M1_Notebook

September 25, 2019

Engineering Tripos Part IB, Experiment M1
Experimental Engineering, Materials Teaching Laboratory

0.1 Introduction

In this experiment you will measure natural frequencies of vibrating beams after impulsive loading, and use these to calculate material properties. The frequency of free vibration of beams depends on the density (the inertia) and the stiffness: low density and high stiffness mean higher frequencies. The stiffness of beams in turn depends on the geometry (the length and cross-sectional shape) and the material (the Young's modulus). The decay rate of the vibrations also provides information on damping i.e. energy loss during vibration. Two materials will be considered: aluminium alloy and the polymer PMMA.

0.2 Method

Start by importing the necessary modules (pydvma is a python package written for data acquisition at CUED).

Remember: * to actually run a cell of code, click inside the cell then press 'shift+enter' * the cell is running while [*] is displayed * the cell has finished running when it changes to a number

[ ]:
```
%gui qt
```

Wait a second before running the imports: it takes a moment for the previous command to take effect behind the scenes. If you get an error, just try it again.

[ ]:
```
import matplotlib
import numpy as np
import pydvma as dvma
import m1
```

[ ]:
```
matplotlib.use('nbagg')
```

Choose your acquisition settings. The setting we want for the m1 lab are:

- channels=1 (number of channels to record)
- fs=8000 (sampling rate in Hz)
- stored_time=2 (time in seconds to record data for)

- viewed_time=2 (time in seconds to display on oscilloscope)
- device_index = 1 (Windows default input)

We want to use the default soundcard for the microphone, we only need one channel, and a lower sampling rate works well for this lab:

[ ]:
```
settings = dvma.MySettings(channels=1,
                           fs=8000,
                           stored_time=2,
                           viewed_time=2,
                           device_index=1)
```

Now open an oscilloscope using your settings. This shows three plots:

- the top one is like a normal oscilloscope showing the signal (toggle on/off with 'T');
- the middle one shows the frequency spectrum of the signal (toggle on/off with 'F');
- the bottom one shows the signal amplitudes (toggle on/off with 'L') - you probably want to turn this one off;

The oscilloscope window will always be on top: to turn that off make sure the oscilloscope window is selected then press 'A'.

Try the following:

- **Tap a tuning fork** on your knee or the table, hold it over the microphone, then watch the oscilloscope window. You should see a signal in the time-domain plot (top), and a sharp peak in the frequency-domain plot (middle) at the tuning fork's natural frequency.
- **Whistle at the microphone** and change pitch: you should see a peak which moves with the changing frequency.
- **Place the beam on the rubber band supports** (so it's suspended over the microphone) and tap the beam while watching oscilloscope window. You will see the signal pass through the time-domain view, and clear peaks in the frequency domain view.

[ ]:
```
osc = dvma.Oscilloscope(settings)
```

Press the **space bar** to record data from the past 'stored_time' seconds.

- The first time you press it you will be prompted for where to save your data.
- The default location should be the folder 'lab_M1' in your filespace on the teaching system.
- Subsequent times you press it will auto-save to the same folder with a number added to the filename.
- Press 's' if you want to save data to a new filename or location. Pressing space after that will auto-save with the new name.

If needed then you can restart this notebook using the 'kernel' menu and select 'restart kernel and clear output'. You can then run the import section above and start again from the top, or jump to the dvma.read_data() section below.

Note that pressing 'space' captures the past N seconds of data: so you need to tap the beam, wait a second or so, then press space!

Once you have a good example dataset you can read your data files and plot them.

[ ]:
```
dataset = dvma.load_data()
dataset
```

[ ]:
```
time_plot = dataset.plot_time_data()
```

Use the box-zoom tool to zoom in on the section of time where you can see an impact and its decay down to (nearly) zero. We will use this section of data to calculate its spectrum and find the resonant peaks of the beam.

[ ]:
```
dataset.calculate_fft_set(time_range=time_plot)
```

[ ]:
```
freq_plot = dataset.plot_freq_data()
```

Zoom in on the required resonant peaks in the frequency domain plot, then use 'find_peaks' from the m1 labs module to record their exact values.

Note that the peak finder is using the zoomed range of your 'freqplot'. It will only find peaks in this frequency range, and only those that are in view on the y-axis. So if you zoom in and chop off the noisy low-amplitude data then that can help isolate the most prominant peaks.

[ ]:
```
p,a = m1.find_peaks(dataset.freq_data_list[0],freq_range=freq_plot)
```

Next, we will be calculating the Young's Modulus of the beam.

First of all, measure the length **(L)**, width **(b)**, thickness **(d)** and mass **(M)** of your beam in millimeters and grams, and enter their values below.

[ ]:
```
### enter measured values
L = ...  # mm
b = ...  # mm
d = ...  # mm
M = ...  # g

### convert to m and kg (already completed) ###
L /= 1e3 # convert to kg
b /= 1e3 # convert to kg
d /= 1e3 # convert to kg
M /= 1e3 # convert to kg
```

Finally, use the code cell below to find the Young's Modulus of aluminium.

You need to enter: * the natural frequency in Hz of one of the modes of the beam * the mode number corresponding to that frequency (counting 1 as the first mode)

Now run the cell below for each mode and make sure your values agree with databook values. If any value is very far from what you expect then this is probably because the frequency and mode number are not matched correctly. Also check that the calculated density is reasonable.

E is calculated using the following equation:

$$(2\pi f)^2 \times \left(\frac{L}{\alpha_L}\right)^4 \times \rho \times \frac{A}{I}$$

Record the measured natural frequencies and calculated Young's modulus values on the Results Sheet

[ ]:
```
### Enter values for natural frequency and corresponding mode
frequency = ... # Hz
mode = ...      # mode number, counting 1 as first mode
```

```
### Don't change this bit ###
alphaL = np.array([4.730, 7.853, 10.996, 14.137]) # properties of a free-free␣
↪beam - don't change
alpha = alphaL[mode-1]/L # select alpha for chosen mode
##############################

### calculate properties
rho = M/(L*b*d) # density
A = b*d # cross-section area
I = (b*d**3)/12 # second moment of area
E = ((2*np.pi*frequency)**2)*((1/alpha)**4)*rho*A/I# Young's Modulus

### print results
print('The Young''s Modulus calculated from Mode {} is {:.2f} GPa'.
↪format(mode,E*1e-9))
print('Density = {:.2f} kgm^-3'.format(rho))
```

0.3 AT THE END OF THE EXPERIMENT…

- double check that your data files (*.npy) are stored in your teaching system file space (use the desktop shortcut to 'Home' → 'Documents' → 'lab_M1')
- share your data with lab-group partners (e.g. using Firefox Send or wetransfer)
- **sign out** from the PC.

0.4 … or if time allows: PMMA polymer beam

Repeat the testing procedure above using the PMMA beam (the code is copied below).

Record the measured natural frequencies and calculated Young's modulus values on the Results Sheet.

[ ]:
```
osc = dvma.Oscilloscope(settings)
```

How does the time series response for the PMMA beam compare with the aluminium case?

The most notable difference is the much higher damping for this material. Recall from IA linear vibrations that damping may be characterised by the damping factor ζ

This can be estimated from the time series data using the 'logarithmic decrement' approach (see the Mechanics Data Book, p.7) which compares the amplitudes of successive peaks (y_1 and y_2). Rearranging the Data Book formula gives:

$$\zeta = \left[1 + \left(\frac{2\pi}{\ln(y_1/y_2)}\right)^2\right]^{-\frac{1}{2}}$$

Apply this equation to successive pairs of peaks to estimate the damping factor, and note this on the Results Sheet.

[ ]:
```
dataset_pmma = dvma.load_data()
time_plot = dataset_pmma.plot_time_data()
```

[ ]:
```
y1 =
y2 =
zeta = ( 1 + (2*np.pi/np.log(y1/y2))**2 )**(-1/2)
print(zeta)
```

As discussed in IA linear vibrations, damping also alters the modal frequencies. The damped frequencies (ω_d) are related to the undamped frequencies (ω_n) via the equation in the Data Book):

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}$$

We neglected this correction in the calculation of Young's modulus: was this a significant omission?

[ ]:
```

```

0.5 AT THE END OF THE EXPERIMENT

- double check that your data files (*.npy) are stored in your teaching system file space (use the desktop shortcut to 'Home' $\rightarrow$ 'Documents' $\rightarrow$ 'lab_M1')
- share your data with lab-group partners (e.g. using Firefox Send or wetransfer)
- **sign out** from the PC.

[ ]:
```

```