Laboratorio 8 – Sistemas de Control: "Two-Wheel Inverted Pendulum"

1st Imar Nayeli Jiménez Arango *1007424872*
2nd Oscar Andrés Gutiérrez Rivadeneira *1193515015*
3rd Angee Lorena Ocampo Ramirez *1004520456*

I. Introducción

En el campo del control son objeto de análisis diversos sistemas, los cuales, debido a su complejidad, son representados mediante modelos no lineales, entre ellos se encuentra el "Two-Wheel Inverted Pendulum", este consiste en una especie de péndulo invertido, que en el caso del presente informe, contrario a poseer una varilla que gira libremente, se compone de un cuerpo rígido de tres pisos, el cual debido a que su centro de gravedad no es del todo estable, tiende a caer hacia los lados.
Para simplificar el análisis y manipulación de este tipo de sistemas, es necesario usar métodos que permitan linealizar estos modelos como se verá en el presente documento.

II. Procedimiento

A. Análisis del sistema

Con el propósito de determinar el comportamiento del péndulo invertido, se inicia por plantear un diagrama simple del sistema a modelar con se observa en la figura 1.

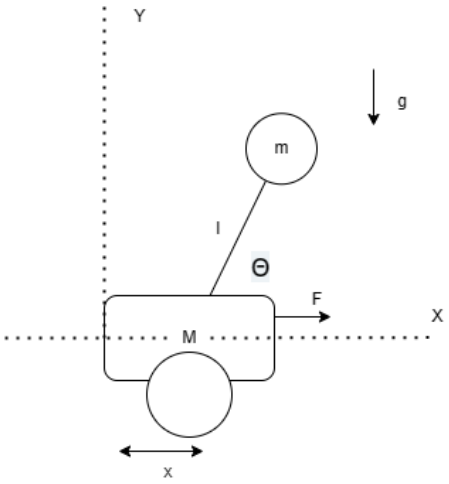

Fig. 1: Diagrama del péndulo invertido

Ahora, se procede a enunciar las variables usadas en este.

- M = Masa del chasis, este incluye las llantas ubicadas en el primer nivel del péndulo, así como el peso de este, también se tiene en cuenta el peso de los motores.
- m = Masa de
- l = distancia desde el centro del chasis hasta el eje de la masa m.
- g = Gravedad.
- x = Desplazamiento en la horizontal.
- θ = Ángulo formado por el péndulo con respecto a la horizontal.
- F = Fuerza externa aplicada al péndulo.

Al realizar el respectivo análisis del sistema, se pudo identificar que este solo posee dos grados de libertad, los cuales corresponden a su desplazamiento en la horizontal, X, y el ángulo que se forma con respecto a esta, que es θ; con lo anterior, se elige el método de las ecuaciones de Lagrange para plantear el modelo como se ve a continuación:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{1}$$

$$L = T - V \tag{2}$$

Donde T es la energía cinética y V es la energía potencial. Para la velocidad del péndulo se obtiene:

$$V_P = (V_x, V_y) = (\dot{x} + lcos(\theta)\dot{\theta}, -l\dot{\theta}sen(\theta)) \tag{3}$$

$$V_P^2 = \dot{x}^2 + 2lcos(\theta)\dot{\theta}\dot{x} + l^2\dot{\theta}^2 \tag{4}$$

Para la energía cinética:

$$T_xp = \frac{1m(\dot{x}^2 + 2lcos(\theta)\dot{\theta}\dot{x} + l^2\dot{\theta})}{2} \tag{5}$$

$$T_x = \frac{1M\dot{x}^2}{2} + \frac{1m(\dot{x}^2 + 2lcos(\theta)\dot{\theta}\dot{x} + l^2\dot{\theta})}{2} \tag{6}$$

Para la energía potencial:

$$U = -mgh = -mglcos(\theta) \tag{7}$$

Ecuación para $\dot{x}$:

$$[M+m]\ddot{x} + ml\ddot{\theta}cos(\theta) - ml\dot{\theta}^2sen(\theta) = 0 \tag{8}$$

Ecuación para θ:

$$ml\ddot{x}cos(\theta) + l^2\dot{\theta}m - mglsen(\theta) = 0 \tag{9}$$

Una vez encontradas las ecuaciones de segundo orden que rigen el comportamiento del sistema, se procede a usar la matriz Jacobiana mediante la herramienta de simulación de MATLAB con el propósito de linealizarlo; este procedimiento permite calcular las razones de cambio de los estados, los cuales dependen de funciones no lineales, lo que quiere decir

que dichas funciones dependen de los estados y las entradas; este procedimiento es realizado mediante las derivadas parciales de cada una de las funciones con respecto a cada uno de los estados para obtener la matriz A, del mismo modo se realiza para el vector B; sin embargo, en este caso la derivada parcial se realiza con respecto a cada una de las entradas; seguidamente, se procede a evaluar tanto la matriz como el vector en cada uno de sus puntos de equilibrio. Este procedimiento es realizado mediante el comando **jacobian** de la herramienta de simulación, después se procede a establecer los puntos de equilibrio y los valores medidos del sistema, los cuales son:

- M = 0.187 kg
- m = 0.3387 kg
- l = 0.143 m.
- g = 9.81 $\frac{m}{s^2}$
- $x(0)$ = 0 m.
- $\dot{x}(0)$ = 0 $\frac{m}{s}$.
- $\theta(0)$ = 0.
- $\dot{\theta}(0)$ = 0 $\frac{rad}{s}$.

Con lo anterior se llega a lo siguiente:

$$\begin{matrix} 0 & 1.0000 & 0 & 0 \\ 0 & 0 & -17.7682 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 192.8543 & 0 \end{matrix}$$

Fig. 2: Matriz A

$$\begin{matrix} 0 \\ -37.3958 \\ 0 \\ 405.8907 \end{matrix}$$

Fig. 3: Matriz B

Una vez obtenido el espacio de estados linealizado, mediante el comando ss, se lleva el sistema a un modelo de espacio de estados y se procede a graficar su respuesta con respecto al escalón unitario, como se ve a continuación:

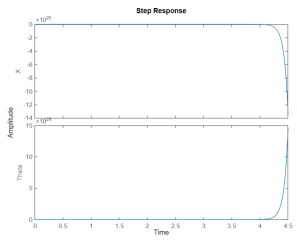

Fig. 4: Respuesta del sistema ante el escalón

Listing 1: Código de MATLAB

```
 Resolviendo las ecuaciones
     diferenciales
Sol = solve([eqx1,eqx2],[x1dd,x2dd]);
Sol.x1dd = simplify(Sol.x1dd);
Sol.x2dd = simplify(Sol.x2dd);

syms y1 y2 y3 y4
fx1 = y2
fx2=subs(Sol.x1dd,{x1,x1d,x2,x2d},{y1,y2
    ,y3,y4})
fx3 = y4
fx4=subs(Sol.x2dd,{x1,x1d,x2,x2d},{y1,y2
    ,y3,y4})

F = [fx1;fx2;fx3;fx4]

Hallando el Jacobiano de cada un de las
    matrices
A_symbolic = jacobian(F, [y1,y2,y3,y4])
B_symbolic = jacobian(F, u2);

Evaluando en los puntos de equilibrio
A_algebraic = simplify(subs(A_symbolic,
    {M m L g y1 y2 y3 y4 u2}, ...
    [sym(0.187) sym(0.3387) sym(0.143)
        sym(9.81) sym(0) sym(0) sym(0)
        sym(0.00) sym(0.00)]));
B_algebraic = simplify(subs(B_symbolic,
    {M m L g y1 y2 y3 y4 u2}, ...
    [sym(0.187) sym(0.3387) sym(0.143)
        sym(9.81) sym(0) sym(0) sym(0)
        sym(0.00) sym(0.00)]));

Obteniendo las expresiones num ricas
A_eval = eval(A_algebraic);
B_eval = eval(B_algebraic);
C = [1 0 0 0; 0 0 1 0];
D = [0;0]

Llevando el sistema a un modelo del
    espacio de estados y evaluando la
    respuesta ante el escal n
sysLTI = ss(A_eval,B_eval,C,D);
step(sysLTI)
```

Al evidenciar que la respuesta del sistema es inestable, se continúa con llevar el modelo de espacio de estados obtenido al entorno de simulación de MATLAB, Simulink, con el fin de desarrollar el controlador que permita que el modelo se encuentre continuamente en un punto de equilibrio para ser implementado posteriormente. Con base en el espacio de estados obtenida se procede a hallar la función de transferencia e implementar la planta en lazo abierto, además, se grafica su respuesta ante el escalón como se aprecia en las figuras 5 y 6.

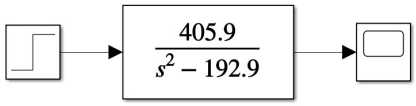

Fig. 5: Modelo del sistema en lazo abierto

Seguidamente, se anexa el control PID al diagrama de la planta y se adiciona la retroalimentación al sistema, asimismo, se comienza a probar diferentes valores para cada una de las constantes del controlador con base en los métodos de sintonía

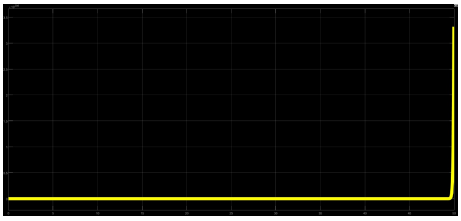

Fig. 6: Respuesta ante el escalón unitario

aprendidos, también, teniendo en cuenta el efecto que tiene cada una de ellas sobre el sistema; gracias a lo anterior se encuentran los valores $K_p = 1$, $K_i = 0.09$ y $K_d = 10$.

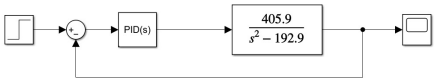

Fig. 7: Sistema en lazo cerrado con el control PID.

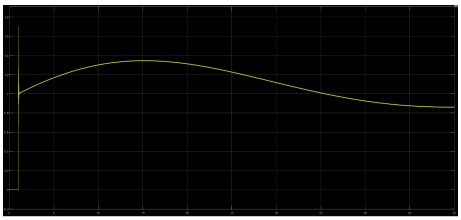

Fig. 8: Respuesta ante el escalón unitario del sistema con el controlador elaborado.

B. Sistema real

Tras los controladores obtenidos anteriormente, se implementa el sistema real, el cual consiste en una especie de carrito de dos llantas, conformado por tres pisos. Se identifica que la variable de estado que será tomada como referencia es θ, por lo que se usa un sensor MPU6050 que permite medir el ángulo de manera continua para que el sistema se mantenga en un estado estable, estos datos llegan al microcontrolador, que en este caso es la ARDUINO UNO, los cuales serán procesados por el mismo, a través de un programa desarrollado en el entorno de programación Arduino IDE. El código para la implementación es el siguiente:

III. Implementación física

Con base en lo obtenido en la sección anterior, se implementa el controlador en el sistema real, para ello fue necesario el uso de un microcontrolador, el cual tomaba la señal del sensor y posterior al cálculo del valor de salida del controlador, enviaba la señal a los motores, ver figura 9.

IV. Lógica implementada

El microcontrolador utilizado fue un Arduino UNO, que requirió el uso de tres librerías principales: ”PID”, ”MPU6050_6Axis_MotionApps20” y ”LMotorController”. Estas librerías permitieron calcular la respuesta necesaria, leer

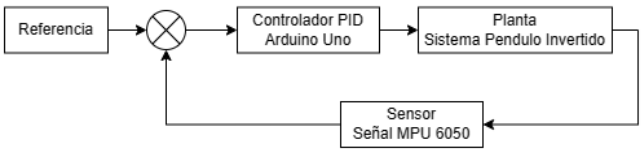

Fig. 9: Diagrama de bloques del sistema

las entradas de la IMU y enviar las señales al controlador de motores $L298N$.

En primer lugar, es necesario calibrar la IMU para obtener los valores de offset, que ya vienen de fábrica, y deben conocerse para evitar mediciones imprecisas. Una vez calibrada la IMU, el sistema comenzará a tomar medidas utilizando interrupciones en el controlador, que indican cuándo los datos están listos para ser procesados.

Los motores tienen la función de ejecutar la respuesta obtenida por el controlador. Para ello, se envía una señal PWM que permite regular la velocidad de los motores. Además, los motores deben mantener una velocidad mínima para evitar quedarse estáticos y así reducir el tiempo de respuesta, ya que no deben romper la inercia de la máquina. También es importante atenuar la velocidad para evitar respuestas bruscas, logrando un movimiento controlado y evitando un efecto contrario en el intento de estabilizar el sistema.

El controlador PID actúa como un intermediario entre los componentes anteriores, recibiendo señales de la IMU y comparándolas con un punto de referencia. Al estar en modo automático, el controlador recalcula los valores de salida para mantener el valor de la consigna deseada. Además, la configuración directa del controlador asegura que la respuesta se incremente en correspondencia con el aumento de la variable de proceso, en este caso, el ángulo. Es importante destacar que la salida fue limitada a un rango de -255 a 255, correspondiente a la amplitud máxima que puede entregar el Arduino en una señal PWM.

Con el fin de evaluar los valores obtenidos en el sistema real se ingresan los valores de las constantes en la planta en lazo cerrado y se evalúa su respuesta ante el escalón unitario COMO se ve a continuación:

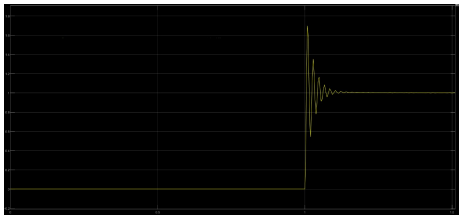

Fig. 10: Respuesta ante el escalón unitario del sistema con el controlador implementado en la arduino.

V. Conclusiones

Es importante destacar que el controlador obtenido mediante simulación difiere del implementado en el sistema real. Esto

se debe a las diferencias entre el sistema simulado y el sistema real, resultado de las simplificaciones y aproximaciones utilizadas en el modelo.

Es fundamental mencionar que la rapidez en la adquisición de datos de la IMU es crucial. Si los datos se toman lentamente, la respuesta calculada por el controlador se basará en momentos pasados, lo que puede resultar en una respuesta insuficiente para estabilizar el sistema.

Es necesario conocer las capacidades y limitaciones del actuador para evitar saturar la respuesta o, por el contrario, saber cuándo limitarla. En el caso de los motores, se estableció un límite inferior para garantizar que siempre estén listos para responder, teniendo en cuenta que la respuesta puede ser lenta debido a la inercia del sistema