Simulación de Vibraciones en Sistemas de un Grado de Libertad usando Python

Alan Sordo

Abstract

En este trabajo se presenta una metodología para la simulación de la respuesta dinámica de estructuras simples, modeladas como sistemas de un grado de libertad (1GDL). Utilizando el lenguaje de programación Python, se implementan las ecuaciones diferenciales que describen la vibración libre y forzada de estos sistemas. Se proporcionan fundamentos teóricos y matemáticos, así como un ejemplo práctico de simulación. Esta metodología es útil para ingenieros y estudiantes que buscan entender la dinámica estructural mediante herramientas computacionales.

1 Introducción

La dinámica estructural es una disciplina fundamental en la ingeniería civil que estudia el comportamiento de las estructuras sometidas a cargas dinámicas, como las producidas por sismos, viento, o máquinas en funcionamiento. Las estructuras simples, como marcos de un solo nivel o tanques elevados, pueden ser idealizadas como sistemas de un grado de libertad (1GDL), lo cual simplifica su análisis sin perder la generalidad del comportamiento dinámico esencial. En este trabajo, se utiliza Python para simular la respuesta dinámica de estos sistemas, proporcionando una herramienta práctica y accesible para el análisis estructural.

2 Fundamentos Teóricos

La dinámica de un sistema de un grado de libertad puede ser descrita mediante una ecuación diferencial de segundo orden:

$$m\ddot{u}(t) + c\dot{u}(t) + ku(t) = F(t) \tag{1}$$

donde m es la masa, c es el coeficiente de amortiguamiento, k es la rigidez, $u(t)$ es el desplazamiento en función del tiempo, y $F(t)$ es la fuerza externa aplicada.

2.1 Vibración Libre

Para el caso de vibración libre (sin fuerza externa), la ecuación se simplifica a:

$$m\ddot{u}(t) + c\dot{u}(t) + ku(t) = 0 \tag{2}$$

La solución de esta ecuación depende del coeficiente de amortiguamiento c:

- **Sin amortiguamiento ($c = 0$)**: La solución es una oscilación armónica simple:

$$u(t) = u_0 \cos(\omega_n t) + \frac{\dot{u}_0}{\omega_n} \sin(\omega_n t) \tag{3}$$

  donde $\omega_n = \sqrt{\frac{k}{m}}$ es la frecuencia natural del sistema.

- **Con amortiguamiento ($c > 0$)**: La solución puede ser subamortiguada, críticamente amortiguada o sobreamortiguada dependiendo del valor del coeficiente de amortiguamiento relativo $\zeta = \frac{c}{2\sqrt{km}}$.

2.2 Vibración Forzada

Para el caso de vibración forzada, se incluye una fuerza externa $F(t)$:

$$m\ddot{u}(t) + c\dot{u}(t) + ku(t) = F(t) \tag{4}$$

Si la fuerza externa es una excitación armónica $F(t) = F_0 \sin(\omega t)$, la solución puede ser obtenida utilizando métodos de la respuesta en frecuencia.

3 Propuesta del Problema

El objetivo es desarrollar un programa en Python que simule la respuesta dinámica de una estructura simple (sistema de un grado de libertad) tanto para vibraciones libres como forzadas. El programa resolverá las ecuaciones diferenciales utilizando métodos numéricos y visualizará los resultados.

3.1 Implementación en Python

3.1.1 Importación de Bibliotecas Necesarias

Librerias

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

3.1.2 Definición de Parámetros y Funciones

Parámetros del sistema

```
m = 1.0  # masa (kg)
k = 100.0  # rigidez (N/m)
c = 2.0  # coeficiente de amortiguamiento (N s/m)
omega_n = np.sqrt(k / m)  # frecuencia natural (rad/s)

def F(t):
    return 10 * np.sin(2 * omega_n * t)

# Ecuaciones de movimiento
def equations(t, y):
    u, v = y
    du_dt = v
    dv_dt = (F(t) - c * v - k * u) / m
    return [du_dt, dv_dt]

# Condiciones iniciales
u0 = [0.1, 0.0]
```

3.1.3 Solución Numérica

Intervalo de tiempo para la simulación

```
t_span = (0, 10)
t_eval = np.linspace(*t_span, 1000)

solution = solve_ivp(equations, t_span, u0, t_eval=t_eval)

t = solution.t
u = solution.y[0]
v = solution.y[1]
```

3.1.4 Visualización de Resultados

Gráfica de desplazamiento

```
plt.figure(figsize=(10, 6))
plt.plot(t, u, label='Desplazamiento (u)')
plt.plot(t, v, label='Velocidad (v)')
plt.xlabel('Tiempo (s)')
plt.ylabel('Respuesta')
plt.title('Respuesta Din mica del Sistema de 1 GDL')
plt.legend()
plt.grid(True)
plt.show()
```

4 Conclusiones

La implementación de un modelo de dinámica estructural en Python permite simular y analizar la respuesta de sistemas simples de un grado de libertad bajo excitaciones diversas. Esta metodología proporciona una herramienta práctica y accesible para ingenieros y estudiantes, facilitando la comprensión y aplicación de los principios de la dinámica estructural.

5 Bibliografía

- Chopra, A. K. (2012). *Dinámica de estructuras* (4a ed.). Pearson Educación.
- Clough, R. W., & Penzien, J. (2003). *Dynamics of Structures* (3rd ed.). McGraw-Hill.