HW4

December 1, 2025

1 HOMEWORK4

Consider a single-degree-of-freedom spring-mass-damper system with mass $m = 3kg$, undamped natural circular frequency $\omega_n = 16rad/s$ and damping ratio $\zeta = 0.06$. The system is subjected to an excitation force $f(t)$ that consists of a sequence of two half-sine pulses, as shown in the figure below.

The first pulse has a height magnitude of 5 N and a frequency $\omega_1 = 13rad/s$, and starts from zero at time $t = 0$ seconds. The second pulse has a height magnitude of 11 N and a frequency of $\omega_2 = 20rad/s$. The second pulse starts from zero at $t = T1$, where $T1 = \frac{2\pi}{\omega_1}$ is the period of a full oscillation at frequency ω_1.

Convolutional integral:

$$x(t) = \int_0^t f(\tau)g(t-\tau)d\tau$$

Where $f(t)$ is excitation. From equation 4.13, impulse response is:

$$g(t) = \frac{1}{m\omega_d}e^{-\zeta\omega_n t}sin\omega_d t$$

[87]:
```
#from numpy import sqrt, exp
import sympy
from sympy import symbols, exp, sin, sqrt, integrate
from numpy import vectorize, linspace, pi
import matplotlib.pyplot as plt

# system parameters
m = 3
w_n = 16
zeta = 0.06
w_d = w_n * sqrt(1- zeta**2)
t, tau = symbols('t tau', real=True, positive=True)
g = 1/(m*w_d) * exp(-zeta*w_n*t)*sin(w_d*t)
g_tau = (1 / (m * w_d)) * exp(-zeta * w_n * (t - tau)) * sin(w_d * (t - tau))
```

[88]:
```
F_1 = 5
F_2 = 11
w_1 = 13
w_2 = 20
```

```
T_1 = 2*pi/w_1
T_2 = 2*pi/w_2
T = [0.0, 5 * T_1]
t_vals = linspace(0, 5*T_1, 2000)
```

[89]:
```
#impluse excitation
f_1 = -F_1 * sin(w_1*tau)
f_2 = F_2 * sin(w_2 * (tau-T_1))

def f_t(t):
    if 0 <= t <= T_1/2:
        return -F_1 * sin(w_1 * t)
    elif T_1 <= t <= T_1 + T_2/2:
        return F_2 * sin(w_2 * (t-T_1))
    else:
        return 0

f_vec = vectorize(f_t)

def solve_convolution(f_func, lower_limit, upper_limit):
    integrand = f_func * g_tau
    result = integrate(integrand, (tau, lower_limit, upper_limit))
    return result

x_1 = solve_convolution(f_1, 0, t) # 0 < t < T_1/2
x_2 = solve_convolution(f_1, 0, T_1/2) # T_1/2 < t < T_1
x_3 = solve_convolution(f_2, T_1, t) # T_1 < t < T_1 + T_2/2
x_4 = solve_convolution(f_2, T_1, T_1 + T_2/2) #T_1 + T_2/2 < t < T_2
x_total = sympy.Piecewise(
    (x_1, (t >= 0) & (t <= T_1/2)),                    # pulse 1
    (x_2, (t > T_1/2) & (t < T_1)),                    # free vibration
    (x_2 + x_3, (t >= T_1) & (t <= T_1 + T_2/2)),      # pulse 2
    (x_2 + x_4, True)                                  # final free vibration
)
```

[90]:
```
# Force plot
plt.figure(figsize=(10,4))
plt.plot(t_vals, f_vec(t_vals))
plt.title("Excitation Force f(t)")
plt.xlabel("Time (s)")
plt.ylabel("Force (N)")
plt.grid()

# Response plot
x_total_func = sympy.lambdify(t, x_total, 'numpy')
x_vals = x_total_func(t_vals) #     t_vals
```

```
plt.figure(figsize=(10,4))
plt.plot(t_vals, x_vals) #   t_vals   x_vals
plt.title("System Response x(t)")
plt.xlabel("Time (s)")
plt.ylabel("Displacement (m)")
plt.grid()

plt.show()
```