Optimization Based Robot Control
Assignment 2

Nicola Farina
ID 229296
Dept. of Information Engineering and Computer Science
nicola.farina@studenti.unitn.it

Paolo Furia
ID 239451
Dept. of Industrial Engineering
paolo.furia@studenti.unitn.it

I. FIRST ANSWER

The implementation of the penalty method for under-actuation can be found, properly commented, in the provided template. We implemented it as follows:

$$running\_cost \mathrel{+}= \frac{\texttt{underact}}{2}||u_2||^2 \tag{1}$$

where u_2 is the control input vector for the second joint, the one without a motor. This cost pushes u_2 towards zero, in order to minimize the cost.

We have adjusted the gradient and hessian of the running cost w.r.t. u accordingly, being careful of taking into consideration the two-dimensional nature of the u vector in the code:

- for equation 1 and its gradient, we used a `u_underact` vector where the first element (corresponding to the first, actuated joint) is set to 0, and the second element is set to u_2. This way, there incurs no cost for the actuated joint;
- for the hessian, we multiply `underact` with a two-dimensional diagonal matrix, where the top-left element is set to 0 (corresponding to the first, actuated joint) and the second element is set to 1, so only the second joint is penalized.

The same solution is found with both the selection matrix and the penalty method. To be more precise, the two methods do not behave exactly the same. While the number of iterations until convergence is the same in all cases, there are two very slight but negligible differences:

- the DDP cost and effort between the selection matrix and the penalty method differ, respectively, by a magnitude of 10^{-3} and 10^{-5};
- with the penalty method, the cost and effort differ between the DDP and the simulation respectively, again, by a magnitude of 10^{-3} and 10^{-5}. Average tracking errors range between the orders of 10^{-7} and 10^{-10}.

As expected, those differences approach zero if we increase the `underact` weight. This is because, with the penalty method, we are not introducing a hard constraint on the fact that the second joint is under-actuated. This would be true only when `underact` approaches infinity. Instead, by only penalizing the actuation of the second joint, we are introducing a discrepancy between the DDP model and the simulation. Nonetheless, the `underact` weight is set high enough so that the differences are negligible. Data supporting this is shown in Table I.

Finally, we also computed the difference between the trajectories computed by DDP with the selection matrix and the penalty method, which confirm the fact that both methods produce the same trajectory (with a negligible difference between 10^{-5} and 10^{-8}). This is shown in Table II.

II. SECOND ANSWER

The tracking performance, as expected, decreases when simulating external pushes. Figure 1 shows the deviation from the reference trajectory, Table III shows the differences in cost and effort between the DDP and the simulation results, and Table IV shows the average tracking errors for joints position, velocity and torque.

The pushes are simulated as an instantaneous variation of the second joint's velocity, which is confirmed by the velocity plot at times 0.25, 0.5 and 1 seconds (the last push actually happens at 3 seconds, thus it is not shown).

As can be seen in the plot, the simulated trajectory tends to return equal to the reference trajectory due to the fact that we apply the optimal control input with an optimal local linear feedback $K \triangleq -Q_{uu}^{-1}Q_{ux}$ which acts as stabilizer. The feedback gain is calculated in the backward pass and then is used (with the feed-forward $\bar{\omega}$) in the forward-pass to compute a new trajectory by integrating the dynamics :

$$\begin{aligned} u_i &= \bar{u}_i + \alpha\bar{\omega}_i + K_i(x_i - \bar{x}_i) \\ x_{i+1} &= f(x_i, u_i) \end{aligned} \tag{2}$$

after simulating the system the new state-control pair is kept if the system converges, otherwise the backward pass is performed again and the feedback gain is recalculated. So, as just stated, the DDP calculates also the feedback gains which results in a more reliable control, keeping the states close to their intended trajectories in the event of external perturbation or noise. Without those gains, the system would steer off-trajectory and become unstable.

The torque plot of joint 1 clearly shows this behavior, which present sharp variations of torque inputs and overshooting, in particular at the times around when the pushes are applied. This can remind of the behavior of a proportional controller, for example. A penalty on the amount of torque applied, for example, in order to keep it within a certain limit, can avoid jagged behavior such as what happens around time 1 second, where there is a clear overshoot in the amount of torque applied in order to get back to the reference trajectory.

III. THIRD ANSWER

Setting `ddp_params['mu_factor']` to 0, the simulation achieved the results shown in Figure 2.

The regularization of the local control-cost Hessian Q_{uu} pushes the matrix towards being positive-definite, thus invertible. Another way of seeing it is adding a quadratic cost around the current control sequence:

$$\bar{l}(\bar{x}+z,\bar{u}+\omega) = l(\bar{x}+z,\bar{u}+\omega) + \frac{1}{2}\mu||\omega||^2$$

(although not entirely equivalent, since this regularization only happens for Q_{uu} and not Q or Q_u). The intuition behind this point of view is that we penalize large variations of the control input u. The benefit of this is that, by avoiding large variations, we keep the control input close to the local linear approximation (moving too far away may invalid the approximation). The main downside is that the convergence is slower.

A value of 10 for μ is very high; typical values range between 10^{-9} and 10^{-2} and are usually adapted at each iteration by the following criterion:

- if Q_{uu} is not invertible or the cost improvement is too small, increase μ;
- otherwise, decrease μ.

A constant value of 10 practically ensures that Q_{uu} is invertible, but results in slower convergence. This is showed by the fact that the algorithm takes 37 iterations to converge, instead of 33.

Additionally, the solution found by DDP is slightly different from the one found with an adaptive μ: the trajectories appear as slightly shifted in time after the 0.5 seconds mark, as shown in Figure 3. We confirmed this behavior by trying with $\mu = 100$, where the shift is even more pronounced. This may happen because changing the regularization directly changes Q_{uu}, which is used in the computations of the gains and the value functions and thus affects the final result. The various trajectories appear as shifted because they are trying to achieve the same result in an optimal manner, therefore they must have a similar behavior; but at the beginning, there probably is enough "freedom" in the system to allow slightly different control inputs to achieve the same results in the end.

The most important thing to notice, though, is that the simulation breaks around the 1.3 seconds mark. In particular, the system becomes unstable: the control inputs take on huge (negative) values, and `nan` appear in the computations.

In order to understand why this happens, we started by focusing on the feedback-gains K, since the reference trajectories are very similar, as explained previously, and the only other output of the DDP solver is K itself. Being that $K \triangleq -\bar{Q}_{uu}^{-1}Q_{ux}$, it is safe to say that having a different μ (and thus different $\bar{Q}_{uu}$) yields a different K. In particular, we analyzed the data and figured out that, with `mu_factor` $= 0$, the feedback gains are lower in absolute value, as shown in Figure 4. Lower feedback gains result in a slower response to state errors. Our assumption is that, with a highly unstable system such as an under-actuated double pendulum, this slower response (even if the gains are only slightly lower) results in a higher accumulation of errors and instability, which leads to numerical problems.

The slower response to errors of the system can be seen even before the huge spike, by the fact that the simulated trajectory appears to be slower at going back to the reference trajectory after the pushes, indicating lower feedback gains.

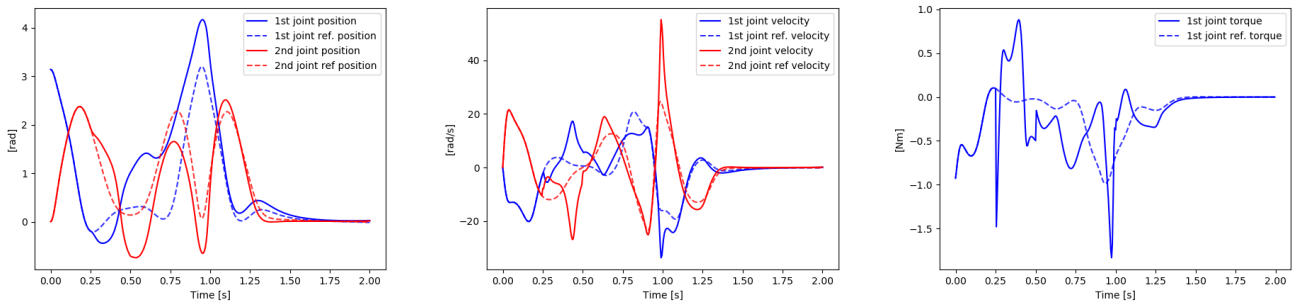
IV. PLOTS AND TABLES

J1 POS (rad)	J2 POS (rad)	J1 VEL (rad/s)	J2 VEL (rad/s)	J1 TORQUE (Nm)
$1.55 * 10^{-7}$	$1.89 * 10^{-7}$	$3.61 * 10^{-5}$	$8.58 * 10^{-5}$	$1.55 * 10^{-8}$

TABLE I: Differences between the trajectories computed by DDP using the selection matrix and the penalty method (computed through mean squared error). J1 and J2 stand for joint 1 and 2 respectively

	SELECTION	PENALTY (1e6)	PENALTY (1e12)
# ITERATIONS	33	33	33
DDP COST	1481.17893	1481.17025	1481.17893
DDP EFFORT	6.53385	6.53384	6.53385
SIM COST	1481.17893	1481.17794	1481.17893
SIM EFFORT	6.53385	6.53388	6.53385
ERR J1 POS (rad)	0	$2.29 * 10^{-9}$	$2.29 * 10^{-21}$
ERR J2 POS (rad)	0	$1.53 * 10^{-9}$	$1.53 * 10^{-21}$
ERR J1 VEL (rad/s)	0	$1.45 * 10^{-7}$	$1.45 * 10^{-19}$
ERR J2 VEL (rad/s)	0	$1.75 * 10^{-7}$	$1.75 * 10^{-19}$
ERR J1 TORQUE (Nm)	0	$6.46 * 10^{-10}$	$6.46 * 10^{-22}$

TABLE II: Differences between selection and penalty methods. The error is a mean squared error between reference and simulated trajectories, and J1 and J2 stand for joint 1 and 2 respectively



(a) Reference and simulated joint position (b) Reference and simulated joint velocity (c) Reference and simulated joint torque

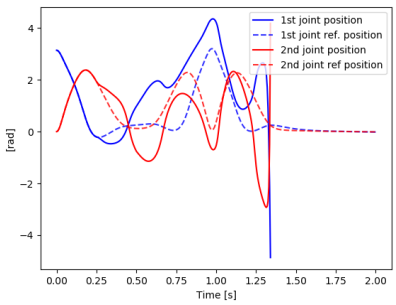
Fig. 1: Results of the Pendubot affected by pushes

	DDP	SIMULATION
COST	1481	2168
EFFORT	6.53	8.50

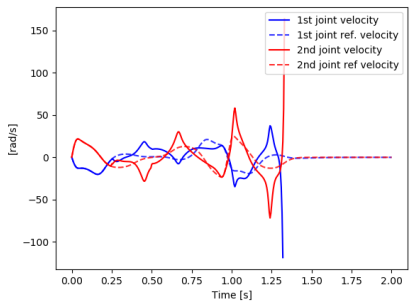
TABLE III: Cost and effort for DDP and simulation, when simulating external pushes

J1 POS (rad)	J2 POS (rad)	J1 VEL (rad/s)	J2 VEL (rad/s)	J1 TORQUE (Nm)
0.372671	0.150520	16.50304	29.24036	0.117177

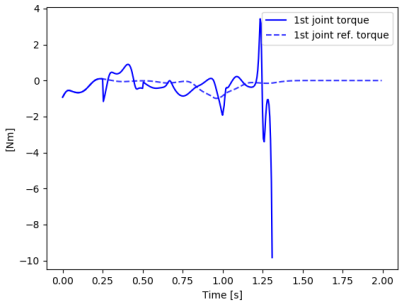
TABLE IV: Average tracking errors when simulating external pushes



(a) Reference and simulated joint position

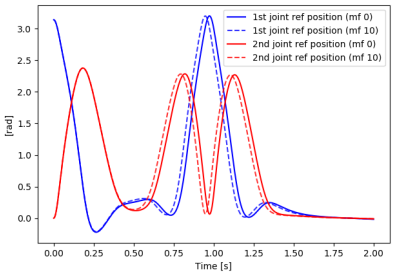


(b) Reference and simulated joint velocity

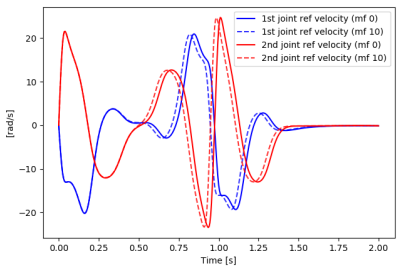


(c) Reference and simulated joint torque

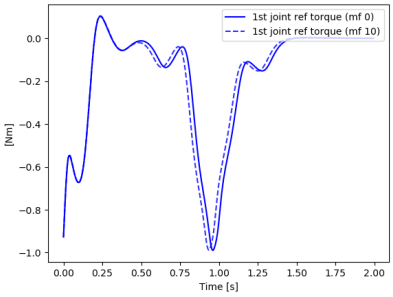
Fig. 2: Results of the Pendubot affected by pushes with regularization factor μ constant at 10 (the simulated trajectories are clipped before they diverge)



(a) Position

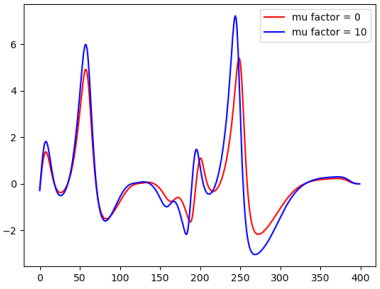


(b) Velocity

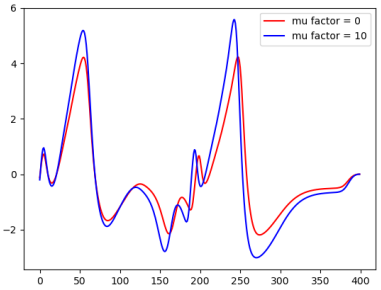


(c) Torque

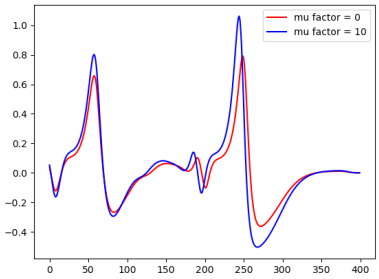
Fig. 3: DDP reference trajectories with `mu_factor` set to 0 and 10



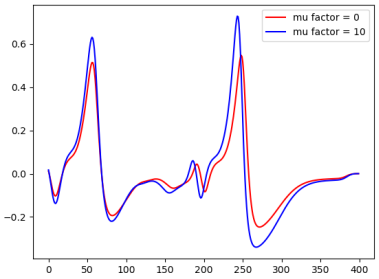
(a) Position gain factors with variable and constant $\mu = 10$ for joint 1



(b) Position gains of joint 2



(c) Velocity gains of joint 1



(d) Velocity gains of joint 2

Fig. 4: Gain factors with variable and constant $\mu = 10$