ARC Assignment 01: Simulation and control of a DC Motor with Coulomb and viscous friction

Andrea Carollo [206343] - Matteo Tomasi [206334]
Advanced Optimization-based Robot Control
University of Trento

April 28, 2020

Question 1

Which friction compensation strategy resulted in the smallest tracking error? Why? How much was it?

The compensation strategy with the smallest tacking error is the one that uses the *sign()* function. In this case the mean tracking error is around of 7.05. However, the *sign()* has a discontinuity in zero. For this reason, when the simulated coulomb torque changes sign, the control torque changes very quickly.
For large amplitude reference trajectory, the compensation using sign gives the best result because the control is more aggressive. However, if the amplitude is small, the jumps could become the main behaviour and the control could give a larger tracking error.

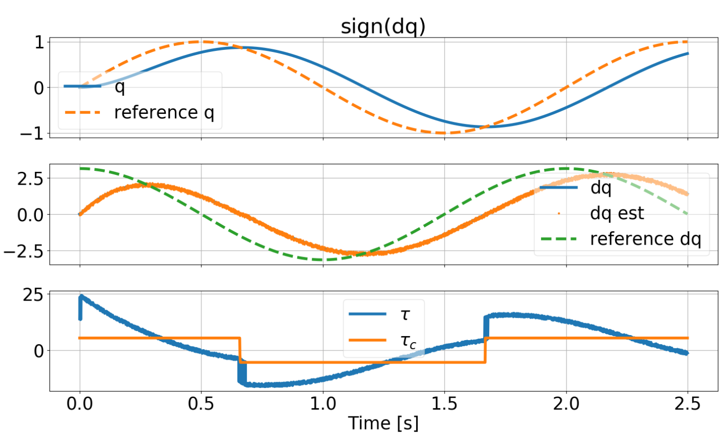

Figure 1: Results using sign compensation strategy

Question 2

Which friction compensation strategy resulted in the largest tracking error? Why? How much was it?

The compensation strategy with the largest tacking error is the one that uses the *tanh()* function, as an approximation of the function *sign()*, with a gain of 0.01. The gain gives an idea of how much the *tanh()* function is close to the *sign()* one.

$$Compensation\,Input\ :\ u_{comp} = tanh(K_{gain}\ \dot{q})\ i_c \tag{1}$$

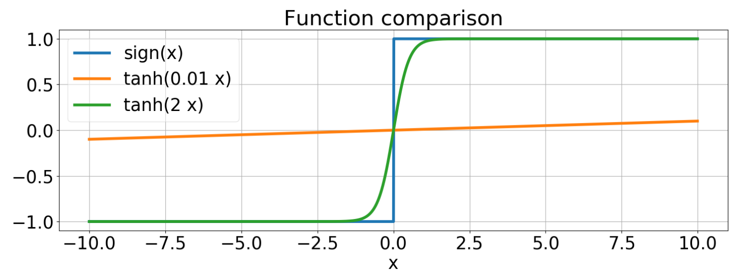

Figure 2: Comparison between the used function

As we can see in the figure 2, the *tanh(0.01 x)* gives a very bad approximation of the *sign* and, for this reason, the mean tracking error is around of 8.29. However, when the coulomb friction changes sign, the changing of the control torque is smoother.
For a good compensation strategy, we should find a trade off between aggressiveness and smoothness.

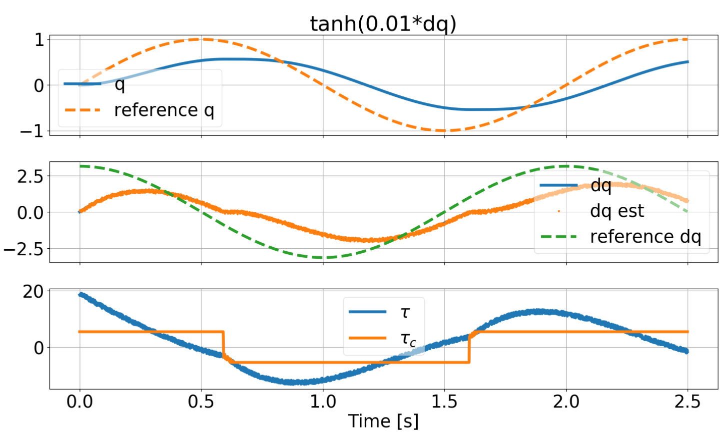

Figure 3: Results using tanh compensation strategy with gain equal to 0.01

Question 3

What is the main issue with using the sign function for friction compensation?

Some jumps are generated in the control torque using the sign function in the compensation strategy. This phenomenon does not reflect what happens in the real world because the friction torque does not jump. To better understand what happens, we can look at the figure 4, where the comparison between the two strategies is shown. We can easily note that the control torque jumps when the coulomb torque changes sign. This is due to the fact that the compensation model has a discontinuity when the velocity is around to zero. This makes the torque jump, coupled with the fact that there is some noise in the measure that can change the velocity sign. While, the hyperbolic tangent technique gives a more realistic behaviour, having a smoother changing when the velocity changes sign.

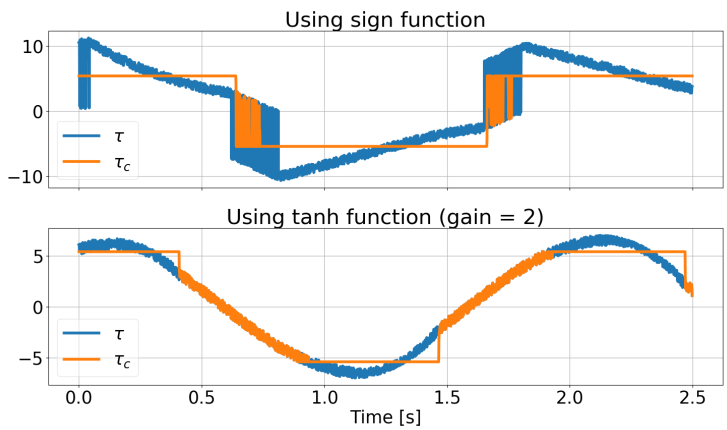

Figure 4: Comparison between sign and tanh compensation method

The jumps are more visible when the torque value is low. To extract the figure 4 we used a lower amplitude reference trajectory.

Question 4

Overall, which one of the three strategies would you implement on a real actuator? Why?

With *sign()* method we get a mean error about 7.05, instead with *tanh()* method, using *tanh_gain = 2* , we get an error of approximately 7.20. As we have seen, using *tanh()* function we will increase the mean error, with respect to the *sign()* method, but the continuity around zero of the compensation model is more important. This because the *sign()* function generates jumps on the torque, while the *tanh()* creates a smother control.
In conclusion, we would use the *tanh()* function for the friction compensation. Moreover, we would use an higher gain, i.e. *tanh_gain = 2* , to approximate better the aggressiveness of the *sign()* function, while keeping the smoother behaviour.

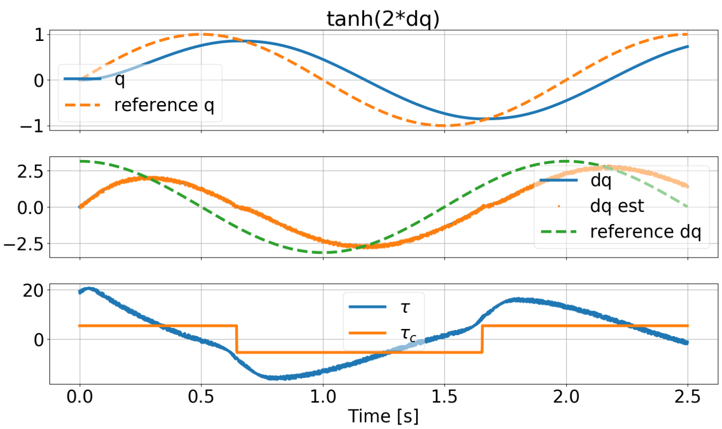

Figure 5: Results using tanh compensation strategy with gain equal to 2

Question 5

Discuss ways for improving the tracking performance (e.g., increasing gains), highlighting pros and cons.

The possibility to increase the performances are:

1. **Increase gain *tanh_gain***
   Increasing the gain of the *tanh()* function, we will reduce the mean tracking error, but the drawback is on the estimated torque: when the estimated velocity passes through zero the torque value could start jumping. The phenomenon is more present with small amplitude of oscillation - figure 7 -. This happens because the *tanh(tan_gain*dq)* function tends to *sign(dq)* function, increasing the gain value.

On the following plots there are the samples of a control with *tanh_gain = 10*, and they show that the torque is approximately similar to the one with the sign-modelled friction on figure 1.

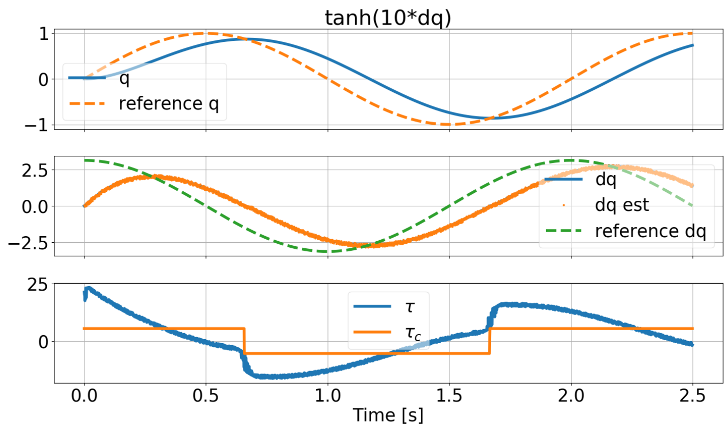

Figure 6: Results using tanh compensation strategy with gain equal to 10

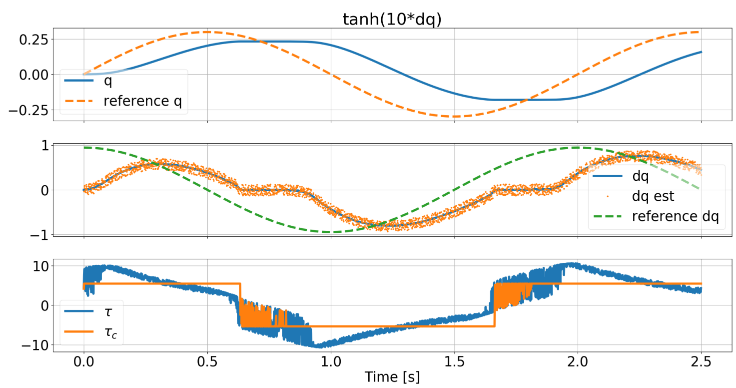

Figure 7: Results with tanh gain equal to 10 and small oscillations

2. **Increase the integral gain in the PID**:
   With a proper tuning of the PID coefficients - Kp, Kd, Ki - we can further reduce the mean error. Simply increasing all the coefficients, the mean error will decrease in a relevant way: using as coefficients $Kp = 2$, $Kd = 2$, $Ki = 0.1$ - instead of $Kp = 1$, $Kd = 1$, $Ki = 0$ - and keeping $tanh\_gain = 2$, we get a mean error around 4.05. Nevertheless, increasing too much the coefficients, we could introduce some undesired behaviours, like overshot or high peak of the control torque.

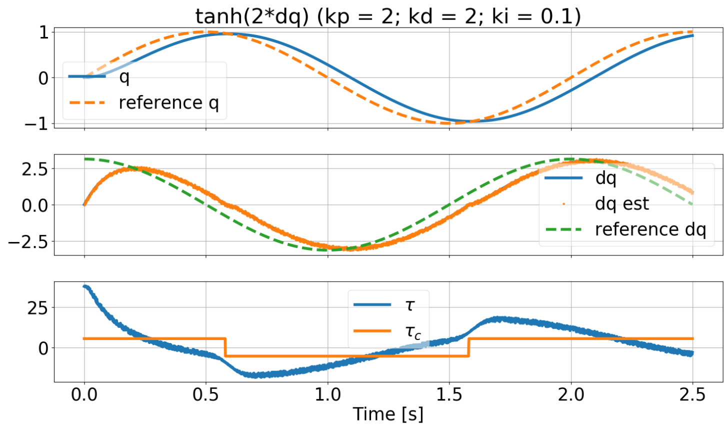

Figure 8: Results with tanh gain equal to 2 and custom PID coefficients