Lab 4: Kinematics and Dynamics of a Biglide Robot

Advanced Modeling of Robots - Ecole Centrale Nantes

Justin Lee
Student ID: 200406B
justin.lee@eleves.ec-nantes.fr

Steven Palma Morera
Student ID: 200443K
steven.palma-morera@eleves.ec-nantes.fr

***Abstract*—This document details the computation of the Geometric, Kinematic and Dynamic Models for a Biglide robotic platform. These models were used for a MATLAB and Simulink simulation where their resulting behaviours was compared to one obtained from an ADAMS model simulation of the same robotic platform. In addition, a control co-simulation for trajectory tracking and a Type 2 Singularity crossing were also developed. All the computations done proved to be accurate since all the the errors between the behaviours were close to zero. The exception to this is the Type 2 Singularity crossing criterion, as the ADAMS model was not able to handle this well.**

***Index Terms*—Biglide, Kinematics, Dynamics, MATLAB, ADAMS, Simulink, Singularity**

I. Introduction

From [1], the architecture of the five-bar robot mechanism, known as Biglide, is shown in Fig.1.

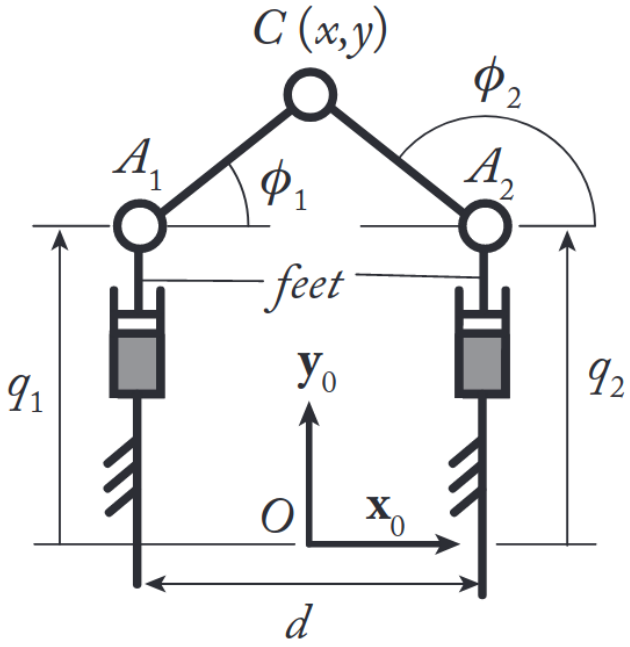

Fig. 1. Biglide robotic platform. Source: [1]

In this robot, the two prismatic joints are active while the three rotational joints are passive. The geometric and dynamic parameters are:

- $d = 0.4$ m
- $l = l_{A1C} = l_{A2C} = 0.3606$ m
- $m_p = 3$ kg, being the mass of the end-effector
- $m_f = 1$ kg, being the mass of each foot
- All other dynamics parameters are neglected.

II. Geometric, Kinematic and Dynamic Model

A. Preliminaries

Following the same notation from the lectures, $\mathbf{x}$ is a vector formed by the planar coordinate values of the end-effector C, $\mathbf{q_a}$ is defined as a vector formed by the values of the active joints and $\mathbf{q_d}$ is defined as a vector formed by the values of the passive joints.

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}; \mathbf{q_a} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}; \mathbf{q_d} = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \tag{1}$$

B. Geometric Model

The objective of this sub-section is to obtain a mathematical expression for the planar coordinates of the end-effector ($\mathbf{x}$) and for the values of the passive joints ($\mathbf{q_d}$) in terms of the active joint values ($\mathbf{q_a}$). The same procedure implemented in Lab 2 was followed for computing $\mathbf{x}$:

$$\begin{aligned} \overrightarrow{OC} &= \overrightarrow{OA_2} + \overrightarrow{A_2M} + \overrightarrow{MC} \\ \overrightarrow{OC} &= \overrightarrow{OA_2} + \overrightarrow{A_2M} \\ &+ \gamma \frac{\|\overrightarrow{MC}\|}{\|\overrightarrow{A_2M}\|} R_z(\frac{\pi}{2}) \overrightarrow{A_2M} \end{aligned} \tag{2}$$

Where M is the mid-point of the vector $\overrightarrow{A_1A_2}$, R_z is the rotational matrix around the z-axis, $\|\overrightarrow{MC}\|$ is the distance between the mid-point M and the point C and γ is either -1 or $+1$ depending on the configuration of the end-effector.

$$\begin{aligned} M &= \begin{bmatrix} 0 \\ (q_1 + q_2)/2 \end{bmatrix}; \\ R_z(\theta) &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}; \\ \|\overrightarrow{MC}\| &= \sqrt{{l_{A2C}}^2 - \|\overrightarrow{A_2M}\|} \end{aligned} \tag{3}$$

After substituting the values of each vector, equation (4) was obtained. This corresponds to the expression of $\mathbf{x}$ in terms of $\mathbf{q_a}$. Notice that the value of γ is -1 since the configuration studied is the one shown in the Fig.1, where the end-effector is on top of the feet.

$$\begin{aligned}\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \overrightarrow{OX} = \begin{bmatrix} d/2 \\ q_2 \end{bmatrix} + \begin{bmatrix} -d/2 \\ (q_1 - q_2)/2 \end{bmatrix} \\ - \sqrt{\frac{4l^2 - d^2 - (q_1 - q_2)^2}{d^2 + (q_1 - q_2)^2}} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -d/2 \\ (q_1 - q_2)/2 \end{bmatrix}\end{aligned} \tag{4}$$

Now, to compute the expression of the passive joint variables ($\mathbf{q_d}$), the loop closure equations of the system are obtained.

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -d/2 + l\cos(\phi_1) \\ q_1 + l\sin(\phi_1) \end{bmatrix} = \begin{bmatrix} d/2 + l\cos(\phi_2) \\ q_2 + l\sin(\phi_2) \end{bmatrix} \tag{5}$$

Equation (5) can then be solved for ϕ_1 and ϕ_2 as shown in equations (6) and (7), which results in equation (8). Therefore, equations (4) and (8) are the results of the Geometric Model since both of them are only in terms of the active joints values $\mathbf{q_a}$.

$$\begin{aligned}\frac{y - q_1}{x + d/2} &= \frac{l\sin(\phi_1)}{l\cos(\phi_1)} \\ \phi_1 &= \arctan(\frac{y - q_1}{x + d/2})\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{y - q_2}{x - d/2} &= \frac{l\sin(\phi_2)}{l\cos(\phi_2)} \\ \phi_2 &= \arctan(\frac{y - q_2}{x - d/2})\end{aligned} \tag{7}$$

$$\mathbf{q_d} = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} \arctan(\frac{y-q_1}{x+d/2}) \\ \arctan(\frac{y-q_2}{x-d/2}) \end{bmatrix} \tag{8}$$

C. Kinematic Model

The objective of this sub-section is to obtain a mathematical expression for the velocity and acceleration of both the end-effector ($\dot{\mathbf{x}}, \ddot{\mathbf{x}}$) and the passive joints ($\dot{\mathbf{q}}_\mathbf{d}, \ddot{\mathbf{q}}_\mathbf{d}$), in terms of the active joint values ($\mathbf{q_a}, \dot{\mathbf{q}}_\mathbf{a}, \ddot{\mathbf{q}}_\mathbf{a}$). To find these values, the first and second derivative of both equations (4) and (5) could be used; however it would be very complex method. Instead the approach seen in class ([2]) will be used, which involves finding constraint equations in the form $\mathbf{h}(\mathbf{x}, \mathbf{q_a}) = 0$ and $\mathbf{h_q}(\mathbf{q_d}, \mathbf{q_a}) = 0$. From these constraints, the first and second derivative of $\mathbf{x}$ and $\mathbf{q_d}$ can be easily computed.

1) Finding $\mathbf{h}(\mathbf{x}, \mathbf{q_a}), \dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$: To find $\mathbf{h}(\mathbf{x}, \mathbf{q_a})$ we can come up with a new constraint by finding the intersection of two circles with centers at A1 and A2 respectively. From this, two equations are found:

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} (x + \frac{d}{2})^2 + (y - q_1)^2 - l^2 \\ (x - \frac{d}{2})^2 + (y - q_2)^2 - l^2 \end{bmatrix} = \mathbf{0} \tag{9}$$

This constraint is utilised to find matrices A and B which form a new constraint:

$$\mathbf{A}\dot{\mathbf{x}} + \mathbf{B}\dot{\mathbf{q}}_\mathbf{a} = 0 \tag{10}$$

To get the values of matrices $\mathbf{A}$ and $\mathbf{B}$, the partial derivative of $\mathbf{h}(\mathbf{x}, \mathbf{q_a})$ is taken with respect to $\mathbf{x}$ and $\mathbf{q_a}$ respectively. The partial derivatives are shown below:

$$\begin{aligned}\frac{\partial h_1}{\partial \mathbf{x}} &= 2(x + \frac{d}{2})\dot{x} + 2(y - q_1)\dot{y} \\ \frac{\partial h_1}{\partial \mathbf{q_a}} &= -2(y - q_1)\dot{q_1} \\ \frac{\partial h_2}{\partial \mathbf{x}} &= 2(x - \frac{d}{2})\dot{x} + 2(y - q_2)\dot{y} \\ \frac{\partial h_2}{\partial \mathbf{q_a}} &= -2(y - q_1)\dot{q_2}\end{aligned} \tag{11}$$

This leads to the following formulation:

$$\begin{aligned}\mathbf{A}\dot{\mathbf{x}} &= \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{x}} \\ \frac{\partial h_2}{\partial \mathbf{x}} \end{bmatrix} = 2 \begin{bmatrix} x + \frac{d}{2} & y - q_1 \\ x - \frac{d}{2} & y - q_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \\ \mathbf{B}\dot{\mathbf{q}}_\mathbf{a} &= \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{q_a}} \\ \frac{\partial h_2}{\partial \mathbf{q_a}} \end{bmatrix} = -2 \begin{bmatrix} y - q_1 & 0 \\ 0 & y - q_2 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}\end{aligned} \tag{12}$$

Therefore, $\mathbf{A}$ and $\mathbf{B}$ are as follows:

$$\begin{aligned}\mathbf{A} &= 2 \begin{bmatrix} x + \frac{d}{2} & y - q_1 \\ x - \frac{d}{2} & y - q_2 \end{bmatrix} \\ \mathbf{B} &= -2 \begin{bmatrix} y - q_1 & 0 \\ 0 & y - q_2 \end{bmatrix}\end{aligned} \tag{13}$$

Rearranging equation (10), we can find $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = -\mathbf{A}^{-1}\mathbf{B}\dot{\mathbf{q}}_\mathbf{a} \tag{14}$$

$\ddot{\mathbf{x}}$ is derived in the slides and is shown below:

$$\ddot{\mathbf{x}} = -\mathbf{A}^{-1}(\mathbf{B}\ddot{\mathbf{q}}_\mathbf{a} + \dot{\mathbf{A}}\dot{\mathbf{x}} + \dot{\mathbf{B}}\dot{\mathbf{q}}_\mathbf{a}) \tag{15}$$

Where $\dot{\mathbf{A}}$ and $\dot{\mathbf{B}}$ are the time derivatives of $\mathbf{A}$ and $\mathbf{B}$ and are shown here:

$$\begin{aligned}\dot{\mathbf{A}} &= 2 \begin{bmatrix} \dot{x} & \dot{y} - \dot{q_1} \\ \dot{x} & \dot{y} - \dot{q_2} \end{bmatrix} \\ \dot{\mathbf{B}} &= -2 \begin{bmatrix} \dot{y} - \dot{q_1} & 0 \\ 0 & \dot{y} - \dot{q_2} \end{bmatrix}\end{aligned} \tag{16}$$

2) Finding $\mathbf{h_q}(\mathbf{q_d}, \mathbf{q_a}), \dot{\mathbf{q}}_\mathbf{d}$ and $\ddot{\mathbf{q}}_\mathbf{d}$: To find $\mathbf{h_q}(\mathbf{q_d}, \mathbf{q_a})$, the x and y components of equation (5) are utilised to find a new constraint:

$$\mathbf{h}_q = \begin{bmatrix} h_{q1} \\ h_{q2} \end{bmatrix} = \begin{bmatrix} l(\cos(\phi_1) - \cos(\phi_2)) - d \\ l(\sin(\phi_1) - \sin(\phi_2)) + q_1 - q_2 \end{bmatrix} = \mathbf{0} \tag{17}$$

This is used to find the components of the following equation:

$$\mathbf{A}_q \dot{\mathbf{q}_d} + \mathbf{B}_q \dot{\mathbf{q}_a} = 0 \tag{18}$$

To find $\mathbf{A}_q$ and $\mathbf{B}_q$, a similar approach is used above where the partial derivative with respect to $\mathbf{q}_d$ and $\mathbf{q}_a$ is taken of $\mathbf{h}_q(\mathbf{q}_d, \mathbf{q}_a)$. This process is shown below:

$$\begin{aligned} \frac{\partial h_{q1}}{\partial \mathbf{q}_d} &= l(-\sin(\phi_1)\dot{\phi_1} + \sin(\phi_2)\dot{\phi_2}) \\ \frac{\partial h_{q1}}{\partial \mathbf{q}_a} &= 0 \\ \frac{\partial h_{q2}}{\partial \mathbf{q}_d} &= l(\cos(\phi_1)\dot{\phi_1} - \cos(\phi_2)\dot{\phi_2}) \\ \frac{\partial h_{q2}}{\partial \mathbf{q}_a} &= \dot{q_1} - \dot{q_2} \end{aligned} \tag{19}$$

This leads to the following formulation:

$$\begin{aligned} \mathbf{A_q}\dot{\mathbf{q_d}} &= \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{q_d}} \\ \frac{\partial h_2}{\partial \mathbf{q_d}} \end{bmatrix} = l \begin{bmatrix} -\sin(\phi_1) & +\sin(\phi_2) \\ \cos(\phi_1) & -\cos(\phi_2) \end{bmatrix} \begin{bmatrix} \dot{\phi_1} \\ \dot{\phi_2} \end{bmatrix} \\ \mathbf{B_q}\dot{\mathbf{q_a}} &= \begin{bmatrix} \frac{\partial h_1}{\partial \mathbf{q_a}} \\ \frac{\partial h_2}{\partial \mathbf{q_a}} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} \end{aligned} \tag{20}$$

Therefore $\mathbf{A_q}$ and $\mathbf{B_q}$ are shown below:

$$\begin{aligned} \mathbf{A_q} &= l \begin{bmatrix} -\sin(\phi_1) & \sin(\phi_2) \\ \cos(\phi_1) & -\cos(\phi_2) \end{bmatrix} \\ \mathbf{B_q} &= \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} \end{aligned} \tag{21}$$

Rearranging equation (18), we can find $\dot{\mathbf{q}_d}$:

$$\dot{\mathbf{q_d}} = -\mathbf{A_q}^{-1}\mathbf{B}_q\dot{\mathbf{q}_a} \tag{22}$$

$\ddot{\mathbf{q}}_\mathbf{d}$ is also derived in the slides in a similar way to $\ddot{\mathbf{x}}$ and is shown below:

$$\ddot{\mathbf{q_d}} = -\mathbf{A_q}^{-1}(\mathbf{B_q}\ddot{\mathbf{q_a}} + \dot{\mathbf{A_q}}\dot{\mathbf{q_d}} + \dot{\mathbf{B_q}}\dot{\mathbf{q_a}}) \tag{23}$$

Similar to above, $\dot{\mathbf{A_q}}$ and $\dot{\mathbf{B_q}}$ are the time derivatives of $\mathbf{A_q}$ and $\mathbf{B_q}$ and are shown below:

$$\begin{aligned} \dot{\mathbf{A_q}} &= l \begin{bmatrix} -\cos(\phi_1)\dot{\phi_1} & \cos(\phi_2)\dot{\phi_2} \\ -\sin(\phi_1)\dot{\phi_1} & \sin(\phi_2)\dot{\phi_2} \end{bmatrix} \\ \dot{\mathbf{B_q}} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{24}$$

D. Dynamic Model

The objective of this section is to obtain a mathematical expression for the effort (τ) of the end effector. To do this, the Lagrange formalism will be used in conjunction with Lagrange multipliers in order to find the total effort of the system from both the active and passive joints.

1) Lagrange Formalism of System: First, the kinematic energy of the system excluding the end effector will be found. The translational ($\mathbf{v_1}$,$\mathbf{v_2}$) and rotational ($\omega_\mathbf{1}$,$\omega_\mathbf{2}$) velocity of each foot is shown below:

$$\begin{aligned} \mathbf{v_1} &= \begin{bmatrix} 0 \\ \dot{q_1} \end{bmatrix}; \mathbf{v_2} = \begin{bmatrix} 0 \\ \dot{q_2} \end{bmatrix} \\ \omega_\mathbf{1} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \omega_\mathbf{2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{25}$$

Using these velocities, the kinematic energy of each foot ($\mathbf{E_1}$,$\mathbf{E_2}$) can be found:

$$\begin{aligned} \mathbf{E_1} &= \frac{1}{2}\mathbf{v_1}^\intercal m_f \mathbf{v_1} \\ \mathbf{E_2} &= \frac{1}{2}\mathbf{v_2}^\intercal m_f \mathbf{v_2} \end{aligned} \tag{26}$$

Therefore, the total kinetic energy of the system ($\mathbf{E}$) is:

$$\begin{aligned} \mathbf{E} &= \mathbf{E_1} + \mathbf{E_2} \\ &= \frac{1}{2}(\mathbf{v_1}^\intercal m_f \mathbf{v_1} + \mathbf{v_2}^\intercal m_f \mathbf{v_2}) \\ &= \frac{1}{2}(m_f(\dot{\mathbf{q}_1}^2 + \dot{\mathbf{q}_2}^2)) \end{aligned} \tag{27}$$

The potential energy ($\mathbf{U}$) of the system is zero because the robot is lying flat, so there is no change in height wherever the robot moves. Therefore the Lagrangian ($\mathbf{L}$) of the system is merely the kinematic energy:

$$\mathbf{L} = \mathbf{E} - \mathbf{U} = \mathbf{E} \tag{28}$$

By using the Lagrange equations with multipliers, we get the following:

$$\begin{aligned} \tau + \mathbf{B_q}^\intercal \lambda = &\quad \frac{\partial}{\partial t}(\frac{\partial \mathbf{L}}{\partial \dot{\mathbf{q}_a}})^\intercal - (\frac{\partial \mathbf{L}}{\partial \mathbf{q_a}})^\intercal = \tau_\mathbf{a} \\ \mathbf{A_q}^\intercal \lambda = &\quad \frac{\partial}{\partial t}(\frac{\partial \mathbf{L}}{\partial \dot{\mathbf{q}_d}})^\intercal - (\frac{\partial \mathbf{L}}{\partial \mathbf{q_d}})^\intercal = \tau_\mathbf{d} \end{aligned} \tag{29}$$

$\tau_\mathbf{a}$ and $\tau_\mathbf{d}$ are the torques from the active and passive joints respectively. Reformulating equation (29) leads to:

$$\tau = \tau_\mathbf{a} + \mathbf{J_q}^\intercal \tau_\mathbf{d} \tag{30}$$

Where $\mathbf{J}_q$ is the following:

$$\mathbf{J_q} = -\mathbf{A_q}^{-1}\mathbf{B_q} \tag{31}$$

The partial derivatives of the Lagrangian are shown below:

$$\begin{aligned}\frac{\partial}{\partial t}(\frac{\partial \mathbf{L}}{\partial \dot{\mathbf{q_a}}}) &= m_f \begin{bmatrix}\ddot{q_1}\\ \ddot{q_2}\end{bmatrix}\\ \frac{\partial \mathbf{L}}{\partial \mathbf{q_a}} &= \mathbf{0}\\ \frac{\partial}{\partial t}(\frac{\partial \mathbf{L}}{\partial \dot{\mathbf{q_d}}}) &= \mathbf{0}\\ \frac{\partial \mathbf{L}}{\partial \mathbf{q_d}} &= \mathbf{0}\end{aligned} \tag{32}$$

2) Lagrange Formalism of End Effector: Now the Lagrange formalism of the end effector will be calculated with very similar steps. The translational ($\mathbf{v_p}$) and rotational ($\omega_\mathbf{p}$) velocity of the end effector is as follows:

$$\mathbf{v_p} = \dot{\mathbf{x}} = \begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}; \omega_\mathbf{p} = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{33}$$

From this, the kinematic energy of the end effector ($\mathbf{E_p}$) can be found:

$$\mathbf{E_p} = \frac{1}{2}\mathbf{v_p}^\intercal m_p \mathbf{v_p} \tag{34}$$

The potential energy ($\mathbf{U_p}$) of the end effector is zero for the same reasons listed above. Therefore the Lagrangian ($\mathbf{L_p}$) of the end effector is merely the kinematic energy:

$$\mathbf{L}_p = \mathbf{E}_p - \mathbf{U}_p = \mathbf{E}_p \tag{35}$$

By again using Lagrange multipliers, the following equations are derived:

$$\begin{aligned}\tau_p + \mathbf{B}^\intercal\lambda = &\quad \frac{\partial}{\partial t}(\frac{\partial \mathbf{L}_p}{\partial \dot{\mathbf{q_a}}})^\intercal - (\frac{\partial \mathbf{L_p}}{\partial \mathbf{q_a}})^\intercal\\ \mathbf{A}^\intercal\lambda = &\quad \frac{\partial}{\partial t}(\frac{\partial \mathbf{L}_p}{\partial \dot{\mathbf{x}}})^\intercal - (\frac{\partial \mathbf{L}_p}{\partial \mathbf{x}})^\intercal\end{aligned} \tag{36}$$

Simplifying equation (36) leads to:

$$\tau_\mathbf{p} = m_p \mathbf{J}^\intercal \ddot{\mathbf{x}} \tag{37}$$

Where $\mathbf{J}$ is the following:

$$\mathbf{J} = -\mathbf{A}^{-1}\mathbf{B} \tag{38}$$

3) Lagrange Formalism of Total System: Finally, the total effort $\tau_\mathbf{tot}$ of the system is:

$$\begin{aligned}\tau_\mathbf{tot} &= \tau + \tau_\mathbf{p}\\ &= \tau_\mathbf{a} + \mathbf{J_q}^\intercal \tau_\mathbf{d} + m_p \mathbf{J}^\intercal \ddot{\mathbf{x}}\\ &= m_f \ddot{\mathbf{q}}_\mathbf{a} - m_p \mathbf{A}^{-1}\mathbf{B}\ddot{\mathbf{x}}\end{aligned} \tag{39}$$

E. Type 2 Singularity Analysis

In order to understand the conditions when a Type 2 Singularity happens, the matrix $\mathbf{A}$ should be studied. When the robotic platform enters a singularity prone pose, the matrix $\mathbf{A}$ becomes singular, that is, its determinant equals zero. So by studying when its determinant goes to zero, the singularity conditions can be obtained. The determinant of the matrix $\mathbf{A}$ is shown in equation (40).

$$|\mathbf{A}| = x(q_1 - q_2) - \frac{d}{2}(q_1 + q_2) + yd \tag{40}$$

Now, substituting the values of x and y from equation (5) into equation (40) and setting it equal to zero, equation (41) is obtained.

$$\begin{aligned}\tan(\mathbf{q}_d)d + (q_1 - q_2) &= 0\\ \mathbf{q}_d = \arctan(\frac{(q_2 - q_1)}{d})&\end{aligned} \tag{41}$$

It can be seen that the singularity happens when the two arms are co-linear. At this moment, it can be seen that $x = 0$ and ($|q_1 - q_2| = \sqrt{4l^2 - d^2}$). Relating these conditions and equations (6) and (7) by substituting them into the matrix $\mathbf{A}$, equation (42) is obtained, which corresponds to the matrix $\mathbf{A}$ evaluated at the singularity point.

$$\mathbf{A}_{singular} = \begin{bmatrix}d & -\sqrt{4l^2 - d^2}\\ -d & \sqrt{4l^2 - d^2}\end{bmatrix} \tag{42}$$

In order to get a criterion for a Type 2 Singularity crossing, the equation (44) must hold, with v being the kernel of matrix $\mathbf{A}$ evaluated at the singularity point. By doing this, a relationship between $\ddot{x}$ and $\ddot{y}$ can be found. Equation (45) shows this relationship, which is the criterion needed for the end effector to successfully pass through the Type 2 Singularity.

$$v = \ker(\mathbf{A}_{singular}) = \begin{bmatrix}\sqrt{4l^2 - d^2}\\ d\end{bmatrix} \tag{43}$$

$$0 = \mathbf{v}^\intercal \begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} - \mathbf{v}^\intercal \mathbf{A}^\intercal \lambda \tag{44}$$

$$\ddot{y} = \frac{\sqrt{4l^2 - d^2}}{d}\ddot{x} \tag{45}$$

III. Design in MATLAB, Simulink and ADAMS

A. Geometric, Kinematic and Dynamic Analysis in Simulink

In Fig. 2, the implementation of the geometric, kinematic and dynamic models in Simulink are shown.

There are four main blocks in the simulink model. The first one is the Direct Geometric Model (DGM). It implements equations (4) and (8). The next block is the Direct Kinematic Model (DKM), which implements equations (14), (15), (22) and (23). The next block is the Inverse Dynamic Model

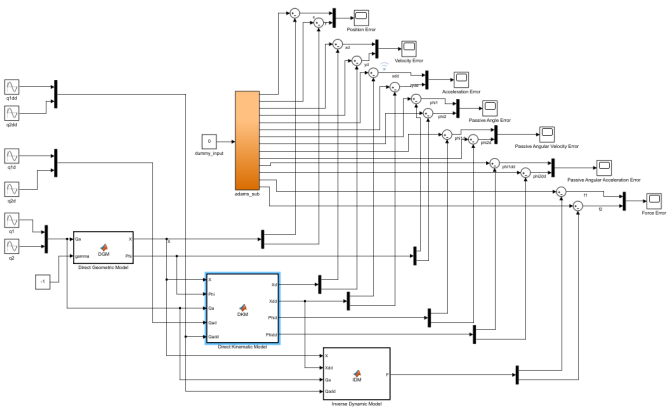

Fig. 2. Simulink model of the Biglide robot for the Geometric, Kinematic and Dynamic Models. Source: Own elaboration.

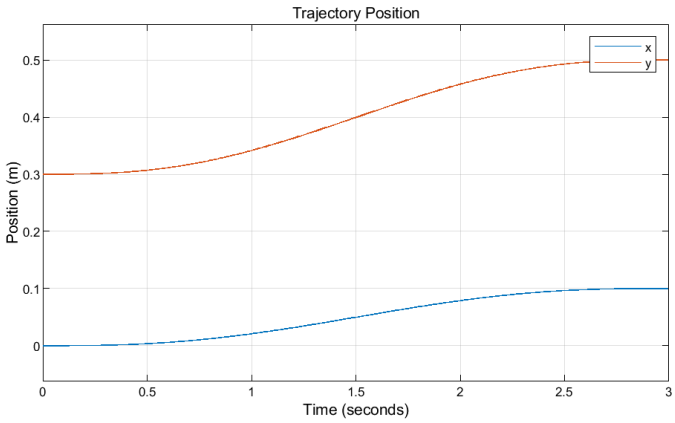

Fig. 4. Test trajectory to be followed by the end effector. Source: Own elaboration.

(IDM), which implements equation (39). The final block is the ADAMS sub module, which is based off the ADAMS model and outputs all the state variables. The procedure to obtain this ADAMS sub module is omitted for brevity, but should follow the steps from the previous labs. The outputs of the ADAMS model and the outputs of the different Simulink blocks are compared against each other to judge the accuracy of the models.

B. Trajectory Tracking

Below is the simulink model of the trajectory tracking.

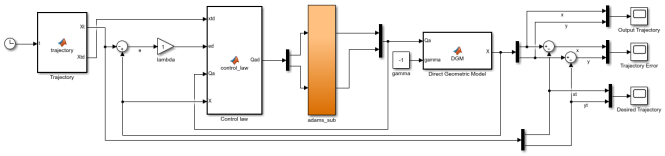

Fig. 3. Simulink model for the Trajectory Tracking. Source: Own elaboration.

There are four main blocks in this model. The Trajectory block implements the desired trajectory that the control law must follow. The trajectory used is a 3rd order polynomial based off given the initial ($\mathbf{x}_i$) and final ($\mathbf{x}_f$) position in meters and the end time (t_f) in seconds. It is assumed that velocity and acceleration at the beginning and end of the trajectory is zero. The equation is below:

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}_i + s(t)(\mathbf{x}_f - \mathbf{x}_i) \\ s(t) &= 10(\frac{t}{t_f})^3 - 15(\frac{t}{t_f})^4 + 6(\frac{t}{t_f})^5 \end{aligned} \tag{46}$$

In this case, the following are the parameters of the trajectory, which produce the trajectory shown in the Fig.4:

$$\mathbf{x}_i = \begin{bmatrix} 0 \\ 0.3 \end{bmatrix}; \mathbf{x}_f = \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix}; t_f = 3 \tag{47}$$

The control law is implemented in the block called Control Law. The object of the control law is to bring the position error to zero based off an input of $\dot{\mathbf{q}}_\mathbf{a}$. The desired trajectory position is $\mathbf{X}_t$. The error is defined then as:

$$\mathbf{e} = \mathbf{x} - \mathbf{x}_t \tag{48}$$

The control law is as follows, where λ = 1 and $\mathbf{A}$ and $\mathbf{B}$ are derived from equation (13):

$$\dot{\mathbf{q}}_\mathbf{a} = -\mathbf{B}^{-1}\mathbf{A}(\dot{\mathbf{x}}_\mathbf{t} - \lambda\mathbf{e}) \tag{49}$$

This then feeds into the the ADAMS block in order to get the active joint positions $\mathbf{q}_\mathbf{a}$. This is then fed into the DGM to get the end-effector position $\mathbf{x}$. This end-effector position is then compared to the trajectory to see how accurate the control law is.

C. Computed Torque Controller

The simulink model of the computed torque controller is shown below:

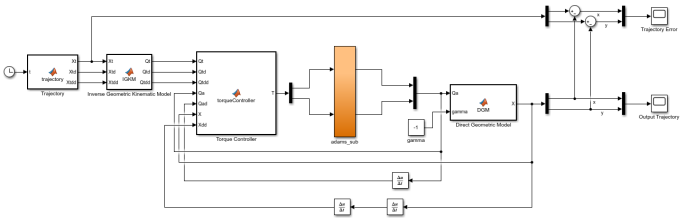

Fig. 5. Simulink model for the Computed Torque Controller. Source: Own elaboration.

In this case, the goal is to create a control law which finds a force where the error, defined in equation (48), goes to 0. This will be done through simulink with ADAMS co simulation. The trajectory is the same as the previous simulink model for simplicity, but it also calculates the desired

velocity and acceleration of the trajectory. The next block is the Inverse Geometric and Kinematic Model (IGKM) which outputs the joint positions, velocities, and accelerations. The IGKM implements the following equations:

$$\begin{aligned}\mathbf{q_a} &= \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -\sqrt{l^2 - (x + \frac{d}{2})^2} + y \\ -\sqrt{l^2 - (x - \frac{d}{2})^2} + y \end{bmatrix} \\ \dot{\mathbf{q}}_\mathbf{a} &= \mathbf{B}^{-1}(\mathbf{A}\dot{\mathbf{x}}) \\ \ddot{\mathbf{q}}_\mathbf{a} &= \mathbf{B}^{-1}(-\mathbf{A}\ddot{\mathbf{x}} - \dot{\mathbf{A}}\dot{\mathbf{x}} - \dot{\mathbf{B}}\dot{\mathbf{q}}_\mathbf{a})\end{aligned} \tag{50}$$

The control law is based off an effort equation structured in the following way:

$$\tau = \mathbf{M}\ddot{\mathbf{q}}_\mathbf{a} + \mathbf{c} \tag{51}$$

The auxiliary input to the control law is defined as follows assuming the effort is non linear:

$$\begin{aligned}\mathbf{v} &= \mathbf{M}^{-1}(\tau - \mathbf{c}) \\ \mathbf{v} &= \ddot{\mathbf{q}}_\mathbf{a} \\ \mathbf{v} &= \ddot{\mathbf{q}}_\mathbf{t} + K_d(\dot{\mathbf{q}}_\mathbf{t} - \dot{\mathbf{q}}_\mathbf{a}) + K_p(\mathbf{q_t} - \mathbf{q_a})\end{aligned} \tag{52}$$

Combining equations (51) and (52), the following is derived, which is implemented in the controller simulink block:

$$\tau = \mathbf{M}(\ddot{\mathbf{q}}_\mathbf{t} + K_d(\dot{\mathbf{q}}_\mathbf{t} - \dot{\mathbf{q}}_\mathbf{a}) + K_p(\mathbf{q_t} - \mathbf{q_a})) + \mathbf{c} \tag{53}$$

The effort is fed into the ADAMS block, where the active joint positions are outputs. This is then directed into the DGM, which gives the trajectory of the end effector based off the control law. The results of this are compared to the desired trajectory to test the accuracy of the controller.

A trajectory was also calculated that tries to pass through a Type 2 Singularity. This was done by combining two 3rd degree polynomial trajectories together, one which goes from the initial position to the singularity point, and one from the singularity point to the final position. The points are below:

$$\begin{aligned}&\mathbf{x}_i = \begin{bmatrix} 0 \\ 0.3 \end{bmatrix}; \mathbf{x}_f = \begin{bmatrix} 0.15 \\ -1.5 \end{bmatrix}; \mathbf{x}_s = \begin{bmatrix} 0 \\ -0.3 \end{bmatrix} \\ &t_s = 3; t_f = 6\end{aligned} \tag{54}$$

D. Crossing Type 2 Singularity

We have shown the criterion necessary cross the singularity in section II-E. Now we must create a trajectory that respects this criterion. To create a trajectory that passes through a Type 2 Singularity, it must pass through a Type 1 Singularity first. A Type 1 Singularity occurs when one of the arms is horizontal, which is when matrix $\mathbf{B}$ is rank deficient. To do this, we use 3 trajectories using 5th degree polynomial equations:

$$\begin{aligned}\mathbf{x}_t(t) &= a_{x0} + a_{x1}t + a_{x2}t^2 + a_{x3}t^3 + a_{x4}t^4 + a_{x5}t^5 \\ \mathbf{y}_t(t) &= a_{y0} + a_{y1}t + a_{y2}t^2 + a_{y3}t^3 + a_{y4}t^4 + a_{y5}t^5\end{aligned} \tag{55}$$

The first trajectory goes from the initial position ($\mathbf{x_i}$) to the Type 1 Singularity position ($\mathbf{x_m}$). The second trajectory goes from $\mathbf{x_m}$ to the Type 2 Singularity position ($\mathbf{x_s}$). The final trajectory goes from $\mathbf{x_s}$ to the final position ($\mathbf{x_f}$). By using a fifth degree polynomial for the trajectories, it ensure we can impose acceleration and velocity constraints at the start and end points, which is useful for passing through the Type 2 Singularity. The points chosen are shown below:

$$\begin{aligned}\mathbf{x}_i &= \begin{bmatrix} 0 \\ 0.3 \end{bmatrix} \dot{\mathbf{x}}_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ddot{\mathbf{x}}_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \mathbf{x}_m &= \begin{bmatrix} 0.1606 \\ 0 \end{bmatrix} \dot{\mathbf{x}}_m = \begin{bmatrix} 0 \\ -0.1 \end{bmatrix} \ddot{\mathbf{x}}_m = \begin{bmatrix} 0 \\ -0.2 \end{bmatrix} \\ \mathbf{x}_s &= \begin{bmatrix} 0 \\ -0.3 \end{bmatrix} \dot{\mathbf{x}}_s = \begin{bmatrix} \frac{-0.1d}{\sqrt{4l^2 - d^2}} \\ -0.1 \end{bmatrix} \ddot{\mathbf{x}}_s = \begin{bmatrix} \frac{-0.1d}{\sqrt{4l^2 - d^2}} \\ -0.1 \end{bmatrix} \\ \mathbf{x}_f &= \begin{bmatrix} -0.1 \\ -0.15 \end{bmatrix} \dot{\mathbf{x}}_f = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ddot{\mathbf{x}}_f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\end{aligned} \tag{56}$$

To find the coefficients for each trajectory, we solve the following matrix for both x and y for each one:

$$\begin{bmatrix} x_i \\ \dot{x_i} \\ \ddot{x_i} \\ x_f \\ \dot{x_f} \\ \ddot{x_f} \end{bmatrix} \begin{bmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5 \\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & 5t_i^4 \\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{57}$$

where $x_i, \dot{x_i}, \ddot{x_i}, x_f, \dot{x_f}, \ddot{x_f}$ are the position, velocity, and acceleration of the initial and final points of the trajectories respectively and t_i, t_f are the initial and start times of the trajectories. Once each trajectory was found, they were combined to create one trajectory.

Another trajectory was created that did not respect the criterion. All the constraints were the same as the previous trajectory except for the following:

$$\ddot{\mathbf{x}}_s = \begin{bmatrix} \frac{-\pi}{30} \\ \frac{-\sqrt{2}}{20} \end{bmatrix} \tag{58}$$

Once the trajectory is created, the active joint values $\mathbf{q}_a$ are found for each point of the trajectory using the IGM and saved into a text file with its associated time as well. This is then imported into ADAMS in order to simulate it through the Spline/Akima fitting method.

IV. Explanation steps for using the simulation

A. Setup

- The following instructions were last tested using Windows 10, MATLAB 2019b and ADAMS 2020 on November 7th, 2020.
- Clone the repository of the project in a desired folder. See repository
- Open MATLAB and Simulink and be sure to change the working directory to the folder that has the files cloned.

B. Geometric, Kinematic and Dynamic Model

- For testing the Geometric, Kinematic and Dynamic model, run the MATLAB file called "Controls_Plant_1" from the MATLAB command window and open the Simulink file by running "Biglide_Lee_Palma".
- Set the simulation time to the desired value and start the Simulink simulation by pressing the green play button.
- The results will be generated and they can be seen by double clicking the scopes elements. The graphs shown in the scopes are actually the difference between the values computed by ADAMS and the ones computed by MATLAB.

C. Trajectory Tracking by Kinematic Control

- For testing the Kinematic Control, run the MATLAB file called "Controls_Plant_3" from the MATLAB command window and the open the Simulink file by running "Trajectory_Tracking".
- Set the parameters of the desired trajectory in the elements on the left side of the scheme and before pressing the green play button, make sure that the simulation time is equal to the final time of the trajectory generator (t_f).
- After running the simulation, the results can be seen in the scopes. In these graphs, the desired trajectory generated and the trajectory followed by the end effector are shown, as well as the error between these.
- You can now try to change the value of the gain called "lambda" to experiment its effects on the system.

D. Trajectory Tracking by Dynamic Control

- For testing the Dynamic Control, run the MATLAB file called "Controls_Plant_5" from the MATLAB command window and the open the Simulink file by running "Torque_Control".
- Set the parameters of the desired trajectory in the elements on the left side of the scheme and before pressing the green play button, make sure that the simulation time is equal to the final time of the trajectory generator (t_f).
- After running the simulation, the results can be seen in the scopes. In these graphs, the desired trajectory generated and the trajectory followed by the end effector are shown, as well as the error between these.
- You can now try to change the value of the gains called "K_p" and "K_d" by double clicking the "Torque Controller" MATLAB function block and changing their numerical value.

E. Singularity Crossing

- For testing the Singularity Crossing using MATLAB, run the MATLAB file called "singularity_crossing" directly from the command window and watch the simulation in the figure generated.
- To test the Singularity Crossing using ADAMS, open the file called "Biglide_singularity.bin" in ADAMS. Simply run the simulation in ADAMS by pressing the play button. Ensure that the time of the simulation is 6 seconds and the step size is 3001. Watch the interactive simulation as it happens.

V. Results and Analysis

A. Geometric Analysis

The results from the geometric model correspond to the planar coordinates of the end-effector ($\mathbf{x}$) and the passive joints positions ($\mathbf{q}_d$) estimated by the functions and Simulink developed. The planar coordinates from the ADAMS block are assumed to be correct. Therefore, the output of the simulink system should be the same as the ADAMS block. This is confirmed by Fig.6 where it is shown that the difference between both simulations is less than 0.06 mm. This shows that the geometric model is correct.

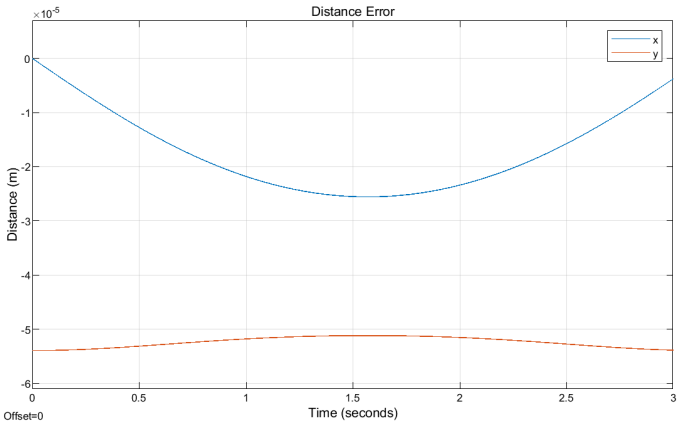

Fig. 6. Difference between the ADAMS and Simulink position of the end effector. Source: Own elaboration.

One source of the error can be attributed to the fact that the ADAMS and Simulink software use different mathematical methods to solve equations. These methods have a different balance between the precision and the speed of computations. Another possible source of error is the communication interval between the Simulink model and the ADAMS internal block. The ADAMS block works by communicating in a discrete and periodic manner between the inputs and outputs of its computations. This can lead to errors since there is a discrete simulation component in a continuous time simulation system. A third possible error source is the fact that both software could be using different data types for each variable, leading to data trim and loss of accuracy. That would happen for example, if MATLAB stores a non integer number in a double data type while ADAMS uses a float data type.

With respect to the passive joints, the same occurrence happens. Fig.7 shows that the error obtained between the ADAMS and Simulink simulation of the passive joints values is less than 0.1 mrad. This means that the Geometric Model is correct. The sources of errors are the sames as before:the computational methods used to solve the mathematical expressions, the data types used by each software and the communication interval in the ADAMS block.

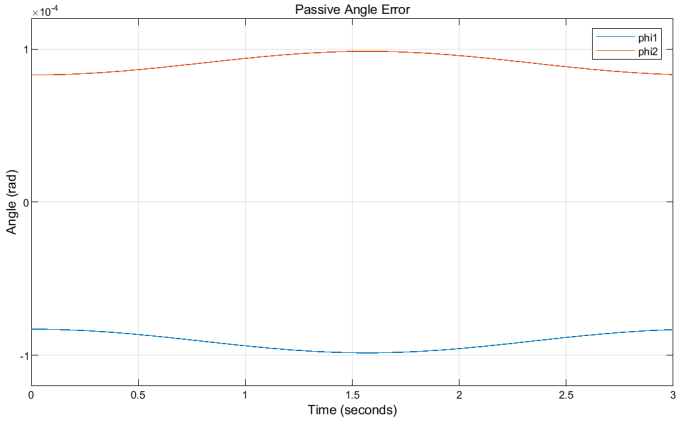

Fig. 7. Difference between the ADAMS and Simulink position of the passive joints. Source: Own elaboration.

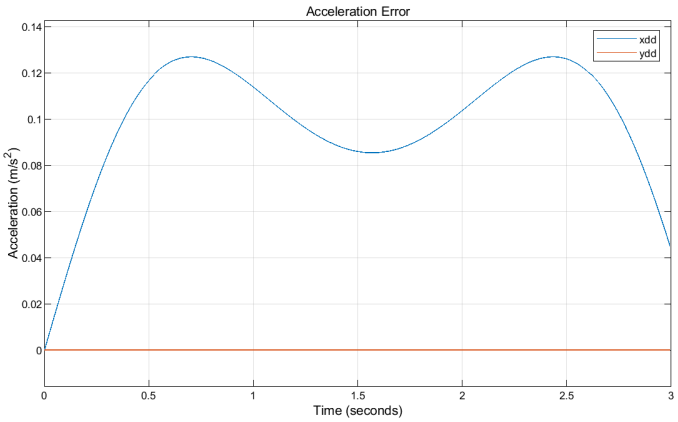

Fig. 9. Difference between the ADAMS and Simulink acceleration of the end effector. Source: Own elaboration.

B. Kinematic Analysis

As the active joints change their positions, not only must a continuous movement of the end effector and the passive joints happen, but a velocity and an acceleration are expected. This velocity and acceleration of both the end effector $(\dot{\mathbf{x}}, \ddot{\mathbf{x}})$ and passive joints $(\dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$ are the results from the kinematic model.

Similar to the analysis of the past subsection, both the ADAMS and Simulink simulations are expected to compute the same results. In Fig.8 and Fig.9 the error for $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ is shown. Notice that for the velocity the error is less than 0.03 mm/s while for the acceleration the error is less than 0.14 mm/s^2. On the other hand, in the Fig.10 and Fig.11 the error for $\dot{\mathbf{q}}_d$ and $\ddot{\mathbf{q}}_d$, respectively, is shown. Notice that the velocity error is less than 0.02 mm/s while for the acceleration the error is less than 0.04 mm/s^2. Also, noticed that the graphs show a periodic behaviour, that makes sense since the active joints positions also move in a periodic way. The magnitude of theses errors leads to think that the Simulink simulation and the equations here developed are correct.

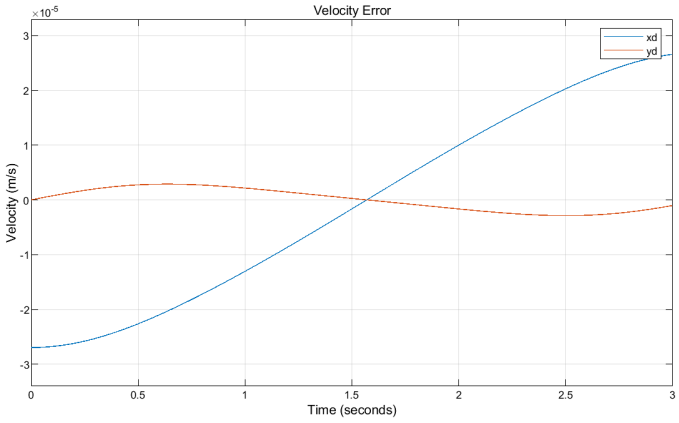

Fig. 8. Difference between the ADAMS and Simulink velocity of the end effector. Source: Own elaboration.

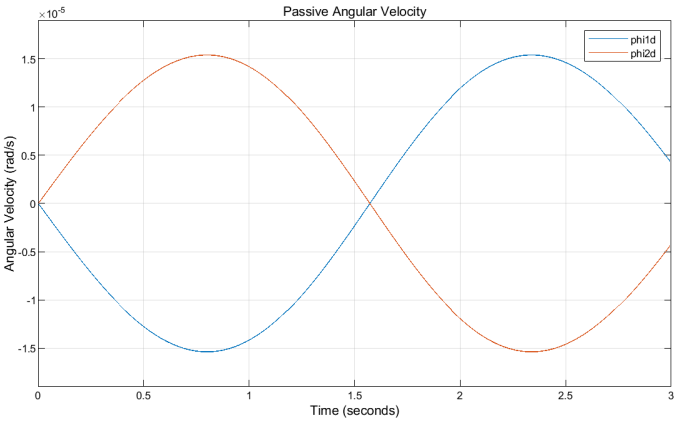

Fig. 10. Difference between the ADAMS and Simulink velocity of passive joints. Source: Own elaboration.

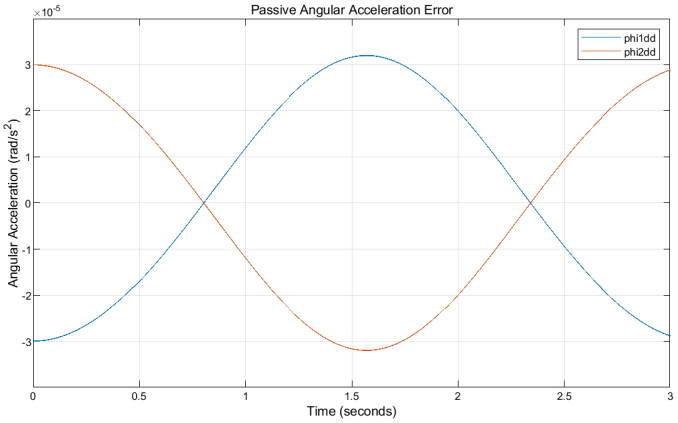

Fig. 11. Difference between the ADAMS and Simulink acceleration of the passive joints. Source: Own elaboration.

Same as the Geometric Analysis, the sources of these errors are attributed mostly to the fact that computational methods used to solve the mathematical expressions and data types used are different for each software and the discrete communication interval in the ADAMS block. Also, there is the possibility of error propagation. Since the computation of the velocities and accelerations depends on the values of the position, if the position wasn't exactly the same for both simulations, then it is very likely that the computation of the velocity and acceleration not only will induce new errors but it will also keep the previous.

C. Dynamic Analysis

The results from the dynamic model correspond to the effort (τ) applied onto the active joints. For this analysis, the feet and end effector of the robotic platform now are considered to have a mass. Since both of them are moving with a certain acceleration ($\ddot{\mathbf{q}}_a$ and $\ddot{\mathbf{x}}$, respectively) then an effort in the active joints must exist to produce the motion. In Fig.12 the difference between the effort computed by the MATLAB and the same computed by the ADAMS simulation is shown.

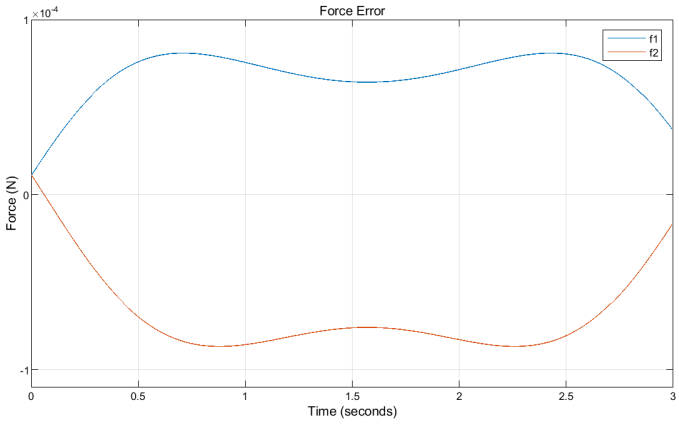

Fig. 12. Difference between the ADAMS and Simulink effort in the active joints. Source: Own elaboration.

The effort applied to the active joints are only force components and not moments. This makes sense since the active joints are prismatic and can't input a torque to the robotic platform. Similar to the geometric and kinematic analysis, the error obtained in the Fig.12 is less than 0.1 mN, which means that the equations and simulation developed in MATLAB are correct, with the same sources of errors as before.

D. Kinematic Control Analysis

The x and y coordinates of the desired trajectory are shown in Fig.4, and the ones obtained in Fig. 13 with the parameters shown in equation (47).

The error of the trajectory is shown in Fig. 14. This was generated by subtracting the generated trajectory from the output from the ADAMS block. It can be seen that the error is quite small, as the magnitude of the values are to the ten-thousandth. This means that the output trajectory matches

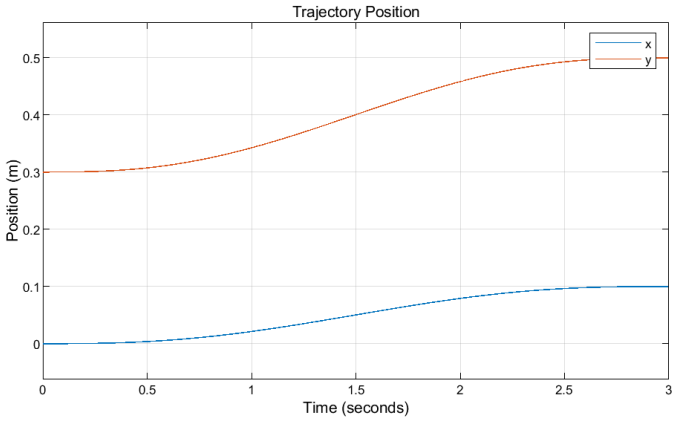

Fig. 13. Output end effector trajectory obtained with a kinematic control. Source: Own elaboration.

the generated trajectory quite closely, which implies that the control law is working quite well.

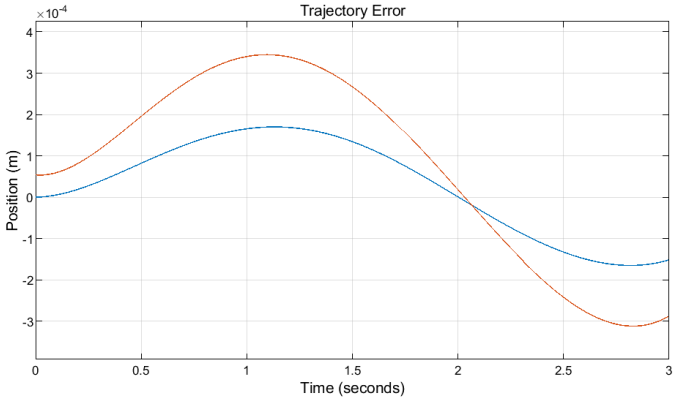

Fig. 14. Output end effector trajectory error obtained with a kinematic control. Source: Own elaboration.

Now an experiment is conducted where the communication interval of the ADAMS block is changed to see what happens to the error. For reference, the communication interval used above is 0.01. In Fig. 16, the communication interval is changed to 0.1, so the communication interval has become larger. It can be seen that the the output trajectory now has large flat lines. This is due to the large communication interval, so there are larger periods of time where no new information is being fed to the output trajectory, leading to the flat lines. This causes the error (Fig. 16) to have oscillations (of small amplitude), as the desired trajectory uses a continuous clock which means it is continuously updating its value.

E. Computed Torque Control Analysis

Same as for the Kinematic Control Analysis, the curve from Fig.4 is used as the desired trajectory to be followed by the end effector, while in Fig.17 the output trajectory from the Computer Torque Controller implemented (Fig.5) is shown, being the main difference that now it was obtained by only

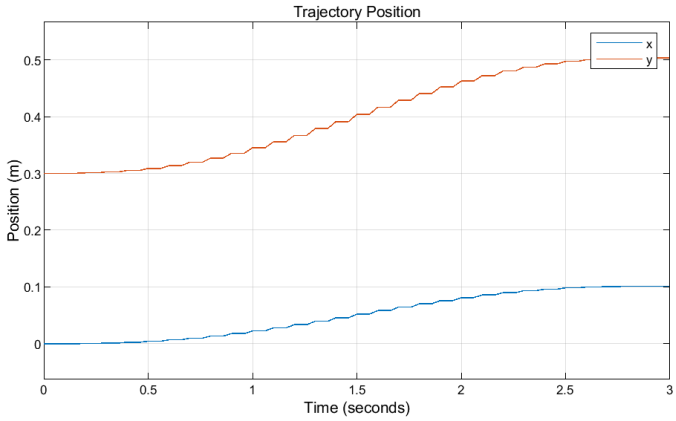

Fig. 15. Output trajectory when communication interval time is 0.1. Source: Own elaboration.

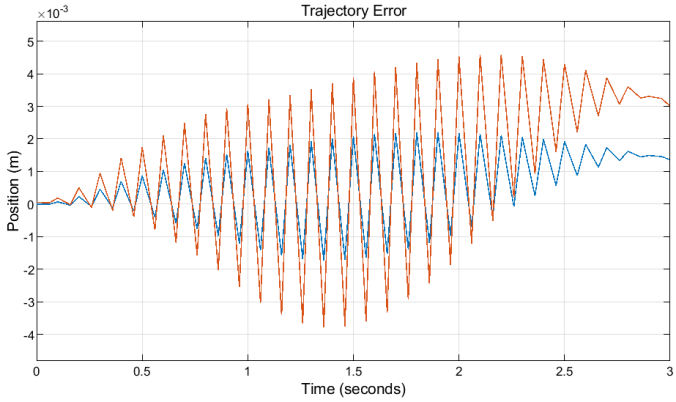

Fig. 16. Trajectory error when communication interval time is 0.1. Source: Own elaboration.

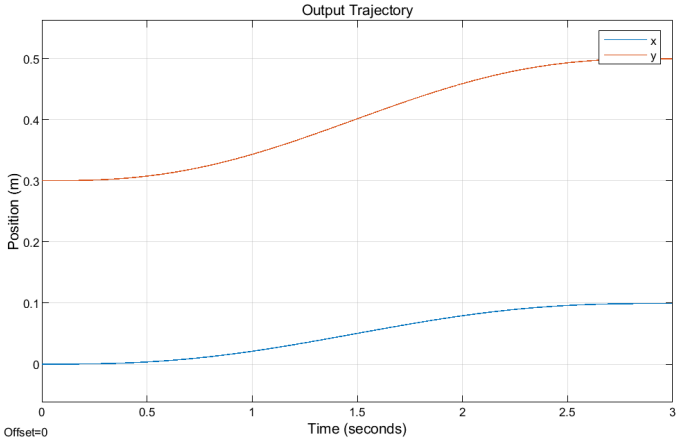

Fig. 17. Output end effector trajectory obtained with a torque control. Source: Own elaboration.

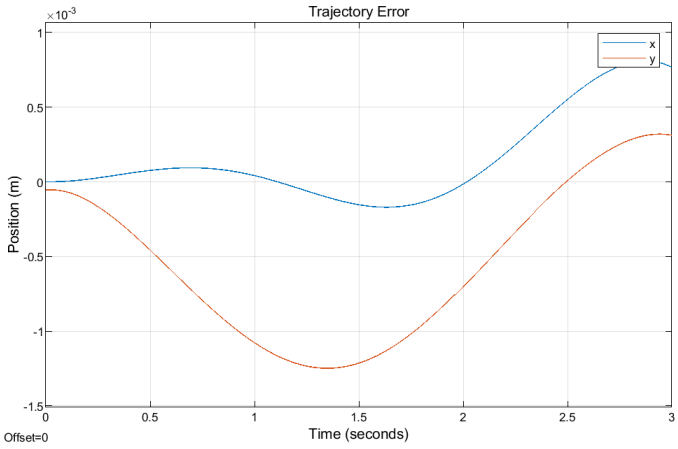

Fig. 18. Output end effector trajectory error obtained with a torque control. Source: Own elaboration.

feeding values of τ. This graph was obtained by setting the two control variables to 100.

The difference between the desired and obtained trajectory is shown in Fig.18. It can be noticed that the resulted error is less than 1mm, which means that the control law is working properly. However, the error was inversely proportional to the value of the control gains K_d and K_p. What this means is that for the robotic platform modeled, the desired acceleration isn't enough to control the system and therefore the velocity and position error should have a bigger effect in the computation of the input τ.

When trying to use the Torque Controller to pass through a Type 2 Singularity, the model fails. When the model gets to the singularity position, complex values show up in the DGM, which causes the simulation to crash. This speaks to the difficulty of modeling and simulating around singularity positions.

F. Singularity Crossing Analysis

First of all, we will examine the results of simulating a trajectory which respected the criterion necessary to cross the Type 2 Singularity. In ADAMS, it simulated everything perfectly before the singularity position. After the singularity position however, it became very jerky and appeared to switch between working modes (The sign of γ was being flipped constantly). At the singularity point, it can be seen in Fig. 21 that the input torques required to follow this trajectory become incredibly high. It can be shown in Fig. 20 that even after it passes the singularity, the torque becomes very erratic. This phenomenon doesn't occur in the MATLAB simulation of the singularity crossing, since the value of gamma is precisely changed after the singularity pose is reached, hence, it works.

Now looking at the trajectory which did not respect the criterion, similar results are shown. The robot behaves in the same way, where after it passes the singularity point it appears to flip between working modes. The input torques required for the motions (Fig. 21 and 22) both look similar to Fig. 19 and 20. There is a massive force spike at the singularity point, and it becomes very erratic after it.

All of this leads us to the conclusion that ADAMS is not capable of accurately modeling the behaviour of the Biglide robot around a Type 2 Singularity or that the configuration

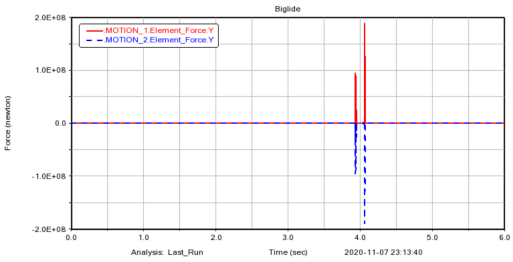

Fig. 19. Input Force required to control trajectory respecting criterion. Source: Own elaboration.

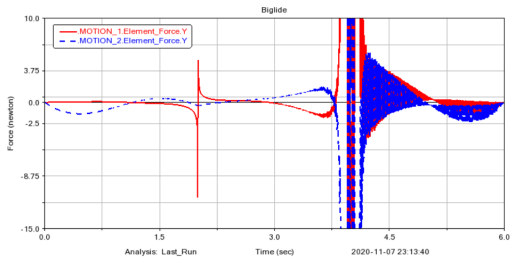

Fig. 20. Same as Fig. 19 but zoomed in. Source: Own elaboration.

chose in ADAMS wasn't the right one.

VI. Conclusions

- Given the magnitude of the errors obtained, it can be concluded that the Geometric, Kinematic and Dynamic model were correctly computed and implemented at a theoretical and simulation level.
- The sources of errors are most likely attributed first to the fact that the computational methods used to solve the mathematical expressions are different between the software, leading to loss of accuracy. Second, to the fact that the software use different data types for the variables, leading to data trim. Third, to the fact that the ADAMS Simulink block has a discrete communication interval while the signals in Simulink are continuous, this leads to an amplitude quantization type of error. And finally, error propagation since most of the values computed depend on error prone inputs.
- Given the magnitude of the errors obtained, it can be concluded that the Kinematic and Torque Control were correctly computed and implemented at a theoretical and simulation level.
- For the two controls implemented, special attention should be given to the communication interval used in the ADAMS block, since if it is slow, the outputs will flatten up. Also, it is important to notice that both controls fail when the robot approaches the singularity conditions, this is expected since the Geometric and Kinematic model fall down at this moment.
- Two trajectories were tested to try and cross a Type 2 Singularity, one respecting the criterion necessary for crossing it while the other did not. In both cases, the ADAMS simulation failed to model it properly as the behaviour became very erratic after the singularity and magnitude of the torques became incredibly high. This is due to some of the limitations in the ADAMS model, namely that it is not able to switch between working modes smoothly. While in the MATLAB simulation, since the value of gamma is precisely changed, the robot platform was able to cross the Type 2 singularity successfully.

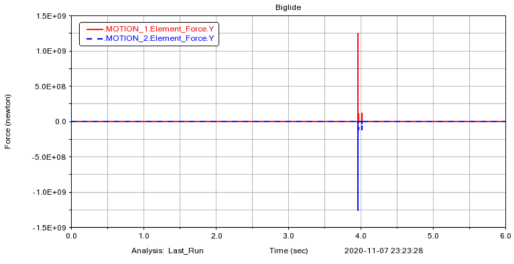

Fig. 21. Input Force required to control trajectory not respecting criterion. Source: Own elaboration.

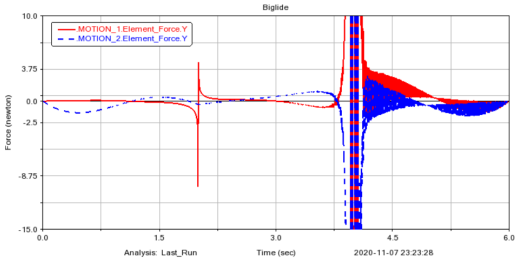

Fig. 22. Same as Fig. 21 but zoomed in. Source: Own elaboration.

References

[1] Jeanneau, G. and Uday-Nayak, A. (2018). Lab 4: Kinematics and Dynamics of a Biglide. Retrieved from: https://hippocampus.ec-nantes.fr/mod/folder/view.php?id=12398

[2] Briot,S. (2018). Advanced Dynamic Modelling of Robots - Lecture 4: Dynamics of parallel robots. Retrieved from: http://pagesperso.ls2n.fr/~briot-s/