Simulación del Movimiento de un Oscilador Armónico Simple

Autor: Ludmila Gianela Cabrera

Resumen

En este trabajo se estudia el comportamiento de un oscilador armónico simple ideal, representado por un sistema masa-resorte sin fricción ni amortiguamiento. A través de una simulación programada en Python utilizando el entorno Spyder 6, se analiza la evolución temporal de la posición, así como la energía cinética, potencial y mecánica total del sistema, bajo condiciones iniciales definidas por el usuario. Los gráficos generados permiten visualizar e interpretar de forma clara las propiedades cinemáticas y energéticas del Movimiento Armónico Simple (MAS), y muestran una concordancia con las predicciones teóricas. El objetivo es ilustrar el carácter periódico del movimiento y destacar el valor de la programación como herramienta para comprender fenómenos físicos.

Palabras clave

Oscilador armónico, energía, movimiento oscilatorio, resorte.

Introducción

El oscilador armónico simple (MAS) es un sistema fundamental en física, ya que permite modelar una amplia gama de fenómenos, desde el comportamiento de un resorte hasta las vibraciones de átomos y moléculas. Su estudio es esencial porque muchas interacciones físicas complejas, como las fuerzas entre átomos descritas por el potencial de Lennard-Jones, pueden aproximarse mediante este modelo para pequeños desplazamientos. Los sistemas oscilatorios permiten analizar fenómenos como la conservación e intercambio de energía entre sus formas cinética y potencial. Además, al estudiar la superposición de oscilaciones de diferente frecuencia, es posible comprender fenómenos más complejos como los batimientos. En este trabajo se propone simular numéricamente el comportamiento de un MAS, representado por una masa unida a un resorte ideal, utilizando herramientas computacionales como Python. Esta aproximación permite visualizar e interpretar cuantitativamente las propiedades del movimiento oscilatorio,

resaltando el valor de la programación como recurso para explorar fenómenos físicos de forma precisa y accesible.

Desarrollo

Marco Teórico

El Movimiento Armónico Simple (MAS) es un tipo de movimiento periódico presente en sistemas mecánicos donde la fuerza que actúa sobre un objeto es proporcional a su desplazamiento respecto de una posición de equilibrio, y dirigida en sentido opuesto. Un ejemplo clásico es el sistema masa-resorte, en el que una masa *m* está unida a un resorte de constante elástica *k*.

Cuando el resorte no está ni estirado ni comprimido, se dice que el sistema se encuentra en su posición de equilibrio ($x = 0$). Si se desplaza la masa una distancia *x*, el resorte ejerce una fuerza restauradora (*F*) conocida como la Ley de Hooke:

$$F = -kx$$

donde *x* es la posición del bloque, *k* es la constante de elasticidad del resorte (en N/m), y el signo negativo indica que la fuerza siempre apunta hacia el equilibrio.

Aplicando la Segunda Ley de Newton, se obtiene:

$$-kx = ma_x \Rightarrow -\frac{k}{m}x = a_x$$

Esto muestra que la aceleración es proporcional a la posición, pero en sentido opuesto, es decir, la aceleración siempre apunta hacia la posición de equilibrio, en dirección contraria al desplazamiento.

Por lo tanto, un objeto se mueve con movimiento armónico simple siempre que su aceleración sea proporcional a su posición y dirigida en sentido opuesto al desplazamiento desde el equilibrio.

Ecuación diferencial del movimiento armónico simple

Recordando que la aceleración es la derivada de la velocidad con respecto al tiempo, podemos escribir:

$$a = \frac{dv}{dt} \Rightarrow \frac{d^2x}{dt^2} = -\frac{k}{m}\,x$$

Para simplificar la notación y resaltar la forma estándar de las soluciones trigonométricas, se define:

$$\omega^2 = \frac{k}{m}$$

la ecuación queda:

$$\frac{d^2x}{dt^2} = -\omega^2 x$$

Esta ecuación diferencial muestra que la segunda derivada de *x* con respecto al tiempo es proporcional a *x* pero con signo negativo, lo que lleva a soluciones en términos de funciones trigonométricas (seno y coseno).

Movimiento y ciclo

Supongamos que el bloque se desplaza hasta la posición $x = A$ (amplitud máxima). La aceleración en ese punto es:

$$a = -\omega^2 A$$

Cuando el bloque pasa por la posición de equilibrio $x = 0$, la aceleración es cero:

$$a = 0$$

Luego, al desplazarse a la posición $x = -A$, la aceleración cambia de signo:

$$a = \omega^2 A$$

El bloque completa un ciclo cuando regresa a $x = A$, pasando por $x = 0$, punto donde la velocidad es máxima. Entonces, la posición del bloque oscila entre $x = \pm A$.

La solución general de esta ecuación es una función trigonométrica:

$$x(t) = Acos(\omega t + \phi)$$

donde:

- A es la amplitud, valor máximo del desplazamiento respecto al equilibrio.
- ω es la frecuencia angular (en rad/s).
- ϕ es la fase inicial, que depende de las condiciones en $t = 0$.
- $\omega t + \phi$ es la fase del movimiento.

De la derivada y segunda derivada de la posición se puede deducir la velocidad y la aceleración:

$$v(t) = \frac{dv}{dt} = -A\omega sin(\omega t + \phi)$$

$$a(t) = \frac{d^2x}{dt^2} = -A\omega^2 cos(\omega t + \phi)$$

Energía en el Movimiento Armónico Simple

En el MAS ideal (sin rozamiento), la energía mecánica total se conserva y se intercambia entre energía cinética *K* y energía potencial elástica *U*:

- Energía cinética:

$$K(t) = \frac{1}{2}mv^2 \Rightarrow K(t) = \frac{1}{2}mA^2\omega^2 sin^2(\omega t + \phi)$$

- Energía potencial elástica:

$$U(t) = \frac{1}{2}kx^2 \Rightarrow U(t) = \frac{1}{2}kA^2 cos^2(\omega t + \phi)$$

Sumando ambas y simplificando, se obtiene la energía total:

$$E = K + U = \frac{1}{2}kA^2$$

que permanece constante en el tiempo. Esto refleja que la energía del sistema se transforma continuamente entre potencial (cuando $x = \pm A$) y cinética (cuando $x = 0$, con velocidad máxima).

Interpretación física

Durante la oscilación:

- En $x = \pm A$, la masa se detiene momentáneamente ($v = 0$), y la energía es completamente potencial.
- En $x = 0$, la masa alcanza su velocidad máxima, y la energía es completamente cinética.

Este comportamiento es representativo de muchos sistemas físicos reales, como oscilaciones de átomos en moléculas, circuitos eléctricos oscilantes, o péndulos (en el caso de pequeñas amplitudes).

El programa

Para ilustrar estos conceptos, se desarrolló un programa en Python utilizando el entorno de desarrollo Spyder. Este programa permite simular el movimiento de una masa unida a un resorte, graficando su posición en función del tiempo, así como la evolución de la energía cinética, potencial y total del sistema.
Las condiciones iniciales como la amplitud, la masa y la constante del resorte son ingresadas por el usuario, y el programa calcula automáticamente los valores de velocidad, aceleración y energías. Esta herramienta resulta útil para visualizar cómo varían las magnitudes físicas durante el ciclo del movimiento y para comprobar la conservación de la energía en un sistema ideal.

Primero, se importan las librerías necesarias:

- math: funciones matemáticas (aunque no se usa directamente).
- numpy: para cálculos numéricos eficientes y vectores.
- matplotlib.pyplot: para generar gráficos.
- matplotlib.animation: para crear la animación del sistema masa-resorte.

Además, se fuerza el uso del backend "*Qt5Agg*" para asegurar compatibilidad con animaciones dentro de Spyder.

Luego, se imprime un título informativo en consola y se ingresan los datos físicos del sistema masa-resorte:

- A: amplitud del movimiento.
- ω: frecuencia angular $\omega = \sqrt{\frac{k}{m}}$ directamente).
- φ: fase inicial.
- m: masa del bloque.
- k: constante del resorte.
- t_final: tiempo total a simular.

Posteriormente, se definen 500 puntos de tiempo entre 0 y el tiempo final (esto es para que el gráfico sea suave y se vea bien).

A esto le siguen los cálculos, donde se calcula la posición *x(t)* según la fórmula del MAS. Se deriva esa expresión para obtener la velocidad *v(t)*. Se calcula la energía cinética, la energía potencial y la energía mecánica total, que debe ser constante si no hay pérdidas.

Hecho esto, se comienzan a configurar las gráficas que aparecerán al final.

Para la primera, se crea una figura donde se grafica *x(t)* en azul, se ajustan los ejes, el título y la leyenda, se guarda el gráfico como imagen y se muestra en pantalla.

Para la segunda, se grafican las 3 energías del sistema: cinética (amarillo), potencial (verde) y total (rojo). Se ajustan etiquetas y se guarda el gráfico final como imagen.

Además de los gráficos estáticos, el programa incluye una animación que representa visualmente el movimiento del sistema masa-resorte. Esto se logra usando `matplotlib.animation.FuncAnimation`.

Para lograrlo, se utiliza una figura adicional donde se representa un resorte horizontal unido a una masa rectangular que se desplaza en el eje horizontal. Los elementos involucrados en la animación son:

- La función *resorte(x1, x2)*: genera un conjunto de puntos que forman un zigzag entre dos posiciones x_1 y x_2, simulando la forma de un resorte. El número de espiras y la amplitud del zigzag están definidos dentro de la función.
- *spring_line*: es la línea azul oscuro que representa gráficamente al resorte. Inicialmente se crea vacía, pero se actualiza en cada instante de tiempo para dibujar el resorte desde el origen hasta la posición de la masa.
- *mass_rect*: es un rectángulo rojo que representa la masa unida al resorte. Se mueve horizontalmente a lo largo del eje x, siguiendo la posición *x(t)* calculada anteriormente.
- La función *inicio()*: deja el gráfico en blanco antes de comenzar la animación. Esta función reinicia la línea del resorte y coloca la masa en la posición inicial. Es requerida por *FuncAnimation* cuando se utiliza la opción *blit = True*.
- La función *animacion(i)*: se ejecuta en cada cuadro (frame) de la animación. Toma la posición *x[i]*, actualiza la forma del resorte hasta ese punto, y reposiciona la masa en la nueva coordenada horizontal correspondiente al instante de tiempo actual.
- *FuncAnimation(...)*: Es una clase incluida en el módulo *matplotlib.animation.* Esta clase coordina todo el proceso de animación. Al invocarla, se crea un objeto de animación que controla qué figura se va a animar, qué función se ejecuta en cada cuadro (*animacion*), cómo se inicializa el gráfico (*inicio*), cuántos cuadros tendrá (*frames = N*), cada cuánto se actualiza (*interval* = ...) y si la animación debe repetirse (*repeat = True*). Finalmente, *plt.show()* se ejecuta y muestra la animación. Este componente visual ayuda a comprender intuitivamente cómo varían la posición y la forma del resorte en función del tiempo, reforzando la interpretación de los conceptos del movimiento armónico simple.

Referencias

- Serway, Raymond A. y Jewett, John W. Jr., Física para Ciencias e Ingeniería, Vol. 1, CENGAGE.. Capitulo 15, secciones 15.1 a 15.3.
- Documentación oficial de Python y Matplotlib.

Anexos

1) Capturas del código desde Spyder:

```python
# Importamos los módulos necesarios

import matplotlib
matplotlib.use("Qt5Agg")  # Forzar backend compatible con animaciones en Spyder

from math import *
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# ------------------------------------------------------
# ENTRADA DE DATOS FÍSICOS PARA EL OSCILADOR ARMÓNICO
# ------------------------------------------------------

print("SIMULADOR DE OSCILADOR ARMÓNICO SIMPLE")
print("--------------------------------------")

A = float(input("Ingrese la amplitud A (en metros): "))
omega = float(input("Ingrese la frecuencia angular ω (en rad/s): "))
phi = float(input("Ingrese la fase inicial φ (en radianes): "))
m = float(input("Ingrese la masa (en kilogramos): "))
k = float(input("Ingrese la constante elástica k (en N/m): "))
t_final = float(input("Ingrese el tiempo total de simulación (en segundos): "))

```

```python
# ------------------------------------------------------
# PARÁMETROS NUMÉRICOS (DEFINIDOS EN EL CÓDIGO)
# ------------------------------------------------------

N = 500                          # Cantidad de puntos de simulación
t = np.linspace(0, t_final, N)  # Vector de tiempos

# ------------------------------------------------------
# CÁLCULO DEL MOVIMIENTO Y ENERGÍAS
# ------------------------------------------------------

x = A * np.cos(omega * t + phi)              # Posición
v = -A * omega * np.sin(omega * t + phi)     # Velocidad

K = 0.5 * m * v**2                           # Energía cinética
U = 0.5 * k * x**2                           # Energía potencial
E = K + U                                    # Energía total

# ------------------------------------------------------
# GRÁFICO 1: POSICIÓN VS TIEMPO
# ------------------------------------------------------

fig1, ax1 = plt.subplots()
ax1.plot(t, x, label = "x(t)", color = 'blue')
ax1.set_xlabel("Tiempo (s)", fontsize = 14)
ax1.set_ylabel("Posición (m)", fontsize = 14)
ax1.set_title("Posición en el oscilador armónico simple")
ax1.legend(loc = "best")
fig1.savefig("MASposición.png")
plt.show()

```

```
63  # ---------------------------------------------------------
64  # GRÁFICO 2: ENERGÍAS VS TIEMPO
65  # ---------------------------------------------------------
66
67  fig2, ax2 = plt.subplots()
68  ax2.plot(t, K, label = "Energía cinética", color = 'gold')
69  ax2.plot(t, U, label = "Energía potencial", color = 'green')
70  ax2.plot(t, E, label = "Energía total", color = 'crimson')
71  ax2.set_xlabel("Tiempo (s)", fontsize = 14)
72  ax2.set_ylabel("Energía (J)", fontsize = 14)
73  ax2.set_title("Energías del sistema oscilante")
74  ax2.legend(loc = "best")
75  fig2.savefig("MASenergías.png")
76  plt.show()
77
```

```
78  # ---------------------------------------------------------
79  # ANIMACIÓN DE OSCILADOR ARMÓNICO SIMPLE (RESORTE)
80  # ---------------------------------------------------------
81
82  # Preparar figura
83
84  fig3, ax3 = plt.subplots()
85  ax3.set_xlim(-A * 1.5, A * 1.5)
86  ax3.set_ylim(-0.5, 1.5)
87  ax3.set_aspect('equal')
88  ax3.set_title("Masa unida a un resorte")
89
90  # Función que genera los puntos x e y en zigzag entre dos posiciones para crear forma de resorte
91  def resorte(x1, x2, n_espiras = 10, amplitud = 0.2):
92      espiras_x = np.linspace(x1, x2, 2 * n_espiras + 2)  # Puntos a lo largo del eje x
93      espiras_y = np.zeros_like(espiras_x)                # Puntos a lo largo del eje y
94      espiras_y[1:-1:2] = amplitud                        # Alterna + y - para formar el zigzag
95      espiras_y[2:-1:2] = -amplitud
96      return espiras_x, espiras_y
97
98  # Elementos gráficos
99  spring_line, = ax3.plot([], [], lw = 1, color = 'navy')
100 mass_rect = plt.Rectangle((0, 0), 0.2, 0.2, color = 'crimson')
101 ax3.add_patch(mass_rect)
102
```

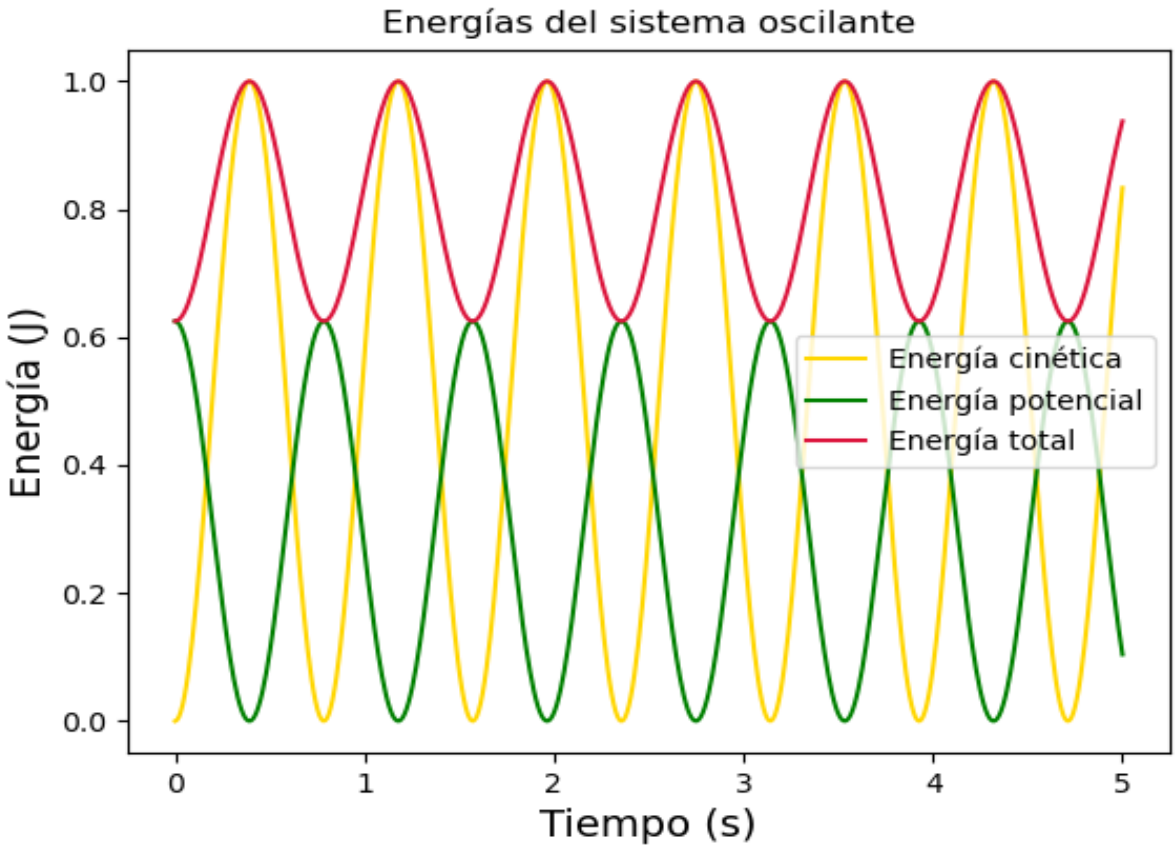
```
103 # Función que deja el gráfico en blanco antes de que empiece la animación
104 def inicio():
105     spring_line.set_data([], [])
106     mass_rect.set_xy((0, 0))
107     return spring_line, mass_rect
108
109 # Limpia los elementos gráficos antes de comenzar la animación
110 def animacion(i):
111     x_posicion = x[i]
112     # Resorte
113     xs, ys = resorte(0, x_posicion)            # genera el resorte hasta la posición actual
114     spring_line.set_data(xs, ys)
115     # Masa
116     mass_rect.set_xy((x_posicion - 0.1, -0.1))  # centra la masa en x_posicion
117     return spring_line, mass_rect
118
119 animacion_resorte = FuncAnimation(fig3, animacion, frames = N, init_func = inicio,
120                     blit = True, interval = 1000 * t_final / N, repeat = True)
121
122 plt.show()
123
```

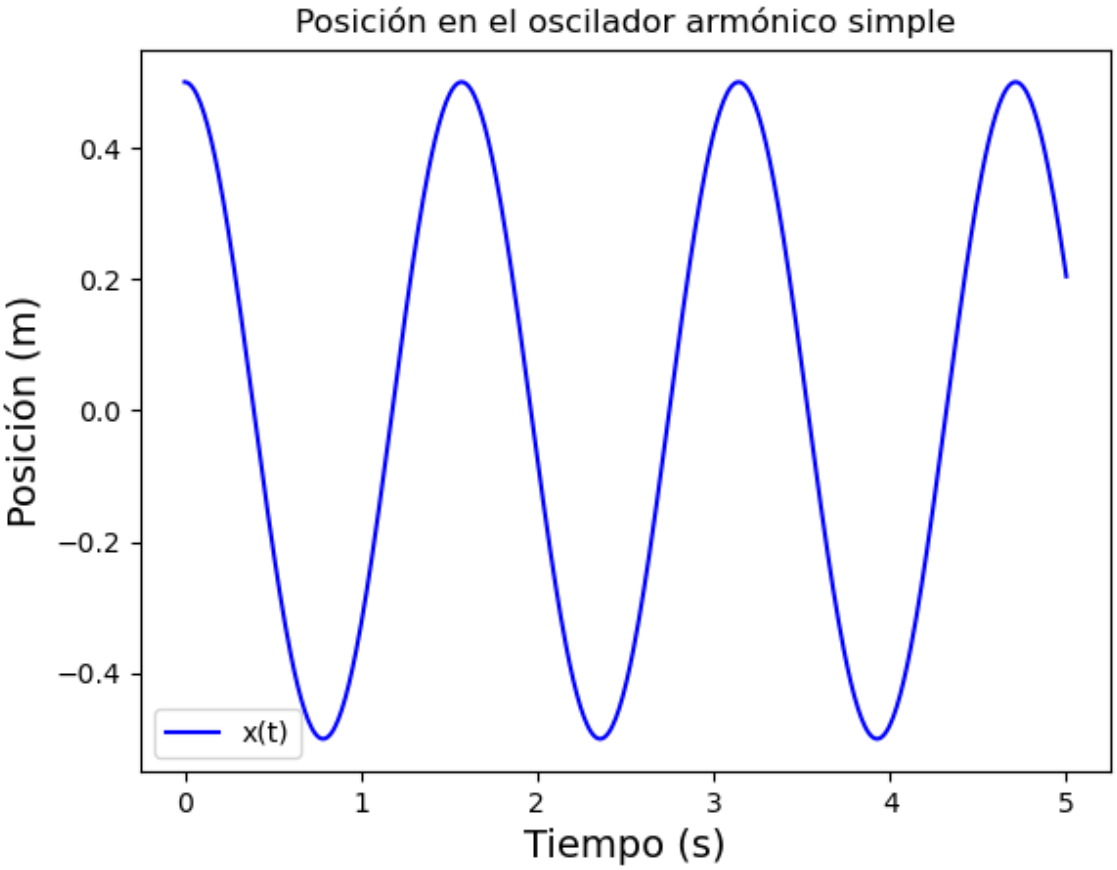
2) Captura de la terminal en la que se ingresan los valores para que el código genere las gráficas. Se eligieron los siguientes valores:
    - Para la Amplitud: 0,5 m (50cm)
    - Para la frecuencia angular: 4 rad/s

- Para la fase inicial: 0 rad
- Para la masa: 0,5 kg
- Para la constante del resorte: 5 N/m
- Para el tiempo: 5 segundos

```
SIMULADOR DE OSCILADOR ARMÓNICO SIMPLE
----------------------------------------
Ingrese la amplitud A (en metros): 0.5
Ingrese la frecuencia angular ω (en rad/s): 4
Ingrese la fase inicial ϕ (en radianes): 0
Ingrese la masa (en kilogramos): 0.5
Ingrese la constante elástica k (en N/m): 5
Ingrese el tiempo total de simulación (en segundos): 5
```

3) Gráficas generadas:





4) Captura de un frame de la animación del sistema masa-resorte: