AMORO Lab - Report

Raffaele Pumpo, Emanuele Buzzurro

13/11/2023

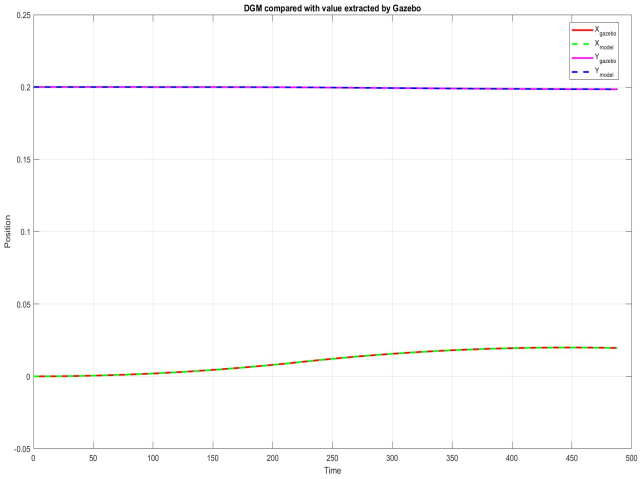
1 Introduction

This part of the lab with subject the Biglide robot has led us to a profound exploration of the different types of models and the intricate interplay between theory and practice.
Within the confines of our lab, every line of code and adjustment to models has been a labor of dedication and ingenuity. Our Biglide robots have been subject to meticulous scrutiny, with a focus on comparing the theoretical models against the real-world values extracted by Gazebo simulations.
During the work, challenges were met head-on, and discrepancies were identified and addressed. The scrutiny, though rigorous, has been instrumental in refining our models and controls.
The models and controls, once corrected the errors, could be now considered almost perfect.

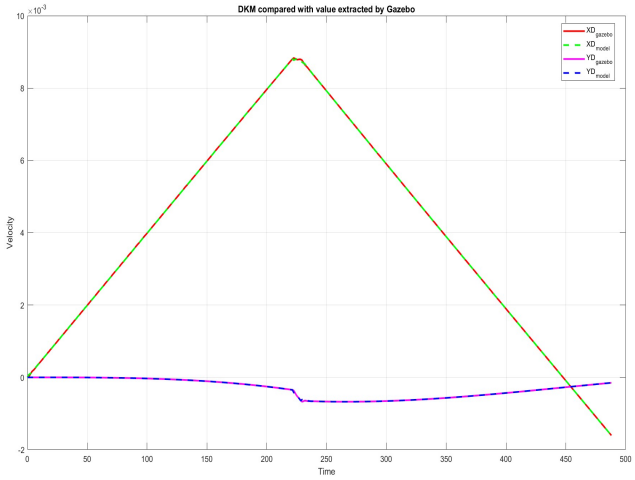
2 Comparison of models

Conducting this comparative study to ensure the reliability of the models, the Biglide robot model was subjected to a battery of simulations replicating real-world scenarios. The resulting data from these simulations were meticulously extracted and compared with the data computed by the model in a controlled environment.
The graphical representation of the datasets further, underscores the reliability of the Biglide robot model.
The Plots show direct,inverse and dynamic models, reveal a striking overlap between the model-generated and simulation-extracted data. This convergence is consistent across various operational scenarios, signifying a robust agreement between the model and simulation outputs.
Additionally, the quantitative assessment of our model's performance includes mean error and max error computations. These metrics provide a succinct summary of the disparities between the model and simulation data. The mean error, representing the average deviation between the datasets, attests to the overall accuracy of our model. Simultaneously, the max error, indicating the maximum deviation at any given point, offers insight into the model's performance under extreme conditions. These error metrics further emphasize the robustness of our model in achieving a high degree of alignment with the Gazebo simulation, validating its efficacy for real-world applications.
In our comprehensive comparison between the model-generated data and the simulation-extracted data for the Biglide robot, the visual congruence between the datasets is strikingly evident across various parameters. The plots reveal a remarkable overlap, affirming the accuracy of our model in replicating the behavior observed in the Gazebo simulation.

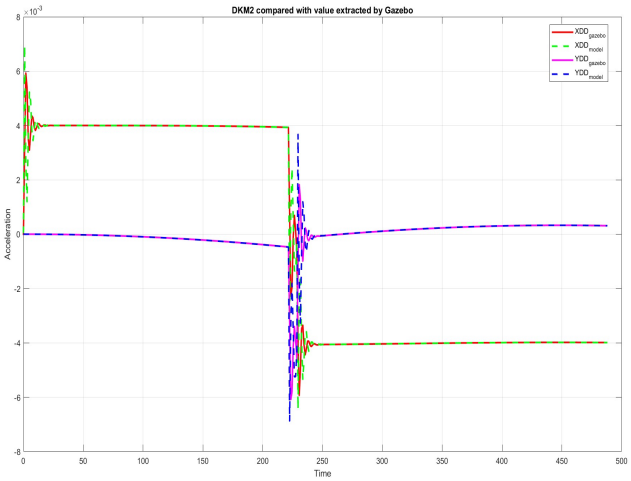
2.1 Direct Models of the End-Effector



(a) DGM



(b) DKM



(c) DKM2

Figure 1: Direct models compared with Gazebo's values

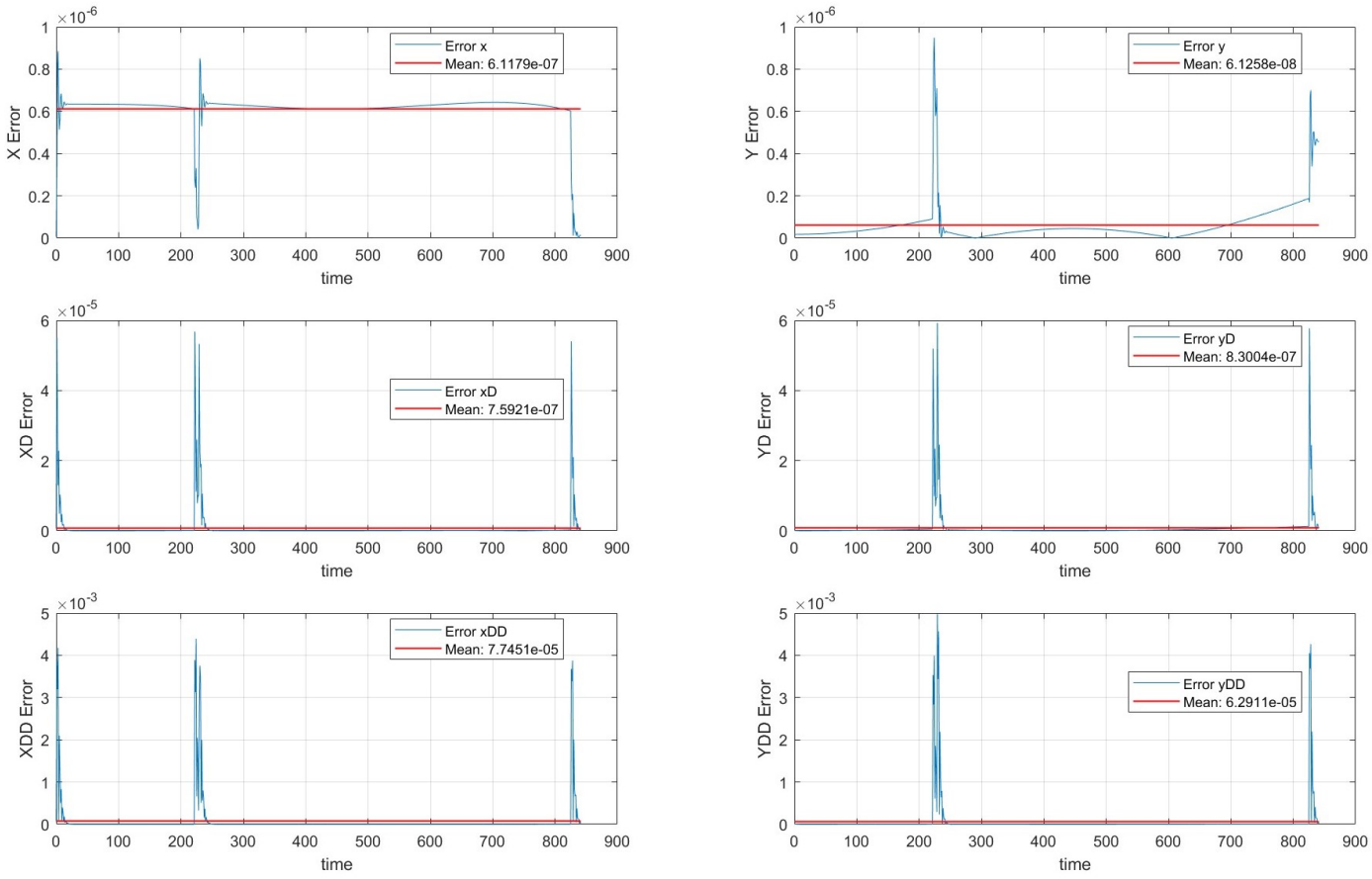
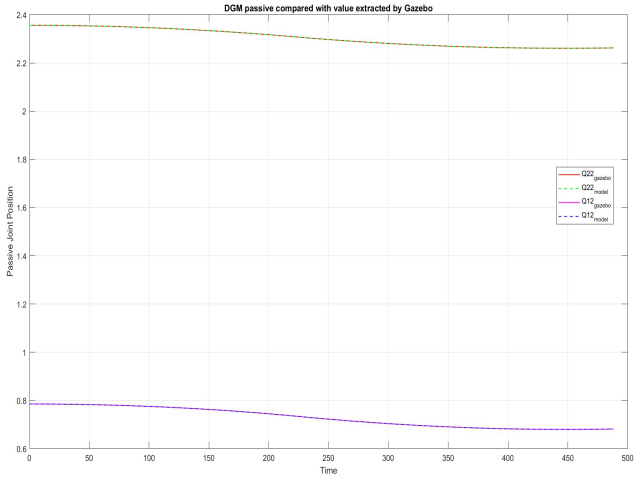


Figure 2: Errors in the DGM,DKM,DKM2

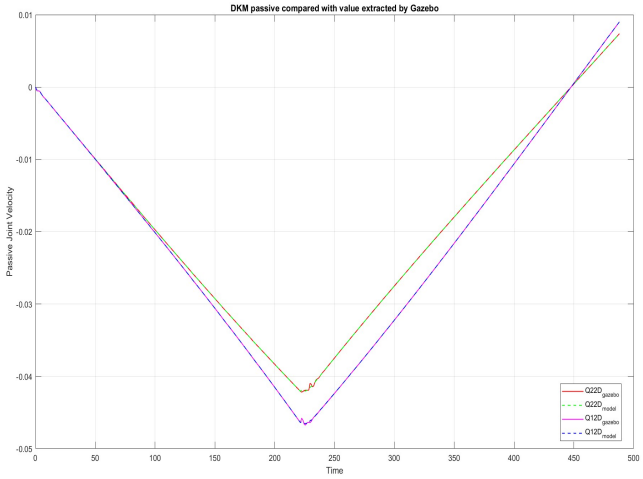
Error max of DGM = [8.8463e-07, 9.4899e-07]
Error max of DKM = [5.6781e-05,5.9263e-05]
Error max of DKM2 = [0.0044,0.0050]

In each plot of the errors is shown also the value of the mean error as red line.
For Direct Geometric Model (DGM), the mean errors is 6.11e-07 along x and 6.12e-07 for y, while the maximum error values are impressively small, approximately 8.8463e-07 in the x-direction and 9.4899e-07 in the y-direction.
In the realm of Direct Kinematic (DKM), the model exhibits commendable accuracy, with mean and maximum errors respectively 7.59e-07 and 5.6781e-05 in the x-direction, 8.30e-07 and 5.9263e-05 in the y-direction, further substantiating the reliability of our model in capturing intricate kinematic details.
The analysis extends to Second-Order Kinematic Model (DKM2), where the model demonstrates robust performance. The mean and maximum errors in this domain amount respectively to 7.74e-05 and 0.0044 in the x-direction, 6.29e-05 and 0.0050 in the y-direction. Although slightly larger than the errors observed in DGM and DKM, these values remain within an acceptable range, underscoring the model's capability to accurately replicate dynamic kinematic behavior.

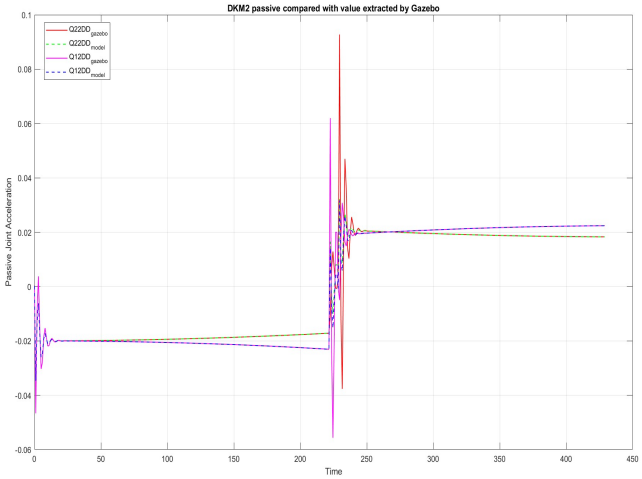
2.2 Direct Models of Passive Joints



(a) DGM Passive



(b) DKM Passive



(c) DKM2 Passive

Figure 3: Direct models of the passive joints compared with Gazebo's values

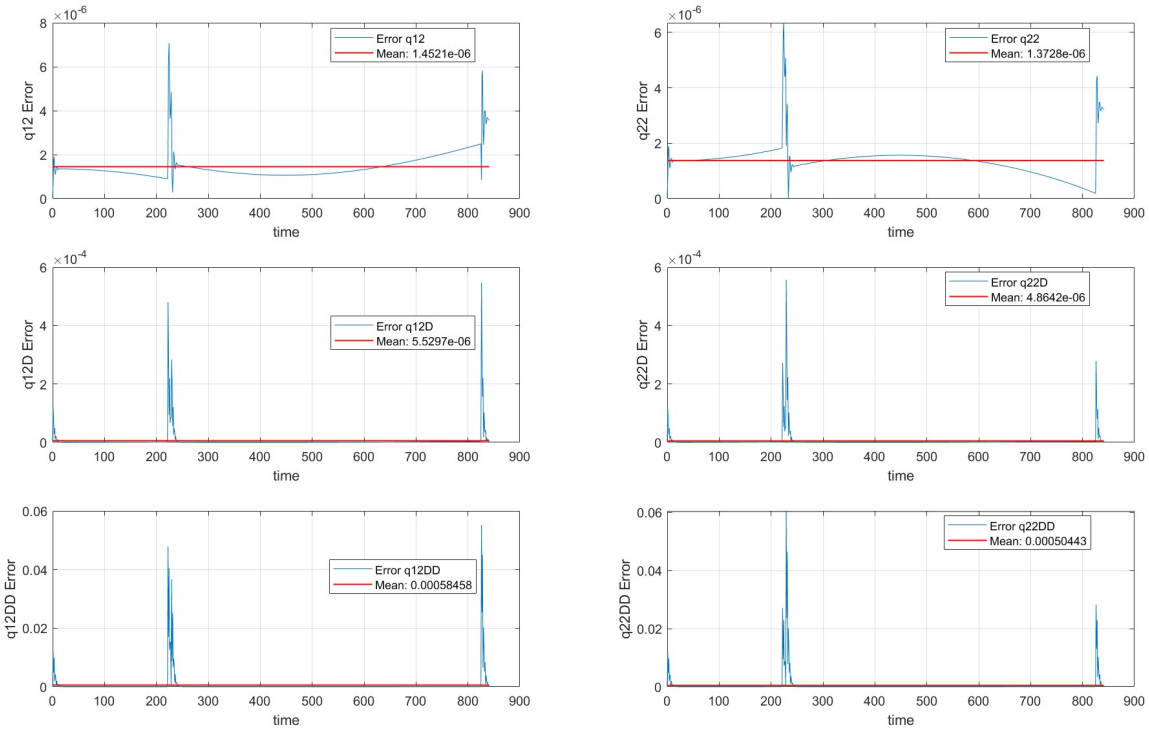
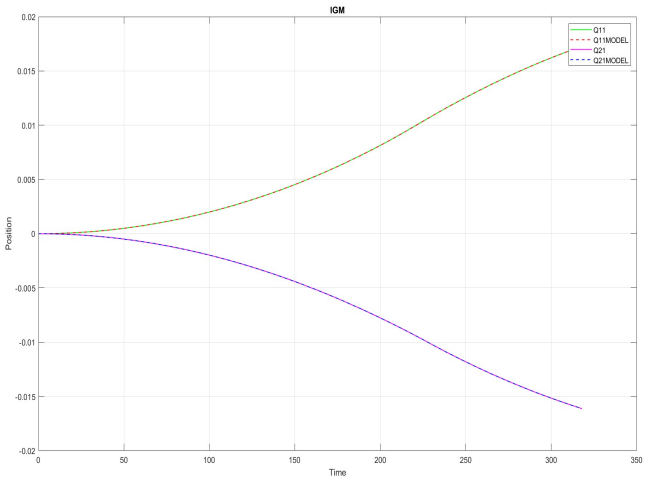


Figure 4: Errors in the DGM,DKM,DKM2 of the passive Joints

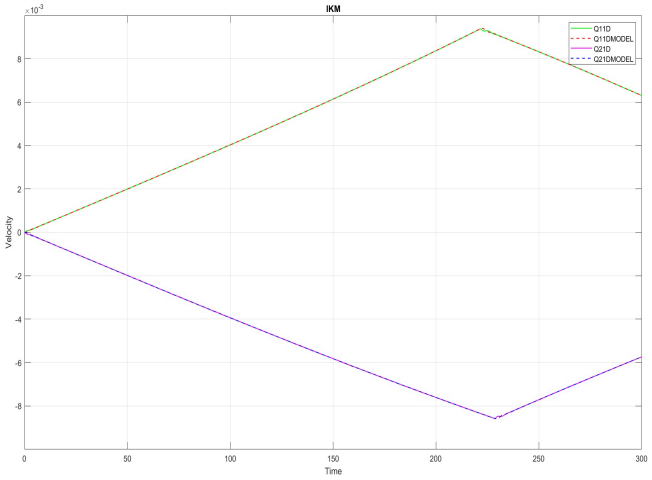
Error max of DGM Passive = [0.7072e-05 0.6347e-05]
Error max of DKM Passive = [0.5477e-03 0.5565e-03]
Error max of DKM2 Passive = [0.0552 0.0605]

As done in the Direct model of the end effector, also for the revolute joint has been done the same analysis regarding the value of the errors for each models. The values of the mean errors can be easily seen in the plot through the red line, and the values of the maximum errors are written above respectively for the first passive joint (q_{12}/ $\phi 1$) and the second one (q_{22}/ $\phi 2$).
In summary, the precision exhibited by our model in reproducing the behavior of the two passive Revolute joints, as evidenced by the mean errors and maximum error values, underscores its reliability and suitability for capturing the intricacies of the Biglide robot's.

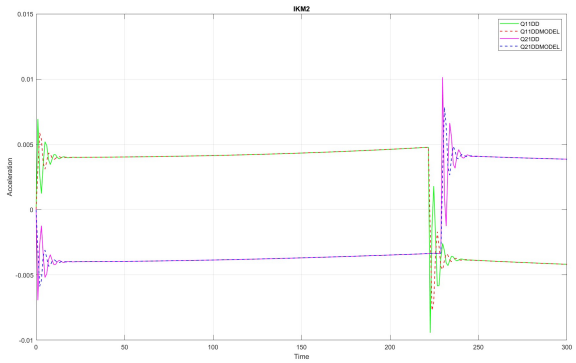
2.3 Inverse Models



(a) IGM



(b) IKM



(c) IKM2

Figure 5: Inverse models compared with Gazebo's values

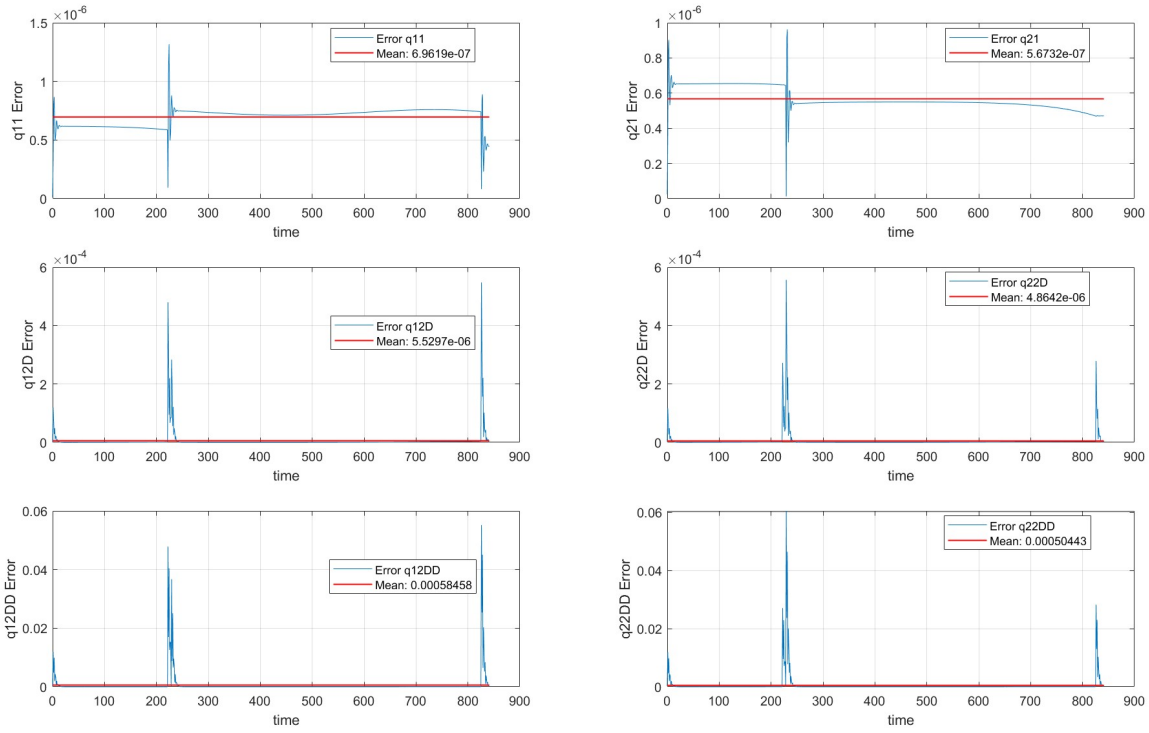


Figure 6: Errors in the IGM,IKM,IKM2

Error max of IGM = [0.1317e-05 0.0962e-05]
Error max of IKM = [0.1148e-03 0.1074e-03]
Error max of IKM2 = [0.0089 0.0077]

Observing the values of the errors and the plots of the dataset, it is possible to conclude that the Inverse Models effectively translate desired end-effector positions,velocities and accelerations into corresponding joint ones, providing a seamless and accurate inverse mapping.

2.4 Dynamic Model

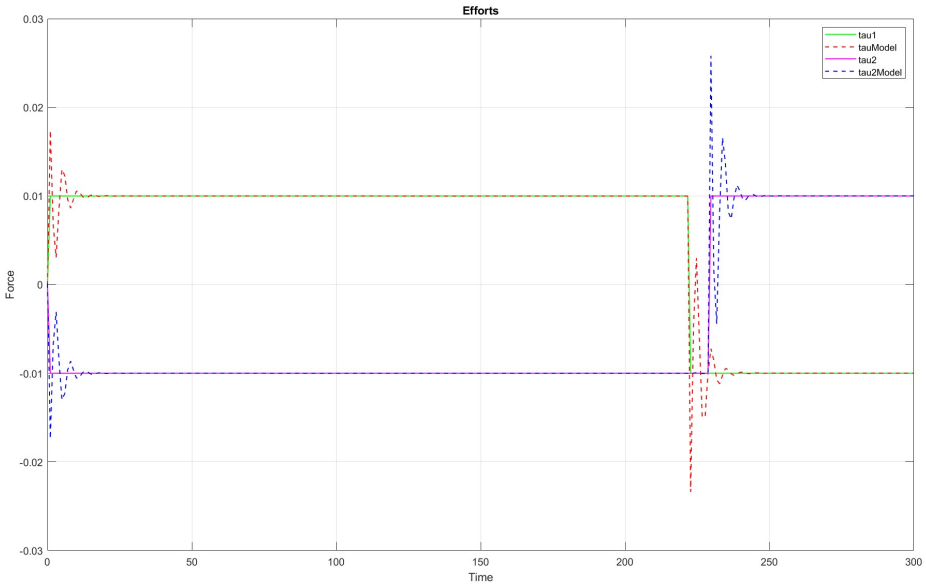


Figure 7: Comparison Efforts computed and Gazebo values

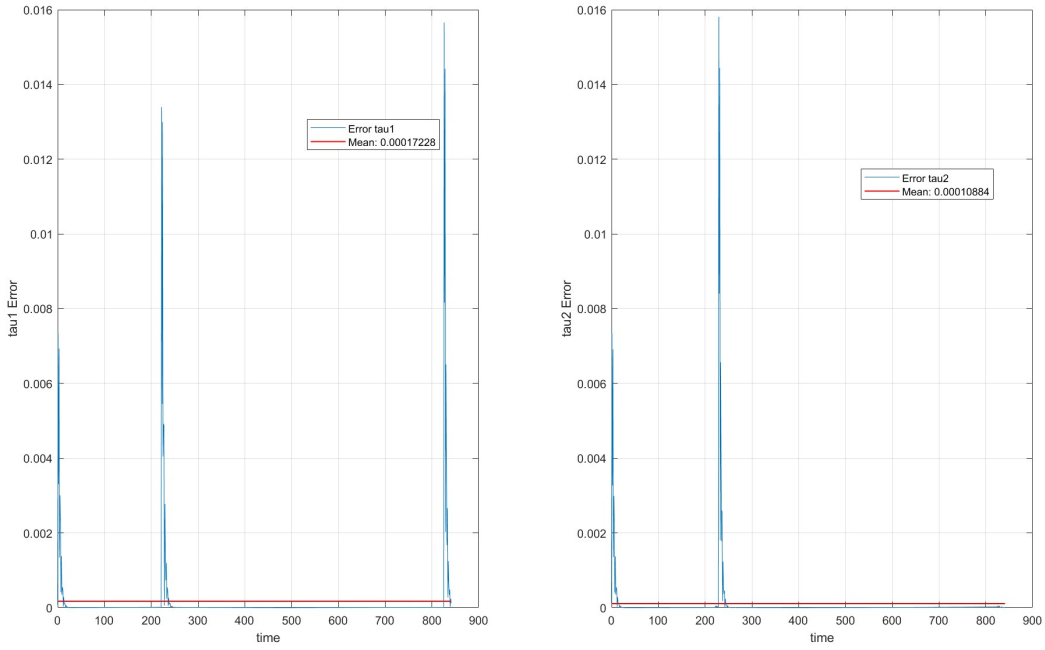


Figure 8: Errors in the Dynamic Model

Error max of Dynamic Model = [0.0157 0.0158]

Even the accuracy of our dynamic model is notably evident, particularly considering the maximum error values of efforts and the mean errors (red line in the plots).
These errors point out the model's capability to accurately predict the forces required for controlling the Biglide robot.

3 Control

After validating the precision of our geometric, kinematic, and dynamic models through the GAZEBO simulation, we harnessed their capabilities in executing a Computed Torque control algorithm.
This control technique excels in orchestrating the robot's motion trajectory through a predefined set of spatial points. It takes into account not only the desired position in space but also the velocities and accelerations to be tracked during the movements toward the target position.
Then the control has been done in three main steps:

- Trajectory, Velocity, Acceleration in the Cartesian space
- Trajectory, Velocity, Acceleration in the Joint space
- Controller

3.1 Cartesian Space

This part has been done through a polynomial trajectory, and given the values of initial and final position of the end effector it's easily computed the values of positions, velocities and accelerations for each time instant, as following:

$$\begin{cases} x(t) = x_A + s(t) \cdot (x_B - x_A) \\ y(t) = y_A + s(t) \cdot (y_B - y_A) \end{cases}$$

$$\begin{cases} \dot{x}(t) = \dot{s}(t) \cdot (x_B - x_A) \\ \dot{y}(t) = \dot{s}(t) \cdot (y_B - y_A) \end{cases}$$

$$\begin{cases} \ddot{x}(t) = \ddot{s}(t) \cdot (x_B - x_A) \\ \ddot{y}(t) = \ddot{s}(t) \cdot (y_B - y_A) \end{cases}$$

with :

$$s(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5$$

$$\dot{s}(t) = \frac{30}{t_f}\left(\frac{t}{t_f}\right)^2 - \frac{60}{t_f}\left(\frac{t}{t_f}\right)^3 + \frac{30}{t_f}\left(\frac{t}{t_f}\right)^4$$

$$\ddot{s}(t) = \frac{60}{t_f^2}\left(\frac{t}{t_f}\right) - \frac{180}{t_f^2}\left(\frac{t}{t_f}\right)^2 + \frac{120}{t_f^2}\left(\frac{t}{t_f}\right)^3$$

Where:

- (x_A, y_A): coordinates of the starting point of the trajectory.
- (x_B, y_B): coordinates of the ending point of the trajectory.
- t_f: time duration of the trajectory following.

3.2 Joint Space

In this part of the controller, obtained the trajectory, velocity and acceleration of the end effector, the corresponding ones in the joint space are simply computed through the inverse models IGM, IKM, IKM2. Since there are anyway some errors, they are only the desired values, for this reason it necessary a controller.

3.3 Controller

As previously said, for any type of errors that the robot could run into, such as error in model,in sensor and so on, it's necessary to use a controller to guarantee that the robot's behaviour is, as close as possible, to the desired one.
The controller chosen in this lab is a PD controller. It operates based on two key components: proportional and derivative actions.
The proportional component responds to the current error, which is the difference between the desired trajectory and the actual system output.
The controller adjusts the system output in proportion to this error. This helps in reducing steady-state errors and brings the system closer to the desired setpoint.
The derivative component, on the other hand, considers the rate of change of the error over time. By incorporating the rate of change information, the PD controller anticipates future trends and reacts to prevent overshooting and oscillations. This aspect enhances the system's stability and improves its transient response.
The dynamic model of the robot is the following:

$$\tau = M\ddot{q_a} + c$$

Where M, c are the Inertial and Centrifugal/Coriolis matrices extract by the Dynamic models.
For the PD controller:

$$\ddot{q_a} = \ddot{q}(t) + K_d(\dot{q}(t) - \dot{q_a}) + K_p(q(t) - q_a)$$

Then:

$$\tau = M(\ddot{q}(t) + K_d(\dot{q}(t) - \dot{q_a}) + K_p(q(t) - q_a)) + c$$

During the experiments we have tried different values for the gains, and we can conclude that:

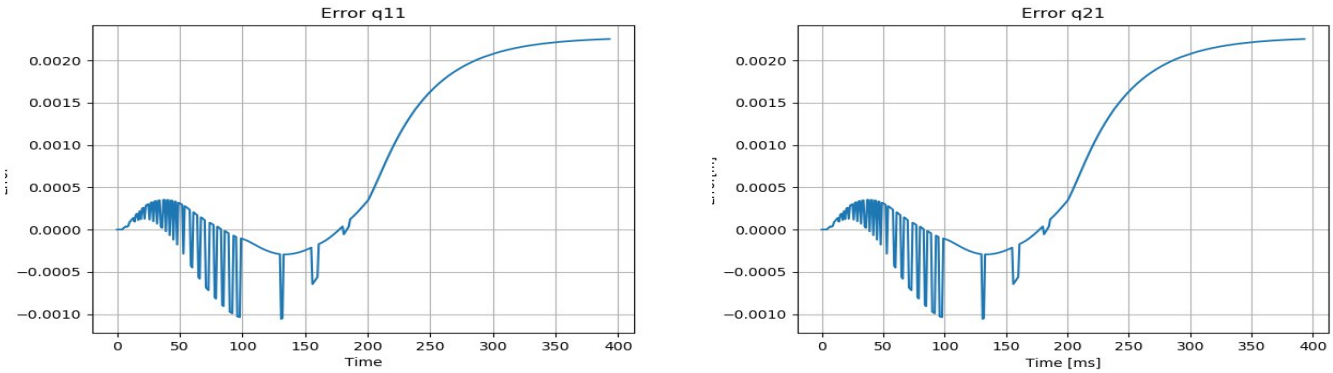
- Increasing K_p the controller tries to compensate as fast as possible the error, but it could lead to oscillation,instability and overshooting.
  In fact when the robot ideally should reach the final position, it passes that position and then goes back (overshooting).

- Increasing K_d helps to reduce overshooting and oscillations caused by changes in the rate of error.
  On the other hand it leads to a slower response of the robot, it increases an initial error between desired trajectory and real one, but avoid oscillations and to pass the overshooting.

For this the gains work together to provide a delicate balance between responsiveness and stability.
So, we have tried to tune Kp and Kd , for minimizing the error between the ideal trajectory of the joint and real one and considering the fact above explained.
We have chosen $Kp = 30$ and Kd= 15 obtaining :

- Mean error of q11 = 0.0011114647905801688
- Max error of q11 = 0.0022743228022460427
- Mean error of q21 = 0.0011114647905801688
- Max error of q21 = 0.0022743228022460427

And the following errors behaviours:



(a) Error in q11 trajectory

(b) Error in q21 trajectory

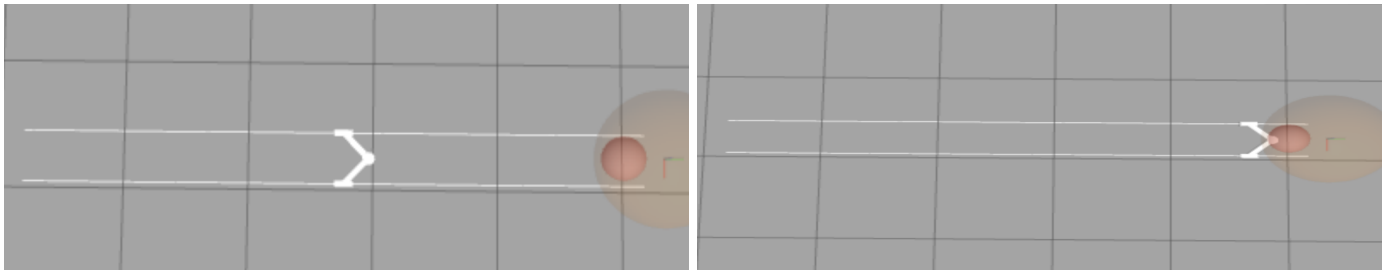
Figure 9: Errors joint trajectories

Both graphics are computed with the first trajectory (along y) where we have a worst behaviour, in the first trajectory the errors are smaller.

4 Trajectory

Once that the controller has been built, for choosing the trajectory is sufficient change the value of the positions.
In our case we have chosen as initial point, through the data of the joints given by gazebo and the DGM, the position in which the robot is put, considering the DGM accurate enough, and for the final position we have chosen two different trajectories.

- The first one where has been incremented only the value of the y coordinates, with a random value, such that is always reachable because there are no joint limits for the actuated joints and they move along y-direction.
  As following, where the red sphere is the final position to reach:

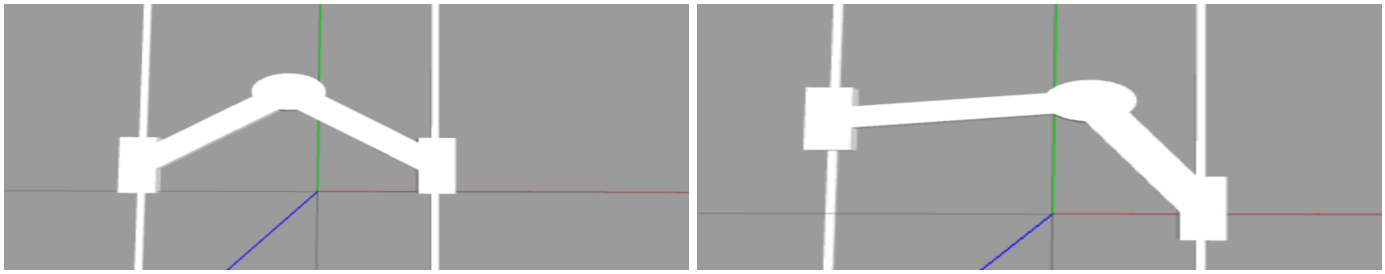


(a) Initial Position

(b) Final Position

Figure 10: Trajectory only along y

- The second trajectory has been chosen keeping the y coordinates constant, and the x can be changed inside an interval that is allowed by the structure of the robot, in such way that the IGM can be computed, then both the square root (inside the IGM) must have positive value, for this reason x must be inside the range:
  $-\frac{l}{2} + \frac{d}{2} \leq x \leq \frac{l}{2} - \frac{d}{2} \implies$ -0.0828427$\leq x \leq$ 0.0828427
  So the final position has been chosen with x = 0.08:



(a) Initial Position (b) Final Position

Figure 11: Trajectory changing x

In any case the initial position should be at the origin since the value of l and d are not expressed in the absolute frame. From the two trajectories, we can observe that there are no problem if we change only the y coordinates, since the actuated joint are prismatic and moving along y.
Instead the value of the x, for any trajectories, must be carefully analyzed to allow the mechanism to reach it.

5 Conclusion

In this lab the Biglide robot has been analysed. Firstly all the model have been computed through a geometrical analysis. Initially, we made different errors booth in the geometrical analysis and then in the code, but through a statistical analysis and comparison with Gazebo's data all the errors have been identified. Once obtain the correct models, we proceeded with a control analysis, using our models and finding the tuned parameters for the controller, such that the robot is able to compute the trajectories as close as possible to desired ones.

A possible improvement, that we were not able to reach yet, is to extend the controller also for trajectories which include the 2-type singularity. The robot run into this configuration when the two legs are aligned, it leads to an uncontrollable motion of the end effector even if the actuators are blocked.
This is caused by the singularity of the A matrix that relates the velocity and acceleration of the end effector to those of the actuated joints.
It could be solved doing the pseudoinverse of A and imposing an acceleration in that configuration of the end-effector along y coordinates = 0. Then the robot will exit from those configuration and It can continue its trajectory

In conclusion, the comprehensive exploration, validation, and refinement of models lead to precision in reproducing the robot's behavior, coupled with the adaptability of the control system, underscores the reliability and suitability of our approach for real-world applications.
The lab has been necessary to pass from theory to practice and it has not only deepened our understanding of robotic systems but has also equipped us with tools to navigate the complexities of kinematics, dynamics, and control in practical scenarios.