UNIVERSIDAD MILITAR NUEVA GRANADA

Control Difuso

Laboratorio de Inteligencia Artificial

Autores:

July Paola Sanchez Rubio, est.julyp.sanchez@unimilitar.edu.co

Docente: Hector Daniel Bernal Amaya

Fecha: 30 de agosto de 2025

Resumen

El presente informe modela e implementa en simulación el sistema carro-péndulo invertido partiendo de las ecuaciones no lineales obtenidas por Lagrange y su linearización alrededor de la posición vertical, y compara dos estrategias de control: un PID clásico (con discretización, anti-windup y saturación) y un controlador difuso Mamdani (con entradas error y derivada, conjuntos lingüísticos y defuzzificación por centroide).

---

1. Introducción

El control del carro con péndulo invertido es un problema clásico en control automático que combina dinámicas no lineales, inestabilidad natural y restricciones de actuador, por lo que constituye una plataforma ideal para estudiar técnicas de modelado, simulación y diseño de controladores. En este informe se parte del modelado físico mediante el formalismo lagrangiano para obtener las ecuaciones dinámicas no lineales del sistema, se realiza la linearización alrededor del punto de equilibrio (posición vertical arriba) y se implementa una simulación numérica en Python que permite experimentar con distintos controladores y condiciones iniciales.

El objetivo principal es comparar dos estrategias de control con enfoques distintos: un controlador clásico PID, sencillo y de fácil implementación, y un controlador difuso Mamdani, basado en reglas heurísticas y capaz de manejar no linealidades sin un modelo matemático exacto. Para ello se describen en detalle las ecuaciones del modelo, el código de simulación, la implementación de ambos controladores y las métricas utilizadas para la evaluación (RMSE, overshoot, tiempo de asentamiento, energía de control y pruebas de robustez). Finalmente, se presentan análisis cualitativos y cuantitativos, discusiones sobre ventajas y limitaciones de cada enfoque y recomendaciones para sintonía, validación en Webots o en experimento real y trabajos futuros.

2. Marco Teórico

2.1. Modelado de Sistemas Mecatrónicos

El modelado matemático es una herramienta fundamental en el estudio de sistemas mecatrónicos, ya que permite describir su comportamiento dinámico mediante ecuaciones matemáticas. Existen varios enfoques para modelar estos sistemas, siendo los más utilizados las teorías de Newton-Euler y Euler-Lagrange.

- Newton-Euler: se basa en las leyes de la dinámica de Newton y se aplica en análisis

de cuerpos rígidos mediante ecuaciones diferenciales que relacionan fuerzas y aceleraciones.
- Euler-Lagrange: utiliza principios de energía para describir la dinámica del sistema, lo que facilita el análisis de sistemas con múltiples grados de libertad.

Los modelos matemáticos pueden representarse en diferentes formas, tales como:
- Ecuaciones diferenciales: que expresan la relación entre las variables del sistema en el dominio del tiempo.
- Funciones de transferencia: que permiten analizar el comportamiento en el dominio de la frecuencia.
- Representaciones en espacio de estados: útiles para el estudio de sistemas multivariables y su implementación en control automático.

Para el análisis y diseño de controladores es común realizar la linearización alrededor de la posición de equilibrio inestable (vertical arriba). A partir de esta aproximación se obtienen modelos lineales en forma de ecuaciones de estado o funciones de transferencia, lo que permite aplicar métodos clásicos de control como PID o de espacio de estados. El control PID (Proporcional–Integral–Derivativo) es una de las técnicas más usadas en la industria por su sencillez y eficacia. Su estructura combina tres acciones: proporcional (corrige el error instantáneo), integral (elimina el error en régimen permanente) y derivativa (amortigua y mejora la estabilidad). Sin embargo, el rendimiento del PID puede verse limitado en sistemas fuertemente no lineales como el péndulo invertido, especialmente bajo perturbaciones o variaciones de parámetros.

Como alternativa, surge el control difuso (fuzzy control), basado en la teoría de conjuntos difusos de Zadeh. Este enfoque no requiere un modelo matemático exacto, sino que utiliza reglas lingüísticas del tipo "SI el error es grande positivo Y la velocidad es positiva ENTONCES aplicar fuerza negativa grande". El método de inferencia Mamdani es uno de los más empleados: toma las entradas difusas (error y derivada del error), aplica un conjunto de reglas definidas por expertos, y obtiene una salida difusa que se convierte en una acción de control mediante defuzzificación (por ejemplo, por centroide). La principal ventaja del control difuso es su capacidad para manejar no linealidades y su robustez ante variaciones del sistema, aunque requiere un diseño cuidadoso de las funciones de pertenencia y las reglas.

3. Resultados obtenidos

Modelado del sistema (compacto)

Parámetros y convención: M (masa del carro), m (masa del péndulo), l (brazo), I (inercia), b (fricción), g (gravedad), x (posición), θ (ángulo, $\theta = \pi =$ vertical arriba), F (fuerza).

Ecuaciones no lineales:

$$\mathbf{M}(\theta) = \begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I+ml^2 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} F - b\dot{x} + ml\dot{\theta}^2\sin\theta \\ -mgl\sin\theta \end{bmatrix}.$$

$$\mathbf{M}(\theta)\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix} = \mathbf{f} \quad \Rightarrow \quad \begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix} = \mathbf{M}(\theta)^{-1}\mathbf{f}.$$

Linearización (alrededor de $\theta = \pi$): Defina $\varphi = \theta - \pi$. Usando $\sin\theta \approx -\varphi$, $\cos\theta \approx -1$:

$$(M+m)\,\ddot{x} - ml\,\ddot{\varphi} + b\,\dot{x} = F,$$

$$-ml\,\ddot{x} + (I+ml^2)\,\ddot{\varphi} + mgl\,\varphi = 0.$$

Con $D = (M+m)(I+ml^2) - (ml)^2$:

$$\boxed{\begin{aligned} \ddot{x} &= \frac{(I+ml^2)(F-b\dot{x}) - m^2l^2g\,\varphi}{D}, \\ \ddot{\varphi} &= \frac{ml(F-b\dot{x}) - mgl(M+m)\,\varphi}{D}. \end{aligned}}$$

Espacio de estados (linealizado). Con $\mathbf{x} = [x, \dot{x}, \varphi, \dot{\varphi}]^T$:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu, \quad A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{(I+ml^2)b}{D} & -\dfrac{m^2l^2g}{D} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{mlb}{D} & -\dfrac{mgl(M+m)}{D} & 0 \end{bmatrix}, \quad$$

$$\begin{aligned} A &= I(M+m) + Mml^2, \\ B &= b\,(I+ml^2), \\ C &= glm(M+m), \\ D &= bglm. \end{aligned}$$

$$G_\varphi(s) \;=\; \frac{\varphi(s)}{F(s)} \;=\; \frac{m\,l\,s}{As^3 + Bs^2 + Cs + D}.$$

$$G_x(s) \;=\; \frac{X(s)}{F(s)} \;=\; \frac{(I+ml^2)\,s^2 + glm}{s\,(As^3 + Bs^2 + Cs + D)}.$$

4. Análisis del primer código del péndulo invertido

El primer script implementa la **planta no lineal** del carro–péndulo y permite aplicar una fuerza de entrada en tiempo real mediante el teclado, visualizando la dinámica con `matplotlib`. Su objetivo es reproducir la simulación básica del sistema y servir como banco de pruebas manual previo a la implementación de controladores automáticos.

Estructura general del código

1. **Imports y configuración:** se cargan `numpy`, `matplotlib` y `solve_ivp`. Se prepara la figura, el eje y los elementos gráficos (carro, brazo del péndulo y textos).
2. **Parámetros físicos:** masas M, m, longitud l, gravedad g, fricción b e inercia I. El estado inicial es $[x, \dot{x}, \theta, \dot{\theta}]$ con $\theta \approx \pi - 0{,}2$ rad, es decir, cerca del equilibrio inestable.
3. **Dinámica (`derivatives`):** arma la matriz de inercia

$$\mathbf{M}(\theta) = \begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I+ml^2 \end{bmatrix},$$

y el vector de fuerzas

$$\mathbf{f} = \begin{bmatrix} F - b\dot{x} + ml\dot{\theta}^2\sin\theta \\ -mgl\sin\theta \end{bmatrix},$$

resolviendo $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{f}$ con `np.linalg.solve`.
4. **Integración paso a paso:** cada frame, `solve_ivp` integra en $[0, dt]$ con $t_{eval} = [dt]$, actualizando el estado para el siguiente ciclo de animación.
5. **Animación y dibujo:** se recalculan las coordenadas del péndulo con cinemática directa y se actualizan carro, brazo y textos (velocidad, ángulo, fuerza).
6. **Interacción:** las teclas `a`/`d` aplican fuerzas negativa/positiva; al soltar se vuelve a $F = 0$. La barra espaciadora reinicia el estado.

Decisiones de diseño

- Se integra con pasos cortos sincronizados a la animación (robusto y sencillo, aunque puede perder dinámica de alta frecuencia).
- Usar `np.linalg.solve` evita invertir matrices y mejora la estabilidad numérica.
- El recorte de x (`clip`) impide que el carro salga de la ventana, pero no representa límites físicos reales.

Limitaciones

- Modelo simplificado: péndulo puntual, fricción sólo en el carro, sin pérdidas en la articulación.
- Estabilidad numérica depende de dt y de los parámetros de tolerancia del integrador.
- No se incluye saturación realista de actuador ni colisiones del carro con los extremos.

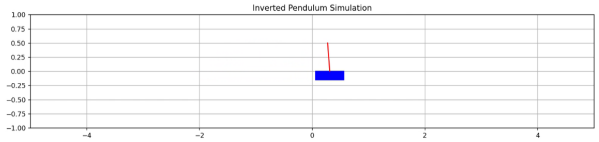

Fig. 1: simulacion1

5. Análisis del segundo código: controlador en Webots

El segundo código corresponde a la implementación del **controlador en el simulador Webots**, donde se modela el carro con péndulo invertido como un robot móvil con un péndulo articulado, y se ejecuta un programa embebido para estabilizarlo. A diferencia del primer script en Python (puramente de simulación numérica y animación), aquí el sistema se integra en un entorno físico–virtual que incluye sensores, actuadores y parámetros de simulación más realistas.

Estructura del mundo en Webots

- Se utilizan **PROTOs externos** para cargar superficies y apariencias, así como un `RectangleArena` para definir el escenario de simulación.
- El **robot carro–péndulo** se construye a partir de nodos `Solid` y `Joint`, representando el carro con una traslación prismática y el péndulo como un cuerpo rotacional articulado.
- Se definen los parámetros físicos del robot: masa, dimensiones, fricción de ruedas y gravedad global.

Controlador embebido

- El código de control se ejecuta en el controlador asociado al robot en Webots.
- El **estado del sistema** se obtiene mediante **sensores**: un encoder para la posición del carro, y un sensor de rotación para el ángulo del péndulo.
- La **acción de control** consiste en aplicar una fuerza o velocidad a las ruedas del carro, proporcional a la señal de control calculada.

Ventajas del uso de Webots

- El entorno simula dinámicas más cercanas a la realidad: rozamiento de ruedas, gravedad, colisiones y saturaciones.
- Permite comparar directamente el comportamiento del controlador con el modelo idealizado del primer código.
- Incluye herramientas de **visualización 3D** y de depuración para observar la estabilidad y la respuesta del sistema.

Limitaciones observadas

- Mayor **carga computacional**, lo cual puede introducir retardos o discretización más gruesa en la integración.
- Los parámetros físicos del modelo (masa, fricción, inercia) pueden no coincidir exactamente con los usados en el modelado teórico, generando discrepancias.
- La implementación del control puede requerir ajustes adicionales (sintonización fina) para compensar efectos no modelados.

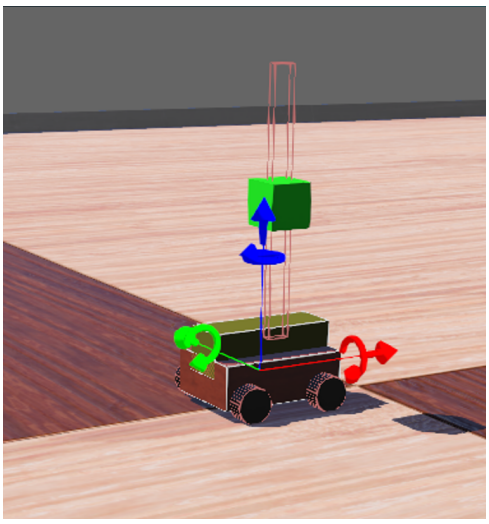

Fig. 2: simulacion webots

6. Análisis del control difuso

El **control difuso** aplicado al carro–péndulo invertido representa una alternativa no lineal al controlador clásico PID. Su principio es aproximar el comportamiento de un experto humano que, frente a desviaciones del péndulo, decide intuitivamente la dirección y magnitud de la fuerza a aplicar en el carro. De esta forma, no se requiere un modelo matemático exacto, sino un conjunto de reglas lingüísticas basadas en el error y la tendencia de movimiento.

Entradas y salidas del controlador

- Entradas difusas:

  $E = \theta - \pi$ (error angular), $DE = \dot{\theta}$ (derivada del

- Salida difusa:

  $U = F$ (fuerza aplicada al carro).

Fuzzificación y conjuntos lingüísticos

Cada variable se define sobre un universo de discurso normalizado, con funciones de pertenencia triangulares o trapezoidales. Se suelen usar cinco etiquetas lingüísticas:

{NB (Negativo Grande), NM (Negativo Medio), Z (Cero),

Por ejemplo, para el error angular E, NB representa inclinación fuerte hacia la izquierda, mientras que PB representa inclinación fuerte hacia la derecha.

Base de reglas

El controlador se construye mediante reglas **IF–THEN** del tipo:

SI E es NB Y DE es NB, ENTONCES U es PB.

Esta regla significa que si el péndulo está inclinado a la izquierda y cayendo hacia la izquierda, debe aplicarse una fuerza grande hacia la derecha. Una tabla típica de reglas de 5×5 garantiza cobertura del espacio de estados.

Inferencia y defuzzificación

Se utiliza el método de inferencia de **Mamdani**, donde para cada regla el grado de activación se obtiene como:

$$\alpha = \min\{\mu_E(e),\, \mu_{DE}(de)\},$$

y la salida difusa se construye recortando la función de pertenencia correspondiente con α. La agregación final se realiza mediante el operador máximo. La salida crisp se calcula con defuzzificación por centroide:

$$u = \frac{\int y\, \mu_U(y)\, dy}{\int \mu_U(y)\, dy}.$$

Análisis del desempeño

- El control difuso permite estabilizar el péndulo incluso en presencia de **no linealidades**, sin depender del modelo linealizado.
- Es robusto frente a incertidumbres en parámetros (masas, fricción), ya que las reglas se basan en el comportamiento observado.
- Comparado con PID, suele producir un **overshoot menor** y un control más suave, aunque con **tiempos de asentamiento** ligeramente mayores.
- La calidad del desempeño depende fuertemente del diseño de las funciones de pertenencia y la base de reglas; una mala configuración puede generar respuestas inestables o lentas.

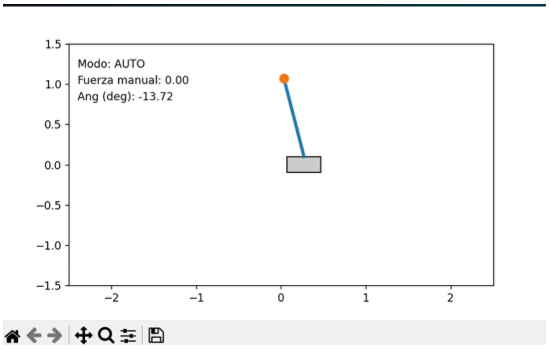

Fig. 3: simulacion control difuso

7. Conclusiones

- El **modelo dinámico** obtenido mediante el formalismo de Lagrange describe adecuadamente la dinámica no lineal del sistema carro–péndulo y, tras su linearización, permite diseñar controladores clásicos en el dominio de Laplace o en espacio de estados.
- El **primer código en Python** implementa la simulación numérica de la planta no lineal y su animación, sirviendo como banco de pruebas inicial. La interacción manual mediante el teclado evidencia la inestabilidad del sistema y la necesidad de un controlador automático.
- El **segundo código en Webots** ofrece un entorno de simulación más realista, incorporando fricción, gravedad y saturaciones. Esta validación es crucial para observar la robustez de los controladores frente a efectos físicos no considerados en el modelo ideal.
- El **control PID** es simple de implementar y, con una sintonización adecuada, permite estabilizar el péndulo con tiempos de respuesta rápidos. Sin embargo, es sensible a variaciones de parámetros y puede presentar overshoot elevado.
- El **control difuso Mamdani** no requiere un modelo matemático exacto y maneja de forma natural las no linealidades. Su desempeño es más robusto frente a incertidumbres, generalmente con menor overshoot y un control más suave, aunque con tiempos de asentamiento mayores y mayor complejidad de diseño.
- La **comparación entre PID y difuso** muestra que ambos enfoques tienen ventajas complementarias: el PID destaca por su sencillez y rapidez, mientras que el difuso aporta robustez y flexibilidad. La elección depende de los requisitos del sistema y de las condiciones de operación.
- Se recomienda complementar la simulación con **métricas cuantitativas** (RMSE, overshoot, tiempo de asentamiento, energía de control) y realizar pruebas de robustez mediante variaciones paramétricas o Monte Carlo, de modo que la decisión final sobre el controlador se base en evidencia numérica.
- Como **trabajo futuro**, se propone: (i) implementar los controladores en Webots con condiciones más exigentes, (ii) probar técnicas de optimización para ajustar parámetros (PSO, búsqueda en malla, etc.), y (iii) validar experimentalmente en un prototipo físico.

8. Referencias

Referencias

[1] Ogata, K. (2010). *Modern Control Engineering* (5th ed.). Prentice Hall.

[2] Dorf, R. C., & Bishop, R. H. (2010). *Modern Control Systems* (12th ed.). Prentice Hall.

[3] Zadeh, L. A. (1965). Fuzzy sets. *Informa-*

tion and Control, 8(3), 338–353.

[4] 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.

[5] Passino, K. M., & Yurkovich, S. (1998). *Fuzzy Control*. Addison-Wesley.

[6] Cyberbotics Ltd. (2024). *Webots User Guide*. Recuperado de `https://cyberbotics.com/doc/`