Data Analysis of Recordings for a Single Oscillator (C1) (Section 4)

1 Resonance Frequency of a Single Lever

[205]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.random import randn
from scipy.stats import norm
#from scipy import misc
import glob
from imageio import imread
import os

from skimage.feature import blob_dog, blob_log, blob_doh
from skimage.morphology import erosion
from skimage.morphology import disk
from skimage.filters import median
import glob
from scipy.optimize import curve_fit

from plotly.offline import download_plotlyjs, init_notebook_mode, iplot
#import plotly.graph_objs as go
from plotly.graph_objs import *
init_notebook_mode(connected=True)

# inline plotting
%matplotlib inline

# this is just for Macbooks to create higher resolution previews of the plots in␣
 ↪the notebook
%config InlineBackend.figure_format = 'retina'
import av
# the lines below set a number of parameters for plotting, such as label font␣
 ↪size,
# title font size, which you may find useful
plt.rcParams.update({'font.size': 16,
```

```
                    'font.family': 'serif',
                    'axes.titlesize': 16,
                    'axes.labelsize': 20,
                    'axes.labelpad': 14,
                    'lines.linewidth': 1,
                    'lines.markersize': 10,
                    'xtick.labelsize' : 18,
                    'ytick.labelsize' : 18,
                    'xtick.top' : True,
                    'xtick.direction' : 'in',
                    'ytick.right' : True,
                    'ytick.direction' : 'in',})
```

[156]:
```
folders=np.sort(glob.glob('* Hz'))
```

[172]:
```
freq=[]
gn=[]
for folder in folders:
    f=open(folder+'/gain.txt')
    try:
        gain=float(f.read())
    except:
        print(folder)
    freq.append(float(folder[:4]))
    gn.append(gain)
```

[173]:
```
plt.plot(freq,gn,'.')
plt.plot(freq,gn)
plt.xlabel('frequency [Hz]')
plt.ylabel('gain [Hz]')
#plt.xlim(0.9,2.1)
plt.axhline(y=1,ls='--')
plt.show()
```

[186]:
```
def amp(x, *p):
    k,a=np.array(p)
    m=0.0073
    return(a*np.abs(k/(k-m*(x*2*np.pi)**2)))
```

[187]:
```
frequency=np.linspace(0.0,4.0,200)
plt.plot(frequency,amp(frequency, 2.344, 0.0073))
plt.show()
```

[188]:
```
p=np.zeros(2)
p[0]=2.5
#p[1]=0.0073
```

```
p[1]=0.39
pfit,varmatrix =curve_fit(amp, freq, gn, p0=p)
#freq= np.array(freq)
#freq_omega= freq * 2 * np.pi
```

[208]:
```
frequency=np.linspace(2.0,4.0,200)
plt.figure(figsize=(8,6))
plt.plot(frequency,amp(frequency, *p))
plt.plot(freq,gn,'bo',alpha=0.4)
plt.ylim(0,30)
plt.xlabel('Frequency $f$ [$Hz$]')
plt.ylabel('Amplitude ratio $X_m/X_{M_0}$')
plt.tight_layout()
plt.savefig('condition3_amplitude_ratio.pdf')
plt.show()
```