HANDS ON DAY 5: TIME SERIES

Farah Najla - IVC18 IVC on Astrostatistics and Machine learning 2021

In [1]:
```
# Importing necessary modules
import numpy as np
import pandas as pd
import pylab
import matplotlib.pyplot as plt
```

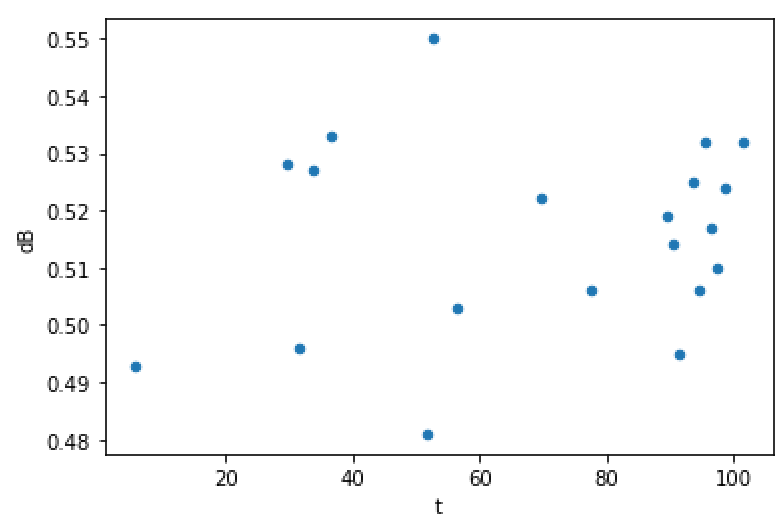
In [2]:
```
df = pd.read_table("dtvir.data", sep="\s+", names = ['t','dB'])
```

In [3]:
```
dB_mean = df['dB'].mean()
```

In [4]:
```
#f is the difference between each dB and the mean value of dB
df['f'] = df['dB'] - dB_mean
```

In [5]:
```
df.plot(x ='t', y='dB', kind = 'scatter')
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4e6a2b9d0>
```



In [6]:
```
# Nyquist frequency (nq_omega) = twice minimum spacing between two observasion
nq_omega = 1/(np.min(np.diff(df['t']))*2)
nq_omega
```

Out[6]:
```
0.519210799584627
```

In [7]:
```
# Trial frequency (omega)
N = len(df['t'])
t = df['t']
omega = np.arange(0.0001, nq_omega, 0.001)
dt = df['t'][N-1] - df['t'][0]
#do = omega[-1] - omega[0]
```

In [8]:
```
len(omega)
```

Out[8]:
```
520
```

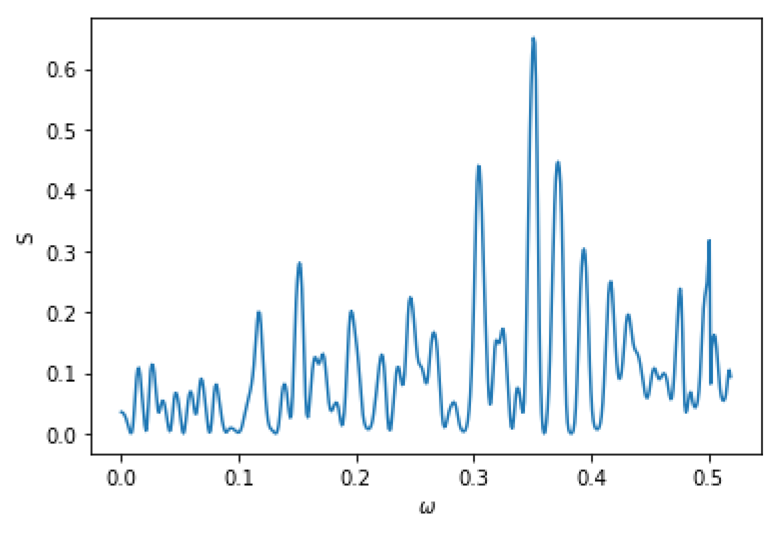
In [9]:
```
for i in range(N):
    do = omega[i]-omega[i-1]
```

In [10]:
```
# Defining functions
def x(t,omega):
    return 2*np.pi*omega*t

def cos_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += np.cos(x(t[i],omega))
    return test
def sin_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += np.sin(x(t[i],omega))
    return test
def cos_squared_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += (np.cos(x(t[i],omega)))**2
    return test
def sin_squared_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += (np.sin(x(t[i],omega)))**2
    return test
def f_cos_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += df['f'][i]*np.cos(x(t[i],omega))
    return test
def f_sin_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += df['f'][i]*np.sin(x(t[i],omega))
    return test
def cos_sin_sum(N,t,omega):
    test = 0
    for i in range(N):
        test += (np.sin(x(t[i],omega)))*(np.cos(x(t[i],omega)))
    return test
def f_sum(N):
    test = 0
    for i in range(N):
        test += df['f'][i]
    return test
def f_squared_sum(N):
    test = 0
    for i in range(N):
        test += (df['f'][i])**2
    return test
```

In [11]:
```
a0 = 1/(np.sqrt(N))
M = cos_sin_sum(N,t,omega) - (a0**2)*sin_sum(N,t,omega)*cos_sum(N,t,omega)
a1=1/(np.sqrt((cos_squared_sum(N,t,omega))-((a0**2)*(cos_sum(N,t,omega))**2)))
a2=1/(np.sqrt((sin_squared_sum(N,t,omega))-((a0**2)*(sin_sum(N,t,omega))**2)-((a1**2)*(M**2))))
c1=a1*f_cos_sum(N,t,omega)
c2=(a2*f_sin_sum(N,t,omega))-(a1*a2*c1*M)
S= ((c1**2)+(c2**2))/((f_squared_sum(N)))
G = - ((N - 3) / 2) * np.log(1 - S)
H = ((N-4)/(N-3))*(G + np.exp(-G) - 1)
alpha = (2*(N-3))*dt*do/(3*(N-4))
C = (1 - np.exp(-H))**alpha
```

In [12]:
```
plt.plot(omega,S)
plt.xlabel(r'$\omega$')
plt.ylabel('S')
plt.show()
```

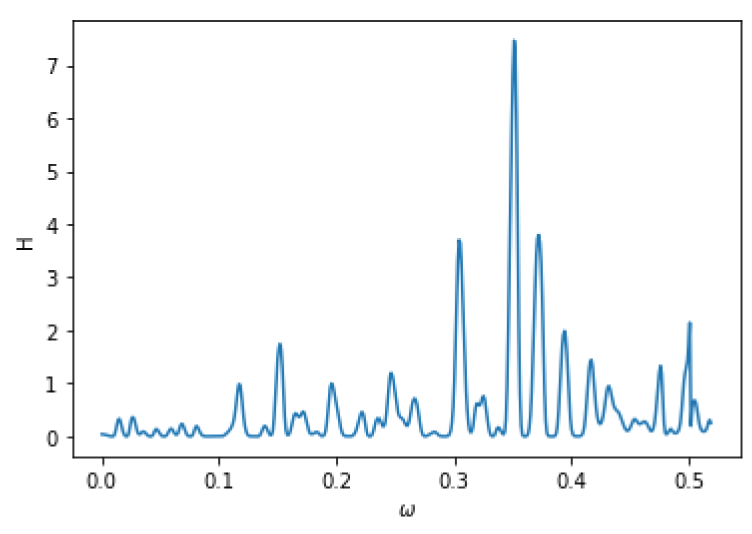


In [13]:
```
# omega_Smax is the value of omega at which the value of S is at the maximum
Smax = np.amax(S)
omega_Smax = omega[np.argmax(S)]
# Getting the confidence level for omega_Smax
Gmax = - ((N - 3) / 2) * np.log(1 - Smax)
Hmax = ((N-4)/(N-3))*(Gmax + np.exp(-Gmax) - 1)
Cmax = (1 - np.exp(-Hmax))**alpha
print("Confidence level:", Cmax)
print("The probability of having the highest peak by chance only:", 1-Cmax)
```

```
Confidence level: 0.9999611854940154
The probability of having the highest peak by chance only: 3.88145059846412e-05
```

In [14]:
```
# Period = 1/omega_Smax
P = 1/omega_Smax
Phase_abs = np.abs(df['t'] - df['t'][0])/P
Phase = np.mod(Phase_abs,1)
df['Phase'] = Phase
```

In [15]:
```
plt.plot(omega,H)
plt.xlabel(r'$\omega$')
plt.ylabel('H')
plt.show()
```



In [16]:
```
# The current data with the addition of f and Phase
df
```

Out[16]:

	t	dB	f	Phase
0	5.682	0.493	-0.02265	0.000000
1	29.630	0.528	0.01235	0.408143
2	31.664	0.496	-0.01965	0.122280
3	33.651	0.527	0.01135	0.819916
4	36.671	0.533	0.01735	0.880238
5	51.596	0.481	-0.03465	0.120405
6	52.583	0.550	0.03435	0.466941
7	56.511	0.503	-0.01265	0.846062
8	69.518	0.522	0.00635	0.412820
9	77.499	0.506	-0.00965	0.214949
10	89.477	0.519	0.00335	0.420424
11	90.458	0.514	-0.00165	0.764854
12	91.503	0.495	-0.02065	0.131753
13	93.468	0.525	0.00935	0.821665
14	94.464	0.506	-0.00965	0.171360
15	95.443	0.532	0.01635	0.515087
16	96.461	0.517	0.00135	0.872507
17	97.479	0.510	-0.00565	0.229927
18	98.442	0.524	0.00835	0.568036
19	101.469	0.532	0.01635	0.630816

In [20]:
```
# Plotting the light curve of DT Vir
plt.figure(figsize=(8,3))
plt.scatter(Phase,df['dB'],color='k')
plt.ylim(0.3,0.7)
plt.gca().invert_yaxis()
plt.xlabel('Phase')
plt.ylabel('$\Delta$B')
plt.title('Light Curve of DT Vir')
plt.show()
```