TRABAJO DE GRADO

Opción Pasantía en Investigación o Proyección social.

Comparación del desempeño de un controlador PID convencional y un controlador PID adaptativo basado en lógica difusa tipo Mamdani para el seguimiento de trayectorias articulares en un robot manipulador planar de dos grados de libertad

Corporación Universitaria Remington.

Ingenierías.

Ingeniería de Sistemas

Autor: Lucas José Beltrán Pacheco.

Tutor Extarno: Dr. José Román García Martínez.

Asesora: Gloria Amparo Lora Patiño

Opción de Trabajo de grado Pasantía en investigación.

2026

Agradecimientos

En primer lugar, agradezco a Dios por darme la salud, la fortaleza y la sabiduría para poder culminar esta etapa de mi formación profesional. Durante todo este proceso sentí su guía, la cual fue muy importante para afrontar cada uno de los retos que se presentaron.

Quiero expresar mi agradecimiento al Dr. José Román García Martínez por su orientación, disposición y acompañamiento durante el desarrollo de esta pasantía de investigación. Sus conocimientos, recomendaciones y observaciones fueron de gran ayuda para llevar a cabo este proyecto y, al mismo tiempo, aportaron mucho a mi formación tanto académica como profesional.

Agradezco también a la Corporación Universitaria Remington y a la Facultad de Ingenierías por los conocimientos adquiridos, los recursos brindados y el apoyo recibido durante mi proceso de formación como estudiante de Ingeniería de Sistemas.

A mi familia, por estar siempre presente, por su apoyo, su paciencia y la confianza que depositaron en mí durante todo este camino. Su respaldo fue muy importante para poder alcanzar esta meta.

Finalmente, agradezco a todas las personas que, de una u otra forma, hicieron parte de este proceso y aportaron con su apoyo o sus consejos para que este trabajo pudiera llegar a buen término.

Tabla de Contenidos

Resumen 5
Palabras clave 6
Objetivos 9
Metodología 10
1. Enfoque, diseño y alcance de la investigación 10
2. Metodología general por etapas 11
2.1. Qué se hizo 12
2.2. Cómo se hizo el modelado cinemático con Denavit-Hartenberg 12
2.3. Cómo se obtuvo la cinemática inversa 14
2.4. Cómo se construyó el modelo dinámico con Euler-Lagrange 15
2.5. Herramientas y verificación del modelo 18
3. Objetivo 2: Desarrollo de los controladores PID y PID adaptativo difuso 19
3.1. Qué se hizo 19
3.2. Cómo se hizo el controlador PID convencional 20
3.3. Cómo se hizo el controlador PID adaptativo difuso 21
3.4. Compensación gravitacional 22
3.5. Validación del controlador 23
3.6. Resultado obtenido en esta etapa 24
4. Objetivo 3: Implementación e integración en Python y PyBullet 25
4.1. Qué se hizo 25
4.2. Cómo se hizo la implementación 25
4.3. Herramientas utilizadas 27
4.4. Verificación de la implementación 27
4.5. Resultado de esta etapa 28
5. Objetivo 4: Comparación y validación de resultados 29
5.1. Qué se hizo 29
5.2. Criterios de validación considerados 30
5.3. Cómo se validaron los resultados 31
5.4. Resultado obtenido de esta etapa 31
Conclusiones. 36
Referencias 37

Lista de Tablas

Tabla 1. Parámetros físicos nominales del manipulador 2R utilizados en la simulación.....32
Tabla 2. Comparación entre PID clásico y PID difuso para un cuadrado.....33
Tabla 3. Comparación entre PID clásico y PID difuso para un circulo 33

Lista de figuras

Figura 1. Diagrama de etapas del desarrollo de la metodología 12
Figura 2. Código de la función de cinemática inversa del manipulador planar 2R. 15
Figura 3. Implementación de la cinemática directa del manipulador 2R, Implementación de la matriz de inercia del modelo dinámico del manipulador 2R e Implementación del vector de gravedad del modelo dinámico del manipulador 2R. 17
Figura 4. Proceso de obtención del modelo cinemático y dinámico del manipulador planar 2R 19
Figura 5. Visualización de posición del robot y aplicación de cinemática inversa. 19
Figura 6. Implementación del controlador PID convencional con compensación gravitacional. 21
Figura 7. Implementación del ajuste adaptativo de las ganancias PID mediante lógica difusa tipo Mamdani 22
Figura 8. Implementación del vector de compensación gravitacional del manipulador 2R. 23
Figura 9. Incorporación de la compensación gravitacional a la señal de control. 23
Figura 10. Resultados numéricos del desempeño de los controladores PID convencional y PID adaptativo difuso 24
Figura 11. Comparación del seguimiento de trayectoria entre el PID convencional y el PID adaptativo basado en lógica difusa en un tramo de la trayectoria cuadrada. 25
Figura 12. Diagrama de funcionamiento de los controladores PID convencional y PID adaptativo basado en lógica difusa. 25
Figura 13. Implementación de la ley de control PID y cálculo del torque articular. 26
Figura 14. Implementación del modelo dinámico y cálculo de la aceleración articular. 27
Figura 15. Visualización de la ejecución del robot manipulador planar 2R en PyBullet 29
Figura 16. Flujo de implementación y comunicación entre la trayectoria, el controlador, el modelo dinámico y PyBullet 29
Figura 17. Comparación de seguimiento de trayectoria en un cuadrado 34
Figura 18. Comparación de seguimiento de trayectoria en un circulo 34
Figura 19. Pybullet en ejecución 35

Resumen

Este proyecto busca desarrollar y comparar dos estrategias de control para un robot manipulador planar de dos grados de libertad (2R): un controlador PID convencional y un controlador PID adaptativo basado en lógica difusa tipo Mamdani. En primer lugar, se realizará el modelado cinemático y dinámico del robot utilizando la convención de Denavit-Hartenberg y el formalismo de Euler-Lagrange. Como resultado, se desea obtener la ecuación dinámica del sistema en su forma matricial $D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) = \tau$, donde la matriz de inercia $D(q)$, los términos de Coriolis y centrífugos $C(q, \dot{q})$ y el vector de gravedad $G(q)$describen el comportamiento dinámico del manipulador.

A partir de este modelo se desarrollará, en primera instancia, un controlador PID convencional y, posteriormente, un controlador PID adaptativo basado en lógica difusa tipo Mamdani. El sistema difuso empleará como variables de entrada el error de posición y la derivada del error, ambos normalizados. Asimismo, se definirán siete funciones de pertenencia para cada variable y una base de 49 reglas difusas que permitirán ajustar las ganancias del controlador PID durante la simulación. Adicionalmente, se incorporará compensación gravitacional (*Gravity Feedforward*) con el propósito de reducir el esfuerzo de control requerido durante el movimiento del robot.

La validación de ambos controladores se llevará a cabo mediante una simulación en Python utilizando PyBullet, lo que permitirá evaluar su desempeño en un entorno físico cercano al comportamiento real. Durante las pruebas se analizará el seguimiento de trayectorias y diversos indicadores de desempeño, entre ellos el error cuadrático medio (RMSE), el error absoluto integrado

(IAE), el sobreimpulso, el tiempo de establecimiento y el esfuerzo de control. La comparación permitirá establecer las diferencias de desempeño entre ambas estrategias de control con base en indicadores como el RMSE, el IAE, el sobreimpulso, el tiempo de establecimiento y el esfuerzo de control. De igual manera, se prevé que responda mejor frente a las no linealidades propias del manipulador y a las perturbaciones que puedan presentarse durante las simulaciones.

Finalmente, se espera que las pruebas desarrolladas en PyBullet permitan validar el funcionamiento del modelo y comprobar que esta estrategia de control puede mejorar el comportamiento dinámico del manipulador bajo diferentes condiciones.

Palabras clave

Robot Manipulador

Lógica difusa

Control PID adaptativo

Modelado dinámico (Euler-Lagrange)

Simulación PyB

Problemática abordada en la pasantía de investigación

Los robots manipuladores de dos grados de libertad son muy utilizados para estudiar diferentes estrategias de control porque presentan un comportamiento dinámico no lineal. En este tipo de sistemas intervienen factores como el acoplamiento entre las articulaciones, las fuerzas centrífugas, los efectos de Coriolis y la gravedad, haciendo que controlar el movimiento del robot no sea una tarea sencilla. Cuando existen cambios en el modelo o aparecen perturbaciones externas, mantener un seguimiento preciso de la trayectoria puede resultar complicado (Craig, 2018; Spong et al., 2020).

Uno de los controladores más utilizados en la industria sigue siendo el PID, principalmente porque es fácil de implementar y ofrece buenos resultados en muchas aplicaciones. Aun así, cuando se trabaja con sistemas no lineales, como ocurre con los robots manipuladores, el uso de ganancias fijas puede hacer que el desempeño disminuya dependiendo de las condiciones de operación (Åström & Hägglund, 2006).

Como alternativa al controlador PID convencional, se propone el empleo de la lógica difusa tipo Mamdani. Este tipo de control utiliza reglas del tipo **Si–Entonces**, construidas a partir del comportamiento esperado del sistema y de la experiencia del diseñador. Una de sus principales ventajas es que permite modificar la acción de control cuando cambian las condiciones del sistema, sin depender completamente de un modelo matemático exacto (Zadeh, 1965; Mamdani & Assilian, 1975).

Investigaciones recientes continúan reportando mejoras en el seguimiento de trayectorias mediante controladores difusos adaptativos aplicados a manipuladores robóticos, especialmente en presencia de incertidumbres y dinámicas no lineales (Ma et al., 2025).

En este trabajo se propone desarrollar un controlador PID adaptativo basado en lógica difusa tipo Mamdani para un robot manipulador planar de dos grados de libertad. Primero se obtiene el modelo dinámico del robot utilizando el formalismo de Euler-Lagrange Posteriormente, se analizará su comportamiento frente a un controlador PID convencional mediante simulaciones realizadas en Python y PyBullet. Para esta comparación se considerarán el seguimiento de trayectorias, el error obtenido y el esfuerzo de control en cada caso, con el propósito de evaluar el desempeño de ambos controladores en un entorno dinámico antes de una posible implementación experimental.

A partir de esto surge la siguiente pregunta de investigación:

¿En qué medida un controlador PID adaptativo basado en lógica difusa tipo Mamdani mejora el seguimiento de trayectorias y el desempeño de un robot manipulador de dos grados de libertad, en comparación con un controlador PID convencional?

Este proyecto busca aportar una metodología para el diseño y evaluación de estrategias de control aplicadas a robots manipuladores de dos grados de libertad. Para ello se combina el modelado dinámico mediante Euler-Lagrange, un controlador PID adaptativo basado en lógica difusa y la simulación física utilizando PyBullet.

Se espera que los resultados obtenidos puedan servir como apoyo en proyectos relacionados con robótica y control automático, especialmente en actividades académicas o de investigación. También se considera que este proyecto puede servir como punto de partida para futuros trabajos donde se quiera implementar este tipo de control en robots de bajo costo o en plataformas experimentales.

Se eligió utilizar lógica difusa porque permite ajustar las ganancias del controlador según el comportamiento del robot durante la simulación. Esto puede ofrecer un mejor desempeño frente a un PID convencional cuando el sistema presenta no linealidades o cambios en sus condiciones de funcionamiento.

Este proyecto se limita al desarrollo y evaluación de los controladores dentro de un entorno de simulación utilizando Python y PyBullet. No se contempla una implementación sobre un robot físico, ya que el objetivo principal es analizar el comportamiento de ambas estrategias de control antes de llevarlas a un sistema real.

Las pruebas se realizarán utilizando trayectorias de referencia previamente definidas y se evaluarán indicadores como el RMSE, el IAE, el sobreimpulso, el tiempo de establecimiento y el esfuerzo de control. Por lo tanto, aspectos como la validación experimental, el funcionamiento en tiempo real o la aplicación en robots con un mayor número de grados de libertad quedan fuera del alcance de este proyecto.

Objetivos

Objetivo general:

Evaluar y comparar el desempeño de un controlador PID convencional y un controlador PID adaptativo basado en lógica difusa tipo Mamdani en el seguimiento de trayectorias articulares de un robot manipulador planar de dos grados de libertad.

Objetivos específicos:

1. Obtener el modelo cinemático y dinámico del robot manipulador de dos grados de libertad utilizando la convención de Denavit-Hartenberg y el formalismo de Euler-Lagrange, con el fin de describir el comportamiento dinámico del sistema.

2. Desarrollar un controlador PID convencional y un controlador PID adaptativo basado en lógica difusa tipo Mamdani, incorporando compensación gravitacional (Gravity Feedforward) para mejorar el comportamiento del robot durante el seguimiento de trayectorias.

3. Implementar ambos controladores en Python e integrarlos con una simulación física en PyBullet para evaluar el comportamiento del manipulador bajo diferentes trayectorias de referencia.

4. Comparar el desempeño de ambos controladores mediante indicadores como el error cuadrático medio (RMSE), el error absoluto integrado (IAE), el tiempo de establecimiento, el sobreimpulso y el esfuerzo de control, con el fin de determinar las ventajas del controlador difuso frente al PID convencional.

Metodología

1. Enfoque, diseño y alcance de la investigación

La investigación se desarrolló bajo un enfoque cuantitativo, ya que el análisis se basó en los datos obtenidos a partir del modelo matemático del robot manipulador de dos grados de libertad y de los resultados obtenidos durante las simulaciones. Para evaluar el comportamiento del sistema se

utilizaron indicadores como el error de seguimiento, el esfuerzo de control y los tiempos de respuesta de cada controlador.

El estudio adoptó un diseño experimental basado en simulación, debido a que se compararon dos estrategias de control utilizando las mismas condiciones iniciales, las mismas trayectorias de referencia y los mismos parámetros dinámicos del robot. La principal diferencia entre las pruebas estuvo en la estrategia de control implementada: un controlador PID convencional y un controlador PID adaptativo basado en lógica difusa tipo Mamdani con compensación gravitacional.

El alcance del trabajo fue descriptivo, comparativo y explicativo. Primero se obtuvo el modelo cinemático y dinámico del manipulador y, posteriormente, se comparó el comportamiento de ambos controladores mediante indicadores como el error de seguimiento, el torque aplicado y la respuesta general del sistema durante las simulaciones. Esto sirvió analizar con mayor claridad las diferencias entre las dos estrategias de control.

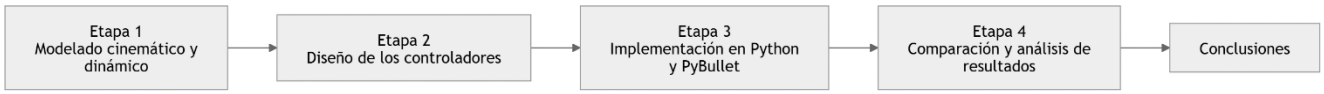
La validación se realizó en un entorno de simulación utilizando PyBullet, por lo que no se contempló una implementación sobre un robot físico. Con ello se obtuvo una base de análisis para evaluar el desempeño de los controladores antes de una posible implementación experimental.

2. Metodología general por etapas

La metodología del proyecto se organizó en cuatro etapas principales, siguiendo el orden en el que se desarrolló cada uno de los objetivos planteados. En cada etapa se fueron obteniendo los elementos necesarios para continuar con la siguiente fase del trabajo.

1. Modelado cinemático y dinámico del robot manipulador de dos grados de libertad.
2. Diseño e implementación de los controladores PID convencional y PID adaptativo basado en lógica difusa con compensación gravitacional.
3. Desarrollo de la simulación e integración de los modelos en Python y PyBullet.
4. Comparación del desempeño de ambos controladores mediante indicadores cuantitativos y análisis de los resultados.

Figura 1. Diagrama de etapas del desarrollo de la metodología



Objetivo 1: Modelado cinemático y dinámico del robot manipulador 2R

2.1. Qué se hizo

En esta etapa se obtuvo el modelo cinemático y dinámico del robot manipulador planar de dos grados de libertad. Para ello se definieron los parámetros geométricos y físicos del sistema, se establecieron las transformaciones entre los eslabones mediante la convención de Denavit-Hartenberg y, posteriormente, se obtuvo el modelo dinámico utilizando el formalismo de Euler-Lagrange.

2.2. Cómo se hizo el modelado cinemático con Denavit-Hartenberg

Para realizar el modelado cinemático, primero se definieron los sistemas de referencia del marco base, el primer eslabón, el segundo eslabón y el efector final. Como el robot corresponde a un

manipulador planar de dos grados de libertad, se trabajó con dos articulaciones rotacionales que se mueven en un mismo plano, mientras que sus ejes de giro permanecen perpendiculares al plano de movimiento. A partir de esta configuración se construyó el modelo cinemático utilizando la convención de Denavit-Hartenberg.

Los parámetros geométricos definidos para el manipulador fueron la longitud del primer eslabón (l_1), la longitud del segundo eslabón (l_2), los ángulos articulares (q_1)y (q_2), además de considerar una torsión y un desplazamiento iguales a cero para ambas articulaciones, ya que el robot se mueve en un mismo plano.

Con estos parámetros se construyó la tabla de Denavit-Hartenberg mostrada a continuación.

Eslabón	a_i	α_i	d_i	θ_i
1	l_1	0	0	q_1
2	l_2	0	0	q_2

A partir de esta información se obtuvo la matriz de transformación homogénea entre marcos consecutivos:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Después de reemplazar los parámetros del manipulador 2R, se obtuvo la transformación total del sistema,

$$ {}^{0}T_2 = \quad {}^{0}T_1 \quad {}^{1}T_2 $$

a partir de la cual se calcularon las ecuaciones de la cinemática directa del efector final:

$$x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

Estas expresiones permitieron obtener la posición cartesiana del efector final a partir de los ángulos de las articulaciones y sirvieron como base para el desarrollo del modelo dinámico y de las simulaciones realizadas posteriormente.

2.3. Cómo se obtuvo la cinemática inversa

Para generar las referencias articulares a partir de una trayectoria en el espacio cartesiano, se obtuvo la cinemática inversa del manipulador 2R. A partir de una posición deseada (x, y), primero se calculó el término intermedio:

$$D = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

```python
def inverse_kinematics_2r(x, y, p, elbow="down"):
    l1 = p["l1"]
    l2 = p["l2"]

    D = (x**2 + y**2 - l1**2 - l2**2) / (2.0 * l1 * l2)
    D = np.clip(D, -1.0, 1.0)

    if elbow == "down":
        q2 = np.arctan2(-np.sqrt(1.0 - D**2), D)
    else:
        q2 = np.arctan2(np.sqrt(1.0 - D**2), D)
```

```
    q1 = np.arctan2(y, x) - np.arctan2(l2 * np.sin(q2), l1 + l2 * np.cos(q2))
    return np.array([q1, q2])
```

Figura 2. Código de la función de cinemática inversa del manipulador planar 2R.

Con este valor se obtuvo el ángulo $q_2$utilizando la relación trigonométrica del manipulador planar. Despues se calculó q_1a partir de las relaciones geométricas entre los dos eslabones y la posición del efector final. Para las simulaciones se trabajó con la configuración de **codo hacia abajo**, ya que fue la que se utilizó para generar las trayectorias de referencia.

2.4. Cómo se construyó el modelo dinámico con Euler-Lagrange

El modelo dinámico se obtuvo utilizando el formalismo de Euler-Lagrange. Primero se calcularon la energía cinética y la energía potencial del manipulador, teniendo en cuenta la masa de cada eslabón, la ubicación de sus centros de masa y sus respectivos momentos de inercia.

La energía cinética total del sistema quedó definida como:

$$T = T_1 + T_2$$

mientras que la energía potencial se obtuvo a partir del efecto de la gravedad sobre cada uno de los eslabones:

$$V = V_1 + V_2$$

Con estas dos expresiones se definió la función de Lagrange:

$$L = T - V$$

Después se aplicaron las ecuaciones de Euler-Lagrange para cada coordenada generalizada:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i$$

A partir de este procedimiento se obtuvo la ecuación dinámica del manipulador:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) = \tau$$

donde $D(q)$corresponde a la matriz de inercia, $C(q,\dot{q})$agrupa los efectos de Coriolis y centrífugos, $G(q)$representa el efecto de la gravedad y $F(\dot{q})$considera la fricción viscosa del sistema.

Las ecuaciones obtenidas fueron utilizadas durante las simulaciones para calcular el comportamiento dinámico del robot y servir como base para la implementación de los controladores evaluados en este trabajo.

```python
def forward_kinematics_2r(q, p):
    q1, q2 = q
    l1 = p["l1"]
    l2 = p["l2"]

    x = l1 * np.cos(q1) + l2 * np.cos(q1 + q2)
    y = l1 * np.sin(q1) + l2 * np.sin(q1 + q2)
    return np.array([x, y])
```

```python
def inertia_matrix(q, p):
    q1, q2 = q
    m1 = p["m1"]
    m2 = p["m2"]
```

```
    l1 = p["l1"]
    lc1 = p["lc1"]
    lc2 = p["lc2"]
    I1 = p["I1"]
    I2 = p["I2"]

    M11 = I1 + I2 + m1 * lc1**2 + m2 * (l1**2 + lc2**2 + 2.0 * l1 * lc2 * np.cos(q2))
    M12 = I2 + m2 * (lc2**2 + l1 * lc2 * np.cos(q2))
    M21 = M12
    M22 = I2 + m2 * lc2**2

    return np.array([[M11, M12], [M21, M22]])
```

```python
def gravity_vector(q, p):
    q1, q2 = q
    m1 = p["m1"]
    m2 = p["m2"]
    l1 = p["l1"]
    lc1 = p["lc1"]
    lc2 = p["lc2"]
    g = p["g"]

    G1 = (m1 * lc1 + m2 * l1) * g * np.cos(q1) + m2 * lc2 * g * np.cos(q1 + q2)
    G2 = m2 * lc2 * g * np.cos(q1 + q2)

    return np.array([G1, G2])
```

Figura 3. Implementación de la cinemática directa del manipulador 2R, Implementación de la matriz de inercia del modelo dinámico del manipulador 2R e Implementación del vector de gravedad del modelo dinámico del manipulador 2R.

2.5. Herramientas y verificación del modelo

El desarrollo y la verificación del modelo se realizaron utilizando **Python** como herramienta principal. Donde se empleó **NumPy** en los cálculos matriciales, **SciPy** para resolver las ecuaciones diferenciales del sistema, **Matplotlib** para visualizar los resultados obtenidos y **PyBullet** para validar el comportamiento del robot dentro de un entorno de simulación física.

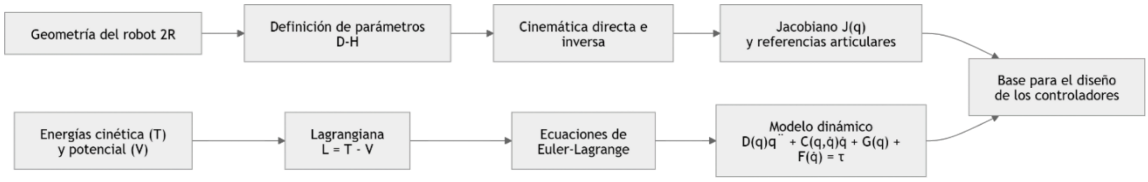
La verificación del modelo se hizo durante las diferentes etapas del desarrollo. Primero se revisó que las ecuaciones obtenidas fueran consistentes y que las variables de posición, velocidad, aceleración y torque mantuvieran unidades correctas. También se comprobó que la cinemática directa y la cinemática inversa entregaran resultados coherentes entre sí.

Finalmente se realizaron diferentes simulaciones para observar el comportamiento dinámico del manipulador y verificar que el robot pudiera seguir las trayectorias de referencia sin presentar inconsistencias importantes durante la simulación.

2.6. Resultado obtenido de esta etapa

Al finalizar esta etapa se obtuvo el modelo cinemático y dinámico del robot manipulador de dos grados de libertad. Estas ecuaciones sirvieron como base para implementar los controladores, realizar las simulaciones y desarrollar la comparación entre el PID convencional y el PID difuso.

Figura 4. Proceso de obtención del modelo cinemático y dinámico del manipulador planar 2R



2.7. Evidencia visual del funcionamiento del objetivo 1

Como evidencia del funcionamiento del modelo cinemático, se generó una figura [Figura 5] directamente desde la implementación del proyecto. En esta se muestra la posición del robot para unos ángulos articulares determinados y la posición obtenida al aplicar la cinemática inversa. Esto permitió comprobar que ambas relaciones cinemáticas son consistentes.

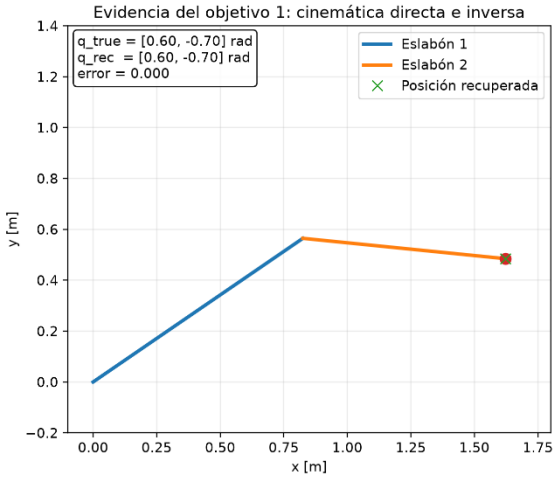


Figura 5. Visualización de posición del robot y aplicación de cinemática inversa.

3. Objetivo 2: Desarrollo de los controladores PID y PID adaptativo difuso

3.1. Qué se hizo

En esta etapa se desarrollaron dos estrategias de control para el robot manipulador. La primera

corresponde a un controlador PID convencional y la segunda a un controlador PID adaptativo basado en lógica difusa tipo Mamdani. Ambos se implementaron sobre el mismo modelo del robot para que la comparación se realizara bajo las mismas condiciones.

3.2. Cómo se hizo el controlador PID convencional

El controlador PID se implementó para cada una de las articulaciones del manipulador utilizando la expresión:

$$\tau = K_p e + K_d \dot{e} + K_i \int e \ dt$$

donde el error corresponde a la diferencia entre la posición deseada y la posición actual de cada articulación:

$$e = q_d - q$$

y su derivada se calculó como:

$$\dot{e} = \dot{q}_d - \dot{q}$$

Las ganancias iniciales se definieron mediante matrices diagonales, permitiendo controlar cada articulación de forma independiente. También se agregó un límite para la acción integral con el fin de evitar que esta creciera demasiado durante la simulación.

```python
e = qd - q
de = dqd - dq
```

```
integral_error += e * Ts
integral_error = np.clip(integral_error, -integral_limit, integral_limit)


tau = Kp @ e + Kd @ de + Ki @ integral_error


if useGravityFF:
    G_ff = gravity_vector(qd, p)
    tau = tau + G_ff
```

Figura 6. Implementación del controlador PID convencional con compensación gravitacional.

3.3. Cómo se hizo el controlador PID adaptativo difuso

Después de implementar el controlador PID convencional, se desarrolló un controlador adaptativo utilizando lógica difusa tipo Mamdani. Para ello se tomaron como entradas el error de posición y la derivada del error, ambos previamente normalizados para trabajar dentro del mismo rango.

A partir de estas variables, el sistema difuso ajustó en tiempo real las ganancias K_p, K_i y K_d de cada articulación. La idea fue que el controlador pudiera responder de una mejor manera segun el comportamiento del robot en cada instante.

```
```python
if useFuzzyGains and (t - lastFuzzyUpdate) >= fuzzyUpdateTime:
    eNorm1 = np.clip(e[0]/0.20, -1, 1)
    deNorm1 = np.clip(de[0]/0.50, -1, 1)


    eNorm2 = np.clip(e[1]/0.20, -1, 1)
    deNorm2 = np.clip(de[1]/0.50, -1, 1)
```
```

```
    kp1, ki1, kd1 = fuzzyPID.updateGains(KpBase[0,0], KiBase[0,0], KdBase[0,0], eNorm1, deNorm1)
    kp2, ki2, kd2 = fuzzyPID.updateGains(KpBase[1,1], KiBase[1,1], KdBase[1,1], eNorm2, deNorm2)

    Kp = np.diag([kp1, kp2])
    Ki = np.diag([ki1, ki2])
    Kd = np.diag([kd1, kd2])
```

Figura 7. Implementación del ajuste adaptativo de las ganancias PID mediante lógica difusa tipo Mamdani

En general, cuando el error era grande o cambiaba rápidamente, el sistema aumentaba la acción proporcional y derivativa. En cambio, cuando el error era pequeño, pero permanecía durante más tiempo, se reforzaba la acción integral para disminuir el error restante. Además, todas las ganancias se mantuvieron dentro de unos límites para evitar comportamientos inestables.

3.4. Compensación gravitacional

Como parte del controlador también se incorporó una compensación gravitacional, expresada por:

$$\tau_{ff} = G(q_d)$$

Este término se sumó a la señal de control para compensar el efecto del peso de los eslabones durante el movimiento. Con esto se buscó disminuir el esfuerzo que debía realizar el controlador y mejorar el seguimiento de las trayectorias, especialmente cuando la gravedad tenía mayor influencia sobre el manipulador.

```python
```

```
def gravity_vector(q, p):
    q1, q2 = q
    m1 = p["m1"]
    m2 = p["m2"]
    l1 = p["l1"]
    lc1 = p["lc1"]
    lc2 = p["lc2"]
    g = p["g"]

    G1 = (m1 * lc1 + m2 * l1) * g * np.cos(q1) + m2 * lc2 * g * np.cos(q1 + q2)
    G2 = m2 * lc2 * g * np.cos(q1 + q2)

    return np.array([G1, G2])
```

Figura 8. Implementación del vector de compensación gravitacional del manipulador 2R.

Y luego se incorporó a la ley de control como un feedforward adicional:

```python
if useGravityFF:
    G_ff = gravity_vector(qd, p)
    tau = tau + G_ff
```

Figura 9. Incorporación de la compensación gravitacional a la señal de control.

3.5. Validación del controlador

La validación se realizó mediante simulaciones utilizando las mismas trayectorias, las mismas condiciones iniciales y el mismo paso de integración para ambos controladores. En un caso se evaluó el PID convencional con ganancias fijas y, en el otro, el PID adaptativo junto con la compensación

gravitacional.

La comparación facilitó observar el comportamiento de cada estrategia, revisando principalmente el error de seguimiento, la respuesta del robot y el esfuerzo de control durante toda la simulación.

3.6. Resultado obtenido en esta etapa

Al finalizar esta etapa se obtuvieron dos controladores completamente funcionales y listos para trabajar junto con el modelo dinámico del robot. Esto permitió continuar con las simulaciones en PyBullet y realizar posteriormente la comparación entre ambas estrategias de control.

Como evidencia del funcionamiento del controlador del objetivo 2, se ejecutó la simulación bajo las mismas condiciones para el PID clásico y el PID difuso con compensación gravitacional. La figura 11 [Figura 11] presenta un tramo de la trayectoria cuadrada y permite comparar visualmente el seguimiento realizado por ambos controladores frente a la referencia. la figura [Figura 10] muestra el esquema de funcionamiento de las dos estrategias de control, desde las referencias articulares y el cálculo del error hasta la generación del torque de control. Los resultados mostraron una reducción del error de seguimiento con el controlador adaptativo.

```text
=== EVIDENCIA OBJETIVO 2 ===
RMSE q1 PID=0.006616 PID-Fuzzy=0.000228
RMSE q2 PID=0.001980 PID-Fuzzy=0.000132
RMSE cartesiano PID=0.007294 PID-Fuzzy=0.000199
Max |tau1| PID=47.518276 PID-Fuzzy=52.221645
Max |tau2| PID=31.610541 PID-Fuzzy=33.466877
```

Figura 10. Resultados numéricos del desempeño de los controladores PID convencional y PID adaptativo difuso

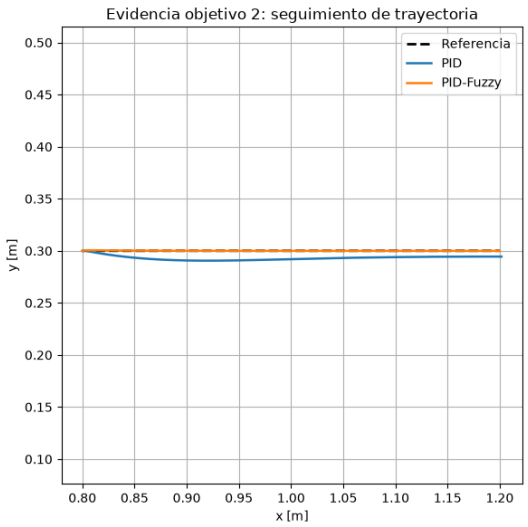
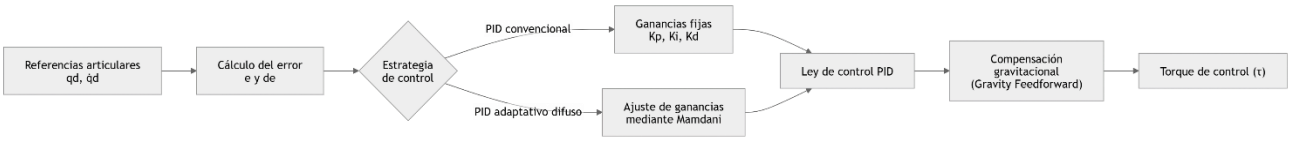


Figura 11. Comparación del seguimiento de trayectoria entre el PID convencional y el PID adaptativo basado en lógica difusa en un tramo de la trayectoria cuadrada.

Figura 12. Diagrama de funcionamiento de los controladores PID convencional y PID adaptativo basado en lógica difusa.



4. Objetivo 3: Implementación e integración en Python y PyBullet

4.1. Qué se hizo

En esta etapa se reunieron todas las partes del proyecto en un solo programa. Se integró el modelo cinemático, el modelo dinámico y los dos controladores desarrollados en Python. Después, la simulación se conectó con PyBullet para observar el movimiento del manipulador y comprobar que la respuesta fuera coherente con lo esperado.

4.2. Cómo se hizo la implementación

Primero se definió la trayectoria que debía seguir el robot. Esa trayectoria se generó en el espacio cartesiano y luego se transformó al espacio articular utilizando la cinemática inversa.

Con las referencias articulares ya calculadas, en cada instante de la simulación se obtuvo el error, la derivada del error y la parte integral. A partir de esa información, el controlador generó los torques necesarios para mover cada una de las articulaciones. Después esos torques se aplicaron al modelo dinámico del robot y las ecuaciones del movimiento se resolvieron numéricamente mediante un integrador de paso variable.

```python
e = qd - q
de = dqd - dq

integral_error += e * Ts
integral_error = np.clip(integral_error, -integral_limit, integral_limit)

tau = Kp @ e + Kd @ de + Ki @ integral_error
```

Figura 13. Implementación de la ley de control PID y cálculo del torque articular.

Gracias a eso fue posible obtener la evolución de las posiciones, velocidades, errores y torques durante toda la simulación.

```python
M = inertia_matrix(q, p)
C = coriolis_matrix(q, dq, p)
G = gravity_vector(q, p)
F = friction_vector(dq, p)

ddq = np.linalg.solve(M, tau - C @ dq - G - F)

dx[0:2] = dq
```

```
dx[2:4] = ddq

dx[4:6] = e

```

Figura 14. Implementación del modelo dinámico y cálculo de la aceleración articular.

Por otro lado, PyBullet se utilizó para visualizar el movimiento del manipulador. Esto se empleó como base para revisar si el comportamiento del robot realmente coincidía con los resultados obtenidos en la simulación y detectar cualquier comportamiento que no fuera el esperado.

4.3. Herramientas utilizadas

Para el desarrollo de esta etapa se utilizaron varias herramientas que permitieron implementar y probar el funcionamiento del sistema.

- **Python** se utilizó como lenguaje principal para desarrollar toda la simulación y la lógica del proyecto.
- **SciPy** se empleó para resolver las ecuaciones diferenciales del modelo dinámico.
- **NumPy** se utilizó para realizar los cálculos matriciales y las operaciones numéricas necesarias durante la simulación.
- **Matplotlib** se utilizó para generar las gráficas y comparar los resultados obtenidos con cada controlador.
- **PyBullet** se empleó para visualizar el movimiento del robot y comprobar su comportamiento en un entorno físico simulado.

4.4. Verificación de la implementación

Para verificar que la implementación estuviera funcionando de la manera esperada, se revisaron varios aspectos durante las simulaciones. Primero se comprobó que las trayectorias pudieran seguirse de forma continua, sin presentar saltos ni movimientos que no correspondieran con la referencia.

También se revisó que las posiciones, velocidades y las demás variables del sistema mantuvieran un comportamiento coherente a lo largo de toda la simulación. De la misma forma, se verificó que el controlador PID convencional respondiera según el diseño realizado y que el controlador difuso modificara las ganancias únicamente a partir del error y de la derivada del error.

Finalmente, se comparó el movimiento observado en PyBullet con los datos obtenidos durante la simulación, buscando que ambos mostraran el mismo comportamiento del manipulador y que no existieran diferencias importantes entre la parte numérica y la representación visual.

Disponibilidad del código fuente. Para facilitar la revisión y reproducibilidad del proyecto, el código utilizado para el modelado del robot, los controladores y las simulaciones en PyBullet se encuentra disponible en un repositorio público de GitHub: [JBMODSYSTEM/FuzzyController].

4.5. Resultado de esta etapa

Al finalizar esta etapa se obtuvo una plataforma de simulación completamente funcional. Con ella fue posible ejecutar el PID convencional y el PID adaptativo bajo las mismas condiciones, registrar las señales más importantes del sistema y realizar una comparación entre ambos controladores de una forma mucho más clara.

Figura 15. Visualización de la ejecución del robot manipulador planar 2R en PyBullet

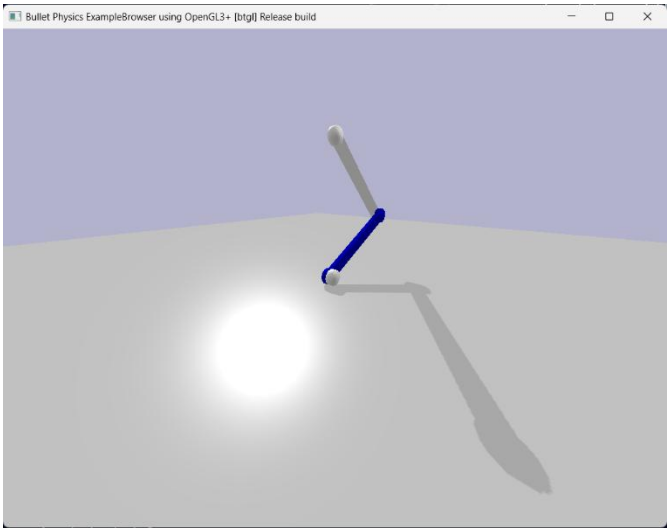
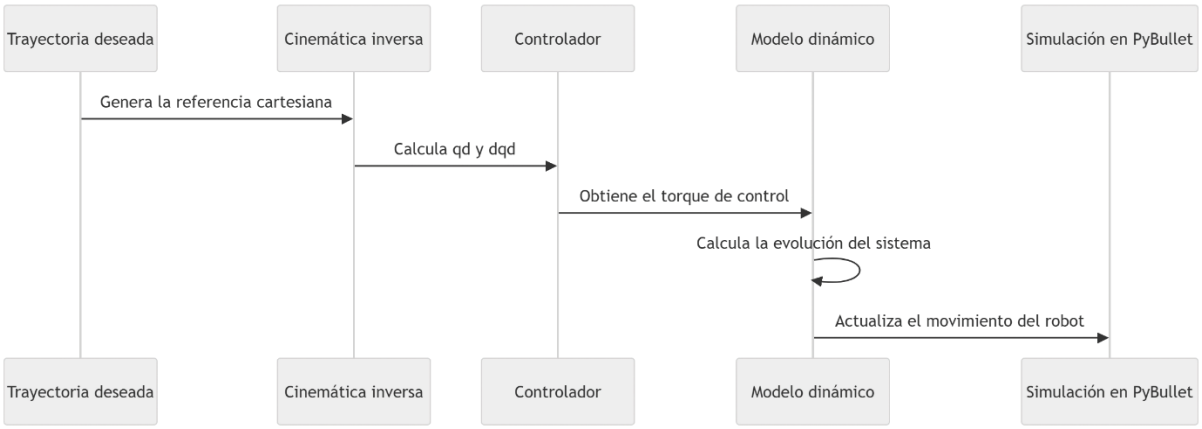


Figura 16. Flujo de implementación y comunicación entre la trayectoria, el controlador, el modelo dinámico y PyBullet



5. Objetivo 4: Comparación y validación de resultados

5.1. Qué se hizo

En esta última etapa se comparó el desempeño del controlador PID convencional con el controlador PID adaptativo basado en lógica difusa. Para que la comparación fuera lo más justa posible, ambos controladores trabajaron con la misma trayectoria de referencia y bajo las mismas condiciones de simulación.

5.2. Criterios de validación considerados

Para analizar el comportamiento de cada controlador se tomaron en cuenta varios indicadores que permitieron evaluar tanto la precisión como la respuesta del sistema.

- **RMSE**, para medir el error de seguimiento tanto en el espacio articular como en el cartesiano.
- **IAE**, con el fin de conocer el error acumulado durante toda la simulación.
- **Sobreimpulso**, para verificar si la respuesta del sistema sobrepasaba la referencia establecida.
- **Tiempo de establecimiento**, que permitió observar qué tan rápido el robot alcanzaba la trayectoria deseada.
- **Esfuerzo de control**, evaluando los torques aplicados por cada estrategia durante el movimiento.

Además de estas métricas, también se revisó el comportamiento visual del robot en PyBullet y la trayectoria seguida por el efector final para comprobar que los resultados fueran coherentes con los datos obtenidos.

5.3. Cómo se validaron los resultados

La comparación se realizó siguiendo el mismo procedimiento para ambos controladores. Primero se ejecutó la simulación utilizando el PID convencional y posteriormente se repitió utilizando el controlador PID adaptativo con lógica difusa y compensación gravitacional.
En ambos casos se mantuvieron iguales la trayectoria de referencia, el tiempo de simulación, el paso de integración y las condiciones iniciales del robot. Después se calcularon las mismas métricas de desempeño y se compararon las gráficas obtenidas para analizar cuál de las dos estrategias presentó un mejor comportamiento durante el seguimiento de la trayectoria.

5.4. Resultado obtenido de esta etapa

Como resultado de esta etapa se obtuvieron diferentes gráficas y valores numéricos que permitieron comparar de manera objetiva el comportamiento de ambos controladores. Con esta información fue posible analizar las ventajas y limitaciones de cada estrategia y verificar si el controlador PID adaptativo basado en lógica difusa ofrecía un mejor seguimiento de trayectoria frente al PID convencional.

La evidencia del cuarto objetivo se encuentra en las Tablas 1, 2 y 3 y en las Figuras 17, 18 y 19. La Tabla 1 muestra los parámetros físicos utilizados en el modelo del robot, los cuales se mantuvieron iguales en todas las simulaciones para que la comparación entre los controladores fuera bajo las mismas condiciones. Las Tablas 2 y 3 presentan los resultados obtenidos para las trayectorias cuadrada y circular, incluyendo los errores y el esfuerzo de control.

Por otro lado, las Figuras 17 y 18 permiten observar el seguimiento de las trayectorias y las

diferencias entre el PID convencional y el PID adaptativo difuso. La Figura 19 muestra el movimiento del robot en PyBullet durante las pruebas. En conjunto, estas evidencias permiten relacionar los datos obtenidos con el comportamiento del robot y comparar de forma más clara el desempeño de ambos controladores.

Tabla 1. Parámetros físicos nominales del manipulador 2R utilizados en la simulación

Parámetro	*Símbolo*	*Valor*	*Unidad*
Masa del eslabón 1	m_1	**1.00**	**kg**
Masa del eslabón 2	m_2	**0.80**	**kg**
Longitud del eslabón 1	l_1	**1.00**	**m**
Longitud del eslabón 2	l_2	**0.80**	**m**
Distancia al centro de masa del eslabón 1	l_{c1}	**0.50**	**m**
Distancia al centro de masa del eslabón 2	l_{c2}	**0.40**	**m**
Momento de inercia del eslabón 1	I_1	**0.083**	**kg·m²**
Momento de inercia del eslabón 2	I_2	**0.043**	**kg·m²**
Coeficiente de fricción viscosa de la articulación 1	b_1	**0.05**	**N·m·s/rad**
Coeficiente de fricción viscosa de la articulación 2	b_2	**0.05**	**N·m·s/rad**
Aceleración de la gravedad	g	**9.81**	**m/s²**

Descripción:

La tabla presenta los parámetros físicos utilizados en el modelo dinámico del robot manipulador de dos grados de libertad. Estos valores se definieron en la simulación y se mantuvieron constantes durante todas las pruebas. Con ellos fue posible representar el comportamiento del manipulador y evaluar los controladores bajo las mismas condiciones.

Tablas de resultados.

Tabla 2. Comparación entre PID clásico y PID difuso para un cuadrado

Métrica	*PID clásico*	*PID difuso*
RMSE q1 (rad)	0.004164	0.000399
RMSE q2 (rad)	0.001068	0.000105
RMSE x (m)	0.001694	0.000214
RMSE y (m)	0.004368	0.000407
RMSE cartesiano (m)	0.004684	0.000460
Torque máximo $\|\tau_1\|$ (N·m)	47.518323	52.221623
Torque máximo $\|\tau_2\|$ (N·m)	61.808144	59.753568

Descripción:

Para el seguimiento de la trayectoria cuadrada, el controlador PID difuso obtuvo mejores resultados que el PID convencional. Los errores en las articulaciones y en el espacio cartesiano fueron mucho menores, lo que permitió recorrer la trayectoria con mayor fidelidad, incluso en los cambios de dirección propios del cuadrado. Aunque los torques máximos aumentaron un poco, ese incremento fue moderado y se vio compensado por la mejora en la precisión y en la respuesta general del robot.

Tabla 3. Comparación entre PID clásico y PID difuso para un circulo

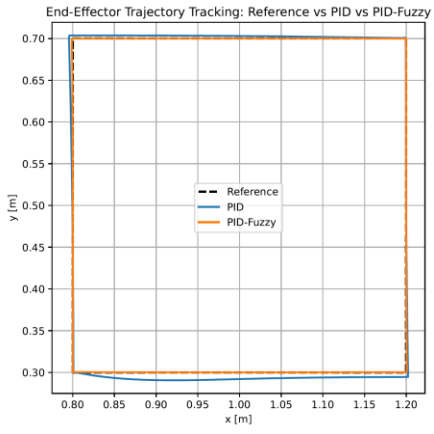
Métrica	*PID clásico*	*PID difuso*
RMSE q1 (rad)	0.010502	0.001373
RMSE q2 (rad)	0.005332	0.000823

RMSE x (m)	0.003398	0.000671
RMSE y (m)	0.014809	0.001714
RMSE cartesiano (m)	0.015194	0.001841
Torque máximo $\|\tau_1\|$ (N·m)	12.502210	14.176228
Torque máximo $\|\tau_2\|$ (N·m)	2.831200	5.496453

Descripción:

Al comparar los resultados obtenidos para la trayectoria circular, se observó que el controlador PID basado en lógica difusa presentó un mejor comportamiento que el PID convencional. Los errores de seguimiento disminuyeron tanto en las articulaciones como en el espacio cartesiano, permitiendo seguir el recorrido del círculo con mayor precisión. Aunque el controlador difuso necesitó un poco más de esfuerzo de control, el aumento en los torques fue moderado y se vio compensado por la mejora obtenida durante el seguimiento de la trayectoria.

Figura 17. Comparación de seguimiento de trayectoria en un cuadrado



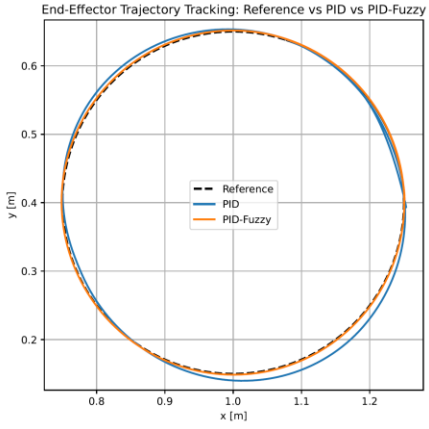
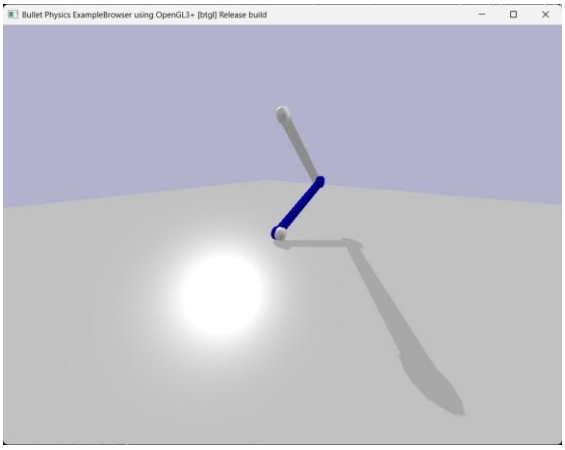


Figura 18. Comparación de seguimiento de trayectoria en un circulo

Figura 19. Pybullet en ejecución



Conclusiones.

Con el desarrollo de este proyecto se obtuvo el modelo dinámico de un robot manipulador planar de dos grados de libertad mediante el formalismo de Euler-Lagrange. Este modelo permitió representar el comportamiento dinámico del sistema y sirvió como base para diseñar e implementar las estrategias de control evaluadas durante las simulaciones.

También se desarrolló un controlador PID adaptativo basado en lógica difusa tipo Mamdani y se comparó con un controlador PID convencional. Los resultados mostraron que el controlador adaptativo logró un seguimiento de trayectorias más preciso, reduciendo los valores de RMSE e IAE tanto en el espacio articular como en el cartesiano. Además, presentó un movimiento más suave y un mejor desempeño frente a las no linealidades propias del manipulador. Aunque en algunos casos fue necesario aplicar un poco más de torque, ese aumento fue pequeño si se compara con la mejora obtenida en la precisión del seguimiento.

De igual manera, la incorporación de la compensación gravitacional (Gravity Feedforward) ayudó a disminuir parte del esfuerzo que debía realizar el controlador para compensar el peso de los eslabones, mejorando el comportamiento general del sistema durante las simulaciones realizadas en PyBullet.

Como trabajo futuro, sería interesante implementar esta estrategia de control en un robot físico para comparar los resultados obtenidos en simulación con un entorno real. También podría evaluarse el uso de otras técnicas de control inteligente, incorporar mecanismos de aprendizaje para ajustar automáticamente las ganancias y extender el desarrollo a manipuladores con un mayor número de grados de libertad.

Referencias

Åström, K. J., & Hägglund, T. (2006). Advanced PID control. ISA—The Instrumentation, Systems, and Automation Society.

Craig, J. J. (2018). Introduction to robotics: Mechanics and control (4th ed.). Pearson.

Ma, H., As'arry, A., Li, C., Ismail, M. I. S., Ramli, H. R., Delgoshaei, A., & Zuhri, M. Y. M. (2025). Improved DBO algorithm tunes fuzzy-PD controller for robot manipulator trajectory tracking. Journal of Algorithms & Computational Technology. https://doi.org/10.1177/17483026251331506

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies, 7(1), 1–13. https://doi.org/10.1016/S0020-7373(75)80002-2

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). Robot modeling and control (2nd ed.). John Wiley & Sons.

Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353.