(0) Packages

In [ ]:

```
# Run this cell to make sure the mdof module is installed.
!pip install mdof
```

In [1]:

```
# Import the required packages.
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
import mdof
```

(1) Load Files

In [2]:

```
# Path to the directory where files are saved
DATA_DIR = Path("../../uploads/mini_shake_table/Flexible vs. Stiff")
# Get all the files ending in csv
files = list(DATA_DIR.glob("**/*.csv"))
# Identify the row with the column labels
header_row = 8
# Get the column labels
with open(files[0], "r") as readfile:
    header_keys = readfile.readlines()[header_row-1].split(',')
# Identify the column indices for each acceleration component
x_index = header_keys.index('Acc_X')
y_index = header_keys.index('Acc_Y')
z_index = header_keys.index('Acc_Z')

# Populate a dictionary with the data. Each file is its own item.
data = {}
for file in files:
    filename = f"{file.parent.parent.name} {file.parent.name} - {file.name.s
    data[filename] = np.loadtxt(file,
                                delimiter=",",
                                skiprows=header_row, # Get all the rows afte
                                usecols=[x_index,y_index,z_index] # Get only
                               )
```

Print the filenames used as keys in the `data` dictionary

In [ ]:

```
# We'll use this list of filenames as reference for the options
# of records used in the system identification code down below.

for filename in data.keys():
    print(filename)
```

(2) Plot the records

- top floor only
- x direction only

In [3]:

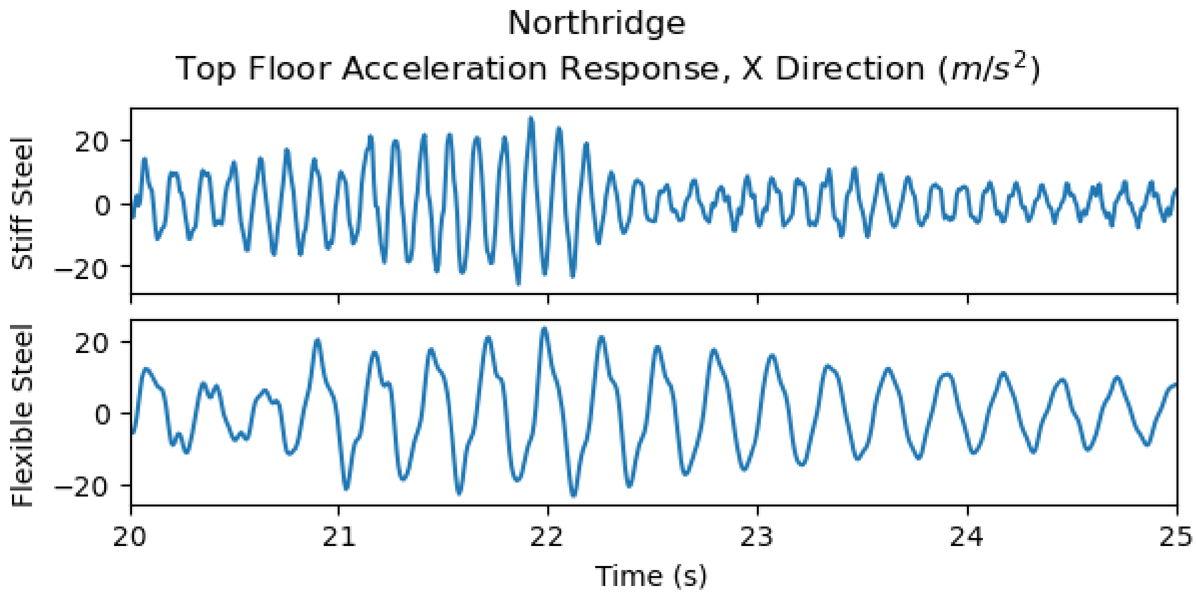
```
# Get the time array!
SAMPLE_RATE = 120 #Hz
# If there are sampling_rate samples taken per second, what is the amount of
# That is, how many seconds per sample?
TIME_STEP = 1/SAMPLE_RATE
# Create a function that returns a time array that starts at zero,
# has a length of num_points, and has a step of time_step between
# each point.
# note: the syntax for np.linspace is np.linspace(start,stop,length)
def time_array(time_step, num_points):
    return np.linspace(0,time_step*num_points,num_points)
```

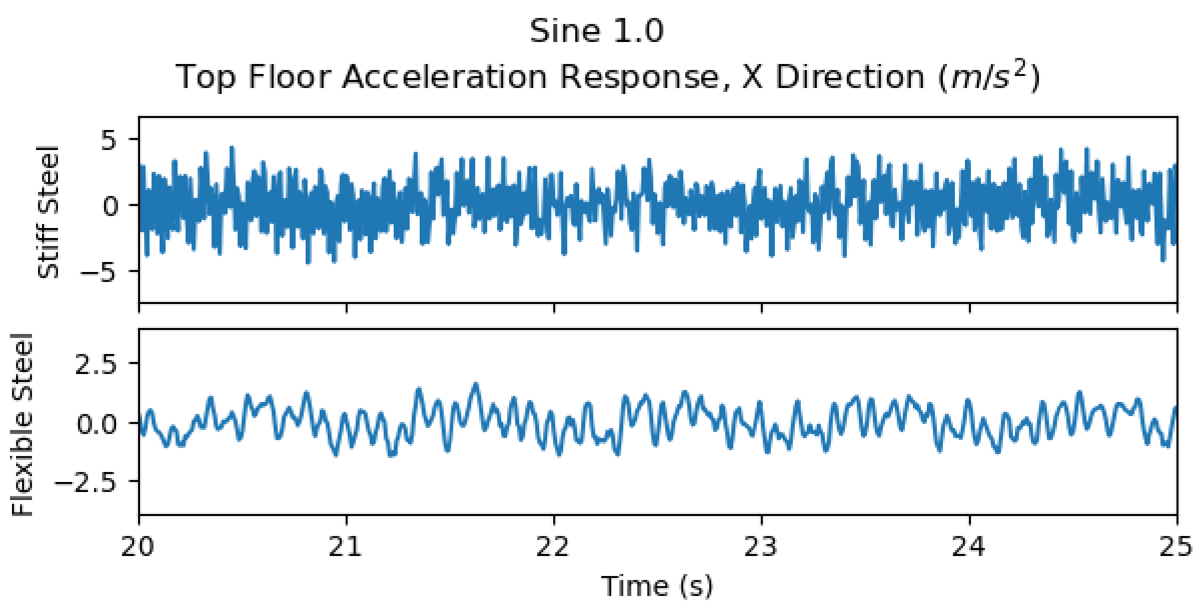
Compare the stiff to the flexible three story model.

In [4]:

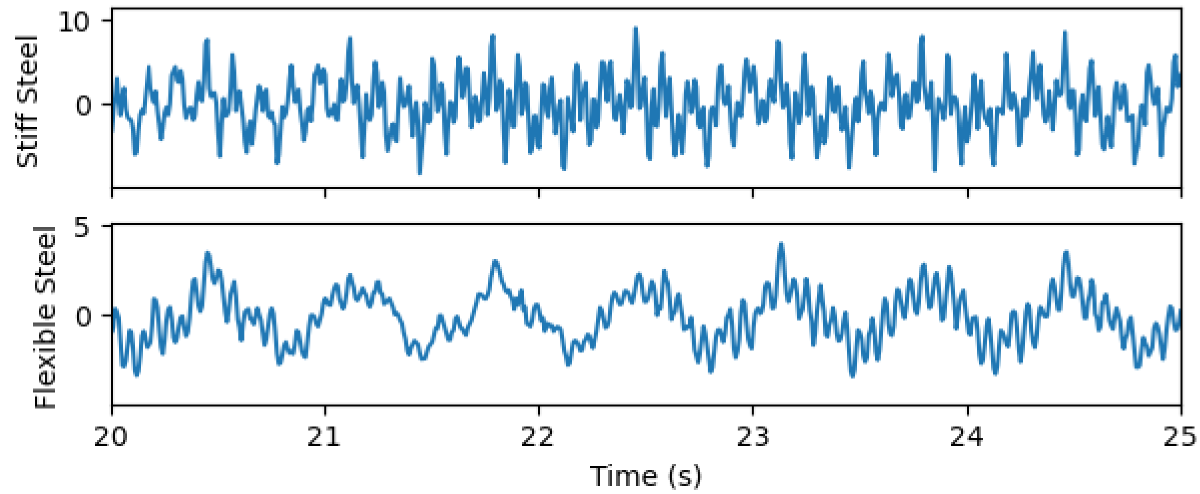
```
# Path to a directory where we'll save figures
OUT_DIR = Path("out/")
if not OUT_DIR.exists():
    OUT_DIR.mkdir()

event_names = ["Northridge", "Sine 1.0", "Sine 1.5", "Sine 2.0", "Bonus Ever

# Loop through all events
for event_name in event_names:
    # Plot the records!
    # Only plot the top floor response.
    # Plot stiff 3 story and flexible 3 story steel side-by-side.
    outputs_stiff = data[f'Stiff 3 story 3rd Floor (Top floor) - ({event_nam
    outputs_flex = data[f'Flexible 3 story 3rd Floor (Top floor) - ({event_r
    # Make the records the same length.
    npts = min(outputs_stiff.shape[0], outputs_flex.shape[0])
    outputs_stiff = outputs_stiff[:npts]
    outputs_flex = outputs_flex[:npts]
    # Construct the time array
    time = time_array(TIME_STEP, npts)
    # Create a figure with two subplots stacked vertically (2 rows, 1 column
    fig,ax = plt.subplots(2, 1, figsize=(6,3), sharex=True, constrained_layc
    # Plot the X direction, which is the first column of record
    ax[0].plot(time, outputs_stiff[:,0])
    ax[1].plot(time, outputs_flex[:,0])
    # Labels, limits and title
    ax[0].set_ylabel('Stiff Steel')
    ax[1].set_ylabel('Flexible Steel')
    ax[1].set_xlabel('Time (s)')
    ax[1].set_xlim((20,25))
    if "Bonus" in event_name:
        ax[1].set_xlim((17,20))
```

```
    fig.suptitle(f"{event_name} \n Top Floor Acceleration Response, X Direct
    # Save the figure
    fig.savefig(OUT_DIR/f"{event_name}.png")
```

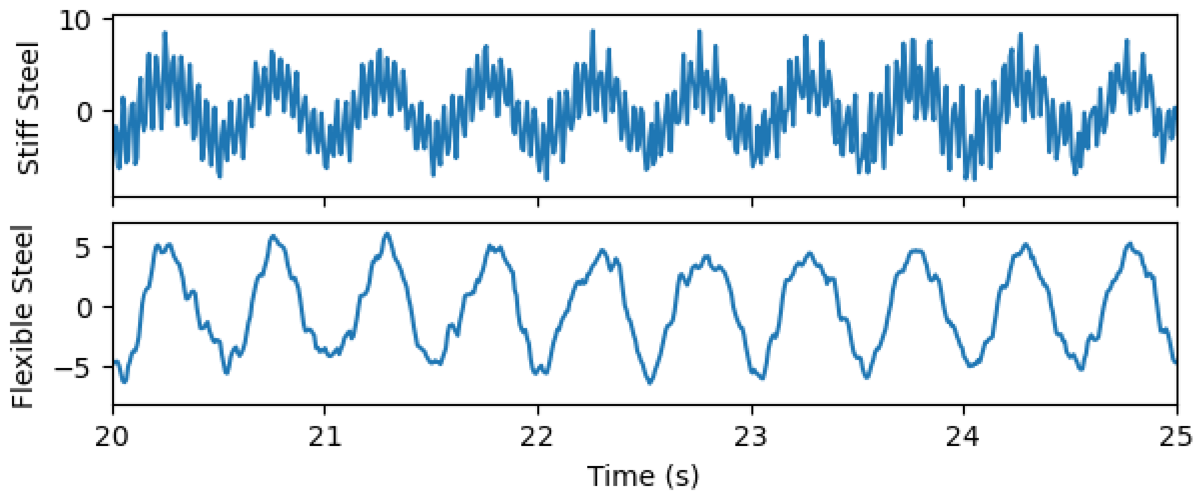




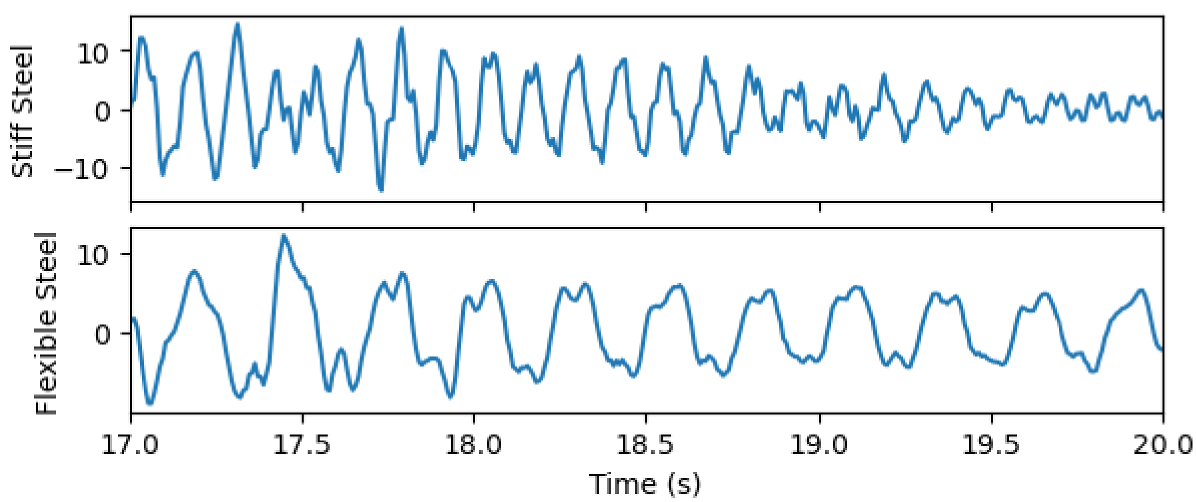
Sine 1.5
Top Floor Acceleration Response, X Direction (m/s^2)



Sine 2.0
Top Floor Acceleration Response, X Direction (m/s^2)



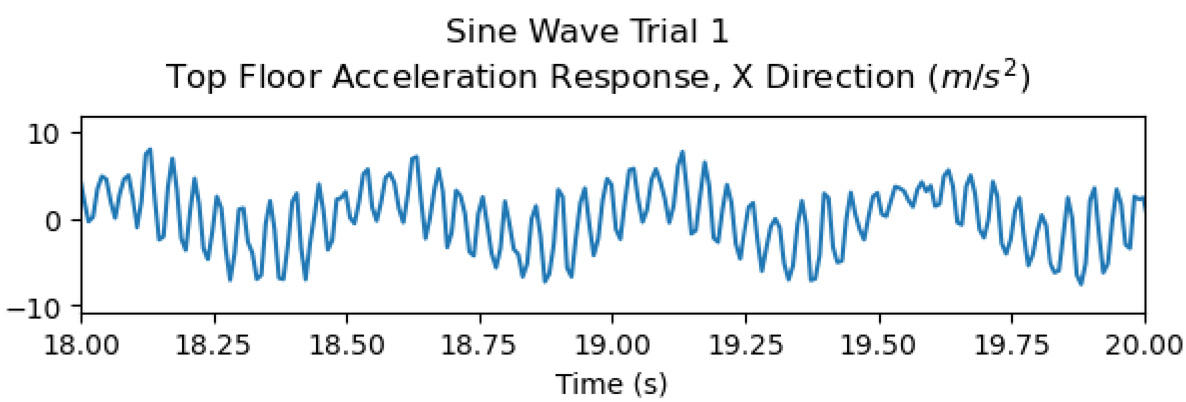
Bonus Event
Top Floor Acceleration Response, X Direction (m/s^2)



Plot the stiff one story model.

Just plot a preview of the Sine Wave record.

In [5]:
```
event_names = [
    # "El Centro Trial 1", "El Centro Trial 2", "Northridge Trial 1", "North
    "Sine Wave Trial 1",
    # "Sine Wave Trial 2",
               ]

# Loop through all events
for event_name in event_names:
    # Plot the records!
    # Only plot the top floor response.
    outputs = data[f'Stiff 1 story Top Floor - ({event_name})']
    # Construct the time array
    npts = outputs.shape[0]
    time = time_array(TIME_STEP, npts)
    # Create a figure
    fig,ax = plt.subplots(figsize=(6,2), sharex=True, constrained_layout=Tru
    # Plot the X direction, which is the first column of record
    ax.plot(time, outputs[:,0])
    # Labels, limits and title
    ax.set_xlabel('Time (s)')
    if "Sine" in event_name:
        ax.set_xlim((18,20))
    elif "El Centro" in event_name:
        ax.set_xlim((8,15))
    else:
        ax.set_xlim((5,12))
    fig.suptitle(f"{event_name} \n Top Floor Acceleration Response, X Direct
    # Save the figure
    fig.savefig(OUT_DIR/f"{event_name}.png")
```



(3) Discussion: what can we say about how ***frequency of oscillation*** varies with ***stiffness*** of the steel walls?

How does this make sense with the relation, $\omega_n = \sqrt{\frac{k}{m}}$?

Write your observations below.

- Frequency of oscillation increases as stiffness of the steel increases.
- k in the equation is stiffness. For stiffer steel, k is higher.
- m in the equation is mass. mass is constant.
- ω_n in the equation is natural frequency.
- Therefore, as stiffness increases in the equation, natural frequency increases by a square root relationship with the stiffness.

(4) Obtain natural frequencies with system identification

Unknown system with one input and one output

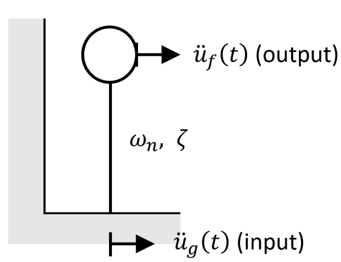
Let's treat these building models as single-degree-of-freedom oscillators with concentrated mass at the top, like the "lollipop" shown below.

This is a very common idealization for structural systems.

There are two parameters that define how this system moves: natural frequency and damping ratio.

parameter	value
ω_n	natural frequency
ζ	damping ratio

These two parameters can be determined experimentally by system identification of the structure's vibrations.



3 Story Stiff Steel

In [6]:

```
event_names = ["Northridge", "Sine 1.0", "Sine 1.5", "Sine 2.0", "Bonus Ever

# Loop through all events
for event_name in event_names:
    # Load the X component of the bottom sensor as input
    inputs = data[f'Stiff 3 story Ground Floor - ({event_name})'][:,0]
    # Don't use the entire time series. Start halfway through the record anc
    sixth_points = int(inputs.shape[0]/6)
    inputs = inputs[3*sixth_points:5*sixth_points]
    # Load the top sensor as output
    outputs = data[f'Stiff 3 story 3rd Floor (Top floor) - ({event_name})'][
    # Construct the time array
    time = time_array(TIME_STEP, len(inputs))
    # Use the mdof package to perform system identification and determine th
    P, Phi = mdof.modes(inputs, outputs, dt=TIME_STEP)
    # The fundamental natural period is the longest period.
    print(f"Event: {event_name} - Fundamental Period: {max(P):.3f} seconds")
```

```
Event: Northridge - Fundamental Period: 0.134 seconds
Event: Sine 1.0 - Fundamental Period: 0.125 seconds
Event: Sine 1.5 - Fundamental Period: 0.125 seconds
Event: Sine 2.0 - Fundamental Period: 0.125 seconds
Event: Bonus Event - Fundamental Period: 0.128 seconds
```

3 Story Flexible Steel

In [7]:

```
event_names = ["Northridge", "Sine 1.0", "Sine 1.5", "Sine 2.0", "Bonus Ever

# Loop through all events
for event_name in event_names:
    # Load the X component of the bottom sensor as input
    inputs = data[f'Flexible 3 story Ground Floor - ({event_name})'][:,0]
    # Don't use the entire time series. Start halfway through the record anc
    sixth_points = int(inputs.shape[0]/6)
    inputs = inputs[3*sixth_points:5*sixth_points]
    # Load the top sensor as output
    outputs = data[f'Flexible 3 story 3rd Floor (Top floor) - ({event_name})
    # Construct the time array
    time = time_array(TIME_STEP, len(inputs))
    # Use the mdof package to perform system identification and determine th
    P, Phi = mdof.modes(inputs, outputs, dt=TIME_STEP)
    # The fundamental natural period is the longest period.
    realization = mdof.sysid(inputs, outputs)
    print(f"Event: {event_name} - Fundamental Period: {max(P):.3f} seconds")
```

```
Event: Northridge - Fundamental Period: 0.273 seconds
Event: Sine 1.0 - Fundamental Period: 4.781 seconds
Event: Sine 1.5 - Fundamental Period: 0.288 seconds
Event: Sine 2.0 - Fundamental Period: 0.267 seconds
Event: Bonus Event - Fundamental Period: 0.270 seconds
```

1 Story Stiff Steel

In [8]:
```
event_names = ["El Centro Trial 1", "El Centro Trial 2", "Northridge Trial 1

# Loop through all events
for event_name in event_names:
    # Load the X component of the bottom sensor as input
    inputs = data[f'Stiff 1 story Ground Floor - ({event_name})'][:,0]
    # Don't use the entire time series. Start halfway through the record and
    sixth_points = int(inputs.shape[0]/6)
    inputs = inputs[3*sixth_points:5*sixth_points]
    # Load the top sensor as output
    outputs = data[f'Stiff 1 story Top Floor - ({event_name})'][:,0][3*sixth
    # Construct the time array
    time = time_array(TIME_STEP, len(inputs))
    # Use the mdof package to perform system identification and determine th
    P, Phi = mdof.modes(inputs, outputs, dt=TIME_STEP)
    # The fundamental natural period is the longest period.
    realization = mdof.sysid(inputs, outputs)
    print(f"Event: {event_name} - Fundamental Period: {max(P):.3f} seconds")
```

```
Event: El Centro Trial 1 - Fundamental Period: 0.050 seconds
Event: El Centro Trial 2 - Fundamental Period: 0.049 seconds
Event: Northridge Trial 1 - Fundamental Period: 0.049 seconds
Event: Northridge Trial 2 - Fundamental Period: 0.049 seconds
Event: Sine Wave Trial 1 - Fundamental Period: 0.054 seconds
Event: Sine Wave Trial 2 - Fundamental Period: 0.050 seconds
```

(5) Fundamental periods of the three steel models

What is your estimate of the fundamental natural period, in seconds, of the 3 story stiff steel model? of the 3 story flexible steel model? of the 1 story stiff steel model?

Save these values as `period_stiff3`, `period_flex3`, and `period_flex1`.

In [9]:
```
period_stiff3 = np.mean([0.134,0.125,0.125,0.125,0.128]) # fundamental natur
period_flex3 = np.mean([0.273,0.288,0.267,0.270]) # fundamental natural peri
period_stiff1 = np.mean([0.050, 0.049, 0.049, 0.049, 0.054, 0.050]) # fundam

print(f"{period_stiff3=:.5f}")
print(f"{period_flex3=:.3f}")
print(f"{period_stiff1=:.3f}")
```

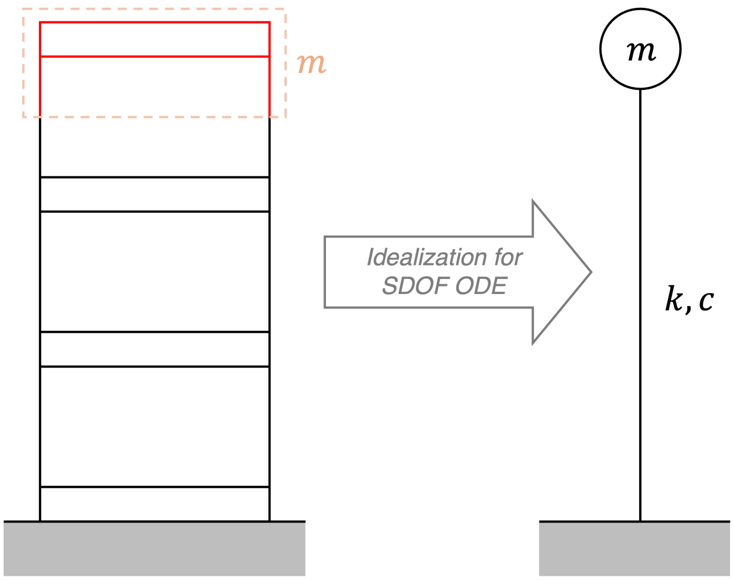
```
period_stiff3=0.12740
period_flex3=0.274
period_stiff1=0.050
```

(6) Stiffness of each type of steel

Remember the relations, $\omega_n = \sqrt{\frac{k}{m}}$ and $T_n = \frac{2\pi}{\omega_n}$. Note that T_n is the natural period.

1. Derive the relation $\omega_n = \sqrt{\frac{k}{m}}$ from the governing differential equation of motion for a single degree of freedom (undamped) harmonic oscillator in free vibration, $m\ddot{u} + ku = 0$.

2. Use the relations above, along with the mass of the model materials, to determine the stiffness of each model. Assume the mass is "lumped" at the top floor, and equal to the mass of one polycarbonate block with sensor attached plus a half story of steel wall. Note: the units of stiffness should be a unit of force per unit displacement, such as kg/cm.



Write your answers below, or in a separate document.

1. If $m\ddot{u} + ku = 0$, then $u(t) = c_1 \cos \omega_n t + c_2 \sin \omega_n t$.

How do we solve for ω_n in terms of the values of m and k?

[SOLUTION, OPTION A]:

Using the general method for solving a second-order, linear, constant-coefficient ODE:

- Form the characteristic polynomial: $mr^2 + kr = 0$.
- Solve the quadratic equation for r: $r = \pm\frac{\sqrt{-4mk}}{2m}$
- Simplify: $r = \pm i\sqrt{\frac{k}{m}}$
- There are two complex roots, $\alpha + i\beta$, which means that the solution is of the form $u(t) = c_1 e^{\alpha t} \cos \beta t + c_2 e^{\alpha t} \sin \beta t$.

- Therefore, $\boxed{\omega_n = \beta = \sqrt{\dfrac{k}{m}}}$.
- (P.S. note that $\alpha = 0$, so $e^{\alpha t} = 1$ and that term disappears from the solution.)

[SOLUTION, OPTION B]:

Since the solution, $u(t)$, of the differential equation is given, we can take its derivative and plug it back into the ODE:

- ODE solution (zeroth derivative): $u(t) = c_1 \cos \omega_n t + c_2 \sin \omega_n t$
- First derivative: $\dot{u}(t) = -\omega_n c_1 \sin \omega_n t + \omega_n c_2 \cos \omega_n t$
- Second derivative: $\ddot{u}(t) = -\omega_n^2 c_1 \cos \omega_n t - \omega_n^2 c_2 \sin \omega_n t$
- Plug back into ODE:
  $-m\omega_n^2 c_1 \cos \omega_n t - m\omega_n^2 c_2 \sin \omega_n t + kc_1 \cos \omega_n t + kc_2 \sin \omega_n t = 0$
- Rearrange: $\text{\red}(k - m\omega_n^2)(c_1 \cos \omega_n t + c_2 \sin \omega_n t) = 0$
- The term, $(c_1 \cos \omega_n t + c_2 \sin \omega_n t)$, can never be zero for all time t ($\cos \omega_n t$ and $\sin \omega_n t$ are *"independent"*). So, the above equation holds true for all time t if and only if $(k - m\omega_n^2) = 0$.
- Rearranging, $\boxed{\omega_n = \sqrt{\dfrac{k}{m}}}$.

2. Start by calculating the "lumped mass."

m = (mass of 1 polycarb block with sensor attached) + 0.5(mass of 6 inches (1 floor) of 2 steel walls).

Then compute k from the relations $\omega_n = \sqrt{\frac{k}{m}}$ and $T_n = \frac{2\pi}{\omega_n}$.

[SOLUTION]:

First, calculate the lumped mass.

- Mass of the block with sensor attached = 0.124kg.
- 3 Story Stiff model:
  - Weight of steel per unit length:
    $(0.8245\text{kg} - 4 * 0.124\text{kg})/(2 * 18\text{in}) = 0.00913\text{kg/in}$
  - Lumped weight: $w_{3\text{st}} = 0.124\text{kg} + 0.5(6\text{in} * 2 * 0.00913\text{kg/in}) = 0.179\text{kg}$
- 1 Story Stiff model:
  - Weight of steel per unit length:
    $(0.3589\text{kg} - 2 * 0.124\text{kg})/(2 * 6\text{in}) = 0.00924\text{kg/in}$
  - Lumped weight: $w_{1\text{st}} = 0.124\text{kg} + 0.5(6\text{in} * 2 * 0.00924\text{kg/in}) = 0.179\text{kg}$
- Flexible model:
  - Compute the mass of the 3 story flexible steel per unit length:
    $(0.6045\text{kg} - 4 * 0.124\text{kg})/(2 * 18\text{in}) = 0.00301\text{kg/in}$

- Lumped mass of the stiff steel models:
  $w_{3\text{fl}} = 0.1240\text{g} + 0.5(6\text{in} * 2 * 0.00301\text{g/in}) = 0.142\text{kg}$

Then compute k.

- Rearranging the two equations, $k = m(\frac{2\pi}{T})^2$.
- 3 Story Stiff Steel Stiffness: $k_{3\text{st}} = 0.179\text{kg} * \frac{1}{981\text{cm/s}^2}(\frac{2\pi}{0.127})^2 = \boxed{0.44\text{kg/cm}}$
- 1 Story Stiff Steel Stiffness: $k_{1\text{st}} = 0.179\text{kg} * \frac{1}{981\text{cm/s}^2}(\frac{2\pi}{0.050})^2 = \boxed{2.87\text{kg/cm}}$
- 3 Story Flexible Steel Stiffness: $k_{3\text{fl}} = 0.142\text{kg} * \frac{1}{981\text{cm/s}^2}(\frac{2\pi}{0.274})^2 = \boxed{0.08\text{kg/cm}}$

Note: due to rounding, the stiffnesses may be off by up to 0.01 kg/cm.

In [10]:
```
m_block = 0.1240 # kilograms

w_st3 = (0.8245-4*m_block)/(2*18) # kilograms per inch
print(f"Weight of 3 story stiff steel per unit length: {w_st3:.5f} kg/in")
w_st3_lumped = m_block + 0.5*(6*2*w_st3) # kilograms
print(f"Lumped weight of 3 story stiff steel model: {w_st3_lumped:.3f} kg")
k_st3 = (w_st3_lumped/981)*(2*np.pi/period_stiff3)**2 # kilograms per centim
print(f"Stiffness, k, of 3 story stiff steel model: {k_st3:.3f} kg/cm")

w_st1 = (0.3589-2*m_block)/(2*6) # kilograms per inch
print(f"\nWeight of 1 story stiff steel per unit length: {w_st1:.5f} kg/in")
w_st1_lumped = m_block + 0.5*(6*2*w_st1) # kilograms
print(f"Lumped weight of 1 story stiff steel model: {w_st1_lumped:.3f} kg")
k_st1 = (w_st1_lumped/981)*(2*np.pi/period_stiff1)**2 # kilograms per centim
print(f"Stiffness, k, of 1 story stiff steel model: {k_st1:.3f} kg/cm")

w_fl3 = (0.6045-4*m_block)/(2*18) # kilograms per inch
print(f"\nWeight of 3 story flexible steel per unit length: {w_fl3:.5f} kg/i
w_fl3_lumped = m_block + 0.5*(6*2*w_fl3) # kilograms
print(f"Lumped weight of 3 story flexible steel model: {w_fl3_lumped:.3f} kg
k_fl3 = (w_fl3_lumped/981)*(2*np.pi/period_flex3)**2 # kilograms per centime
print(f"Stiffness, k, of 3 story flexible steel model: {k_fl3:.3f} kg/cm")
```

```
Weight of 3 story stiff steel per unit length: 0.00913 kg/in
Lumped weight of 3 story stiff steel model: 0.179 kg
Stiffness, k, of 3 story stiff steel model: 0.443 kg/cm

Weight of 1 story stiff steel per unit length: 0.00924 kg/in
Lumped weight of 1 story stiff steel model: 0.179 kg
Stiffness, k, of 1 story stiff steel model: 2.869 kg/cm

Weight of 3 story flexible steel per unit length: 0.00301 kg/in
Lumped weight of 3 story flexible steel model: 0.142 kg
Stiffness, k, of 3 story flexible steel model: 0.076 kg/cm
```

In [ ]: