École Centrale de Nantes
Advanced Modeling of Robots - 2020/2021

LAB 4 REPORT

Kinematics and Dynamics of a Biglide

Marco Demutti
Emanuele-Riccardo Rosi

November 08, 2020

ABSTRACT

The geometric, kinematic and dynamic models of the Biglide robot were modeled in the MATLAB environment and compared to the results obtained with ADAMS. It was possible to observe that the error between the two simulations was always negligible when compared to the absolute value of the corresponding quantity. Moreover two different control schemes were implemented on SIMULINK and used to perform trajectory tracking through ADAMS co-simulation. The first one was based on a Kinematic Control Law, while the second one applied a Computed Torque Control law: both schemes showed a good convergence behavior for the tracking error, except when the robot was approaching a Type 2 singularity. This latter issue was overcome by designing a trajectory that respected the criterion necessary to cross such singularity.

Contents

Chapter 1

Introduction

The aim of this report is first to perform the geometric, kinematic and dynamic analyses of a Biglide mechanism, by implementing the corresponding models in MATLAB and comparing the results with the ones obtained with ADAMS.
Subsequently, a Kinematic Control and a Computed Torque Control schemes are implemented and tested to track a desired trajectory, and the problem of crossing a Type 2 singularity is addressed.
Finally, the last chapter contains detailed information on how the files are organized and named, along with some instructions on how to run the different simulations.

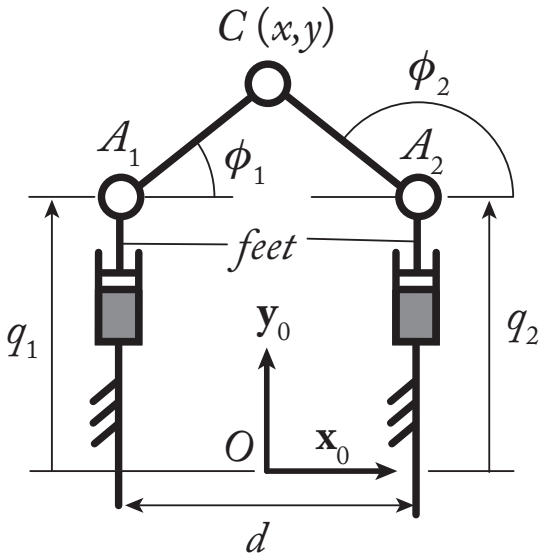

Figure 1.1: Kinematic model of the Biglide

The Biglide robot used in this lab is a 2-dof parallel robot with two actuated prismatic joints and two passive rotational joints.
Its kinematic architecture is shown in Figure 1.1. In the provided ADAMS mock-up the geometric parameters are:

- $d = 0.4$ m
- $l_{A_1C} = 0.3606$ m
- $l_{A_2C} = 0.3606$ m

while the base dynamic parameters are:

- $m_p = 3$ kg, mass of the end effector
- $m_f = 1$ kg, mass of each foot.

All other dynamic parameters are neglected.

Chapter 2

Models and analyses

In this chapter the kinematic and dynamic analyses are carried out on the Biglide mechanism: in the following sections, each model is computed and implemented as a MATLAB function. Then, the outputs of each function are compared with the corresponding outputs coming from the ADAMS simulation and the error between the two is shown and commented. Figure 2.1 shows how the procedure described above was carried out on SIMULINK for the DGM, but the same structure was used for all the models involved in order to obtain the overall scheme.

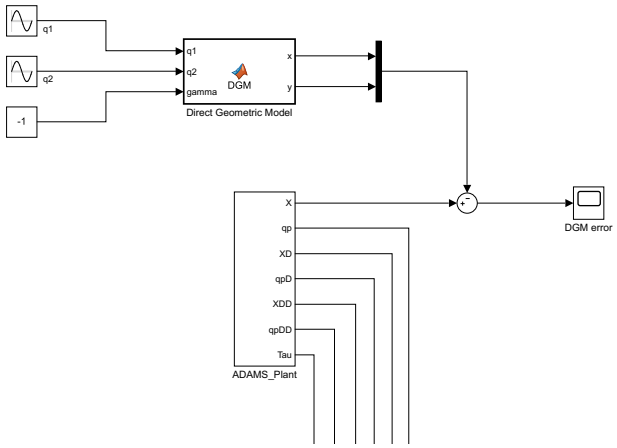

Figure 2.1: Error on the DGM - SIMULINK scheme

2.1 Geometric Analysis

2.1.1 Direct Geometric Model (DGM)

The Direct Geometric Model provides the location (x, y) of the end effector as a function of the active joint variables, namely the values of the two actuated prismatic joints (q_1, q_2):

$$[x, y] = DGM(q_1, q_2, \gamma)$$

where $\gamma = \pm 1$. In order to obtain a DGM such that each value of γ corresponds exactly to one of the two different *working modes* of the robot, the solution is computed as:

$$\overrightarrow{OC} = \begin{bmatrix} x \\ y \end{bmatrix} = \overrightarrow{OA_2} + \overrightarrow{A_2H} + \overrightarrow{HC} \tag{2.1}$$

where H is the mid-point of $\overrightarrow{A_1A_2}$ (Figure 2.2).

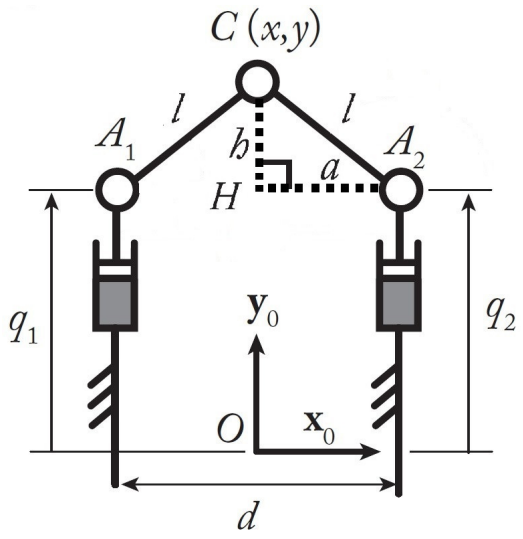

Figure 2.2: Kinematic model of the Biglide

The 3 vectors in (2.1) can be explicitly computed by using the following procedure:

- $\overrightarrow{OA_1} = -\frac{d}{2}\mathbf{x}_0 + q_1\mathbf{y}_0$

- $\overrightarrow{OA_2} = \frac{d}{2}\mathbf{x}_0 + q_2\mathbf{y}_0$
- $\overrightarrow{A_2A_1} = \overrightarrow{OA_1} - \overrightarrow{OA_2} = -d\mathbf{x}_0 + (q_1 - q_2)\mathbf{y}_0$
- $\overrightarrow{A_2H} = \frac{\overrightarrow{A_2A_1}}{2} = -\frac{d}{2}\mathbf{x}_0 + \frac{(q_1 - q_2)}{2}\mathbf{y}_0$
  knowing that H is the mid-point of $\overrightarrow{A_1A_2}$
- There are two solutions for the point C, obtained by the 90° rotation of the vector $\overrightarrow{A_2H}$, normalization and multiplying by the distance h, giving:
$$\overrightarrow{HC} = \gamma\frac{h}{a}\mathbf{E}\overrightarrow{A_2H}$$
  where:
  - $a = ||\overrightarrow{A_2H}||$
  - $h = \sqrt{l^2 - a^2}$
  - $\mathbf{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ represents the 90° rotation matrix

By replacing all the known quantities, (2.1) can be explicitly written as:

$$\overrightarrow{OC} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{d}{2} \\ q_2 \end{bmatrix} + \begin{bmatrix} -\frac{d}{2} \\ \frac{q_1 - q_2}{2} \end{bmatrix} + \gamma\frac{h}{a}\mathbf{E}\overrightarrow{A_2H} \tag{2.2}$$

Analysis

Figure 2.3 shows the comparison between the results of the outputs (x, y) obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on x, while the blue one represents the error on y.
We can observe that both errors have values around 10^{-5} m. It can be considered as a satisfactory result, given that the geometric parameters of the robot, as well as the absolute values of (x, y) in the simulation, are in the order of 10^{-1} m, hence the percent error is around 0.01%.

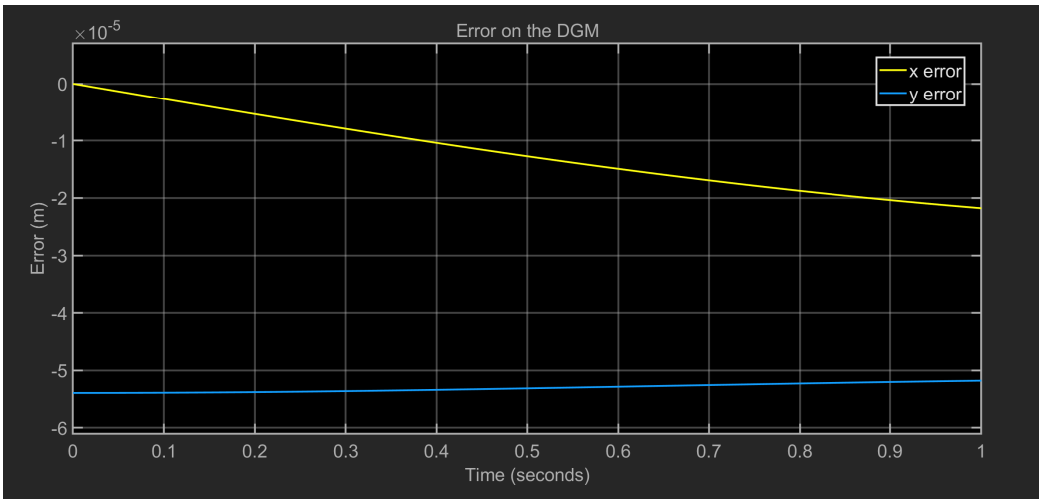

Figure 2.3: Error on DGM

2.1.2 Inverse Geometric Model (IGM)

The Inverse Geometric Model provides the active joint variables, namely the values of the two actuated prismatic joints (q_1, q_2), as a function of the location (x, y) of the end effector:

$$[q_1, q_2] = IGM(x, y, \gamma_1, \gamma_2)$$

where $\gamma_1, \gamma_2 = \pm 1$ are used to distinguish the 4 different solutions (*working modes*).
The expression of the IGM can be computed starting from the **loop closure equations** of the two legs:

$$\overrightarrow{OC} = \begin{bmatrix} x \\ y \end{bmatrix} = \overrightarrow{OA_i} + \overrightarrow{A_iC} \qquad i = 1, 2$$

which leads to the following equations for leg 1:

$$\begin{bmatrix} x + \frac{d}{2} \\ y - q_1 \end{bmatrix} = l \begin{bmatrix} \cos\phi_1 \\ \sin\phi_1 \end{bmatrix} \tag{2.3}$$

and similarly for leg 2:

$$\begin{bmatrix} x - \frac{d}{2} \\ y - q_2 \end{bmatrix} = l \begin{bmatrix} \cos\phi_2 \\ \sin\phi_2 \end{bmatrix} \tag{2.4}$$

Then, squaring and summing both sides of the two lines of (2.3) and (2.4) we obtain:

$$\mathbf{h(x,q}_a) = \begin{bmatrix} \left(x + \frac{d}{2}\right)^2 + (y - q_1)^2 - l^2 \\ \left(x - \frac{d}{2}\right)^2 + (y - q_2)^2 - l^2 \end{bmatrix} = \mathbf{0} \tag{2.5}$$

from which we can find:

$$q_1 = y \pm \sqrt{l^2 - \left(x + \frac{d}{2}\right)^2} = y + \gamma_1 \sqrt{l^2 - \left(x + \frac{d}{2}\right)^2} \tag{2.6}$$

$$q_2 = y \pm \sqrt{l^2 - \left(x - \frac{d}{2}\right)^2} = y + \gamma_2 \sqrt{l^2 - \left(x - \frac{d}{2}\right)^2} \tag{2.7}$$

where, as stated above, the 4 solutions of the IGM (2 solutions for q_1, two solutions for q_2) are determined by using γ_1, γ_2.
The IGM was successfully tested with the DGM to check its validity.

2.1.3 Passive angle positions

This model provides the passive angle positions (ϕ_1, ϕ_2) as functions of the two actuated prismatic joints (q_1, q_2) and the end effector position (x, y):

$$[\phi_1, \phi_2] = passive\_angles(q_1, q_2, x, y)$$

Each passive angle position can be obtained by (2.3) and (2.4) by dividing the second line by the first one:

$$\phi_1 = atan2\left(y - q_1, x + \frac{d}{2}\right) \tag{2.8}$$

$$\phi_2 = atan2\left(y - q_2, x - \frac{d}{2}\right) \tag{2.9}$$

Analysis

Figure 2.4 shows the comparison between the results of the outputs (ϕ_1, ϕ_2) obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on ϕ_1, while the blue one represents the error on ϕ_2. We can observe that both errors have values around 10^{-4} rad. It can be considered as a satisfactory result, given that the absolute values of the passive angles in the simulation are in the order of 1 rad, hence the percent error is around 0.01%.

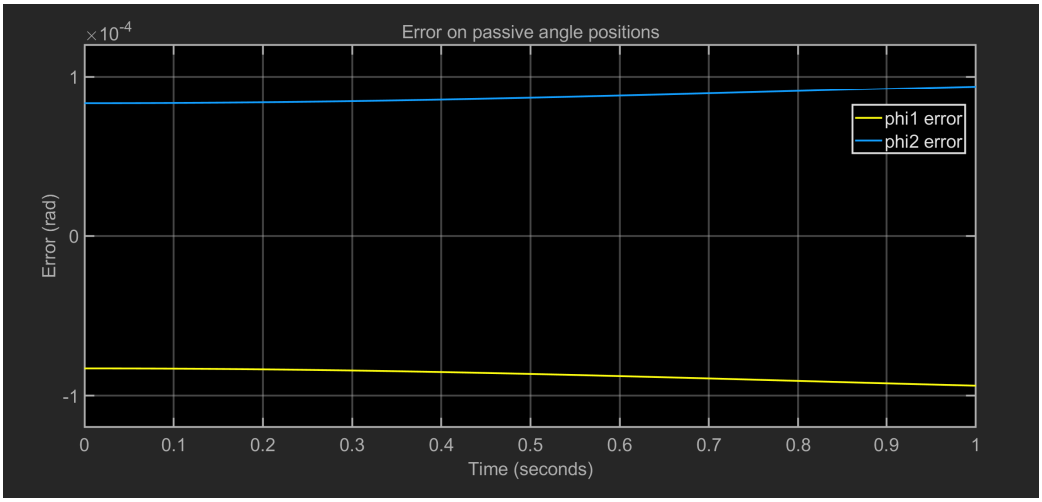

Figure 2.4: Error on passive angle positions

2.2 Velocity Analysis

2.2.1 Direct Kinematic Model (DKM)

The Direct Kinematic Model allows to express the end effector velocity $(\dot{x}, \dot{y})$ as a function of the passive joint coordinates (ϕ_1, ϕ_2) and the active joint velocities $(\dot{q_1}, \dot{q_2})$:

$$[\dot{x}, \dot{y}] = DKM(\phi_1, \phi_2, \dot{q_1}, \dot{q_2})$$

Defining $\mathbf{P} = [x, y]^T$, equations (2.3) and (2.4) can be rewritten in vector form as in the following system:

$$\begin{cases} \mathbf{P} = -\dfrac{d}{2}\mathbf{x}_0 + q_1\mathbf{y}_0 + l\mathbf{u}_1 \\ \mathbf{P} = \dfrac{d}{2}\mathbf{x}_0 + q_2\mathbf{y}_0 + l\mathbf{u}_2 \end{cases} \tag{2.10}$$

where $\mathbf{u}_i$ is the unit vector along $\overrightarrow{A_iC}$:

$$\mathbf{u}_i = \begin{bmatrix} \cos\phi_i \\ \sin\phi_i \end{bmatrix} \qquad i = 1, 2$$

By computing the time derivative of equations (2.10) we can obtain:

$$\begin{cases} \dot{\mathbf{P}} = \dot{q_1}\mathbf{y}_0 + l\dot{\phi_1}\mathbf{E}\mathbf{u}_1 \\ \dot{\mathbf{P}} = \dot{q_2}\mathbf{y}_0 + l\dot{\phi_2}\mathbf{E}\mathbf{u}_2 \end{cases} \tag{2.11}$$

where $\mathbf{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ represents the 90° rotation matrix.
Then, in order to lose the dependency of $\dot{\mathbf{P}}$ on $(\dot{\phi_1}, \dot{\phi_2})$, equations in (2.11) are premultiplied by $\mathbf{u}_1^T$ and $\mathbf{u}_2^T$ respectively.
In matrix form, given that $\mathbf{u}_i^T \mathbf{y}_0 = \sin\phi_i$, they become:

$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix} \dot{\mathbf{P}} = \begin{bmatrix} \sin\phi_1 & 0 \\ 0 & \sin\phi_2 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{2.12}$$

which can be written in the form:

$$\mathbf{A}\dot{\mathbf{P}} + \mathbf{B}\dot{\mathbf{q}}_a = 0 \tag{2.13}$$

where:

- $\dot{\mathbf{q}}_a = [\dot{q_1}, \dot{q_2}]^T$
- $\mathbf{A} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix}$
- $\mathbf{B} = -\begin{bmatrix} \sin\phi_1 & 0 \\ 0 & \sin\phi_2 \end{bmatrix}$

If $det(\mathbf{A}) \neq 0$, i.e. the robot is not in a parallel singularity, we obtain:

$$\dot{\mathbf{P}} = -\mathbf{A}^{-1}\mathbf{B}\dot{\mathbf{q}}_a = \mathbf{J}\dot{\mathbf{q}}_a \tag{2.14}$$

where $\mathbf{J}$ is the kinematic Jacobian matrix of the robot.

Analysis

Figure 2.5 shows the comparison between the results of the outputs $(\dot{x}, \dot{y})$ obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on $\dot{x}$, while the blue one represents the error on $\dot{y}$.
We can observe that the errors have values around 10^{-5} and 10^{-6} m/s respectively. It can be considered as a satisfactory result, given that the absolute values of the end effector velocity $(\dot{x}, \dot{y})$ in the simulation are in the order of 10^{-1} and 10^{-2} m/s respectively. Hence, in both cases the percent error is around 0.01%.

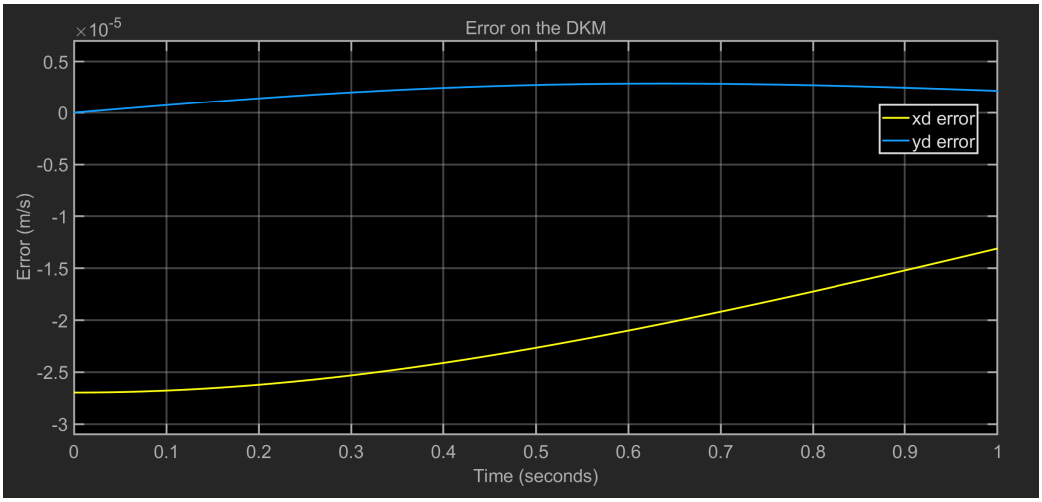

Figure 2.5: Error on DKM

2.2.2 Passive angle velocities

This model gives the passive angle velocities $(\dot{\phi}_1, \dot{\phi}_2)$ as a function of the end effector velocities $(\dot{x}, \dot{y})$, the active joint velocities $(\dot{q}_1, \dot{q}_2)$ and the passive angles (ϕ_1, ϕ_2):

$$[\dot{\phi}_1, \dot{\phi}_2] = passive\_angle\_velocities(\dot{x}, \dot{y}, \dot{q}_1, \dot{q}_2, \phi_1, \phi_2)$$

Premultiplying the equations (2.11) by $(\mathbf{Eu}_1)^T$ and $(\mathbf{Eu}_2)^T$ respectively, and noting that $(\mathbf{Eu}_1)^T\mathbf{y}_0 = \cos\phi_1$ and $(\mathbf{Eu}_2)^T\mathbf{y}_0 = \cos\phi_2$, we obtain:

$$\begin{bmatrix}(\mathbf{Eu}_1)^T\\(\mathbf{Eu}_2)^T\end{bmatrix}\dot{\mathbf{P}} = \begin{bmatrix}\cos\phi_1 & 0\\0 & \cos\phi_2\end{bmatrix}\begin{bmatrix}\dot{q}_1\\\dot{q}_2\end{bmatrix} + \begin{bmatrix}l & 0\\0 & l\end{bmatrix}\begin{bmatrix}\dot{\phi}_1\\\dot{\phi}_2\end{bmatrix} \tag{2.15}$$

Which can be written in the form:

$$\mathbf{J}_t\dot{\mathbf{P}} = \mathbf{J}_{ta}\dot{\mathbf{q}}_a + \mathbf{J}_{td}\dot{\mathbf{q}}_d \tag{2.16}$$

where:

- $\dot{\mathbf{q}}_d = \begin{bmatrix}\dot{\phi}_1\\\dot{\phi}_2\end{bmatrix}$
- $\mathbf{J}_t = \begin{bmatrix}(\mathbf{Eu}_1)^T\\(\mathbf{Eu}_2)^T\end{bmatrix}$
- $\mathbf{J}_{td} = \begin{bmatrix}l & 0\\0 & l\end{bmatrix}$

- $\mathbf{J}_{ta} = \begin{bmatrix} \cos\phi_1 & 0 \\ 0 & \cos\phi_1 \end{bmatrix}$

Finally we can obtain an explicit equation for the passive angle velocities:

$$\dot{\mathbf{q}}_d = \mathbf{J}_{td}^{-1}(\mathbf{J}_t\dot{\mathbf{P}} - \mathbf{J}_{ta}\dot{\mathbf{q}}_a) \tag{2.17}$$

Analysis

Figure 2.6 shows the comparison between the results of the outputs $(\dot{\phi}_1, \dot{\phi}_2)$ obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on $\dot{\phi}_1$, while the blue one represents the error on $\dot{\phi}_2$. We can observe that both errors have values around 10^{-5} rad/s. It can be considered as a satisfactory result, given that the absolute values of the passive joint velocities $(\dot{\phi}_1, \dot{\phi}_2)$ in the simulation are in the order of 10^{-1} rad/s. Hence, the percent error is around 0.01%.

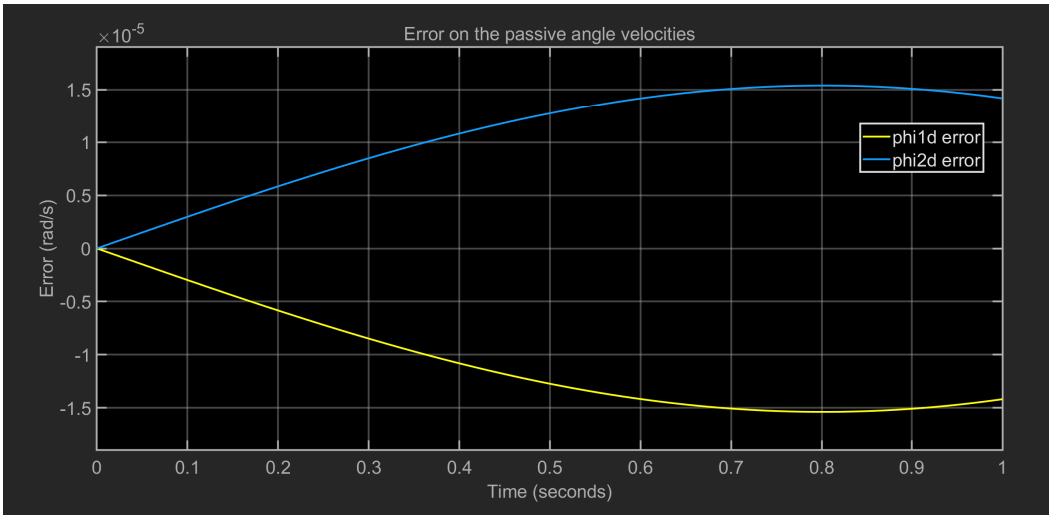

Figure 2.6: Error on passive angle velocities

2.3 Acceleration analysis

2.3.1 Second-order Direct Kinematic Model

The Second-order Direct Kinematic Model provides the accelerations of the end effector $(\ddot{x}, \ddot{y})$ as a function of the active joint accelerations $(\ddot{q}_1, \ddot{q}_2)$, the

passive angles (ϕ_1, ϕ_2) and the passive angle velocities $(\dot{\phi}_1,\ \dot{\phi}_2)$:

$$[\ddot{x}, \ddot{y}] = DKM2(\ddot{q}_1, \ddot{q}_2, \phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2)$$

By computing the time derivative of equations (2.11) we obtain:

$$\begin{cases} \ddot{\mathbf{P}} = \ddot{q}_1 \mathbf{y}_0 + l\ddot{\phi}_1 \mathbf{E}\mathbf{u}_1 - l\dot{\phi}_1^2 \mathbf{u}_1 \\ \ddot{\mathbf{P}} = \ddot{q}_2 \mathbf{y}_0 + l\ddot{\phi}_2 \mathbf{E}\mathbf{u}_2 - l\dot{\phi}_2^2 \mathbf{u}_2 \end{cases} \tag{2.18}$$

Then, in order to lose the dependency of $\ddot{\mathbf{P}}$ on $(\ddot{\phi}_1, \ddot{\phi}_2)$, equations in (2.18) are premultiplied by $\mathbf{u}_1^T$ and $\mathbf{u}_2^T$ respectively.
In matrix form, given that $\mathbf{u}_i^T \mathbf{y}_0 = \sin\phi_i$, we get:

$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix} \ddot{\mathbf{P}} = \begin{bmatrix} \sin\phi_1 & 0 \\ 0 & \sin\phi_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} - l \begin{bmatrix} \dot{\phi}_1^{\,2} \\ \dot{\phi}_2^{\,2} \end{bmatrix} \tag{2.19}$$

Which can be written in the form:

$$\mathbf{A}\ddot{\mathbf{P}} = \mathbf{B}\ddot{\mathbf{q}}_a - \mathbf{b}_c \tag{2.20}$$

where:

- $\ddot{\mathbf{q}}_a = \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$
- $\mathbf{b}_c = l \begin{bmatrix} \dot{\phi}_1^{\,2} \\ \dot{\phi}_2^{\,2} \end{bmatrix}$

Finally we can write the expression for the end effector accelerations as:

$$\ddot{\mathbf{P}} = \mathbf{A}^{-1}(\mathbf{B}\ddot{\mathbf{q}}_a - \mathbf{b}_c) \tag{2.21}$$

Analysis

Figure 2.7 shows the comparison between the results of the outputs $(\ddot{x}, \ddot{y})$ obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on $\ddot{x}$, while the blue one represents the error on $\ddot{y}$. We can observe that the errors have values around 10^{-5} and 10^{-6} m/s^2 respectively. It can be considered as a satisfactory result, given that the absolute values of the end effector accelerations $(\ddot{x}, \ddot{y})$ in the simulation are in the order of 10^{-1} and 10^{-2} m/s^2 respectively. Hence, in both cases the percent error is around 0.01%.

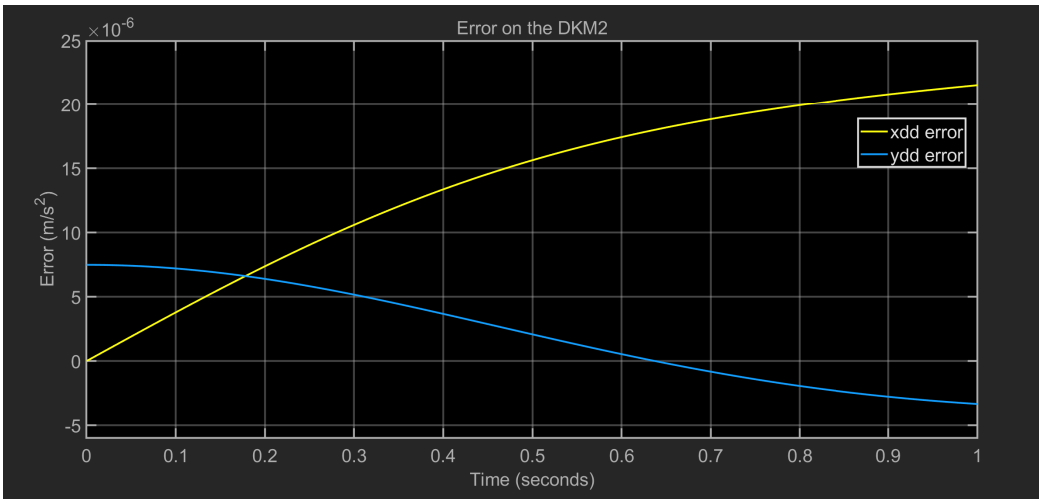

Figure 2.7: Error on DKM2

2.3.2 Passive angle accelerations

This model gives the passive angle accelerations $(\ddot{\phi}_1, \ddot{\phi}_2)$ as a function of the end effector accelerations $(\ddot{x}, \ddot{y})$ the active joint accelerations $(\ddot{q}_1, \ddot{q}_2)$ and the passive angles (ϕ_1, ϕ_2):

$$[\ddot{\phi}_1, \ddot{\phi}_2] = passive\_angle\_accelerations(\ddot{x}, \ddot{y}, \ddot{q}_1, \ddot{q}_2, \phi_1, \phi_2)$$

Premultiplying the equations (2.18) by $(\mathbf{Eu}_1)^T$ and $(\mathbf{Eu}_2)^T$ respectively, and noting that $(\mathbf{Eu}_1)^T\mathbf{y}_0 = \cos\phi_1$ and $(\mathbf{Eu}_2)^T\mathbf{y}_0 = \cos\phi_2$, we obtain:

$$\begin{bmatrix}(\mathbf{Eu}_1)^T\\(\mathbf{Eu}_2)^T\end{bmatrix}\ddot{\mathbf{P}} = \begin{bmatrix}\cos\phi_1 & 0\\0 & \cos\phi_2\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\\ddot{q}_2\end{bmatrix} + \begin{bmatrix}l & 0\\0 & l\end{bmatrix}\begin{bmatrix}\ddot{\phi}_1\\\ddot{\phi}_2\end{bmatrix} \tag{2.22}$$

Which can be written in the form:

$$\mathbf{J}_t\ddot{\mathbf{P}} = \mathbf{J}_{ta}\ddot{\mathbf{q}}_a + \mathbf{J}_{td}\ddot{\mathbf{q}}_d \tag{2.23}$$

where $\ddot{\mathbf{q}}_d = \begin{bmatrix}\ddot{\phi}_1\\\ddot{\phi}_2\end{bmatrix}$

Finally we can obtain an explicit equation for the passive angle accelerations:

$$\ddot{\mathbf{q}}_d = \mathbf{J}_{td}^{-1}(\mathbf{J}_t\ddot{\mathbf{P}} - \mathbf{J}_{ta}\ddot{\mathbf{q}}_a) \tag{2.24}$$

Analysis

Figure 2.8 shows the comparison between the results of the outputs $(\ddot{\phi}_1, \ddot{\phi}_2)$ obtained from the MATLAB function and the ADAMS simulation: the yellow

line corresponds to the error on $\ddot{\phi}_1$, while the blue one represents the error on $\ddot{\phi}_2$. We can observe that both errors have values around 10^{-5} rad/s^2.
It can be considered as a satisfactory result, given that the absolute values of the passive angle accelerations $(\ddot{\phi}_1, \ddot{\phi}_2)$ in the simulation are in the order of 10^{-1} rad/s^2. Hence, in both cases the percent error is around 0.01%.

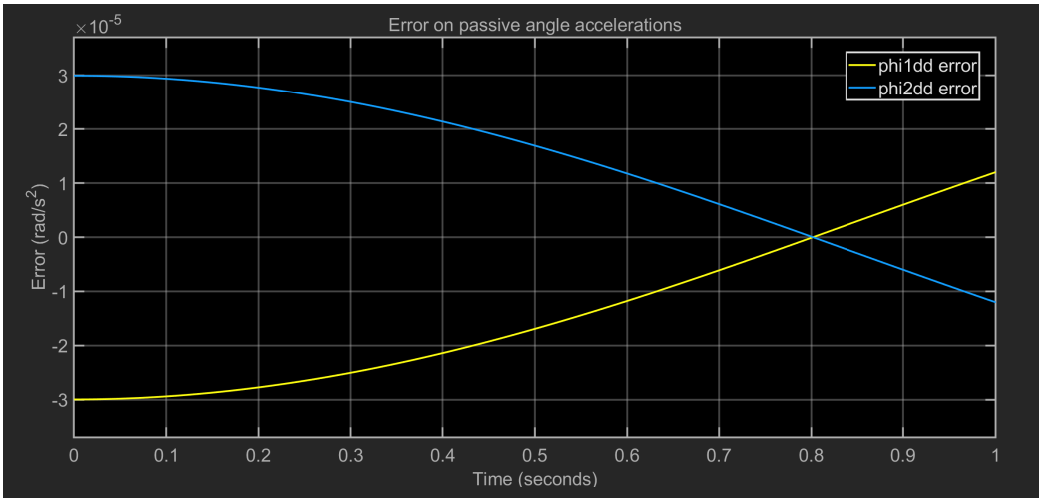

Figure 2.8: Error on passive angle accelerations

2.4 Dynamic Analysis

2.4.1 Inverse Dynamic Model (IDM)

Hereafter the Inverse Dynamic Model of the Biglide robot is computed following the Lagrange method. This model gives the input efforts $\boldsymbol{\tau}$ of the robot as a function of the active joints accelerations $\ddot{\mathbf{q}}_a$, the active joint velocities $\dot{\mathbf{q}}_a$ and the active joints positions $\mathbf{q}_a$. Considering the potential energy U due to gravity effects to be zero, the Lagrangian function is equal to the kinetic energy of the system $(L = E)$. The contributions to the kinetic energy are given by the end effector with mass m_p and by the feet with mass of m_f for each foot.
The kinetic energy due to the end effector can be expressed as:

$$E_p = \frac{1}{2} m_p (\dot{x}^2 + \dot{y}^2) \tag{2.25}$$

Since $\mathbf{v}_1 = \dot{q}_1 \mathbf{z}_1 = \dot{q}_1 \mathbf{y}_0$ and $\boldsymbol{\omega}_1 = \mathbf{0}$, the contribution of the first foot is:

$$E_{f_1} = \frac{1}{2} m_f {\dot{q}_1}^2 \tag{2.26}$$

In the same way, since $\mathbf{v}_2 = \dot{q}_2\mathbf{z}_2 = \dot{q}_2\mathbf{y}_0$ and $\boldsymbol{\omega}_2 = \mathbf{0}$, the contribution of the second foot is:

$$E_{f_2} = \frac{1}{2}m_f{\dot{q}_2}^2 \tag{2.27}$$

Finally, summing (2.25), (2.26), (2.27), the overall kinetic energy of the system is:

$$E = \frac{1}{2}m_p(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}m_f({\dot{q}_1}^2 + {\dot{q}_2}^2) \tag{2.28}$$

We know that using the Lagrange multipliers and the closure loop equations, we can conclude that the solution to the IDM can be obtained solving the following system of equations with respect to $\boldsymbol{\tau}$:

$$\begin{cases} \boldsymbol{\tau} + \mathbf{B}^T\boldsymbol{\lambda} = \boldsymbol{\tau}_a \\ \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{w}_r \end{cases} \tag{2.29}$$

where $\boldsymbol{\tau}_a$ and $\mathbf{w}_r$ are the open loop torques of the structure and can be found as follows:

$$\boldsymbol{\tau}_a = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}_a}\right)^T - \left(\frac{\partial L}{\partial \mathbf{q}_a}\right)^T = m_f \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \tag{2.30}$$

$$\mathbf{w}_r = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{x}}}\right)^T - \left(\frac{\partial L}{\partial \mathbf{x}}\right)^T = m_p \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{2.31}$$

Then deriving the expression of λ from the second equation of the system and substituting its expression in the first equation, we can find that:

$$\boldsymbol{\tau} = m_f\ddot{\mathbf{q}}_a + m_p\mathbf{J}^T\ddot{\mathbf{P}} \tag{2.32}$$

where $\mathbf{J} = -\mathbf{A}^{-1}\mathbf{B}$.
Furthermore, we know that the IDM can also be expressed under the generic form:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) \tag{2.33}$$

where $\mathbf{M}(\mathbf{q})$ is the robot generalized inertia matrix and $\mathbf{c}(\mathbf{q},\dot{\mathbf{q}})$ is the vector of Coriolis, centrifugal and gravity effects.
In order to do so, equation (2.21) derived in the *DKM2* can be used to express the input torques as a function of the joint coordinates only, and replaced in (2.32):

$$\boldsymbol{\tau} = \left(\begin{bmatrix} m_f & 0 \\ 0 & m_f \end{bmatrix} + m_p\mathbf{J}^T\mathbf{J}\right)\ddot{\mathbf{q}}_a + m_p\mathbf{J}^T\mathbf{A}^{-1}\mathbf{b}_c \tag{2.34}$$

By comparing equations (2.34) and (2.33) we can conclude that for this robot:

$$\begin{cases} \mathbf{M}(\mathbf{q}) = \begin{bmatrix} m_f & 0 \\ 0 & m_f \end{bmatrix} + m_p \mathbf{J}^T \mathbf{J} \\ \\ \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) = m_p \mathbf{J}^T \mathbf{A}^{-1} \mathbf{b}_c \end{cases} \tag{2.35}$$

Analysis

Figure 2.9 shows the comparison between the results of the input efforts $\boldsymbol{\tau} = [f_1, f_2]^T$ obtained from the MATLAB function and the ADAMS simulation: the yellow line corresponds to the error on f_1, while the blue one represents the error on f_2. We can observe that both errors have values around 10^{-4} N. It can be considered as a satisfactory result, given that the absolute values of the efforts (f_1, f_2) in the simulation are in the order of 10^{-1} N. Hence, in both cases the percent error is around 0.1%.

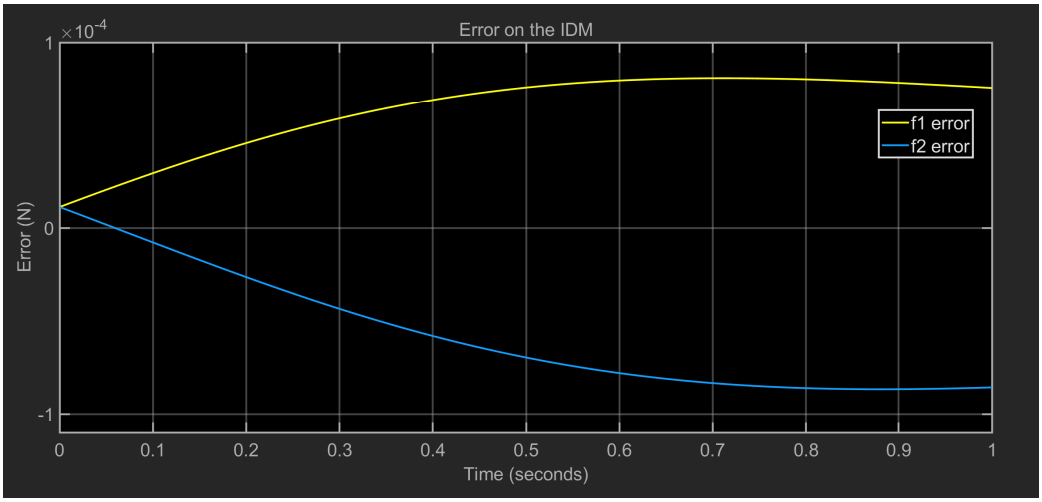

Figure 2.9: Error on IDM

Chapter 3

Control co-simulation

This chapter presents two types of control co-simulations implemented on SIMULINK using the plant exported from ADAMS. The first one is the Kinematic Control, which aims at controlling the joint angle velocities $(\dot{q}_1, \dot{q}_2)$, given as input to the ADAMS plant. In the same way, the second one, that is the Computed Torque Control, aims at controlling the forces applied by the actuated joints (f_1, f_2).

3.1 Trajectory generation

For both control schemes it was necessary to design a trajectory to be followed by the robot, which includes the desired end effector positions (x_t, y_t), velocities $(\dot{x}_t, \dot{y}_t)$ and accelerations $(\ddot{x}_t, \ddot{y}_t)$. To do this a polynomial of the 5^{th} order has been built as follows:

$$\begin{cases} x_t(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \\ y_t(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \end{cases} \tag{3.1}$$

By defining:

- $\mathbf{A} = [a_0, ..., a_5]^T$
- $\mathbf{x}_t = [x_0, \dot{x}_0, \ddot{x}_0, x_f, \dot{x}_f, \ddot{x}_f]^T$
- $\mathbf{B} = [b_0, ..., b_5]^T$
- $\mathbf{y}_t = [y_0, \dot{y}_0, \ddot{y}_0, y_f, \dot{y}_f, \ddot{y}_f]^T$

- $\mathbf{T} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 1 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 1 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}$

we can compute:

$$\mathbf{A} = \mathbf{T}^{-1}\mathbf{x}_t$$

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{y}_t$$

The desired velocities and accelerations are taken as first and second derivatives of the $x_t(t), y_t(t)$ expressions from (3.1).

3.2 Kinematic Control law

The knowledge of the initial and final positions of the end effector, along with its velocities and accelerations are required to build these trajectories. The initial position of the end effector was chosen at $(0, 0.30005393)$ for it corresponds to the initial position of the robot in the ADAMS file. The final position was chosen to be $(0.1, 0.1)$. All the conditions for the velocities and accelerations were set to zero. The duration of the simulation is 2 seconds (hence $t_f = 2$).

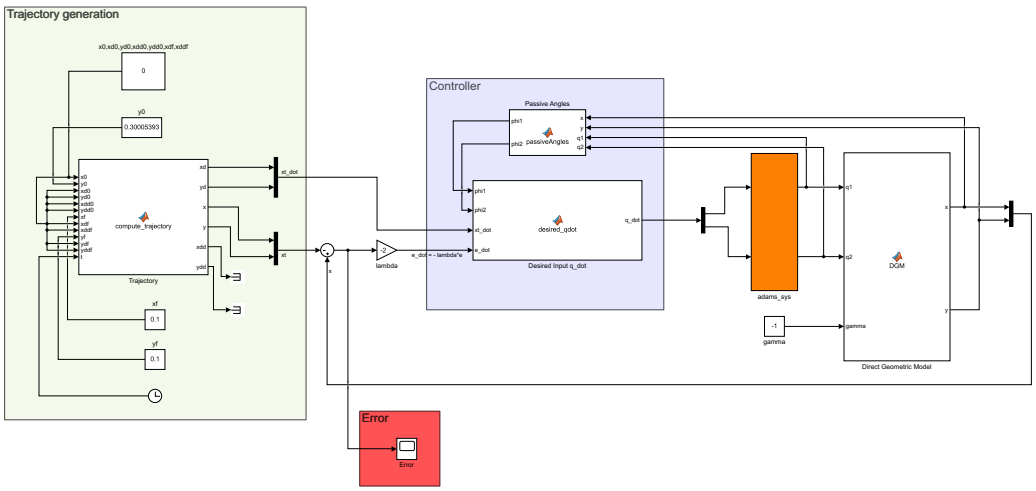

Figure 3.1: Kinematic Control - SIMULINK scheme

The desired behavior in closed-loop is the following:

$$\dot{\mathbf{e}} = -\lambda(\mathbf{x} - \mathbf{x}_t) \tag{3.2}$$

where $\lambda > 0$ and given that $\mathbf{e} = \mathbf{x} - \mathbf{x}_t$ is the tracking error. The control input $\dot{\mathbf{q}}$ to the ADAMS plant is computed by an additional block, that takes as input the passive angle joints (ϕ_1, ϕ_2), the desired velocities of the end effector $(\dot{x}_t, \dot{y}_t)$ and the error variation $\dot{\mathbf{e}}$, and exploits the following formula:

$$\dot{\mathbf{q}} = -\mathbf{B}^{-1}\mathbf{A}(\dot{\mathbf{x}}_t - \dot{\mathbf{e}}) \tag{3.3}$$

The SIMULINK scheme is reported in Figure 3.1.

The current joint variables, outputs of the ADAMS plant, are given to the DGM block in order to find the corresponding current end effector positions, which constitute the feedback to compute the tracking error.
A value of λ equal to 2 has been set, for it gives good convergence behavior, as shown in Figure 3.2.

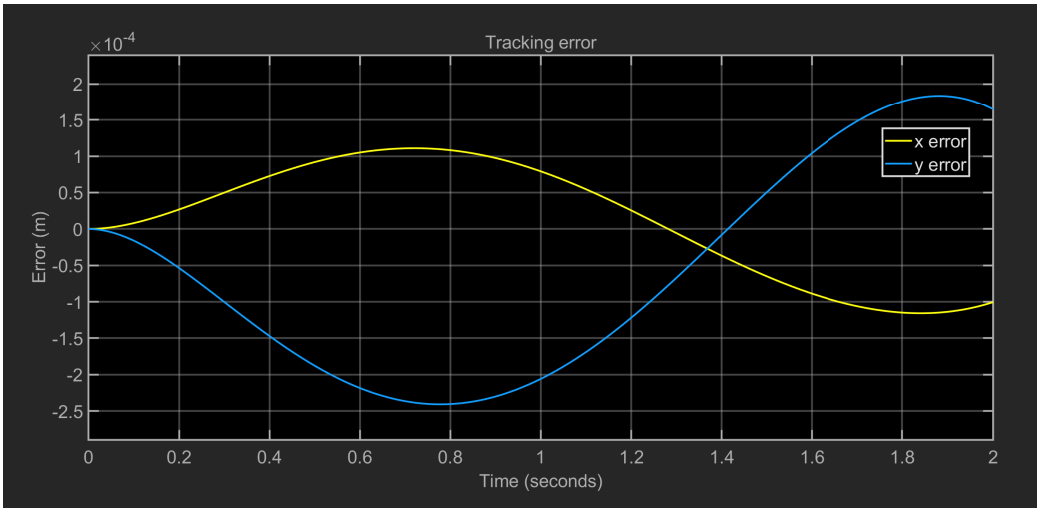

Figure 3.2: Kinematic Control - tracking error

Furthermore, different communication times were tested: starting from the default value set to 0.05 s, the behavior of the plant was tested using 0.1 s, then 0.025 s. Then, from our observations we can conclude that doubling the communication interval doubles the relative tracking error, as well as halving it halves the error. Results are shown in the following Figures 3.3 and 3.4.

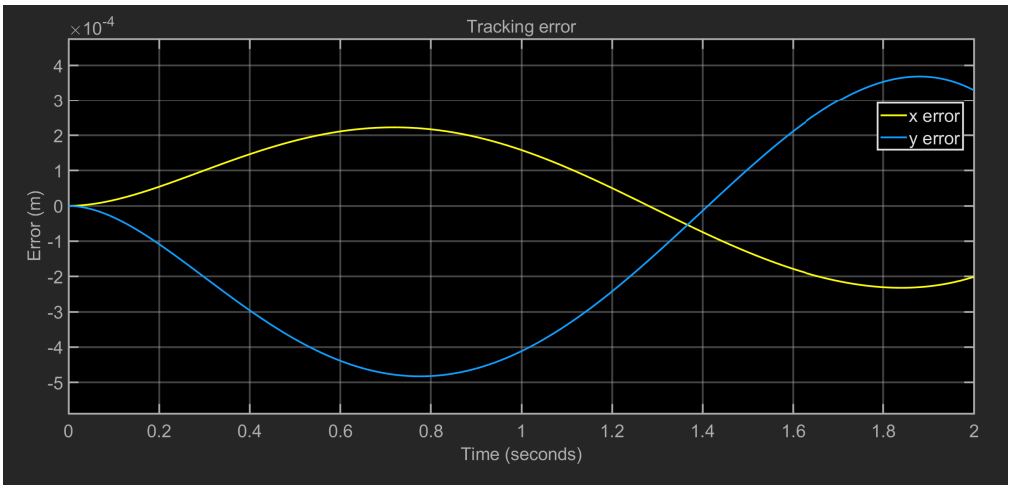

Figure 3.3: Tracking error for communication time equal to 0.1 s

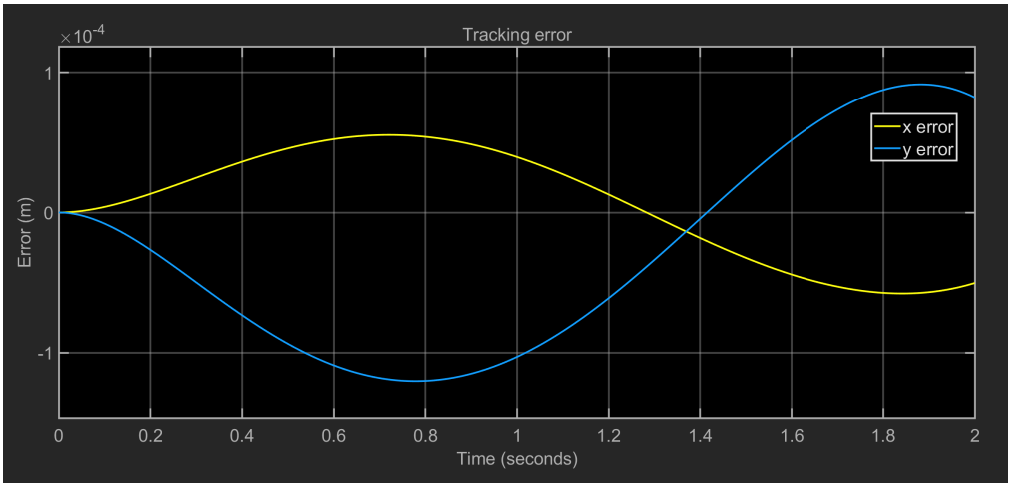

Figure 3.4: Tracking error for communication time equal to 0.025 s

3.3 Computed Torque Control law

The Computed Torque Control is based on the feedback linearization of the system through the inverse dynamic model, written in the generic form:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) \tag{3.4}$$

as shown in section 2.4.
The expressions of $\mathbf{M}(\mathbf{q})$ and $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ are computed, at each time instant,

from the passive joint positions and velocities, as in (2.35).
The auxiliary control is defined as:

$$\mathbf{v} = \mathbf{M}^{-1}(\boldsymbol{\tau} - \mathbf{c}) = \ddot{\mathbf{q}}_a \tag{3.5}$$

and computed as the output of a PD controller, as:

$$\mathbf{v} = \ddot{\mathbf{q}}_t + \mathbf{K}_d(\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_a) + \mathbf{K}_p(\mathbf{q}_t - \mathbf{q}_a) \tag{3.6}$$

where:

- $\mathbf{K}_p$ and $\mathbf{K}_d$ are two independent positive definite diagonal matrices; in the simulation they are set to $\mathbf{I}_2$ and $2\mathbf{I}_2$
- $\mathbf{q}_t$, $\dot{\mathbf{q}}_t$, $\ddot{\mathbf{q}}_t$ are the desired positions, velocities, and accelerations of the joints corresponding to the desired trajectory; they are obtained by applying the *IGM* to the (x, y) positions coming from the trajectory generation.

Then, the torque input deduced from the control law:

$$\boldsymbol{\tau} = \mathbf{M}\mathbf{v} + \mathbf{c} = \mathbf{M}(\ddot{\mathbf{q}}_t + \mathbf{K}_d(\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_a) + \mathbf{K}_p(\mathbf{q}_t - \mathbf{q}_a)) + \mathbf{c} \tag{3.7}$$

is the input of the ADAMS model, hence the torque that will be applied to the motors, ensuring the convergence of the error $\tilde{\mathbf{q}} = \mathbf{q}_{ref} - \mathbf{q}_a$ to 0.
The overall SIMULINK scheme described above is shown in Figure 3.5.

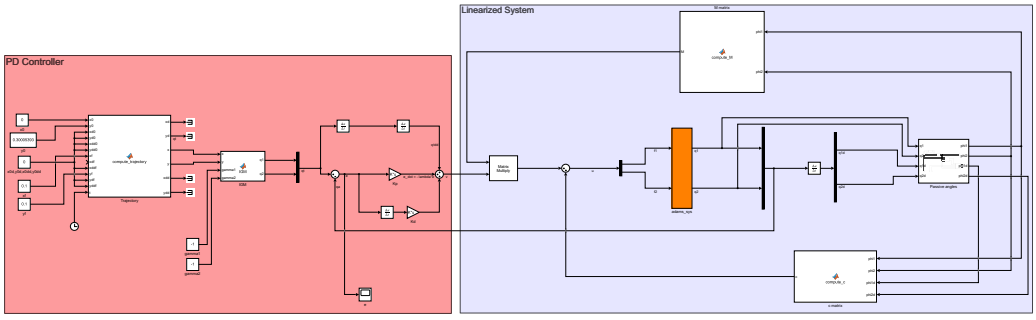

Figure 3.5: Computed Torque Control - SIMULINK scheme

Two trajectories were defined to test the controller: the first one, to check the trajectory tracking, was defined from point $(0, 0.30005393)$ to point $(0.1, 0.1)$ (as for the Kinematic Control Law); the second one was chosen so that the

robot would cross a Type 2 singularity, with the initial point at $(0.1, 0.4)$ and the final point at $(-0.1, 0.2)$, and the robot in the working mode characterized by $\gamma_1 = 1$, $\gamma_2 = -1$ (the exact singularity conditions are further detailed in Chapter 4).
In this latter case, the initial positions of the actuated joints had to be properly set in the ADAMS plant, in order to ensure the initial tracking error to be 0.
Figures 3.6 and 3.7 show the tracking error $\tilde{\mathbf{q}}$ in the two cases: the first one exhibits a good convergence behavior, while in the second case the simulation stops and the error increases instantaneously.
This is due to the fact that the robot approaches the singularity and fails to track the trajectory: Figure 3.8 indeed shows that, while approaching the singularity, the input forces (f_1, f_2) that should be exerted by the two actuators tend to infinity.

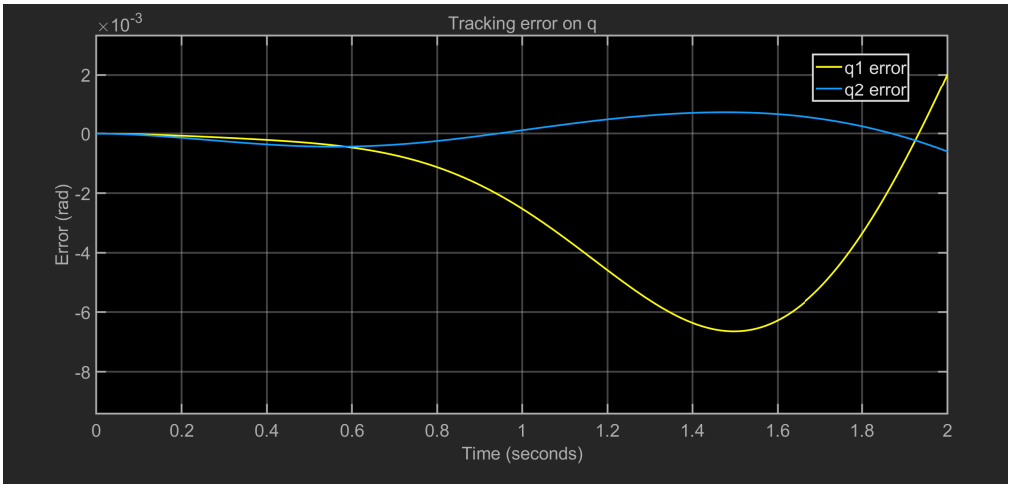

Figure 3.6: CTC - tracking error

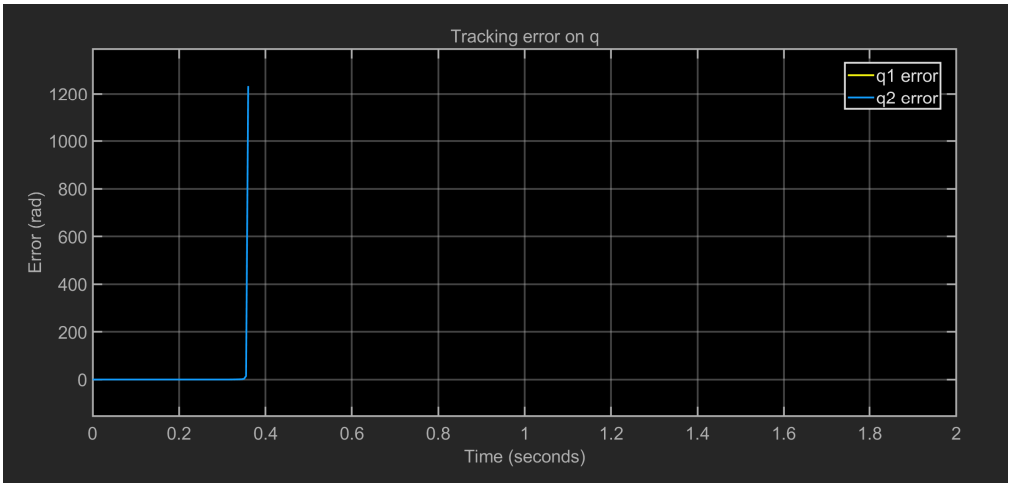

Figure 3.7: CTC - tracking error with singularity

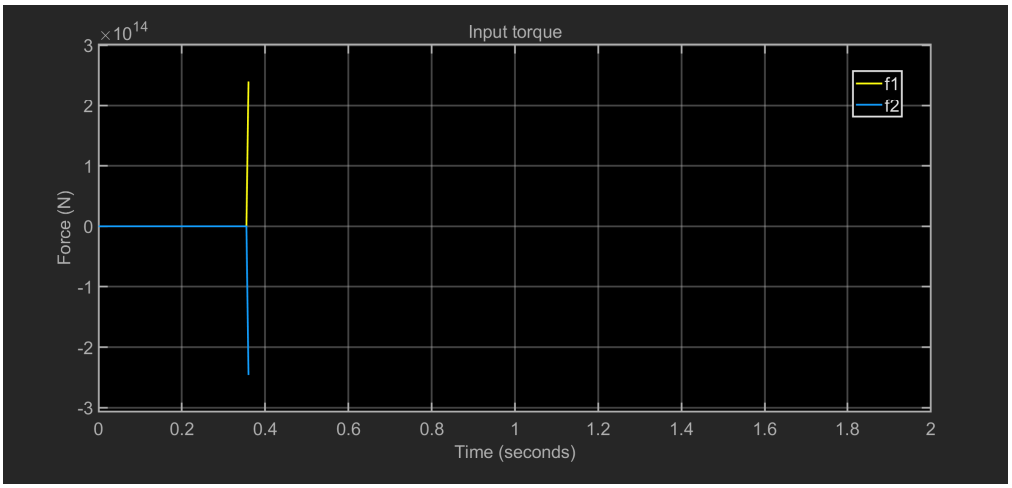

Figure 3.8: CTC - forces exerted by the actuators with singularity

Chapter 4

Trajectory crossing a Type 2 singularity

In this chapter the aim is to implement a trajectory to be followed by the end effector of the robot in order to cross a Type 2 singularity (or parallel singularity). When the determinant of the matrix $\mathbf{A}$, defined in equation (2.13), is equal to zero, the robot is in a Type 2 singularity, that is matrix $\mathbf{A}$ is not invertible anymore.
In such a situation there exists a twist $\mathbf{t}_s \neq 0$ such that $\mathbf{A}\mathbf{t}_s = 0$.
Given that $\mathbf{A} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix}$, we have that:

$$det(\mathbf{A}) = 0 \iff \mathbf{u}_1 \parallel \mathbf{u}_2$$

which is true when

$$\phi_2 = \phi_1 + \pi \tag{4.1}$$

Substituting this last condition in matrix $\mathbf{A}$ we can derive a base for the kernel of $\mathbf{A}$ which can be written as

$$\mathbf{t}_s = [\sin\phi_1, -\cos\phi_1]^T$$

Starting from the second equation in (2.29) and left multiplying it by $\mathbf{t}_s$ we obtain $\mathbf{t}_s^T \mathbf{w}_r = 0$ which gives the following condition:

$$\ddot{y} = \ddot{x}\tan\phi_1 \tag{4.2}$$

This is the criterion to be respected in order to cross a Type 2 singularity, since there's no more degeneracy of the dynamic model.
Substituting the degeneracy condition (4.1) in (2.3) and (2.4) we get that the singularity of Type 2 only occurs along the line $x = 0$.

In order to generate an overall trajectory suitable to cross a Type 2 singularity, two different trajectories were generated and stacked: the first one starting at the initial position of the robot (we chose (0.1,0.3)) and ending in the singularity point (we chose (0,1)). The other one starts at the singularity point and ends at the final position (chosen at (-0.1,1.8)). The resulting trajectory is the stack of the two, taking the singularity point just once. Then we imposed the condition (4.2) on the accelerations. Suitable values for velocities were chosen. The corresponding values of joints (q_1, q_2), were then computed by means of the *IGM* model, with $\gamma_1 = 1$ and $\gamma_2 = -1$. Those joint values were finally imported in ADAMS and used as splines to generate the motions of the actuated joints.
The results of the simulation are shown in Figure 4.1 and 4.2.

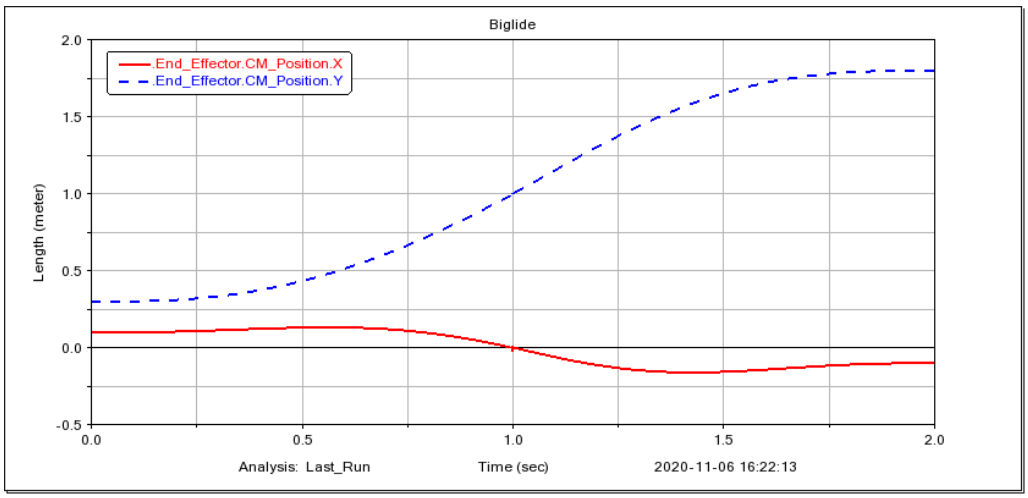

Figure 4.1: Trajectories on x and y to cross a Type 2 singularity respecting the criterion

As we can notice the trajectories on x and y start and end in the desired positions and the shape of the functions is the same as the desired ones computed by MATLAB.

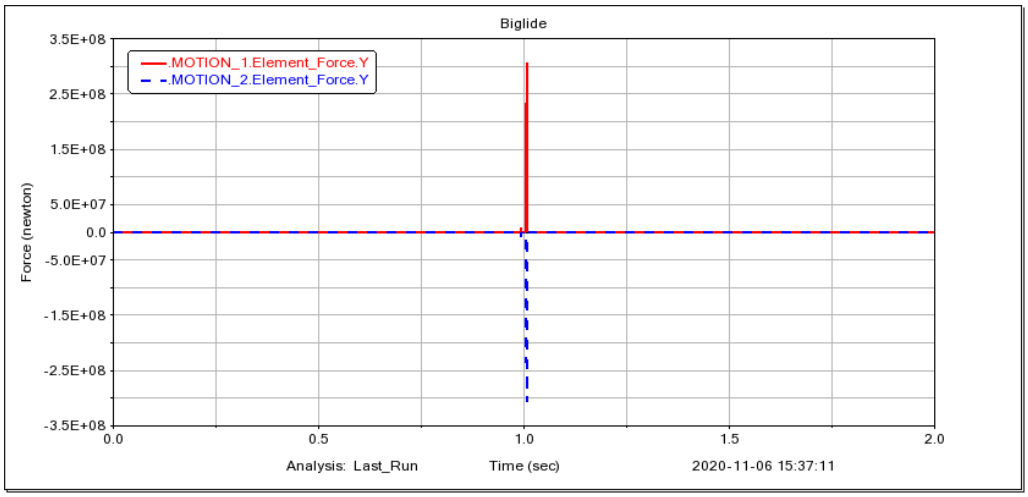

Figure 4.2: Input forces necessary to follow the trajectory when respecting the criterion

A trajectory equal to the one already described was also defined, but this time it was chosen in such a way that it did not respect the criterion, by putting $\ddot{y} = 0$.
The results are shown in Figures 4.3 and 4.4.
From Figure 4.3 we can notice how the trajectory performed by the robot in the ADAMS simulation is not the desired one in terms of starting and ending positions, as well as oscillating phenomena happen in correspondence of the singularity point.
Regarding the forces, we can notice that in the singularity point they take values in the order of 10^8 in the first case and of 10^7 in the second case. This may be due to numerical issues in the first case, while in the second one it is due to the fact that the degeneracy on $\mathbf{A}$ matrix produces infinite forces.

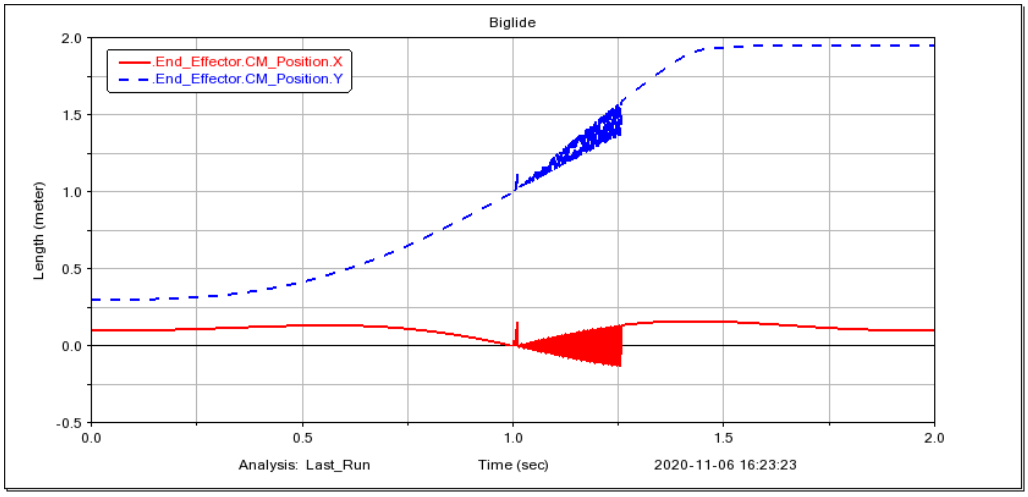

Figure 4.3: Trajectories on x and y to cross a Type 2 singularity without respecting the criterion

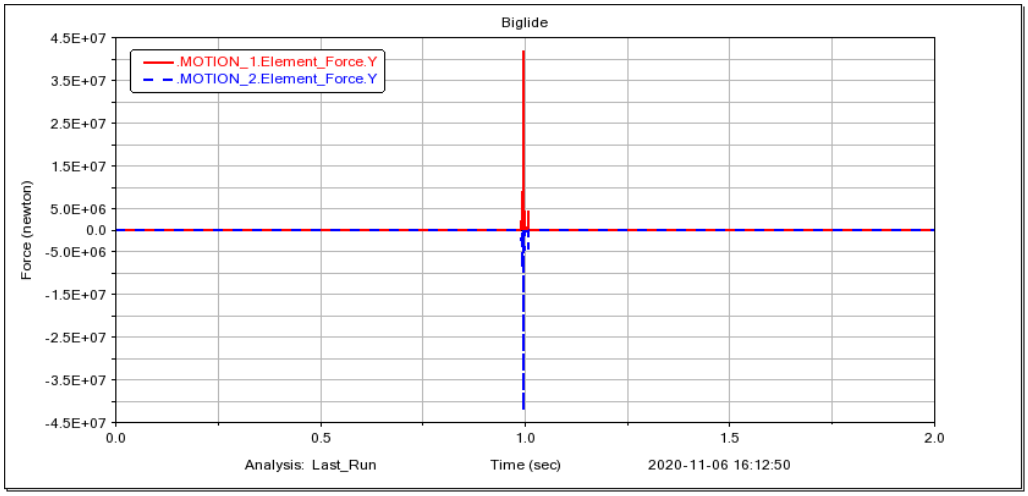

Figure 4.4: Input forces necessary to follow the trajectory when not respecting the criterion

Chapter 5

Files organization and instructions

5.1 Folders and files

All the files are organized into the following folders:

- **Models**: it contains the Simulink file where all the models are compared (*biglide.slx*) and the associated plant *.m* file (*Controls_Plant_1.m*)
- **Kinematic Control**: it contains the Simulink file with the scheme for the Kinematic Controller (*Biglide_KC.slx*) and the associated plant *.m* file (*Controls_Plant_2.m*)
- **CT Control**: it contains the Simulink files for the Computed Torque Control, both the one containing the scheme following the *standard* trajectory (*Biglide_CTC.slx*), and the one following the trajectory crossing the Type 2 Singularity (*Biglide_CTC_singularity.slx*); it also contains the two corresponding plants *.m* files (*Controls_Plant_3.m* and *Controls_Plant_3_singularity.m*), and the *IGM* in a separate file (*IGM.m*) that is needed when generating the trajectory for the joints
- **Singularity Crossing**: it contains the *Biglide.bin* ADAMS file with both the splines respecting the singularity criterion (*SPLINE_q1_cross* and *SPLINE_q2_cross*) and the ones that do not (*SPLINE_q1_not_cross* and *SPLINE_q2_not_cross*); anyway they can be generated respectively using the *singularity_crossing_traj.m* and *singularity_not_crossing_traj.m* scripts; the folder also contains the corresponding *.txt* files

5.2 Instructions on how to run the simulations

In order to make the simulations run, the following actions need to be done: run the plant .m file and then the corresponding SIMULINK file. The ADAMS subsystems on SIMULINK are set in "interactive" mode, so that when running the simulation the robot motion can be observed on ADAMS. In the case of the crossing Type 2 singularity open the ADAMS .bin file and start a simulation directly on ADAMS. As default the spline respecting the criterion can be found in the current imposed motions, but they can be changed to simulate the different behaviors.