Assignment 2, Dynamics of structures

In [ ]:
```
# Import the basic packages for the notebook

import numpy as np # Numerical library
import pandas as pd # Data analysis library
import csv as csv#csv reader etc.
import matplotlib.pyplot as plt # Plotting library
import matplotlib.ticker as maticker
import matplotlib.colors as mcolors # Colours
from sympy.solvers import solve
from sympy import Symbol, Eq, nsolve # For solving equations
from pathlib import Path
```

Exercise 1

Write a programming script in the programming language of your choice to reproduce the response spectra of given earthquake series for 5% and 8% damping ratio. The range of periods should be from 0 to 3 seconds. Depending on the algorithm you choose state if any stability conditions shall be respected. Two ground motions are given to you:

1. Canoga Park earthquake record (from 1994 Northridge Earthquake in California)
2. Iwanuma earthquake record (from 2011 Tohoku earthquake in Japan)

In [ ]:
```
Canoga_Park = np.genfromtxt('Canoga_Park.csv', delimiter=',', skip_header=1)

g = 9810  # gravity, [mm/sec^2]
m = 1     # mass, [kN*sec^2/mm]
dt = 0.01  # time step [sec]
Tmax = 3  # max period [sec]
damping_ratio = [0.05, 0.08]
Periods = np.arange(0.05, Tmax + 0.01, 0.01)

# We have to verify that dt/Tn<1/pi

# Canoga Park damping 5%
p0 = -m * Canoga_Park[0, 1] * g
u0 = 0  # initial displacement (t=0)
v0 = 0  # initial velocity (t=0)

D_damp1 = np.zeros_like(Periods)
V_damp1 = np.zeros_like(Periods)
A_damp1 = np.zeros_like(Periods)

for t in range(len(Periods)):
    Tn = Periods[t]  # natural period of vibration, [sec]
    wn = 2 * np.pi / Tn  # circular frequency [rad/sec]
    k = (2 * np.pi / Tn) ** 2 * m  # stiffness [kN/mm]

    c_damp1 = damping_ratio[0] * 2 * m * wn

    # initial conditions
    p0 = -m * Canoga_Park[0, 1] * g
    a_damp1 = (p0 - c_damp1 * v0 - k * u0) / m
    u_1_damp1 = u0 - dt * v0 + (dt) ** 2 * a_damp1 / 2
    k_h_damp1 = m / (dt ** 2) + c_damp1 / (2 * dt)
```

```
    alpha_damp1 = m / (dt ** 2) - c_damp1 / (2 * dt)
    beta_damp1 = k - 2 * m / (dt ** 2)

    u_damp1 = np.zeros_like(Canoga_Park[:, 0])
    v_damp1 = np.zeros_like(Canoga_Park[:, 0])
    a_damp1 = np.zeros_like(Canoga_Park[:, 0])
    t_damp1 = np.zeros_like(Canoga_Park[:, 0])

    for j in range(len(Canoga_Park)):
        T = j * dt
        p0 = -m * Canoga_Park[j, 1] * g

        p_hut = p0 - alpha_damp1 * u_1_damp1 - beta_damp1 * u0

        u_damp1[j] = p_hut / k_h_damp1

        a_damp1[j] = (u_damp1[j] - 2 * u0 + u_1_damp1) / (dt ** 2)
        v_damp1[j] = (u_damp1[j] - u_1_damp1) / (2 * dt)
        t_damp1[j] = dt * j

        u_1_damp1 = u0
        u0 = u_damp1[j]

    D_damp1[t] = np.max(np.abs(u_damp1))  # peak displacement [mm]
    V_damp1[t] = wn * np.max(np.abs(u_damp1))  # peak velocity [mm/sec]
    A_damp1[t] = wn ** 2 * np.max(np.abs(u_damp1)) / g  # peak acceleration [g]
```

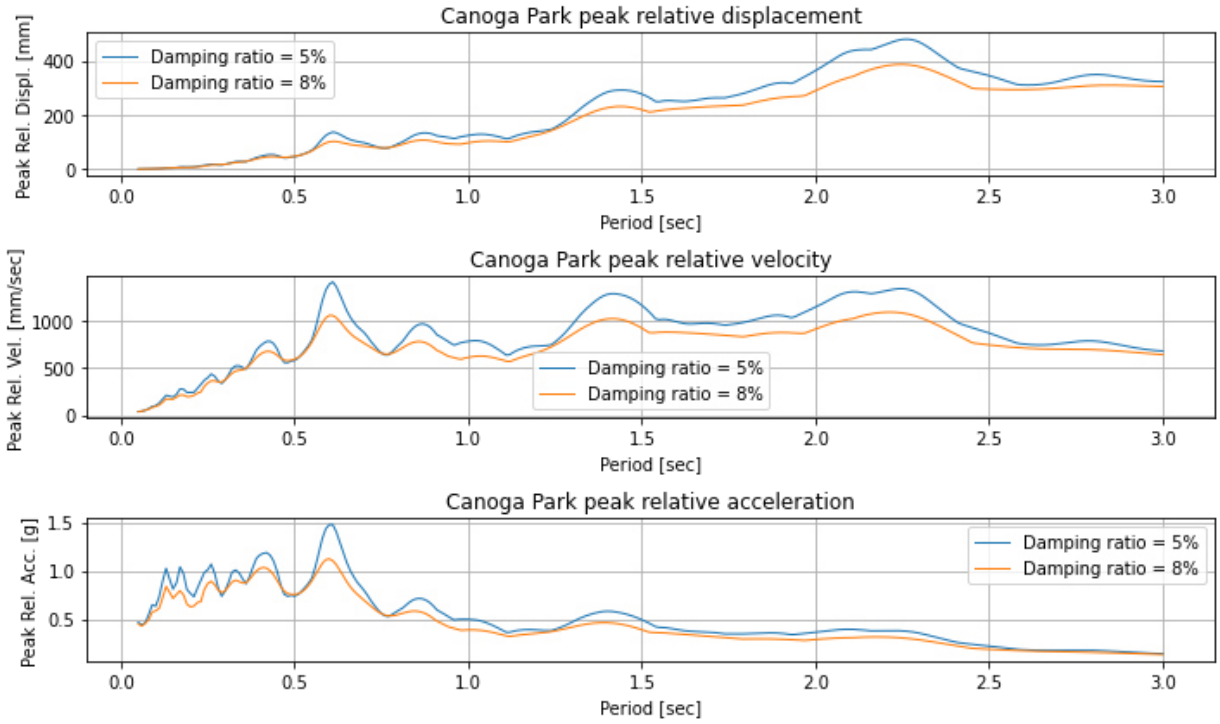
In [ ]:

```
# Canoga Park damping 8%
p0 = -m * Canoga_Park[0, 1] * g
u0 = 0  # initial displacement (t=0)
v0 = 0  # initial velocity (t=0)

D_damp2 = np.zeros_like(Periods)
V_damp2 = np.zeros_like(Periods)
A_damp2 = np.zeros_like(Periods)

for t in range(len(Periods)):
    Tn = Periods[t]  # natural period of vibration, [sec]
    wn = 2 * np.pi / Tn  # circular frequency [rad/sec]
    k = (2 * np.pi / Tn) ** 2 * m  # stiffness [kN/mm]

    c_damp2 = damping_ratio[1] * 2 * m * wn

    # initial conditions
    a_damp2 = (p0 - c_damp2 * v0 - k * u0) / m
    u_1_damp2 = u0 - dt * v0 + (dt) ** 2 * a_damp2 / 2
    k_h_damp2 = m / (dt ** 2) + c_damp2 / (2 * dt)
    alpha_damp2 = m / (dt ** 2) - c_damp2 / (2 * dt)
    beta_damp2 = k - 2 * m / (dt ** 2)

    u_damp2 = np.zeros_like(Canoga_Park[:, 0])
    v_damp2 = np.zeros_like(Canoga_Park[:, 0])
    a_damp2 = np.zeros_like(Canoga_Park[:, 0])
    t_damp2 = np.zeros_like(Canoga_Park[:, 0])

    for j in range(len(Canoga_Park)):
        T = j * dt
        p0 = -m * Canoga_Park[j, 1] * g

        p_hut = p0 - alpha_damp2 * u_1_damp2 - beta_damp2 * u0
```

```
        u_damp2[j] = p_hut / k_h_damp2

        a_damp2[j] = (u_damp2[j] - 2 * u0 + u_1_damp2) / (dt ** 2)
        v_damp2[j] = (u_damp2[j] - u_1_damp2) / (2 * dt)
        t_damp2[j] = dt * j

        u_1_damp2 = u0
        u0 = u_damp2[j]

    D_damp2[t] = np.max(np.abs(u_damp2))  # peak displacement [mm]
    V_damp2[t] = wn * np.max(np.abs(u_damp2))  # peak velocity [mm/sec]
    A_damp2[t] = wn ** 2 * np.max(np.abs(u_damp2)) / g  # peak acceleration [g]
```

In [4]:

```
plt.figure(figsize=(10, 6))

plt.subplot(3, 1, 1)
plt.plot(Periods, D_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, D_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Displ. [mm]')
plt.legend(loc='best')
plt.title('Canoga Park peak relative displacement')
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(Periods, V_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, V_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Vel. [mm/sec]')
plt.legend(loc='best')
plt.title('Canoga Park peak relative velocity')
plt.grid(True)

plt.subplot(3, 1, 3)
plt.plot(Periods, A_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, A_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Acc. [g]')
plt.legend(loc='best')
plt.title('Canoga Park peak relative acceleration')
plt.grid(True)

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



In [8]:

```
Iwanuma = np.genfromtxt('Iwanuma100X.csv', delimiter=',', skip_header=1)

g = 9810  # gravity, [mm/sec^2]
m = 1     # mass, [kN*sec^2/mm]
dt = 0.01  # time step [sec]
Tmax = 3  # max period [sec]
damping_ratio = [0.05, 0.08]
Periods = np.arange(0.05, Tmax + 0.01, 0.01)

# We have to verify that dt/Tn<1/pi

# Canoga Park damping 5%
p0 = -m * Iwanuma[0, 1] * g
u0 = 0  # initial displacement (t=0)
v0 = 0  # initial velocity (t=0)

D_damp1 = np.zeros_like(Periods)
V_damp1 = np.zeros_like(Periods)
A_damp1 = np.zeros_like(Periods)

for t in range(len(Periods)):
    Tn = Periods[t]  # natural period of vibration, [sec]
    wn = 2 * np.pi / Tn  # circular frequency [rad/sec]
    k = (2 * np.pi / Tn) ** 2 * m  # stiffness [kN/mm]

    c_damp1 = damping_ratio[0] * 2 * m * wn

    # initial conditions
    p0 = -m * Iwanuma[0, 1] * g
    a_damp1 = (p0 - c_damp1 * v0 - k * u0) / m
    u_1_damp1 = u0 - dt * v0 + (dt) ** 2 * a_damp1 / 2
    k_h_damp1 = m / (dt ** 2) + c_damp1 / (2 * dt)
    alpha_damp1 = m / (dt ** 2) - c_damp1 / (2 * dt)
    beta_damp1 = k - 2 * m / (dt ** 2)

    u_damp1 = np.zeros_like(Iwanuma[:, 0])
    v_damp1 = np.zeros_like(Iwanuma[:, 0])
    a_damp1 = np.zeros_like(Iwanuma[:, 0])
    t_damp1 = np.zeros_like(Iwanuma[:, 0])
```

```
    for j in range(len(Iwanuma)):
        T = j * dt
        p0 = -m * Iwanuma[j, 1] * g

        p_hut = p0 - alpha_damp1 * u_1_damp1 - beta_damp1 * u0

        u_damp1[j] = p_hut / k_h_damp1

        a_damp1[j] = (u_damp1[j] - 2 * u0 + u_1_damp1) / (dt ** 2)
        v_damp1[j] = (u_damp1[j] - u_1_damp1) / (2 * dt)
        t_damp1[j] = dt * j

        u_1_damp1 = u0
        u0 = u_damp1[j]

    D_damp1[t] = np.max(np.abs(u_damp1))  # peak displacement [mm]
    V_damp1[t] = wn * np.max(np.abs(u_damp1))  # peak velocity [mm/sec]
    A_damp1[t] = wn ** 2 * np.max(np.abs(u_damp1)) / g  # peak acceleration [g]
```

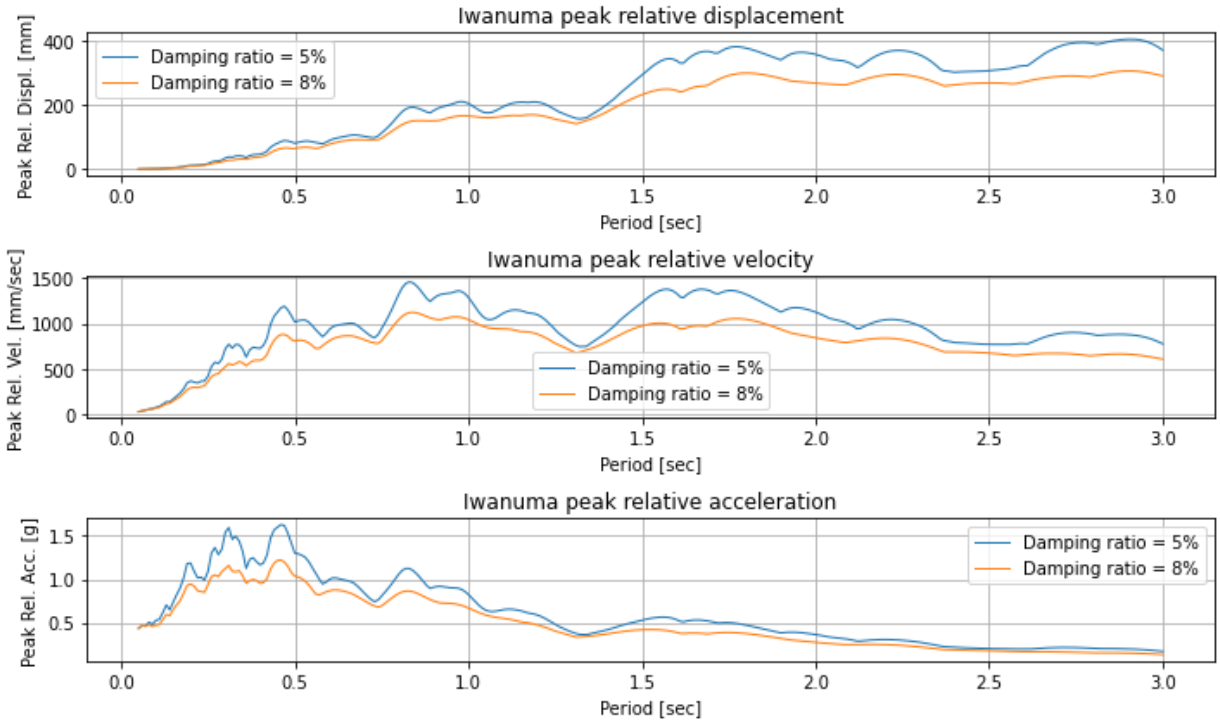
In [11]:

```
p0 = -m * Iwanuma[0, 1] * g
u0 = 0  # initial displacement (t=0)
v0 = 0  # initial velocity (t=0)

D_damp2 = np.zeros_like(Periods)
V_damp2 = np.zeros_like(Periods)
A_damp2 = np.zeros_like(Periods)

for t in range(len(Periods)):
    Tn = Periods[t]  # natural period of vibration, [sec]
    wn = 2 * np.pi / Tn  # circular frequency [rad/sec]
    k = (2 * np.pi / Tn) ** 2 * m  # stiffness [kN/mm]

    c_damp2 = damping_ratio[1] * 2 * m * wn

    # initial conditions
    a_damp2 = (p0 - c_damp2 * v0 - k * u0) / m
    u_1_damp2 = u0 - dt * v0 + (dt) ** 2 * a_damp2 / 2
    k_h_damp2 = m / (dt ** 2) + c_damp2 / (2 * dt)
    alpha_damp2 = m / (dt ** 2) - c_damp2 / (2 * dt)
    beta_damp2 = k - 2 * m / (dt ** 2)

    u_damp2 = np.zeros_like(Iwanuma[:, 0])
    v_damp2 = np.zeros_like(Iwanuma[:, 0])
    a_damp2 = np.zeros_like(Iwanuma[:, 0])
    t_damp2 = np.zeros_like(Iwanuma[:, 0])

    for j in range(len(Iwanuma)):
        T = j * dt
        p0 = -m * Iwanuma[j, 1] * g

        p_hut = p0 - alpha_damp2 * u_1_damp2 - beta_damp2 * u0

        u_damp2[j] = p_hut / k_h_damp2

        a_damp2[j] = (u_damp2[j] - 2 * u0 + u_1_damp2) / (dt ** 2)
        v_damp2[j] = (u_damp2[j] - u_1_damp2) / (2 * dt)
        t_damp2[j] = dt * j

        u_1_damp2 = u0
        u0 = u_damp2[j]

    D_damp2[t] = np.max(np.abs(u_damp2))  # peak displacement [mm]
```

```
    V_damp2[t] = wn * np.max(np.abs(u_damp2))  # peak velocity [mm/sec]
    A_damp2[t] = wn ** 2 * np.max(np.abs(u_damp2))/g   # peak acceleration [g]
```

In [ ]:

```
plt.figure(figsize=(10, 6))

plt.subplot(3, 1, 1)
plt.plot(Periods, D_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, D_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Displ. [mm]')
plt.legend(loc='best')
plt.title('Iwanuma peak relative displacement')
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(Periods, V_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, V_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Vel. [mm/sec]')
plt.legend(loc='best')
plt.title('Iwanuma peak relative velocity')
plt.grid(True)



plt.subplot(3, 1, 3)
plt.plot(Periods, A_damp1, label='Damping ratio = 5%', linewidth=1)
plt.plot(Periods, A_damp2, label='Damping ratio = 8%', linewidth=1)
plt.xlabel('Period [sec]')
plt.ylabel('Peak Rel. Acc. [g]')
plt.legend(loc='best')
plt.title('Iwanuma peak relative acceleration')
plt.grid(True)

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



Question 2 – (25 points)

A 3-m-long vertical cantilever made of a 150-mm-nominal-diameter standard steel pipe supports a 1200-kg mass attached at the tip, as shown in Fig. 2-1. The properties of the pipe are: outside diameter = 168.3 mm, inside diameter = 154.1 mm, thickness = 7.1 mm, Young's modulus E = 200,000 MPa, and mass per unit length = 28.19 kg/m. Determine the peak deformation and the bending stress in the cantilever due to the Canoga Park earthquake record that was given in Question 1; assume that ζ = 5%.

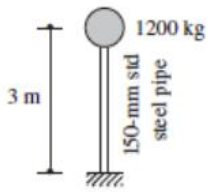


Figure 1-1 – Single degree of freedom oscillator

Question 2 (25 points)

The inertie of the pipe is given as:

$$I = \frac{\pi(D_{ext}^2 - D_{int}^2)}{64} = 11.7 * 10^6[mm^4];$$

The lateral stiffness of the pipe:

(5 points)

$$K = \frac{3EI}{L^3} = 260\ [\frac{N}{mm}];$$

The mass that is participating (half of the pipe mass is considered, to discard it for it is much lesser than that of the main mass is correct too):

$$m = 1200\ [kg] + 28.19\ \left[\frac{kg}{m}\right] * \frac{3[m]}{2} = 1200\ [kg] + 42.28\ [kg] = 1242.28\ [kg]$$

The natural frequency:

(5 points)

$$\omega_n = \sqrt{\frac{k}{m}} = \sqrt{\frac{260 * 10^3\ [\frac{N}{m}]}{1242.28\ [kg]}} = 14.47\ [\frac{rad}{s}]$$

$$T_n = \frac{2\pi}{\omega_n} = 0.434\ [s]$$

According to question 1 (Canoga Park earthquake record assuming that ζ = 5%):

(5 points)

$$T_n = 0.434\ [s] \rightarrow S_u = 53.26\ [mm]$$

The lateral force applied is then:

$$\rightarrow F = K * S_u = 13.85\ [kN]$$

Moment at the base of the column:
(5 points)

$$M = F * h = 13.85\ [kN] * 3[m] = 41.54\ [kN.m]$$

The bending stress is given as:
(5 points)

$$\sigma_{max} = \frac{M}{I} \cdot \frac{D_{ext}}{2} = 298.7 \text{ N/mm}^2$$

Exercise 3

Write a script at the programming language of your preference to compute the response of inelastic SDF systems under earthquake excitations. Use the Central Difference Method. Plot the response (relative displacement and absolute acceleration histories) of the following SDF oscillators in the same graph:

a. T_n = 1 s, ζ = 5%, $\bar{f}$ < = 1 (elastic response)

b. T_n = 1 s, ζ = 5%, $\bar{f}$ < = 0.5

c. T_n = 1 s, ζ = 5%, $\bar{f}$ < = 0.25

Use the Canoga Park record from Question 1. Discuss your findings by comparing the response of the three SDF systems

Solution exercise 3

We import the Canoga Park record. We will name it p.

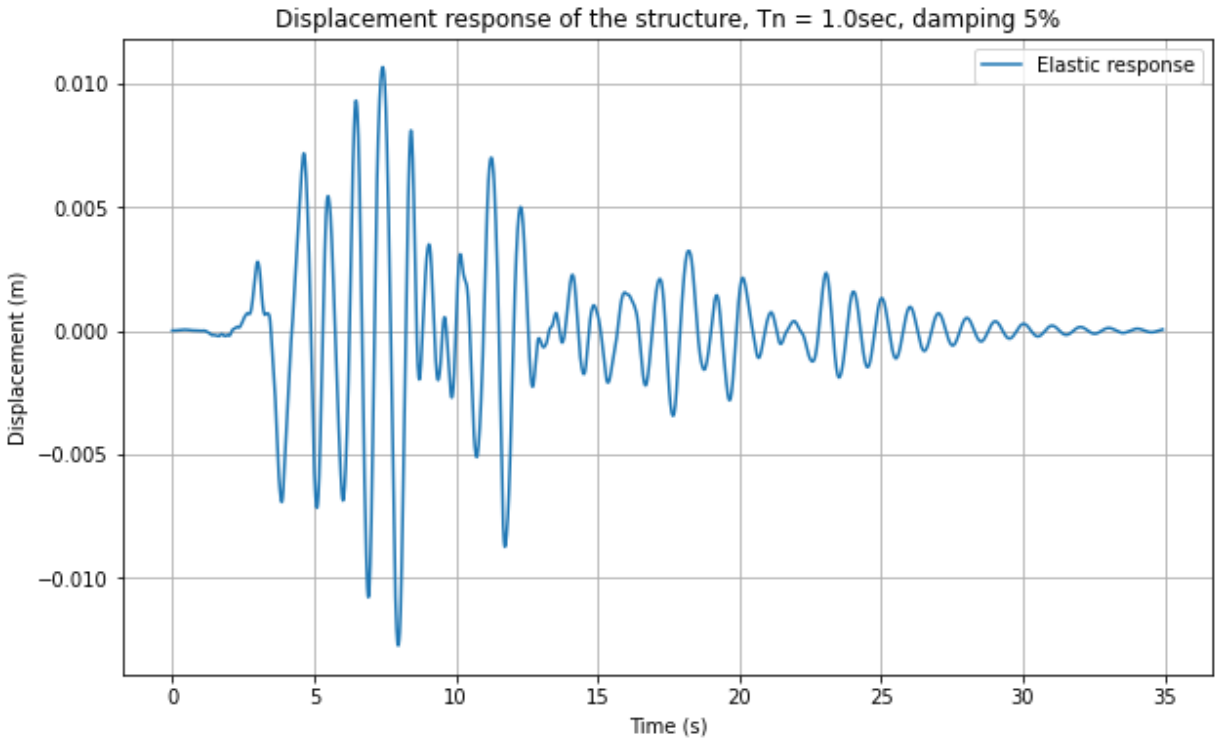
In [26]:
```
file_path = Path('Canoga_Park.csv')
data = pd.read_csv(file_path, header = None)
timestep = np.array(data[0]) #extract the time steps as an array
g = 9.81
p = np.array(data[1])    #extract the corresponding ground acceleration as an array

for ii in range(1, 1000):
    timestep = np.append(timestep, timestep[-1] + (timestep[1] - timestep[0]))
    p = np.append(p, 0)
```

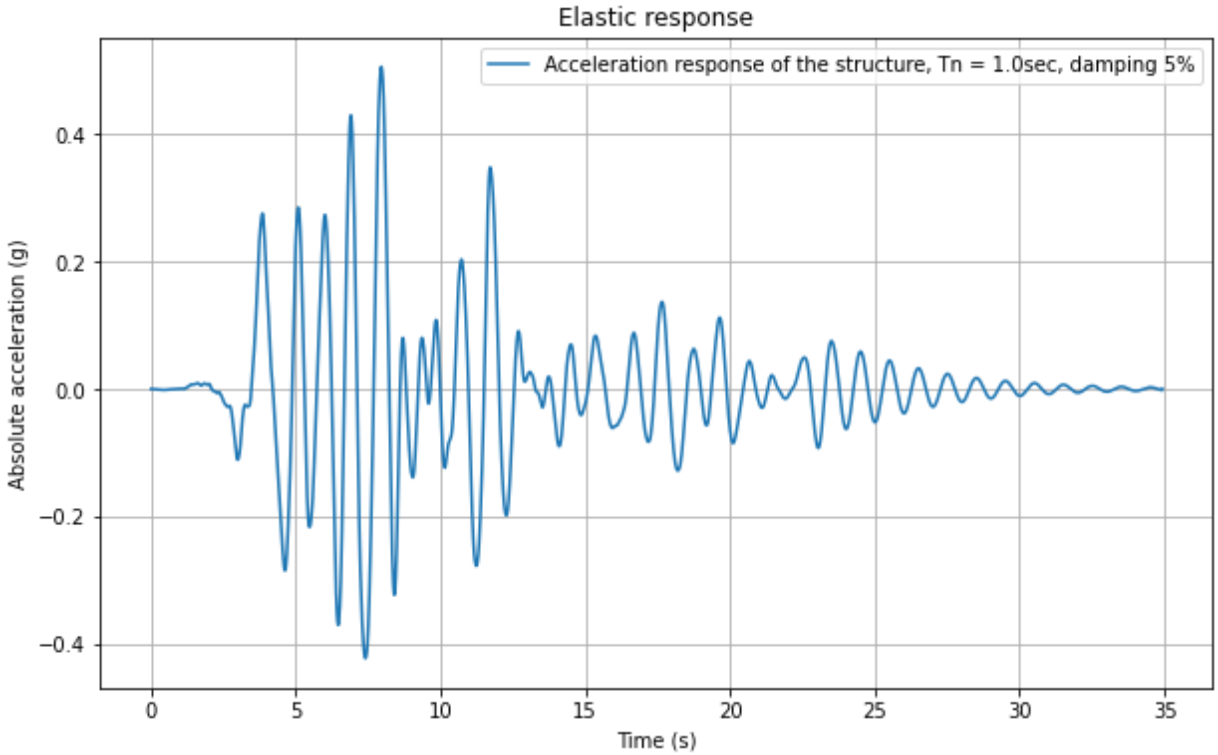
Part a - Elastic response

First, we will look at the system that behaves purely elastic (case a). The graphs for the response of the structure are given below in terms of displacement, velocity and acceleration of the system.

In [27]:
```
def elastic_response(ground_acceleration, time, Tn, damping_ratio, initial_displacem
    # Time step
    dt = time[1]-time[0]  # we will take the same time step as given in the record

    # Mass, stiffness, and damping parameters of your structure
    omega_n = 2*np.pi/Tn                          # Circular frequency of the system
    # m = 1  we consider a unit mass and choose the stiffness accordingly
    k = 4*np.pi**2/(Tn**2)                        # Stiffness of the structure (N/m)
    c = damping_ratio*2*omega_n                     # Damping ratio of the structure

    # Initialize arrays to store the response
    num_steps = len(ground_acceleration)
    displacement = np.zeros(num_steps)
    displacement[0] = initial_displacement

    velocity = np.zeros(num_steps)
    velocity[0] = initial_velocity

    acceleration = np.zeros(num_steps)
    acceleration[0]=-p[0]-c*velocity[0]-k*displacement[0]
```

```
    abs_acceleration =np.zeros(num_steps)

    displacement[1] = initial_displacement-dt*initial_velocity-dt**2*acceleration[0]


    k_hat = dt**(-2)+c/(2*dt)
    alpha = dt**(-2)-(c/(2*dt))
    beta = k-(2/(dt**2))

    # Perform dynamic analysis using the central difference method
    for i in range(1,num_steps-1):
        p_hat = -ground_acceleration[i]-(alpha*displacement[i-1])-(beta*displacement
        displacement[i+1]=p_hat/k_hat

        velocity[i]=(displacement[i+1]-displacement[i-1])/(2*dt)
        acceleration[i]=(displacement[i+1]-(2*displacement[i])+displacement[i-1])/(d
        abs_acceleration[i]=acceleration[i]+ground_acceleration[i]


    time = np.arange(0, num_steps * dt, dt)

    return np.array([time,displacement,velocity,abs_acceleration])

[time_1, displacement_1, velocity_1,acceleration_1] = elastic_response(p,timestep,1.(

# Plot the response
plt.figure(figsize=(10, 6))
plt.plot(time_1, displacement_1, label='Elastic response')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Displacement response of the structure, Tn = 1.0sec, damping 5%')
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(time_1, acceleration_1, label='Acceleration response of the structure, Tn =
plt.xlabel('Time (s)')
plt.ylabel('Absolute acceleration (g)')
plt.title('Elastic response')
plt.grid(True)
plt.legend()
plt.show()
```





Before coninuing to the next part of the exercise, we calclate the maximum displacement.This will give us the the maximum resistance for elastic behaviour that we can then use to calculate the elastic limits for cases b and c.

In [28]:
```
u_max=max(abs(displacement_1))
print(f"The maximum displacement the system can handle is {u_max} m")
```

```
The maximum displacement the system can handle is 0.012749537655215536 m
```

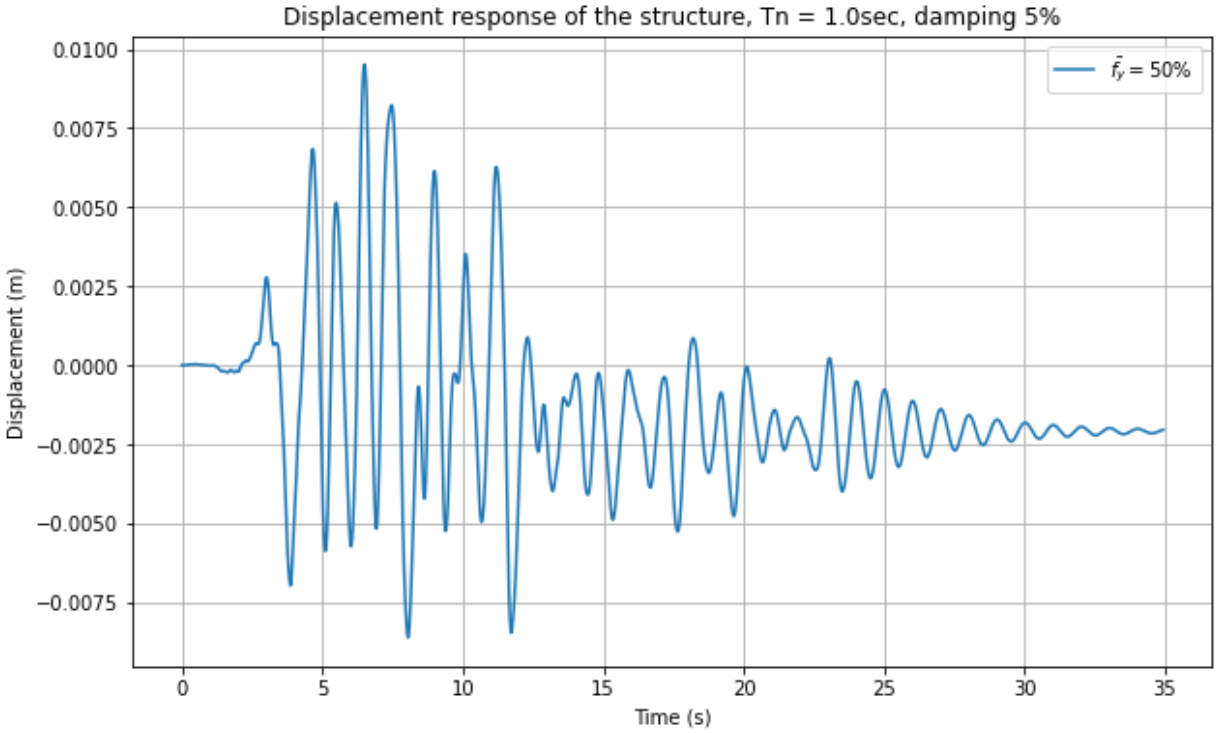
Part b - Plastic limit at half the demand

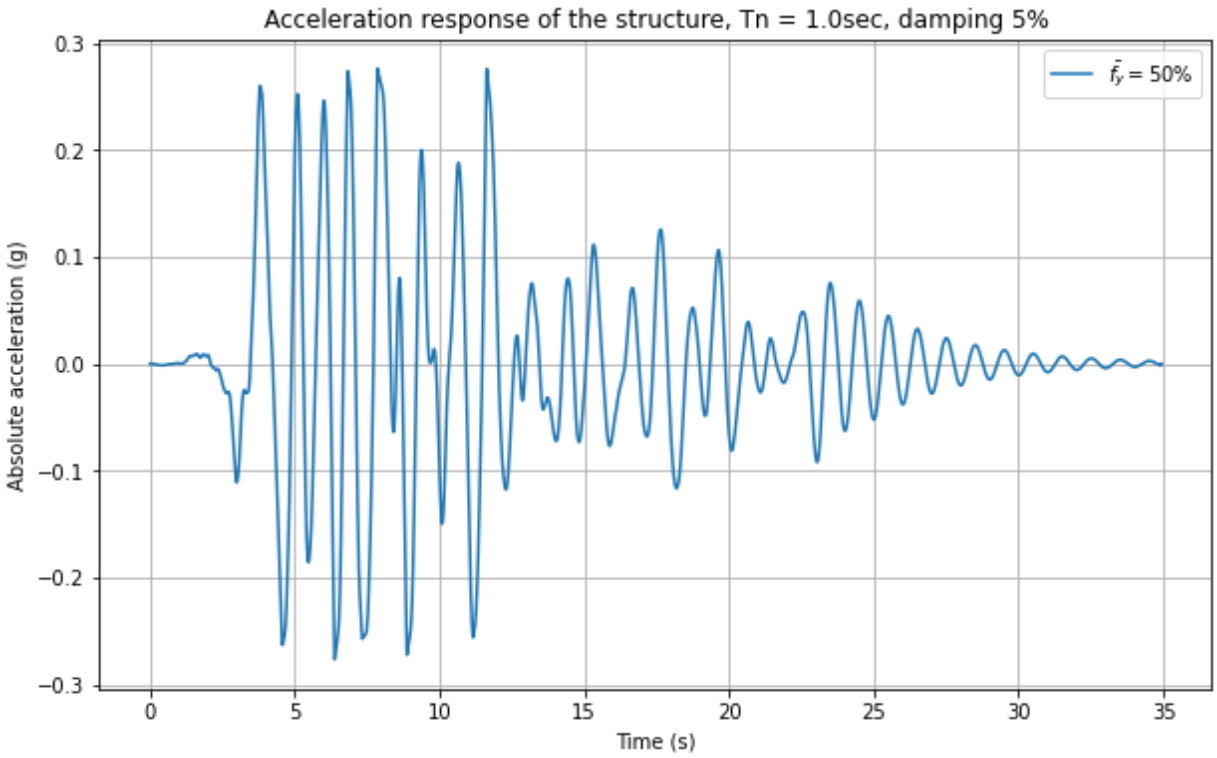
In the second part we want to look at the plastic response of the structure if it can only handle half of the displacement demand from the elastic system in an elastic way. This means that the structure will start to plastify after reaching a displacement of 50 % of 0.013 m (or 13 mm).

In [29]:

```
def plastic_response(ground_acceleration, time, Tn, damping_ratio, initial_displacem
     # Time step
    dt = time[1]-time[0]  # we will take the same time step as given in the record

    # Mass, stiffness, and damping parameters of your structure
    omega_n = 2*np.pi/Tn                          # Circular frequency of the system
    m = 1  #we consider a unit mass and choose the stiffness accordingly
    k = m*4*np.pi**2/(Tn**2)                        # Stiffness of the structure (N/
    c = m*damping_ratio*2*omega_n                     # Damping ratio of the structu

    # Define the plastic behaviour
    f_el = k * max_disp  # This is the plastic limit for 100 % elasticity i.e. the r
    f_y = f_el * f_y_bar  # This is the elastic limit at f_y_bar of the elastic resi
    u_y = max_disp * f_y_bar  # This is the maximum elastic displacement

    # Initialize arrays to store the response
    num_steps = len(ground_acceleration)
    displacement = np.zeros(num_steps)
    displacement[0] = initial_displacement
    fs = k * initial_displacement
    f_s = np.zeros(num_steps)
    df_s = np.zeros(num_steps)

    velocity = np.zeros(num_steps)
    velocity[0] = initial_velocity

    acceleration = np.zeros(num_steps)
    acceleration[0]= -p[0]-c*velocity[0]-k*displacement[0]
    abs_acceleration =np.zeros(num_steps)

    displacement[1] = initial_displacement-dt*initial_velocity-dt**2*acceleration[0]


    k_hat = dt**(-2)+c/(2*dt)
    alpha = dt**(-2)-(c/(2*dt))
    beta_initial = k-(2/(dt**2))

    # Perform dynamic analysis using the central difference method
    for i in range(1,num_steps-1):
        if i == 1:
            f_s[i] = fs
        beta = f_s[i]/displacement[i]-2/dt**2
        # first we need to determine if the structure still behaves elastically or i
        #if abs(displacement[i])> u_y:
            #f_s = f_y
            #beta = (f_s/abs(displacement[i])) - (2/dt**2)
        #else:
         #   beta=beta_initial
        p_hat = -ground_acceleration[i]-(alpha*displacement[i-1])-beta*displacement[
        displacement[i+1]=p_hat/k_hat
        velocity[i]=(displacement[i+1]-displacement[i-1])/(2*dt)
        acceleration[i]=(displacement[i+1]-(2*displacement[i])+displacement[i-1])/(d
        abs_acceleration[i] = acceleration[i] + ground_acceleration[i]

        df_s[i]=k*(displacement[i+1]-displacement[i])
        f_s[i+1] = f_s[i] + df_s[i]

        if abs(f_s[i+1])>f_y:
            f_s[i+1] = np.sign(velocity[i])*f_y

    time = np.arange(0, num_steps * dt, dt)

    return np.array([time,displacement,velocity,abs_acceleration])
```

```
# Plot the response
[time_2, displacement_2, velocity_2,acceleration_2] = plastic_response(p,timestep,1.(
plt.figure(figsize=(10, 6))
plt.plot(time_2, displacement_2, label='$\\bar{f_y}=50\\%$')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Displacement response of the structure, Tn = 1.0sec, damping 5%')
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(time_2, acceleration_2, label='$\\bar{f_y}=50\\%$')
plt.xlabel('Time (s)')
plt.ylabel('Absolute acceleration (g)')
plt.title('Acceleration response of the structure, Tn = 1.0sec, damping 5%')
plt.grid(True)
plt.legend()
plt.show()
```



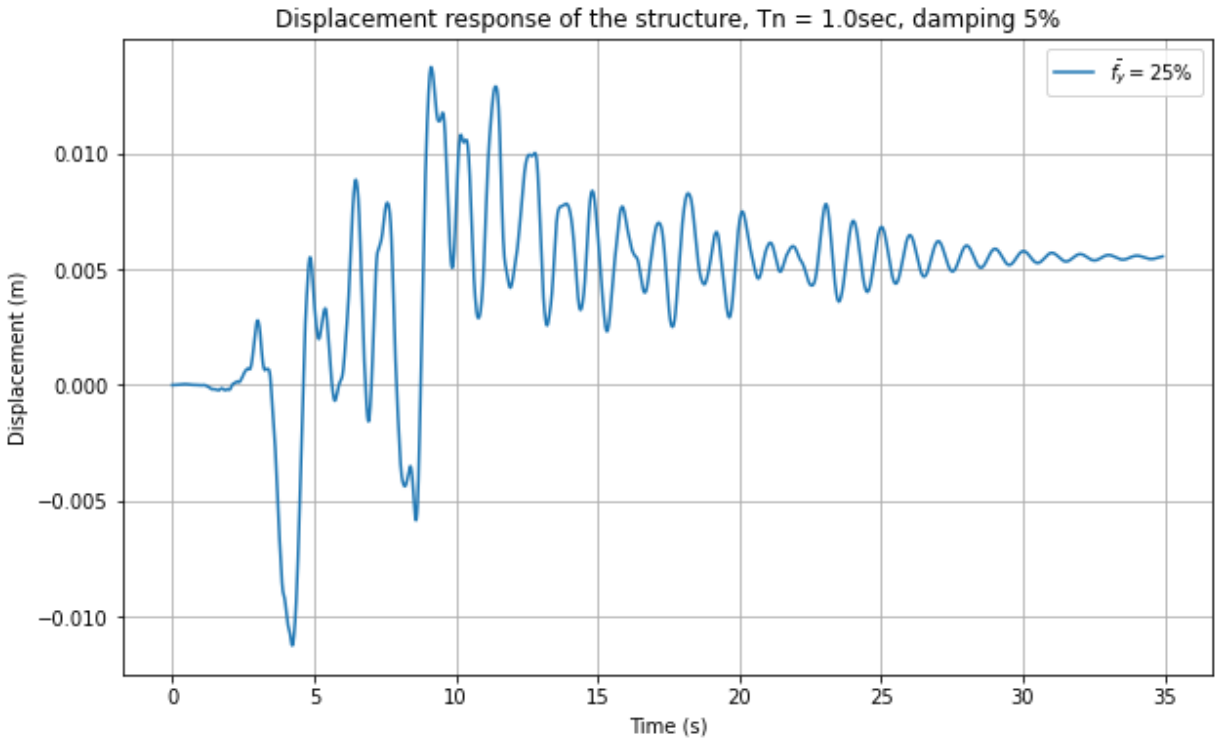


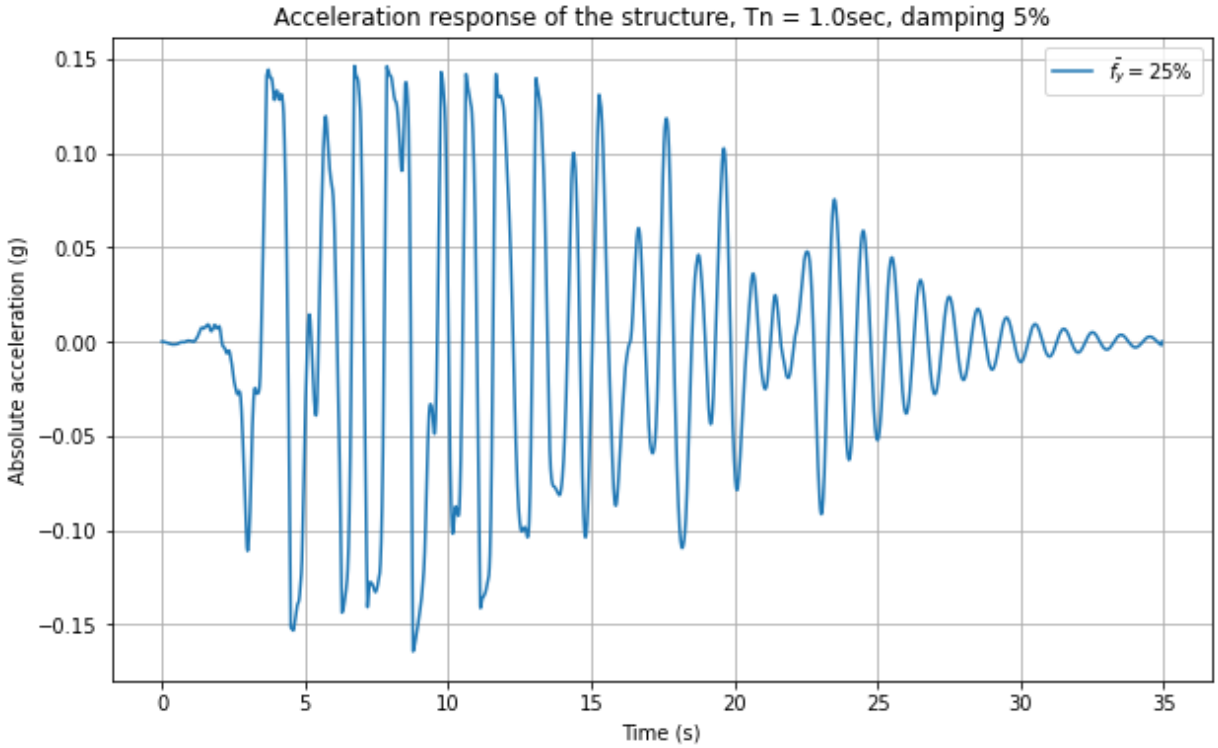
Part c - Plastic limit at a quarter of the demand

In the second part we want to look at the plastic response of the structure if it can only handle half of the displacement demand from the elastic system in an elastic way. This means that the structure will start to plastify after reaching a displacement of 25 % of 0.013 m (or 13 mm).

In [30]:

```
[time_3, displacement_3, velocity_3,acceleration_3] = plastic_response(p,timestep,1.(

# Plot the response
plt.figure(figsize=(10, 6))
plt.plot(time_3, displacement_3, label='$\\bar{f_y}=25\\%$')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Displacement response of the structure, Tn = 1.0sec, damping 5%')
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(time_3, acceleration_3, label='$\\bar{f_y}=25\\%$')
plt.xlabel('Time (s)')
plt.ylabel('Absolute acceleration (g)')
plt.title('Acceleration response of the structure, Tn = 1.0sec, damping 5%')
plt.grid(True)
plt.legend()
plt.show()
```



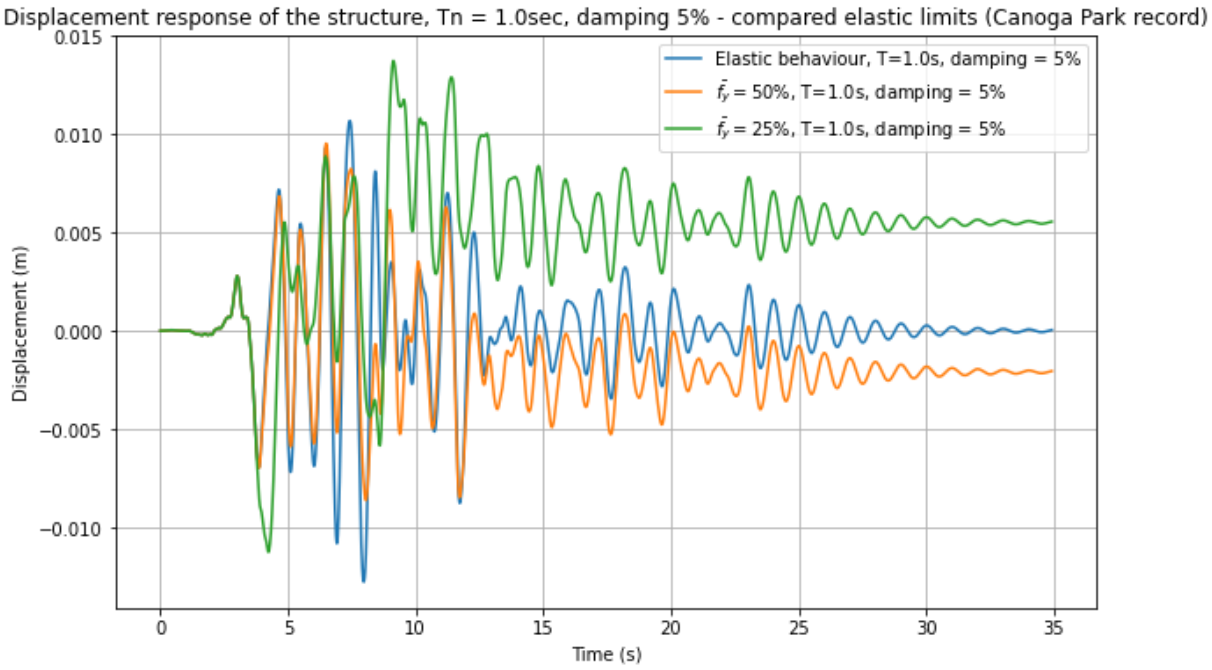


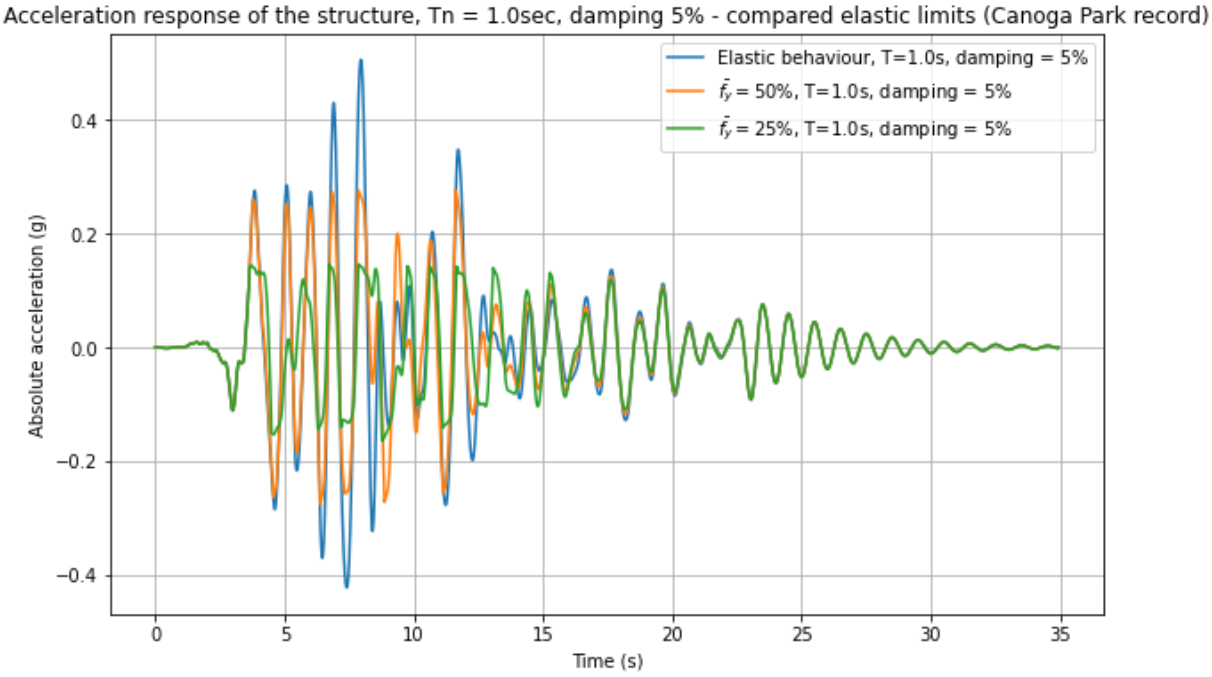
Summary of the results and discussion

In [31]:

```
# Plot the comleteresponse
plt.figure(figsize=(10, 6))
plt.plot(time_1, displacement_1, label='Elastic behaviour, T=1.0s, damping = 5%')
plt.plot(time_2, displacement_2, label='$\\bar{f_y}=50\\%$, T=1.0s, damping = 5%')
plt.plot(time_3, displacement_3, label='$\\bar{f_y}=25\\%$, T=1.0s, damping = 5%')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Displacement response of the structure, Tn = 1.0sec, damping 5% - compare
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
```

```
plt.plot(time_1, acceleration_1, label='Elastic behaviour, T=1.0s, damping = 5%')
plt.plot(time_2, acceleration_2, label='$\\bar{f_y}=50\\%$, T=1.0s, damping = 5%')
plt.plot(time_3, acceleration_3, label='$\\bar{f_y}=25\\%$, T=1.0s, damping = 5%')
plt.xlabel('Time (s)')
plt.ylabel('Absolute acceleration (g)')
plt.title('Acceleration response of the structure, Tn = 1.0sec, damping 5% - compare
plt.grid(True)
plt.legend()
plt.show()
```





Let's first look at the displacement graphs: As can be seen, the displacement response of the elasitc system comes back to the initial position at the end. Looking at the two other curves, we can also see that the absolute residual displacement of the system that has a lower elastic limit $\bar{f_y} = 0.25$ (green) is bigger than for the system with the higher elastic limit $\bar{f_y} = 0.5$ (orange). Since the system with the lower elastic limit plastifies for a lower force / displacement, the plastic deformation starts earlier and accumulates more. Looking at the absolute acceleration graph: We can see that the acceleration of the system decreases for lower elastic limits. This can be explained with the energy dissipation of the system. While the purely elastic system only has the

dampers to dissipate energ, the plastifying systems also dissipate energy with the plastification, making the acceleration of the system smaller. This is particularly of advantage for buildings with sensitive contents as machinery or valuable items. Learnings for buildings: The lower the elastic limit of a structure, the sooner it plastifies and the bigger the residual displacement at the end of the earthquake. This means that the structure will be damaged and maybe even needs to be demolished afterwards (eventhough no collapse happened. The positive aspect on the other hand is, that the acceleration of the building is smaller which is good for sensible contents that need to be preserved at all cost.

Exercise 4

For a system with $T_n = 0.5\,s$ and ζ = 5% and the El Centro ground motion, show that fro $\bar{f}_y = 0.25$ the ductilitiy factor μ is unaffected by scaling the ground motion.

Solution exercise 4

We first need to start to introduce the ground motion of El Centro to the code: (you will need to change the path to where you stored the ground motion on your computer)

In [32]:
```
# Specify the path to your file
file_path = Path('elcentro.txt')

# Read the DAT file into a DataFrame
data = pd.read_csv(file_path, header=None, delim_whitespace=True)

# Extract the time steps and ground acceleration as arrays
timestep = np.array(data[0])
p = np.array(data[1])

for ii in range(1, 1000):
    timestep = np.append(timestep, timestep[-1] + (timestep[1] - timestep[0]))
    p = np.append(p, 0)
```

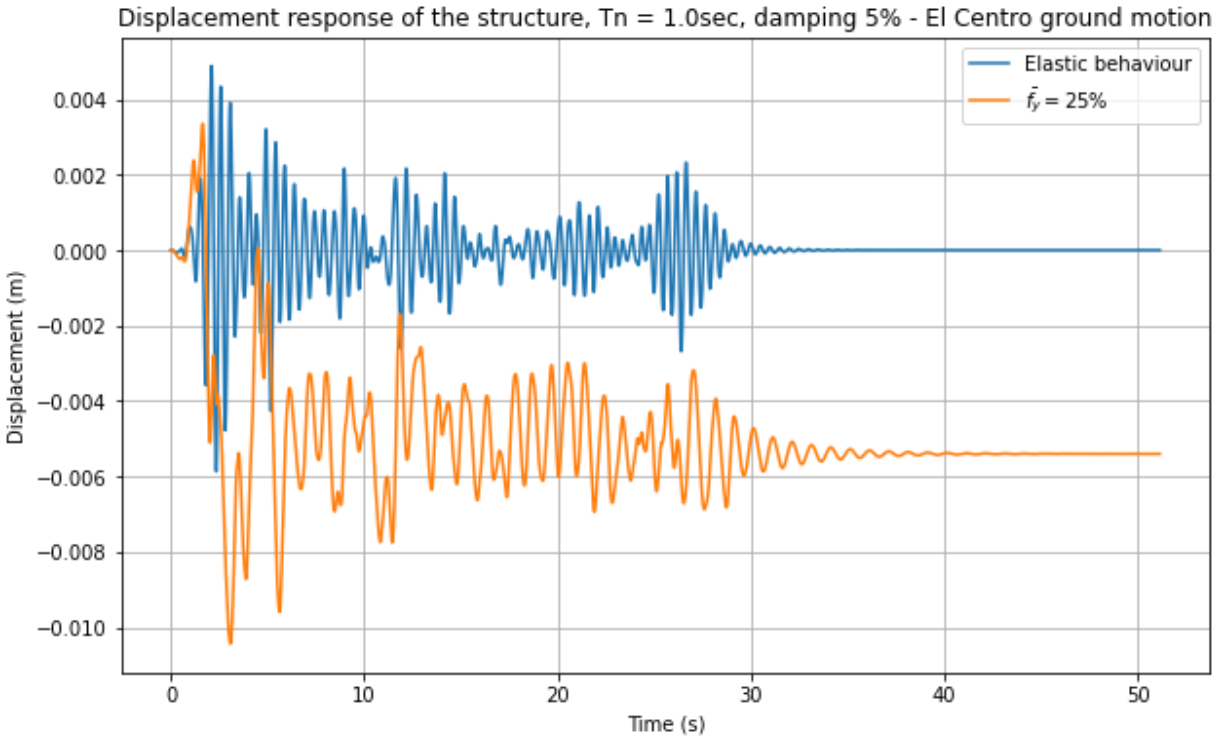
First we need to compute the elastic response to be able to determine the elastic limit.

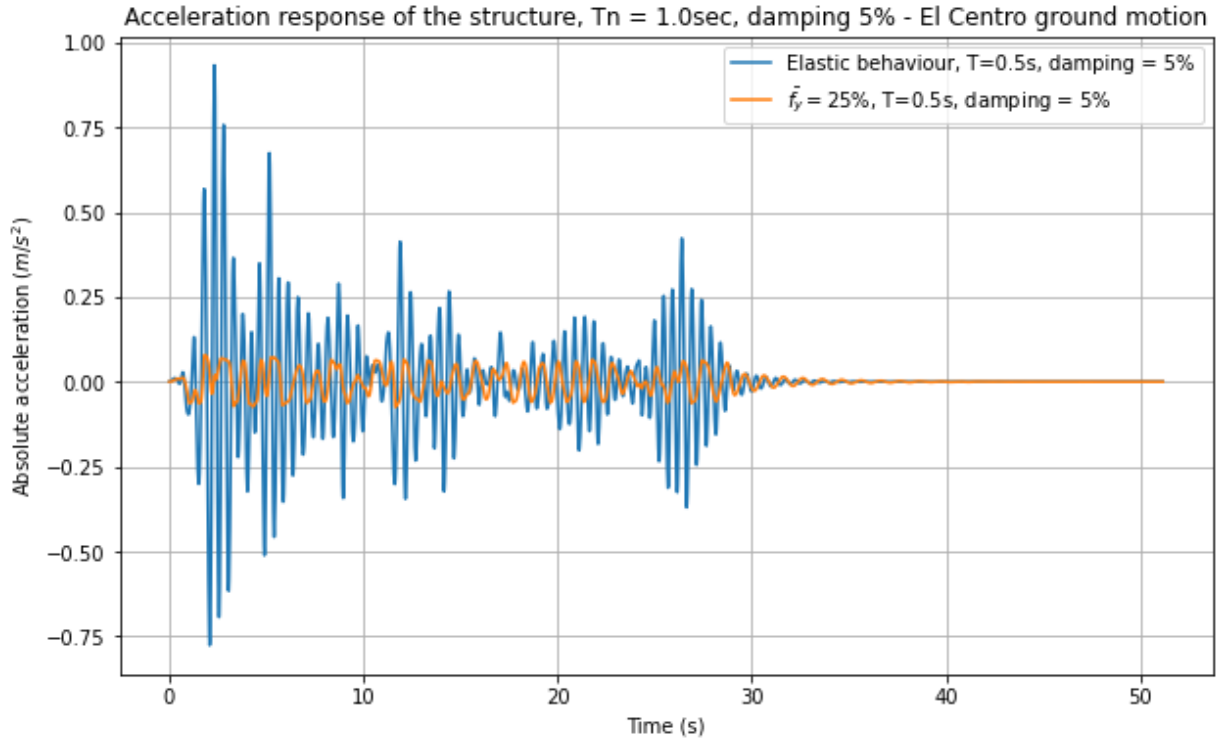
In [33]:
```
[time_4, displacement_4, velocity_4,acceleration_4] = elastic_response(p,timestep,0.

u_max=max(abs(displacement_4))
```

Now that we have the elastic response we can proceed as in exercise 3 b/c and compute the plastic response:

In [34]:
```
[time_5, displacement_5, velocity_5,acceleration_5] = plastic_response(p,timestep,1.(

# Plot the results
# Plot the comlete response
plt.figure(figsize=(10, 6))
plt.plot(time_4, displacement_4, label='Elastic behaviour')
plt.plot(time_5, displacement_5, label='$\\bar{f_y}=25\\%$')
plt.xlabel('Time (s)')
plt.ylabel('Displacement (m)')
plt.title('Displacement response of the structure, Tn = 1.0sec, damping 5% - El Centr
plt.grid(True)
plt.legend()
plt.show()
```

```
plt.figure(figsize=(10, 6))
plt.plot(time_4, acceleration_4, label='Elastic behaviour, T=0.5s, damping = 5%')
plt.plot(time_5, acceleration_5, label='$\\bar{f_y}=25\\%$, T=0.5s, damping = 5%')
plt.xlabel('Time (s)')
plt.ylabel('Absolute acceleration ($m/s^{2}$)')
plt.title('Acceleration response of the structure, Tn = 1.0sec, damping 5% - El Centr
plt.grid(True)
plt.legend()
plt.show()
```





Now to come back to the original question: We want to prove that the ductility factor is unaffected by the ground motion. First of all, we know that the ductility factor is something that is proper to the structural system and its material parameters. We can describe the ductility factor as follows:

$$\mu = u_m / u_y$$

Where u_m is the maximum plastic displacement a structure can accomodate and u_y is the maximum elastic displacement. Then we also know that the f_y factor is defined as:

$$f_y = u_y / u_0$$

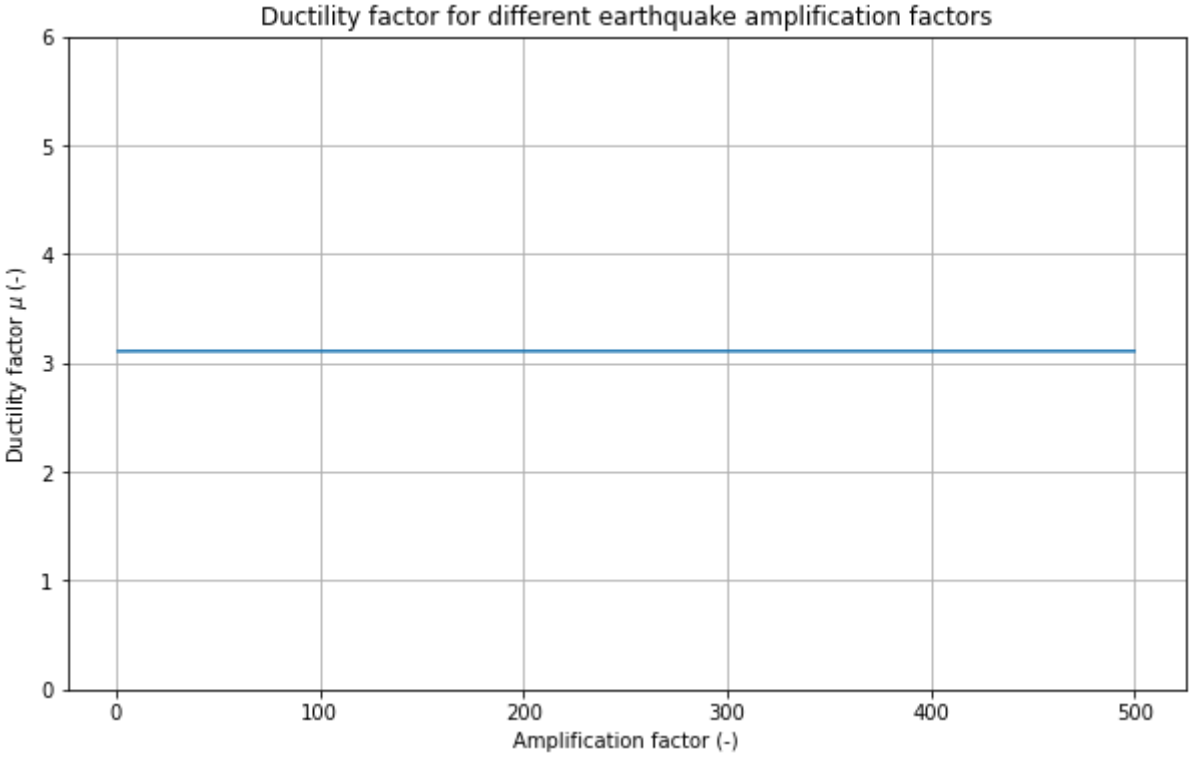
Where u_0 is the maximum elastic displacement for a purely elastic system (meaning f_y is equal to 1.0). By combining the two formulas we can get:

$$\mu = u_m / (f_y * u_0)$$

If we define u_m as the maximum displacement of the plastic system and u_0 as the maximum displacement of the elastic system and we loop through different earthquake factors we can see that the ductility factor stays constant.

In [35]:

```
f_y_bar = 0.25
factor = np.linspace(1, 500, 250, dtype=int)
mu = np.zeros(len(factor))

for i in range(0,len(factor)):
    [time_e, displacement_e, velocity_e, acceleration_e] = elastic_response(p * facto
    u_0 = max(abs(displacement_e))
    [time_p, displacement_p, velocity_p, acceleration_p] = plastic_response(p * facto
    u_m = max(abs(displacement_p))
    mu[i] = u_m / (u_0 * f_y_bar)

plt.figure(figsize=(10, 6))
plt.plot(factor, mu)
plt.xlabel('Amplification factor (-)')
plt.ylabel('Ductility factor $\mu$ (-)')
plt.title('Ductility factor for different earthquake amplification factors')
plt.grid(True)
plt.ylim(0, 6)
plt.show()
```


Ductility factor for different earthquake amplification factors
Ductility factor μ (-)
Amplification factor (-)
0
1
2
3
4
5
6
0
100
200
300
400
500