Data Analysis of Recordings for a Chain of Coupled Oscillators with an Internal Degree of Freedom (C3) (Section 4)

1 Amplitude of the Oscillation as a Function of Frequency

[14]:
```
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',})
```

[3]:
```
folders=np.sort(glob.glob('* HZ'))
```

[4]:
```
freq=[]
gn=[]
data=[]
for folder in folders:
    print(folder)
    f=open(folder+'/gain.txt')
    gain=float(f.read())
    freq.append(float(folder[:4]))
    gn.append(gain)
    df=pd.read_csv(folder+'/Results.csv')
    df['z']=ff=np.ones(df.shape[0])*float(folder[:4])
    data.append(df)
```

[50]:
```
fig=plt.figure(figsize=(6,8))
plt.plot(gn, freq, 'bo',alpha=0.2)
plt.plot(gn,freq)
plt.xlabel('Amplitude ratio $X_{20}/X_2$')
plt.ylabel('Frequency $f$ [$Hz$]')
plt.ylim(-0.05, 4.5)
plt.xlim(-0.2, 1.1)
plt.tight_layout()
plt.savefig('condition2_amplitude_ratio.pdf')
plt.show()
```

[8]:
```
traces=[]
for df in data:
    xpos=[]
    ypos=[]
    for i in range(len(bins)-1):
        dd=df.x[(df.x<bins[i+1]) & (df.x>bins[i])]
```

```
        xpos.append(np.mean(dd.values))
        dd=df.y[(df.x<bins[i+1]) & (df.x>bins[i])]
        ypos.append(np.max(dd.values)-np.mean(dd.values))
    zpos=np.ones(len(xpos))*df.z[0]
    trace = Scatter3d(
        x=xpos,
        y=zpos,
        z=ypos,
        mode='lines',
        marker=dict(
            size=2,
    #        line=dict(
    #            color='rgba(1, 1, 1,0)',
    #            width=1
    #        ),
            opacity=1
        )
    )
    traces.append(trace)

dat = traces
layout = Layout(
    margin=dict(
        l=0,
        r=0,
        b=0,
        t=0
    )
)
fig = Figure(data=dat, layout=layout)
iplot(fig, filename='simple-3d-scatter')
```

[7]:
```
bins=[0,60,110,170,220,270,330,390,450,510,560,620,680,740,800,860,930,990,1050
,1110,1170,1240]
```

[9]:
```
xpos=[]
ypos=[]
for i in range(len(bins)-1):
    dd=df.x[(df.x<bins[i+1]) & (df.x>bins[i])]
    xpos.append(np.mean(dd.values))
    dd=df.y[(df.x<bins[i+1]) & (df.x>bins[i])]
    ypos.append(np.max(dd.values)-np.mean(dd.values))
```

[10]:
```
z=np.zeros([21,27])

for j,df in enumerate(data):
    xpos=[]
```

```
    ypos=[]
    for i in range(len(bins)-1):
        dd=df.x[(df.x<bins[i+1]) & (df.x>bins[i])]
        xpos.append(np.mean(dd.values))
        dd=df.y[(df.x<bins[i+1]) & (df.x>bins[i])]
        ypos.append(np.max(dd.values)-np.mean(dd.values))
        z[i,j]=np.max(dd.values)-np.mean(dd.values)
    zpos=np.ones(len(xpos))*df.z[0]
```

[11]:
```
colorscale = [[0, 'rgb(0,0,255)'], [1, 'rgb(255,255,0)']]
fig = Figure(data=[Surface(z=z,cmin=0,cmax=50,colorscale=colorscale)])

fig.update_layout(title='Mt Bruno Elevation', autosize=True,
                  width=800, height=800,
                  margin=dict(l=5, r=0, b=0, t=0))

iplot(fig, filename='simple-3d-scatter')
```

1.1 Dispersion relation

[6]:
```
def m_eff(omega):
    m0=2*0.0073
    m=0.0073
    omega_0=np.sqrt(2.477/m)
    meff=m0+m*omega_0**2/(omega_0**2-omega**2)

    return(meff)
```

[7]:
```
def dispersion(omega):
    a=0.06
    K=2.477/2
    meff=m_eff(omega)
    aa=np.sqrt(meff*omega**2/4/K)
    #print(aa)
    aa1=np.where(np.isnan(aa),0,aa)
    q=np.where(aa<0.9999,aa1,0)
    qq=np.arcsin(q)*2/a
    return(qq)
```

[8]:
```
om=np.linspace(0,30,1000)
meff=m_eff(om)
```

[9]:
```
fig=plt.figure(figsize=(8,5))
ax=fig.gca()
m=1
```

```
plt.plot(om/np.pi/2,meff/m)
plt.xlabel('Frequency f [Hz]')
plt.ylabel(r'M$_{\rm eff}$ [kg]')
plt.axhline(y=0,ls='--',color='k')
#plt.axvline(x=,ls='--',color='k')
plt.ylim(-0.1,0.1)
plt.xlim(1.9,4.1)
ax.fill_between(om/np.pi/2, 0, meff/m, where=meff< 0,
                facecolor='red', alpha=0.2)
ax.fill_between(om/np.pi/2, 0, meff/m, where=meff> 0,
                facecolor='blue', alpha=0.2)

plt.tight_layout()
plt.savefig('effective_mass.pdf')
plt.show()
```

[10]:
```
q=dispersion(om)
qm=q*0.06/np.pi
omm=om/2/np.pi
```

[11]:
```
plt.plot(np.where(qm==0,np.nan,qm),np.where(qm==0,np.nan,omm))
```

[86]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, sharey=True, gridspec_kw={'wspace': 0},␣
 ↪figsize=(12,8))

#fig, (ax1, ax2)= .figure(figsize=(6,8))

fr=[1.8,1.9,2.0,3.7,3.8,3.9,4.0]
kk=np.array([0.3,0.33,0.47,0.18,0.25,0.28,0.35])*6/np.pi
plt.ylim(-0.05,4.5)
plt.xlim(-0.02,1.02)
ax1.set_xlabel('Wave number $q$ $[a/\pi]$')
ax1.set_ylabel('Frequency $f$ [$Hz$]')
ax1.plot(kk,fr,'o')
ax1.axhline(y=2.07,ls='--',color='k')
ax1.axhline(y=3.595,ls='--',color='k')
ax1.plot(np.where(qm==0,np.nan,qm),np.where(qm==0,np.nan,omm))

ax2.plot(gn, freq, 'bo',alpha=0.2)
ax2.plot(gn,freq)
ax2.set_xlabel('Amplitude ratio $X_{20}/X_2$')
plt.ylim(-0.05, 4.5)
plt.xlim(-0.2, 1.1)
```