Campus Puebla

Materia

Integración de Robótica y Sistemas Inteligentes TE3003B

Challenge I

Integrantes

José Jezarel Sánchez Mijares A01735226

Antonio Silva Martínez A01173663

Dana Marian Rivera Oropeza A00830027

Fecha: 12 de abril de 2024

Contenidos

Resumen

Dentro de este reporte se va a desarrollar el planteamiento del problema, la solución propuesta y la metodología detrás de la solución, como puede ser información adicional para la comprensión del mini challenge, así como un análisis de los resultados obtenidos dentro de nuestra solución, definimos paso a paso los elementos involucrados en este mini challenge e integramos información que consideramos pertinente para el desarrollo del mismo.

Objetivos

Para este primer mini challenge se espera que en nuestra solución cumplamos con los siguientes objetivos:

- Manipulación de simulación en RViz
- Creación de un nodo “sli_sim.py”
- Implementación de los parámetros y ecuaciones del sistema dinámico en la simulación
- Generar las señales del movimiento usando “rqt_plot” para visualizar los cambios en la simulación

Introducción

Semanalmente se nos van a presentar diferentes mini challenges por parte de Manchester Robotics, se nos introduce a los temas que vamos a abordar dentro del mini challenge para darnos una idea más clara de lo que necesitamos conocer y hacer para llegar a la solución deseada, en este caso se nos pidió que realizáramos una simulación utilizando RViz basándonos en un archivo donde ya existía la construcción del modelo a simular, nuestra tarea consistía en la creación de un nodo adicional donde se implemente la ecuación que define el movimiento de péndulo que se debe agregar al brazo robótico proporcionado al igual que la implementación de los valores iniciales necesarios para el funcionamiento de la simulación.
A partir de esta premisa consideramos que los siguientes conceptos son necesarios para construir la solución del mini challenge:

RViz - “ROS visualization” es una herramienta de ROS para la visualización en 3D de simulaciones, entre sus cualidades se encuentra la posibilidad de manipular la simulación, con esto nos referimos a generar cambios en el entorno simulado, en las condiciones iniciales o añadir elementos a la misma simulación, de igual manera es permisivo en la edición de la visualización que se tiene en la simulación.

Sistemas Dinámicos - Se refiere a modelos matemáticos que representan la variación de una función a lo largo del tiempo, conformados por diferentes variables cuyos valores en un

momento dado son capaces de determinar el estado del sistema que es la información mínima requerida para conocer el comportamiento a futuro del sistema.

Representaciones del estado - Representa una forma de modelar el comportamiento de un sistema por medio de la relación entre las diferentes variables que intervienen en su comportamiento así como las ecuaciones diferenciales que las relacionan entre sí, estas normalmente son usadas para representar sistemas dinámicos, es decir aquellos con cambios a lo largo del tiempo.

Péndulo - Sistema mecánico que en su forma más simple es representado como una masa unida a un hilo de peso despreciable de tal manera que al momento de mover esta masa fuera de su punto de equilibrio se inicie un movimiento oscilatorio dentro del plano vertical. Este concepto es relevante en la solución del reto ya que la ecuación de estada usada para representar el movimiento está basada en este principio, es decir se modelan las variables que influyen en que la simulación genere un movimiento oscilatorio en el joint del brazo robótico considerando la influencia de las variables en el mismo.

Solución del challenge

Creamos un script en python “slm_sim.py” donde por medio de funciones se va a realizar lo siguiente:

- Declaración de variables y valores iniciales para la simulación
- Declaración de posición inicial en los nodos
- Solución de la ecuación de estado del sistema dinámico
- Comunicación con RViz

En esta sección del código iniciamos declarando los parámetros iniciales para la simulación y nos suscribimos al tópico “joint_states” mediante el cual vamos a mandar la información

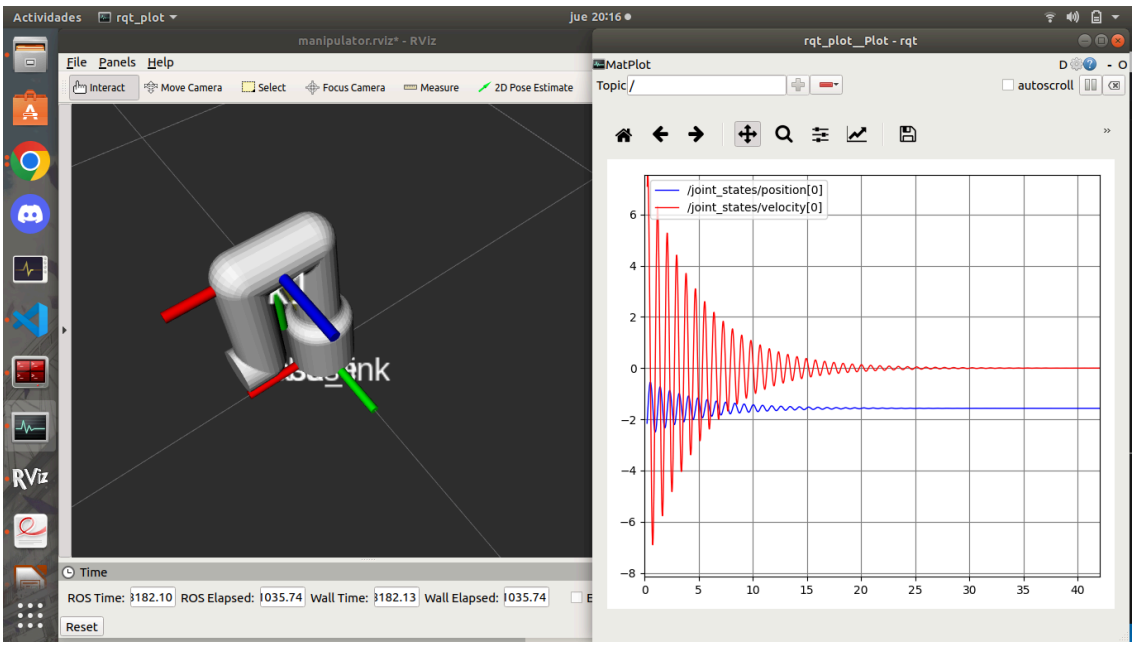
```
class PendulumSimulator:
    def __init__(self):
        #Initialize
        rospy.init_node("pendulum_sim")

        #Parameters
        self.k = 0.01  # Damping coefficient
        self.m = 0.75  # Mass
        self.l=0.36
        self.g = 9.8  # Gravity
        self.tao=0.0
        self.a = self.l/2
        self.j = (4/3) *self.m *self.a**2
        self.x1=0.0
        self.x2=0.0
        self.loop_rate = rospy.Rate(rospy.get_param("~node_rate",100))
        self.first=True
        ##Publishers
        self.pub_custom_joint_state = rospy.Publisher('/joint_states', JointState, queue_size=10)
```

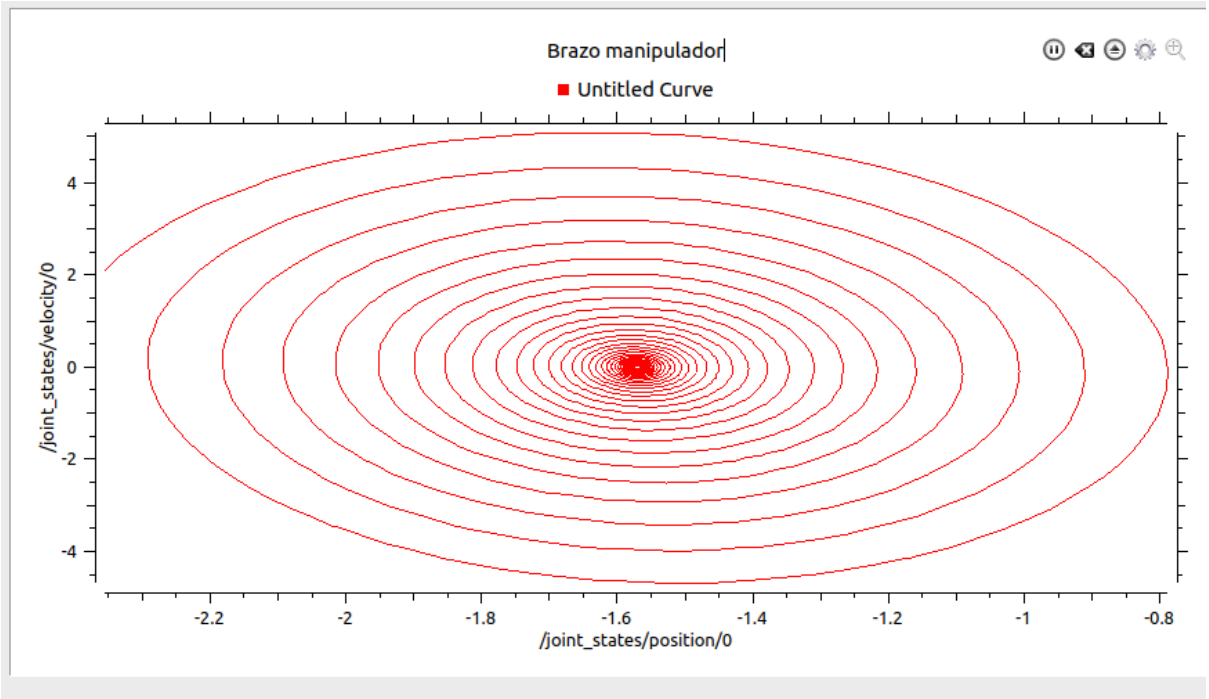
Dentro de esta función obtenemos el tiempo actual y realizamos el cálculo de x1, x2 y x2_dot, estas variables nos van a proporcionar los valores de posición y velocidad con respecto al tiempo calculado, esto alimenta a la simulación por medio del nodo al que nos suscribimos anteriormente “joint_states”

```python
def simulate(self):
    self.init_joints()

    while not rospy.is_shutdown():
        current_time = rospy.Time.now().to_sec()  # Get current time

        if self.first:
            self.previous_time = current_time
            self.first = False
        else:
            dt = current_time - self.previous_time  # Calculate time difference
            self.previous_time = current_time
            self.x1 += self.x2 * dt
            self.x2_dot = (1 / self.j) * (self.tao - self.m * self.g * self.a * np.cos(self.x1) - self.k * self.x2)
            self.x2 += self.x2_dot*dt

            # Update message
            self.msg.header.stamp = rospy.Time.now()
            self.msg.position[0] = self.wrap_to_Pi(self.x1)
            self.msg.velocity[0] = self.x2
            self.pub_custom_joint_state.publish(self.msg)
            self.loop_rate.sleep()
```

Resultados

Al iniciar la simulación, podemos apreciar como el brazo robótico va describiendo movimientos circulares que paulatinamente van disminuyendo en intensidad hasta parar por completo y regresar al punto de equilibrio, su estado inicial, esto se da ya que no hay una fuerza externa que lo esté afectando, entonces se toma el impulso del cambio de posición inicial y se generan oscilaciones que se ven en las señales, el tamaño es el mismo en ambos lados del límite de cero en “y” porque como no afectan fuerzas externas en el modelo la fuerza máxima que alcanza en un lado del equilibrio es idéntica en el otro y como no hay fuerzas que lo impulsen a continuar con el movimiento, este va disminuyendo hasta volver al punto de equilibrio.





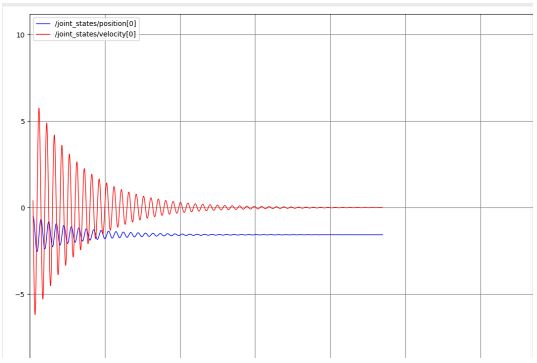
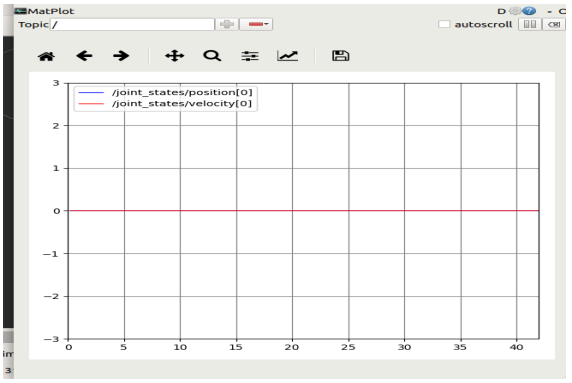
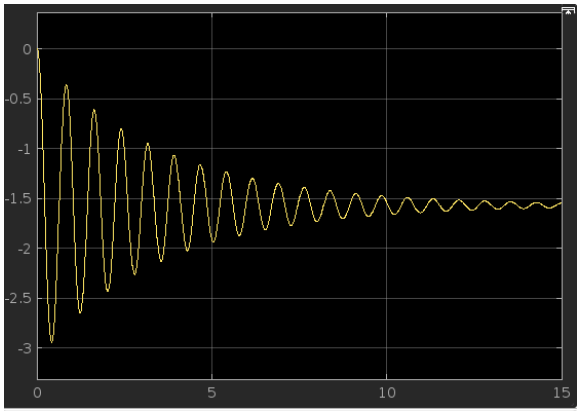
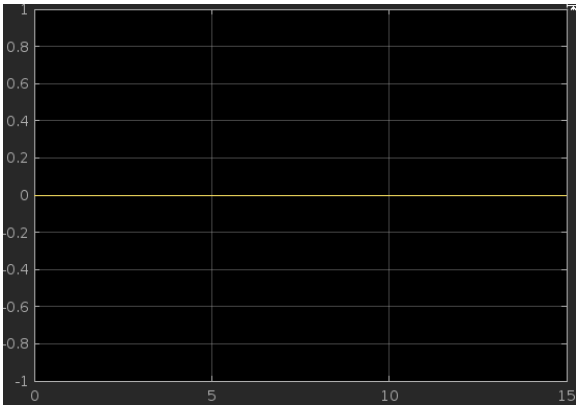
Conclusiones

Después de realizar el desarrollo de la solución del mini challenge logramos cumplir con los objetivos planteados ya que logramos establecer la conexión entre el nodo creado y RViz para alimentar la simulación de manera correcta, de igual manera pudimos visualizar el movimiento en las gráficas dando lugar a un análisis más profundo de los elementos que afectan al movimiento en la simulación.

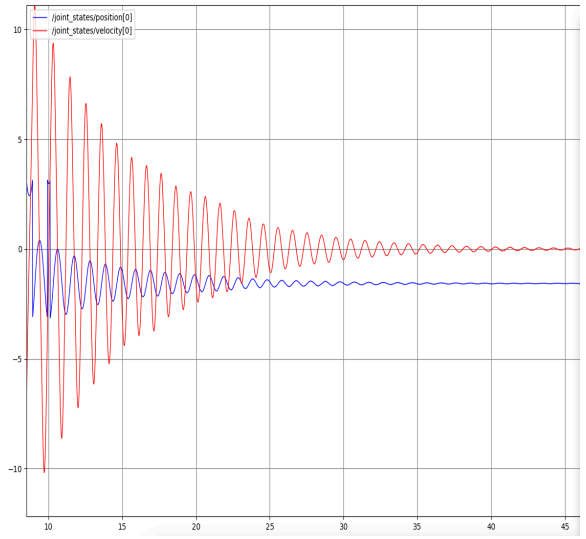
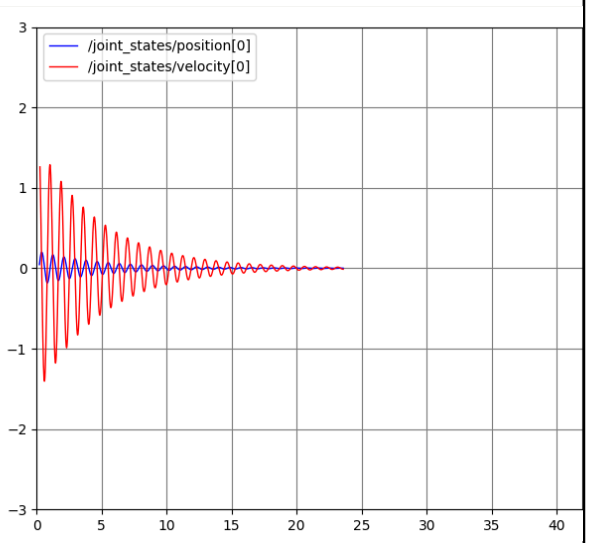
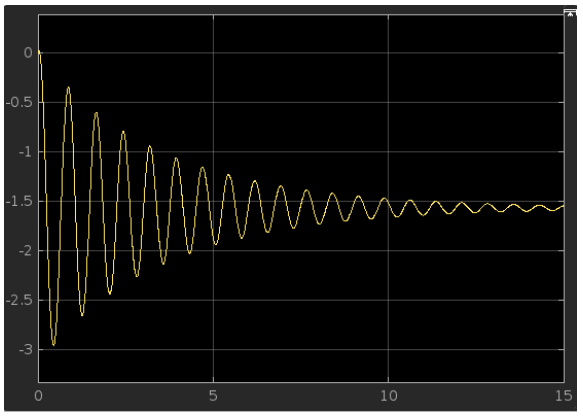
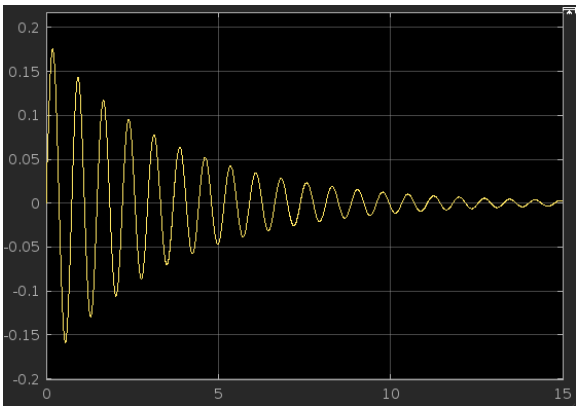
Anexo

Señales generadas por las simulaciones

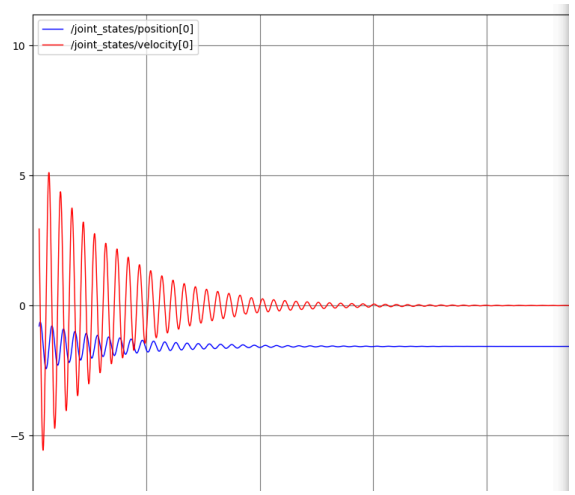
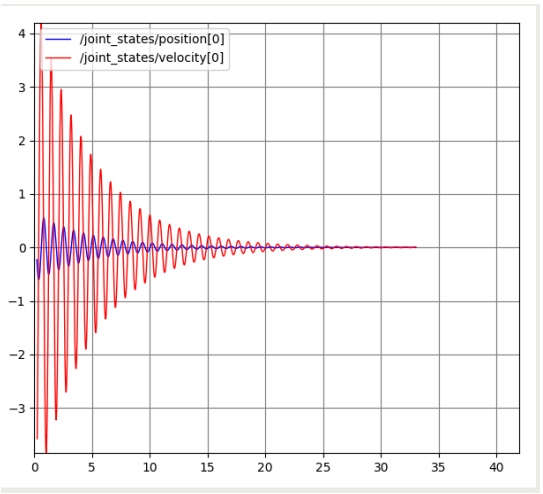
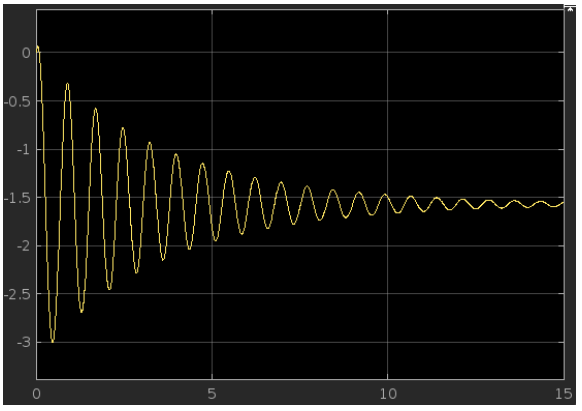
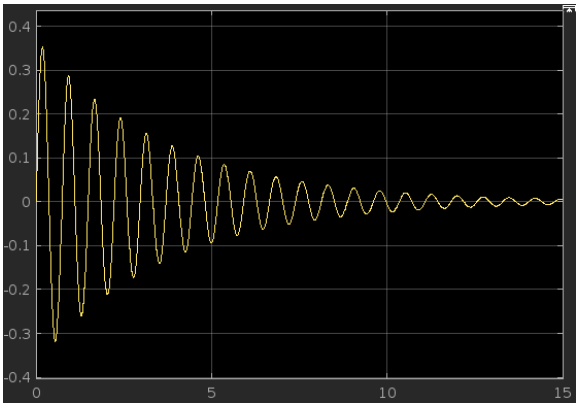
a) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = 0*pi, x2 = 0.0

<table>
<tr><td colspan="2">ROS</td></tr>
<tr><td>Cos</td><td>Sin</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Matlab</td></tr>
<tr><td>Cos</td><td>Sin</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

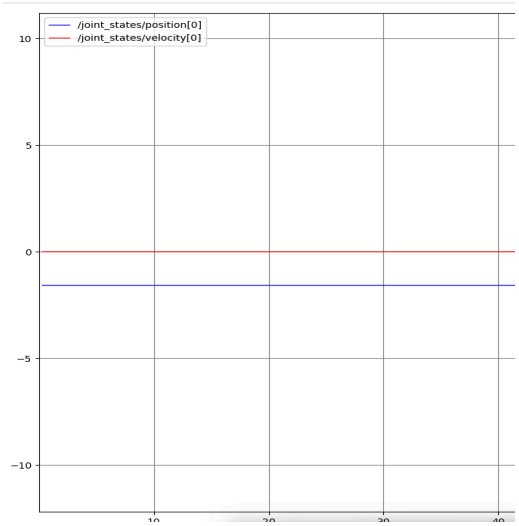
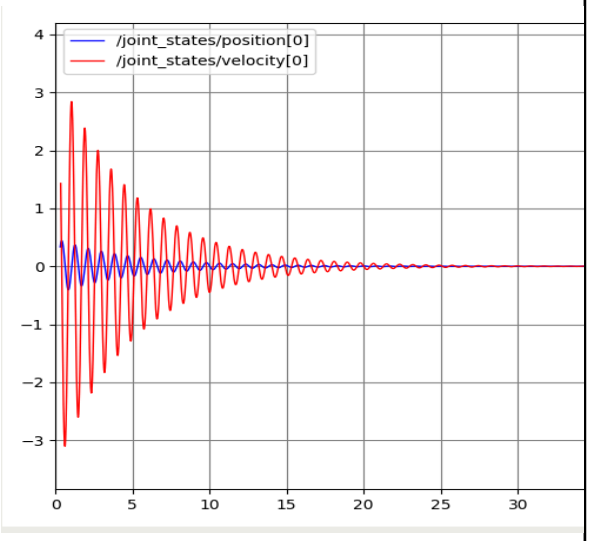
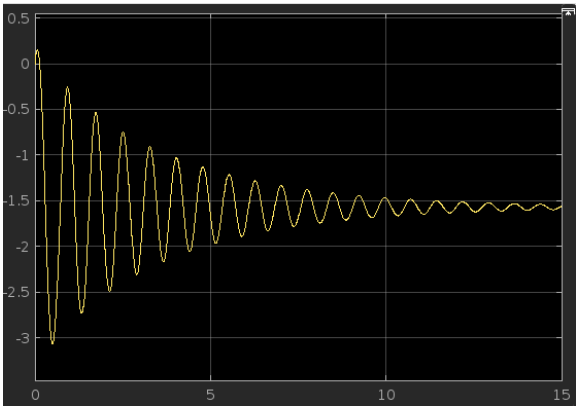
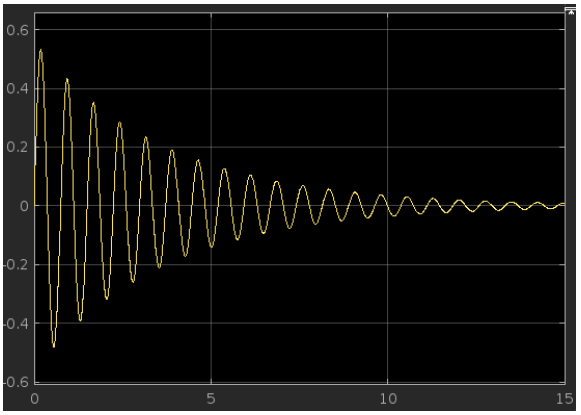
b) k = 0.01, m = 0.75, l = 0.36 , g = 9.8, Tau = 0.0, x1 = pi/2, x2 = 0.0

<table>
<tr><td colspan="2">ROS</td></tr>
<tr><td>Cos</td><td>Sin</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Matlab</td></tr>
<tr><td>Cos</td><td>Sin</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

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

<table>
<tr><th colspan="2">ROS</th></tr>
<tr><th>Cos</th><th>Sin</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th colspan="2">Matlab</th></tr>
<tr><th>Cos</th><th>Sin</th></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

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

ROS	
Cos	Sin
	
Matlab	
Cos	Sin
	

Análisis comparativo de la respuesta del péndulo (Matlab vs ROS)

En todos los casos las señales regresadas por las cuatro simulaciones son las mismas, es decir no hay diferencias apreciables entre las señales al momento de realizarse en cualquiera de las dos plataformas lo que nos deja ver que ambos son modelos confiables por la redundancia en sus respuestas, siendo la única excepción el último caso donde difieren los valores regresados en ROS y los de Matlab ya que nos muestra una lectura igual a cero en ROS, sin embargo ese resultado se puede considerar una anomalía del modelo ya que en el resto de los casos son resultados idénticos.

Referencias

Automatic Addison. (n.d.). What is the Difference Between RViz and Gazebo?. Retrieved from https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/

UNIR. (2021). Sistemas Dinámicos: Matemáticas. Retrieved from https://www.unir.net/ingenieria/revista/sistemas-dinamicos-matematicas/

Centro de Estudios Interdisciplinarios en Física (2013). Péndulo Simple. Retrieved from http://fisica.ciens.ucv.ve/proyectosfisica/PenduloSimple/Contenido.html