Control péndulo invertido

Santiago Corcobado Sánchez - 7003566

Resumen – ***En este laboratorio se desarrolló un simulador de péndulo invertido montado sobre un carro utilizando Python con visualización gráfica. El objetivo fue estabilizar el péndulo en posición vertical mediante control PID y control difuso. Se modelaron las ecuaciones de movimiento considerando la gravedad y la inercia del sistema. El control PID se implementó mediante retroalimentación del ángulo del péndulo, mientras que el control difuso se basó en variables lingüísticas de ángulo y velocidad angular, aplicando reglas Max-Min y defuzzificación por centroide para determinar la fuerza sobre el carro. La simulación permitió interactuar con el sistema mediante un botón de inicio y observar cómo ambos métodos lograban mantener el péndulo estable, mostrando la necesidad de sintonización precisa en el PID y la robustez del control difuso frente a variaciones del sistema.***

Abstract - ***In this laboratory, an inverted pendulum simulator mounted on a cart was developed using Python with graphical visualization. The objective was to stabilize the pendulum in a vertical position using PID control and fuzzy control. The equations of motion were modeled considering the gravity and inertia of the system. PID control was implemented using pendulum angle feedback, while fuzzy control was based on linguistic variables of angle and angular velocity, applying Max-Min rules and centroid defuzzification to determine the force on the cart. The simulation allowed interaction with the system using a start button and observation of how both methods managed to keep the pendulum stable, demonstrating the need for precise tuning in the PID and the robustness of fuzzy control in the face of system variations.***

I. DESARROLLO

Para el desarrollo de la guía de laboratorio se tiene que realizar inicialmente el modelado de péndulo invertido el cual se toma la figura 1, como base para realizar el modelado.

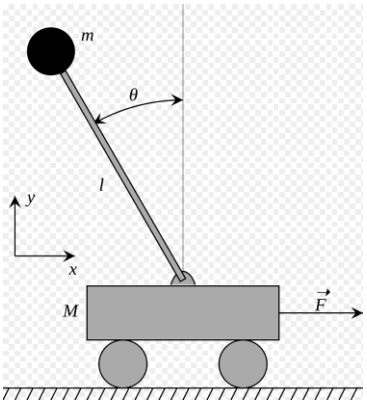

Figura 1, Péndulo invertido

Para realizar el modelado del sistema se usa el método de Euler-Lagrange.

$$T_1 = \frac{1}{2}Mx^2$$

$$T_2 = \frac{1}{2}M(x^2 + y^2)$$

Donde

$$x_2 = x - l * \sin(\theta)$$
$$y = l * \cos(\theta)$$
$$x_2 = \dot{x} - l * cos(\theta)$$
$$\dot{y_2} = -l * \dot{\theta} * \sin(\theta)$$

Por lo tanto, tenemos que el lagrangiano es.

$$L = \frac{1}{2}Mx^2 + \frac{1}{2}m\left[(\dot{x}^2 - 2\dot{x}\dot{\theta} * l * \cos(\theta) + l^2\theta^2\right] - m * g * \cos(\theta)$$

Sacamos los espacios de estados

$$A = \begin{matrix} 0 & 1 & 0 & \\ 0 & 0 & \frac{-mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{ML} & 0 \end{matrix}$$

$$B = \begin{matrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{ML} \end{matrix}$$

Luego en Matlab se saca la función de transferencia y comprobamos la controlabilidad y observabilidad.

```
ft =

  0.1 s^2 + 1.11e-17 s - 0.03532
  ------------------------------
  s^4 - 1.11e-16 s^3 - 0.3237 s^2
```

Figura 2. Función de transferencia

```
El sistema es controlable.
El sistema es observable.
```

Figura 3. Controlabilidad y observabilidad

Simulamos la función de transferencia para ver su respuesta.

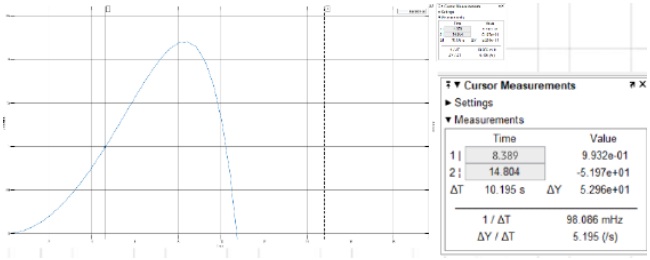

Figura 4. Respuesta función de transferencia.

Luego realizamos un control por retroalimentación de estados donde ponemos un tiempo de establecimiento deseado de $tsd = 6seg$ y un factor de amortiguamiento de $\zeta = 1$.

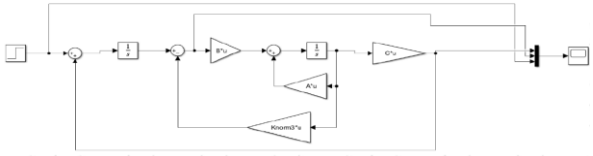

Figura 5. Retroalimentación de estados en simulink

Donde se obtiene la siguiente respuesta.

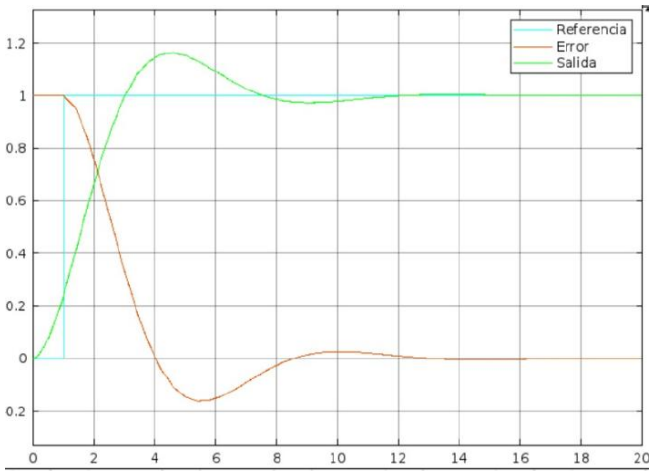

Figura 6. Respuesta PID

Con ello se observa que el control PID funciona correctamente con el modelo hecho y ya se puede pasara a la programación en Python.

Las constantes obtenidas son las siguientes.

$$K_p = 150 \quad K_i = 0 \quad K_d = 25$$

Control por mouse péndulo invertido.

Para el control con el mouse del péndulo invertido se le pide a ChatGPT que nos del código teniendo en cuenta el modelo que realizamos.

```
# ----------------------------
# Modelo del péndulo invertido con fricción
# ----------------------------
def pendulo_invertido_modelo(state, F, params, dt):
    x, dx, theta, dtheta = state
    M = params["M"]
    m = params["m"]
    L = params["L"]
    g = params["g"]
    b = params["b"]
    mu = params["mu"]

    sin_theta = np.sin(theta)
    cos_theta = np.cos(theta)
    denominator = M + m - m * cos_theta**2

    ddx = (F - b * dx + m * sin_theta * (L * dtheta**2 + g * cos_theta - mu * dtheta * cos_theta / (m * L))) / denominator
    ddtheta = (-F * cos_theta + b * dx * cos_theta - (M + m) * g * sin_theta - mu * dtheta * (M + m) / (m * L)) / (L * denominator)
```

Figura 7. Modelado en el código.

Con ello se usa *"pygame.mouse"* para obtener la posición en X del ratón para simular luego esa fuerza.

```
# ----------------------------
# Control por posición del mouse
# ----------------------------
mouse_x, _ = pygame.mouse.get_pos()
x_pix = int(WIDTH // 2 + state[0] * escala_pixeles)
distancia_pix = mouse_x - x_pix
F = k_mouse * distancia_pix / escala_pixeles  # convierte a Newtons aprox
```

Figura 8. Medición de la fuerza del mouse.

Luego se ponen las variables para controlar el modelado

```
# ---------------------------
# Texto con variables físicas
# ---------------------------
labels = [
    f"x = {state[0]:.2f} m",
    f"dx = {state[1]:.2f} m/s",
    f"θ = {np.degrees(state[2]) % 360:.2f}°",
    f"dθ = {np.degrees(state[3]):.2f} °/s",
    f"F = {F:.2f} N"
]
for i, label in enumerate(labels):
    text = font.render(label, True, (0, 0, 0))
    screen.blit(text, (10, 10 + i * 25))

pygame.display.flip()
clock.tick(50)
```

Figura 9. Variables físicas a controlar.

Control péndulo PID

Para el control PID se usa el modelo obtenido y las constantes en Matlab, con ello se procede a poner las constantes PID, límites físicos y el estado inicial.

```
# Límites
force_limit = 25.0   # Límite fuerza del motor
theta_ref_limit = math.radians(18)  # límite del angulo
x_limit = 3  # m, maximo que el carro puede moverse

def initial_state():
    # estado: [x, x_dot, Angulo Inicial, theta_dot]
    return np.array([0.0, 0.0, math.radians(8.0), 0.0], dtype=float)
```

Figura 10. Condiciones iniciales.

Para linealizar se usa el método de Rungen-Kutta de orden 4, que a diferencia de Euler permite iterar muchas más veces permitiendo que la señal de control siga mejor la referencia como se ve en la figura 11.

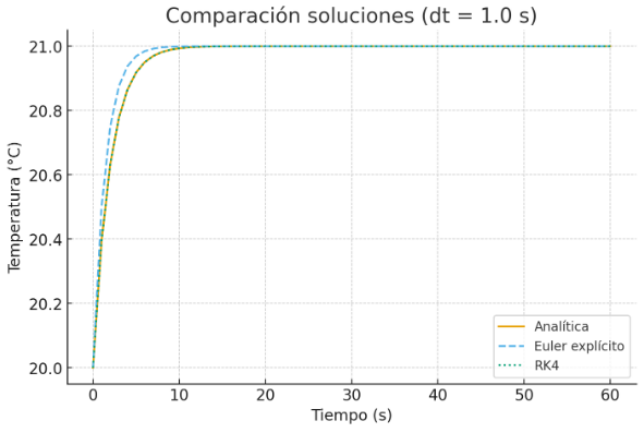

Figura 11. Comparación Euler y Rugen-Kutta

```
def f_nonlinear(y, F):
    x, x_dot, th, th_dot = y
    st = math.sin(th)
    ct = math.cos(th)
    Den = (I + m * l**2) * (M + m) - (m * l * ct)**2
    if abs(Den) < 1e-9:
        Den = 1e-9

    Fx = F - b * x_dot - m * l * th_dot**2 * st
    T = m * g * l * st + b_p * th_dot

    th_dd = ((m * l * ct) * Fx - (M + m) * T) / Den
    x_dd  = ((I + m * l**2) * Fx + (m * l * ct) * T) / Den
    return np.array([x_dot, x_dd, th_dot, th_dd], dtype=float)

# RK4 integrator step
def rk4_step(y, F, dt):
    k1 = f_nonlinear(y, F)
    k2 = f_nonlinear(y + 0.5*dt*k1, F)
    k3 = f_nonlinear(y + 0.5*dt*k2, F)
    k4 = f_nonlinear(y + dt*k3, F)
    return y + dt*(k1 + 2*k2 + 2*k3 + k4)/6.0
```

Figura 12. Modelado y método Rungen-Kutta de 4to orden.

Con ello al igual que en el péndulo por ratón, se usa *"pygame"* para visualizar el control del péndulo, adicionalmente se le pide a ChatGPT ayuda para poner la función de 2 botones, RUN y RESET.

Control péndulo difuso

Las diferencias entre un control PID y un control difuso es que el control PID es un análisis más matemático, mientras que el difuso depende de reglas lingüísticas como, ángulo grande, ángulo pequeño, etc.

Haciendo uso de la clase que nos dio el docente sobre control difuso se usan las mismas variables y condiciones que se dieron en clase para el código, por lo tanto, se tienen las siguientes condiciones.

Entrada 1 (Ángulo, θ):

- AMN → Ángulo muy negativo
- AN → Ángulo negativo
- AP → Ángulo positivo
- AMP → Ángulo muy positivo

Entrada 2 (Velocidad angular, θ̇):

- VAN → Velocidad angular negativa
- VAP → Velocidad angular positiva

Salida (Fuerza, F):

- XMN → Fuerza muy negativa
- XN → Fuerza negativa
- X0 → Fuerza neutra (cero)
- XP → Fuerza positiva
- XMP → Fuerza muy positiva

Es decir, si el ángulo es **positivo** y además la velocidad es hacia la derecha (**VAP**), se empuja el carro hacia la derecha (**XP**) para compensar; si el ángulo es **positivo** pero la velocidad es hacia la izquierda (**VAN**), conviene no aplicar fuerza (**X0**) porque la inercia lo ayuda a corregirse; similar lógica para cuando el ángulo es negativo; los extremos (**AMP** y **AMN**) necesitan fuerzas más grandes (**XMP**, **XMN**) para contrarrestar el desbalance.

```
# Reglas difusas
rules = [
    ("ap", "vap", "xp"),
    ("ap", "van", "x0"),
    ("an", "vap", "x0"),
    ("an", "van", "xn"),
    ("amp", "vap", "xmp"),
    ("amp", "van", "xp"),
    ("amn", "van", "xn"),
    ("amn", "vap", "xmn"),
]
```

Figura 13. Modelado y método Rungen-Kutta de 4to orden

II. RESULTADOS

Péndulo con mouse.
Simulando se obtiene la siguiente interfaz.

Figura 14. péndulo invertido controlado por ratón

En este caso se logra controlar con el ratón el carrito, si bien se debe tener practica al inicio para controlar la fuerza que se aplique a este debido a que depende de los dpi que tenga el mouse saber con cuanta fuerza se va a mover.

Pendulo PID

Como se obtuvieron constantes Kp y Kd, se tiene un controlador PD, con ello e ingresadas en el código se tiene la siguiente interfaz.

Figura 15. péndulo invertido con control PD

En este se logra un control rápido y eficaz con cualquier condición inicial que se coloque en la línea 32 del código.

Control péndulo difuso.

Cuando se compila el código se obtiene la siguiente interfaz.

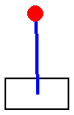

Figura 16. péndulo invertido con control PD

En este caso si bien se tiene un control un poco más lento que el control PD, pero en este caso el carrito termina corriendo hasta el infinito para mantener vertical el péndulo.

Webots

La implementación en webots no se logro concretar debido a que el Robot terminaba atravesando el suelo, error el cual no se pudo resolver por más que la configuración del suelo se encontraba de buena forma y la compilación del código no arrojaba algún tipo de error.

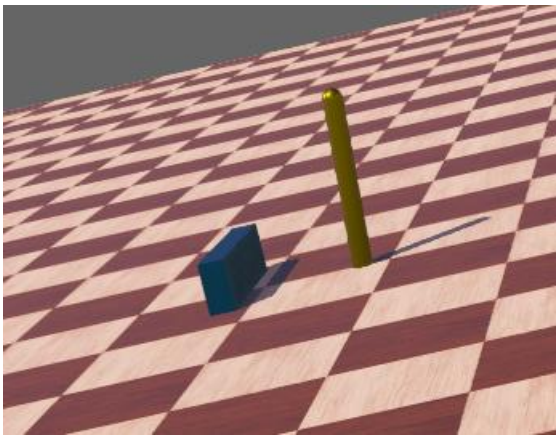

Figura 17. péndulo invertido con control PD

III. CONCLUSIONES

El uso del control manual con ratón permite una interacción intuitiva con el sistema, aunque requiere práctica inicial. La sensibilidad depende de los DPI del dispositivo, lo que introduce una variabilidad externa al sistema y lo hace menos predecible.

El controlador PD demostró un desempeño superior al permitir estabilizar el péndulo de manera rápida y confiable, incluso bajo diferentes condiciones iniciales. Esto confirma la efectividad de los controladores clásicos en sistemas dinámicos inestables.

El control difuso mostró capacidad para mantener el péndulo en posición vertical, pero lo hizo con un comportamiento menos eficiente, ya que el carro tiende a desplazarse indefinidamente. Esto evidencia que, aunque el control difuso es flexible, requiere ajustes más finos en sus reglas y funciones de pertenencia para igualar el rendimiento de un controlador PD.

La implementación en Webots presentó limitaciones técnicas debido a problemas con la simulación física, lo que resalta la importancia de una correcta configuración del entorno virtual y de comprender los parámetros de simulación antes de intentar validar un modelo de control en un entorno más complejo.