Optimization Based Robot Control Assignmet 02

Alessandro Assirellli, mat. 231685

December 11, 2022

1 First answer

Implement the additional penalty method setting underact to 1e6 (SELECTION MATRIX=0, Additional PENALTY=1, PUSH=0). Verify that DDP finds the same solution found with the selection matrix method (SELECTION MATRIX=1, Additional PENALTY=0, PUSH=0).

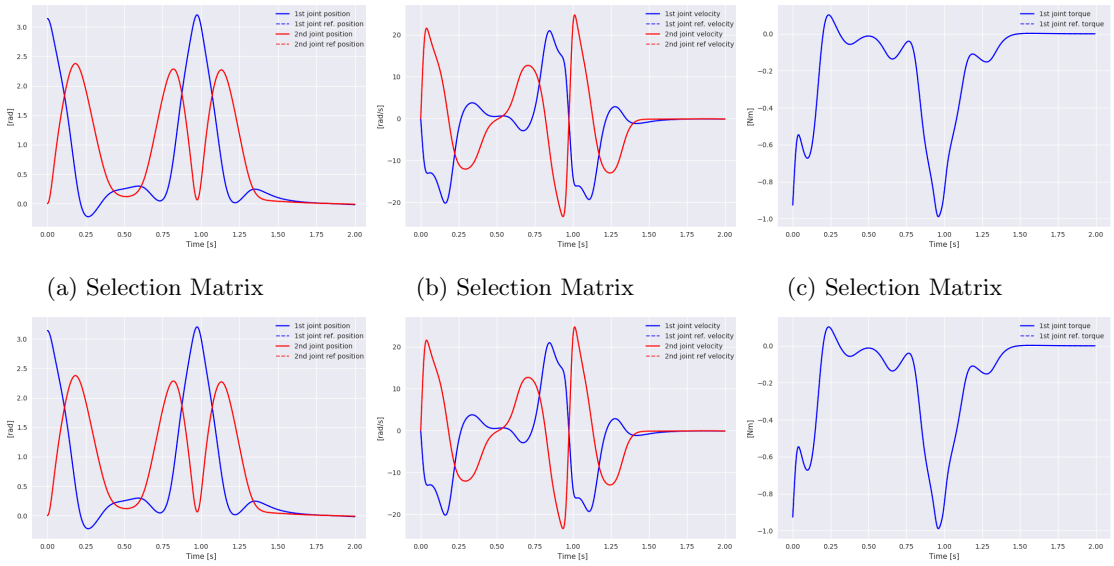
In order to use the additional penalty method, the derivatives of the cost with respect to the state and control of the additional term must be computed. The additional penalty method is defined as:

$$l_{add} = \frac{underact}{2}||u_i||^2 \tag{1}$$

The derivatives with respect to state are thus 0, while the first and second derivative with respect to control, considering the double pendulum case are:

$$l_{add,u} = \begin{bmatrix} \frac{\partial l_{add}}{\partial u_1} \\ \frac{\partial l_{add}}{\partial u_2} \end{bmatrix} = underact \begin{bmatrix} 0 \\ u_i \end{bmatrix} \quad l_{add,uu} = \begin{bmatrix} \frac{\partial^2 l_{add}}{\partial u_1^2} & \frac{\partial^2 l_{add}}{\partial u_1 \partial u_2} \\ \frac{\partial^2 l_{add}}{\partial u_2 \partial u_1} & \frac{\partial^2 l_{add}}{\partial u_2^2} \end{bmatrix} = underact \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

Since a very large penalty on the control has been used, the solver is forced to keep the second joint torque at zero, which is equivalent to the selection matrix approach and their solutions are thus equivalent. The results as presented in Figure1



(a) Selection Matrix (b) Selection Matrix (c) Selection Matrix

(d) Additional Penalty (e) Additional Penalty (f) Additional Penalty

Figure 1: Comparison between Selection Matrix and Additional penalty

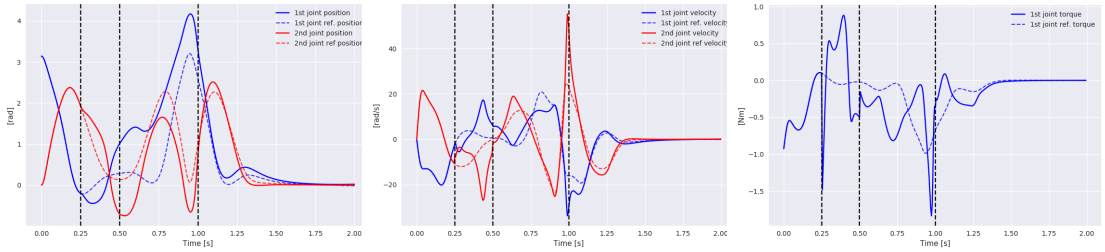
2 Second answer

*Choose the selection matrix method and set PUSH=1. This will simulate four external pushes (instantaneous increase of the second joint velocity) occurring after N/8, N/4, N/2 and 3*N/4 simulation steps. Describe what happens in terms of tracking performance of the reference trajectories.*

When the double pendulum get pushed its second link velocty increase by a certain amount. This imply that both position and velocity of the simulated robot will be different

from the predicted ones. The robot could anyway reach the target. The reason for this is that DDP provide not only a list of control actions, but a control policy in the form of optimal feedforward torques and optimal feedback gains K. This means that even if the state deviates from the predicted one, the feedback gains can drive the state back to the desired one. Indeed the more the actual state of the robot is different from the predicted $\bar{x}$, the stronger the control component $K(\bar{x} - x)$.

Figure 2 shows the simulation and the dashed lines indicate when the pushes occur. As can be seen, when the push is applied the state abruptly changes. However both position and velocity converge to the desired one by the end of the simulation.



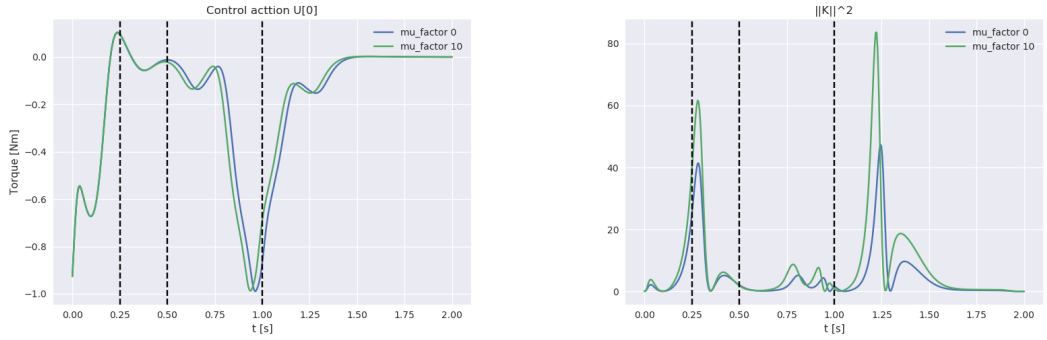
(a) Positions (b) Velocities (c) Torques

Figure 2: Simulation with PUSH=1

3 Third answer

Now set ddp params['mu factor'] to 0, which means keeping the regularization term μ constant and equal to the initial value 10. What differences do you notice compared to the previous test? Can you explain them?

In Figure 3 The DDP solutions of the two problems are shown. In order to visualize a comparison between the Riccati Gains, Figure (b) shows the 2-norm squared of the two matrices. The two results are different and the reason behind it, is that when mu factor is



(a) Control action (b) Riccati gains

Figure 3: Comparison of the two solutions

set to 0 the regularization mu is kept costant and equal to 10, which is a big value. *mu* is a regularization term which is needed when inverting Q_{uu}, which appear in the control action w and in the optimal feedback gains K.

$$\bar{Q}_{uu} = (Q_{uu}^{-1} + \mu I) \quad w = -\bar{Q}_{uu}^{-1} Q_u \quad K = -\bar{Q}_{uu}^{-1} Q_{ux} \tag{3}$$

If mu is too large the policy is no more optimal for the real system because it refers to a too much shifted local approximation.

During the simulation, by setting $PUSH = 1$ three external pushes are considered and the simulation breaks. When introducing the pushes a very large state error z_{err} is produced and so a too large control action $w + Kz_{err}$ is generated, which produces NaN in the state vector and the renderer breaks.