FÍSICA COMPUTACIONAL

# TAREA 7

## PENDULOS

Presentado por:
Cinthia Alejandra Olvera Bautista

# Índice general

2

# Pendulos acoplados con un resorte

Los péndulos acoplados con un resorte representan un sistema físico fascinante que combina el movimiento oscilatorio de dos péndulos simples con la interacción proporcionada por un resorte que los conecta. Este sistema permite estudiar fenómenos como las oscilaciones normales y el intercambio de energía entre los péndulos, lo que ilustra conceptos fundamentales de la mecánica clásica, como el acoplamiento y la resonancia.

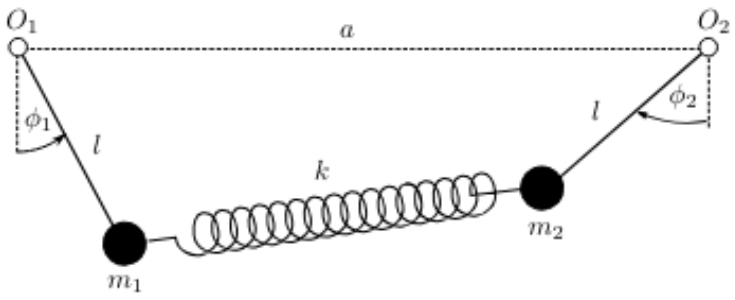


Figura 1.1: Imagen del sistema

## 1.1 Ecuaciones de Movimiento en Sistemas Acoplados

### 1.1.1 Derivación de las Ecuaciones Diferenciales

Consideremos dos péndulos simples de longitud $l$ y masa $m$, acoplados por un resorte de constante elástica $k$.

La energía cinética $T$ está dada por:

$$T = \frac{1}{2}ml^2\dot{\theta}_1^2 + \frac{1}{2}ml^2\dot{\theta}_2^2$$

La energía potencial $V$ es la suma de las contribuciones gravitacionales y el acoplamiento elástico:

$$V = mgl(1-\cos\theta_1) + mgl(1-\cos\theta_2) + \frac{1}{2}k\left[l(\sin\theta_1 - \sin\theta_2)\right]^2$$

Aquí:

- $mgl(1-\cos\theta_1)$ y $mgl(1-\cos\theta_2)$ representan las energías potenciales gravitacionales de cada péndulo.
- $\frac{1}{2}k\left[l(\sin\theta_1-\sin\theta_2)\right]^2$ corresponde a la energía almacenada en el resorte debido a la diferencia en las posiciones horizontales de las masas.

El Lagrangiano $\mathcal{L}$ se define como:

$$\mathcal{L} = T - V$$

Sustituyendo $T$ y $V$:

$$\mathcal{L} = \frac{1}{2}ml^2\dot{\theta}_1^2 + \frac{1}{2}ml^2\dot{\theta}_2^2 - \left[mgl(1-\cos\theta_1) + mgl(1-\cos\theta_2) + \frac{1}{2}kl^2(\sin\theta_1-\sin\theta_2)^2\right]$$

Agrupando términos:

$$\mathcal{L} = \frac{1}{2}ml^2\dot{\theta}_1^2 + \frac{1}{2}ml^2\dot{\theta}_2^2 - mgl(1-\cos\theta_1) - mgl(1-\cos\theta_2) - \frac{1}{2}kl^2(\sin\theta_1-\sin\theta_2)^2$$

Las ecuaciones de movimiento se derivan de las ecuaciones de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_i}\right) - \frac{\partial\mathcal{L}}{\partial\theta_i} = 0, \quad i = 1,2$$

El Hamiltoniano $\mathcal{H}$ se expresa en términos de las coordenadas generalizadas y sus momentos conjugados:

$$\mathcal{H} = \sum_{i=1}^{2} p_i\dot{\theta}_i - \mathcal{L}, \quad p_i = \frac{\partial\mathcal{L}}{\partial\dot{\theta}_i}$$

Las ecuaciones son

$$\ddot{\theta}_1 = -\frac{g}{l}\sin\theta_1 + \frac{k}{m}\cos\theta_1\,(\sin\theta_1-\sin\theta_2)$$
$$\ddot{\theta}_2 = \frac{-g}{l}\sin\theta_2 - \frac{k}{m}\sin\theta_1 - k \quad \theta_2\,(\sin\theta_1-\sin\theta_2)$$

Físicamente, el Hamiltoniano representa la energía total del sistema, que es la suma de la energía cinética y la energía potencial:

$$\mathcal{H} = T + V$$

Los momentos conjugados $p_i$ son:

$$p_1 = \frac{\partial\mathcal{L}}{\partial\dot{\theta}_1} = ml^2\dot{\theta}_1, \quad p_2 = \frac{\partial\mathcal{L}}{\partial\dot{\theta}_2} = ml^2\dot{\theta}_2$$

El Hamiltoniano $\mathcal{H}$ se calcula como:

$$\mathcal{H} = \sum_{i=1}^{2} p_i\dot{\theta}_i - \mathcal{L} - -$$

Sustituyendo $p_i = ml^2\dot{\theta}_i$ y el Lagrangiano:

$$\mathcal{H} = p_1\dot{\theta}_1 + p_2\dot{\theta}_2 - \mathcal{L}$$

$$\mathcal{H} = \frac{p_1^2}{2ml^2} + \frac{p_2^2}{2ml^2} + mgl(1-\cos\theta_1) + mgl(1-\cos\theta_2) + \frac{1}{2}kl^2(\sin\theta_1-\sin\theta_2)^2$$

### 1.1.2 Soluciones Aproximadas y Analíticas

Para pequeños ángulos, $\cos\theta \approx 1 - \frac{\theta^2}{2}$, y las ecuaciones de movimiento se simplifican a:

$$\ddot{\theta}_1 + \frac{g}{l}\theta_1 + \frac{k}{m}(\theta_1 - \theta_2) = 0$$

$$\ddot{\theta}_2 + \frac{g}{l}\theta_2 + \frac{k}{m}(\theta_2 - \theta_1) = 0$$

Estas ecuaciones pueden resolverse utilizando métodos analíticos para sistemas de ecuaciones diferenciales lineales.

## 1.2 Modos Normales de Oscilación

Los modos normales son soluciones en las que ambos péndulos oscilan con la misma frecuencia. En el modo simétrico, ambos péndulos oscilan en fase, mientras que en el modo antisimétrico, oscilan en oposición de fase.

### 1.2.1 Cálculo de Frecuencias Naturales

Las frecuencias naturales del sistema se obtienen resolviendo el determinante de la matriz de coeficientes del sistema de ecuaciones. Para el modo simétrico, la frecuencia es:

$$\omega_s = \sqrt{\frac{g}{l}}$$

Y para el modo antisimétrico:

$$\omega_a = \sqrt{\frac{g}{l} + \frac{2k}{m}}$$

La animación del fenómeno se puede ver en

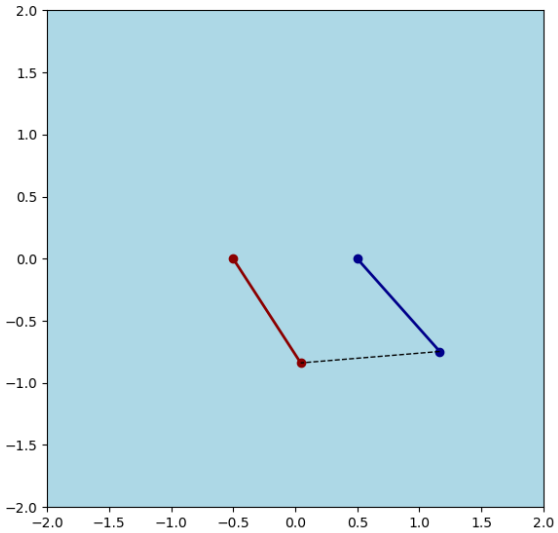


Figura 1.2: Imagen de la animación

Igual podemos graficar el espacio fase

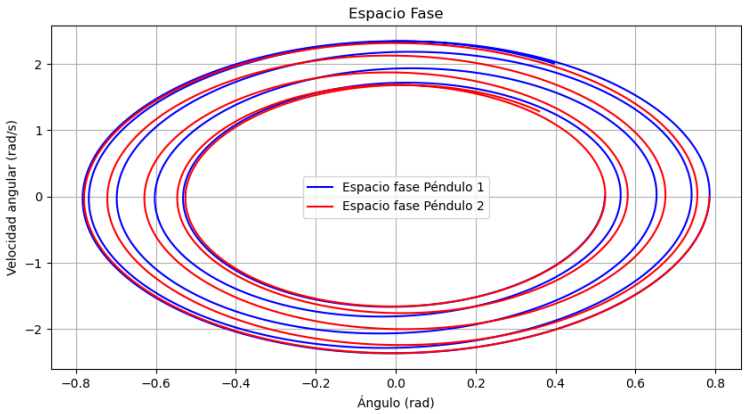


Figura 1.3: Espacio fase del pendulo acopldo

Los códigos se encuentran en el anexo o en el link https://github.com/cinthia-bao/Fisica-Computacional-.git en la carpeta **TAREA 7**.

6

# Péndulo doble

El péndulo doble es un sistema no lineal que consta de dos péndulos unidos en serie, donde el segundo cuelga del extremo del primero. Este sistema exhibe un comportamiento complejo y caótico en ciertos regímenes, siendo un ejemplo paradigmático de la transición de movimientos predecibles a dinámicas caóticas en la mecánica clásica. Su estudio es esencial para comprender sistemas dinámicos y fenómenos caóticos.

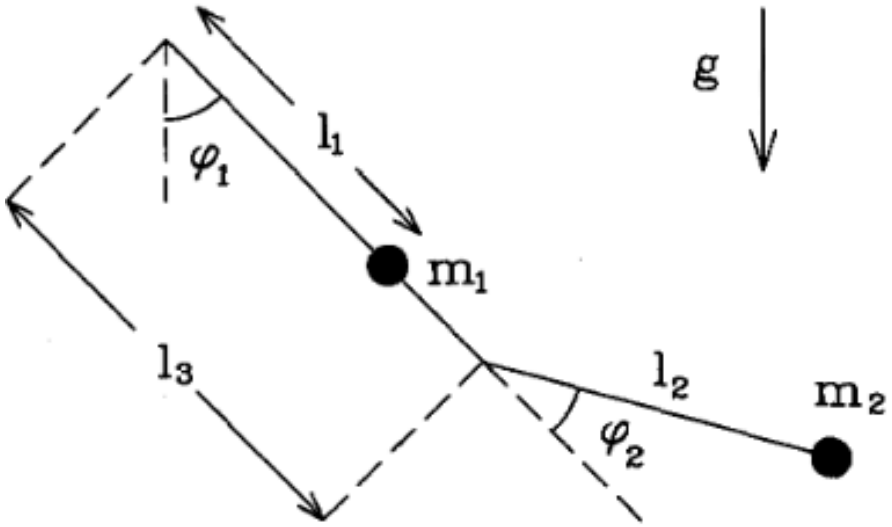


Figura 2.1: Imagen del sistema

## 2.1 Dinámica del Péndulo Doble

Consideremos un péndulo doble compuesto por dos masas $m_1$ y $m_2$, y dos longitudes $l_1$ y $l_2$. Las coordenadas de las masas están definidas como:

$$x_1 = l_1 \sin\theta_1, \quad y_1 = -l_1 \cos\theta_1$$

$$x_2 = x_1 + l_2 \sin\theta_2, \quad y_2 = y_1 - l_2 \cos\theta_2$$

La energía cinética total $T$ incluye las contribuciones de ambas masas:

$$T = \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2)$$

Calculamos las derivadas temporales:

$$\dot{x}_1 = l_1\dot{\theta}_1\cos\theta_1, \quad \dot{y}_1 = l_1\dot{\theta}_1\sin\theta_1$$

$$\dot{x}_2 = \dot{x}_1 + l_2\dot{\theta}_2\cos\theta_2, \quad \dot{y}_2 = \dot{y}_1 + l_2\dot{\theta}_2\sin\theta_2$$

Sustituyendo:

$$T = \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\left[l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right]$$

La energía potencial está dada por las posiciones verticales de las masas:

$$V = -m_1gy_1 - m_2gy_2$$

Sustituyendo $y_1$ y $y_2$:

$$V = -(m_1 + m_2)gl_1\cos\theta_1 - m_2gl_2\cos\theta_2$$

$$V = m_1gl_1\cos\theta_1 + m_2g(l_1\cos\theta_1 + l_2\cos\theta_2)$$

El Lagrangiano se define como:

$$\mathcal{L} = T - V$$

Sustituyendo $T$ y $V$:

$$\mathcal{L} = T - V = = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2l_2^2\dot{\theta}_2^2 + m_2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) + (m_1 + m_2)gl_1\cos\theta_1 + m_2gl_2\cos\theta_2$$

$$-m_1gl_1\cos\theta_1 - m_2g(l_1\cos\theta_1 + l_2\cos\theta_2)$$

Las ecuaciones de Euler-Lagrange son:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_i}\right) - \frac{\partial\mathcal{L}}{\partial\theta_i} = 0, \quad i = 1,2$$

Para $\theta_1$,$\theta_2$:

$$\ddot{\theta}_1 = \frac{g\,(2m_1+m_2)\,\text{sen}\,\theta_1 - m_2 g\,\text{sen}\,(\theta_1-2\theta_2) - 2\,\text{sen}\,(\theta_1-\theta_2)\,m_2\left(\dot{\theta}_2^2 l_2 + \dot{\theta}_1^2 l_1 \cos(\theta_1-\theta_2)\right)}{l_1\,(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}$$

$$\ddot{\theta}_2 = \frac{2\,\text{sen}\,(\theta_1-\theta_2)\left(\dot{\theta}_1^2 l_1\,(m_1+m_2) + g\,(m_1+m_2)\cos\theta_1 + \dot{\theta}_2^2 l_2 m_2 \cos(\theta_1-\theta_2)\right)}{l_2\,(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}$$

Los momentos conjugados son:

$$p_1 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}, \quad p_2 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}$$

$$p_1 = (m_1+m_2)l_1^2\dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_1-\theta_2)$$

$$p_2 = m_2 l_2^2 \dot{\theta}_2 + m_2 l_1 l_2 \dot{\theta}_1 \cos(\theta_1-\theta_2)$$

El Hamiltoniano es:

$$\mathcal{H} = \sum_{i=1}^{2} p_i \dot{\theta}_i - \mathcal{L}$$

Sustituyendo:

$$\mathcal{H} = \frac{1}{2}(m_1+m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) + m_1 g l_1 \cos\theta_1 + m_2 g(l_1\cos\theta_1 + l_2\cos\theta_2)$$

El espacio fase junto con la simulación se puede ver en

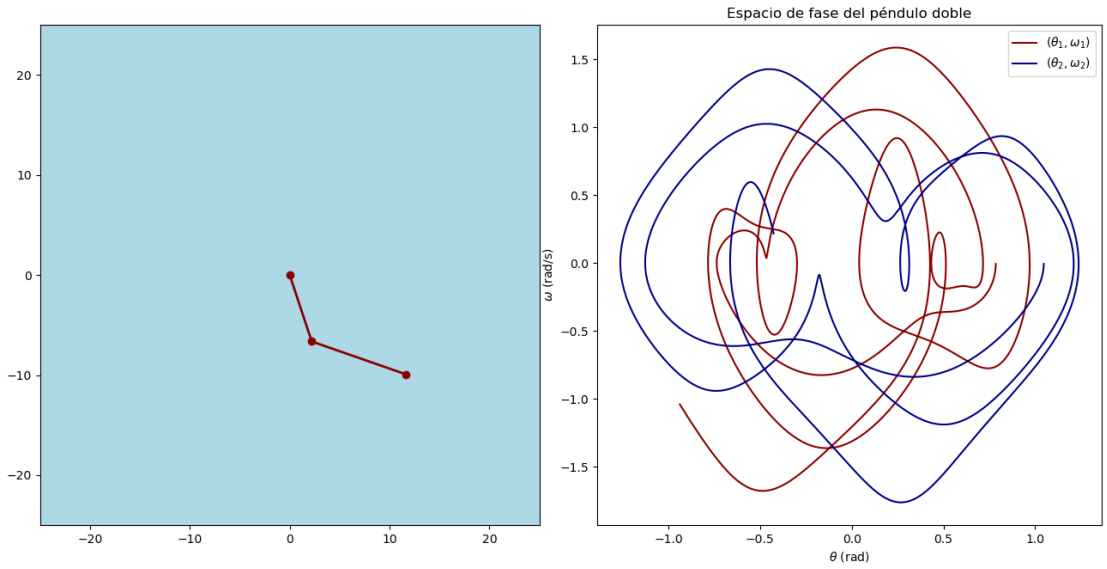


Figura 2.2: Simulación y espacio fase péndulo doble

Los códigos se encuentran en el anexo o en el link https://github.com/cinthia-bao/Fisica-Computacional-.git en la carpeta **TAREA 7**.

# Conclusión

En este trabajo exploré la dinámica de sistemas mecánicos oscilatorios, específicamente el comportamiento de péndulos acoplados por un resorte separados por una distancia

y y el péndulo doble. Utilicé el formalismo de Euler-Lagrange para derivar sus ecuaciones de movimiento y, a partir de ellas, analicé sus soluciones tanto de manera analítica como numérica. Para comprender mejor la evolución temporal de estos sistemas, implementé simulaciones computacionales con el objetivo de visualizar sus trayectorias en el espacio fase, lo que permitió observar la transición entre movimientos regulares y caóticos.

Además, para facilitar la representación visual de los resultados, recurrí a herramientas de inteligencia artificial, como ChatGPT y DeepSeek, que me ayudaron en la implementación de las animaciones. Estas animaciones fueron fundamentales para interpretar la dinámica de los sistemas de una manera más intuitiva y verificar la consistencia de los resultados obtenidos. En general, este trabajo me permitió reforzar la conexión entre la formulación teórica y su aplicación computacional, lo que resultó esencial para un estudio más profundo de estos sistemas mecánicos

# Anexos

Los códigos se encuentran en el link https://github.com/cinthia-bao/Fisica-Computacional-.git en la carpeta **TAREA 7**.

## 4.1 pendulo_doble.py

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parmetros del sistema
l1, l2 = 7, 10 # Longitudes de los pndulos
m1, m2 = 7, 14 # Masas de los pndulos
g = 9.81 # Gravedad
dt = 0.01 # Intervalo de tiempo
t10, t20 = np.pi / 4, np.pi / 3 # ngulos iniciales
w1_, w2_ = 0, 0 # Velocidades angulares iniciales
fin = 20 # Tiempo total de simulacin

# Funcin para calcular las aceleraciones angulares
def accelerations(theta1, theta2, omega1, omega2):
    delta = theta1 - theta2
    denom1 = l1 * (2 * m1 + m2 - m2 * np.cos(2 * delta))
    alpha1 = (
        -g * (2 * m1 + m2) * np.sin(theta1)
        - m2 * g * np.sin(theta1 - 2 * theta2)
        - 2 * m2 * np.sin(delta) * (omega2**2 * l2 + omega1**2 * l1 * np.
            cos(delta))
    ) / denom1
    denom2 = l2 * (2 * m1 + m2 - m2 * np.cos(2 * delta))
```

```
    alpha2 = (
        2
        * np.sin(delta)
        * (
            omega1**2 * l1 * (m1 + m2)
            + g * (m1 + m2) * np.cos(theta1)
            + omega2**2 * l2 * m2 * np.cos(delta)
        )
    ) / denom2
    return alpha1, alpha2

# Funcin para calcular las posiciones
def coord(theta1, theta2):
    x1 = l1 * np.sin(theta1)
    y1 = -l1 * np.cos(theta1)
    x2 = x1 + l2 * np.sin(theta2)
    y2 = y1 - l2 * np.cos(theta2)
    return x1, x2, y1, y2

# Simulacin del movimiento
def simulate(theta1_0, theta2_0, omega1_0, omega2_0, fin, dt):
    steps = int(fin / dt)
    theta1, theta2 = [theta1_0], [theta2_0]
    omega1, omega2 = omega1_0, omega2_0
    x1, x2, y1, y2 = [], [], [], []
    theta1_dot, theta2_dot = [], [] # Guardar las velocidades angulares
    for _ in range(steps):
        x1_i, x2_i, y1_i, y2_i = coord(theta1[-1], theta2[-1])
        x1.append(x1_i)
        x2.append(x2_i)
        y1.append(y1_i)
        y2.append(y2_i)
        alpha1, alpha2 = accelerations(theta1[-1], theta2[-1], omega1,
            omega2)
        omega1 += alpha1 * dt
        omega2 += alpha2 * dt
        theta1.append(theta1[-1] + omega1 * dt)
        theta2.append(theta2[-1] + omega2 * dt)
        theta1_dot.append(omega1)
        theta2_dot.append(omega2)
```

```
    return x1, x2, y1, y2, theta1, theta2, theta1_dot, theta2_dot

# Ejecutar la simulacin
x1, x2, y1, y2, theta1, theta2, theta1_dot, theta2_dot = simulate(t10, t20,
     w1_, w2_, fin, dt)

# Configuracin de la animacin y del grfico
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))

# Subgrfico 1: Animacin del movimiento
ax1.set_xlim(-25, 25)
ax1.set_ylim(-25, 25)
ax1.set_aspect('equal')
ax1.set_facecolor('lightblue')
line, = ax1.plot([], [], 'o-', lw=2, color='darkred')

# Funcin de actualizacin para la animacin
def update(frame):
    line.set_data([0, x1[frame], x2[frame]], [0, y1[frame], y2[frame]])
    return line,

# Subgrfico 2: Espacio de fase
ax2.set_xlabel(r'$\theta$ (rad)')
ax2.set_ylabel(r'$\omega$ (rad/s)')
ax2.plot(theta1[:-1], theta1_dot, label=r'$(\theta_1, \omega_1)$', color='
    darkred')
ax2.plot(theta2[:-1], theta2_dot, label=r'$(\theta_2, \omega_2)$', color='
    darkblue')
ax2.legend()
ax2.set_title('Espacio de fase del pndulo doble')

# Crear la animacin
ani = FuncAnimation(fig, update, frames=len(x1), interval=dt * 1000, blit=
    True)

plt.tight_layout()
plt.show()
```

## 4.2 pendulo_acoplado

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parmetros del sistema
g = 9.81 # Gravedad
l = 1.0 # Longitud de las cuerdas de los pndulos
k = 1.0 # Constante del resorte
m = 1.0 # Masa de cada pndulo
d = 1.0 # Distancia entre los puntos de suspensin de los pndulos
dt = 0.01 # Paso de tiempo
fin = 10 # Tiempo total de simulacin

# Funcin para simular el movimiento de los pndulos
def two_pends_spring_with_distance(theta1_0, theta2_0, omega1_0, omega2_0,
    fin, dt):
    steps = int(fin / dt)
    theta1, theta2 = [theta1_0], [theta2_0]
    omega1, omega2 = [omega1_0], [omega2_0] # Cambiar omega1 y omega2 a
        listas
    times = [0]

    for n in range(steps):
        # Coordenadas de cada pndulo
        x1 = d / 2 + l * np.sin(theta1[-1])
        y1 = -l * np.cos(theta1[-1])
        x2 = -d / 2 + l * np.sin(theta2[-1])
        y2 = -l * np.cos(theta2[-1])

        # Longitud efectiva del resorte
        delta_x = x2 - x1
        delta_y = y2 - y1
        spring_length = np.sqrt(delta_x**2 + delta_y**2)
        spring_force = k * (spring_length - d) / m # Fuerza del resorte

        # Direccin de la fuerza del resorte
        force_x = spring_force * (delta_x / spring_length)
        force_y = spring_force * (delta_y / spring_length)
```

```
        # Ecuaciones del movimiento
        omega1_new = omega1[-1] - ((g / l) * np.sin(theta1[-1]) + (force_x *
            np.cos(theta1[-1]) + force_y * np.sin(theta1[-1])) / m) * dt
        theta1_new = theta1[-1] + omega1_new * dt

        omega2_new = omega2[-1] - ((g / l) * np.sin(theta2[-1]) - (force_x *
            np.cos(theta2[-1]) + force_y * np.sin(theta2[-1])) / m) * dt
        theta2_new = theta2[-1] + omega2_new * dt

        # Actualizar valores
        theta1.append(theta1_new)
        theta2.append(theta2_new)
        omega1.append(omega1_new) # Agregar valores a la lista
        omega2.append(omega2_new) # Agregar valores a la lista
        times.append(n * dt)

    return theta1, theta2, omega1, omega2, times

# Simulacin inicial
theta1, theta2, omega1, omega2, times = two_pends_spring_with_distance(np.
    pi / 6, np.pi / 4, 0, 0, fin, dt)

# Calcular posiciones en 2D de los pndulos
x1 = d / 2 + l * np.sin(theta1)
y1 = -l * np.cos(theta1)
x2 = -d / 2 + l * np.sin(theta2)
y2 = -l * np.cos(theta2)

# Configuracin de la animacin y del grfico
fig, ax = plt.subplots(figsize=(8, 8))

# Lmites del grfico
ax.set_xlim(-2, 2)
ax.set_ylim(-2, 2)
ax.set_aspect('equal')
ax.set_facecolor('lightblue')

# Inicializar elementos de la animacin
line1, = ax.plot([], [], 'o-', lw=2, color='darkblue', label='Pndulo 1')
```

```
line2, = ax.plot([], [], 'o-', lw=2, color='darkred', label='Pndulo 2')
spring_line, = ax.plot([], [], '--', lw=1, color='black', label='Resorte')

# Funcin de actualizacin para la animacin
def update(frame):
    # Actualizar las posiciones del primer pndulo
    line1.set_data([d / 2, x1[frame]], [0, y1[frame]])
    # Actualizar las posiciones del segundo pndulo
    line2.set_data([-d / 2, x2[frame]], [0, y2[frame]])
    # Actualizar la lnea del resorte
    spring_line.set_data([x1[frame], x2[frame]], [y1[frame], y2[frame]])
    return line1, line2, spring_line

# Crear la animacin
ani = FuncAnimation(fig, update, frames=len(x1), interval=dt * 1000, blit=
    True)

# Espacio fase - nueva ventana
fig2, ax2 = plt.subplots(figsize=(10, 5))
ax2.plot(theta1, omega1, label='Espacio fase Pndulo 1', color='blue')
ax2.plot(theta2, omega2, label='Espacio fase Pndulo 2', color='red')
ax2.set_xlabel('ngulo (rad)')
ax2.set_ylabel('Velocidad angular (rad/s)')
ax2.set_title('Espacio Fase')
ax2.legend()
ax2.grid()

# Mostrar la animacin
plt.legend()
plt.show()

# Mostrar tambin el grfico del espacio fase
plt.show()
```