University of Trento
Department of Industrial Engineering

UNIVERSITY OF TRENTO

Optimization Based Robot Control

Assignment 02

Taking Account of Underactuation in DDP

Professor:
Prof. Andrea Del Prete

Students:

Lorenzo Colturato	lorenzo.colturato@studenti.unitn.it	233301	from DII
Luca Zardini	luca.zardini@studenti.unitn.it	229366	from DISI

Academic Year 2022-2023

Additional Penalty Implementation

In order to implement the Additional Penalty method to under-actuate the double pendulum, a penalizing term is added in the running cost, which is then used in the DDP algorithm. Below is the implementation of the additional term of the running cost:

$$\begin{aligned}\text{running cost} &= \ldots + \frac{1}{2}\left(\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}u_1\\ u_2\end{bmatrix}\right)^T \cdot \mathit{underact}\cdot\left(\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}u_1\\ u_2\end{bmatrix}\right)\\ &= \ldots + \frac{1}{2}\cdot\begin{bmatrix}0 & u_2\end{bmatrix}\cdot \mathit{underact}\cdot\begin{bmatrix}0\\ u_2\end{bmatrix}\\ &= \ldots + \frac{1}{2}\cdot \mathit{underact}\cdot u_2^2\end{aligned}$$

where:

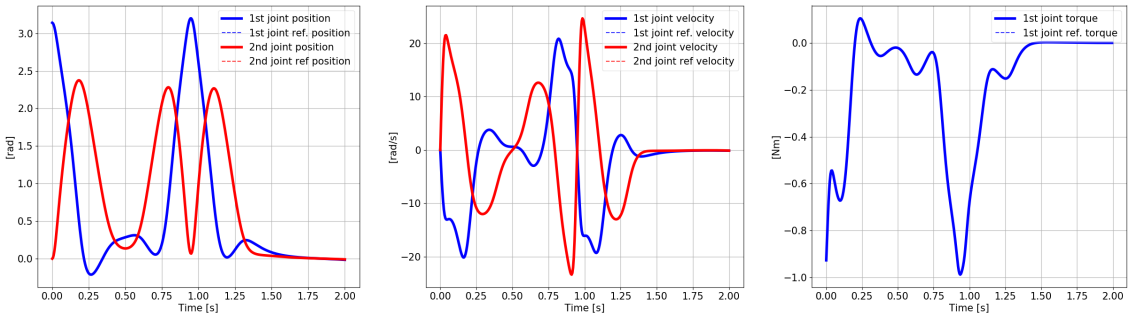
- $\mathit{underact}$ is the weight of the penalty set as follows: $\mathit{underact} = 1\cdot 10^6$
- $\vec{u} = \begin{bmatrix}u_1\\ u_2\end{bmatrix}$ is the vector of controls, i.e. the torques at the 1^{st} and 2^{nd} joint

Since we want to under-actuate the 2^{nd} joint, $\vec{u}$ is multiplied by matrix $\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}$ in order to penalize only the torque applied at the passive joint. This additional term makes it necessary to modify the gradient and the hessian of the running cost with respect to $\vec{u}$, as did below:

$$\begin{aligned}\frac{\partial\,(\text{running cost})}{\partial\vec{u}} &= \ldots + \mathit{underact}\cdot\begin{bmatrix}0\\ u_2\end{bmatrix}\\ \frac{\partial^2\,(\text{running cost})}{\partial\vec{u}^2} &= \ldots + \mathit{underact}\cdot\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\end{aligned}$$

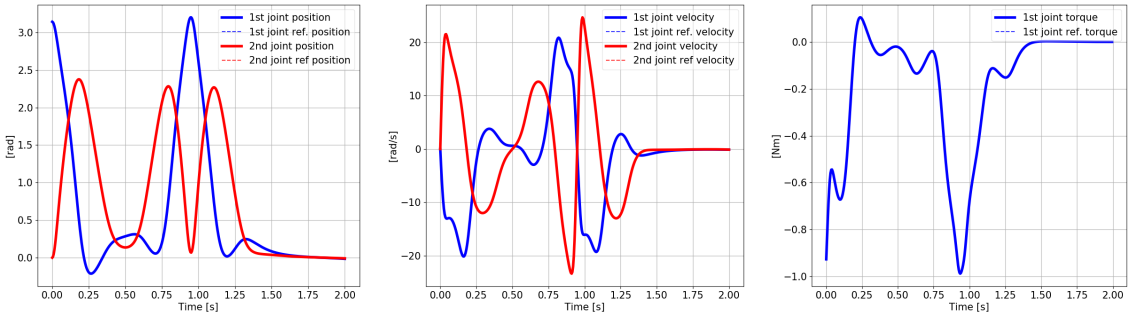
1st QUESTION

Figure 1 and 2 show the position and the velocity of the two joints and the torque at the actuated joint (i.e. the first one) when the selection matrix approach and the additional penalty approach are used.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

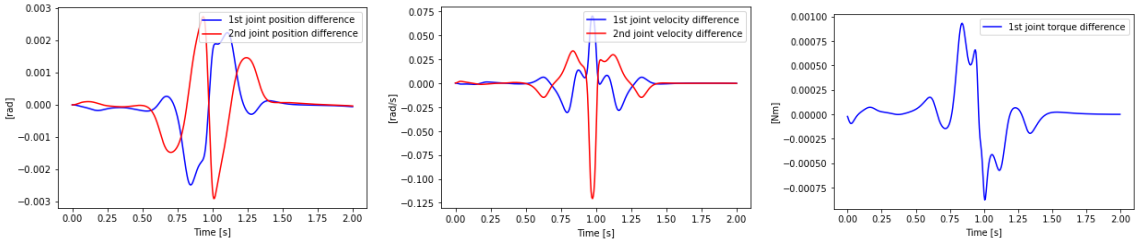
Figure 1: Results of the selection matrix method



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 2: Results of the additional penalty method

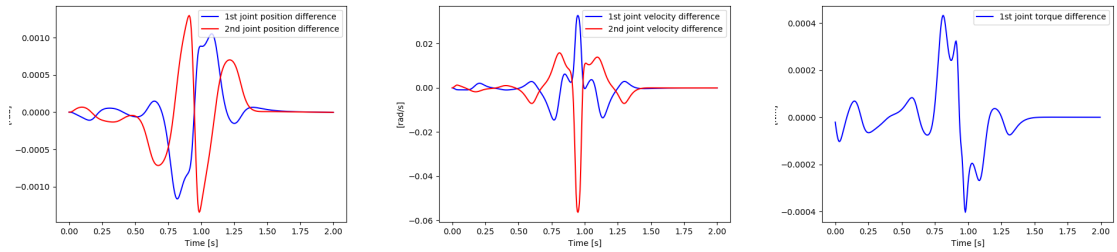
From a graphical point of view, the best way to check if the results obtained from two different methods are the same is to calculate the difference between the two results and plot it over the entire time axis: this allows us to see if the error is close to zero. The difference in the results of the two methods is shown in Figure 3, both for the position and the velocity of the two joints, and for the torque at the first joint.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 3: Difference in the results of the two methods in the simulation

Then we have analyzed the reference trajectories and torque coming from the DDP (without simulating the system) using the two approaches, and we have seen that also the reference curves are slightly different, as reported in Figure 4.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 4: Difference in the results of the two methods in the DDP computation

With respect to the errors in the simulation plots of Figure 3, we can say that the differences between the computed reference trajectories by the DDP with the two methods (Figure 4) are much lower. Table 1 collects the maximum, minimum and mean errors between the solutions found by the DDP when using the additional penalty method and when using the selection matrix method. The errors between the two methods are reported in absolute value.

		Mean error	**Min error**	**Max error**
Joint 1	**position**	$1.65 \cdot 10^{-6}$	0.0	$1.16 \cdot 10^{-3}$
	velocity	$1.78 \cdot 10^{-6}$	0.0	$3.27 \cdot 10^{-2}$
	torque	$2.83 \cdot 10^{-6}$	0.0	$4.31 \cdot 10^{-4}$
Joint 2	**position**	$9.92 \cdot 10^{-6}$	0.0	$1.34 \cdot 10^{-3}$
	velocity	$1.03 \cdot 10^{-6}$	0.0	$5.63 \cdot 10^{-2}$
	torque	$4.16 \cdot 10^{-7}$	0.0	$2.04 \cdot 10^{-5}$

Table 1: Differences in DDP solutions using different approaches: Additional Penalty vs Selection Matrix

We have also compared the solutions found simulating the system with the two approaches with the solutions found by the DDP with the same two approaches, i.e. we have found the differences between the simulated results and the reference results coming from the DDP when using the selection matrix approach and then we have done the same also when using the additional penalty approach. What we have found is collected in Table 2 (the errors are expressed in absolute value).

		Selection Matrix			**Additional Penalty**		
		Mean error	**Min error**	**Max error**	**Mean error**	**Min error**	**Max error**
Joint 1	**position**	0.0	0.0	0.0	$2.76 \cdot 10^{-5}$	0.0	$1.30 \cdot 10^{-4}$
	velocity	0.0	0.0	0.0	$3.17 \cdot 10^{-5}$	0.0	$1.44 \cdot 10^{-3}$
	torque	0.0	0.0	0.0	$8.80 \cdot 10^{-6}$	0.0	$8.72 \cdot 10^{-5}$
Joint 2	**position**	0.0	0.0	0.0	$1.40 \cdot 10^{-5}$	0.0	$1.02 \cdot 10^{-4}$
	velocity	0.0	0.0	0.0	$2.05 \cdot 10^{-5}$	0.0	$1.78 \cdot 10^{-3}$
	torque	0.0	0.0	0.0	$4.16 \cdot 10^{-7}$	0.0	$2.04 \cdot 10^{-5}$

Table 2: Mean, Min. and Max. error between simulated and reference curves for the two approaches

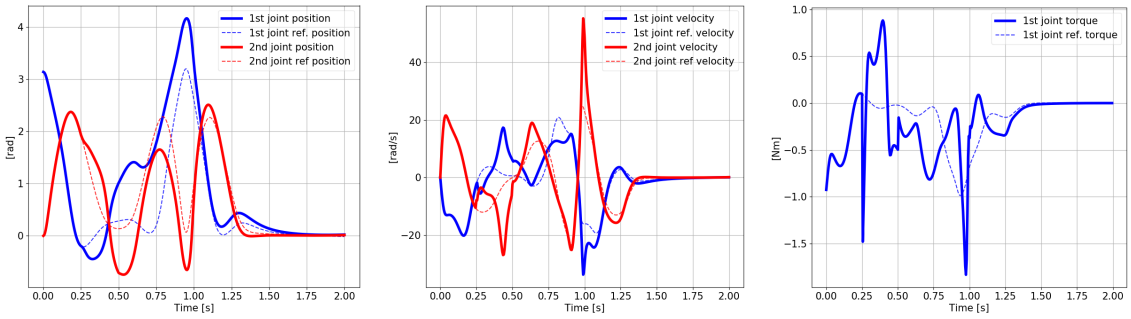
In particular, from Table 2 it is possible to notice that for the selection matrix approach there are no differences/errors between the reference curves and the simulated ones, whereas the additional penalty method has some uncertainties. This can be explained as follows: the selection matrix approach acts directly on the system dynamics (in particular on the partial derivative w.r.t. $\vec{u}$ of the dynamics, where the inverse of the joint space inertia matrix, which maps the joint accelerations to the joint torques, is multiplied by

the transpose of the selection matrix) where the torque at the passive joint is put to zero. In this way, the input to the DDP solver and the simulation do not take into account the second joint torque, resulting in a in a perfect tracking of the reference curves; on the other hand, the additional penalty method acts passively during the computation of the running cost. More in detail, we apply a specific parameter *underact* as weight to penalize the additional penalty in the running cost. This does not completely remove the torque at the passive joint, but tries to limit it by making it as close to zero as possible. This results in slightly different trajectory and torque solutions in the DDP solver.

To support our considerations, we have tried to simulate the system with different values of the *underact* weight and we have seen that the best result is achieved with *underact* set to $1{\cdot}10^{15}$. With higher values the cost explodes, while with lower values the DDP solver finds a solution that, to a very small extent, is worse.

2nd QUESTION

When we set the PUSH flag to *true* it means that, at 0.25 s, 0.50 s and 1.00 s of simulation, the passive joint receives a push which takes the form of an instantaneous velocity increase equal to 3 rad/s. This three pushes have an effect only in the simulated trajectories and do not affect the reference ones computed by the DDP. Figure 5 shows the differences between the reference trajectories and the reference torque and the simulated counterparts obtained with the applied pushes.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 5: Tracking performance of the reference trajectories when PUSH is *true*

Table 3 collects the maximum, minimum and average tracking errors computed in absolute value. It shows how the joints' position and velocity are affected by an external push. In particular, the highest error is obtained at the second joint velocity, which is where the push acts directly.

		Mean error	**Min error**	**Max error**
Joint 1	**position**	$3.27 \cdot 10^{-1}$	0.0	1.60
	velocity	$1.97 \cdot 10^{-2}$	0.0	17.6
	torque	$1.93 \cdot 10^{-2}$	0.0	1.57
Joint 2	**position**	$1.40 \cdot 10^{-1}$	0.0	$9.08 \cdot 10^{-1}$
	velocity	$2.40 \cdot 10^{-2}$	0.0	30.8
	torque	0.0	0.0	0.0

Table 3: Tracking errors when PUSH is *true* with the selection matrix approach

Observing Table 3 we notice that the largest errors are on the angular velocity of the two joints and this is due to the fact that the push is in fact defined as an instantaneous increase in the velocity of the passive joint by a factor of 3 rad/s.

From Figure 5 it can be seen that, before the first push, the simulated curves track perfectly the reference ones. Before the first push arrives, the pendulum starts to move upwards as a consequence of the torque

applied at the actuated joint. At 0.25 s the first joint angle is about -0.196 rad while the second joint angle is about 1.93 rad, meaning that the first link, rotating clockwise, has passed the positive vertical axis reaching the configuration of Figure 6.

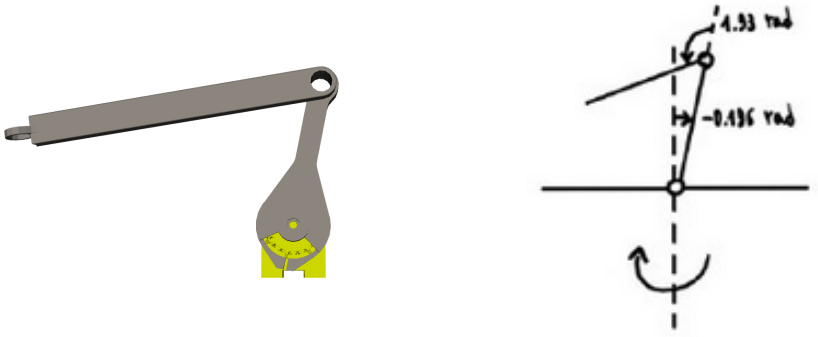


Figure 6: System configuration at 0.25 s

The push acts to increase the velocity of the passive joint and, in doing so, it affects also the motion of the actuated one. Without the push, at 0.25 s, the angle of the first joint starts to increase (and so the first link starts to go down). However, the push causes the first joint to further rotation towards even more negative angles and consequently shifts the subsequent descent phase of the first link (and therefore the increase of the angle of the first joint) towards a longer time. For the passive joint, it can be seen that, before the push, the angle of the joint was about to decrease (meaning that the second link is going to align upward with the first) and, after the push and compared to the no-push case, the angle decreases more slowly (i.e. with a smaller slope). Furthermore, after the push, the angle reaches much lower values, rotating upwards and opposite with respect to the first joint. Since the DDP gives feedback gains, they can be used to steer the system in the presence of noise or disturbances such as the applied external pushes. However, in the initial stages of the simulation, since the second push arrives just after 0.25 s w.r.t. the first one, the gains are not such as to allow the simulated trajectories to return to track perfectly the reference ones. For this reason, when the system receives the second push at 0.5 s (being in the configuration of Figure 7), there is an increase in the velocity of the passive joint, which goes up to 20 rad/s.

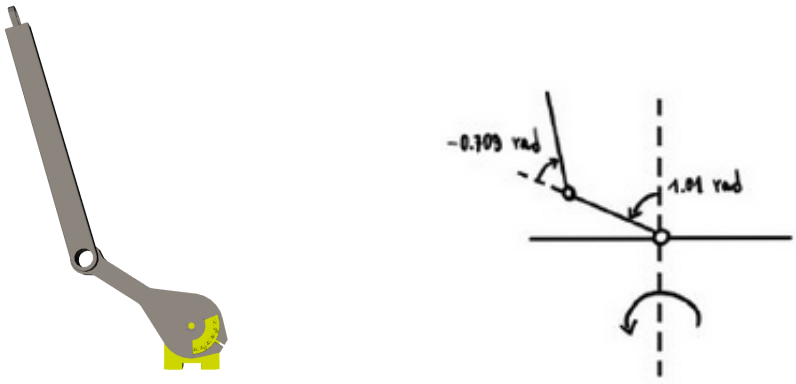


Figure 7: System configuration at 0.50 s

Under the thrust received by the downward rotation at high velocity of the passive joint, the actuated joint reaches an angle greater than 4 rad, which means that, in contrast to the no-push case where the joint on descent stops at 3.14 rad, it rotates more going beyond the negative vertical axis. At 1 s, when the system has reached the configuration of Figure 8, the passive joint receives a final push (which helps the upward clockwise rotation) that increases its velocity to 40 rad/s and the velocity of the first joint to -29.71

rad/s (negative because its angle decreases when it rotates upwards in clockwise direction). This allows the system to rotate upwards to the point where the angle of both joints goes to zero, meaning that both links are aligned to the positive vertical axis, which is the desired goal.

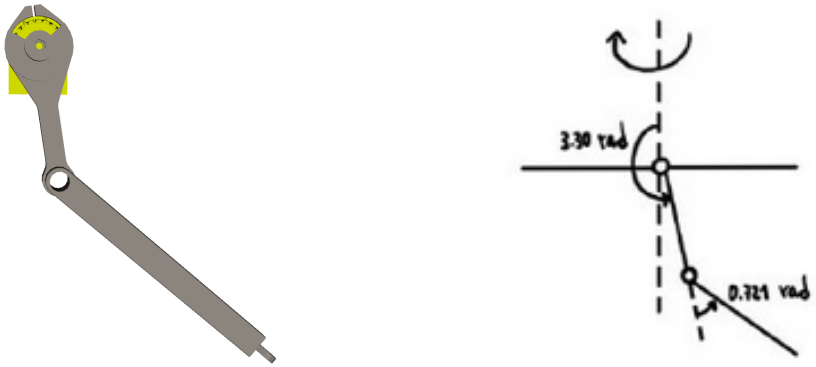
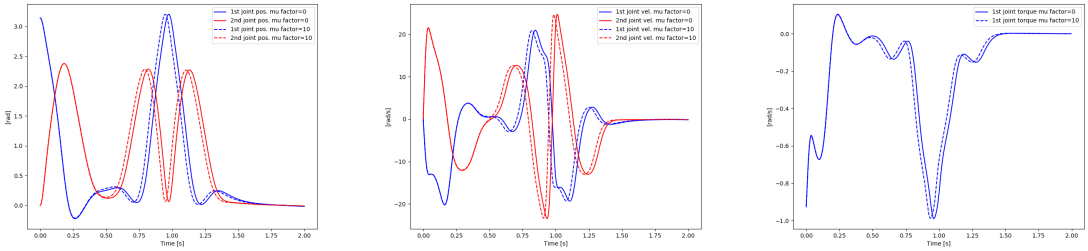


Figure 8: System configuration at 1 s

The pushes affect also the torque at the actuated joint. When *PUSH* is set to *false* the torque at the first joint is given entirely by the torque supplied by the motor; when the pushes are considered, since they are interpreted as a velocity on the passive joint, they change the velocity of the two joints and so they provide an angular acceleration (or deceleration). In this way, the inertia of the first link determines the change in the torque needed for the new angular acceleration of the first joint about the vertical axis, which is the rotational axis. This can be seen in Figure 5c, noticing that the torque at the actuated joint gets significantly decreased at 0.25 s and 1 s. Furthermore, from the case without the push, we can see that the torque almost always remains negative. This is due to the fact that, to rotate the system upwards, it is necessary to rotate it clockwise, which causes a decrease in the angle of the first joint: a negative torque is required to do this. When the push is active it is noted that, between 0.25 s and 0.50 s, the torque becomes positive. This can be explained as follows: when the pendulum receives the first push at 0.25 s it is rotating downwards and thus counter-clockwise, however the push tends to bring the system back to rotate upwards. To continue following the reference trajectory generated by the DDP, and therefore continue to rotate downwards by increasing the angle at the first joint, the motor which actuates the first joint must necessarily supply a positive torque.

3rd QUESTION

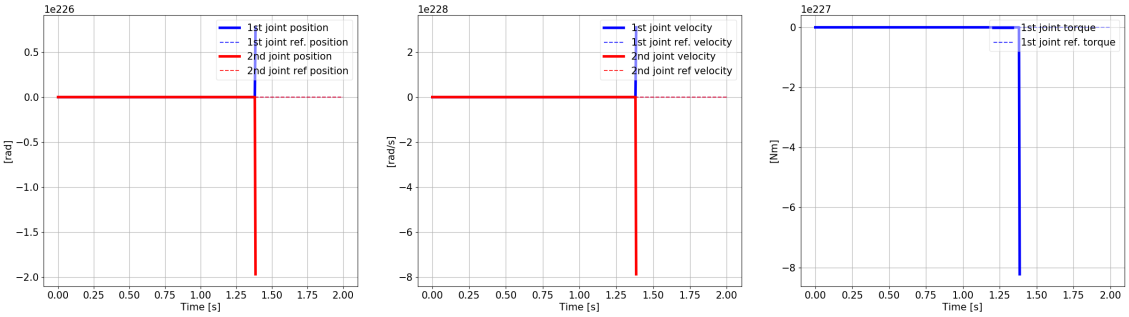
Carrying out the same simulation as in the 2nd question (i.e. with the three pushes at 0.25 s, 0.5 s and 1 s), but setting *mu_factor* to 0, meaning that the regularization term μ remains constant at the value 10, we noticed that the simulation returns *NaN* values both for the cost and the effort of the simulated curves. Figure 9 shows the solutions computed by the DDP when *mu_factor* is first set to 0 and then to 10. It is possible to see that, after an initial overlapping of the trajectories, when the regularization term varies at each iteration the curves shift slightly to the left, i.e. the optimization values in the subsequent steps are found at slightly shorter times.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 9: Comparison between the reference curves computed by the DDP when *mu_factor* is equal to 0 and 10

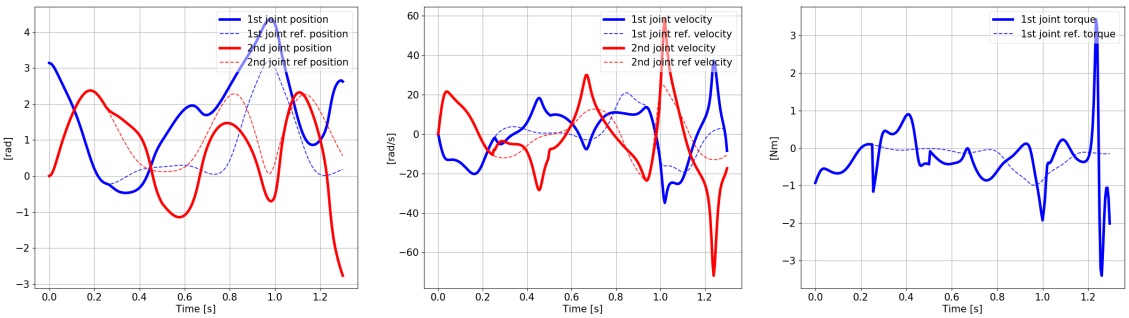
In general it is good practice to have adaptive regularization because there are problems in which, in some points, the line search fails, thus requiring to increase μ, which penalizes deviations from the previous solution. There can also be points where the line search succeeds. At these points, it is preferred to decrease μ in such a way as to make the optimization faster and therefore converge faster. Constant regularization works for easy problems but probably requires more time to converge to the solution. Indeed, the DDP with adaptive regularization takes 33 iterations to converge to the desired objective, whereas the one with constant regularization takes 37 iterations. This assumption does not hold for more complex problems. Figure 10 shows the results of the simulation when *mu_factor* is 0 and when *PUSH* is *true*.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 10: Position, velocity and torque of the joints with *mu_factor* = 0 and *PUSH* = *true*. Entire time axis

As can be seen in Figure 10, the solution does not converge but reaches very large values (till 10^{228}), approximately at 1.4 s. We therefore tried to stop the simulation at 1.3 s to see the trend of the curves before they diverge (see Figure 11).

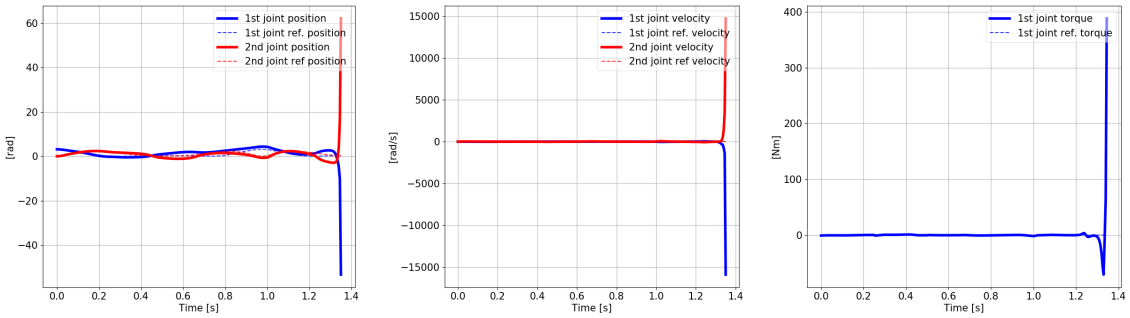


(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 11: Position, velocity and torque of the joints with *mu_factor* = 0 and *PUSH* = *true*. Till 1.3 s

Figure 11 shows that up to about 1.3 seconds (after all three pushes have been made), the state and

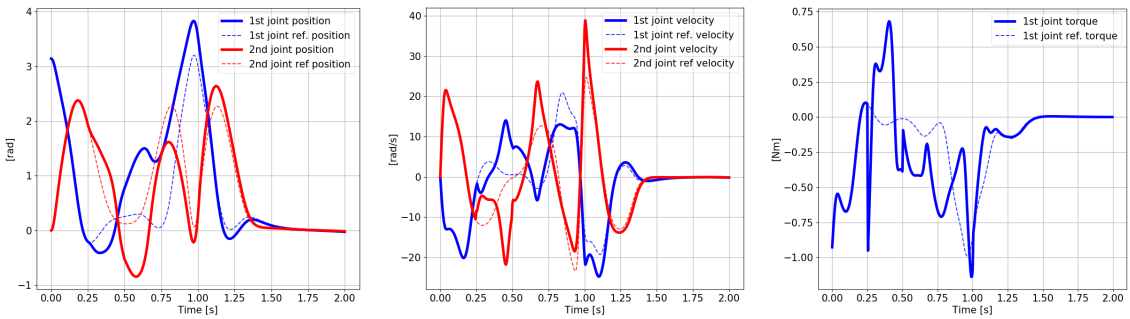
control have reasonable but wrong values respect to their respective references. In the next 0.1 seconds, as shown in Figure 12, the position, velocity and torque start to diverge reaching very high values. These values are so high that the code can no longer handle them, starting to introduce *NaN* values.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 12: Position, velocity and torque of the joints with $mu\_factor = 0$ and $PUSH = true$. Till 1.4 s

We have tried to reduce the velocity increment from 3 rad/s to 2.5 rad/s and, as shown in Figure 13, the simulation works and the system reaches the desired goal.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure 13: Position, velocity and torque of the joints with $mu\_factor = 0$ and $PUSH = true$. $PUSH = \{0, 2.5\}$

In conclusion, we have found that the DDP solution (with no push) obtained with constant regularization slightly diverges from the one obtained with adaptive regularization. This difference is relevant when external pushes are applied to the system: with this value of regularization we have seen that the applicable push (on the passive joint) limit value is about 2.7 rad/s. With higher values the system does not reach the desired goal.

Appendix A

Torque limit implementation

We now want to add an inequality constraint on the control, limiting the maximum torque at the actuated joint. The optimization problem becomes of the following type:

$$\begin{cases} \text{Minimize} & \text{cost function} \\ \text{subject to} & \vec{u} \leq max\_torque \end{cases}$$

where max_torque is the maximum allowable torque at the 1^{st} joint. Since the DDP does not allow to have inequality constraints, to implement the control bound we have to add a term in the running cost that takes care of it. The implementation is done below:

$$\text{running cost} = \ldots + \frac{1}{2} \cdot w\_bounds \cdot \beta^2 \cdot \log\left(max\_torque - \vec{u}\right)^2$$

where:

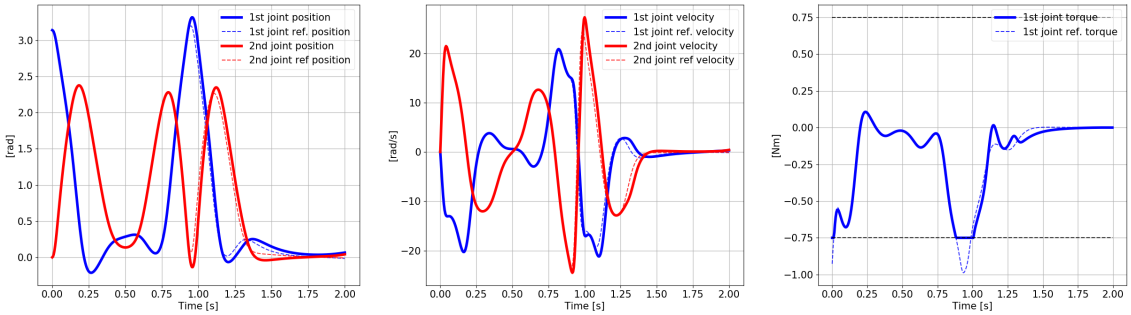
- β is a free parameter, which we set to $1{\cdot}10^{-3}$, that is used in order to guarantee that the additional term in the running cost is not discontinuous
- $\beta \cdot \log\left(max\_torque - \vec{u}\right)$ is the barrier function that allows to replace the inequality constraint by a penalizing term in the objective function that is easier to handle
- w_bounds is the penalty we give to the barrier function in order to satisfy the constraint

We decided to use a logarithmic barrier function because it introduces a gradient to the function being optimized that favors less extreme values of $\vec{u}$ (in this case values lower than max_torque), while having relatively low impact on the function away from these extremes.

The gradient and the hessian of the running cost are computed as follows:

$$\frac{\partial\left(\text{running cost}\right)}{\partial \vec{u}} = \ldots - w\_bounds \cdot \beta^2 \cdot \frac{\log\left(max\_torque - \vec{u}\right)}{max\_torque - \vec{u}}$$
$$\frac{\partial^2\left(\text{running cost}\right)}{\partial \vec{u}^2} = \ldots + w\_bounds \cdot \beta^2 \cdot \frac{\left(1 - \log\left(max\_torque - \vec{u}\right)\right)}{\left(max\_torque - \vec{u}\right)^2}$$

We have then simulated the system with $TORQUE\_LIMITS = true$, $PUSH = false$, $mu\_factor = 10$ and $\beta = 1{\cdot}10^{-3}$ and the results are displayed in Figure A.1.



(a) Joints position **(b)** Joints velocity **(c)** First joint torque

Figure A.1: Position, velocity and torque of the joints with $TORQUE\_LIMITS = true$

From Figure A.1 we can notice that the simulated trajectories track very well the reference ones despite the limit on the supplied torque, which can be clearly seen in Figure A.1c, where there are some regions in which the torque exceeds the imposed limit and is suitably truncated.