Dynamic modeling and AI-based control of a cable-driven parallel robot

Abir Bouaouda, Rémi Pannequin, François Charpillet, Dominique Martinez, Mohamed Boutayeb

Preprints of the 22nd IFAC World Congress
Yokohama, Japan, July 9-14, 2023

Dynamic modeling and AI-based control of a cable-driven parallel robot

Abir Bouaouda [*,**,****] **Rémi Pannequin** [**]
François Charpillet [****] **Dominique Martinez** [****,†]
Mohamed Boutayeb [***]

[*] *e-mail: abir.bouaouda@univ-lorraine.fr*
[**] *Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France*
[***] *Université de Lorraine, CNRS, CRAN, Inria, F-54000 Nancy, France*
[****] *Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*
[†] *Aix-Marseille Université, CNRS, ISM, 13009 Marseille, France*

Abstract: Controlling over-constrained cable-driven parallel robots (CDPRs) is a challenging task due to the complex dynamics of the system. Classical controllers require force distribution algorithms that involve an optimization problem, which is time consuming. In this paper, we propose an AI-based approach that learns a controller from simulated trajectories. A dynamic model of the CDPR is first validated experimentally on a real robot. Then, the controller is trained on the CDPR simulator with randomly generated trajectories using a deep deterministic policy gradient (DDPG). Finally, the trained controller is tested on different trajectories. Validation results show that the proposed approach is able to track unknown trajectories with a good accuracy.

Keywords: Reinforcement learning, deep learning, control, cable-driven parallel robot, deep deterministic policy gradient, trajectory tracking.

1. INTRODUCTION

Cable-Driven Parallel Robots (CDPRs) are among the most appealing type of parallel robots in the industry. Because of flexible cables and few moving parts, they provide a large workspace, payload, high speed, and high acceleration making them suitable for a wide range of applications.

In this paper, we consider a challenging application of CDPRs, which is to track fast moving targets. A CDPR that can track free-flying insects has been developed by Pannequin et al. (2020). The main difficulty in this application in previous work was to develop a controller capable of tracking the insect motion, which is highly unpredictable.

In this study, we intend to use the deep deterministic policy gradient (DDPG) algorithm to reduce tracking error and track the insect more efficiently. This algorithm was first introduced in the paper (Lillicrap et al., 2016), and described as an adaptation of the "Deep Q Network" (DQN) algorithm to continuous domains.

The advantage of using reinforcement learning is that it is a model-free approach, which means that it can be used to learn the optimal control policy directly from the interaction with the environment. It can thus reduce the effect of model uncertainty and can be used to learn the optimal policy for complex systems. Moreover, the control policy can be learned in an online fashion, which means that the policy can be updated as the environment changes over time, unlike the traditional control strategies like PID controller which requires an offline tuning.

The use of reinforcement learning for the control of CDPR has been investigated in many works, some of them are interested in hybrid methods that combine reinforcement learning with other control strategies. For example, in (Grimshaw and Oyekan, 2021), Q-learning is used to balance unstable loads on a CDPR, the deep neural network is providing the desired platform response to a nested PID controller. Another example of a hybrid approach is in applying Q-learning to optimize the tension of the cables (Xie et al., 2021).

Other works consider end-to-end reinforcement learning to solve the position control problem, e.g. in (Vu and Alsmadi, 2020; Ma et al., 2019; Sancak et al., 2022). Yet most of them focus on the trajectory tracking problem in specific conditions. In (Vu and Alsmadi, 2020), the trajectories are basically point-to-point trajectories for pick-and-place tasks. Reinforcement learning is used to track rotational trajectories in (Ma et al., 2019), and planar trajectories in (Sancak et al., 2022). In both cases, the performance of the control strategy is tested in low accelerations. Moreover, none of these previous works has been tested in a real environment.

In this study, we first establish a dynamic model for a two-DOF point mass CDPR, we validate the dynamic model with the real CDPR and use this model to create

the training environment for the reinforcement learning algorithm. The reason for using a simplified CDPR with only 4 cables is to be able to investigate the reward and the input trajectories influence on the training process in shorter training times. We then use the deep deterministic policy gradient algorithm to learn the optimal control policy. Finally, we test the performance of the control strategy in simulation. The proposed AI-based control strategy will be applied later to an existing robot with eight cables (Pannequin et al., 2020) in a real environment.

2. DYNAMIC MODELING OF TWO-DOFS POINT-MASS CDPR AND VALIDATION WITH REAL DATA

The aim of our work is to control cable-driven parallel robots (CDPRs) with a reinforcement learning based approach to track trajectories that are unknown in advance. In this section, we examine the dynamic model of the robot.

2.1 System description

The robot is composed of an end-effector and four motors, each one driving a cable that is attached to the end-effector. The motors are placed at the corners (A_k) of a vertical rectangle in the $x-z$ plane. We suppose that the end-effector is a point mass p of mass m.

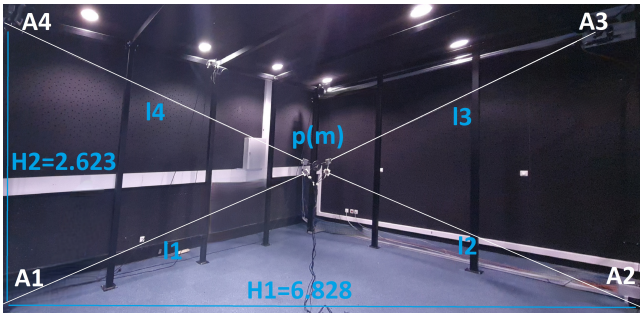

Fig. 1. Two-DOFs point-mass CDPR with four cables, l_k is the length of the cable k, A_k is the position of the motor k, p is the position of the end-effector.

2.2 Dynamic model

Dynamic modeling consists in finding the equations between the set of variables that describe the system in our case the position and the speed of the end-effector and the set of variables that describe the inputs of the system, in our case the speed applied to each motor.

According to the Newton-Euler equations, the dynamic model of the robot is given by:

$$m\boldsymbol{a} = T_1.\boldsymbol{n}_1 + T_2.\boldsymbol{n}_2 + T_3.\boldsymbol{n}_3 + T_4.\boldsymbol{n}_4 + \boldsymbol{P} \tag{1}$$

$\boldsymbol{a}$ represents the acceleration vector of the end-effector, $\boldsymbol{n}_i$ represents the unit vector of the cable i, T_i represents the tension of the cable i and $\boldsymbol{P}$ represents the Earth's gravitational force applied to the end-effector. This equation can be rewritten as:

$$m\ddot{x} = T_1.n_{1x} + T_2.n_{2x} + T_3.n_{3x} + T_4.n_{4x} \tag{2}$$
$$m\ddot{z} = T_1.n_{1z} + T_2.n_{2z} + T_3.n_{3z} + T_4.n_{4z} - mg \tag{3}$$

n_{kx} and n_{kz} are the components of the unit vector $\boldsymbol{n}_k$ in the x and z axis, respectively. The Jacobian matrix of the robot is given by (Pott, 2018):

$$\boldsymbol{J} = \begin{bmatrix} n_{1x} & n_{2x} & n_{3x} & n_{4x} \\ n_{1z} & n_{2z} & n_{3z} & n_{4z} \end{bmatrix} \tag{4}$$

The dynamic model of the robot can be rewritten as:

$$m.\ddot{\boldsymbol{X}} = \boldsymbol{J}(X).\boldsymbol{T} + \begin{bmatrix} 0 \\ -mg \end{bmatrix} \tag{5}$$

where $\boldsymbol{X} = \begin{bmatrix} x \\ z \end{bmatrix}$ *and* $\boldsymbol{T} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}$.

The relation between the speed of the cables and the speed of the end-effector is given by:

$$\dot{\boldsymbol{l}} = \boldsymbol{J}^{\intercal}(\boldsymbol{X}).\dot{\boldsymbol{X}} \tag{6}$$

where $\boldsymbol{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix}$.

For the sake of simplicity, we assume that the cables are inextensible and that the torque produced by the motors is proportional to the tension of the cable as follows:

$$\tau_i = rT_i \tag{7}$$

where r is the drum radius.

As we use DC motors, the torque produced by the motors is proportional to the current of the motor.

$$\tau_i = K_c I_i \tag{8}$$

Where I_i is the current of the motor i and K_c is the torque constant. Thus, the expression of the cable tension for motor i is given by:

$$T_i = \frac{I_i}{rK_c} \tag{9}$$

Moreover, we consider that the proportional control loop of the current of the motors is part of the dynamic model of the system. So the relation between the motor current, the desired speed, and measured speed of the motor is given by:

$$I_i = k_{motor}(u_i - v_{mi}) \tag{10}$$

where u_i is the desired speed of the motor i, v_{mi} is the measured speed of the motor i, and k_{motor} is the motor gain.

We further assume that the cables are under sufficient tension so that the speed of the cable is the same as the speed of the motor:

$$v_{mi} = l_i \ . \tag{11}$$

2.3 Training environment

We created a training environment for the robot using the previous model. In the perspective of the Agent, the

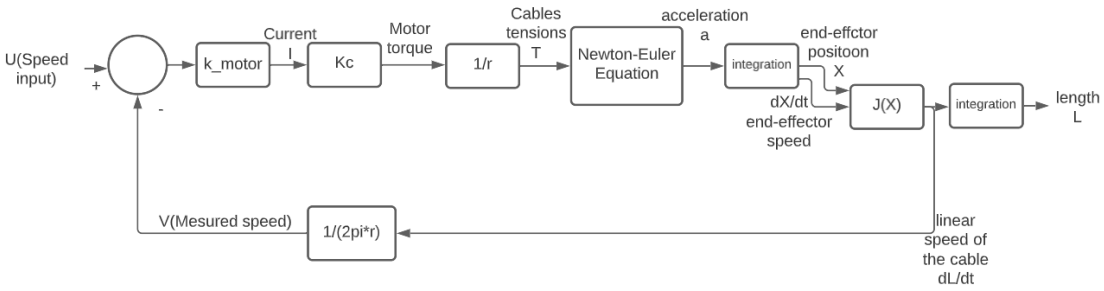

Fig. 2. Dynamic model of CDPR setting down the relation between the inputs (desired motors speed and the outputs (measured motors speed, end-effector position and speed).

environment is a black box that takes as input the action of the agent and returns the reward and the next state.

We define the state of the environment as the last position of the end-effector, the current position of the end-effector, the tracking error and the electric current of the motors. At the time t, the state s_t and action a_t write as follows :

$$s_t = (X_t, X_{t-1}, e_t, I_t) \tag{12}$$

$$a_t = u = (u_1, u_2, u_3, u_4) \tag{13}$$

The use of the position in the state space rather than cable lengths is justified by the fact that there is a combination of lengths that does not describe any position, as well as to reduce the dimension of the state space.

We will use this environment in the next section to train the agent.

3. CONTROL LAW

Our goal is to use the deep deterministic policy gradient (DDPG) to control the robot in order to track an arbitrary trajectory.

3.1 DDPG algorithm

The DDPG algorithm is composed of two networks: the actor and the critic. The actor is a neural network that maps the state of the system to the action of the system and the critic is a neural network that maps the action and the state to the Q function, an estimation of the future rewards. See Fig 3.

Each network is associated with another network with the same architecture (*target network*) that is updated more slowly, which makes the training more stable. The parameters of the Actor and the Critic network are optimized so that the agent recommends actions that maximize the rewards.

3.2 Action and state space exploration

One of the main assumptions of the reinforcement learning algorithms to converge is that all the states and actions are visited during the training. To ensure that, we use a Gaussian noise to explore the action space, and we use a random trajectory to explore the state space.

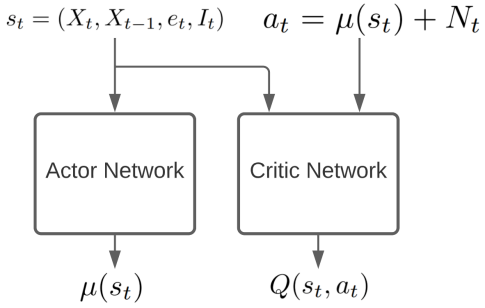

Fig. 3. Actor-Critic Networks. The input of the actor is the state of the system s_t and the output is the action $\mu(s_t)$. The input of the critic is the state s_t and the action with noise $a_t = \mu(s_t) + N_t$ of the system, and the output is the $Q(s_t, a_t)$, the value of the Q function for the action a_t at state s_t.

Unlike the original paper (Lillicrap et al., 2016), we found that the ornstein-uhlenbeck noise doesn't have real advantage, and other papers (Barth-Maron et al., 2018; Fujimoto et al., 2018) simply use Gaussian noise, so, we decided to use it as well.

For the state space exploration, we generate a random initial position for each episode and random trajectory which starts from this initial position. The components of the target position are generated using the following equation:

$$\begin{aligned} v_{t+1} &= v_t + a_t * dt \\ x_{t+1} &= x_t + v_t * dt \end{aligned} \tag{14}$$

where a_t is the acceleration generated randomly between minimum and maximum values, $v_0 = 0$ and x_0 is the initial position.

3.3 Current limits

In order to keep the cables tensed and to avoid reaching the maximum current of the motors, we use a current saturation function. This function is applied on the action of the agent before sending it to the robot. Unlike the old control method (Pannequin et al., 2020), using a saturation function doesn't affect the control law as the agent is trained with this condition, so it can learn to control the robot with this constraint.

$$i_{sat} = \begin{cases} i_{max} & \text{if } i > i_{max} \\ i & \text{if } i_{min} < i < i_{max} \\ i_{min} & \text{if } i < i_{min} \end{cases} \quad (15)$$

And as the current is related to the speed of the motor, we use equation 10 to compute the maximum speed of the motor:

$$i_{sat} = k_{motor}(u_{sat} - v_m) \quad (16)$$

So the action of the agent is deduced using this function:

$$u_{sat} = \frac{i_{sat}}{k_{motor}} + v_m \quad (17)$$

3.4 Reward function

The reward function is the most important part of the reinforcement learning algorithm. It is used to evaluate the performance of the agent. We use the following reward function modified from (Sancak et al., 2022):

$$r_t = -(\|e_t\| + (\|e_t\| - \|e_{t-1}\|) + 0.05\sqrt{\|i_t\|}) \quad (18)$$

where e_t is the tracking error and i_t is the current of the motors. The reward function is composed of three parts:

- The first part is the tracking error. The goal of the agent is to minimize this error to get closer to the target trajectory.
- The second part is the difference between the tracking error in current and previous steps. Thus, it penalizes the agent when the tracking error increases.
- The third part is the current of the motors, the goal of the agent is to minimize this current to minimize the energy consumption, and to avoid reaching the maximum current of the motors.

3.5 DDPG algorithm for Cable Robot control:Algorithm 1

Algorithm 1: DDPG algorithm for CDPR control.

Initialize the parameters of the critic network $Q(s, a)$ and copy them to the critic target network $Q'(s, a)$;
Initialize the parameters of the actor network $\mu(s)$ and copy them to the actor target network $\mu'(s)$;
for *episode* $\leftarrow 1$ **to** *Total number of episodes* **do**
  Initialize s ;
  Generate a random trajectory;
  for *step* $\leftarrow 1$ **to** *Maximum number of steps* **do**
    Generate a random noise N_t for action exploration ;
    Select an action $a_t = \mu(s_t) + N_t$;
    Compute u_{sat} using the equations (16, 17) to ensure the current limits;
    Use u_{sat} to deduce the saturated action $u_{sat}(t)$;
    Apply $u_{sat}(t)$ to the robot and get the next state $s_{t+1} = (X_{t+1}, X_t, e_{t+1})$ and the reward r_t ;
    Store the transition (s_t, a_t, r_t, s_{t+1}) in the replay buffer;
    Sample a random minibatch of transitions (s_i, a_i, r_i, s_{i+1}) from the replay buffer. ;
    Update the Q network by minimizing the training loss ;
    Update the μ network policy using the sampled policy gradient ;
    Update the parameters of the target networks (slowly) by using an update rate τ;
    if *the new position can't be reached* **then**
      End this episode;
    end
  end
end

4. RESULTS

4.1 Validation of the model in simulation with real data

We simulated the developed model in Fig. 2 of the robot using Simulink. To validate the model, we used real data that we collected during a trajectory tracking using classical control law with real CDPR (Pannequin et al., 2020).

We used the same motors speed generated during the real trajectory tracking experiment to compare the real and simulated end-effector position for the same inputs. The results are shown in Fig. 4. The simulation and the real data are very close. The maximum error is 0.04 m in both x and z axis. We also measured the cable tensions experimentally using Phidget 22 force sensors during trajectory tracking and found that the current is proportional to the cable tension (Pearson correlation $R^2 = 0.98$, Fig. 5). Thus, the measured motor currents i_t can be used as estimators of the cable tensions in the reward function.

4.2 Hyperparameters and training

We use the hyperparameters in Table 1 for the DDPG algorithm, the parameters in Table 2 for the trajectory generation and the parameters in Table 3 for the current limits.

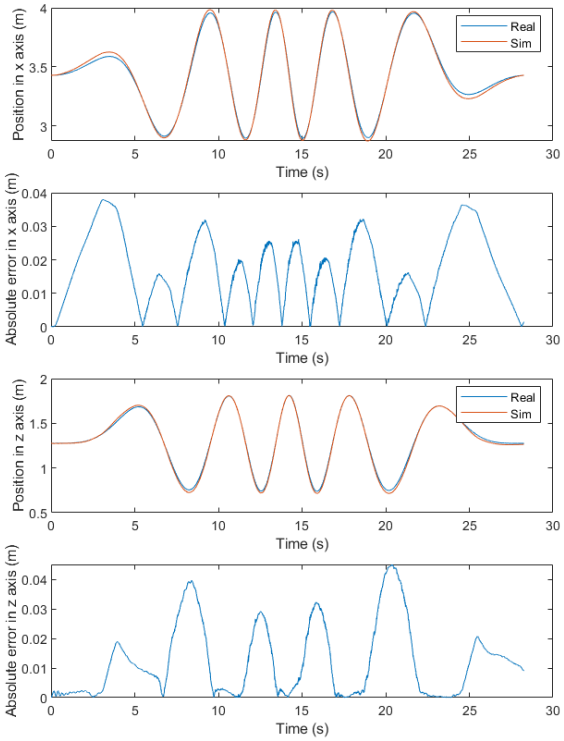

Fig. 4. Comparison between the real and simulated end-effector trajectories using the same inputs.

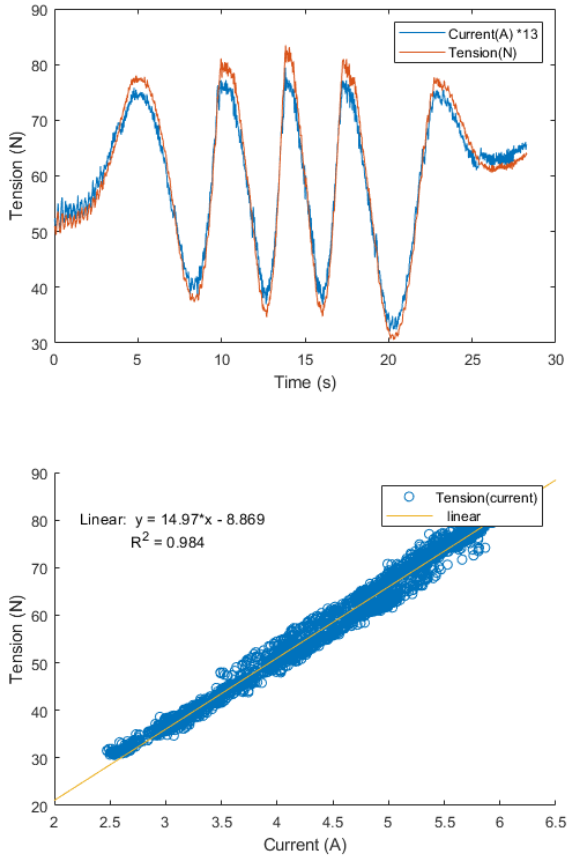

Fig. 5. Correlation between the cable tension and the motor electrical current. Experimental data recorded for one motor.

Table 1. Hyperparameters

Hyperparameter	Value
Number of episodes	10000
Time step	0.01 s
Maximum number of steps	1000
Critic learning rate	0.0002
Actor learning rate	0.0001
Update rate for target networks	0.0005
Discount factor for future rewards	0.99
Buffer size	50000
Mini-batch size	128

Table 2. Acceleration limits

Parameter	Value
Maximum acceleration in x axis	1 m/s^2
Maximum acceleration in z axis	0.5 m/s^2
Maximum velocity in x axis	4 m/s
Maximum velocity in z axis	2 m/s

4.3 Training results

In Fig 6 we can see the increasing of the reward during the training of the agent until it converges to a stable value.

The total training time is 1 day for 10000 episodes using an i7-10850H CPU (6 cores) with 16 GB RAM.

Table 3. Current limits

Parameter	Value
Maximum current	6 A
Minimum current	0.3 A
Maximum input speed	300 rpm
Minimum input speed	-300 rpm

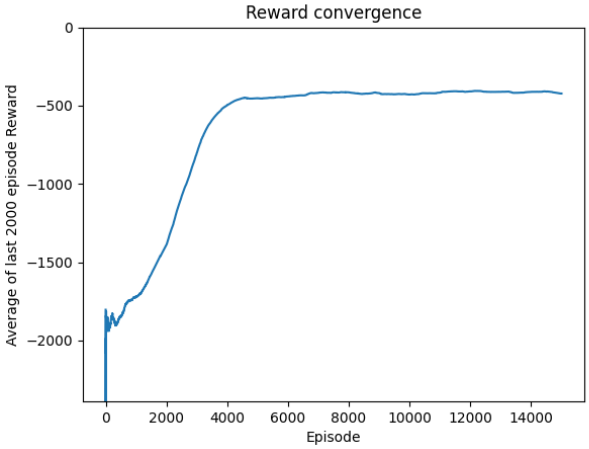

Fig. 6. Average reward during the training, it increases during the training and reaches a stable value after 5000 episodes.

4.4 Testing

In Fig 7 we test the agent with a trajectory that the robot has never seen before. The robot is able to track the trajectory with a maximum tracking error of 0.03 m in both x and z axis. In Fig 8 we can see the trajectory projection in the x-z plane. In Fig 9 we can see that the average current is less than 3 A during the trajectory tracking while the maximum allowed current is 6 A and the average current for the motors in the bottom is so close to the minimum current which is 0.3 A.

5. CONCLUSION

In this study, we developed a test bench for the control of a CDPR using a reinforcement learning algorithm. From the proposed model, we generated an environment to test different training algorithms. Then we used this environment to develop an approach to control a CDPR using a deep deterministic policy gradient algorithm. The results show that the proposed approach is able to track a trajectory with a maximum error of 3 cm in both x and z axes while optimizing current consumption during the trajectory. This study was conducted with some limitation in motor speed so as to reduce the training time. As future works, we intend to investigate the effect of the motor input speed on the training process, to use the trained agent to control the real robot, and to extend the proposed approach to cable robots with more cables and more degrees of freedom.

ACKNOWLEDGEMENTS

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest

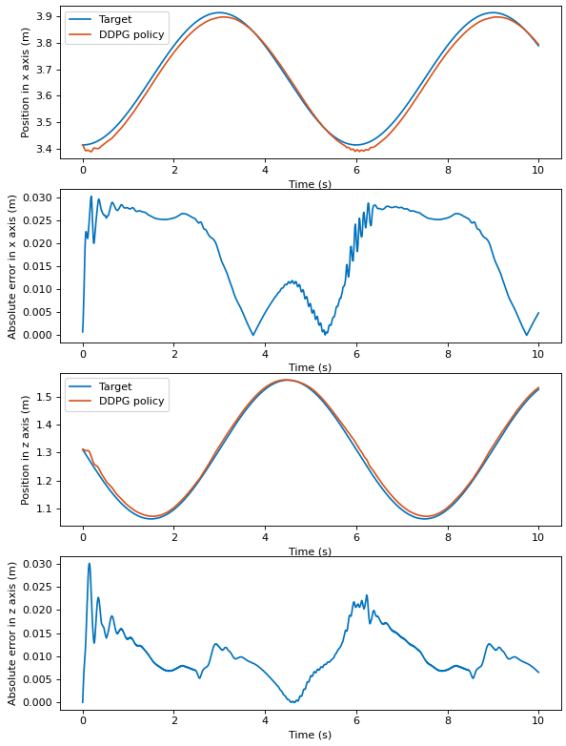

Fig. 7. x and z positions during trajectory tracking using the trained agent in simulation vs target trajectory.

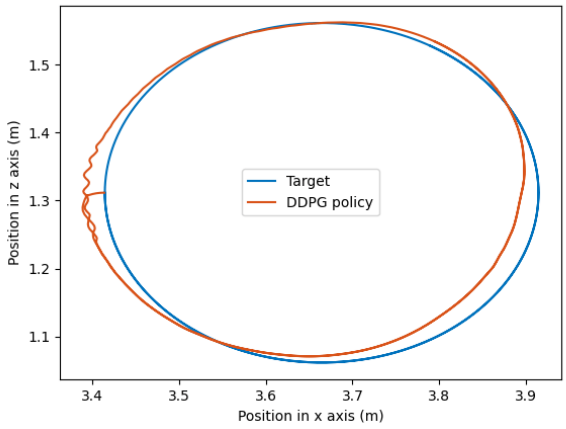

Fig. 8. Trajectory projection in the x-z plane.

group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr/).

REFERENCES

Barth-Maron, G., Hoffman, M.W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. (2018). Distributional policy gradients. In *International Conference on Learning Representations*.

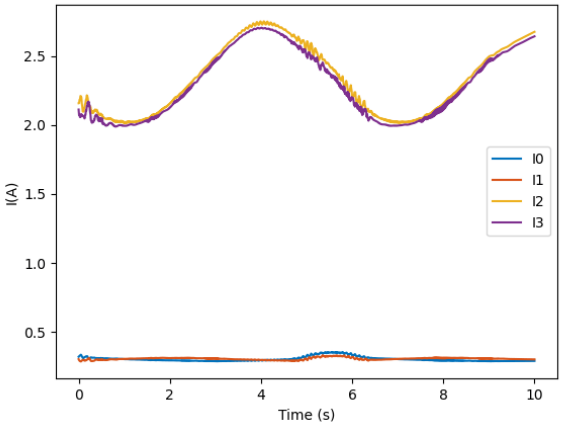

Fig. 9. Average current over 1 second during the trajectory tracking in Fig 7.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477.

Grimshaw, A. and Oyekan, J. (2021). Applying Deep Reinforcement Learning to Cable Driven Parallel Robots for Balancing Unstable Loads: A Ball Case Study. *Frontiers in Robotics and AI*, 7, 611203.

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 Y. Bengio and Y. LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Ma, T., Xiong, H., Zhang, L., and Diao, X. (2019). Control of a Cable-Driven Parallel Robot via Deep Reinforcement Learning. In *2019 IEEE International Conference on Advanced Robotics and its Social Impacts (ARSO)*, 275–280. IEEE, Beijing, China.

Pannequin, R., Jouaiti, M., Boutayeb, M., Lucas, P., and Martinez, D. (2020). Automatic tracking of free-flying insects using a cable-driven robot. *Science Robotics*, 5(43), eabb2890.

Pott, A. (2018). *Cable-Driven Parallel Robots: Theory and Application*, volume 120. Springer International Publishing.

Sancak, C., Yamac, F., and Itik, M. (2022). Position control of a planar cable-driven parallel robot using reinforcement learning. *Robotica*, 1–18. Publisher: Cambridge University Press.

Vu, D.S. and Alsmadi, A. (2020). Trajectory Planning of a CableBased Parallel Robot using Reinforcement Learning and Soft Actor-Critic. *Wseas transactions on applied and theoretical mechanics*, 15, 165–172.

Xie, C., Zhou, J., Song, R., and Xu, T. (2021). Deep Reinforcement Learning Based Cable Tension Distribution Optimization for Cable-driven Rehabilitation Robot. In *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, 318–322. IEEE, Chongqing, China.