AMORO Lab - Report

Ambra Ierardi and Alessandro Trovatello

18/10/2024

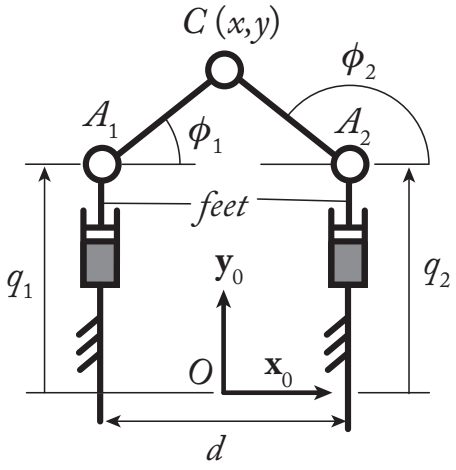


Figure 1: Kinematic model of the Biglide. The prismatic joints are actuated.

1 Introduction

The main goal of this laboratory is to compute the geometric, kinematic, of first and second order, and dynamic models of a **Biglide** mechanism and to analyze the results on GAZEBO. These models are used to estimate the position, represented by the **C** variable in the figure, and its first and second derivatives, of the *end-effector* of the robot, knowing the values of the variables, and their derivatives, of the active joints, *i.e.* the two vertical prismatic joints, or the other way around; as last, it is also possible to determine these values for the passive joints, *i.e.* the two revolute joints. The dynamic model is the algorithm that allows the user to compute the inertial matrix **M** and the Coriolis term **C**, which allow to compute the torque necessary to move the joints of the robot in order to reach the desired final configuration.
In the second part, a controller has been designed to track an arbitrary trajectory, exploiting the error between the desired value of the joints positions and their derivatives and the actual ones.

2 Comparison of models

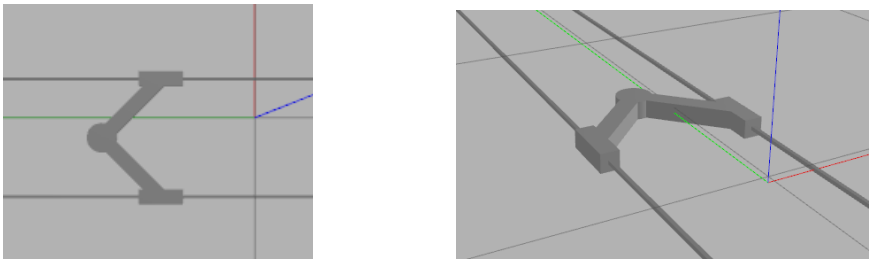


Figure 2: Gazebo model of the Biglide.

The step before the comparison of models was to compute the geometric and kinematic models of the *Biglide*. Once the models have been defined, the *model_test.py* was created in order to test the forward geometric/kinematic model, computing functions as DGM (Direct Geometric Model), DKM (Direct Kinematic Model) and DK2M (Direct Kinematic Model of 2nd Order) both for active and passive joints. Afterwards, the *inverse_model_test.py* was created in order to test the IGM (Inverse Geometric Model), the IKM (Inverse Kinematic Model) and IK2M (Inverse Kinematic Model of 2nd Order), both for active and passive joints. A way to clearly plot the results is to use the the *Matplotlib* library. Furthermore, the *model_eval.py* was used in order to check if all the errors between the simulation and the models were below a threshold of 10^{-3}, meters or radiants, and their derivatives, depending on the relative quantity examined.

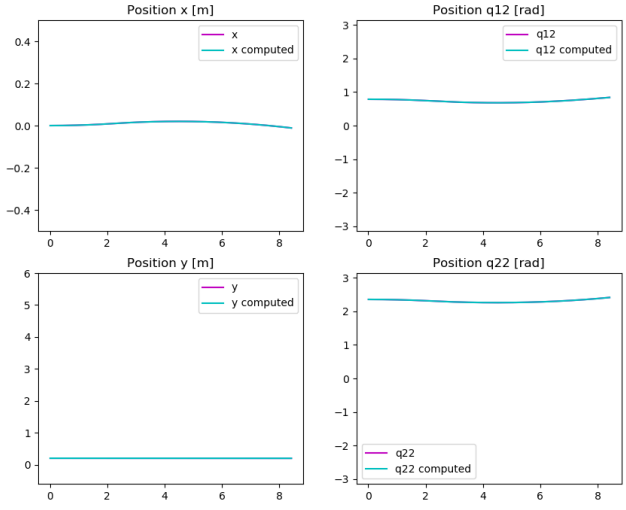


Figure 3: On the left: Comparison between real and computed end-effector position. On the right: Comparison between real and computed angular position of passive joints.

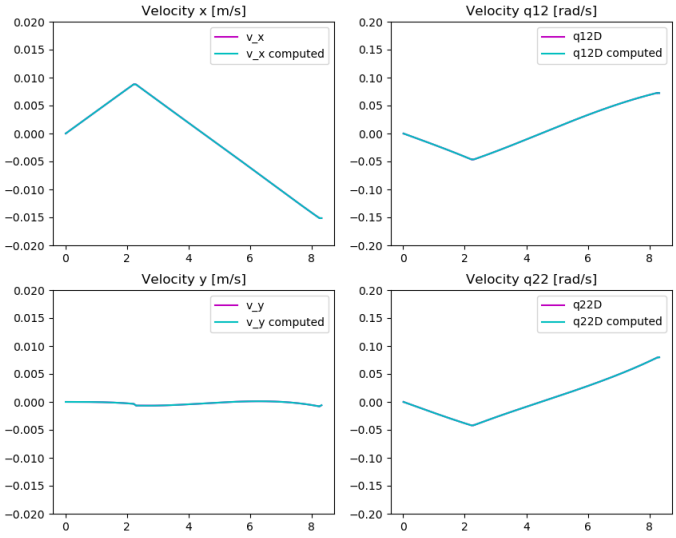


Figure 4: On the left: Comparison between real and computed end-effector velocity. On the right: Comparison between real and computed angular velocity of passive joints.

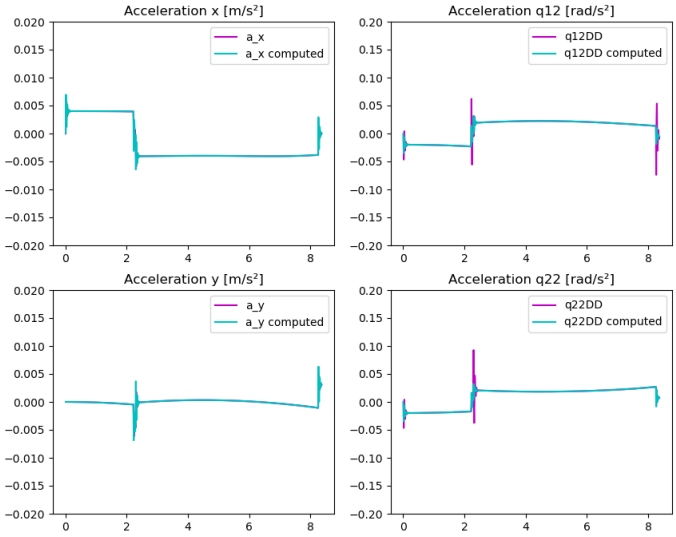


Figure 5: On the left: Comparison between real and computed end-effector acceleration. On the right: Comparison between real and computed angular acceleration of passive joints.

3 Control

The *Computed Torque Control* (CTC) is a feedback control technique based on linearization of the examined system, exploiting the inverse dynamic algorithm, to find a decoupled solution for the individual control of each robot joint.
The trajectory was designed to start from the initial position computed based on the actuated joints' initial asset, up to the desired value, with *(x,y)* equal to *(x+0.06[m],y+5[m])*. Different position and velocity gains, respectively K_p and K_v, were tested. Furthermore, the relative error throughout the whole trajectory was computed as the difference between the desired and the actual position of each joint, and then displayed in the plots below.

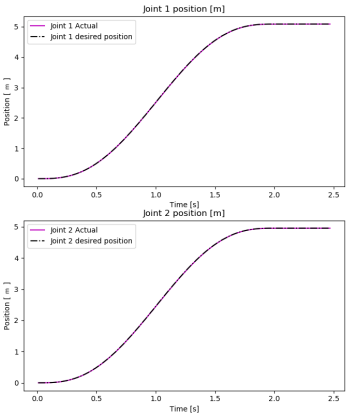


Figure 6: Joint 1 and 2 trajectory following, comparison of simulation and model.

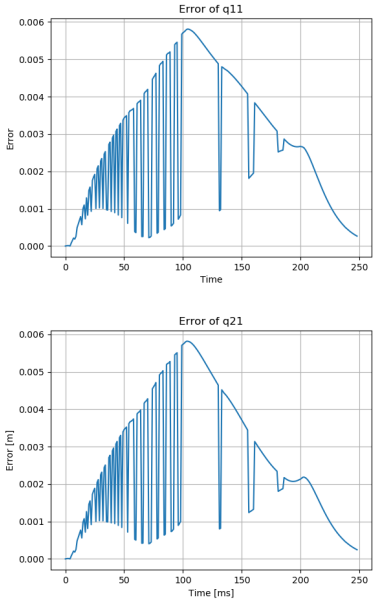


Figure 7: K_p=60, K_v=15

The mean value of the error for this combination is equal to 0.0027 m for *q11* and 0.0025 m for *q21*, while the maximum one is 0.0058 m for both joints.

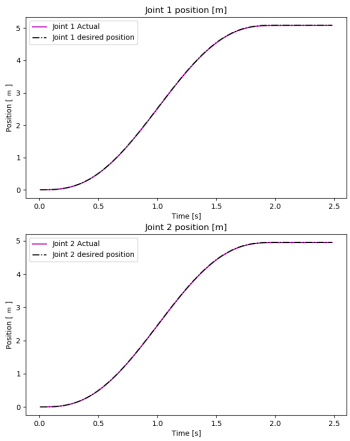


Figure 8: Joint 1 and 2 trajectory following, comparison of simulation and model.

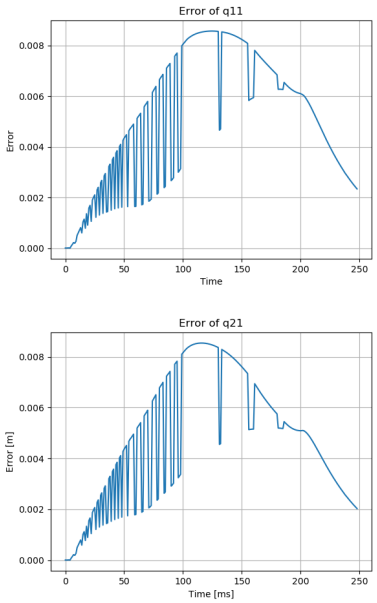


Figure 9: K_p=30, K_v=15

In this configuration, the mean errors are 0.0051 m for *q11* and 0.0048 m for *q21*, while the maximum one is 0.0085 m for both joints.

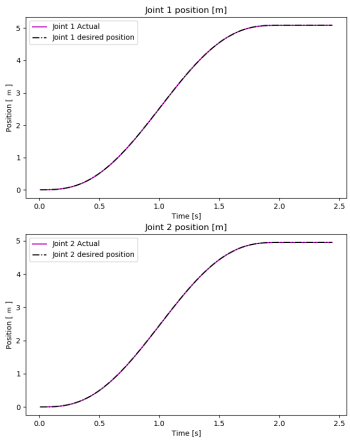


Figure 10: Joint 1 and 2 trajectory following, comparison of simulation and model.

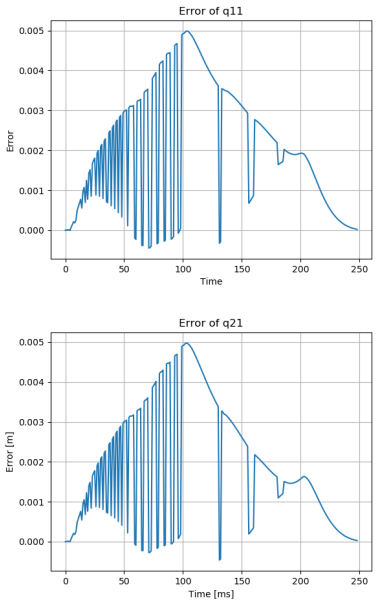
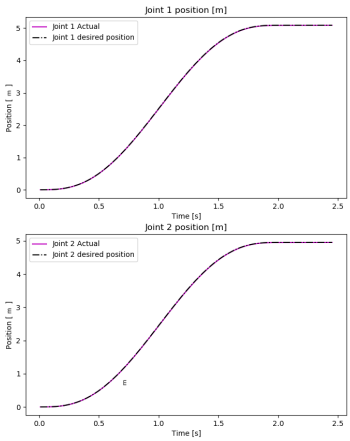


Figure 11: K_p=80, K_v=15

In this combination of position and velocity gains, the mean error is 0.0020 m for *q11* and 0.0019 m for *q21*, while the maximum one is 0.0049 m for both joints.



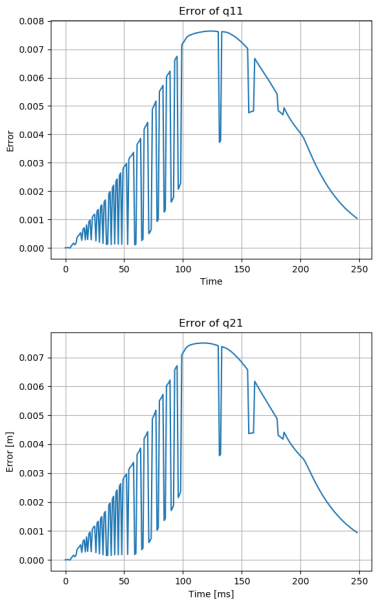


Figure 12: Joint 1 and 2 trajectory following, comparison of simulation and model.

Figure 13: K_p=80, K_v=30

In this last configuration, the mean error is 0.0038 m for *q11* and 0.0037 m for *q21*, while the maximum ones are 0.0076 m and 0.0075 m, for the first and second joint respectively.
Here it is possible to see that the best combination of gains is K_p=80 and K_v=15, since the maximum error is the lowest one for both actuated joints, as well as the mean ones. Meanwhile, the worst combination is K_p=30 and K_v=15, hence it's not a good solution to decrease too much the position gain.
The other undesired configuration is the combination of K_p=80 and K_v=30, therefore it's not recommended to increase too much the velocity gain as well.
As last, the controller was tested in a situation of different initial conditions of the trajectory, with respect to the actual position of the robot's joints. In particular, it was chosen a trajectory with initial position of the end-effector equal to *(x,y)=(0.0,1.0)*, with position gain of 80 and velocity gain of 15, so far the best combination of values tested.

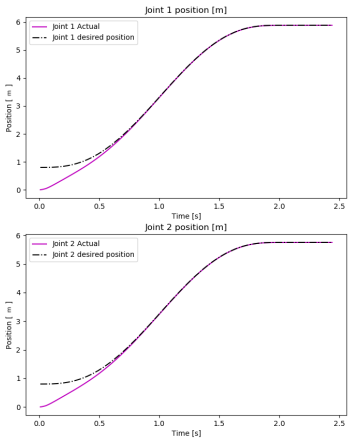


Figure 14: Joint 1 and 2 trajectory following, comparison of simulation and model.

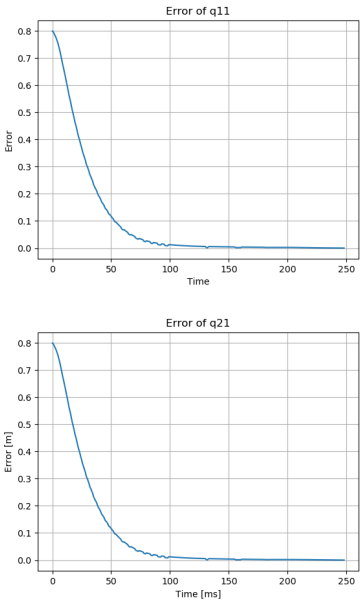


Figure 15: K_p=80, K_v=15

For this last experiment, the mean value of the error is 0.0947 m for *q11* and 0.0943 m for *q21*, while the maximum one is 0.8 m for both joints.
It is possible to notice that the mean error is so high with respect to the other cases because of the initial error position of the end-effector. Despite this, after a bit more than 0.5 s this error is compensated, and both joints follow correctly the desired joint trajectory.

4 Conclusion

This lab on the modeling and control of the *Biglide* mechanism provided a valuable opportunity to apply theoretical concepts from the course, including geometric, kinematic and dynamic modeling of parallel robots and the use of *Computed Torque Control* (CTC). We developed first and second-order geometric and kinematic models and tested them in GAZEBO to compare computed positions, velocities, and accelerations with actual data. The comparison showed a good level of accuracy, with acceptable maximum and mean errors, confirming the effectiveness of the model.

In the control phase, the choice of gains K_p and K_v significantly impacted performance, demonstrating that overly high or low gains can hinder accurate trajectory tracking. Among various tests, the combination $K_p = 80$ and $K_v = 15$ showed the best results, with minimal errors both under standard conditions and when starting from different initial positions. This confirmed the importance of selecting gains to pursue a precise and stable control.

At the end, this lab provided in-depth insight into the modeling and control

of the *Biglide*, enhancing our capability to apply analytical tools and software for robotic system verification and optimization. The results demonstrate the validity of our control approach while highlighting the importance of accurate parameter selection and ongoing experimental validation.