Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Puebla

TE3003B.501

Integración de robótica y sistemas inteligentes

Reto Semanal 1 | Manchester Robotics

Frida Lizett Zavala Pérez A01275226
Diego Garcia Rueda A01735217
Alejandro Armenta Arellano A01734879

15 de Abril del 2024

ÍNDICE

Resumen

Para la realización de este reto fue necesario contar con el sistema operativo ubuntu versión 20.04 y cambiar de *ROS melodic* a *ROS noetic*, posteriormente se descargó el nodo proporcionado por *Manchester Robotics*, agregándolo al entorno de trabajo para su próxima edición. Posteriormente se preparó la ecuación que se utilizará, despejando la variable requerida para poder resolverla. La ecuación requiere retroalimentación doble, por lo que se ha ubicado en el núcleo del código que se repite constantemente. Finalmente, se hizo uso del *UDRF* proporcionado por Manchester para enviar los valores a *RVIZ*, permitiendo así visualizar el movimiento y comportamiento de la ecuación.

Objetivos

El objetivo del reto de semana 1 que presentó *Manchester Robotics* consiste en analizar y mostrar el comportamiento de un brazo robótico, utilizando ecuaciones derivadas de la metodología de Euler-Lagrange, haciendo uso de este método para convertir la ecuación otorgada a un conjunto de ecuaciones con diferentes variables. La simulación se basa en valores ideales ya que para el sistema se ignoran otros factores que podrían cambiar el sistema. Esto principalmente permite estudiar los conceptos básicos del movimiento de un robot de un solo brazo sin los problemas adicionales que podrían ocurrir en la vida real. Este reto se diseñará en el sistema de robot *ROS* para hacerlo funcionar y poder visualizarlo en *RVIZ*, una herramienta 3D.

Introducción

En el contexto de la robótica, un sistema dinámico modela el comportamiento en función del tiempo de un robot en movimiento, teniendo en cuenta factores como la posición, la velocidad, las fuerzas aplicadas y la interacción con su entorno. La dinámica del sistema puede describirse mediante ecuaciones diferenciales que relacionan las variables de estado del sistema con las entradas y las salidas. Los sistemas dinámicos pueden ser lineales o no lineales, dependiendo de si cumplen con el principio de superposición y homogeneidad. Además, pueden ser estables o inestables, según cómo respondan a las perturbaciones externas.

Los espacios de estado dinámicos son una forma de representar el estado y la evolución de un sistema dinámico a lo largo del tiempo. En el contexto de un robot péndulo, los espacios de estado dinámicos se utilizan para describir la posición, la velocidad y posiblemente otras variables relevantes del sistema, así como las fuerzas y momentos que actúan sobre él.

Las ecuaciones de Euler-Lagrange son una herramienta matemática utilizada en la mecánica para derivar las ecuaciones de movimiento de un sistema dinámico a partir de una función llamada lagrangiana. Para un robot péndulo, las ecuaciones de Euler-Lagrange se pueden utilizar para modelar su comportamiento dinámico, teniendo en cuenta factores como la longitud del péndulo, la masa del péndulo, la aceleración debido a la gravedad, entre otros.

Representación del péndulo en espacio de estados:

Partiendo de la ecuación diferencial que describe el comportamiento del péndulo (ecuación 1) es que podemos obtener las variables de estado (ecuación 3) observando las entradas y salidas del sistema (ecuación 2), en este caso la salida del sistema es la posición del eslabón del péndulo (q).

$$Jq\ddot{} + kq\dot{} + mga\cos(q) = \tau \quad (1)$$

$$y = q \quad (2)$$

$$x_1 = y,\ x_2 = y\dot{} \quad (3)$$

A partir de las variables de estado obtenidas en (3) tenemos que realizar derivaciones que nos permitan desarrollar el sistema en el espacio de estados (4). Para la obtención de la segunda derivada de la posición (aceleración) es necesario hacer un despeje utilizando (1) quedando así la ecuación 5. Tras la obtención de (5) es que podemos completar el desarrollo de (4) resultando así en la ecuación 6 y la ecuación 7.

$$x_1\dot{} = q\dot{},\ x_2\dot{} = q\ddot{} \quad (4)$$

$$q\ddot{} = \frac{\tau - kq\dot{} - mga\cos(q)}{J} \quad (5)$$

$$x_1\dot{} = q\dot{} \rightarrow x_1\dot{} = x_2 \quad (6)$$

$$x_2\dot{} = q\ddot{} \rightarrow x_2\dot{} = \frac{\tau - kq\dot{} - mga\cos(q)}{J} \quad (7)$$

Serán las ecuaciones (6) y (7) las que nos permitirán simular el comportamiento del péndulo en la simulación, en la siguiente sección se mostrará la implementación en código de este sistema junto al desarrollo de la solución del problema.

Solución del problema

Para el desarrollo de la solución de este reto semanal se buscó desarrollar el nodo planteado por el socio formador *Manchester Robotics*, el nodo se muestra en la figura 1.

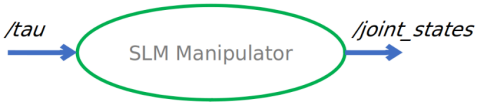


Figura 1. Nodo propuesto por *Manchester Robotics para solución del Reto*

En la figura 1 se puede mostrar como la entrada del nodo es el tópico */tau* que de acuerdo a la sección anterior corresponde a la ecuación diferencial del péndulo, recordando que esta función se obtiene mediante la derivada y la doble derivada de la posición del eslabón final del péndulo.

Para el desarrollo del nodo se decidió implementar un nodo representado en una clase utilizando *rospy* declarando aspectos como lo serían atributos de este nodo/clase que vendrían a representar las constantes o variables necesarias para el desarrollo y cálculo de la ecuación diferencial dentro del nodo. Posteriormente se declaran y se inicializan valores iniciales para la posición y la velocidad (x1 y x2 respectivamente de acuerdo al desarrollo en el espacio de estados) Finalmente se declaran los publicadores de este módulo en los cuales se encuentran tópicos como */velocity* y */position* además del tópico requerido por el socio formador */joint_states* que será el encargado de mover el péndulo dentro de la simulación. Todas estas inicializaciones se encuentran en el *Apéndice 1* en la función ***init()***.

Tras esto se inicializan los valores de las juntas necesarios para la simulación, declarando aspectos como el *nombre*, *tiempo* e *id*. en el desarrollo de las funciones del espacio de estados es que el parámetro ***contJoints.position*** se modificara, siendo posteriormente publicado mediante el tópico */joint_states.*

El desarrollo de las ecuaciones del espacio de estados del sistema se encuentra ejecutándose en la función main utilizando para el cálculo los atributos previamente declarados de la clase del nodo *SLM_sim_Node.* El desarrollo de las líneas de código se explica a continuación mostrando el análogo con el desarrollo previamente explicado.

- `SLM.x1 += SLM.x2 * dt`

Esta línea es el equivalente a la primera parte de la ecuación (3), se especifica en la ecuación (4) que este valor se tiene que derivar, es por este motivo que se realiza un a multiplicación por *dt* (diferencial establecido como 0.001 mediante parámetros de ROS)

- `x2_dot = ((-SLM.m*SLM.g*SLM.a*math.sin(SLM.x1))/SLM.j)-((SLM.k*SLM.x2)/SLM.j)+(SLM.Tau/SLM.j)`

Esta línea es el equivalente a la ecuación (5) que representa el despeje de la aceleración a partir de su ecuación diferencial del sistema del péndulo. Este valor vuelve a utilizarse posteriormente

- `SLM.x2 += x2_dot * dt`

Al igual que en la primera línea, esta línea representa la segunda parte de la ecuación (3) que al igual que se establece en la ecuación (7) es necesario multiplicar la segunda línea por el diferencial anteriormente planteado.

Los cálculos de la posición con las anteriores ecuaciones son publicadas mediante el tópico */joint_State.* En la siguiente sección se mostrarán los resultados del nodo.

Resultados

Con el nodo anterior es que pudimos realizar la simulación usando el archivo launch de la simulación proporcionado por *Manchester Robotics* mostrando así resultados como en las figuras 2-a, 2-b, 2-c y 2-d. Estas imágenes muestran el péndulo simulado en diferentes posiciones iniciales, estas simulaciones fueron realizadas con motivo de comprobar el comportamiento del péndulo dependiendo de la posición inicial que tuviera.

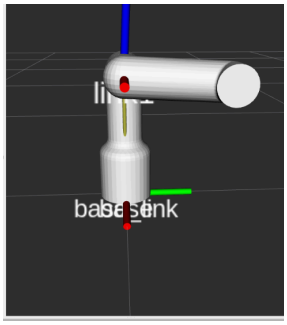


Figura 2-a. Simulación con posición inicial en 0*pi

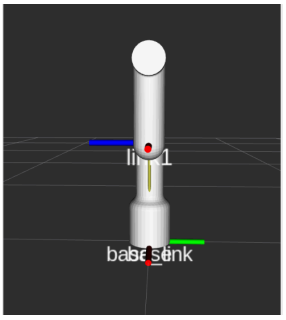


Figura 2-b. Simulación con posición inicial en pi/2

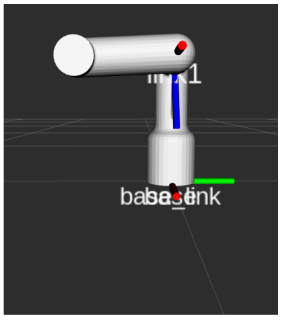


Figura 2-c. Simulación con posición inicial en pi

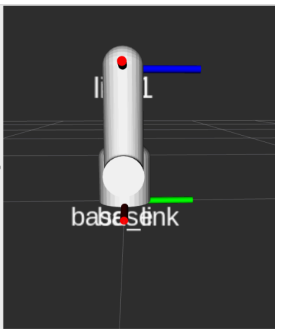


Figura 2-d. Simulación con posición inicial en pi * 1.5

Las figuras anteriormente descritas serán utilizadas en la siguiente sección haciendo un análisis del comportamiento de cada una de la simulaciones

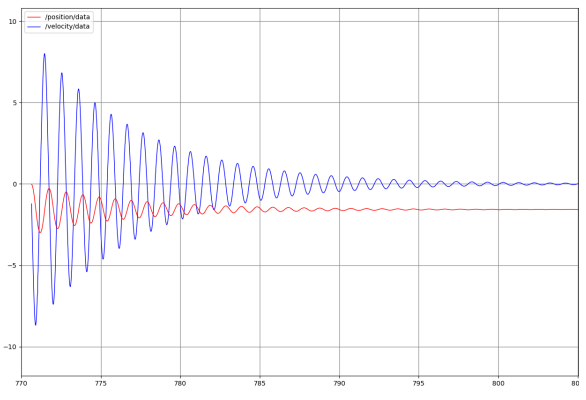
Video de la simulación

https://youtu.be/zlQwQUNA9ys

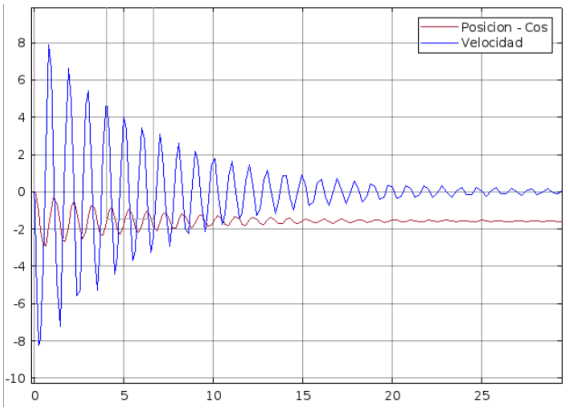
Simulaciones con diferentes posiciones iniciales.

a) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = 0*pi, x2 = 0.0

$$Jq\ddot{}+kq\dot{}+mga\cos(q)=\tau$$



Gráfica 1. *rqt plot*. Escenario a) función cos.



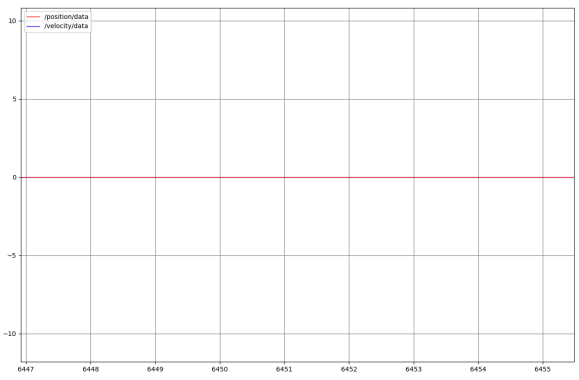
Gráfica 2. *simulink*. Escenario a) función cos.

El comportamiento del péndulo mostrado en este escenario en las gráficas (1) y (2) es idéntico.

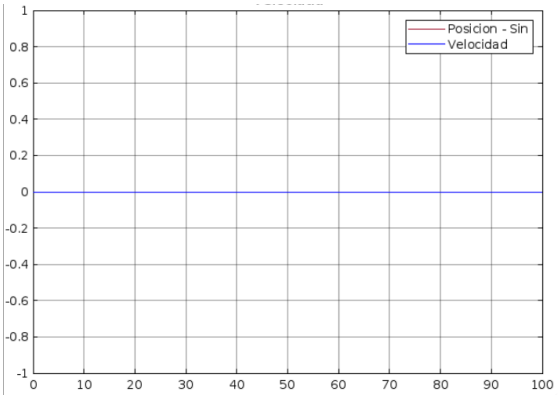
El comportamiento de la velocidad (*/velocity\data*) en *rqt plot* va decreciendo de manera constante similar a un comportamiento logarítmico hasta llegar a un valor de 0 indicando el estado de reposo en el que el eslabón del péndulo se encuentra, llegando a un estado de equilibrio estable conforme el paso del tiempo, al ser un estado de reposo y no haber una fuerza de torque involucrada en el sistema es concluyente que la velocidad final del sistema sea 0.

Para la interpretación de la gráfica (1) y las graficaciones en *rqt plot* posteriores es importante destacar que la posición (*/position\data*) se encuentra representada en radianes, siendo la posición de inicio (0° / pi*0) está representada en el eje con un valor de 0 (se puede observar esta posición inicial en la figura 2-a). Tras el inicio de la simulación el valor de la posición se encuentra oscilando en valores negativos, esto demuestra la oscilación del eslabón del péndulo en la parte inferior. De acuerdo al sistema de referencia, el estado de reposo o equilibrio estable final se encuentra en -90° que al realizar la conversión a radianes, da como resultado: 1.57 radianes negativos (figura 2-b). Siendo por lo tanto correcta esta representación ya que el último valor graficado para la posición llega a estos valores.

$$\boldsymbol{J\ddot{q}+k\dot{q}+mga}\,\mathbf{sin}(\boldsymbol{q})=\boldsymbol{\tau}$$



Gráfica 3. *rqt plot*. Escenario a) función sin.

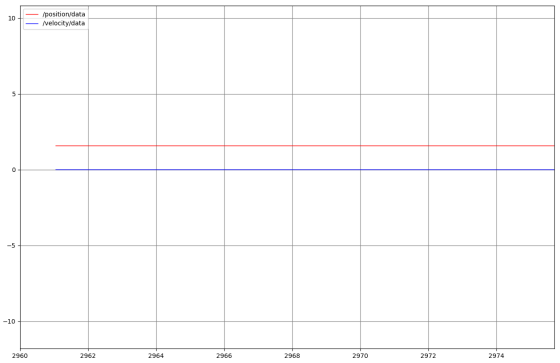


Gráfica 4. *simulink*. Escenario a) función sin.

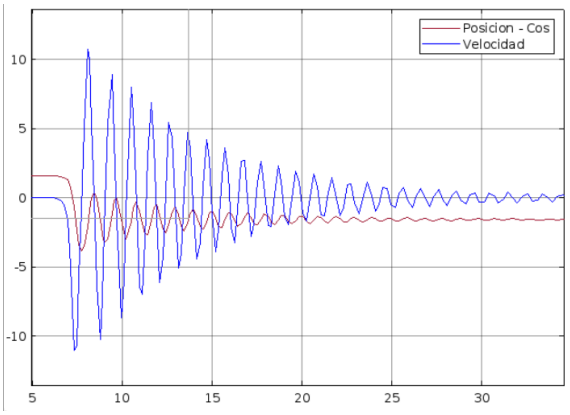
Un comportamiento interesante ocurre al cambiar la identidad trigonometria por el seno, al momento de observar la simulación, la nueva posición de equilibrio del péndulo es en 0° (el estado de equilibrio se muestra en la figura 2-a), al ser un estado de equilibrio tanto la posición como la velocidad se mantienen constantes con un valor de 0, ambas simulaciones muestran un resultado correcto.

b) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = pi/2, x2 = 0.0

$$Jq\ddot{}+kq\dot{}+mga\cos(q)=\tau$$



Gráfica 5. *rqt plot*. Escenario b) función cos.



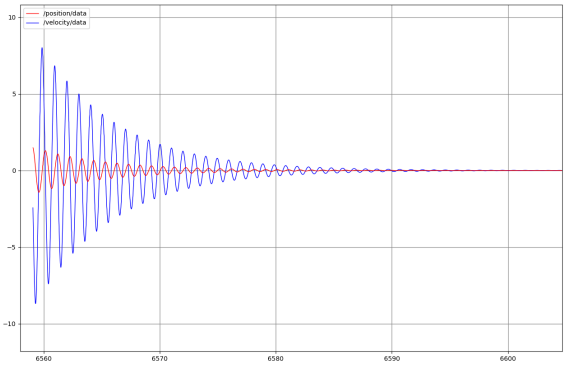
Gráfica 6. *simulink*. Escenario b) función cos.

El comportamiento en las gráficas (5) y (6) es bastante diferente. Esto se debe a que la simulación en ROS muestra ser ideal no teniendo factores externos que puedan romper este estado de equilibrio. Realmente este estado se considera de equilibrio inestable debido a que

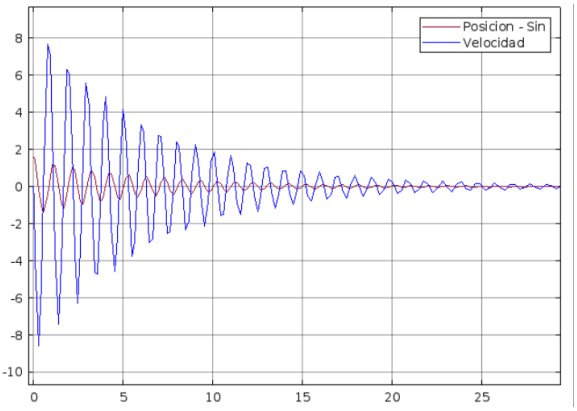
en la realidad una pequeña disturbación puede romper este equilibrio, al no presentarse ninguna de ellas en la simulación es que se mantiene este estado de perpetuidad. El valor constante de la posición (*/position\data*) es de 1.57, esto muestra coherencia debido al marco de referencia (90°/ pi/2) al cambio de radianes es de 1.5704, al mantenerse de manera constante este valor muestra su estado de permanencia en este punto de equilibrio.

Debido a que el estado de equilibrio es constante es que no se presentaba ningún movimiento, por lo tanto es que la velocidad tiene un valor constante de 0 al no existir el movimiento. Por otro lado la respuesta obtenida de simulink inicialmente muestra el mismo comportamiento (se mantiene estable en 90°, figura 2-b), sin embargo después de cierto periodo de tiempo se observa un crecimiento negativo, lo que podría indicar que el péndulo está cayendo y generando oscilaciones posteriores al movimiento.

$$\boldsymbol{Jq\ddot{}+kq\dot{}+mga}\ \mathbf{sin}(\boldsymbol{q})=\boldsymbol{\tau}$$



Gráfica 7. *rqt plot*. Escenario b) función sin.



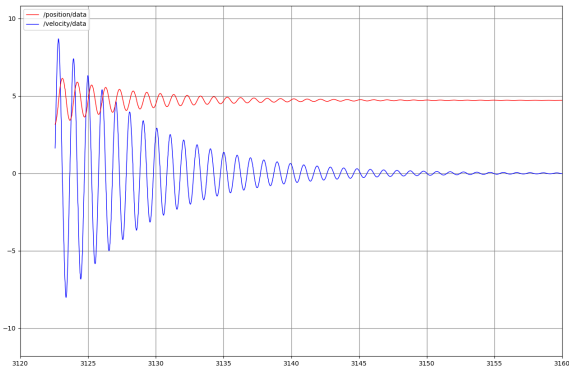
Gráfica 8. *simulink*. Escenario b) función sin.

El comportamiento de las gráficas (6) y (7) es similar al mostrado en la gráfica (1) y (2), en este caso el péndulo vuelve a considerar el estado de equilibrio en 0° debido al marco de referencia, al ser este estado de equilibrio la posición del eslabón se encuentra oscilando en valores positivos y negativos, resultados correctos considerando el marco de referencia.

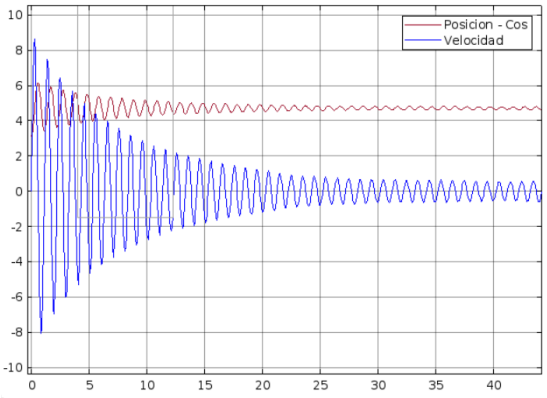
En el caso de la función de la posición inicial planteada en esta situación cia en pi/2 debido a la condición inicial planteada en este escenario es que inicia con un valor de 1.54 radianes (pi/2) decreciendo oscilatoria mente hasta llegar al reposo en 0 radianes. La velocidad igual se encuentra decreciendo periódicamente hasta que se llega al punto de reposo/equilibrio estable, demostrando la desaceleración del eslabón.

c) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = pi , x2 = 0.0

$$Jq\ddot{}+kq\dot{}+mga\cos(q)=\tau$$



Gráfica 9. *rqt plot.* Escenario c) función cos.



Gráfica 10. *simulink.* Escenario c) función cos.

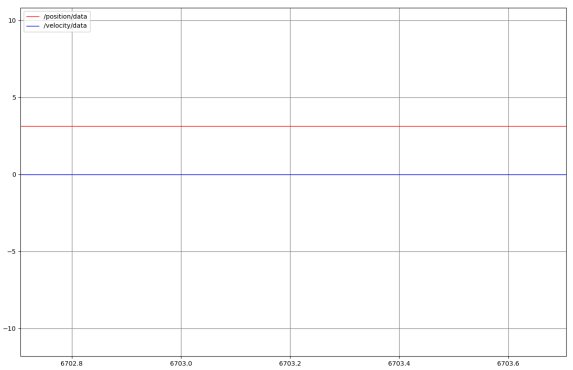
El comportamiento de las gráficas (9) y (10) es idéntico para su velocidad y posición.

En el análisis de la velocidad es que podemos ver un comportamiento prácticamente idéntico al de las gráficas (1) y (2) solo que opuesto, esto debido a que la posición inicial del péndulo es opuesta a la de ese escenario, sin embargo las magnitudes de la velocidades suelen ser las mismas.

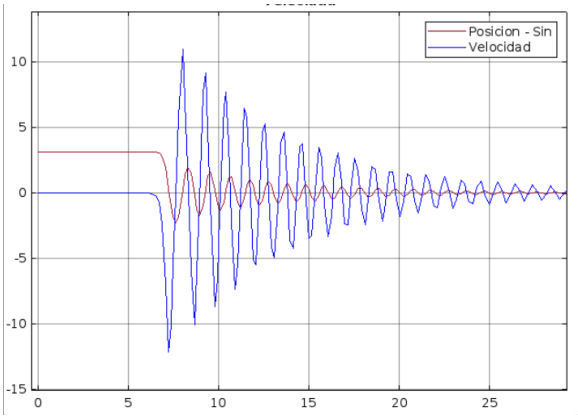
La diferencia principal radica en las gráfica de la posición, en este escenario es que podemos apreciar que tiene un valor aproximado de 4.7 radianes, esta es otra interpretación de las

gráficas (1) y (2) ya que la conversión a grados es de 270° que en el marco de referencia prueba ser equivalente a -1.57 radianes (resultado del primer escenario).

$$Jq\ddot{}+kq\dot{}+mga\ \mathbf{sin}(\ q)=\tau$$



Gráfica 11. *rqt plot*. Escenario c) función sin.



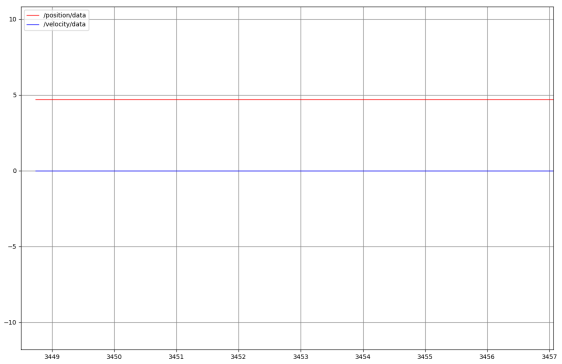
Gráfica 12. *simulink*. Escenario c) función sin.

El comportamiento de las gráficas anteriores vuelve a ser bastante diferente, en este caso igualmente se mantienen valores constantes a lo largo de toda la simulación (*rqt plot)*, la velocidad permanece en 0 debido de que se encuentra en un estado de reposo, en este caso de acuerdo a la función trigonométrica del seno la posición del péndulo en la que se encontraría en un estado de equilibrio inestable (mismo caso mostrado en la gráfica 5) de acuerdo al marco de referencia del sistema es en 180° o pi radianes.

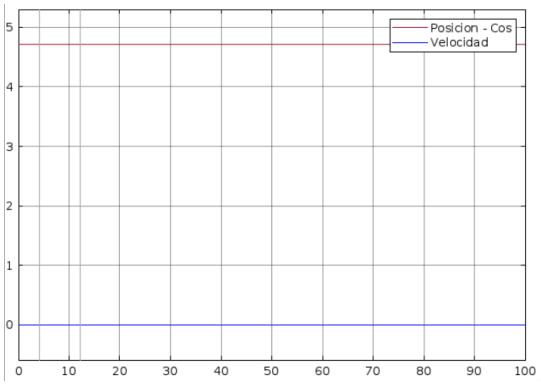
Al estar en este estado de equilibrio inestable es que la velocidad a lo largo de la simulación es de 0 debido a que no se presenta ningún movimiento o pequeña disrupción que altere este equilibrio.

d) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = 1.5*pi, x2 = 0.0

$$Jq\ddot{}+kq\dot{}+mga\ \mathbf{cos}(\ q)=\tau$$



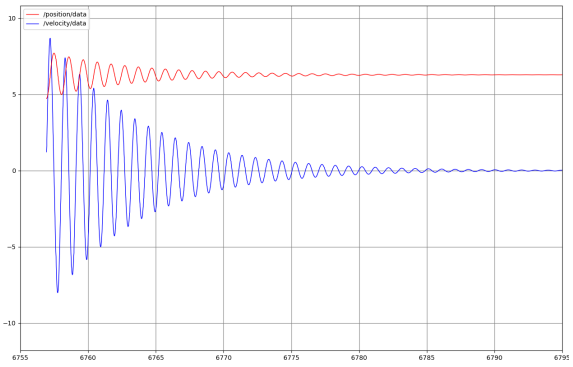
Gráfica 13. *rqt plot*. Escenario d) función cos.



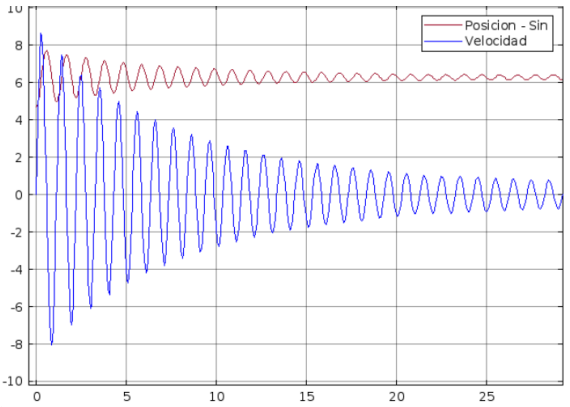
Gráfica 14. *simulink*. Escenario d) función cos.

En este cuarto caso el comportamiento de la simulaciones en *Simulink* y *rqt plot* muestra ser idéntico tanto para la posición como para la velocidad, en este escenario el valor inicial de la posición es de 1.5 pi que al hacer la conversión a grados resulta en 270° que al igual que en el segundo escenario resulta ser un estado de equilibrio, sin embargo este es estable e inmutable (a menos de que existan condiciones ambientales). Al no haber un cambio de movimiento es por lo mismo que el valor de la velocidad se mantiene constante en 0.

$$\boldsymbol{Jq\ddot{}+kq\dot{}+mga}\ \mathbf{sin}(\boldsymbol{q})=\boldsymbol{\tau}$$



Gráfica 15. *rqt plot*. Escenario d) función sin.



Gráfica 16. *simulink*. Escenario d) función sin.

Conclusiones

Tras la realización del reto se lograron los objetivos planteados al analizar el comportamiento de un brazo robótico con ecuaciones de Euler-Lagrange. Sin embargo, la simulación se basó en valores ideales, lo que limitó su precisión para representar el movimiento en situaciones del mundo real. Una posible solución o mejora para abordar estas limitaciones podría ser la inclusión de factores adicionales en la simulación para hacerla más realista. Esto podría implicar la consideración de variables como la fricción, la masa de los objetos manipulados, y otros factores ambientales que podrían influir en el comportamiento del brazo robótico. Además, se podría explorar la posibilidad de validar los resultados de la simulación mediante pruebas físicas en un entorno controlado, lo cuál podría darnos una representación más precisa y completa del comportamiento. Sin embargo para fines de simulación e introducción este reto fué de mucha utilidad para comenzar a familiarizarnos con la representación en espacios de estado de un sistema dinámico, así como la utilización de *RVIZ.*

Apéndices

```
#!/usr/bin/env python
import rospy
import logging
import math
import numpy as np
from std_msgs.msg import Float32
from sensor_msgs.msg import JointState

#Declare Variables to be used


class SLM_sim_Node:
    def __init__(self):
```

```
        self.logger = logging.getLogger(__name__)
    # Setup Variables to be used
        self.k = 0.01
        self.m = 0.75
        self.l = 0.36
        self.g = 9.8
        self.Tau = 0.0

        self.x1 = 0.0
        self.x2 = 0.0

        self.a = self.l/2
        self.j = (4/3)*self.m*self.a*self.a

        self.joint_pub = rospy.Publisher("/joint_states", JointState,
queue_size=10)
        self.vel_pub = rospy.Publisher("/velocity", Float32,
queue_size=10)
        self.pos_pub = rospy.Publisher("/position", Float32,
queue_size=10)
        self.init_joints()

    def init_joints(self):
        self.contJoints = JointState()

        self.contJoints.header.frame_id = "link1"
        self.contJoints.header.stamp = rospy.Time.now()
        self.contJoints.name.extend(["joint2"])

        self.contJoints.position.extend ([0.0])
        self.contJoints.velocity.extend ([0.0])
        self.contJoints.effort.extend ([0.0])


    def wrap_to_Pi(self,theta):
        result = np.fmod((theta + np.pi),(2 * np.pi))
        if(result < 0):
            result += 2 * np.pi
        return result - np.pi


if __name__=='__main__':
    #Initialise and Setup node
    rospy.init_node("SLM_Sim")
    SLM = SLM_sim_Node()
```

```
    loop_rate = rospy.Rate(rospy.get_param("~node_rate",100))
    dt = rospy.get_param("~dt",0.01)


    print("The SLM sim is Running")
    try:
        while not rospy.is_shutdown():

            SLM.x1 += SLM.x2 * dt
            x2_dot = ((-SLM.m * SLM.g * SLM.a * math.sin(SLM.x1)) /
SLM.j ) - ((SLM.k * SLM.x2) / SLM.j) + (SLM.Tau / SLM.j)
            SLM.x2 += x2_dot * dt
            SLM.contJoints.header.stamp = rospy.Time.now()
            t = rospy.Time.now().to_sec()

            #Topic to show in RVIZ the movement
            SLM.contJoints.position[0] = SLM.wrap_to_Pi(SLM.x1)
            SLM.joint_pub.publish(SLM.contJoints)
            #Topics to graph position and velocity in RQT plot
            SLM.vel_pub.publish(SLM.x2)
            SLM.pos_pub.publish(SLM.x1)


            loop_rate.sleep()

    except rospy.ROSInterruptException:
        pass #Initialise and Setup node
```