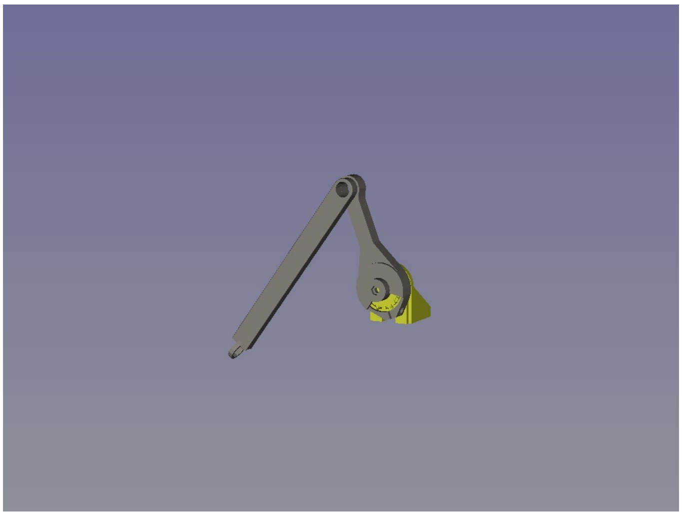
# UNIVERSITY OF TRENTO

Industrial Engineering Department

Master of Mechatronics Engineering

# Assignment 02: Comparison between different methods for taking account of underactuation in DDP




| Professor: | Student: | Matr: N. |
| --- | --- | --- |
| Prof. Andrea Del Prete | Andrea Cardone | 224836 |


Academic Year 2021-2022

**1.**
The penalty method is implemented adding in the cost the penalization for the control of the second motor. The overall running cost results to be:

$$l_t(x_t, u_t) = \sum_{t=0}^{N} \frac{1}{2}\mathbf{x_t}^{\mathsf{T}}\mathbf{H}_{xx,t}\mathbf{x_t} \; + \; \mathbf{h}_{x,t}\mathbf{x_t} \; + \; \mathbf{h}_{s,t} \; +$$
$$+ \sum_{t=0}^{N-1} \frac{1}{2}\lambda||\mathbf{u_t}|| \; + \; \frac{1}{2}\texttt{underact}\mathbf{u_{1,t}}^2$$

where in the second part is added a further cost for the second component of the control $\mathbf{u} = [u_0, u_1]^{\mathsf{T}}$ which is related to the control of the second motor. The parameter `underact` is the related weight.
The gradient w.r.t. $\mathbf{u}$ of the new running cost is:

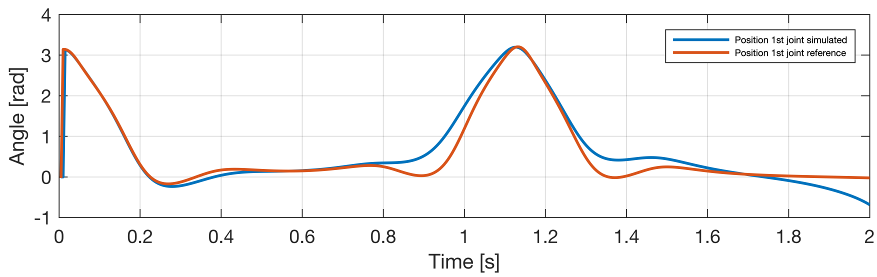
$$\nabla_u l_t = \lambda \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \; + \; \texttt{underact} \begin{bmatrix} 0 \\ u_1 \end{bmatrix}$$

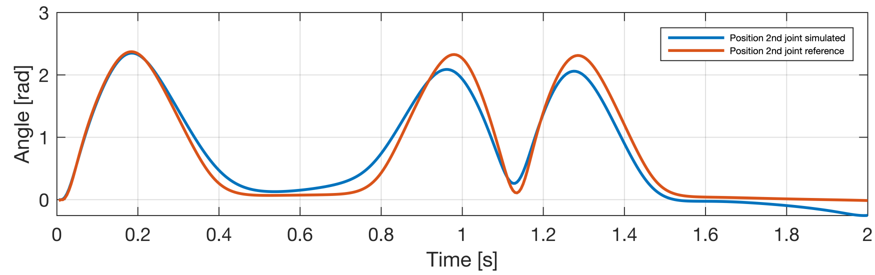
The hessian of the running cost w.r.t. u results to be:

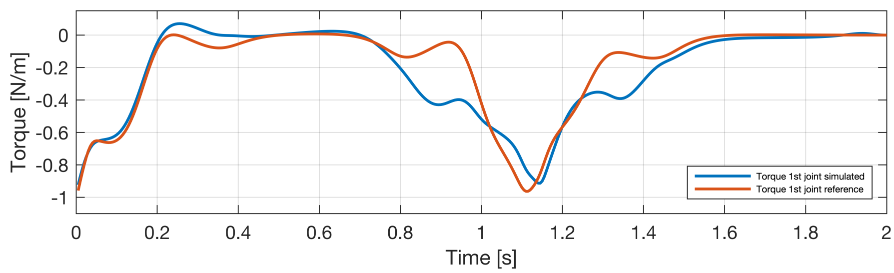
$$\mathbf{H}_{l_t,u} = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \; + \; \texttt{underact} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

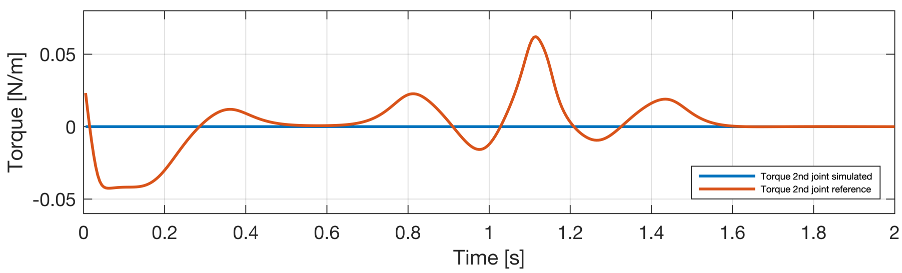
In the following images are reported the plots showing the reference and simulated position and control trajectories. In this case the parameter `underact` is set to 1e2. The DDP algorithm is able to generate a reference trajectory to reach the desidred state but simulating the system the Pendubot is not able to perform a succesful swing-up maneuver. At the end of the optimization the costs turned out to be:

```
cost = 737.6092
cost sim = 763.4777
```






**2.**
The selection matrix method is implemented premultiplynig the control torque vector with a matrix S that force the second torque to be zero.

$$\bar{\mathbf{u}} = \mathbf{S}\mathbf{u}$$

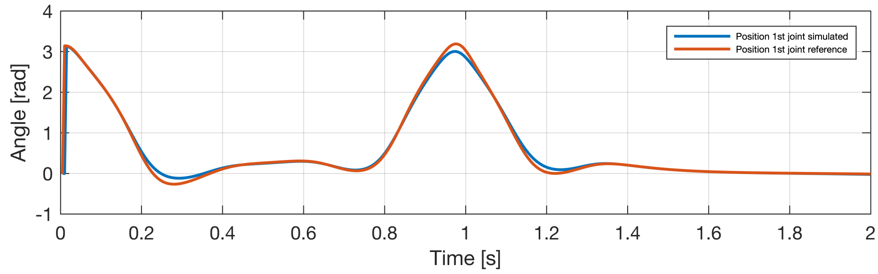
$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

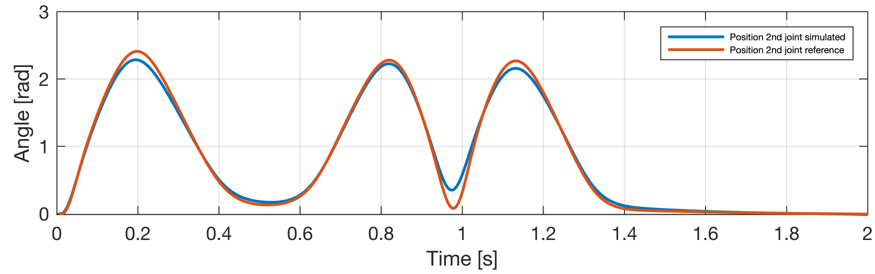
Where $\bar{\mathbf{u}}$ is the new control torque vector and $\mathbf{S}$ the selection matrix.

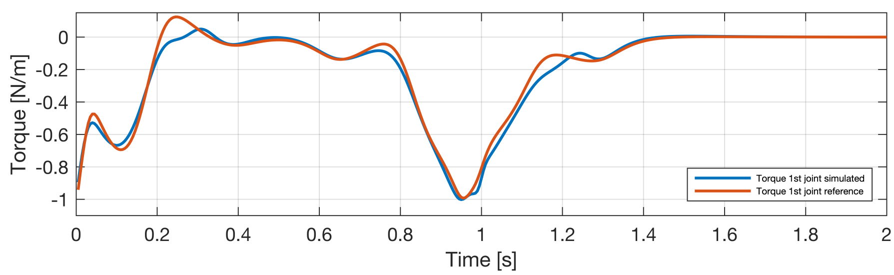
The derivative of the dynamics w.r.t. $\bar{\mathbf{u}}$ results to be:

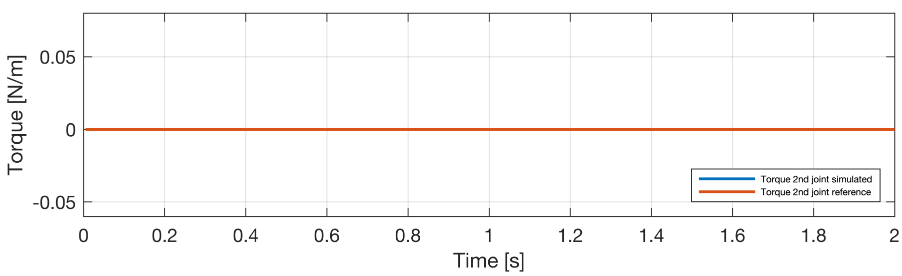
$$f_{\bar{u}} = f_u \mathbf{S}$$

The DDP algorithm is able to compute the reference trajectory to reach the final state and in the simulation the system is able to reach succesfully the desired position performing the swing-up maneuver. In the following images are reported the plots showing the reference and simulated position and control trajectories.
At the end of the optimization the costs turned out to be:
`cost = 749.3244`
`cost sim = 736.5214`


Position 1st joint simulated
Position 1st joint reference
Angle [rad]
Time [s]


Position 2nd joint simulated
Position 2nd joint reference
Angle [rad]
Time [s]

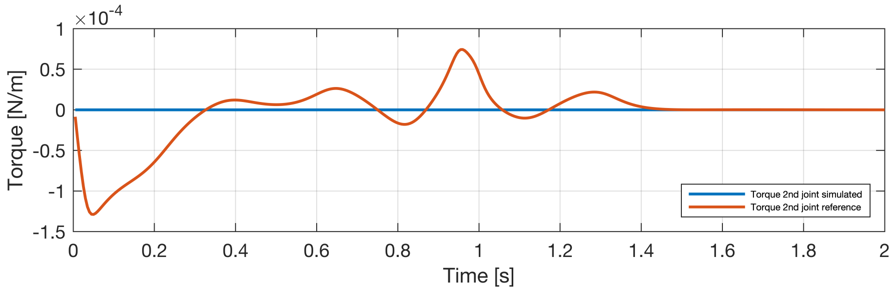

Torque 1st joint simulated
Torque 1st joint reference
Torque [N/m]
Time [s]


Torque 2nd joint simulated
Torque 2nd joint reference
Torque [N/m]
Time [s]

**3.**
In the first case the parameter `underact` is set to `1e2`. The DDP algorithm is able to generate a reference trajectory to reach the desidred state, but the algorithm provides a considerable control for the second motor. On the other hand, in the method with the selection matrix, the component of the control relating to the second motor is equal to zero. At the end of the optimization, the cost in the first method is lower than the second because also the second motor is used to reach the desired state. This is due to the fact that, despite the additional cost, the algorithm still finds convenient to use the second motor.
In the simulation, the first method fails to obtain a succesful swing-up maneuver because the control component of the second motor is zero unlike what is obtained in the reference control. Therefore the simulation and reference trajectories appear quite different and the simulation cost is higher. On the contrary, in the second method, the simulated and reference trajectories are quite similar, and the cost of the simulation is lower. This is due to the fact that the selection matrix intervenes in the dynamics of the system, eliminating the control component relating to the second motor. On the other hand, in the first method the dynamics of the system does not change, in fact the use of the second motor is only discouraged. Moreover in the first case the algorithm takes 19 iterations while in the second 32, this is due to the fact that the use of the second engine, even if discouraged, gives a considerable advantage in achieving the convergence of the algorithm.

**4.**

Increasing `underact` to `1e5` the discrepancies disapperar. This is due to the fact that in this case the use of the second motor to reach the desired state is strongly discouraged, as the underact parameter has been increased by three orders of magnitude. In fact, if we observe the reference trajectory of the second motor control we see that it is not equal to zero for the entire time interval, but it is very small, in particular of the order of magnitude of 1e-4, as we can see in the following image.



**5.**
In a real robot I would implement the selection matrix method. This method is more robust than the additional penalty method. First of all, the selection matrix intervenes directly on the dynamics of the system, therefore during the optimization the second motor is not considered and the underactuated component remains equal to zero. In addition this method does not add further computational cost, besides the multiplication by the matrix S in the dynamics and in the derivative with respect to the control u. On the contrary, the penalty method adds a computational cost in the running cost, in the gradient and in the hessian with respect to u. Another problem with this method is the fact that we don't know the value of the underact parameter a priori to get a succesful maneuver and it has to be set manually. This may mean that the same value of this parameter may be fine for some cases, but in others it may lead to the goal not being achieved. This is due to the fact that in some cases even a small control of the underactuated component can be decisive for the achievement of the objective and therefore determine its failure in the simulation where the underactuated component is forced to be zero. For these reasons the selection matrix method is a better choice in terms of reliability, computational cost and simplicity, because it avoids finding the right size of a particular parameter.