OPTIMIZATION BASED ROBOT CONTROL
SECOND ASSIGNMENT

Alessandro Grassi 221224 Samuel Bortolin 221245

Note:

- We changed the following line: $q, v, dv, f = simu.simulate(tau, dt\_sim)$ into: $q, v, f = simu.simulate(tau, dt\_sim)$ to avoid an exception since the method $simulate$ returns only these 3 values (dv is missing).
- We computed the cost with the additional penalty to under actuate the second joint as: $underact * \left|\left|u_i\right|\right|^2$ instead of: $underact * \left|\left|u_i\right|\right|$. That is because applying only the norm (absolute value) makes the cost function not derivable for all points. More than that, the squared norm makes the DDP algorithm converge, with just the norm the algorithm wasn't able to find any solution.

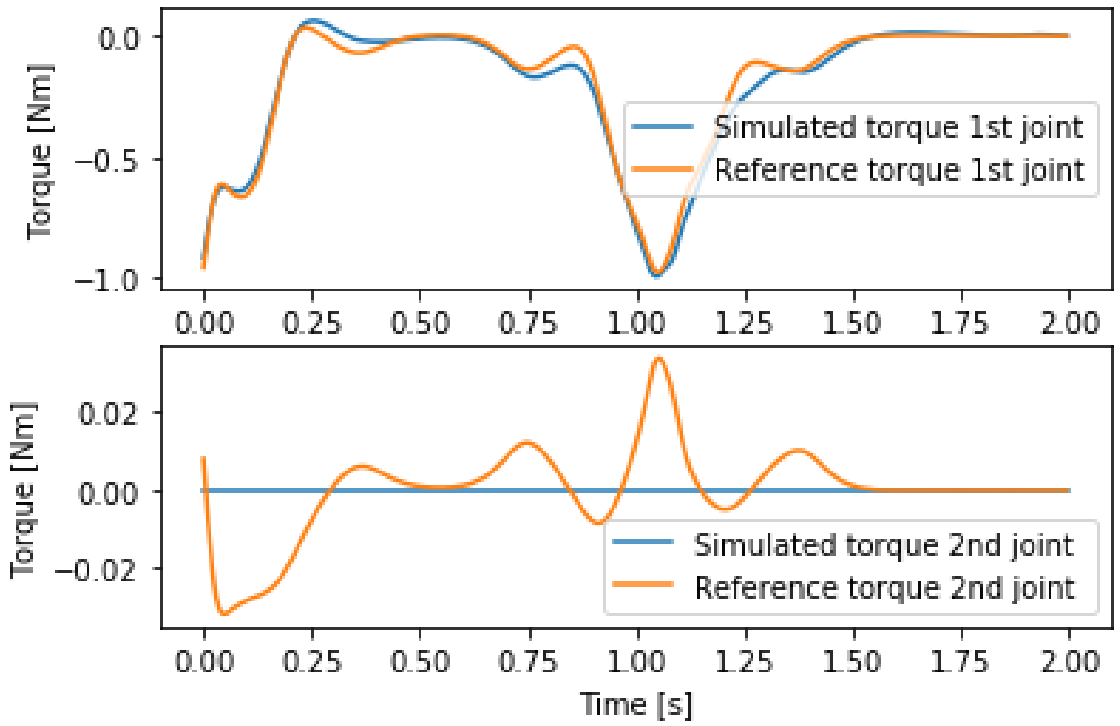
Answers:

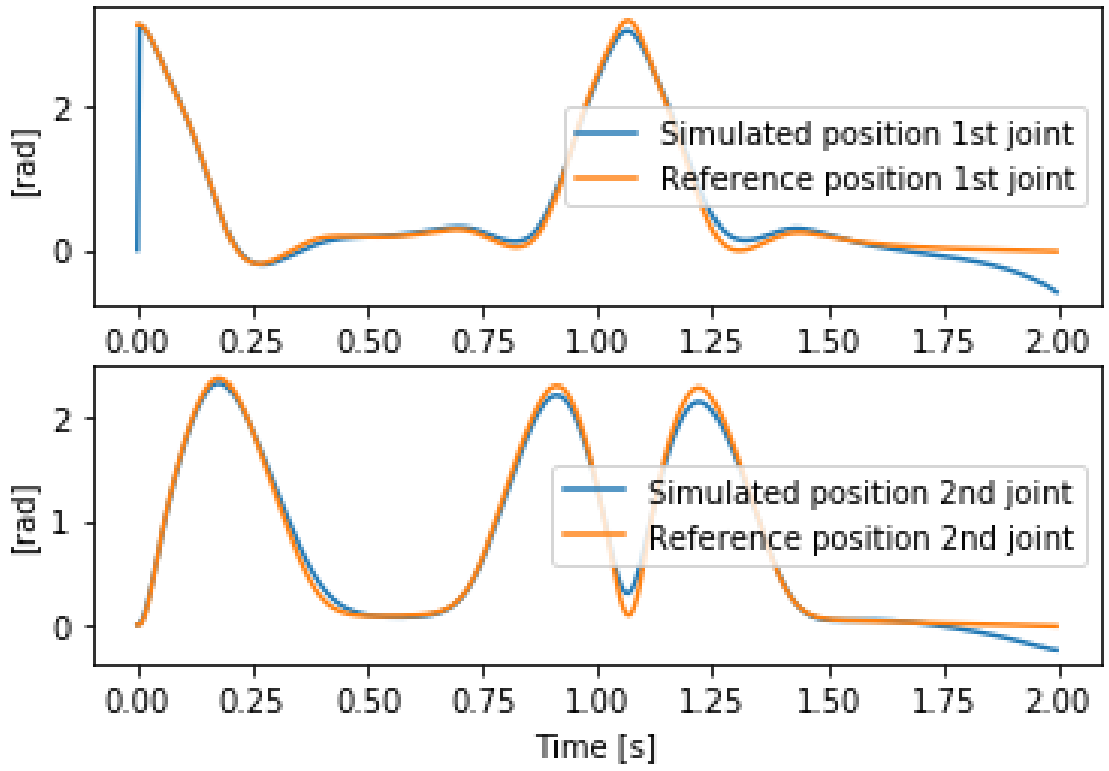
1. The DDP algorithm is able to compute a reference trajectory capable of reaching the final desired state, by looking at the plots regarding the reference position it can be seen that both joints are capable of keeping the final state for a small period of time.
   When simulating the system the pendubot is actually capable of performing the swing up maneuver, it must be noticed though that after reaching the goal state the arm starts to fall on the other side, so it is actually overshooting it.
   This discrepancy between simulated and reference positions could be due to the fact that during the reference control trajectory the robot would apply little torques to the second joint, probably to do little corrections, while during the simulation the second joint is kept completely off.

```
After 23 Iterations
Cost improved from 741.679 to 741.679. Exp. impr -0.001. Rel. impr. 101.7%
Line search succeeded with alpha 1
Decreasing mu to  1.0000000000000002e-28
Algorithm converged. Expected improvement -0.000511729616704008

METHOD = ACTUATION_PENALTY
Execution time: 122.58211660385132

********************************** RESULTS ***********************************
Cost   741.6785042086074
Effort 6.532466790128228
Cost Sim. 737.5322691669988
Effort Sim. 6.81127637163218
```


Torque [Nm]
0.0
−0.5
−1.0
Simulated torque 1st joint
Reference torque 1st joint
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Torque [Nm]
0.02
0.00
−0.02
Simulated torque 2nd joint
Reference torque 2nd joint
Time [s]

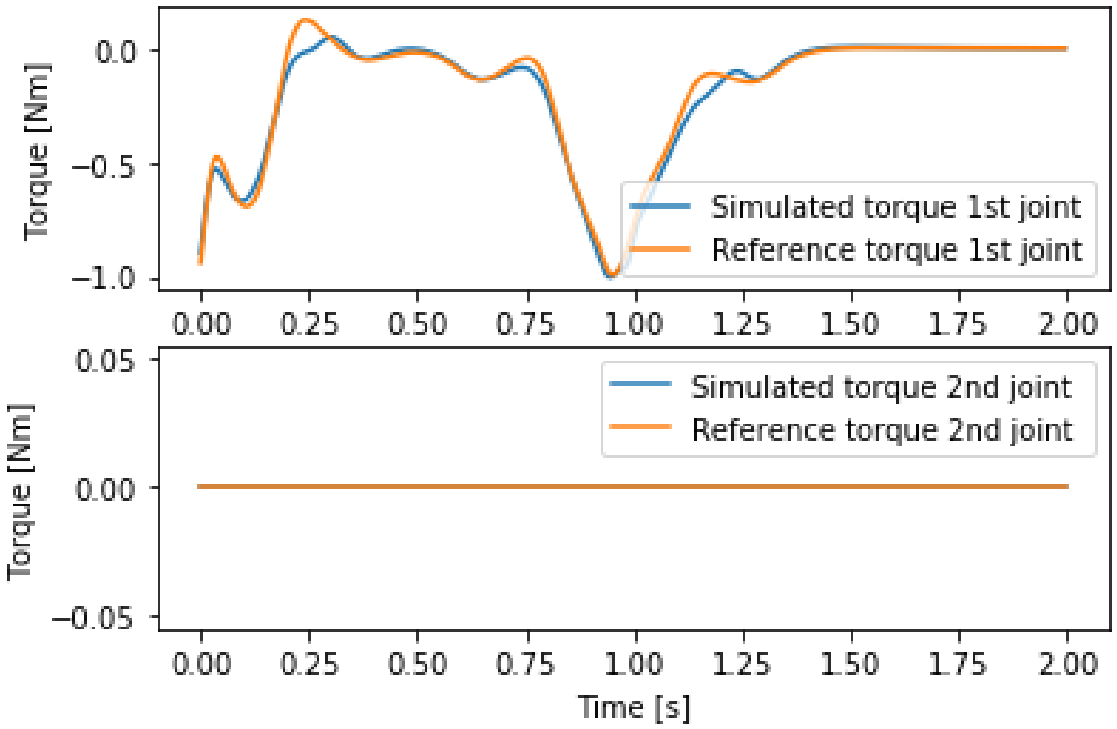

[rad]
2
0
Simulated position 1st joint
Reference position 1st joint
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
[rad]
2
1
0
Simulated position 2nd joint
Reference position 2nd joint
Time [s]

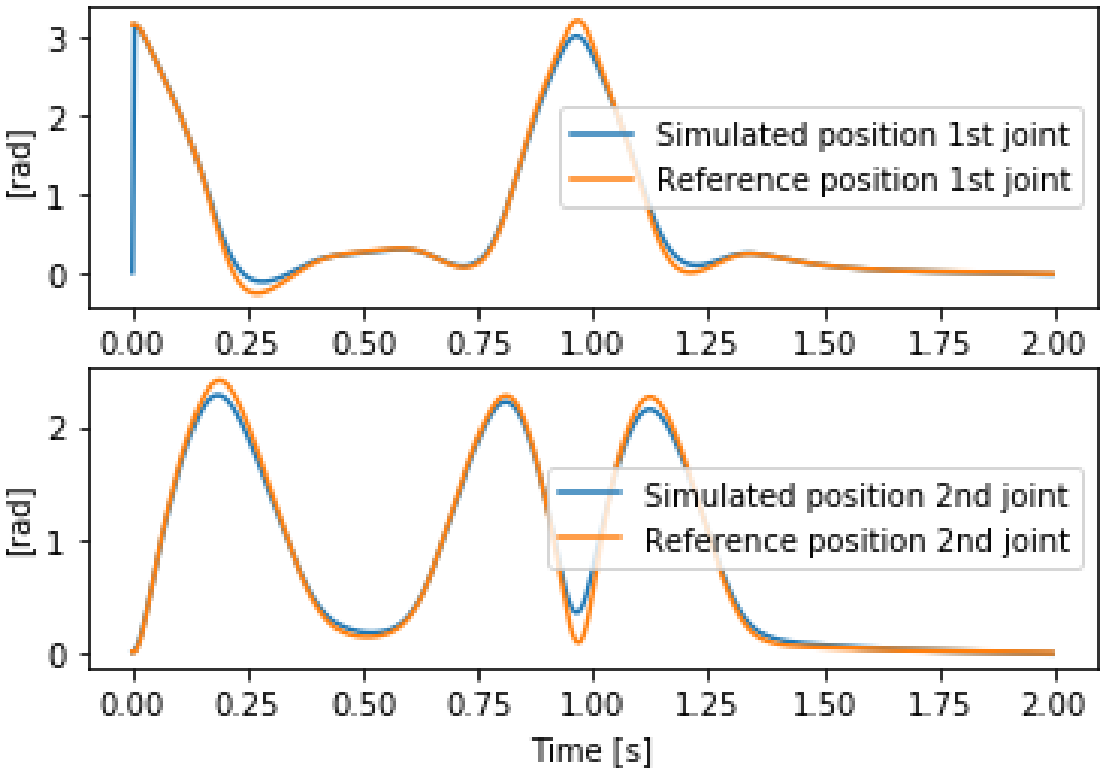
2. The DDP algorithm is able to compute a reference trajectory capable of reaching the final desired state, by looking at the plots regarding the reference position it can be seen that both joints are capable of keeping the final state for a period of time. Probably, according to the reference trajectory, the pendulum could keep the goal state for longer than the simulation time.
   When simulating the system the pendubot is actually capable of performing the swing up maneuver. Both reference and simulation reach the goal state gradually, this will probably avoid overshooting it as we observed adding the additional cost on the underactuated joint.

```
After 32 Iterations
Cost improved from 749.325 to 749.324. Exp. impr -0.000. Rel. impr. 105.7%
Line search succeeded with alpha 1
Decreasing mu to  1.0000000000000005e-37
Algorithm converged. Expected improvement -0.00037183651420207714

METHOD = SELECTION MATRIX
Execution time: 170.02644658088684

************************************ RESULTS ************************************
Cost   749.3243744213954
Effort 6.562519458531781
Cost Sim. 736.5213889914673
Effort Sim. 6.802281184921877
```





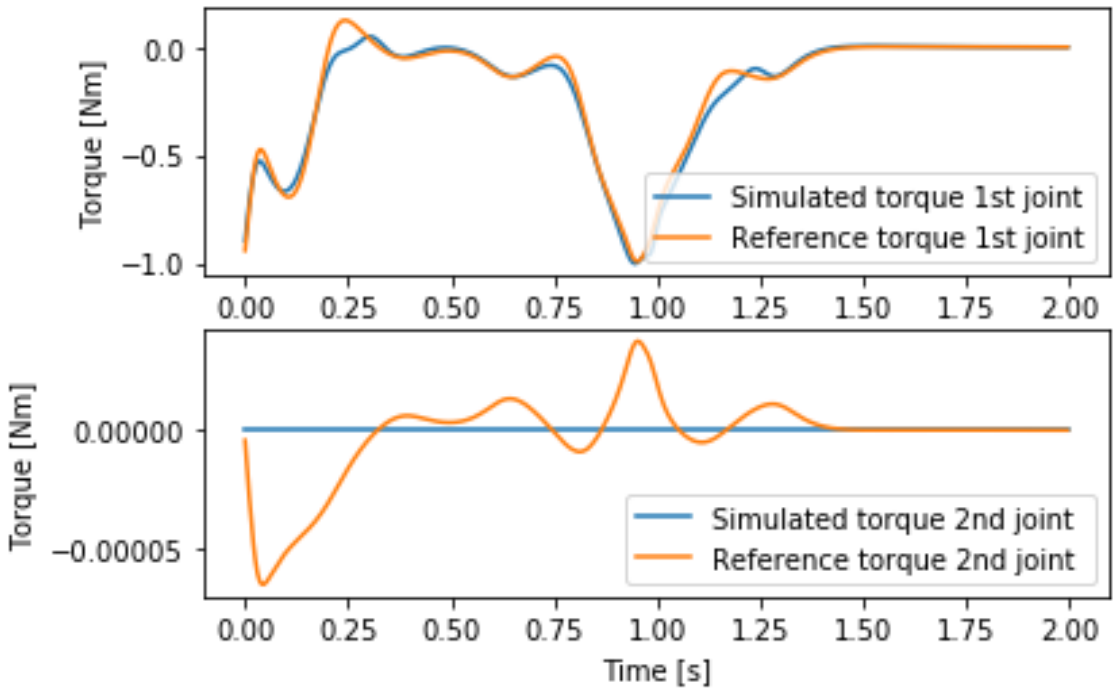
3. Using the selection matrix the discrepancies between reference and simulation positions decrease slightly with respect to the additional penalty method and the robot reaches the goal state in a smoother way avoiding overshooting it during the simulation using selection matrix. This is probably due to the fact that the penalty method allows some little torques on the second joint during reference, these little torques are probably used to correct the trajectory, since these torques aren't available during simulation the pendubot overshoots the target.
   The simulation cost is better with the selection matrix but the difference is irrelevant. When adding a penalty on the torques on the second joint the simulation reached convergence with just 23 iterations with a final reference cost of 741.6785042086074, the selection matrix approach converged in 32 iterations with a reference cost of 749.3243744213954, so it took 9 more iterations (almost 40% more time) to reach a reference cost a bit worse.
   For the simulation cost for the additional penalty method it was 737.5322691669988 and for the selection matrix approach it was 736.5213889914673. It must be noted that using the selection matrix the simulation cost is better but the increase in performance is irrelevant.

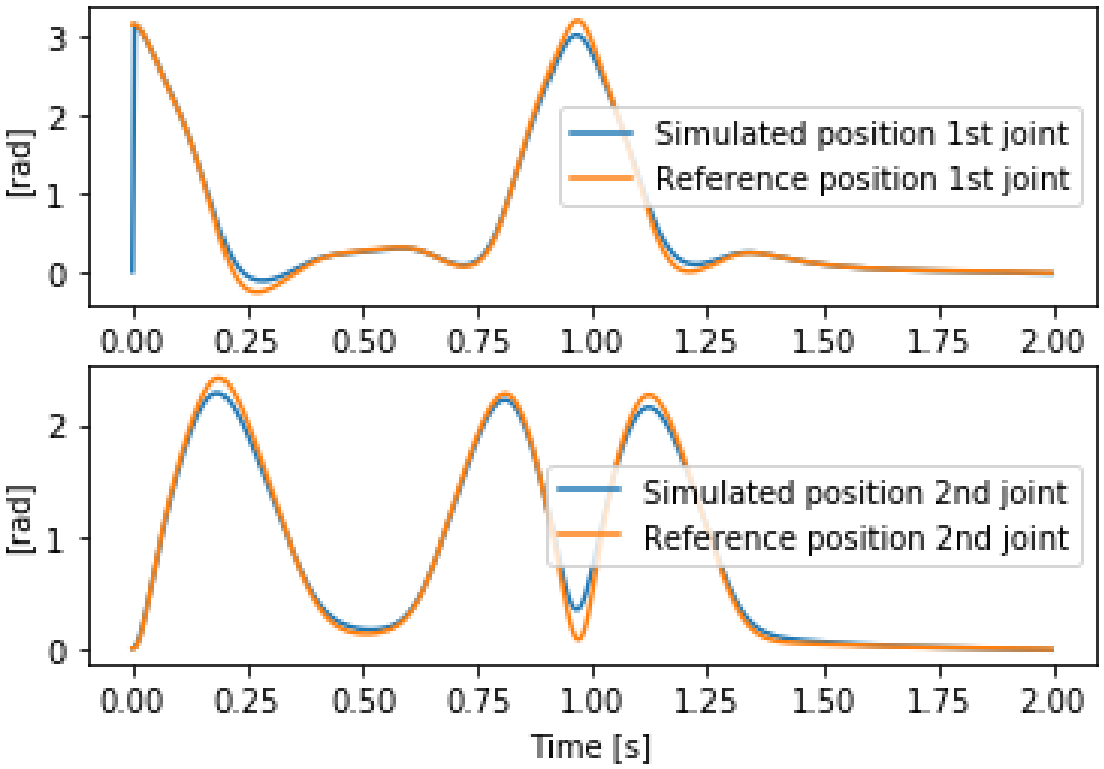
4. Increasing underact to $1e5$, the discrepancies observed before when underact was $1e2$ disappear because with a higher cost the algorithm will produce references using almost zero torque on the underactuated motor. This is becoming more similar to the selection matrix approach where it is removed and so the tracking of the reference position trajectory is quite better and almost equal to the one using the selection matrix. As in the case of the selection matrix the number of iterations increased to 32 and the costs increases to 749.3116175734085 but it decreases in simulation to 736.5121963880538, a bit less than the selection matrix. Probably with an infinite cost they will be exactly the same, since the torque on the underactuated motor has to be zero in order to avoid having an infinite cost. The overshooting problem observed with $1e2$ as cost seems to be avoided.

```
After 32 Iterations
Cost improved from 749.312 to 749.312. Exp. impr -0.000. Rel. impr. 105.7%
Line search succeeded with alpha 1
Decreasing mu to  1.0000000000000005e-37
Algorithm converged. Expected improvement -0.0003693100223761582

METHOD = ACTUATION_PENALTY
Execution time: 173.21404933929443

************************************ RESULTS ************************************
Cost   749.3116175734085
Effort 6.5624605652118415
Cost Sim. 736.5121963880538
Effort Sim. 6.802276605714071
```





5. In a real underactuated robot, the best choice depends on the context. In general it is shown by the simulation that the selection matrix works better, but it also takes more time to compute the control inputs. In a more complex robot the computation time could be even higher. On the other side if the discrepancy between reference and tracking trajectories should be as low as possible, maybe the selection matrix is the best solution since the additional penalty method (if with a not so high cost) can generate references applying torques also to joints that are not actuated and this will inevitably cause a tracking error in the real application since it is considering of applying some torques to joints that are missing. This will not happen with the selection matrix, but if we have a larger margin of error and we need fast computations we can consider an additional penalty method with a not so high cost.