SISTEMAS NO LINEALES | CURSO 2023-2024

Trabajo práctico

MODELADO, CONTROL Y SIMULACIÓN DE UN SISTEMA ALTAMENTE NO LINEAL: RECREATIVA MONZA

Grupo:

Jiajun Xu
Vladyslav Korenyak
Daniel Sotelo Aguirre

INDUSTRIALES
ETSII | UPM

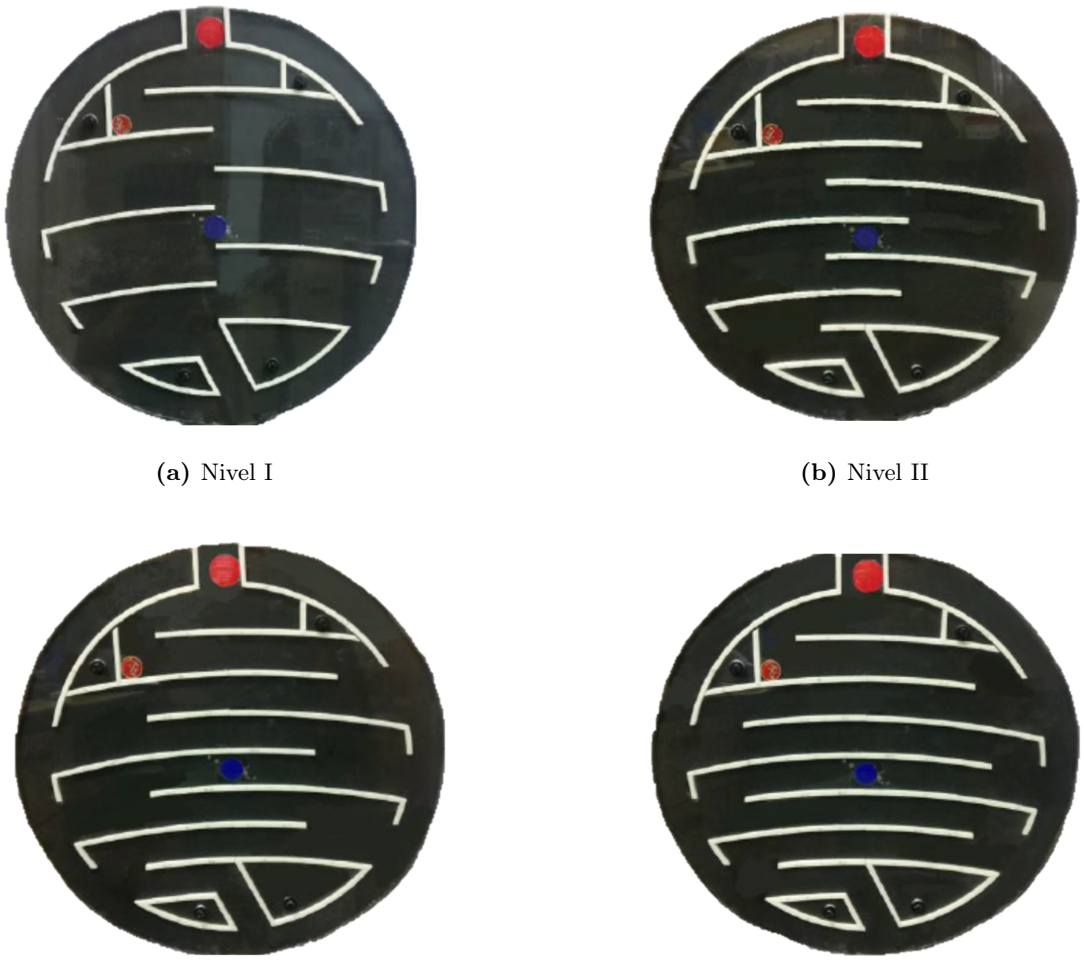
22 de diciembre de 2023

Índice general

1. Introducción

El presente trabajo consiste en conseguir completar el juego de la recreativa Monza, un sistema altamente no lineal, a través del control del actuador que rota el circuito un determinado ángulo θ. Para ello, en primer lugar se realizará un planteamiento de las ecuaciones del movimiento del sistema, se identificarán los parámetros del modelo, se propondrán varias alternativas de control y a continuación se implementarán y validarán los controladores diseñados. Se decidió mejorar el modelo proporcionado, programando desde cero un modelo completo del Monza. El modelo descargable desde `https://github.com/danisotelo/monza_simulator` tiene una interfaz que permite jugar tanto manualmente como probar cada uno de los controladores desarrollados para los distintos niveles de dificultad.

Para completar el juego, se debe rotar el circuito de tal forma que la moneda llegue al final sin caer fuera del área de juego, permitiendo recuperarla. El laberinto se ha modelado como ocho parábolas con la misma ecuación pero desfasadas verticalmente. En la asignatura se ha proporcionado un simulador en Simulink como material de partida, aunque se ha desarrollado otro simulador más completo en Python. El juego cuenta con cuatro niveles de dificultad, según la disposición de los rieles, tal y como se muestra en la Figura 1.1, ordenados de menor a mayor dificultad.



(a) Nivel I **(b)** Nivel II

(c) Nivel III **(d)** Nivel IV

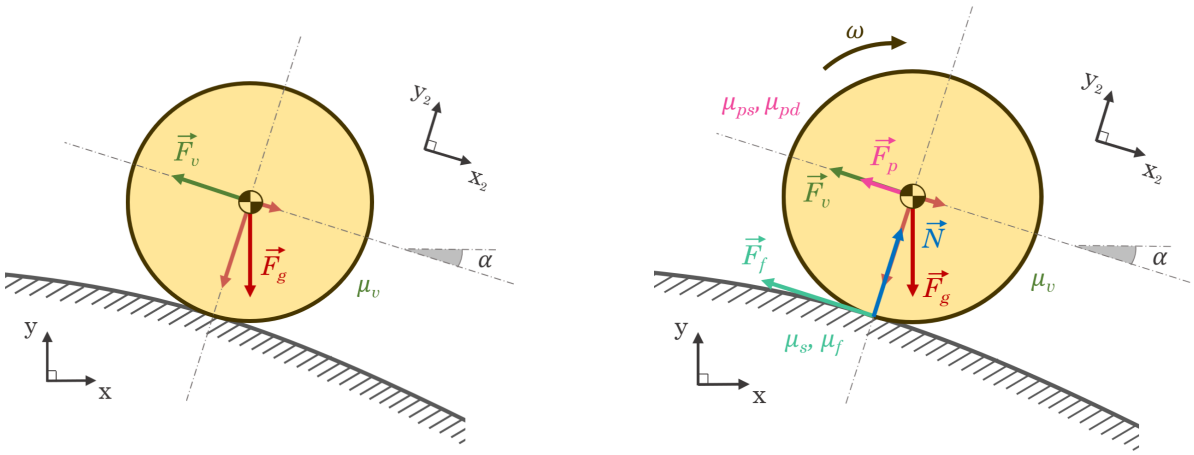
Figura 1.1: Cuatro niveles de dificultad de la recreativa Monza en función de la disposición de los rieles.

2. Ecuaciones del movimiento

Para poder implementar los controladores primero es necesario entender el sistema para poder modelarlo. Se parte del modelo de Simulink proporcionado en la asignatura para a continuación introducir mejoras en el modelo que permitan una mejor aproximación al comportamiento real de la moneda. Se estudia la física de las distintas etapas del juego: el movimiento de la moneda sobre el carril, la caída de la moneda a un carril inferior y la recepción en el carril tras la caída.

2.1. Movimiento por el carril

Cuando la moneda se desplaza por el carril parabólico, actúan varias fuerzas. Tras estudiar el modelo de Simulink proporcionado se observa que únicamente se ha modelado la fuerza gravitatoria (que proyectada en la dirección tangente a la trayectoria produce una aceleración en la moneda) y la fuerza de fricción viscosa, cuyo valor es proporcional a la velocidad de desplazamiento tangencial (y que simula la fricción con el aire y la pared del tablero sobre la que se apoya la moneda). El diagrama de fuerzas puede observarse en la Figura 2.1a. Dado que no se ha modelado ningún rozamiento con el carril, el movimiento de la moneda sobre el raíl sería de deslizamiento puro, es decir, no se produciría rodadura.



(a) Modelo inicial simplificado de Simulink

(b) Modelo detallado

Figura 2.1: Modelos dinámicos del movimiento de la moneda sobre el carril.

El ángulo α que se encuentra rotado el sistema de referencia de la moneda, $\{x_2, y_2\}$, con respecto al sistema de referencia estático $\{x, y\}$, viene dado por la suma del ángulo de rotación del circuito, θ, la variable que se desea controlar; y el ángulo debido a la inclinación del carril parabólico en el punto de contacto x (donde x está medido en el sistema de referencia que gira junto con el tablero $\{x_1, y_1\}$), $\phi(x)$. Dado que las ecuaciones de las parábolas de cada uno de los pisos solo difieren en el término independiente, el ángulo de inclinación debido al carril puede determinarse para cualquiera de ellas a partir de su ecuación geométrica.

$$y_1 = -0{,}54 \cdot x_1^2 + 0{,}16 \Rightarrow \frac{dy_1}{dx_1} = \text{tg}(\phi(x_1)) = -1{,}08 \cdot x_1 \Rightarrow \phi(x_1) = \arctan(1{,}08 \cdot x_1)$$

$$\alpha(\theta, x_1) = \theta + \phi(x_1) = \theta + \arctan(1{,}08 \cdot x_1)$$

Aplicando la Segunda Ley de Newton en el eje x del sistema de referencia $\{x_2, y_2\}$, se obtiene:

$$M \cdot g \cdot \text{sen}(\alpha(\theta, x_1)) - F_v = M \cdot g \cdot \text{sen}(\alpha(\theta, x_1)) - \dot{x} \cdot \mu_v = M \cdot \ddot{x}$$

donde M es la masa de la moneda, g el valor de la gravedad y μ_v el factor de fricción viscoso.

Planteando las ecuaciones en el sistema de referencia $\{x_1, y_1\}$ del tablero con el fin de poder calcular α fácilmente se obtienen las ecuaciones diferenciales 2.1.

$$\begin{cases} \ddot{x} = \cos(\phi(x)) \cdot \left[g \cdot \text{sen}(\alpha(\theta, x)) - \dot{x} \cdot \sqrt{1 + (1{,}08 \cdot x)^2} \cdot \frac{\mu_v}{M} \right] \\ \dot{y} = -1{,}08 \cdot x \cdot \dot{x} \end{cases} \tag{2.1}$$

En la Figura 2.1b se muestra un modelo más detallado que el proporcionado con el fin de conseguir un mayor parecido con la maqueta real. Las principales adiciones al modelo han sido el rozamiento con el carril, el rozamiento con la pared del tablero y el modelado de cierta resistencia a la rodadura.

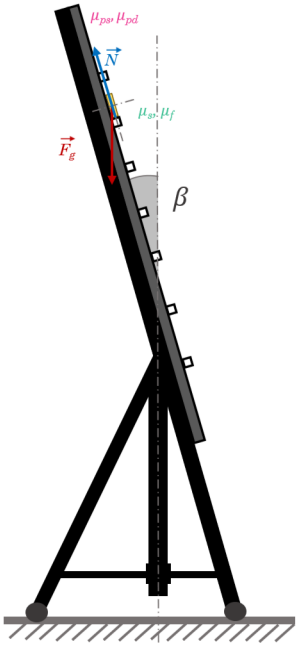


Figura 2.2: Vista de perfil de la maqueta.

2.1.1. Rozamiento con el carril

Para simular la fricción con el carril, se debe tener en cuenta la inclinación del tablero en la maqueta con un ángulo β (ver Figura 2.2). De esta forma, es posible calcular la fuerza de fricción F_f a partir de la fuerza normal (cuyo valor mínimo será nulo cuando la moneda pierda contacto con el carril a partir de cierta velocidad aunque se considera en el modelo que esta velocidad no se supera debido a la limitación física de aceleración angular del volante actuador), la cual se calcula a partir de la aceleración centrípeta sabiendo que el radio de curvatura en la parábola, ρ, viene dado por la siguiente expresión:

$$\rho(x) = \frac{\left(1 + \left(\frac{dy}{dx}\right)^2\right)^{3/2}}{\frac{d^2y}{dx^2}} = -\frac{\left(1 + 1{,}08^2 \cdot x^2\right)^{3/2}}{1{,}08}$$

$$N = M\left[g \cdot \cos(\alpha) \cdot \cos(\beta) - \frac{1{,}08 \cdot \dot{x}^2}{\left(1 + 1{,}08^2 \cdot x^2\right)^{3/2}}\right]$$

$$F_f = \mu_f \cdot N$$

2.1.2. Rozamiento con la pared del tablero

De manera similar a la expresión interior, puede deducirse la expresión para el rozamiento entre la moneda y la pared a partir de la fuerza normal entre las superficies de contacto:

$$F_p = \mu_p \cdot M \cdot g \cdot \text{sen}(\beta)$$

2.1.3. Resistencia a la rodadura y deslizamiento

Además, se ha deseado simular que la moneda no se mueva cuando la inclinación es muy baja, debido a que a la mínima inclinación en el modelo inicial simplificado la fuerza gravitacional genera el par necesario en torno al punto de contacto como para desplazar la moneda y esto no ocurre en la realidad. Para ello, la fuerza de fricción con la pared genera un par que contrarresta al generado por la fuerza gravitatoria teniendo en cuenta un factor de fricción estático μ_{ps} antes de que comience a moverse la moneda y uno dinámico μ_{pd} cuando esta comienza a rodar. Es decir, la moneda no se moverá si $M \cdot g \cdot \text{sen}(\alpha) \cdot \cos(\beta) \leq \mu_{ps} \cdot M \cdot g \cdot \text{sen}(\beta)$, o lo que es lo mismo: $\text{sen}(\alpha) \leq \mu_{ps} \cdot \text{tg}(\beta)$.

Además, en el modelo simplificado se considera que la moneda desliza siempre, cuando en realidad puede cumplirse la condición de rodadura en determinados momentos si la fuerza de fricción es suficiente. Es por ello por lo que la superficie del carril se modela al igual que la de la pared con dos coeficientes de

fricción, uno estático μ_s y otro dinámico μ_f. Si la moneda rueda sin deslizamiento se cumple: $\ddot{x} = \alpha \cdot R$, donde α es la aceleración angular de la moneda. Sabiendo que la fuerza de rozamiento con el carril se relaciona con α a través de la ecuación 2.2 (donde I_c es el momento de inercia de la moneda) es posible obtener la condición de rodadura, así como la expresión para $\ddot{x}$ a partir de la aplicación de la Segunda Ley de Newton de manera análoga a como se realizó anteriormente.

$$F_f \cdot R = I_c \cdot \alpha = \frac{1}{2} \cdot M \cdot R^2 \cdot \alpha \tag{2.2}$$

La moneda deslizará si $\mu_s \cdot N \leq F_{f,rodadura}$. De esta forma se obtiene que el criterio de rodadura es:

$$\mu_s \geq \frac{M \cdot g \cdot \text{sen}(\alpha) \cdot \cos(\beta) - \dot{x} \cdot \mu_v - \mu_{pd} \cdot M \cdot g \cdot \text{sen}(\beta)}{3 \cdot M \cdot \left[g \cdot \cos(\alpha) \cdot \cos(\beta) - \frac{1{,}08 \cdot \dot{x}^2}{(1+1{,}08^2 \cdot x^2)^{1/2}} \right]}$$

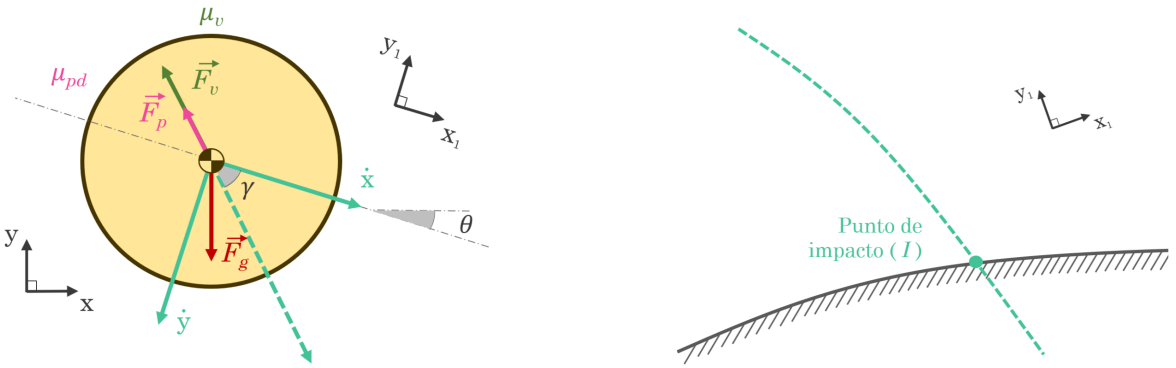
Si se cumple, las ecuaciones del sistema diferencial en condición de rodadura en el sistema de referencia $\{x_1, y_1\}$ vienen dadas por las expresiones 2.3, mientras que las ecuaciones para movimiento con deslizamiento de la moneda vienen dadas por las expresiones 2.4.

$$\begin{cases} \ddot{x} = \frac{2}{3} \cdot \cos(\phi(x)) \cdot \left[g \cdot \text{sen}(\alpha(\theta, x)) \cdot \cos(\beta) - \dot{x} \cdot \sqrt{1 + (1{,}08 \cdot x)^2} \cdot \frac{\mu_v}{M} - \mu_{pd} \cdot g \cdot \text{sen}(\beta) \right] \\ \dot{y} = -1{,}08 \cdot x \cdot \dot{x} \end{cases} \tag{2.3}$$

$$\begin{cases} N = M \cdot \left[g \cdot \cos(\alpha) \cdot \cos(\beta) - \frac{1{,}08 \cdot \dot{x}^2}{(1+1{,}08^2 \cdot x^2)^{1/2}} \right] \\ \ddot{x} = \cos(\phi(x)) \cdot \left[g\,\text{sen}(\alpha(\theta, x)) \cdot \cos(\beta) - \dot{x} \cdot \sqrt{1 + (1{,}08 \cdot x)^2} \cdot \frac{\mu_v}{M} - \mu_{pd} \cdot g \cdot \text{sen}(\beta) - N \cdot \frac{\mu_f}{M} \right] \\ \dot{y} = -1{,}08 \cdot x \cdot \dot{x} \end{cases} \tag{2.4}$$

2.2. Caída al carril inferior

La caída al carril inferior consiste en un movimiento acelerado en los dos ejes del sistema de referencia $\{x_1, y_1\}$ debido a la gravedad y teniendo además en cuenta las fuerzas de rozamiento con la pared y la fuerza de rozamiento viscosa (ver Figura 2.3a, considerando la velocidad final de la moneda al perder contacto con el carril en el punto C.



(a) Modelo de la caída

(b) Cálculo del punto de impacto

Figura 2.3: Modelo de caída al carril inferior y cálculo del punto de impacto.

Nótese que el ángulo γ irá cambiando conforme evolucionen las velocidades en ambos ejes. En el punto de contacto C se determina el valor inicial de este ángulo en la caída como $\gamma = \arctan(\dot{y}/\dot{x})$. Las ecuaciones para la caída libre en el sistema de referencia $\{x_1, y_1\}$ vienen dadas por las expresiones 2.5.

$$\begin{cases} \ddot{x} = g \cdot \cos(\beta) \cdot \text{sen}(\theta) - \dot{x} \cdot \frac{\mu_v}{M} - \mu_{pd} \cdot g \cdot \text{sen}(\beta) \cdot \cos(\gamma) \\ \ddot{y} = \dot{y} \cdot \frac{\mu_v}{M} + \mu_{pd} \cdot g \cdot \text{sen}(\beta) \cdot \text{sen}(\gamma) - g \cdot \cos(\beta) \cdot \cos(\theta) \end{cases} \tag{2.5}$$

2.3. Recepción en carril después de caída

El impacto de la moneda con el carril se modela como un choque inelástico donde la energía perdida en cada impacto (se consideran varios botes) es proporcional a la masa de la moneda, a la velocidad con la que impacta en la superficie del carril, y depende además de un factor k que está relacionado con el rozamiento y la distancia de penetración en la colisión: $Q = \frac{1}{2} \cdot M \cdot (\dot{x}^2 + \dot{y}^2) \cdot (1 - e^{-2k})$.

Realizando la suposición de que la dirección con la que rebota la moneda sobre la superficie de la parábola mantiene el mismo ángulo de impacto con la normal a la superficie que el ángulo que tenía antes del impacto, $\frac{\pi}{2} - \gamma + \phi(x)$, se obtienen las ecuaciones 2.6 para las velocidades tras el impacto a partir de la ecuación de conservación de la energía mecánica (a partir de ellas se pueden volver a aplicar las ecuaciones de caída libre). Se establece un parámetro de velocidad mínima v_{min} para evitar infinitos rebotes. Una vez alcanzada esta velocidad mínima la moneda dejará de rebotar y volverán a aplicar las ecuaciones correspondientes al movimiento sobre el carril.

$$\begin{cases} \dot{x} = \cos(\gamma - 2\phi(x)) \cdot e^{-k} \cdot \sqrt{\dot{x_0}^2 + \dot{y_0}^2} \\ \dot{y} = \text{sen}(\gamma - 2\phi(x)) \cdot e^{-k} \cdot \sqrt{\dot{x_0}^2 + \dot{y_0}^2} \end{cases} \tag{2.6}$$

3. Identificación del modelo

En la sección anterior se han descrito los distintos parámetros que se han tenido en cuenta en el modelo. Para determinar su valor, se han realizado varios experimentos hasta comprobar que su combinación proporciona una simulación realista. No obstante, podría ser necesario un ajuste más fino de acuerdo con el modelo real del Monza. La Tabla 3.1 muestra los valores seleccionados finalmente en `settings.py`.

Parámetro	**Significado**	**Valor**
M (g)	Masa de la moneda	7
R (mm)	Radio de la moneda	15
g (m/s^2)	Gravedad	9.81
μ_v (N·s/m)	Coeficiente de fricción viscosa con el aire	0.01
v_{min} (m/s)	Velocidad mínima para bote	0.4
β (°)	Ángulo de inclinación del tablero	2
k	Coeficiente de choque inelástico	0.8
μ_{ps}	Coeficiente de fricción estática con la pared	0.001
μ_{pd}	Coeficiente de fricción dinámica con la pared	0.0005
μ_s	Coeficiente de fricción estática con el raíl	0.001
μ_f	Coeficiente de fricción dinámica con el raíl	0.0005

Tabla 3.1: Parámetros del modelo.

4. Implementación de los controladores

En esta sección, se presenta el proceso de diseño e implementación de tres controladores distintos, creados específicamente para manejar la dinámica no lineal del juego de la recreativa Monza. Cada enfoque ha sido desarrollado con una estrategia y enfoque particular, ofreciendo diferentes resultados que se comentan más adelante. A través de una revisión detallada, se analizarán los principios detrás de cada uno de ellos, así como las decisiones tomadas durante su creación.

4.1. Alternativa I: Sliding Mode Control (SMC)

El *Sliding Mode Control* es una técnica de control no lineal que se basa en la idea de llevar los estados del sistema a una superficie predefinida en el espacio de estados, conocida como *sliding surface*. El objetivo final es mantener los estados en esta superficie para que se deslicen a lo largo de dicha superficie. En el contexto del control de la recreativa Monza, se busca que el sistema eventualmente converja a un punto de equilibrio siguiendo el *sliding surface*.

Para el diseño del controlador SMC, primeramente se deben definir las variables que definirán el estado actual del sistema. Para ello, se decide asignar las siguientes variables:

- **Error de la bola e:** representa la diferencia del eje x_1 de la bola respecto a la posición deseada. La posición deseada es el final del obstáculo donde tiene que bajar la bola para finalmente, después de recorrer todos los niveles de obstáculos, llegar a la meta.
- **Derivada del error $\dot{e}$:** representa la velocidad a la que el error está variando. Valores positivos indican que el error está aumentando y errores negativos indican que el error disminuye. Su valor absoluto indica la rapidez con la que el error varía.

A partir de la definición de las variables de estado del sistema, el diseño del controlador SMC abarca las dos fases siguientes:

- **Diseño de la superficie deslizante $s(x)$:** Se define $s(t) = e(t) + \lambda \cdot \dot{e}(t)$. El coeficiente λ pondera la importancia relativa de $\dot{e}$ respecto a e. Se debe ajustar λ para influir en la respuesta del sistema y la velocidad a la que converge hacia la trayectoria deseada.
- **Diseño de la ley de control:** La condición teórica del SMC para que el sistema alcance la superficie deslizante es que $s(t) \cdot \dot{s}(t) < 0$. Para ello, existe la técnica de usar una función signo para la ley de control. En este caso se define la acción de control como $u = -k \cdot \text{sign}(s(t))$, donde k es una constante positiva que determina la magnitud de la acción de control, la ganancia de control. Destacar que la acción de control en el contexto de este proyecto es el giro θ del tablero circular.

En la Figura 4.1 se muestra como funciona el SMC gráficamente y como se aproxima hacia el objetivo. En dicha figura, se observan dos comportamientos diferentes:

- ***Reaching Mode*:** Corresponde a la fase en la que el sistema aún no ha llegado a la superficie deslizante. Durante este modo, la dinámica del sistema no es lineal y garantiza que el sistema alcance la superficie deslizante lo más rápido posible.
- ***Sliding Mode*:** En este modo, el sistema se desliza a lo largo de la superficie deslizante hasta llegar al objetivo que corresponde a $e = \dot{e} = 0$, donde la bola ha llegado al final de obstáculo donde debe caer para bajar al siguiente nivel de obstáculo.

Se ha decidido emplear esta estrategia de control no lineal debido a la **simplicidad de su implementación** y porque se considera que su funcionamiento se ajusta bastante bien a las necesidades del problema del Monza y además es **fácilmente configurable** a través de únicamente dos parámetros.

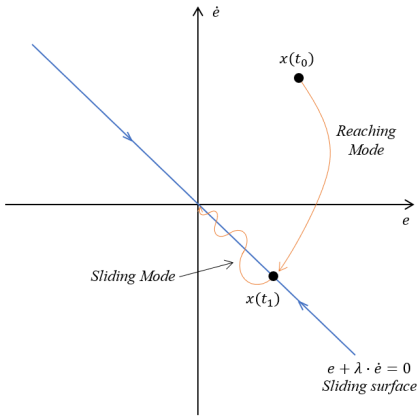


Figura 4.1: Esquema de concepto del funcionamiento del *Sliding Mode Control*.

Conociendo el diseño del control, únicamente falta definir los valores de λ y k. Para ello, se han definido un par de valores distintos para cada nivel, modificando los valores a través de prueba y error hasta conseguir completar los distintos niveles de dificultad. Nótese que estos valores podrían optimizarse más para obtener mejores tiempos para completar la partida.

- **Valores de λ:** 100, 100, 110, 140.
- **Valores de k:** 0.002, 0.004, 0.0095, 0.0095.

Obsérvese que los valores de λ y k correspondientes a la mayor dificultad son de mayor magnitud ya que para λ mayor, el sistema es más sensible a cambios de tasa de cambio error debido a la complejidad del nivel y para k mayor, la ganancia aumenta debido a que se requieren correcciones más rápidas para mantener la bola en la trayectoria deseada. A continuación, se muestra el algoritmo del controlador que se ha empleado para la simulación de la recreativa Monza.

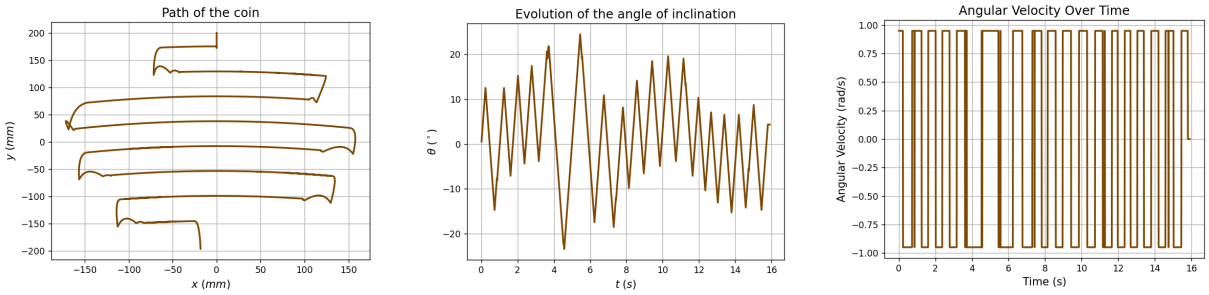
Algorithm 1 Control Action Algorithm

Require: `coin_position`, `target_position`
Ensure: `control_action`
1: `e` $\leftarrow$ `target_position` $-$ `coin_position`
2: `e_prime` $\leftarrow$ `e` $-$ `prev_error`
3: `s` $\leftarrow$ `e` $+ \lambda$`[diff]` $\times$ `e_prime`
4: `control_action` $\leftarrow -k$`[diff]` $\times$ `sign(s)`
5: `prev_error` $\leftarrow$ `e`
6: **return** `control_action` $=0$

Algunos aspectos a destacar en el pseudocódigo presentado:

- La variable `target_position` cambia cuando se pasa de un nivel a otro debido a que la coordenada x_1 objetivo cambia (en un nivel debe bajar por la derecha y en el siguiente nivel inferior debe bajar por la izquierda).
- La variable `coin_position` indica la coordenada local x_1 de la bola.
- La variable λ coge su valor a partir de una lista. La posición de su valor en la lista la define `diff` que corresponde a la dificultad de la recreativa.
- De la misma manera, la variable k también coge su valor a partir de una lista y la dificultad.

A partir de la simulación, para la mayor dificultad se obtienen las trayectorias de la bola, evolución de θ y evolución de $\dot{\theta}$ que se muestran en la Figura 4.2.



(a) Coin path **(b)** θ evolution **(c)** $\dot{\theta}$ evolution

Figura 4.2: Trayectoria y evolución de θ y $\dot{\theta}$ para la máxima dificultad.

4.2. Alternativa II: Fuzzy Logic Control

La lógica difusa es un enfoque donde las variables operan con grados de verdad que pueden ser cualquier número real comprendido entre 0 y 1, lo cual se diferencia de la lógica booleana tradicional, que sólo se define valores de verdadero o falso (1 ó 0). De esta manera, la lógica difusa permite modelar la incertidumbre y la ambigüedad en sistemas complejos, haciéndolo especialmente útil para modelar sistemas reales en los que la información puede ser imprecisa o ambigua. Para entender cómo se emplea este enfoque, se presentan los componentes básicos siguientes:

- **Conjuntos difusos:** En la teoría de conjuntos clásica, un elemento o bien pertenece o no pertenece a un conjunto. En la lógica difusa, se extiende y se añade una función de pertenencia, definida ésta como un número real entre 0 y 1. En las Figuras 4.3a, 4.3b y 4.4a, se observan los conjuntos difusos junto a sus subconjuntos que se han definido para el controlador de esta segunda alternativa. Asimismo, las gráficas representan las diferentes variables que interaccionan para el control:
  - **Distancia:** La variable de entrada distancia se define como la distancia de la moneda a la posición deseada, la cual es el borde del obstáculo del nivel correspondiente por donde debe bajar para pasar al siguiente nivel de obstáculo inferior (`target_position` − `coin_position`). Los subconjuntos para el conjunto distancia son `izquierda`, `derecha` y `centro`, los cuales indican si se encuentran a la derecha de la posición deseada o la izquierda, así como la magnitud de lo que le falta a la moneda para llegar a la posición deseada.
  - **Velocidad:** La variable velocidad, también de entrada, se define como la variación entre la distancia actual y la distancia en el instante anterior (`distance` − `previous_distance`). Los subconjuntos para el conjunto velocidad son `positiva+`, `positiva`, `parada`, `negativa` y `negativa+`. El signo `+` en los subconjuntos laterales denota un gran cambio de la velocidad, lo cual se debe mitigar para el control de la recreativa. La `parada` indica que la variación entre la distancia actual y la distancia previa es mínima o cero. Para entender lo que significa velocidad `positiva` o `negativa` se puede usar el siguiente ejemplo:
    - Dado que el centro del tablero son las coordenadas del origen (x_1, y_1) para el controlador, si la bola se encuentra a la derecha del obstáculo tendrá un valor positivo de posición (`coin_position`). Se define este valor como +20 para el ejemplo. Si se desea llegar al borde derecho del obstáculo, la posición deseada también tendrá valor positivo (`target_position`). Se define este valor como +150 indicando que la bola aún no ha llegado al borde. En este caso, la bola tiene una distancia de +130 que es la resta entre `target_position` menos `coin_position` tal y como se ha definido anteriormente. Si en el siguiente instante, la bola se acerca al obstáculo, tendrá un valor positivo aún mayor respecto a x_1, y_1, +30 por ejemplo. En este caso, la distancia pasa a ser +120 y resulta en que la velocidad es −10. Se observa que si el objetivo en ese obstáculo está a la derecha y

la bola se dirige a ese obstáculo, la velocidad pertenece al subconjunto `negativa`. De igual manera, si se explora para que la posición deseada esté en el lado izquierdo del obstáculo, la velocidad pertenece al subconjunto `positiva` si la bola se acerca a la posición deseada. De esta forma, se deduce lo que siguiente:

1. Si la posición deseada (`target_position`) se encuentra en el lado derecho del obstáculo, velocidad `positiva` indica que la bola se aleja de la posición deseada, dirigiéndose al lado izquierdo.
2. Si la posición deseada (`target_position`) se encuentra en el lado derecho del obstáculo, velocidad `negativa` indica que la bola se acerca a la posición deseada.
3. Si la posición deseada (`target_position`) se encuentra en el lado izquierdo del obstáculo, velocidad `positiva` indica que la bola se acerca a la posición deseada.
4. Si la posición deseada (`target_position`) se encuentra en el lado izquierdo del obstáculo, velocidad `negativa` indica que la bola se aleja de la posición deseada, dirigiéndose al lado derecho.

- **Ángulo:** La variable ángulo, la cual es de salida, define el ángulo al que debe girar el tablero cada instante y en qué sentido. Los subconjuntos para el conjunto ángulo son `izquierda`, indica que el giro sea antihorario, `derecha`, indica que el giro se horario, y `mantener`, indica que se mantenga el ángulo actual.

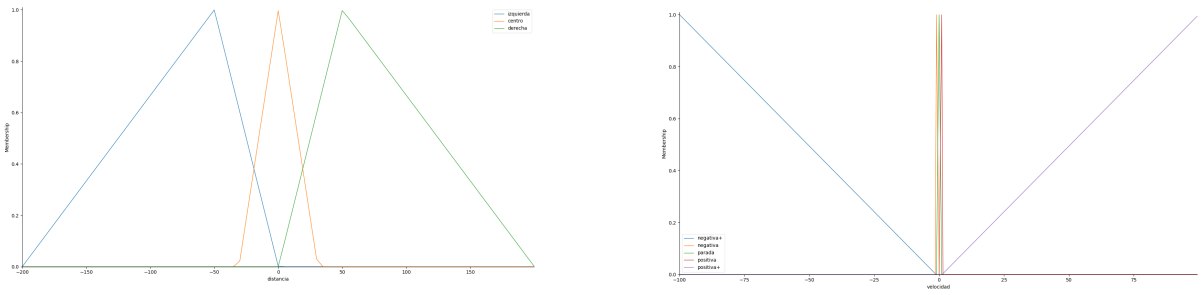
- **Variables lingüísticas:** Los términos definidos en el punto anterior como subconjuntos son las variables lingüísticas que se describen con términos lingüísticos en lugar de valores numéricos. Estos son:
  1. Asociados a la distancia: `izquierda`, `derecha`, `centro`.
  2. Asociados a la velocidad: `positiva+`, `positiva`, `parada`, `negativa`, `negativa+`.
  3. Asociados al ángulo: `izquierda`, `derecha` y `mantener`.
- **Funciones de pertenencia:** Representan el grado de pertenencia de un elemento a un subconjunto definido. A partir de las Figuras 4.3a, 4.3b, 4.4a, se observa que estas funciones de pertencia son de forma triangular y que se ha elegido esta forma debido a los siguientes factores:
  1. **Simplicidad:** Las funciones triangulares son intuitivas, fáciles de entender y visualizar.
  2. **Interpretabilidad:** En el contexto de la recreativa Monza, existe comportamientos que pueden categorizarse de forma relativamente simple como por ejemplo "la bola se encuentra a la derecha". Una función triangular puede capturar adecuadamente estas categorías.
  3. **Eficiencia computacional:** Las funciones triangulares requieren menos cálculos que otras funciones más complejas, lo cual es beneficioso para la velocidad de procesamiento.
- **Reglas difusas:** Son las operaciones entre los subconjuntos que definen la lógica difusa. Estas reglas se formulan como declaraciones condicionales "if-then" a partir de las entradas y retornando las salidas. Las reglas difusas empleadas para el controlador son las que se indican en la Tabla 4.1.
- **Fuzzificación:** Es la operación que convierte la medida de `coin_position` en un valor en cada función de membresía a las cuales pertenece. En la figura 4.4b, se observa un ejemplo de la fuzzificación de la medida en la distancia. Se observa que la medida pertenece a dos subconjuntos con distintos grados de pertenencia lo cual debe emplearse para el resto de las operaciones. Lo mismo se puede hacer con la velocidad.
- **Inferencia difusa:** Es el mecanismo que se encarga de procesar las reglas difusas dada las entradas (los valores de la fuzzificación para distancia y velocidad) para producir un conjunto difuso de salida (ángulo). El valor númerico concreto que se utiliza en el sistema real utiliza la defuzzificación del método del centroide. Este método se basa en encontrar el "centro de gravedad" del área bajo la curva de la función de pertenecia de salida. Los motivos para proceder con este método son:
  1. **Promedio ponderado:** Este método calcula el valor defuzzificado como promedio ponderado de todos los valores posibles, lo cual significa que todos los valores y grados de pertenencia de la salida difusa contribuyen al resultado final.
  2. **Continuidad y suavidad:** Las salidas que se producen varían de manera continua y suavem

Distancia	Velocidad	Ángulo	Descripción
derecha	negativa	derecha	Si el objetivo es ir al borde derecho y la bola se dirige correctamente, seguir girando el tablero en sentido horario.
derecha	negativa+	izquierda	Si el objetivo es ir al borde derecho y la bola se dirige muy rápido, girar el tablero en sentido antihorario para frenar.
derecha	positiva	derecha	Si el objetivo es ir al borde derecho y la bola va en sentido contrario, girar el tablero en sentido horario.
derecha	parada	derecha	Si el objetivo es ir al borde derecho y la bola está parada, girar el tablero en sentido horario.
centro	negativa	izquierda	Si el objetivo es ir al borde derecho y la bola está cerca del borde, girar el tablero en sentido antihorario.
izquierda	positiva	izquierda	Si el objetivo es ir al borde izquierdo y la bola se dirige correctamente, seguir girando el tablero en sentido antihorario.
izquierda	positiva+	derecha	Si el objetivo es ir al borde izquierdo y la bola se dirige muy rápido, girar el tablero en sentido horario para frenar.
izquierda	negativa	izquierda	Si el objetivo es ir al borde izquierdo y la bola va en sentido contrario, girar el tablero en sentido antihorario.
izquierda	parada	izquierda	Si el objetivo es ir al borde izquierdo y la bola está parada, girar el tablero en sentido antihorario.
centro	positiva	derecha	Si el objetivo es ir al borde izquierdo y la bola está cerca del borde, girar el tablero en sentido horario.

Tabla 4.1: Tabla de Reglas Difusas.

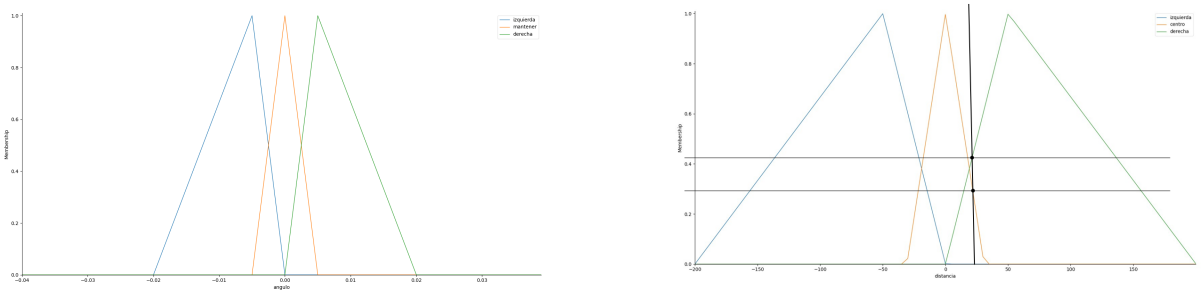
con cambios en las entradas lo cual es adecuado para evitar movimientos bruscos en el contexto de la recreativa Monza.

3. **Intuitivo y ampliamente utilizado:** Este método es intuitivo debido a que proporciona un valor que está "en el centro" de todos los valores recomendados. Por tanto, gracias a su popularidad y eficacia, existe una amplia documentación para poder estudiar y poder implementar la lógica difusa en el controlador.
4. **Adecuado para sistemas de tiempo real:** Dado que la recreativa Monza opera en tiempo real, el método del centroide es adecuado debido a que es computacionalmente eficiente y produce resultados consistentes.



(a) Conjunto difuso para la distancia **(b)** Conjunto difuso para la velocidad

Figura 4.3: Conjuntos difusos para la distancia y la velocidad.



(a) Conjunto difuso para el ángulo **(b)** Fuzzificación en distancia

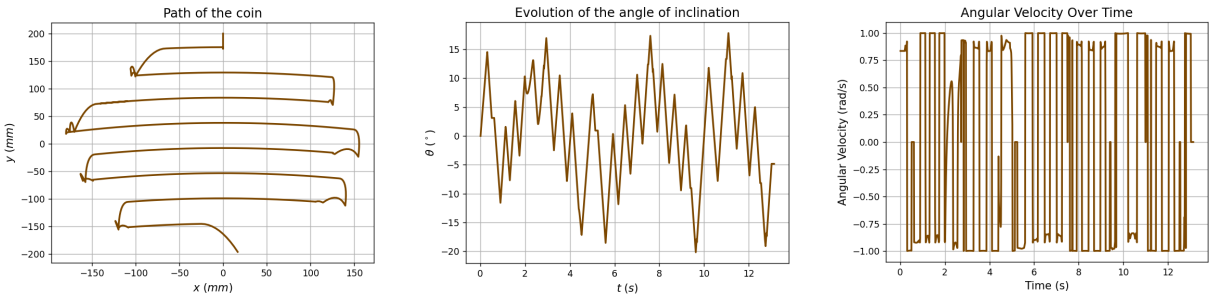
Figura 4.4: Conjunto difuso para el ángulo y fuzzificación en distancia.

Disponiendo todos los componentes definidos para la recreativa Monza, se logra controlar y satisfactoriamente pasar el juego ya que el control retorna en cada momento lo que debe girar el tablero. A continuación, se presenta el algoritmo empleado para el controlador de lógica difusa:

Algorithm 2 Control de lógica difusa

Require: coin_position, target_position, coin_n
Ensure: angle_output
1: $dist \leftarrow target\_position - coin\_position$
2: $vel \leftarrow dist - prev\_dist$
3: Define las variables de entrada y salida
4: Define los conjuntos difusos
5: Define las reglas difusas
6: valores de membresía = Fuzzificación a partir de *dist* y *vel*
7: angle_output = Defuzzificación según centroid a partir de valores de membresía
8: $prev\_dist \leftarrow dist$
9: **return** angle_output =0

Las reglas fuzzy permiten completar el juego para todas las dificultades. Se ha decidido seleccionar esta estrategia de control para el sistema no lineal debido a que **permite el control del sistema de manera muy intuitiva con lenguaje humano** y además para poder **aplicar los conocimientos vistos en las asignaturas de Sistemas No Lineales e Inteligencia Artificial**. A continuación, la Figura 4.5 muestra la trayectoria de la bola, la evolución de θ y la evolución de $\dot{\theta}$ para la mayor dificultad del juego.



(a) Coin path **(b)** θ evolution **(c)** $\dot{\theta}$ evolution

Figura 4.5: Trayectoria y evolución de θ y $\dot{\theta}$ para la máxima dificultad.

4.3. Alternativa III: Controlador entrenado mediante RL

El Aprendizaje por Refuerzo (Reinforcement Learning, RL) es una rama del aprendizaje automático que se enfoca en cómo los agentes deben tomar acciones en un entorno para maximizar alguna noción de recompensa acumulativa. En RL, un agente toma decisiones secuenciales, observando el estado del entorno y ejecutando acciones que alteran dicho estado. La respuesta del entorno a estas acciones se traduce en recompensas, que guían al agente para aprender estrategias óptimas a través de la experiencia.

Aunque ya desde los años 80 se han realizado aplicaciones mediante RL, esta rama se popularizó a partir de 2013, cuando se consiguió que jugara a varios videojuegos de Atari con conocimiento inicial nulo mediante un modelo denominado **Deep Q-Network (DQN)**. El DQN utiliza una red neuronal profunda para aproximar la función de valor Q, que representa el valor esperado de las recompensas futuras obtenidas por una acción en un estado específico. Una explicación más detallada se puede encontrar en el artículo original *"Playing atari with deep reinforcement learning"* de Mnih, V., et al. (2013). En el caso de este trabajo, este es el modelo empleado para desarrollar el controlador. Se ha decidido emplear esta técnica de control no lineal para aprender más sobre RL.

El proceso de entrenamiento del modelo de aprendizaje por refuerzo para el juego Monza involucra varias etapas. Estas etapas incluyen la adaptación del juego para facilitar el entrenamiento mediante RL, la inicialización del modelo y la implementación del sistema de recompensas. Para ello se usó la librería PyTorch. En las pruebas realizadas, al no tener conocimiento previo sobre cómo desarrollar un controlador mediante RL, se han realizado **muchos intentos fallidos**. Se han intentado distintas arquitecturas de la red neuronal, distintas simplificaciones del juego y distintos sistemas de recompensas hasta conseguir dar con un controlador adecuado. En este documento sólo se describe el entrenamiento realizado para conseguir el resultado final y se mencionan algunas de las dificultades enfrentadas, con el ánimo no extender la memoria en exceso.

Adaptación del Juego Monza para RL

Para entrenar el modelo de RL, primero fue necesario adaptar el juego Monza a un formato compatible con los algoritmos de aprendizaje por refuerzo. Esto incluyó la creación de las funciones `reset` y `play_step`, que permiten al modelo interactuar con el juego y recibir retroalimentación. La función `reset` inicializa el juego con una posición de la moneda aleatoria, mientras que `play_step` ejecuta un paso del juego dada una acción del modelo, devolviendo el nuevo estado, la recompensa obtenida y si el juego ha terminado. Estas funciones son esenciales para simular la interacción dinámica entre el agente (modelo de RL) y el entorno (juego Monza).

Además, se ha simplificado el juego reduciendo el número de rieles a solo 2, el inicial y el de caída. Esto se debe a que el juego es simétrico y, si el agente es capaz de superar un solo riel exitosamente, por simetría podrá superarlos todos mientras las condiciones para cada riel sean similares, como es el caso de Monza. Esta característica del juego fue clave para poder desarrollar el controlador, pues numerosos intentos para conseguir que el agente consiguiera resolver múltiples rieles seguidos no han dado buenos resultados.

Inicialización del Modelo DQN

El modelo DQN se inicializó utilizando una **arquitectura** de red neuronal que consta de 3 capas lineales denominadas "*Leaky Rectified Linear Unit (ReLU)*". La función de activación es:

$$f(x) = \begin{cases} x, & \text{if} \quad x \geq 0 \\ \text{scale} \cdot x, & \text{if} \quad x < 0 \end{cases}$$

La **primera capa** contiene **3 neuronas**, que corresponden a la observación del agente. Es decir, de todo el entorno del juego, el agente sólo percibe 3 números. Estos números, definidos en la función `get_state` dentro del archivo `controllers.py`, son el ángulo en el que se encuentra el mapa, la velocidad

de la moneda y la dirección objetivo de la moneda (izquierda o derecha). En el fondo, la red neuronal sólo utiliza los dos primeros números, siendo la dirección objetivo usada fuera de la red neuronal. Esta capa está normalizada en el intervalo [-1, 1], lo cual ayuda al entrenamiento del modelo. La **segunda capa** contiene **20 neuronas**, y es la denominada *capa oculta*. La **última capa** consta de **3 neuronas** que corresponden a las acciones que el agente puede tomar: girar a la izquierda, a la derecha o no girar. El agente escoge la acción de mayor valor Q, es decir, la acción que predice la mayor recompensa. El código asociado se puede ver en la función `get_action`.

Se intentó resolver el juego con numerosos espacios de observación, incluyendo el más completo de ellos la posición, la aceleración, el nivel actual, la distancia hasta el borde, las acciones previas y un historial de todos estos valores, resultando en una capa inicial de cerca de 100 neuronas. Además, se probaron modelos de hasta 6 capas de 1024 neuronas. Incluir toda esta información y neuronas adicionales no dio buenos resultados, probablemente porque el resto de capas sólo introducían ruido en la red y la información adicional era redundante para el problema planteado.

Para conseguir el resultado final no se procedió a entrenar directamente al red neuronal, sino que, de manera similar a como se haría mediante un algoritmo evolutivo, se generaron **1000 instancias del modelo con inicializaciones aleatorias** para explorar diferentes configuraciones iniciales. Después, se seleccionó el modelo con el mejor rendimiento preliminar para entrenarlo mediante el método de descenso de gradiente. Esta solución se adoptó porque el agente inicialmente no obtenía buenos resultados y no aprendía bien. La inicialización de muchos agentes aleatorios permitió superar la primera fase de exploración del agente.

También se debe comentar que la inicialización de los pesos de la red se realizó mediante una distribución uniforme, asignando a cada peso un número aleatorio en el intervalo [-1, 1]. Esta inicialización es distinta a la inicialización por defecto, que se realiza mediante una distribución normal y que facilita el entrenamiento posterior. Este cambio se realizo porque el objetivo de la inicialización no era directamente entrenar la red neuronal sino preseleccionar una red antes de entrenarla, para lo cual una distribución normal introduciría un sesgo que impediría explorar los agentes de forma adecuada.

Por último, un factor decisivo durante el entrenamiento fue el valor del ***learning rate***, que determina la magnitud debe ser la variación de los pesos de la red en cada iteración del entrenamiento. El *learning rate* final usado es de $5 \cdot 10^{-7}$. Se han realizado experientos con valores mayores, pero el aprendizaje se volvía inestable y no convergía en la solución esperada. Esto es una desventaja del modelo DQN, que suele ser sensible a los hiperparámetros del modelo.

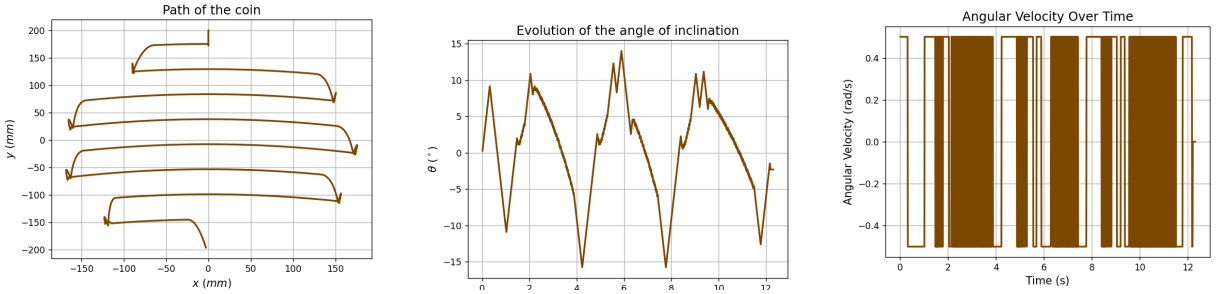
Sistema de Recompensas

El corazón del entrenamiento en aprendizaje por refuerzo es el sistema de recompensas, que guía al modelo hacia comportamientos deseables. En este caso, se diseñó un esquema de recompensas basado en la velocidad de la moneda en el juego. Se otorgan recompensas positivas por acciones que conducen a la moneda hacia un rango de velocidades determinado y se penalizan las acciones que la alejan de este rango. De esta forma, el agente aprende a mantener una velocidad constante sobre la parábola. La implementación de este sistema se puede observar en la función `play_step()` del archivo `game_objectsRL.py` dentro de la carpeta de `training/` del repositorio de GitHub accesible desde https://github.com/danisotelo/monza_simulator.

Se han probado numerosos sistemas de recompensas, siendo el más directo de ellos una recompensa por pasar de un riel al siguiente y una penalización por salirse del juego. Sin embrago, después de muchos intentos el agente no conseguía superar el juego, quedándose estancado en el segundo riel. Esto se debe a que este sistema de recompensas es muy disperso, un problema común en RL donde el algoritmo de entrenamiento no es capaz de "deducir" qué acciones han causado la recompensa y por tanto converger hacia una estrategia que la maximice.

Resultados

Finalmente, se obtuvo el modelo `zecord_model.pth`, capaz de seguir una velocidad constante sobre la parábola. Basado en este control, aplicable a una sola parábola, se construye una lógica sencilla que invierte los datos de entrada al modelo dependiendo de si se encuentra en un nivel par o impar, consiguiendo así que supere todos los niveles del Monza con facilidad. Los resultados obtenidos para la trayectoria de la moneda y la evolución de θ y $\dot{\theta}$ se muestran en la Figura 4.6.



(a) Coin path **(b)** θ evolution **(c)** $\dot{\theta}$ evolution

Figura 4.6: Trayectoria y evolución de θ y $\dot{\theta}$ para la máxima dificultad.

5. Resultados, Conclusiones y Líneas Futuras

A continuación, en la Tabla 5.1 se muestran los tiempos requeridos para completar la partida por cada controlador para cada una de las dificultades.

Controlador	Dificultad 1	Dificultad 2	Dificultad 3	Dificultad 4
Sliding Mode Control	6.59 s	8.79 s	13.32 s	15.91 s
Fuzzy Logic Control	4.36 s	7.65 s	10.22 s	13.13 s
RL Control	5.00 s	7.09 s	12.31 s	12.31 s

Tabla 5.1: Tabla de tiempos para completar la partida.

Tras comparar los resultados y ver las gráficas, puede observarse que las técnicas de control fuzzy y por RL son más agresivas que el SMC, por lo que consiguen superar los niveles en un menor tiempo. Puede observarse que para los controladores SMC y fuzzy la **velocidad angular máxima** es de 1 rad/s, lo que equivale a 10 rpm, mientras que la velocidad angular máxima para el controlador con RL es de 5 rpm. Esta limitación se debe a la resolución que se le ha dado a θ para cada timestep. Estos valores de velocidad angular máxima son realistas, aunque **no lo es la aceleración angular** α. Esto se debe a que el modelo programado presenta la limitación de que la variable de actuación es directamente el ángulo θ y no se han impuesto límites en su segunda derivada. Para implementarlo, sería necesario que la variable de actuación fuera el par sobre el tablero, T. De tal forma que el par estaría relacionado con la aceleración angular del tablero a través de la ecuación 5.1, donde I_{tablero} es el momento de inercia del tablero.

$$T = I_{\text{tablero}} \cdot \alpha = \frac{1}{2} m r^2 \cdot \alpha \tag{5.1}$$

Este par a su vez dependería de la tensión aplicada al motor, por lo que todo depende del grado de detalle al que se desee llegar con el modelado. Realizar este tipo de limitaciones en Simulink es más sencillo debido a que basta con imponer un límite en el integrador. En el caso del código de Python

es algo más complicado y probablemente la mejor opción sería cambiar la variable de actuación por el par. En cualquier caso, el modelo al final sirve para ajustar los controladores y realizar los experimentos que se deseen para calibrarlos correctamente, aunque siempre existe el *model-to-real gap*. Debido a que finalmente no se ha dispuesto de la plataforma real para probar el Monza, no se ha podido comprobar si esta limitación del modelo con aceleraciones angulares infinitas supone que los controladores no son válidos.

Como **líneas futuras** del trabajo, restaría probar los controladores en la **plataforma real y verificar** su correcto funcionamiento. En caso de observar discrepancias significativas con respecto al comportamiento en simulación, el principal causante probablemente fuera la **ausencia de limitación en la aceleración angular del modelo**, la cual habría que modelarla **cambiando la variable de actuación por el par** y por tanto la aceleración angular, por lo que el ángulo θ rotado en cada step de la simulación dependería de la velocidad angular del step anterior a través de una ecuación de movimiento circular uniformemente acelerado. No se ha llegado a implementar este factor en el código debido a motivos de tiempo. En caso de que los controladores siguieran sin ser válidos, podría deberse a la calibración de la cámara al detectar la posición de la moneda o a posibles retrasos que pudiera haber en la transferencia de la posición real de la moneda sobre el tablero, ya que estos retrasos son críticos especialmente en sistemas altamente no lineales e inestables como este.

En conclusión, en este trabajo se ha desarrollado un **simulador completo** del juego Monza en **Python**, modelando con precisión el movimiento de la moneda y las fuerzas que actúan sobre ella y proporcionado una plataforma flexible para probar diferentes estrategias de control. A partir de un **análisis detallado de la dinámica de la moneda**, se han podido incluir en la simulación diferentes fuerzas como la gravedad, la fricción viscosa, la resistencia a la rodadura y el rozamiento con el carril y la pared del tablero permitiendo así un control más preciso y realista de la moneda. La **diversidad en las técnicas de control** implementadas es un aspecto destacado del trabajo. Se han utilizado tres enfoques: Sliding Mode Control, Fuzzy Logic Control y un controlador basado en Aprendizaje por Refuerzo, cada uno ofreciendo perspectivas diferentes en la resolución del problema. El uso del **Aprendizaje por Refuerzo**, y en particular la técnica de Deep Q-Network, ha sido especialmente interesante. A pesar de los desafíos que presenta el aprendizaje por refuerzo, especialmente en términos de ajuste de hiperparámetros y diseño del sistema de recompensas y de la arquitectura de la red neuronal, su aplicación ha demostrado ser eficaz para el control de la moneda.

En resumen, este proyecto ha logrado combinar con éxito la programación en Python, el modelado físico y la implementación de técnicas de control avanzadas para simular y controlar el juego Monza. El código completo está disponible junto con las instrucciones para su instalación en el repositorio de GitHub `https://github.com/danisotelo/monza_simulator`.