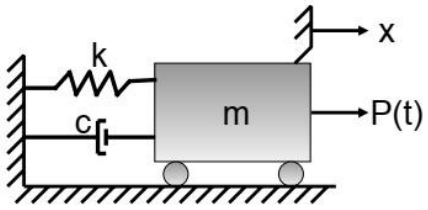
Topic: Signal Processing

Concepts: Fourier Transform, Fast-Fourier Transform

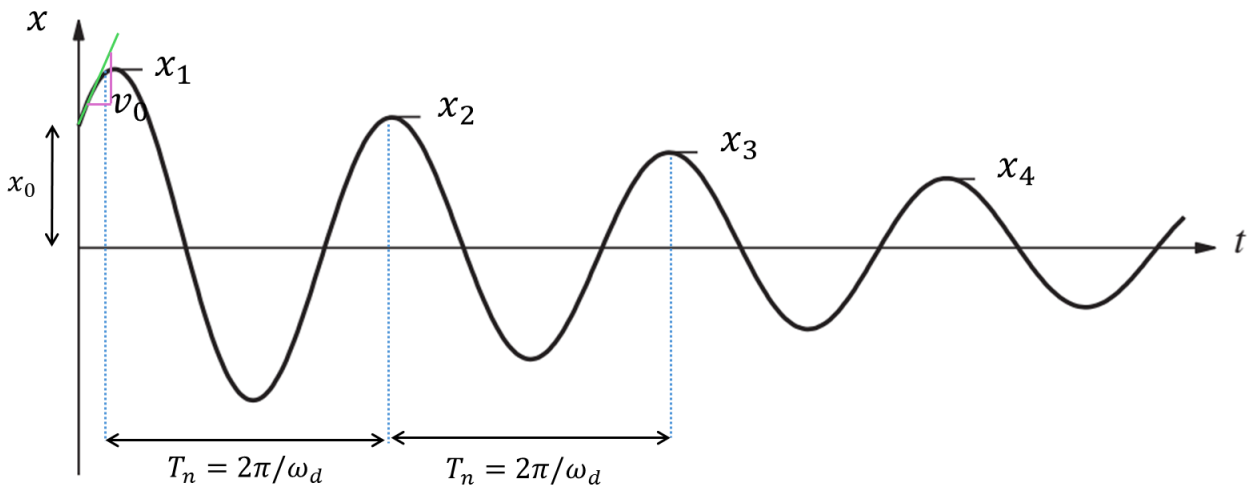
In this assignment, you will see how to use Fast Fourier Transform (FFT) to perform Fourier Transform of free-vibration response of a SDOF system. The SDOF system as we saw in the last assignment looks like:



This system is typically used to model response of a system with a force varying as a function of time. The system therefore also has a response varying with time. SDOF system has the following equation of motion:

$$m\ddot{x} + c\dot{x} + kx = P(t)$$

which describes how the response of the mass varies due to varying forcing. In this equation, m is the mass, k is the stiffness, $P(t)$ is applied force as a function of time and c is the viscous damping coefficient which represents the energy dissipation in the system. For the sake of this assignment, $P(t) = 0$. The system would vibrate only due to initial displacement, x_0 and velocity, v_0 imposed on the mass. The system response in this case can be determined by using the following parameters: $\omega_n = 2\pi f_n = \frac{2\pi}{T_n} = \sqrt{\frac{k}{m}}$ and $\zeta = \frac{c}{2\sqrt{mk}}$. The system response looks like following:



The natural frequency is somewhat easy to compute, $f_n = 1/T_n$, where T_n is the time taken for one cycle. The damping ratio can be calculated by using log decrement as:

$$\delta = \frac{1}{N}\log\left(\frac{x_n}{x_{n+N}}\right)$$

$$\zeta = \frac{\delta}{2\pi}$$

where, x_n is the n^{th} peak on displacement vs time, and x_{n+N} is the $(n+N)^{th}$ peak on displacement vs time.

You are performing an experiment to determine the natural frequency (f_n) and system damping (ζ) for a SDOF system. You recorded the data and now will analyze the data.

Task 1:

In your GitHub folder, you will find a file named "free_vibration.npz". This function contains two arrays: (1) displacement of mass $x(t)$ and (2) corresponding time, t. You will find these loaded into your Python program by using the following commands:

```
filename = "free_vibration.npz"

npzfile = np.load(filename)
```

Then the arrays are unpacked using the commands:

```
t = npzfile['t']                    # contains the time array

x = npzfile['x']                    # contains the displacement array
```

Plot the $x(t)$ vs t. Is this a signal? You will see the signal is very noisy. Clearly, using this signal for determining the natural frequency and damping ratio is not getting us good accuracy. This is how most signals look when you get them.

Task 2:

Our task now is to solve this problem. For this we will use filtering. Filtering involves removing frequency content at all the frequencies we do not care about. This involves (1) Taking Fourier Transform, (2) Setting Fourier Transform for these frequencies to zero and (3) Take inverse Fourier Transform. For this, we first need to understand what frequencies we care about. Filtering out without proper estimation of frequencies might result in losing relevant information too.

Therefore, first you will take Fourier Transform of our signal. This is done using an algorithm called *Fast Fourier Transform*, which performs Discrete Time Fourier Transform and samples the Fourier

Transform for us store and visualize frequency content. Perform FFT of $x(t)$ for the tasks. Noise typically is a high frequency phenomenon. Identify the problematic frequencies, which in this case a band of high frequencies.

Task 3:

Now, we are ready to remove unwanted noise and isolate requisite signal. Go ahead and add the code to remove unwanted frequencies. Plot the response without the noise. Hopefully, this looks closer to what we are used to!

Task 4:

Now we are ready to determine the natural frequency and damping ratio. Record time taken for one cycle and use it to determine natural frequency. Use the log decrement method described earlier to determine the damping ratio. Be careful when determining log decrement using suggested peak finder, $scipy.find\_peaks()$. Peak finders find local peaks and can sometimes find spurious and repeated peaks due to numerics. So, pick the peaks that make most sense. It is best to print the amplitudes picked by the peak finder and use the appropriate indices.

Task 5:

Let us say we do not want to store so much data (time and displacement arrays) and you decide to sub-sample data. You try two values, i.e. sub-sampling of 10 (i.e. you just retain one data point per 10 of your original time and displacement arrays) and 400. Fill in the code to perform this sub-sampling and look at their corresponding frequency content. Comment on which of the two is acceptable and why.