From a trapezoidal acceleration profile to a learnt time optimal control policy for robot braking

Arthur Esquerre-Pourtère, Nicolas Torres Alberto, Vincent Padois

From a trapezoidal acceleration profile to a learnt time optimal control policy for robot braking

Arthur Esquerre-Pourtère[1], Nicolas Torres Alberto[2] and Vincent Padois[3]

***Abstract*— The work presented in this paper tackles the question of braking efficiently with robots. This is an important feature for collaborative robots. These systems need to ensure safety of their surrounding operators without compromising performance.**
The proposed approach leverages existing motion planning methods based on trapezoidal acceleration profiles which provide a sub-optimal control policy for braking, assuming minorant constant joint jerk and acceleration capacities. Based on this initial solution to braking, a control policy is learnt by reinforcement using the Proximal Policy Optimization deep learning algorithm. This learnt policy assumes constant torque and torque derivative actuation capabilities and predicts an optimal variation of the sub-optimal control policy which better exploits the actuation capabilities of the robot.
The principles of this learning approach are first presented and then evaluated in a simulation with an academic example of a Panda robot restricted to 2 degrees of freedom. The obtained results are promising as they lead to a reduction of the stopping time and energy required to brake.

I. Introduction

In the context of collaborative robotics, humans are meant to work near robots and to share their workspace. While this novel paradigm opens many new possibilities [1], it also makes safety an even more central issue [2].
An important aspect of safety is the ability to brake efficiently and stop the robot before an impact occurs. One way of braking very quickly is to use an emergency stop system which mechanically blocks the robot's joints [3] and electrically stops the actuators. However this presents two problems: it requires that the robot is restarted manually afterwards and it can be mechanically detrimental to the robot's longevity. Furthermore, this strategy does not formally guarantee collision avoidance (e.g. with a human). Instead, a general controller implicitly leading to a controlled stop "when necessary" yields better efficiency (no human intervention required to restart the robot) and can more directly embed safety guarantees as constraints expressed at the control level [4].
Nevertheless, this type of approach makes a strong assumption on the motion capabilities, for example expressed in terms of joint space acceleration and jerk, of robots: they are supposed to be constant. While this largely simplifies the problem of computing an optimal stopping trajectory, it also leads to over and/or under estimations of the stopping time. The latter can be detrimental to safety in case of impact [5]. The former raises the question of efficiency: the robot actuation is restricted to a subset of its capabilities and potentially over-sized (and thus more dangerous) so that this subset fits the requirements of the applicative context. Nevertheless, properly accounting for the true capabilities of robots is not specific to the braking problem and some recent work have shown that computing the real capabilities of robots is achievable in real-time [6]. Yet, the obtained bounds are state-dependant and computing an optimal control policy from these non-linear, state-dependent bounds remains a complex optimal control problem which is generally dealt with along a given path [7] (this does not fit the case of an efficient stopping motion). Recent works are attempting to tackle this general problem with Differential Dynamic Programming [8] but accounting for bounds in these approaches is not trivial and the general problem remains open.

While model-based approaches lead to complex high dimensional optimization problems, machine learning, and especially deep learning and reinforcement learning, allows to tackle high dimensional problems and can be applied to the context of robotic control, for instance for the control of robotic manipulation [9]. In this context, the work described in this paper explores the ability to learn a control policy optimally using the dynamic capabilities of a robot to generate efficient stopping motions. A robot can be considered stopped when its velocity and acceleration are null. Reaching such a state can be done in different ways and efficiency can be quantified in terms of braking time, braking distance or braking energy. In this work, time is considered as the central optimization criteria and a discrete control policy is learnt using a neural network trained in simulation with the Proximal Policy Optimization (PPO) deep reinforcement learning algorithm [10]. In order to simplify the learning process, an initial, model-based, control policy is computed assuming constant maximal acceleration and jerk while the proposed learning algorithm only assumes constant maximum actuation torque and torque derivative. While still slightly restrictive, this is, given the velocity/torque characteristics of DC actuators equipping most robots, a reasonable assumption which yields interesting performance gains illustrated with simulation results on a 2 degrees of freedom (DOF) academic example robot.

The paper is organized as follows. Section II rapidly introduces essential elements to describe the braking problem. Section III provides an overview of the whole learning system. Each component of this system is then described in the Section IV. The results of the different methods applied to a simplified Franka Emika Panda robot are then displayed and analyzed in Section V. Finally, the results of this work are summarized and some insight on potential future work is provided in Section VI.

[1]Arthur Esquerre-Pourtère was an intern at Inria in the Auctus research team from March to September 2021. arthur.esquerre@gmail.com

[2]Nicolas Torres Alberto is a PhD student at Auctus, Inria Bordeaux, Talence, France and Stellantis, Centre Technique Vélizy, France. nicolas.torres@stellantis.fr

[3]Vincent Padois is a senior research scientist at Auctus, Inria Bordeaux, Talence, France. vincent.padois@inria.fr

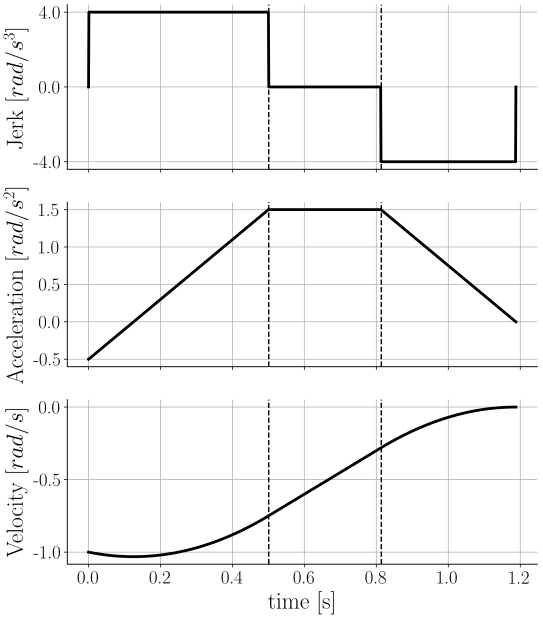

Fig. 1: Trapezoidal acceleration profile (TAP).

II. Initial approach to braking

A. Stopping condition

Let us consider fixed-base serial robotic manipulators with n degrees of freedom (DOF). Their joint space configuration, velocity, acceleration and jerk are respectively denoted $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\dddot{\mathbf{q}}$.

In this work, a robot is considered to be stopped when the absolute value of both the velocity and acceleration of each DOF is lower than a small value ϵ:

$$\forall i \in \{1...n\},\ |\dot{\mathbf{q}}_i| < \epsilon \wedge |\ddot{\mathbf{q}}_i| < \epsilon \tag{1}$$

ϵ can be physically related to the dry friction acting at each axis.

B. TAP: Trapezoidal acceleration profile

A classical approach to trajectory generation is given by "Trapezoidal velocity profiles" [7]. Such profiles allow to reach a desired position starting from any other position, given some constant acceleration and velocity limits as well as initial and final conditions on the velocity (often considered zero).

In this work, the braking problem is slightly different: starting from any arbitrary velocity, we want to reach a velocity and an acceleration equal to zero while considering some actuation limits expressed at the acceleration and jerk levels. In a first approximation, assuming these limits to be constant, the trapezoidal approach can be expressed at the acceleration level as illustrated in Figure 1. Such a "Trapezoidal acceleration profile" (TAP) can be used to reach the stopping condition from an initial velocity and acceleration given constant bounds on jerk and acceleration. In the specific case depicted in Figure 1, a non zero initial acceleration may lead to a temporary increase in velocity before velocity can actually start decreasing.

C. Limits of the simple approach

While jerk can be considered as a control input for this system, actuation limits are more accurately described at the joint

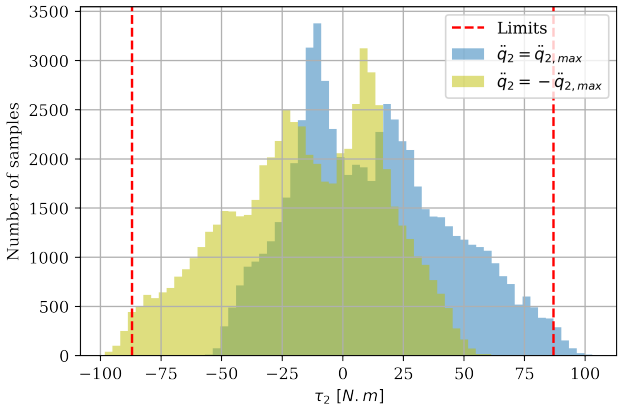

Fig. 2: Distribution of τ_2 when considering the maximum and minimum accelerations in 1000 partially described state, which results in 128000 samples.

torque $\boldsymbol{\tau}$, joint torque derivative $\dot{\boldsymbol{\tau}}$ and joint velocity levels:

$$\begin{array}{ccccc} \boldsymbol{\tau}_{min} & \leq & \boldsymbol{\tau} & \leq & \boldsymbol{\tau}_{max} \\ \dot{\boldsymbol{\tau}}_{min} & \leq & \dot{\boldsymbol{\tau}} & \leq & \dot{\boldsymbol{\tau}}_{max} \\ \dot{\mathbf{q}}_{min} & \leq & \dot{\mathbf{q}} & \leq & \dot{\mathbf{q}}_{max} \end{array} \tag{2}$$

Reaching the stopping condition described by equ. (1) while optimally using the actuation capabilities of the robot described by equ. (2) requires to relate torque and acceleration through the system's equation of motion:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{g}(\mathbf{q}) - \mathbf{B}\dot{\mathbf{q}}) \tag{3}$$

where $\mathbf{M}$ is the inertia matrix, $\mathbf{C}$ centrifugal and Coriolis induced torques, $\mathbf{g}$ is the vector of gravity torques and $\mathbf{B}$ is the diagonal viscous friction matrix. From equ. 3, it is clear that the acceleration and thus jerk capabilities of the robot are not constant over the whole state space. Considering constant bounds over the whole state space can lead to an under-utilization of the robot's real actuation capabilities (or, worse, an over-estimation of these capabilities).

In order to illustrate the under-utilization of the robot's real capabilities when considering constant acceleration bounds, an initial numerical experiment is proposed, using a simulated 7 DOF Franka Emika Panda robot [11] and the limits given by the manufacturer.

In a given state $\{\mathbf{q},\ \dot{\mathbf{q}}\}$, the maximum joint space acceleration $\ddot{q}_{i,max}$ or minimum acceleration $-\ddot{q}_{i,max}$ provided by the Franka Emika documentation is considered for each DOF. A total of $2^7 = 128$ different combinations of maximum and minimum accelerations are tested and for each combination the equivalent joint torque $\boldsymbol{\tau}$ is computed using equ. (3).

The experiment is repeated with 1000 sampled states, generated using existing trajectories as described in section IV-B. Figure 2 displays the distribution of the computed torque on the second DOF τ_2, considering all combinations and all sample states ($1000 \times 128 = 128000$ samples in total).

In most cases, maximum joint level acceleration can be achieved without reaching the actuation torque bounds. For instance, half of the samples allow to reach maximum acceleration with a torque $\tau_2 <= 9.8\,N.m$ while $\tau_{2,max} = 87\,N.m$. While this graph does not illustrate the coupling effects of the joint dynamics, it is sufficient to illustrate the fact that much higher torques could be

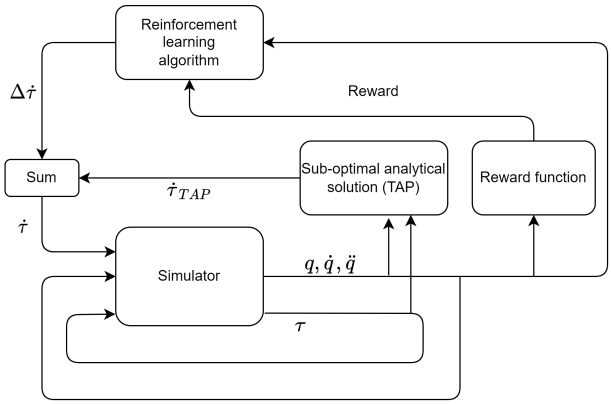

Fig. 3: Learning system architecture.

applied in most states, likely resulting in higher accelerations. Similar results can be observed for all other DOFs, although the 2nd DOF is the only one for which an overestimation of the real capacities have been observed in some states. Indeed, in some rare states, considering the $\ddot{q}_{2,max}$ or $-\ddot{q}_{2,max}$ implies a joint torque τ_2 which exceeds the limits $-\tau_{2,max}$ or $\tau_{2,max}$. This is very likely related to the fact that joint 2, given its specific location and orientation in the kinematic chain is the one which implies the largest dynamic coupling and gravity effects.

Overall, these results illustrate the difficulty to estimate at best constant inner box approximations of polytopes [6] and, as a consequence, the poor quality of the constants limits on the actuation capacities given by the manufacturer. Even though the TAP allows to brake while using an acceleration $\ddot{\mathbf{q}}_{max}$ and a jerk $\dddot{\mathbf{q}}_{max}$, those values do not match the robot's real capacities. This means that the maximum torque $\boldsymbol{\tau}_{max}$ and derivative of the torque $\dot{\boldsymbol{\tau}}_{max}$ are often not reached, or are sometimes exceeded. This emphasizes the fact that a method able to compute an optimal stopping trajectory that makes a better use of the actuation capacities of the robot is needed. Yet, the non-linear state dependence of the equation of motion (3) renders this problem complex.

III. Learning system architecture in a nutshell

Considering the difficulties in computing an optimal analytical solution, the method presented in this paper uses trial and error in order to learn a policy that makes better use of the robot's real capabilities. Yet, instead of learning the policy from scratch, the proposed learning system architecture takes advantage of the existence of a sub-optimal solution to the problem: the TAP. Figure 3 illustrates the working principles of such an hybrid learning architecture.

So as to limit the learning time and for safety reasons, the policy is learned in simulation. The role of the simulator is to integrate the dynamics of the system given torque derivative inputs as well as to generate initial conditions used during the training phase of the learnt controller.

At each time step during learning, the simulator computes the state of the robot. Torque derivative being considered the control input, the considered state is described by $\{\mathbf{q},\ \dot{\mathbf{q}},\ \ddot{\mathbf{q}}\}$. Given this state, the sub-optimal analytical solution to braking, $\dot{\boldsymbol{\tau}}_{TAP}$, is computed. The state is also sent into a Reinforcement learning algorithm whose role is to output a value $\Delta\dot{\boldsymbol{\tau}}$. In this work, the control policy is learnt using a neural network trained in simulation with the Proximal Policy Optimization (PPO) deep reinforcement learning algorithm [10].

The control input $\dot{\boldsymbol{\tau}}$ provided to the simulator is then computed as follows:

$$\dot{\boldsymbol{\tau}} = \dot{\boldsymbol{\tau}}_{TAP} + \Delta\dot{\boldsymbol{\tau}} \tag{4}$$

The proposed reinforcement learning approach is thus guided by a suboptimal, simple to compute, analytical solution. It explores the solution space around an initial solution space in order to find an improved policy. Such an approach is expected to leverage the existing knowledge to reduce the number of learning samples required to learn the braking control policy [12].

IV. Proposed approach

A. Simulation

Given a control input $\dot{\boldsymbol{\tau}}_{k+1}$ for the next control instant, the simulator first saturates this input in order to comply with the maximum torque and torque derivative constraints described by equ. (2) with:

$$\dot{\boldsymbol{\tau}}_{k+1,min} = \max(\dot{\boldsymbol{\tau}}_{min}, (\boldsymbol{\tau}_{min} - \boldsymbol{\tau}_k)/\Delta t) \tag{5}$$

$$\dot{\boldsymbol{\tau}}_{k+1,max} = \min(\dot{\boldsymbol{\tau}}_{max}, (\boldsymbol{\tau}_{max} - \boldsymbol{\tau}_k)/\Delta t) \tag{6}$$

with $\boldsymbol{\tau}_k$ the joint torque at the current control instant and Δt the considered control period. In this work, Δt is taken as 1 ms which corresponds to the recommended sample time when performing torque control with the Panda robot.

Given this control input, the updated torque value is computed as:

$$\boldsymbol{\tau}_{k+1} = \boldsymbol{\tau}_k + \dot{\boldsymbol{\tau}}_{k+1}\Delta t \tag{7}$$

From this torque, the joint acceleration $\ddot{\boldsymbol{q}}_{k+1}$ is computed using the equation of motion (3) evaluated in $\{\boldsymbol{q}_k,\ \dot{\boldsymbol{q}}_k\}$.

In order to reduce the truncature error due to numerical integration, the corresponding joint jerk is estimated as:

$$\dddot{\mathbf{q}}_{k+1} = \frac{\ddot{\mathbf{q}}_{k+1} - \ddot{\mathbf{q}}_k}{\Delta t} \tag{8}$$

and numerical integration is then performed to compute the new joint position and velocity:

$$\dot{\mathbf{q}}_{k+1} = \dot{\mathbf{q}}_k + \ddot{\mathbf{q}}_k\Delta t + \frac{\dddot{\mathbf{q}}_{k+1}\Delta t^2}{2} \tag{9}$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k\Delta t + \frac{\ddot{\mathbf{q}}_k\Delta t^2}{2} + \frac{\dddot{\mathbf{q}}_{k+1}\Delta t^3}{6} \tag{10}$$

Given, the updated state $\{\boldsymbol{q}_{k+1},\ \dot{\boldsymbol{q}}_{k+1},\ \ddot{\boldsymbol{q}}_{k+1}\}$, the stopping condition (1) is then evaluated and the simulation is stopped or continued accordingly.

In this work, it is assumed that the main control law of the robot never reaches a state such that joint position and velocity limits may be reached. This is a strong assumption for robots evolving in dynamic environment as it raises the question of viability of the control input [13]. Yet, this is clearly out of scope for this paper.

B. Initial states generation

For the purpose of learning and testing a braking policy from any initial state, an initial state generator is required. As the neural network used to capture the learnt policy needs to be trained on a high range of different states, the state generator must cover the state space as much as possible. Moreover, the generator needs to generate only feasible states that can actually be reached by the robot[1].

In this work, a method inspired by the notion of "exciting trajectories" [14] is used to generate feasible states. Such trajectories are computed as:

$$q_{i_k} = \sum_{l=1}^{N} \frac{a_i}{F_{[l,i]}} \sin\left(F_{[l,i]} k\Delta t + B_{a[l,i]}\right) - \frac{b_i}{F_{[l,i]}} \cos\left(F_{[l,i]} k\Delta t + B_{b[l,i]}\right) \quad (11)$$

where k is the considered time instant and i the index of the considered DOF. N is a scalar which determines the number of sine and cosine functions to be used (in this work $N = 3$). $\mathbf{F}$ is a $N \times n$ matrix containing prime numbers, with n the number of DOF ($F_{[l,i]}$ is the component of $\mathbf{F}$ on the l-th line and i-th column). Using prime numbers allows to remove periodicity between the different sine and cosine signals. In this work, the $n \times N$ first prime numbers are used and by shuffling the matrix each time a new trajectory is generated, the exploration of the state space is improved. $\mathbf{B_a}$ and $\mathbf{B_b}$ are matrices, used to phase shift the sine and cosine functions. They contain random numbers between 0 and 2π. The values in vectors $\mathbf{a}$ and $\mathbf{b}$ must satisfy the constraints on $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ while being maximized in order to improve the exploration of the state space. In first approximation, these values are computed without accounting for the complex bounds of the true feasible state space:

$$a_i = b_i = \min\left\{ \frac{q_{i,max}}{2\sum_{l=1}^{N} \frac{1}{F_{[l,i]}}}, \frac{\dot{q}_{i,max}}{2N}, \frac{\ddot{q}_{i,max}}{2\sum_{l=1}^{N} F_{[l,i]}} \right\} \quad (12)$$

where $\ddot{q}_{i,max}$ is the constant bound provided by the manufacturer.

To generate a new initial state, a trajectory is generated and a point of the resulting trajectory is selected randomly by drawing a random time instant k. The position is derived to compute the associated velocity and acceleration.

C. Reinforcement learning

The proposed approach method uses deep reinforcement learning to improve the policy given by the TAP. This allows to learn a neural network which gives $\Delta\dot{\boldsymbol{\tau}}$ as an output. The state space S of the associated Markov decision process is described by $3n$ continuous variables corresponding to $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$. The reward function R is formulated as:

$$R(S) = \begin{cases} -1 & \text{if } \exists \dot{\mathbf{q}}_n \geq \epsilon \text{ or } \exists \ddot{\mathbf{q}}_n \geq \epsilon \\ r & \text{otherwise.} \end{cases} \quad (13)$$

where r is scalar with an arbitrary large value. In this work, $r = 100$. This function encourages the robot to reach a stopped state as fast as possible. So as to learn the optimal $\Delta\dot{\boldsymbol{\tau}}_i$ for each DOF i, the action space A is defined by n values. Considering the high dimensionality of the problem, the proposed method uses 5 discrete values for each action $\Delta\dot{\boldsymbol{\tau}}_i$. By reducing the necessary exploration, the learning process converges more easily toward a good solution. The discrete values are defined by a vector of hyper-parameters $\Delta\mathbf{r}$:

$$\forall i \in \{1...n\},\ \Delta\dot{\tau}_i \in \{-\Delta r_i, -\frac{\Delta r_i}{2}, 0, \frac{\Delta r_i}{2}, \Delta r_i\} \quad (14)$$

It is important to note that when $\Delta\dot{\boldsymbol{\tau}}_i = 0$, the controller sends the value $\dot{\boldsymbol{\tau}}_i = \dot{\boldsymbol{\tau}}_{TAP\ i}$ to the DOF i of the robot. When the robot is in a state which allows to reach the stopped state at the next timestep, the optimal action space is very narrow and using the value $\dot{\boldsymbol{\tau}}_{TAP}$ given by the TAP is often optimal.

The neural network is learned using a deep reinforcement learning algorithm called Proximal Policy Optimization (PPO) as it can handle multi-discrete action spaces.

The previously described TAP is designed to reach the acceleration $\ddot{\mathbf{q}}_{max}$ given by the manufacturer. As the TAP is recomputed at each time step, if the current acceleration is over $\ddot{\mathbf{q}}_{max}$, the TAP applies a negative jerk to reduce it. When using the neural network over the TAP, the TAP thus prevents exploring trajectories that uses high values of acceleration. As an aspect of the neural network's role is, in some states, to reach accelerations that are over $\ddot{\mathbf{q}}_{max}$, the TAP must be adapted when used with the neural network. Instead of using a constant value $\ddot{\mathbf{q}}_{max}$ given by the manufacturer, the value $\ddot{\mathbf{q}}_{max,k} = max(\ddot{\mathbf{q}}_{max}, \ddot{\mathbf{q}}_{k-1})$ is used at each time step k. Using the maximum between the $\ddot{\mathbf{q}}_{max}$ given by the manufacturer and the previous acceleration ensures that the TAP will not prevent the learning system from reaching higher accelerations. The TAP may however overestimate the real robot's capacities and the role of the neural network is also to prevent the learning system from overestimating these capacities.

V. Experiments and results

The results presented in this section can be reproduced using the code available online[2].

A. Robot

The experiments are carried out using a simplified version of the Franka Emika Panda robot where only the second and fourth DOF are being controlled while the others DOF are locked. It results in a 2 DOF planar robot affected by the gravity. The robot dynamics is simulated in Python using the Pinocchio library [15]. The real limits on the velocity and on the torque, respectively $\dot{\mathbf{q}}_{max}$ and $\boldsymbol{\tau}_{max}$, given by the manufacturer are used, as well as the lower boundaries of the maximum reachable acceleration $\ddot{\mathbf{q}}_{max}$.

The values of $\dddot{\mathbf{q}}_{max}$ given by the manufacturer overestimate the robot's real capacities and are, consequently, not usable. Indeed, when applying a jerk $\dddot{\mathbf{q}}_{max}$ to the robot, the derivative of the torque $\dot{\boldsymbol{\tau}}$ often exceeds the maximum values $\dot{\boldsymbol{\tau}}_{max}$. Yet, its is not clear how these values have been chosen by the manufacturer. For the purpose of this work, in order to leave space for exploration to the learning algorithm, the values of $\dot{\boldsymbol{\tau}}_{max}$ are chosen $5\times$ larger than the ones provided by the manufacturer. Similarly, the values of $\dddot{\mathbf{q}}_{max}$ used for computing the TAP are chosen $5\times$ lower than the manufacturer's provided limits.

[1]Starting from an unreachable initial state may bias the learning towards an unrealistic control policy

[2]The code used to reproduce the results of this paper is available at: `https://gitlab.inria.fr/auctus-team/publications/shared-paper-code/iros2022-brake`

B. Experiments

The proposed deep reinforcement learning architecture is applied on the previously defined planar robot. The learned neural network is made up of 3 hidden layers of 10 neurons each, as this neural architecture produced the best results on this particular robot. The python library Stable-Baselines3 [16] is used to implement PPO.

The values of $\Delta\mathbf{r}$ must be chosen carefully as it controls how much the learning process is guided by the TAP. A too low value would force the learned policy to be similar to the policy given by the TAP and a too high value would let the learning process explore very different policies, which may prevent the learning process to converge. In this experiments, the value of Δr_i is the same for each DOF, hence $\Delta\dot{\tau}_i \in \{-200, -100, 0, 100, 200\}$.

After learning with a budget of 400k timesteps, the final policy is tested on a test set of initial states and compared with the results obtained when only using the TAP. The test set is created by randomly generating 500 initial states using the exciting trajectories. The comparison between the learned controller and the TAP alone is done by starting a braking episode from each initial state and observing the average braking time and the average value of γ, computed as follows:

$$\gamma = \sum_{k} \sum_{i=1}^{n} |\boldsymbol{\tau}_{i\,k}| \tag{15}$$

where k corresponds to each timestep. The value of γ is correlated with the energy spent during a braking episode and can thus be used to compare the energetic efficiency.

C. Comparison with the TAP on the test set

Considering the stochasticity of the learning process, the experiment needs to be repeated several times. The learning process in launched 25 times, and all the resulting controllers are tested on the same test set. The table I shows the performances on the test set of the TAP and the performance of the learned policy, averaged over the 25 learned controllers. The test set is divided in 3 sub test sets, depending on the kinetic energy of the initial states. The kinetic energy is computed as follows :

$$K(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} \tag{16}$$

As one might expect, these results show that a higher initial kinetic energy implies a longer braking time as well as a higher energy consumption for both methods. However, on average, using the learned neural network allows to brake faster and to reduce the power consumption on each sub test set. Additionally, the time gains get higher when the initial kinetic energy increases. Considering that initial states with higher kinetic energy are the most dangerous in the context of collaborative robotics, this remark emphasizes the usefulness of the method presented in this paper.

Despite the stochasticity of the method presented here, the results obtained with the 25 learned controllers are quite stable. Most of the learned controllers achieve better results than the TAP on both metrics and on each sub test set, with only one learned controller achieving both higher braking time and higher value of γ than the TAP on the sub test set where $0.2 \leq K < 0.4$.

These results demonstrate that learning a value $\Delta\dot{\boldsymbol{\tau}}$ over the TAP allows to improve significantly the braking efficiency.

D. Comparison with the TAP during a braking episode

Figure 4 shows an example of a braking trajectory on the first DOF when starting from a random initial state and using each methods. Each method works and reaches the stopped state, however, the policy learned via reinforcement learning is faster. This is due to applying a higher acceleration and jerk. Indeed, when using the neural network, both the maximal acceleration and the maximal jerk are above the lower boundaries $\ddot{\mathbf{q}}_{max}$ and $\dddot{\mathbf{q}}_{max}$. In that example, the better use of the robot's capacities is reflected by a higher absolute value of the maximum torque $\boldsymbol{\tau}$ and derivative of the torque $\dot{\boldsymbol{\tau}}$.

The actions $\Delta\dot{\boldsymbol{\tau}}$ chosen by the neural network often follow the same pattern: a first phase where the action has a negative value (when the velocity is positive) followed by a second phase with actions of positive values and an final phase where the action is 0. Using an action equal to zero means letting the TAP control the corresponding DOF and is optimal in some situation, as mentioned in the sub-section IV-C.

VI. Conclusion

The method proposed in this paper uses a controller composed a both the TAP and a learned neural network and does not make any simplification regarding the feasible acceleration $\ddot{\mathbf{q}}_{max}$ and jerk $\dddot{\mathbf{q}}_{max}$. It allows to exploit the actuation capacities efficiently and, as a consequence, to brake faster than when using methods traditionally used in the robotics literature. It also improves significantly the energetic efficiency of the braking. This work illustrates how machine learning methods, such as reinforcement learning, may improve both the temporal and energetic efficiency of robots evolving in dynamic contexts such as the ones of collaborative robotics where safety and efficiency are at stake.

A. Limitations

The main limitation of the work presented here is the absence of tests on a real world robot. The proposed algorithm has only been tested on a simulator and, to ensure its safety and its efficiency in the context of collaborative robotics, testing this method on a real robot appears essential.

Another important point to raise is that, for simplicity, the limits regarding the position of each joints have not been taken into account during this work. For this reason, the braking policies learned in simulation may not be feasible on a real robot. To apply this method on a real world robot, a more complex simulator would be needed and collision events should be taken into account. For instance, modifying the reward function by penalizing the robot depending on the impact force could allow to minimize the collision damages.

B. Future works

In this work, the action space is limited to 5 discrete actions per DOF and the actions values are restricted by $\Delta\mathbf{r}$. Although, due to the increasing complexity of exploration, experiments with continuous actions and with higher value of $\Delta\mathbf{r}$ give poorer results, using continuous actions or higher values of $\Delta\mathbf{r}$ could allow the algorithm to explore more solutions and thus improve the braking efficiency.

Lastly, even though the current method improves significantly the braking efficiency of a 2 DOF planar robot, many real world robots have more DOF and, hence, a higher complexity. When using this method on the 7 DOF Panda robot, the efficiency gains

TABLE I: Average braking time and value of γ on the test set using the TAP method and the learning system (TAP+$\Delta\dot{\boldsymbol{\tau}}$). The values given for the learning system are computed by averaging the average braking time on the test set of the 25 learned controllers and the standard deviation computed on the 25 learned controllers is given in parenthesis. The gains columns shows the efficiency gain when using TAP+$\Delta\dot{\boldsymbol{\tau}}$, in comparison with the TAP alone. For each initial state, the ratio between the performance difference and the performance of the TAP is computed. The average value of this ratio on the initial states and on the 25 learned controllers is then computed and shown as a percentage.

Kinetic energy (J)	TAP Time (ms)	TAP+$\Delta\dot{\boldsymbol{\tau}}$ Time (ms)	Time gains	TAP γ (N.m)	TAP+$\Delta\dot{\boldsymbol{\tau}}$ γ (N.m)	γ gains
$K < 0.2$	47.5	41.8 ($\pm$ 2.0)	8.2%	1900.8	1688.5 ($\pm$ 74.8)	7.2%
$0.2 \leq K < 0.4$	73.7	60.1 ($\pm$ 4.8)	16.0%	3149.1	2711.1 ($\pm$ 165.9)	10.2%
$0.4 \leq K$	101.2	76.1 ($\pm$ 4.8)	22.3%	4028.7	3461.2 ($\pm$ 138.4)	6.7%

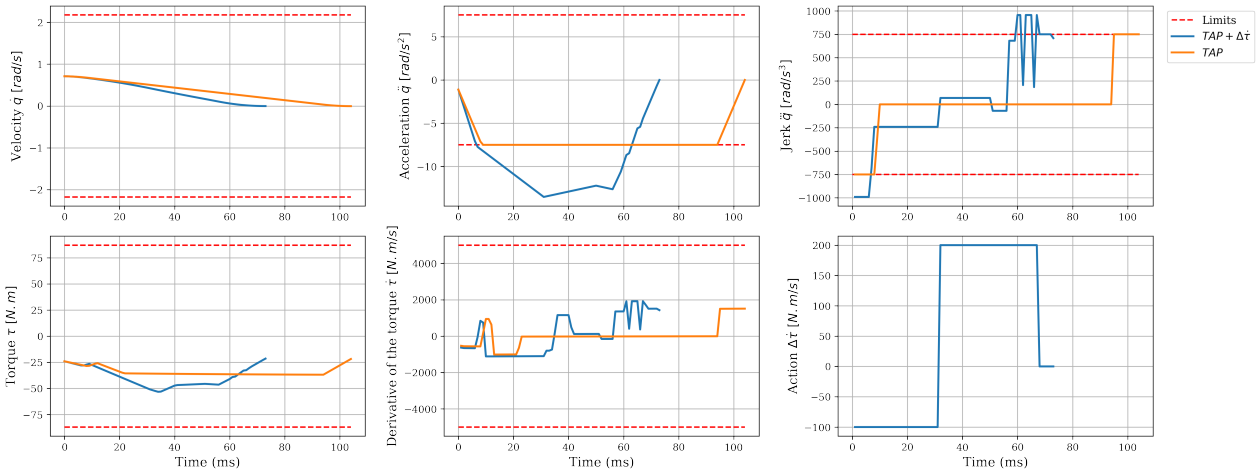

Fig. 4: Evolution of $\dot{\mathbf{q}}_1$, $\ddot{\mathbf{q}}_1$, $\dddot{\mathbf{q}}_1$, $\boldsymbol{\tau}_1$, $\dot{\boldsymbol{\tau}}_1$ and $\Delta\dot{\boldsymbol{\tau}}_1$ during a braking episode using the TAP method and the learning system (TAP+$\Delta\dot{\boldsymbol{\tau}}$). Both methods start from the same random initial state.

are much lower than with the 2 DOF robot. Indeed, tuning the values of $\Delta\mathbf{r}$ becomes much more complicated when the number of DOF increases. The increase in complexity forces the learning process to be more guided by the TAP, which means that smaller values of $\Delta\mathbf{r}$ must be chosen. Additionally, since the inertail properties of robots vary largely from the first degrees to the last ones, the torque needed to create a movement is usually much higher for the first DOFs. Therefore, using different values of $\Delta\mathbf{r}_i$ for each DOF is probably more adequate. The research of methods allowing to adapt the values of $\Delta\mathbf{r}$ to each DOF is thus a promising research subject for the purpose of improving the braking efficiency of high-DOF robots.

References

[1] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[2] S. Haddadin, *Formal Modeling and Verification of Cyber-Physical Systems: 1st International Summer School on Methods and Tools for the Design of Digital Systems, Bremen, Germany, September 2015*. Springer Fachmedien Wiesbaden, 2015, ch. Physical Safety in Robotics, pp. 249–271.

[3] T. Rokahr, A. Spenninger, and C. Calafell Garcia, "German patent de102019112023a1, braking device for a drive device of a robot," Nov. 2020. [Online]. Available: https://patents.google.com/patent/DE102019112023A1/en

[4] L. Joseph, J. K. Pickard, V. Padois, and D. Daney, "Online velocity constraint adaptation for safe and efficient human-robot workspace sharing," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 045–11 051.

[5] S. Haddadin, A. Albu-Schaffer, and G. Hirzinger, "The role of the robot mass and velocity in physical human-robot interaction-part i: Non-constrained blunt impacts," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1331–1338.

[6] A. Skuric, V. Padois, and D. Daney, "On-line force capability evaluation based on efficient polytope vertex search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1700–1706.

[7] K. M. Lynch and F. C. Park, 2017, ch. Trajectory Generation.

[8] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7330–7336.

[9] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review," *Robotics*, vol. 10, no. 1, 2021.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] Franka control interface documentation — franka control interface (FCI) documentation. [Online]. Available: https://frankaemika.github.io/docs/index.html

[12] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 2677–2682.

[13] S. Rubrecht, V. Padois, P. Bidaud, M. De Broissia, and M. Da Silva Simoes, "Motion safety and constraints compatibility for multibody robots," *Autonomous Robots*, vol. 32, no. 3, pp. 333–349, 2012.

[14] J. Swevers, C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE transactions on robotics and automation*, vol. 13, no. 5, pp. 730–740, 1997.

[15] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," 2015–2021.

[16] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.