired robotics, such as snake-like robots and more recently in fish-like aquatic robots. Animal locomotion suggests that in such bioinspired robots, gaits or cyclic changes in kinematics or shape variables lead to efficient and agile motion. Path tracking in such nonholonomic systems that are not purely kinematic can be a challenging problem. In this paper we consider the problem of path tracking by a modified Chaplygin sleigh with a 'tail' which is a four degree of freedom nonholonomic system, possessing a single internal reaction wheel as an actuator. We develop a curriculum based deep Reinforcement Learning (RL) optimal control approach for simultaneous velocity and path tracking for this system. The curriculum based learning approach first leads to a policy for optimal tracking of limit cycles in a reduced velocity space and then in a next step to track a path. This curriculum approach allows an RL agent to learn the 'mechanics on invariant manifolds' of the system and can be a useful approach in the motion control of high degree of freedom robots with increasing model complexity.

Keywords: Nonholonomic Systems, Reinforcement Learning, Path Tracking

1. INTRODUCTION

Nonholonomic constraints are common in the dynamics of mobile robots, see Li and Canny [1993], Laumond et al. [1994]. Robots with wheels are a classic example. More unusual examples of nonholonomic constraints occur in bioinpsired snake-like robots, see for example Wright et al. [2007], Liljebäck et al. [2012], or even models for aquatic robots as has been demonstrated for example in Tallapragada [2015], Tallapragada and Kelly [2016]. Animal locomotion suggests that in such bioinspired robots, gaits or cyclic changes in kinematics or shape variables lead to efficient and agile motion. Path tracking in such nonholonomic systems that are not purely kinematic such as in Laumond et al. [1994], Murray [1993] can be a challenging problem. This paper investigates the problem of path tracking for a modified Chaplygin sleigh with a 'tail' which is a four degree of freedom nonholonomic system, possessing a single internal reaction wheel as an actuator. The Chaplygin sleigh is a well known planar three degree of freedom nonholonomic system, see Bloch [2003] and for a historical review Borisov and Mamaev [2003]. Further, this system is underactuated, possessing a single internal reaction wheel that can deliver a torque on the leading segment. Cyclic motion of the reaction wheel generates periodic torques that lead to limit cycles in a reduced velocity space as shown in Fedonyuk and Tallapragada [2018] reminiscent of central pattern generators (Guertin [2009]) seen in biological systems. Such an underactuated, internally actuated nonholonomic system is a useful model to investigate gaits and resonances in both ground based and aquatic mobile robots, but can present challenges in motion control.

Parametric optimization approaches have been used for gait optimization in the context of other bioinspired robots with periodic motions such as for a quadruped robot, using optimization routines such as genetic algorithms (Chernova and Veloso [2004]) or policy gradients (Kohl and Stone [2004]). Other work has used a discretized action space to numerically optimize a sequence of actions that results in an optimal gait (Belter and Skrzypczyński [2010]). By contrast, we seek a feedback controller in a continuous state-action space that generates the desired gait. Modern data-driven control strategies, such as Reinforcement Learning (RL), offer a new numerical framework to develop optimal control for systems where analytical approaches are intractable. By considering the control policy (or *actor*) as an arbitrary nonlinear state feedback, control policies can generally be developed without a-priori understanding of the structure of the optimal control. However, many RL algorithms, such as the Go algorithm AlphaGo (Silver et al. [2016]), discard this generality and first train the policies to mimic the actions of human experts in a supervised manner before beginning to use RL, which greatly reduces training time. The new field of curriculum learning expands this idea to multiple steps of training: an RL policy is trained on successively more difficult tasks, each of which converges quickly, as opposed to very slow convergence directly to the complex task, or

⋆ This work was supported by grant 2021612 from the National Science Foundation and grant 13204704 from the Office of Naval Research.

10.1016/j.ifacol.2022.11.207

potentially no convergence at all (Narvekar et al. [2020]). Curriculum-based RL learning has recently had success in learning a gait for a bipedal robot (Rodriguez and Behnke [2021]) where joint kinematics are fully prescribed by the policy. In contrast, we extend curriculum learning to a highly underactuated system where the input is a single torque.

In this work, we extend the ideas of curriculum learning to policy learning on invariant manifolds. Specifically, we train a policy to control a two-link Chaplygin sleigh model on a limit cycle in a four-dimensional reduced velocity space with the control input defined as a torque on the front link exerted by an onboard reaction wheel. The resulting limit cycle is trained to track a specified average velocity and heading angle. This is achieved by first *pre-training* the policy to mimic a prescribed sine wave, which is known to result in a sub-optimal periodic gait. This policy is then updated using the Deep Deterministic Policy Gradient (DDPG) algorithm (Silver et al. [2014], Lillicrap et al. [2016]) to update the policy towards the target velocity and heading angle. The target velocity is then taken as a state and allowed to vary with time, at first slightly and later performing increasingly large oscillations as the policy learns. This trained policy is then integrated into a pure-pursuit algorithm to perform path tracking.

2. SYSTEM MODEL

A schematic of the two-link Chaplygin sleigh is shown in Fig. 1, where a and b represent the length of each of the segments. The front segment (shown in orange) will referred to as the 'head' and the rear segment (in blue) will be referred to as the 'tail'. The body frame $x_b - y_b$ is collocated at the joint C between the two links, with x_b aligned with the longitudinal axis of the tail segment. The configuration space of the system is $Q = SE2 \times S^1$. The generalised coordinates and velocities of the system can be written as $q = [x, y, \theta, \delta]$ and $\dot{q} = [\dot{x}, \dot{y}, \omega_1, \omega_2]$ respectively. Point P is modelled as having a small knife edge or a wheel that cannot slip in the transverse direction (along y_b direction). This is a nonholonomic constraint and can be written as

$$u_y = -\sin\theta dx + \cos\theta dy + b\dot{\theta} = 0. \tag{1}$$

where u_y is the transverse velocity at point P in the body frame. The stiffness of the joint is modelled by a torsional spring of stiffness K. The sleigh is actuated by a torque $\tau(t) = A\sin\Omega t$ acting on the head link. This torque is assumed to be generated by the angular acceleration of an internal reaction wheel. The mass of the tail link is denoted by m_c and that of the head link is denoted m_r. I_c and I_r are the moments of inertia of the two links. The angle made by the sleigh with respect to the inertial frame of reference is denoted θ, and $\delta \in S^1$ is the relative inter-link angle between the tail and head, with $\delta = 0$ when the longitudinal axes of the head and tail are aligned. The governing dynamics of this system are reviewed here briefly and further details can be found in Rodwell and Tallapragada [2022].

The Lagrangian of the system is $\mathcal{L} = \frac{1}{2}\dot{q}^T\mathcal{M}(q)\dot{q} - \mathcal{V}(\delta)$ where $\mathcal{T}(q,\dot{q} = \frac{1}{2}\dot{q}^T\mathcal{M}(q)\dot{q}$ is the kinetic energy, $\mathcal{M}$ represents the mass matrix and $\mathcal{V}(\delta) = \frac{1}{2}K\delta^2$ represents elastic the potential energy of the system. The kinetic

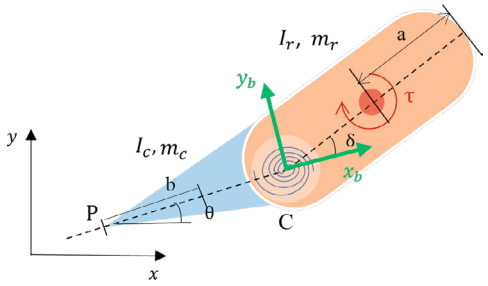


Fig. 1. Multi-Link Chaplygin sleigh

energy can be written explicitly as $\mathcal{T}(q,\dot{q}) = \frac{1}{2}m_c(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}I_c\dot{\theta}^2 + \frac{1}{2}m_r({\dot{x}_r}^2 + {\dot{y}_r}^2) + \frac{1}{2}I_r(\dot{\theta}+\dot{\delta})^2$ where x_r and y_r are the coordinates of the center of mass of the head link, $\mathcal{R} = \frac{1}{2}(c_u(\dot{x}\cos\theta + \dot{y}\sin\theta)^2 + c_{\omega_1}\dot{\theta}^2 + c_{\omega_2}\dot{\delta}^2)$ is the Rayleigh (viscous) dissipation function, c_u is the damping for velocity at the constraint, c_{ω_1} is the rotational damping of the tail link and c_{ω_2} is the interlink rotational damping constant. The Euler-Lagrange equations can be written in the form

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{q}^i}\right) - \frac{\partial\mathcal{L}}{\partial q^i} = \mathcal{W}_{ij}\lambda_j - \frac{\partial\mathcal{R}}{\partial q^i} + \tau_i(t), \tag{2}$$

where λ_j is the Lagrange multiplier for each j constraint and τ_i is the external torque or forces acting on sleigh. Here i corresponds to each of the generalized co-ordinates. After straightforward calculations the Euler Lagrange equations can be written in matrix form as

$$\begin{bmatrix}\mathcal{M} & -\mathcal{W}^T\\ \mathcal{W} & 0\end{bmatrix}\begin{bmatrix}\ddot{q}\\ \lambda\end{bmatrix} = \begin{bmatrix}\mathcal{C}(q,\dot{q})\dot{q}\\ \dot{\mathcal{W}}\dot{q}\end{bmatrix} + \frac{\partial\mathcal{L}}{\partial q} + \tau(t), \tag{3}$$

where $C_{ijk} = \frac{1}{2}\left(\frac{\partial\mathcal{M}_{kj}}{\partial q_i} + \frac{\partial\mathcal{M}_{ki}}{\partial q_j} - \frac{\partial\mathcal{M}_{ij}}{\partial q_k}\right)$ are the Christoffel symbols of the first kind. The equations can be transformed to a body frame as

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \mathbf{R}\begin{bmatrix}u\\ 0\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} = \mathbf{R}\begin{bmatrix}\dot{u}\\ 0\end{bmatrix} + \omega_1 \times \mathbf{R}\begin{bmatrix}u\\ 0\end{bmatrix}, \tag{4}$$

where $\mathbf{R}$ is the rotation matrix transforming body frame velocities to spatial velocities. In order to non-dimensionalize the equations the following substitutions are used: $u' = \frac{u}{l}$, $c'_{\omega_1} = \frac{c_{\omega_1}}{ml^2}$, $c'_u = \frac{c_u}{m}$, $c'_{\omega_2} = \frac{c_{\omega_2}}{ml^2}$, $a = (1-\epsilon)l$, $b = \epsilon l$, $m_r = (1-\epsilon)m$, $m_c = \epsilon m$, $I_c = \gamma m_c(l^2 + 4\epsilon^2 l^2)$, $I_r = \gamma m_r(l^2 + 4(1-\epsilon)^2 l^2)$, $K' = \sqrt{\frac{K}{ml^2}}$, $\tau' = \frac{\tau}{ml^2}$. For convenience the superscripts are dropped from the variables we set $c = c_u = c_{\omega 1} = c_{\omega 2}$ rescaled stiffness $K = 10$ and damping $c = 1$ throughout the paper. Since the Lagrangian $\mathcal{L}$ is $SE2$ invariant, the equations governing the body frame velocities are decoupled from the evolution of configuration variables. The reduced velocity equations of motion are expressed in matrix form as

$$M\begin{bmatrix}\dot{u}\\ \dot{\omega_1}\\ \dot{\omega_2}\end{bmatrix} = \begin{bmatrix}\epsilon(2-\epsilon){\omega_1}^2 + (\omega_1+\omega_2{}^2)(1-\epsilon)^2\cos\delta - c_u u\\ (4\omega_1\omega_2 + 2\omega_2^2)(1-\epsilon)^2\epsilon\sin\delta + (\epsilon(\epsilon-2)\\ -(1-\epsilon)^2\cos\delta)u\omega_1 - c_{\omega_1}\omega_1 + \tau\\ -(1-\epsilon)^2\cos\delta u\omega_1 - 2\epsilon(1-\epsilon)^2\sin\delta{\omega_1}^2 - K\delta\\ -c_{\omega_2}\omega_2 + \tau\end{bmatrix}, \tag{5}$$

where M is a 3×3 symmetric matrix with $M_{11} = 1$, $M_{12} = M_{13} = -(1-\epsilon)^2\sin\delta$, $M_{22} = (-12\gamma - 3)\epsilon + (12\gamma + 7)\epsilon^2 - 4\epsilon^3 + 5\gamma + 1 + 4\epsilon(1-\epsilon^2)\cos\delta$, $M_{23} = M_{32} = (1-\epsilon)(4\epsilon^2\gamma - 8\epsilon\gamma + 1 - 2\epsilon + \epsilon^2 + 5\gamma + 2\epsilon\cos\delta - 2\epsilon^2\cos\delta)$ and $M_{33} = (1-\epsilon)(4\epsilon^2\gamma - 8\epsilon\gamma + 5\gamma + 1 - 2\epsilon + \epsilon^2)$. When the

torque τ is periodic, the dynamical system (5) has stable limit cycle solutions for most parameters (Fedonyuk and Tallapragada [2018]) and sample limit cycle is shown in Fig. 2.

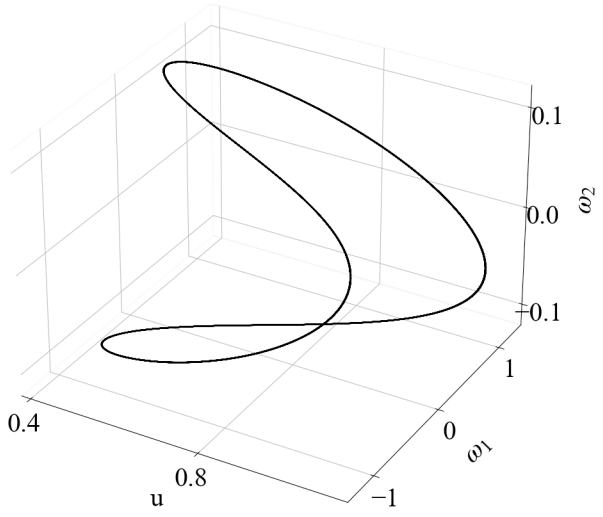


Fig. 2. A limit cycle trajectory in the (u,ω_1,ω_2) space due to a periodic forcing $\tau(t) = 2.5 \sin t$. This trajectory is symmetrical about both ω_1 and ω_2 and generates motion on a serpentine path about a line in (x, y) space.

3. DDPG ALGORITHM

We approach this problem of path tracking by the two link Chaplygin sleigh as a general RL problem where the control input is the torque applied to the head-link. Here our action space is continuous, so instead of artificially discretizing it and losing resolution, we consider a continuous deterministic policy, which can be derived by the DDPG algorithm. DDPG (Lillicrap et al. [2016]) is a policy gradient algorithm where a deterministic policy $\mu(S)$ has an associated *critic* $Q_\mu(S, A)$ which encodes the discounted value of a given state-action pair, allowing the policy gradient to be calculated at every point in the state-space, which can be more data-efficient compared to Monte-Carlo methods, where the policy gradient is calculated over trajectories. Both $Q(S, A)$ and $\mu(S)$ are general functions, which are approximated by neural networks with weights θ_q and θ_μ respectively. Convergence of the DDPG algorithm to a specific set of actor weights θ_μ implies that they represent a (locally) optimal set of weights. Because we have only one actor, we drop the subscript on Q. The Q value of a state-action pair can be calculated by the discounted Bellman equation

$$q_t = r_t + \gamma Q(S_{t+1}, \mu(S_{t+1}|\theta_\mu)|\theta_q), \tag{6}$$

where q_t is the updated estimate of $Q(S_t, A_t)$, r is the reward calculated by a reward function at time step t and $Q(S_{t+1}, \mu(S_{t+1}|\theta_\mu)|\theta_q)$ is the Q-value calculated for the next states and action (S_{t+1}, A_{t+1}) at the next time step. There is a discount factor γ attached to Q_{t+1} which helps to determine the importance of the reward at future time steps. At each time step the critic weights θ_q are updated by stochastic gradient descent such that they minimize the temporal difference loss, which is defined as the mean-squared error over a batch of of times $t \in \{0 \dots T\}$ of $L_{Q,t}(\theta_q)$, where

$$L_{Q,t}(\theta_q) = (Q(S_t, A_t|\theta_q) - q_t)^2. \tag{7}$$

Iteratively updating this approximator of Q using the discounted Bellman equation is known as temporal difference learning, and typically results in convergence to accurate Q values.

The weights of policy $\mu(s|\theta_\mu)$ are updated by a policy gradient. The deterministic policy gradient is the gradient of the critic $Q(S, A)$ with respect to the parameters of the actor network θ_μ (Silver et al. [2014]). The gradient ascent step can instead be written as an optimization that minimizes the loss between the current and desired policy outputs, where the loss at each time is

$$L_{\mu,t}(\theta_\mu) = (\mu(S_t|\theta_\mu) - A'_t)^2 \tag{8}$$

where desired action A'_t is defined as

$$A'_t = \mu(S_t|\theta_\mu) + \alpha \nabla_A Q(S_t, \mu_\theta(S_t|\theta_\mu)|\theta_q) \tag{9}$$

where α is the learning rate. The one dimensional gradient $\nabla_A Q(S, \mu(S|\theta_\mu)|\theta_q)$ can be efficiently calcuated by finite differences, written as

$$\frac{dQ(S, A|\theta_q)}{dA} \approx \frac{Q_(S, A|\theta_q) - Q_{A+dA}(S, A + dA|\theta_q)}{dA}, \tag{10}$$

where dA is chosen to be sufficiently small that the slope of the secant is representative of the tangent slope at A. The optimization of the actor-critic pair is difficult and prone to convergence issues because the Q function is valid for a specific actor, so every update of the actor necessitates retraining the Q function. For this reason, the losses L_Q and L_μ are typically minimized sequentially, and the sequence is repeated until the overall change in both losses over an iteration is low.

4. SLEIGH AGENT AND TRAINING

DDPG requires two deep neural networks (or equivalent deep function approximators) which are illustrated in the 'Training' panel in Fig. 3. There are 6 inputs to the policy network: the states u, ω_1, ω_2, θ, δ, and the target velocity $u'(t)$. The Q network shares the same inputs as the policy network, with an additional input of the control variables, in this case just rotor torque. The aim of training this DDPG agent is to have a controller that can track an average velocity. We later implement a pure-pursuit algorithm, so the policy does not need to itself be able to track a path, it simply needs to regulate the tail angle θ and velocity u to the values specified by the pursuit algorithm. While direct training can accomplish these objectives for this relatively simple system, in this case we attempt a curriculum learning strategy to demonstrate the applicability of a curriculum approach to this type of underactuated nonholonomic system. As defined in Narvekar et al. [2020], this strategy is used to optimize the learning process by optimizing the agent experience to facilitate faster and more data-efficient training.

4.1 Curriculum learning

A curriculum for the sleigh is devised on the basis of knowledge of the sleigh's behaviour in an environment. In

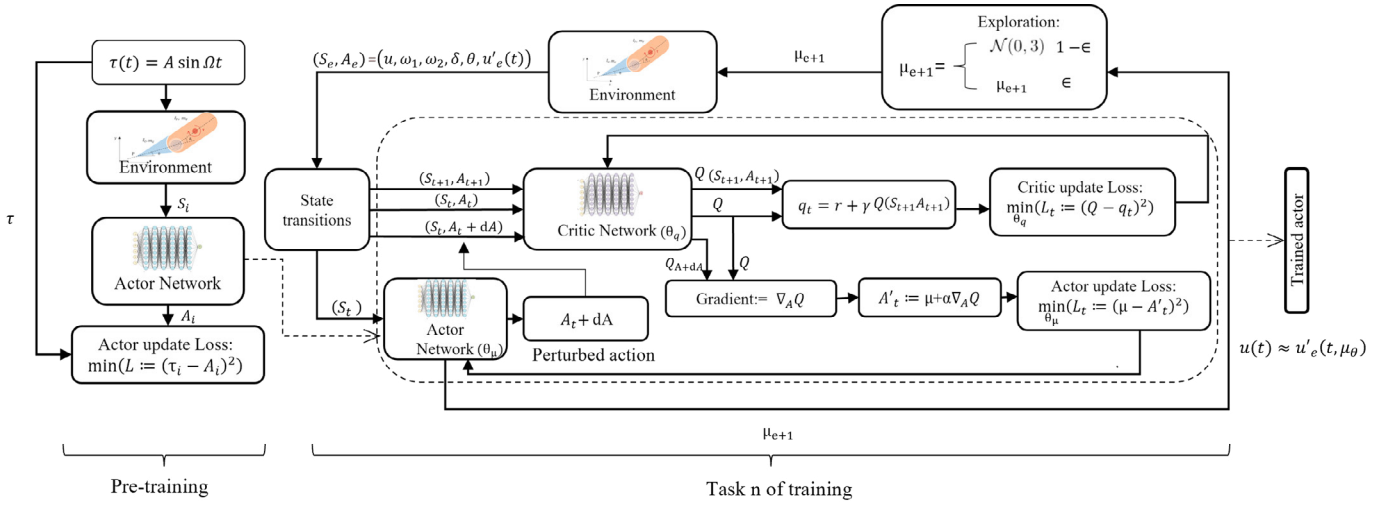


Fig. 3. Here we illustrate the training process. On the left, the DDPG actor $\mu(S)$ is trained to mimic a given $\tau(t)$. In the center, the actor is updated over multiple episodes e of DDPG, where the target velocity function $u'_e(t)$ is changed between episodes according to the curriculum, resulting in a fully-trained actor capable of simultaneous path and velocity tracking.

Rodwell and Tallapragada [2022] and in Fedonyuk et al. [2020] the two-link sleigh exhibits rich dynamics when subjected to periodic inputs. These observations lead to the curriculum for learning. Initially the agent is exposed to observation of the dynamics of the sleigh under a periodic input, in this case a sinusoidal forcing, in order to encode the concept of a periodic gait into the policy. The goal of pre-training is to generate a policy $\mu(S|\theta_\mu)$ that generates a similar periodic gait to that generated by the time-dependent forcing $\tau(t) = 0.25 \sin t$. This is achieved by collecting state and action vectors (S, A) of a trajectory using the latter time-dependent policy, and finding the weights θ_μ that minimize the vector expression $(A - \mu(S|\theta_\mu))^2$ using stochastic gradient descent over the trajectory. The results after pre-training are shown in Fig. 4, with the agent being able to track limit cycles in the reduced velocity space. The reward function used for training is defined as

$$r_t = -0.8 \mid s_u - u'(t) \mid - \mid max(\mid s_\theta - \theta_t \mid, 0.25) - 0.25 \mid .$$

4.2 Training

The training stage is where the agent is trained based on the curriculum shown in Fig. 5. The task of changing velocity is divided by using a method of task simplification where the task is divided into three training sub-tasks or skills, which is similar to skill-chaining (Konidaris and Barto [2009]) or task based sub-goals where at the end of each sub-goal the agent reaches an initialization sequentially approaching high rewards on the main task (Narvekar et al. [2020]). The hierarchy of the process shown in Fig. 3 is such that there are a finite number of episodes e in training and within each episode there are finite number of time steps $T = 100$. At the start of every episode, an environment simulation is performed, and the states and actions are collected into matrices S_e, A_e. The states at each of the time steps within these matrices are then referred to as (S_t, A_t). The update algorithm iterates over every time in the episode; the critic network accesses both observed states and actions while the actor network

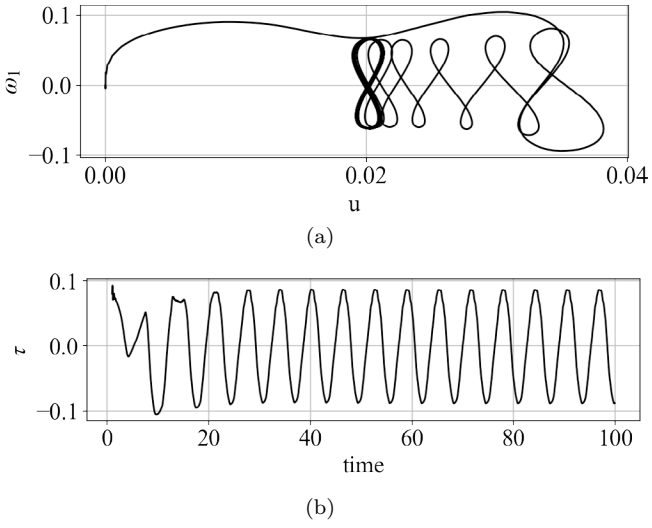


Fig. 4. The pre-trained torque $\tau(t) = \mu(S_t|\theta_\mu)$ (b) and resulting trajectory in (u, ω_1) space (a) for the pre-training step. The selected gait is intentionally a poor fit for the $u'(t) = 1$ of the next training step, but illustrates that teaching an arbitrary periodic gait is an effective starting point for training.

accesses the observed states. The critic network evaluates the current action as well as the action at time step $t + 1$ to implement ((7)). After the update is performed, a new episode begins and new trajectory is generated. An ϵ-greedy action selection method is used in order to maintain a balance between exploration and exploitation. A random action is chosen from a normal distribution $(1 - \epsilon)$ times during an episode as shown in Fig. 3 where ϵ is taken as 0.8. In this case, each task for training has the same type of $u'(t)$ function and is further divided into smaller stages on basis of difficulty of the target function $u'(t)$ or deviation from the mean target velocity u_0. The first task in Fig. 5(a) is to go from the pre-trained policy to tracking a noisy target velocity of $u'(t) \sim \mathcal{N}(1, \sigma)$. This

is designated as task 1 and is divided into smaller sub-tasks where each sub-task has a standard deviation larger than the previous stage. Over the entire duration of task 1, the standard deviation σ starts from 0.01 at first sub-task increasing to 0.1 at the final sub-task. Task 1 of training requires approximately 200 episodes. Task 1 starts at the point reached by the pre-trained actor. The final result at the end of task 1 training is shown in Fig. 5(a).

Task 2, shown in Fig. 5(b), is where sinusoidal functions with unit mean are used. The frequency and amplitude of the sinusoidal functions are varied at each sub-task. The frequency is first increased gradually from 0.01 to 0.1 and then the amplitude is steadily increased from 0.1 to 1. The objective of changing the amplitude is to achieve a higher velocity and for facilitating acceleration or deceleration of the sleigh. This is important for the path tracking case that is considered in the next section. Since the path tracking objective is to make the sleigh track a sharp corner with the least cornering radius it is necessary for the sleigh to slow down as it is reaching the corner and then change its velocity for the rest of the path. This is the reason for defining the final sub-task as $u'(t) = 1 + \sin(t)$.

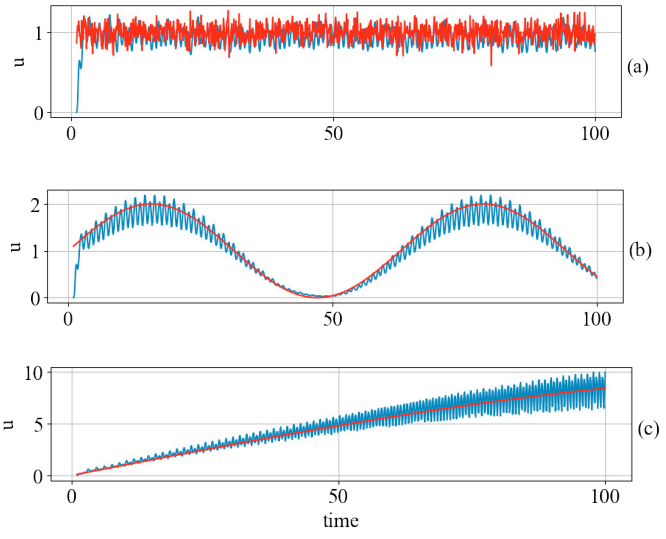
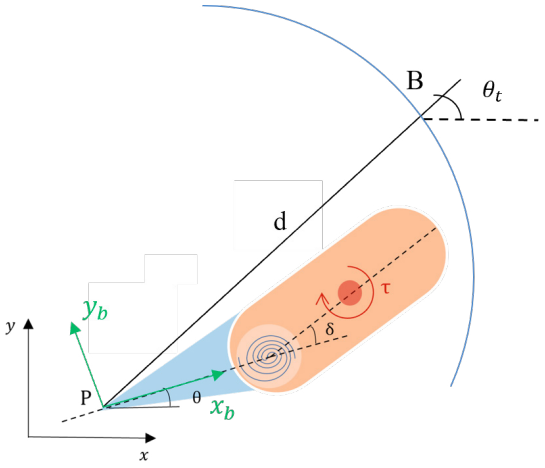


Fig. 5. Training curriculum for the policy. In (a) the target velocity for each time interval is sampled from normal distribution $u'(t) \sim \mathcal{N}(1, \sigma^2)$, where σ increases between iterations. In (b) the agent learns a policy to track a sinusoidal $u'(t)$ with increasing amplitude and frequency. In (c) the agent learns to track monotonically increasing velocities to a maximum of $u'(t) = 10$.

Task 3, shown in Fig. 5(c), trains the agent over a wider range of velocities. In this case the range considered is $0 \leq u'(t) \leq 10$. Because of the width of this velocity range compared to the range encountered in the previous task, we first consider a subtask with $0 \leq u'(t) \leq 5$, before advancing to the full range. This task is slow compared to the previous task, as each sub-task takes 100 episodes to match $u'(t)$. Here $u'(t)$ is a low frequency sine function as shown in Fig. 5(c). Tasks 1, 2, and 3 form the curriculum of the training stage and the sequence of the curriculum is a recursive manual process where the curriculum is modified by modifying sub-tasks according to the progress of the agent's learning.

5. PATH TRACKING RESULTS

The algorithm used for path-tracking is a pure pursuit algorithm. The angle θ_t between the body and spatial frames is the target angle being tracked at a fixed sight distance. The path tracking algorithm relies on two states of the sleigh: the current tail angle θ and the velocity u of the sleigh. In the pure pursuit path tracking, the agent receives feedback in the form of correction angle and a velocity target. The correction angle is defined as the difference between the tail angle and the target angle in the spatial frame. Fig. 6 illustrates the pure pursuit algorithm for this system. The circle of diameter d centered at the constraint point shows the visible area for pursuit. At point B the sight horizon intersects the forward path, and a pursuit vector $\bar{r}_{B/P}$ from P to B is calculated. Using the policy, the tail angle is regulated towards the angle of the pursuit vector.



(a)

Fig. 6. Schematic diagram of pure pursuit.

The use of the pure pursuit algorithm in combination with the previously trained RL agent is shown through a numerical simulation where the sleigh is required to track a path shown by the red lines in Fig. 7 (a) at a target velocity shown by the red graph in Fig. 7 (b) that varies with time. Tracking the piecewise constant target velocity is generates switches between different limit cycles in the reduced velocity space as shown in Fig. 7 (c).

6. CONCLUSION

The results in this paper demonstrate the control of an underactuated and nonholonomic two-link Chaplygin sleigh using Reinforcement Learning with a DDPG agent. A curriculum based on known physics of the system was found to result in rapid convergence to a gait that satisfies a target velocity and heading angle. In combination with a pure-pursuit algorithm, this policy was found to accurately track a prescribed path while simultaneously following a prescribed velocity. The curriculum learning approach can be used to teach an RL agent the qualitative state space features which can speed up its learning of the policy. Curriculum learning can also be used to train an agent, to not just perform tasks of growing complexity, but in

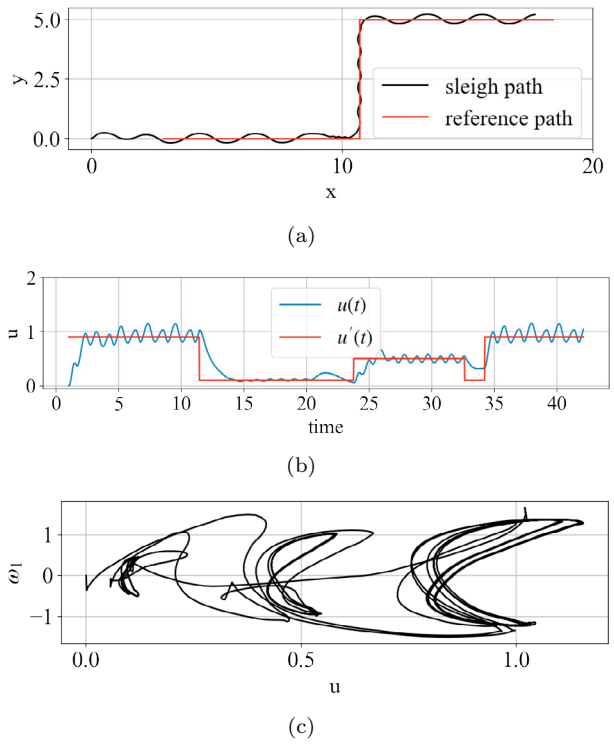


Fig. 7. (a) The trajectory of the constraint point (b) translational velocity of the sleigh as it tracks a prescribed path with prescribed target velocity $u'(t)$ and (c) the trajectory (u, ω_1) in the reduced velocity space.

fact to change the physics of the systems itself in growing levels of complexity. This is equivalent to learning a policy for a simpler physical system first and progressing to more complex physical systems.

REFERENCES

Belter, D. and Skrzypczyński, P. (2010). A biologically inspired approach to feasible gait learning for a hexapod robot.

Bloch, A.M. (2003). *Nonholonomic Mechanics and Control*. Springer Verlag.

Borisov, A.V. and Mamaev, I.S. (2003). On the history of the development of the nonholonomic mechanics. *Regular and Chaotic Dynamics*, 7(1), 43–47.

Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, 2562–2567. IEEE.

Fedonyuk, V. and Tallapragada, P. (2018). Sinusoidal control and limit cycle analysis of the dissipative chaplygin sleigh. *Nonlinear Dynamics*.

Fedonyuk, V., Rodwell, C., and Tallapragada, P. (2020). Nonholonomic systems with redundant degrees of freedom can exploit nonlinear frequency response to improve speed and efficiency of locomotion. In *Dynamic Systems and Control Conference*. ASME.

Guertin, P.A. (2009). The mammalian central pattern generator for locomotion. *Brain research reviews*, 62(1), 45–56.

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, 2619–2624. IEEE.

Konidaris, G. and Barto, A. (2009). Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22.

Laumond, J.P., Jacobs, P.E., Taix, M., and Murray, R.M. (1994). A motion planner for nonholonomic mobile robots. *IEEE Transactions on robotics and automation*, 10(5), 577–593.

Li, Z. and Canny, J. (1993). *Nonholonomic motion planning*. Springer Science & Business Media.

Liljebäck, P., Pettersen, K., Stavdahl, , and Gravdahl, J. (2012). A review on modelling, implementation, and control of snake robots. *Robotics and Autonomous Systems*, 60(1), 29–40.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

Murray, R.M. (1993). Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 28(5).

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*.

Rodriguez, D. and Behnke, S. (2021). Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 3033–3039. IEEE.

Rodwell, C. and Tallapragada, P. (2022). Induced and tunable multistability due to nonholonomic constraints. *Nonlinear Dynamics*, 1–12.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., Collobert, R., and Weston, J. (2014). Deterministic policy gradient algorithms. In *31st International Conference on Machine Learning*.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587), 484–489.

Tallapragada, P. (2015). A swimming robot with an internal rotor as a nonholonomic system. *Proceedings of the American Control Conference, 2015*.

Tallapragada, P. and Kelly, S.D. (2016). Integrability of velocity constraints modeling vortex shedding in ideal fluids. *Journal of Computational and Nonlinear Dynamics*, 12(2), 021008.

Wright, C., Johnson, A., Peck, A., McCord, Z., Naaktgeboren, A., Gianfortoni, P., Gonzalez-Rivero, M., Hatton, R., and Choset, H. (2007). Design of a modular snake robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2609–2614.