Laboratorio virtual para el control de sistemas dinámicos subactuados

Karina Moreno Aguilar
Instituto Politécnico Nacional
ESIME Culhuacán
Ciudad de México
karimoa02@hotmail.com

Domingo Cortés Rodríguez
Instituto Politécnico Nacional
ESIME Culhuacán
Ciudad de México
domingo.cortes@gmail.com

Francisco Hernández Salas
Instituto Politécnico Nacional
ESIME Culhuacán
Ciudad de México
fhernandezs094@gmail.com

Diego Martínez Guillén
Instituto Politécnico Nacional
ESIME Culhuacán
Ciudad de México
diegomtzg@hotmail.com

***Abstract*—Se presenta el desarrollo de un laboratorio virtual de sistemas mecánicos subactuados implementado en un sitio web. El laboratorio virtual creado contiene: videos tutoriales acerca del modelado de los sistemas, una herramienta de simulación, un instrumento de visualización dinámica, los resultados son presentados a través de una animación 3D. El usuario puede interactuar con el laboratorio seleccionando el modelo de sistema subactuado, a este modificar sus parámetros, implementar o modificar un controlador existente, realizar la simulación y observar los resultados en la animación. Los modelos disponibles son: el péndulo simple, péndulo invertido sobre un móvil, péndulo de Furuta y péndulo doble.**

***Index Terms*—double pendulum, dynamical systems, Furuta pendulum, inverted pendulum, simple pendulum, Underactuated Mechanical Systems**

I. Introducción

Para entender el motivo de este trabajo, es necesario tener clara la definición de sistema mecánico y sistema subactuado. Un sistema mecánico es un conjunto de elementos que tienen como propósito principal transmitir movimiento, mientras que un sistema subactuado es aquel sistema que cuenta en su estructura con más grados de libertad que actuadores. [1]

Se decide trabajar con sistemas subactuados ya que son frecuentemente utilizados en laboratorios de centros de investigación, con el fin de estudiar conceptos teóricos realacionados con el control automático.

Un laboratorio virtual de sistemas mecánicos subactuados se definirá como una simulación en computadora de los diferentes sistemas mecánicos propuestos en un ambiente interactivo, es decir, se puede simular el comportamiento de estos sistemas y aunque no se interactúa con ellos de forma física, la experimentación con estos sistemas virtuales será comparable con la realidad.

El propósito del laboratorio virtual es apoyar al entendimiento y funcionamiento de sistemas complejos y compararlos con sistemas físicos reales. Algunos ejemplos son: cohetes que requiere un control para mantener la posición vertical durante el despegue, un robot bípedo que pueda controlar el equilibrio en su posición verical, transportes de ruedas con un solo eje (Segway), entre otros.

Una problemática que existe para el estudio de este tipo de sistemas, radica en la necesidad de operar el equipo de forma presencial en un laboratorio donde se tenga acceso de forma física a los diferentes sistemas, así como a los equipos de medición y control.

Para solucionar este problema se construyó un laboratorio virtual que nos permite tener acceso al los sistemas de forma virtual y poder realizar simulaciones, propuestas de control y observar su funcionamiento mediante animaciones 3D en un servidor Web.

II. Componentes del laboratorio virtual

El laboratorio virtual está montado en un servidor Web, por ahora utilizando un servidor local para continuar con su desarrollo. El laboratorio virtual consta de 4 elementos principales, los cuales son: A) Videos tutoriales, B) Modelos matemáticos, C) Solución de ecuaciones diferenciales y D) Animaciones, y se describen a detalle a continuación.

A. Videos Tutoriales

Los videos tutoriales se contruyeron en Manim, una herramienta gratuita de animación, creada por Grant Sanderson matemático de Stanford y dueño del canal de YouTube 3Blue1Brown. Manim se especializa en temas científicos, principalmente de carácter matemático, por lo que está basado en comandos de LaTeX. [2]
Manim está compuesto por cinco partes:

- LaTeX. Como procesador de textos, el cual utiliza los comandos TeX (con algunas excepciones), para la escritura de fórmulas.
- PyCairo. Módulo de Python que proporciona enlaces para la biblioteca de gráficos de cairo, se utiliza para crear archivos SVG vector en Python.
- FFMPEG. Biblioteca que se utiliza para grabar, convertir y hacer streaming de audio y vídeo.
- SoX. Permite la reproducción, grabación, lectura y escritura en archivos de audio con distintos formatos.
- Python. Lenguaje en el que se escriben los comandos para crear las animaciones y une los puntos anteriores.

B. Modelos Matemáticos

El modelado matemático de los sistemas se obtiene utilizando el método de Euler - Lagrange, el procedimiento no es mostrado para evitar extender el artículo, se muestran los resultados para el péndulo simple, péndulo invertido sobre un móvil, péndulo de Furuta y péndulo doble en (1),(2) ,(6) y (12) respectivamente.

1) Péndulo Simple: Utilizando el diagrama de Fig. [1], se obtiene el modelo matemático mostrado en Ec.(1).

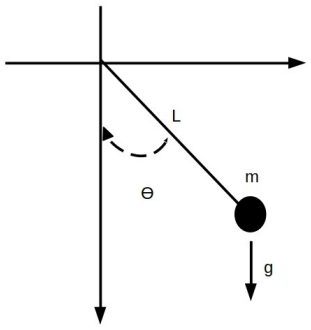

Fig. 1. Diagrama del Péndulo Simple.

$$\frac{d}{dt}\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ -\frac{g}{l}\sin x_0 - kx_1 + \tau \end{bmatrix} \tag{1}$$

Donde:
m: masa del péndulo.
l: longitud del péndulo.
k: fricción del pivote
θ: ángulo
g: gravedad.

2) Péndulo invertido sobre móvil: Utilizando el diagrama de la Fig. [2], se obtiene el modelo matemático mostrado en Ec. (2)

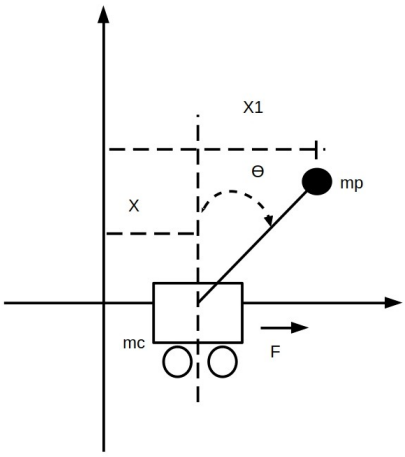

Fig. 2. Diagrama del Péndulo invertido sobre móvil.

$$M\ddot{q} + C\dot{q} + gq = F \tag{2}$$

$$M = \begin{bmatrix} m_c + m_p & m_p l\cos q_2 \\ m_p l\cos q_2 & mpl^2 \end{bmatrix} \tag{3}$$

$$C = \begin{bmatrix} 0 & -m_p l\dot{q}_2 \sin q_2 \\ 0 & 0 \end{bmatrix} \tag{4}$$

$$g = \begin{bmatrix} 0 \\ m_p gl\sin q_2 \end{bmatrix} \tag{5}$$

Donde:
m_c: masa del carro.
m_p: masa del péndulo
l: longitud del péndulo.
θ: ángulo
g: gravedad.

3) Péndulo de Furuta: Utilizando el diagrama de la Fig. [3], se obtiene el modelo matemático mostrado en Ec. 6 [3].

$$M\ddot{q} + C\dot{q} + gq + f_v + f_c = F \tag{6}$$

$$M = \begin{bmatrix} J_0 + ml_b^2 + ml_p^2\sin^2(\theta_1) & ml_b l_p\cos\theta_1 \\ ml_b l_p\cos(\theta_1) & J_1 + ml_p^2 \end{bmatrix} \tag{7}$$

$$C = \begin{bmatrix} ml_p^2\dot{q}_1\cos\theta_1\sin\theta_1 & ml_p^2\dot{q}_0\cos\theta_1\sin\theta_1 - ml_b l_p\dot{q}_1\sin\theta_2 \\ -ml_p^2\dot{q}_0\cos\theta_1\sin\theta_1 & 0 \end{bmatrix} \tag{8}$$

$$g = \begin{bmatrix} 0 \\ -mgl_p\sin\theta_1 \end{bmatrix} \tag{9}$$

$$f_v = \begin{bmatrix} f_{v1} \\ f_{v2} \end{bmatrix} \tag{10}$$

$$f_c = \begin{bmatrix} f_{c1} \\ f_{c2} \end{bmatrix} \tag{11}$$

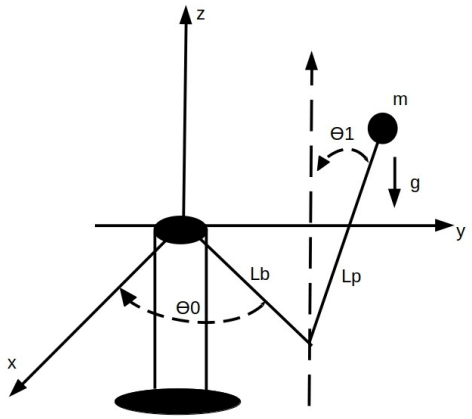

Fig. 3. Diagrama del Péndulo de Furuta.

Donde:
m: masa del péndulo.
l_p: longitud del péndulo.
l_b: longitud del brazo.
θ_0: ángulo del brazo respecto al eje x.
θ_1: ángulo del péndulo respecto al eje y.
m: masa del péndulo.
g: gravedad.
J_0: constante de inercia del brazo.
J_1: constante de inercia del péndulo.
f_{v1}: coeficiente de fricción viscosa del motor.
f_{v2}:coeficiente de fricción viscosa del péndulo.
f_{c1}: coeficiente de fricción de Coulomb del motor.
f_{c2}: coeficiente de fricción de Coulomb del péndulo.

4) Doble Péndulo: Utilizando el diagrama de la Fig. [4], se obtiene el modelo matemático de la misma forma que el de la en Ec.(12).

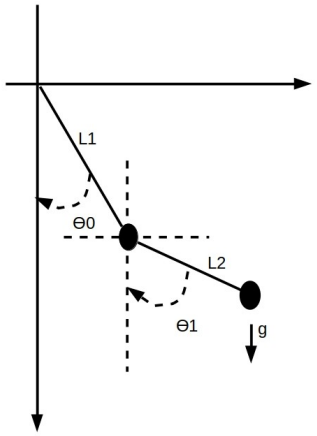

Fig. 4. Diagrama del Doble Péndulo.

$$M\ddot{q} + C\dot{q} + gq = F \tag{12}$$

$$M = \begin{bmatrix} (m_1 + m_2)\, l_1 & l_2 m_2 \cos(q_1 - q_2) \\ l_1 \cos(q_1 - q_2) & l_2 \end{bmatrix} \tag{13}$$

$$C = \begin{bmatrix} 0 & -m_2 l_2 x_3 \sin(q_1 - q_2) \\ l_1 x_1 \sin(q_1 - q_2) & 0 \end{bmatrix} \tag{14}$$

$$g = \begin{bmatrix} (m_1 + m_2)\, g l_1 \sin q_1 \\ m_2 g l_2 \sin q_2 \end{bmatrix} \tag{15}$$

Donde:
m_1: masa del péndulo 1.
m_2: masa del péndulo 2.
l_1: longitud del péndulo 1.
l_2: longitud del péndulo 2.
θ_0: ángulo del péndulo 1 respecto al eje y.
θ_1: ángulo del péndulo 2 respecto al eje y.
m: masa del péndulo.
g: gravedad.

Ya que tenemos los modelos matemáticos, se van a expresar II-B2, II-B3 y II-B4 de la forma (16) para tener un modelo general para representar estos los sistemas y sea más fácil manipularlos.

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ \dot{q_1} \\ q_2 \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} \dot{q_1} \\ F_1 + G_1\tau \\ \dot{q_2} \\ F_2 + G_2\tau \end{bmatrix} \tag{16}$$

Donde:

$$\begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \frac{1}{|M(q)|}\begin{bmatrix} M_{22} \\ -M_{21} \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \frac{1}{|M(q)|}\begin{bmatrix} m_{22}Z_1 - M_{12}Z_2 \\ -M_{21} + M_{11}Z_2 \end{bmatrix} \tag{18}$$

$$Z_1 = -C_{11}\dot{q_1} - C_{12}\dot{q_2} - g_1 - f_{v1} - f_{c1} \tag{19}$$

$$Z_2 = -C_{21}\dot{q_1} - C_{22}\dot{q_2} - g_2 - f_{v2} - f_{c2} \tag{20}$$

Para II-B2 y II-B4 f_{v1} , f_{c1}, f_{v2} y f_{c1} tienen un valor de cero.

C. Solución de ecuaciones diferenciales

Para la solución de las ecuaciones diferenciales de los sistemas dinámicos subactuados se programó en JavaScript el método de Runge-Kutta de cuarto orden, el cual se utiliza para aproximar las soluciones de ecuaciones diferenciales del sistema y el controlador. Este método es llamado cada vez que se requiera una simulación. Este método está programado con un paso de integración de $stepSz$: 0.0068 ya que los sistemas subactuados propuestos son muy rápidos y "Three.js" no alcanza a construir las animaciones.

D. Animaciones

Para la creación de animaciones se utilizó ”Three.js”. ”Three.js” es una biblioteca de acceso libre y está escrita en JavaScript para crear animaciones 3D en un navegador web [4].

Cada animación de los sistemas propuestos está compuesta por:

- Escena. La escena se compone de todos los elementos a mostrar, en la escena se añaden objetos 3D, cámaras, sonidos, étc.
- Geometría. Objetos que representan las coordenadas de las piezas geométricas por ejemplo una esfera, cubo, plano, un perro, un árbol, étc. ”Three.js” propociona muchas primitivas geométricas, sin embargo se puede crear geometría personalizada o cargarla desde un archivo externo.
- Material. Los objetos material, representan las propiedades visuales de la geometría, como el color y brillo, étc.
- Mesh. Los objetos mesh son objetos compuestos que se añaden a la escena, estos contienen como mínimo una geometría y su material.
- Cámara. Las cámaras son la perspectiva con que se observa el lienzo, no tienen una representación gráfica. Una escena puede contener tantas cámaras como se desee, pero sólo podrá haber una de ellas activa para visualizar el lienzo, se puede alternar entre una y otra sin limitaciones.
- Lienzo. Es el elemento donde se observan las animaciones.
- Renderer. Para crear un render se necesita:
  - Crear sus propiedades del tamaño y color.
  - Asignarle una escena y una cámara.

  Una vez cumpliendo lo anterior, renderiza en el lienzo los objetos que contenga la escena, cada vez que se desee que los cambios realizados en la escena se reflejen en el lienzo, se deberá llamar a este método. [5]

Fig. 5. Mesh péndulo Simple

III. Resultados

A continuación se describen los pasos necesarios para navegar en el laboratorio virtual descrito en este artículo:

Fig. 6. Mesh péndulo invertido sobre un móvil.

Fig. 7. Mesh péndulo de Furuta.

1) Ejecutar un servidor web: Por el momento, se utiliza un servidor local para ejecutar el laboratorio virtual web, para fines prácticos se utilizará el servidor de Python, ya que con una línea de comandos en la terminal se ejecuta el servidor en la máquina. La linea de comandos se muestra en listing 1.

2) Ejecutar el navegador de su preferencia: En la barra de direcciones escribir el código mostrado en listing, en

Fig. 8. Mesh de péndulo doble

Listing 1. Servidor en Python

```
python3 -m http.server 3000
```

Listing 2. Acceso el servidor local por el puerto 3000

```
localhost:3000
```

donde llamamos al servidor local por el puerto que activamos anteriormentecomo se muestra en listing 2.

3) Navegación por el laboratorio virtual: Para ingresar al laboratorio virtual abrir el folder *html* y acceder a index.html, ahí se tiene acceso a la página principal, que cuenta con un menú de opciones como se puede observar en 9, cada una de las pestañas contiene un submenú como se muestra en 10. Describiendo el submenú se tiene:

- Modelo matemático. Se puede acceder a los videos tutoriales como se muestra en 11.
- Animación. Accediendo a esta pestaña se tiene acceso a la animación 12

Para cada sistema se tiene un submenú similar, que nos lleva a los diferentes videos y animaciones, para mostrar las evidencias se utilizará el péndulo de Furuta, los pasos a seguir son los mismos para cada uno de los sistemas. Se muestran las animaciones para los 4 sistemas mecánicos subactuados trabajados a lo largo del artículo en la figuras Fig. [5], Fig.[6], Fig[7] y Fig.[8].

Fig. 9. Index.html.

Fig. 10. Menú.html.

Por el momento para realizar cambios en los parámetros y condiciones iniciales de los sistemas se ingresa a la ruta .../js/main.js desde un editor de texto, se localiza la variable $pendpars$ para los parámetros del sistema, y la variable pic para las condiciones iniciales del sistema como se muestra en

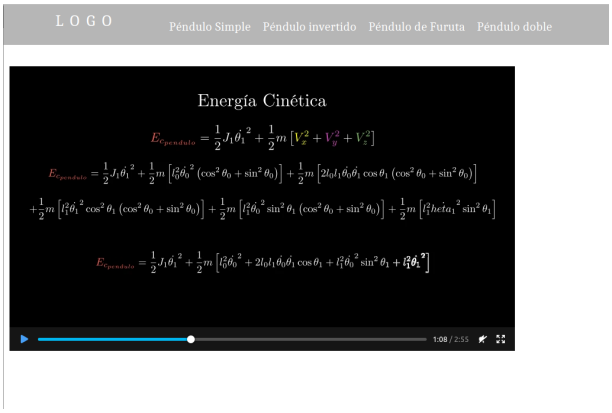

Fig. 11. Videos Tutoriales

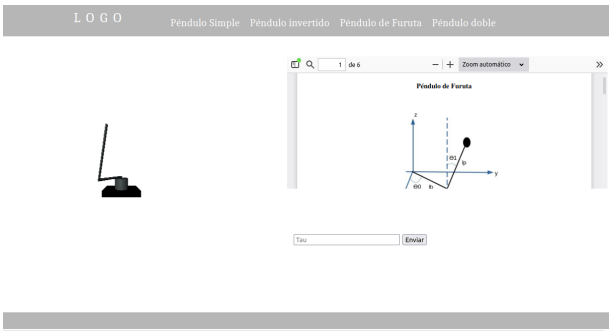

Fig. 12. Animación

listing 4 y se realizan los cambios en los parámetros que el usuario desee, basta con guardar el archivo main.js y refrescar el sitio web. Para el resto de los sistemas se procede de la misma manera.

Para ingresar un controlador por el momento se tiene que acceder a la ruta ../js/sys.js y localizar la función de control, y ahí se programa el controlador deseado, para ejemplificar se muestra un control propuesto para el péndulo de Furuta como se observa en listing 3.

Listing 3. Programar controlador

```
fPendulum.ctrl = function (x,t) {
        let w=3;
        let td1 = sin(w*t);
        let td1_p = cos(w*t);
        let td1_pp = - sin(w*t);
        let y= td1-x[0-x[2];
        let y_p = td1_p-x[1]-x[3];
        let kd1=10;
        let kp1= 30;
        let tau_f=(td1_pp+kd1*y_p+
        kp1*y-F)/G;
        let ro= 0.00123;
        let delta =10;
        let E1= 0.1;
        let E2 = 0.1;
        let p= E2*G2*x[2]+ G2*x[3]
        -ro*E1*G*y-delta*G*y_p;
        let kp2= 0.06;
        let kd2= 0.00123;
        let tau_c = -ro*p-kp2*G2*x[2]-
        kd2*G2*x[3];

        return tau= controlador;}
```

Listing 4. main.js

```
const pendpars = {g:9.8, lb:.9, j1:0.01,
lp:.3, mp:0.9, j2:0.0008795,
beta:1000, fv1:0.0083, fv2:0.0007,
fc1:0.0188, fc2:0.0087};

const pic = [0, 0,pi/4, 0];
```

IV. Conclusiones

En este artículo se ha descrito la primera versión de un laboratorio virtual de sistemas dinámicos subactuados. Dicho laboratorio permite animar diferentes sistemas dinámicos subactuados, así también experimentar con ellos al tener la capacidad de modificar sus parámetros, así como la posibilidad de implementar un control y visualizar todos los cambios y comportamientos del sistema en una animación 3D.

Con este laboratorio virtual se pretende que los usuarios desarrollen la parte intuitiva del comportamiento de los sistemas mecánicos subactuados, dejando de lado el desarrollo de códigos para lograrlo y enfocandose exclusivamente en su comportamiento adquiriendo un mayor sentido de como dirigir la experimentación y la búsqueda de controladores.
Las partes a desarrollar en el laboratorio son:

- Desarrollo de un interprete en donde se pueda ingresar la propuesta de control desde un cuadro de texto.
- Agregar un formulario para realizar el cambio de parámetros desde la animación.
- Realizar los videos tutoriales de las propuestas de control para cada sistema.

References

[1] Fantoni, C. and Lozano, R. (2002). *Non-linear Control for Underactuated Mechanical Systems*.

[2] Sanderson, G. (2020). *3b1b / manim . Recuperado 30 de junio de 2021, de github website: https://github.com/3b1b/manim*.

[3] Abelson, J. and Aguilar, C. (2018). *Motion Control of Underactuated Mechanical Systems*.

[4] Cabello, R. (2010). *three.js . Recuperado 30 de junio de 2021, de github website: https://github.com/mrdoob/three.js/*.

[5] Josa J. (2017). *Diseño de juegos 3D para web, Three.js - HTML5 y WebGL*.