In [1]:
```python
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

Densité spectrale du bruit d'une machine à laver

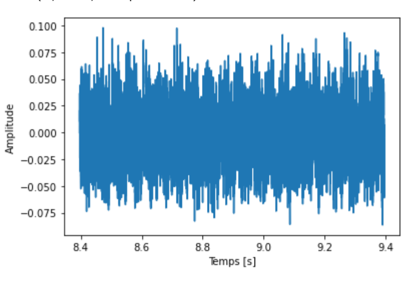
In [2]:
```python
from scipy.io.wavfile import read
samplerate, amplitude = read('data/machine_a_laver.wav')
```

In [3]:
```python
print(samplerate) # Number of points per second
delta_t = 1/samplerate # Time between two sample
N = len(amplitude) # Nombre de points
print(N)
T = N*delta_t #Temps total d'enregistrement
print(T)
```

```
44100
414300
9.394557823129252
```

In [4]:
```python
t = np.arange(N)*delta_t # Time vector

plt.plot(t[-samplerate:], amplitude[-samplerate:])#Plot the last 1 second of recording
plt.xlabel('Temps [s]')
plt.ylabel('Amplitude')
```

Out[4]: Text(0, 0.5, 'Amplitude')

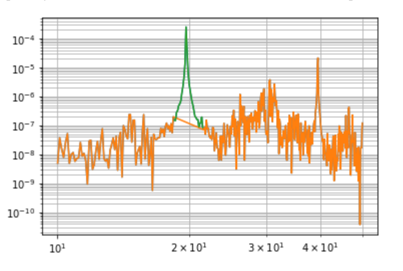


In [ ]:

In [5]:
```python
from scipy.signal import periodogram
freq, psd = periodogram(amplitude, samplerate) # Calculate the PSD (Power Spectral Density)
```

In [6]:
```python
mask = (freq>=10) & (freq<=500) # Frequencies between 10 and 500Hz
plt.loglog(freq[mask], psd[mask])
plt.grid(which='both')

mask_pic = np.abs(freq-20)<1.5 # Frequencies around the peak of signal at 20 Hz (20 +/- 1.5 Hz)

plt.loglog(freq[mask & ~mask_pic], psd[mask & ~mask_pic])
plt.loglog(freq[mask_pic], psd[mask_pic])
```

Out[6]: [<matplotlib.lines.Line2D at [illegible]>]



In [7]:
```python
print(np.var(amplitude)) # Mean power of the recording (Signal + noise) calculated on the time data

Delta_f = 1/T
print(np.sum(psd)*Delta_f) # Mean power of the recording (Signal + noise) calculated from the PSD (should be the same)
```

```
[illegible]
[illegible]
```

In [8]:
```python
print('Signal', np.sum(psd[mask_pic])*Delta_f) # Mean power of the signal
print('Bruit', np.sum(psd[mask & ~mask_pic])*Delta_f) # Mean power of the noise = signal
```

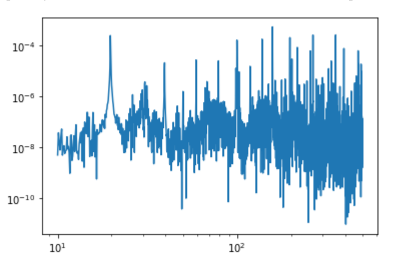
```
Signal [illegible]
Bruit [illegible]
```

In [9]:
```python
Delta_f = freq[1] - freq[0]
print(Delta_f, 1/T) # Check that the frequency resolution is the inverse of the recording time
```

```
[illegible]
```

In [10]:
```python
mask = (freq>=10) & (freq<=500)
plt.loglog(freq[mask], psd[mask]) # Plot on a larger frequency scale.
```

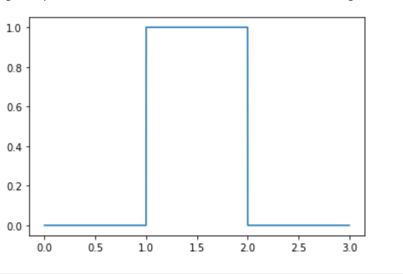
Out[10]: [<matplotlib.lines.Line2D at [illegible]>]



Filtre en Python

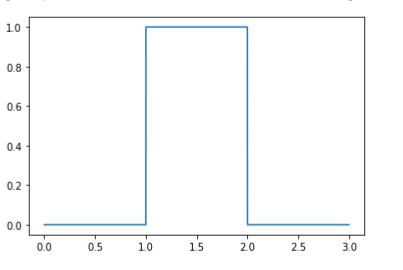
In [11]:
```python
samplerate = 44100
delta_t = 1/samplerate
signal = np.zeros(samplerate*3)
signal[samplerate:samplerate*2] = 1 # 1 s temporal rectangular function
t = np.arange(len(signal))*delta_t
plt.plot(t, signal)
```

Out[11]: [<matplotlib.lines.Line2D at [illegible]>]



In [12]:
```python
import numpy as np
signal_tilde = np.fft.rfft(signal) # Fourier transform of the rectangular function
signal_2 = np.fft.irfft(signal_tilde) # Inverse Fourier transform of Fourier transform = rectangular function
plt.plot(t, signal_2)
```

Out[12]: [<matplotlib.lines.Line2D at [illegible]>]



In [13]:
```python
np.fft.rfftfreq(len(signal), d=delta_t) # Frequency scale from the temporal characteristics of the rectangular
# 1/T <= frequency <= samplerate/2
```

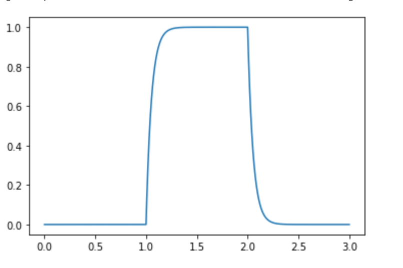
Out[13]: array([0.00000000e+00, 3.33333333e-01, 6.66666667e-01, ..., 2.20493333e+04, 2.20496667e+04, 2.20500000e+04])

In [14]:
```python
def pass_bas(signal, f_c, samplerate=44100):
    signal_tilde = np.fft.rfft(signal) # Fourier transform
    freqs = np.fft.rfftfreq(len(signal), 1/samplerate) # Frequency scale
    H = 1/(1+1j*freqs/f_c) # Transfer function of the low pass filter
    signal_2 = np.fft.irfft(H*signal_tilde) # Filtered temporal signal
    return signal_2
```

In [15]:
```python
def pass_haut(signal, f_c, samplerate=44100):
    signal_tilde = np.fft.rfft(signal) # Fourier transform
    freqs = np.fft.rfftfreq(len(signal), 1/samplerate) # Frequency scale
    H = 1j*(freqs/f_c)/(1+1j*freqs/f_c) # Transfer function of the high pass filter
    signal_2 = np.fft.irfft(H*signal_tilde) # Filtered temporal signal
    return signal_2
```

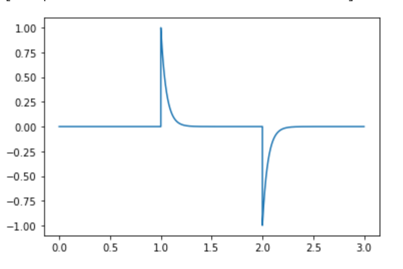
In [16]:
```python
signal_filtre = pass_bas(signal, f_c=5) # Low pas filtered rectangular function, cutoff at 5Hz
plt.plot(t, signal_filtre)
```

Out[16]: [<matplotlib.lines.Line2D at [illegible]>]



In [17]:
```python
signal_filtre = pass_haut(signal, f_c=5) # High pas filtered rectangular function, cutoff at 5Hz
plt.plot(t, signal_filtre)
```

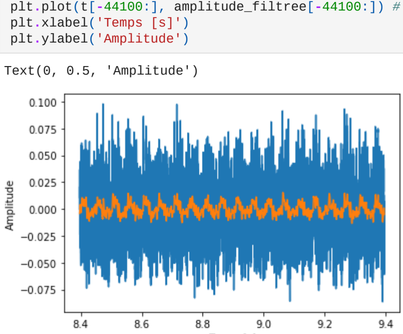
Out[17]: [<matplotlib.lines.Line2D at [illegible]>]



In [18]:
```python
from scipy.io.wavfile import read
samplerate, amplitude = read('data/machine_a_laver.wav')

amplitude_filtre = pass_haut(pass_bas(amplitude, 30, samplerate=samplerate),10, samplerate=samplerate) #High p
```

In [19]:
```python
t = np.arange(N)*delta_t

plt.plot(t[-44100:], amplitude[-44100:])
plt.plot(t[-44100:], amplitude_filtre[-44100:]) # we can see the temporal washing machine signal at 20Hz after
plt.xlabel('Temps [s]')
plt.ylabel('Amplitude')
```

Out[19]: Text(0, 0.5, 'Amplitude')



Onde gravitationnelle

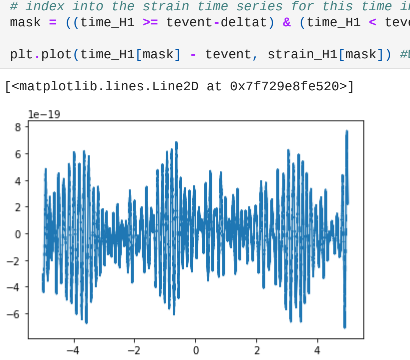
In [20]:
```python
from readligo import loaddata
fileName_H1 = 'data/H-H1_LOSC_4_V1-1126259446-32.hdf5'
strain_H1, time_H1, chan_dict_H1 = loaddata(fileName_H1, 'H1')
```

In [21]:
```python
from readligo import loaddata
fileName_H1 = 'data/H-H1_LOSC_4_V1-1126259446-32.hdf5'
strain_H1, time_H1, chan_dict_H1 = loaddata(fileName_H1, 'H1')#Temporal signal from 1st detector

fileName_L1 = 'data/L-L1_LOSC_4_V1-1126259446-32.hdf5'
strain_L1, time_L1, chan_dict_L1 = loaddata(fileName_L1, 'L1')#Temporal signal from 2nd detector

time = time_H1 # Time
print(len(time))
dt = time[1] - time[0]
print(dt)
samplerate = int(1/dt)
```

```
131072
0.000244140625
```

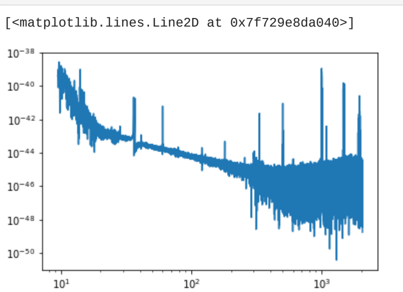
In [26]:
```python
# plot +- 5 seconds around the event:
tevent = 1126259462         # Mon Sep 14 09:50:45 GMT 2015
deltat = 5             # seconds around the event

# index into the strain time series for this time interval:
mask = ((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

# index into the strain time series for this time interval:
mask = ((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

plt.plot(time_H1[mask] - tevent, strain_H1[mask]) #We see that we have parasite oscillations  and noise, we car
```

Out[26]: [<matplotlib.lines.Line2D at 0x7f729e8fe520>]



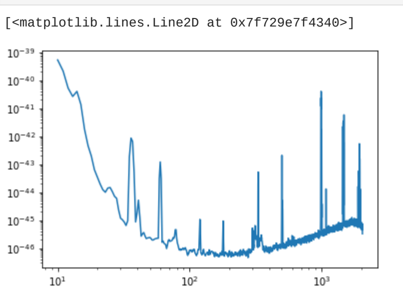
In [27]:
```python
from scipy.signal import periodogram
f_p, psd_p = periodogram(strain_H1, samplerate)

plt.loglog(f_p[300:], psd_p[300:]) # PSD of the measured signal, we see noise and parasite signals as sharp pea
```

Out[27]: [<matplotlib.lines.Line2D at 0x7f729e8da040>]



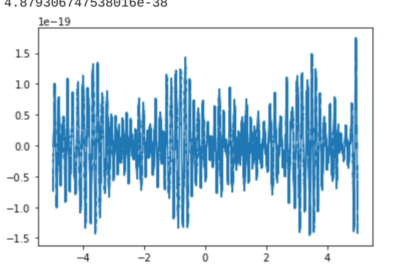
In [28]:
```python
from scipy.signal import welch
f, psd = welch(strain_H1, samplerate, nperseg=samplerate)

plt.loglog(f[10:], psd[10:]) #Average PSD thanks to the welch method, less noisy because of averaging but we lo
```

Out[28]: [<matplotlib.lines.Line2D at 0x7f729e7f4340>]



In [29]:
```python
print(psd[f==100])

strain_H1_filt = pass_haut(pass_bas(strain_H1, f_c=300, samplerate=1/dt), f_c=30, samplerate=1/dt) # Filtered s

# plot +- 5 seconds around the event:
tevent = 1126259462      # Mon Sep 14 09:50:45 GMT 2015
deltat = 5               # seconds around the event

# index into the strain time series for this time interval:
mask = ((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

plt.plot(time_H1[mask] - tevent, strain_H1_filt[mask]) # The signal from the gravitational wave is still not vi

# Variance du signal filtré et non filtré
print(np.var(strain_H1_filt)) #The variance is a bit reduced after filtering
print(np.var(strain_H1))
```

```
[8.51792826e-47]
2.2741360676158992e-39
4.879306747538016e-38
```

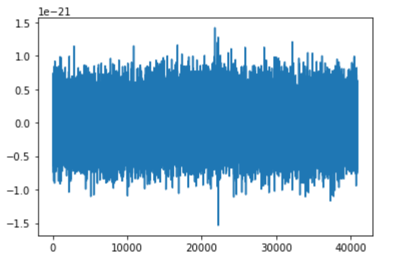


In [30]:
```python
from scipy.interpolate import interp1d

def whiten(strain, dt): # Filter that remove the parasite oscillations
    freqs_welch, psd_welch = welch(strain, fs=1/dt, nperseg=int(1/dt))
    interp_psd = interp1d(freqs_welch, psd_welch)

    strain_tilde = np.fft.rfft(strain)
    N = len(strain)
    freqs = np.fft.rfftfreq(N, dt)

    gain = 1 / np.sqrt(interp_psd(freqs))
    gain = gain/gain.max()
    white_strain_tilde = strain_tilde * gain
    white_strain = np.fft.irfft(white_strain_tilde)
    return white_strain
```

In [31]:
```python
strain_H1_whiten = whiten(strain_H1, dt)
strain_L1_whiten = whiten(strain_L1, dt)
```

In [32]:
```python
# plot +- 5 seconds around the event:
tevent = 1126259462         # Mon Sep 14 09:50:45 GMT 2015
deltat = 5.                    # seconds around the event

# index into the strain time series for this time interval:
mask = ((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

plt.plot(strain_H1_whiten[mask]) #Signal without the parasite oscillations! We still cannot see the gravitation

# Variance du signal sans les oscillations parasites
print(np.var(strain_H1_whiten)) #The variance is considerably reduced because we removed the parasite oscillati
```

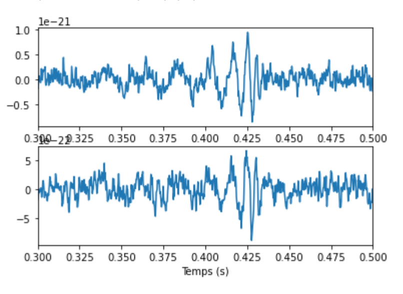
```
6.375591787640518e-43
```



In [33]:
```python
#We apply high and low pass filter on the signal above for which parasite oscillations has been removed
strain_H1_whiten_filt = pass_haut(pass_bas(strain_H1_whiten, f_c=300, samplerate=1/dt), f_c=20, samplerate=1/dt
strain_L1_whiten_filt = pass_haut(pass_bas(strain_L1_whiten, f_c=300, samplerate=1/dt), f_c=20, samplerate=1/dt
```

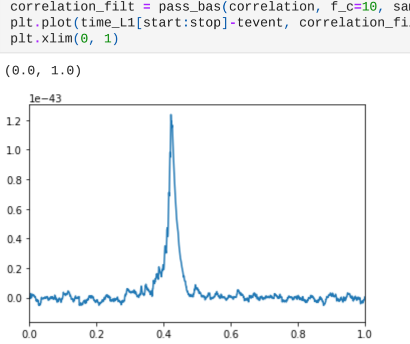
In [34]:
```python
# plot +- 0.5 seconds around the event:
tevent = 1126259462         # Mon Sep 14 09:50:45 GMT 2015
deltat = 0.5                # seconds around the event

# index into the strain time series for this time interval:
mask = ((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

plt.subplot(2, 1, 1)

plt.plot(time_H1[mask] - tevent, strain_H1_whiten_filt[mask]) #We can see now the gravitational wave signal!
plt.xlim(0.3, 0.5)
plt.subplot(2, 1, 2)

delay=0.007
plt.plot(time_L1[mask] - tevent + delay, -strain_L1_whiten_filt[mask])
plt.xlim(0.3, 0.5)

plt.xlabel('Temps (s)')
```

Out[34]: Text(0.5, 0, 'Temps (s)')



In [35]:
```python
## Question 8
# Correlation avec un délai de 7ms

start = int(len(time)/2 - 2.5*samplerate)
stop = int(len(time)/2 + 2.5*samplerate)
delai = int(7E-3*samplerate)
correlation = -strain_L1_whiten_filt[start:stop] * strain_H1_whiten_filt[start+delai:stop+delai]
correlation_filt = pass_bas(correlation, f_c=10, samplerate=samplerate)
plt.plot(time_L1[start:stop]-tevent, correlation_filt)
plt.xlim(0, 1)
```

Out[35]: (0.0, 1.0)



In [ ]: