GW Data NEW

December 1, 2025

1 Downloading O4a Gravitational-Wave Network Dataset with the GW231123 PUT LABELS

We are extracting the data from the GW230529 merger from the O4a network, the first phase of the fourth observing run of the LVK GW network.

[18]:
```
import requests


def fetch_strain_list(run, detector, gps_start, gps_end):
    "Return the list of strain file info for `run` and `detector`."

    # Get the strain list
    fetch_url = (
        f"https://gwosc.org/archive/links/"
        f"{run}/{detector}/{gps_start}/{gps_end}/json/"
    )
    response = requests.get(fetch_url)
    response.raise_for_status()
    return response.json()["strain"]
```

[2]:
```
! pip install h5py
```

```
Collecting h5py
  Using cached
h5py-3.15.1-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata
(3.0 kB)
Requirement already satisfied: numpy>=1.21.2 in /srv/conda/lib/python3.11/site-
packages (from h5py) (2.2.6)
Using cached
h5py-3.15.1-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (4.7 MB)
Installing collected packages: h5py
Successfully installed h5py-3.15.1
```

[19]:
```
strain_files = fetch_strain_list("O4a_4KHZ_R1", "H1", 1382832018, 1385337618)
print(f"Found {len(strain_files)} files")
print(strain_files[0:5])
```

```
Found 932 files
[{'GPSstart': 1382891520, 'UTCstart': '2023-11-01T16:31:42', 'detector': 'H1',
'sampling_rate': 4096, 'duration': 4096, 'format': 'hdf5', 'url': 'https://gwosc
.org/archive/data/O4a_4KHZ_R1/1382023168/H-H1_GWOSC_O4a_4KHZ_R1-1382891520-
4096.hdf5', 'min_strain': -5.372688122354986e-18, 'max_strain':
5.362402307740685e-18, 'mean_strain': 1.9969402872962126e-23, 'stdev_strain':
1.0715299257922202e-18, 'duty_cycle': 46.6796875, 'BLRMS200':
5.278628681909679e-24, 'BLRMS1000': 6.051544953857376e-21, 'BNS':
141.87615444330953}, {'GPSstart': 1382891520, 'UTCstart': '2023-11-01T16:31:42',
'detector': 'H1', 'sampling_rate': 4096, 'duration': 4096, 'format': 'gwf',
'url': 'https://gwosc.org/archive/data/O4a_4KHZ_R1/1382023168/H-
H1_GWOSC_O4a_4KHZ_R1-1382891520-4096.gwf', 'min_strain': -5.372688122354986e-18,
'max_strain': 5.362402307740685e-18, 'mean_strain': 1.9969402872962126e-23,
'stdev_strain': 1.0715299257922202e-18, 'duty_cycle': 46.6796875, 'BLRMS200':
5.278628681909679e-24, 'BLRMS1000': 6.051544953857376e-21, 'BNS':
141.87615444330953}, {'GPSstart': 1382895616, 'UTCstart': '2023-11-01T17:39:58',
'detector': 'H1', 'sampling_rate': 4096, 'duration': 4096, 'format': 'hdf5',
'url': 'https://gwosc.org/archive/data/O4a_4KHZ_R1/1382023168/H-
H1_GWOSC_O4a_4KHZ_R1-1382895616-4096.hdf5', 'min_strain':
-4.821827522699192e-18, 'max_strain': 4.796736982409019e-18, 'mean_strain':
-1.8653898711944875e-22, 'stdev_strain': 1.1782513127685192e-18, 'duty_cycle':
8.7646484375, 'BLRMS200': 5.004639123993648e-24, 'BLRMS1000':
4.952106402602299e-21, 'BNS': 142.60280515507844}, {'GPSstart': 1382895616,
'UTCstart': '2023-11-01T17:39:58', 'detector': 'H1', 'sampling_rate': 4096,
'duration': 4096, 'format': 'gwf', 'url': 'https://gwosc.org/archive/data/O4a_4K
HZ_R1/1382023168/H-H1_GWOSC_O4a_4KHZ_R1-1382895616-4096.gwf', 'min_strain':
-4.821827522699192e-18, 'max_strain': 4.796736982409019e-18, 'mean_strain':
-1.8653898711944875e-22, 'stdev_strain': 1.1782513127685192e-18, 'duty_cycle':
8.7646484375, 'BLRMS200': 5.004639123993648e-24, 'BLRMS1000':
4.952106402602299e-21, 'BNS': 142.60280515507844}, {'GPSstart': 1382899712,
'UTCstart': '2023-11-01T18:48:14', 'detector': 'H1', 'sampling_rate': 4096,
'duration': 4096, 'format': 'hdf5', 'url': 'https://gwosc.org/archive/data/O4a_4
KHZ_R1/1382023168/H-H1_GWOSC_O4a_4KHZ_R1-1382899712-4096.hdf5', 'min_strain':
-5.2316219885767225e-18, 'max_strain': 5.5006260950142426e-18, 'mean_strain':
2.262056136806449e-24, 'stdev_strain': 1.0979359938591247e-18, 'duty_cycle':
37.98828125, 'BLRMS200': 4.8916326868163955e-24, 'BLRMS1000':
5.570929443447215e-21, 'BNS': 143.65362215817004}]
```

[20]:
```python
def download_strain_file(download_url):
    # saving strain file to disk
    filename = download_url.split("/")[-1]
    with requests.get(download_url, stream=True) as r:
        # navigating file in small incriments (chunks)
        # and nicknaming as r for easier coding
        with open(filename, "wb") as f:
            for chunk in r.iter_content(chunk_size=8192):
                f.write(chunk)
```

```
                #iterating over the chunks
        return filename

for file in strain_files:
    if file["GPSstart"] == 1384779776 and file["format"] == 'hdf5':
        print(f"Downloading {file['url']}")
        fname = download_strain_file(file['url'])
```

```
Downloading https://gwosc.org/archive/data/O4a_4KHZ_R1/1384120320/H-
H1_GWOSC_O4a_4KHZ_R1-1384779776-4096.hdf5
```

1.1 Accessing the HDF5 File

[21]:
```
import numpy as np
from matplotlib import mlab
import matplotlib.pyplot as plt
import h5py
```

[22]:
```
filename = 'H-H1_GWOSC_O4a_4KHZ_R1-1384779776-4096.hdf5'
dataFile = h5py.File(filename, 'r')
```

[23]:
```
for key in dataFile.keys():
    print(key)
    # accessing data as dictionary
```

```
meta
quality
strain
```

[24]:
```
strain = dataFile['strain']['Strain']
ts = dataFile['strain']['Strain'].attrs['Xspacing']
print(f"ts = {ts}s, sample rate = {1/ts}Hz")
```

```
ts = 0.000244140625s, sample rate = 4096.0Hz
```

[25]:
```
# metadata
metaKeys = dataFile['meta'].keys()
meta = dataFile['meta']
for key in metaKeys:
    print(key, meta[key])
```

```
Description <HDF5 dataset "Description": shape (), type "|O">
DescriptionURL <HDF5 dataset "DescriptionURL": shape (), type "|O">
Detector <HDF5 dataset "Detector": shape (), type "|O">
Duration <HDF5 dataset "Duration": shape (), type "<i8">
FrameType <HDF5 dataset "FrameType": shape (), type "|O">
GPSstart <HDF5 dataset "GPSstart": shape (), type "<i8">
Observatory <HDF5 dataset "Observatory": shape (), type "|O">
```

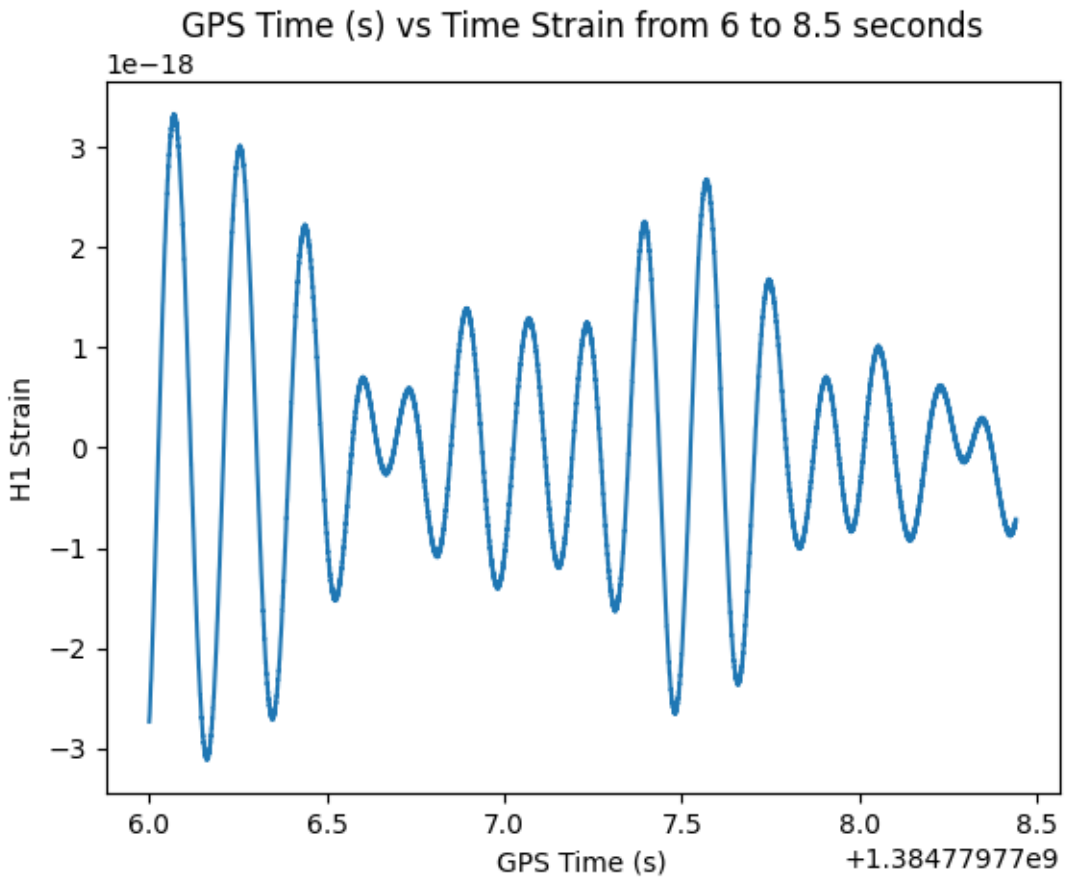
```
StrainChannel <HDF5 dataset "StrainChannel": shape (), type "|O">
Type <HDF5 dataset "Type": shape (), type "|O">
UTCstart <HDF5 dataset "UTCstart": shape (), type "|O">
```

[26]:
```
# creating time vector
gpsStart = meta['GPSstart'][()]
duration = meta['Duration'][()]
gpsEnd   = gpsStart + duration

time = np.arange(gpsStart, gpsEnd, ts)
```

[ ]:
```
# plotting the time series acros the 4096 seconds
plt.plot(time, strain[()])
plt.xlabel('GPS Time (s)')
plt.ylabel('H1 Strain')
plt.title("GPS Time vs H1 Strain")
plt.show()
```

[36]:
```
# zooming in on a specific time series
numsamples = 10000 # 4096 Hz
startTime  = 1264316116.0
startIndex = np.min(np.nonzero(startTime < time))
time_seg   = time[startIndex:(startIndex+numsamples)]
strain_seg = strain[startIndex:(startIndex+numsamples)]
plt.plot(time_seg, strain_seg)
plt.xlabel('GPS Time (s)')
plt.ylabel('H1 Strain')
plt.title("GPS Time (s) vs Time Strain from 6 to 8.5 seconds")
plt.show()
```



2 Accessing Data Quality

[27]:
```
dqInfo = dataFile['quality']['simple']
bitnameList = dqInfo['DQShortnames'][()]
descriptionList = dqInfo['DQDescriptions'][()]
nbits = len(bitnameList)

for bit in range(nbits):
    print(f"Channel #{bit} ({bitnameList[bit].decode()}): {descriptionList[bit].
 ↪decode()}")
    # 7 data quality categories and 5 injection categories
    # each represented as 1 Hz time series
    # printing dq channel names & descriptions
```

```
Channel #0 (DATA): data present
Channel #1 (CBC_CAT1): passes the cbc CAT1 test
Channel #2 (CBC_CAT2): passes cbc CAT2 test
Channel #3 (CBC_CAT3): passes cbc CAT3 test
```

```
Channel #4 (BURST_CAT1): passes burst CAT1 test
Channel #5 (BURST_CAT2): passes burst CAT2 test
Channel #6 (BURST_CAT3): passes burst CAT3 test
Channel #7 (STOCH_CAT1): passes stoch CAT1 test
Channel #8 (CW_CAT1): passes cw CAT1 test
```

[28]:
```
dqmask = dqInfo['DQmask'][()]
```

[29]:
```
value = dqmask[2400]
print("Value in decimal: {0}".format(value))
print("Same value in binary (with 7 bits): {0:#09b}".format(value))
# 1 in binary means that it's good data
# 0 means it's bad
```

```
Value in decimal: 511
Same value in binary (with 7 bits): 0b111111111
```

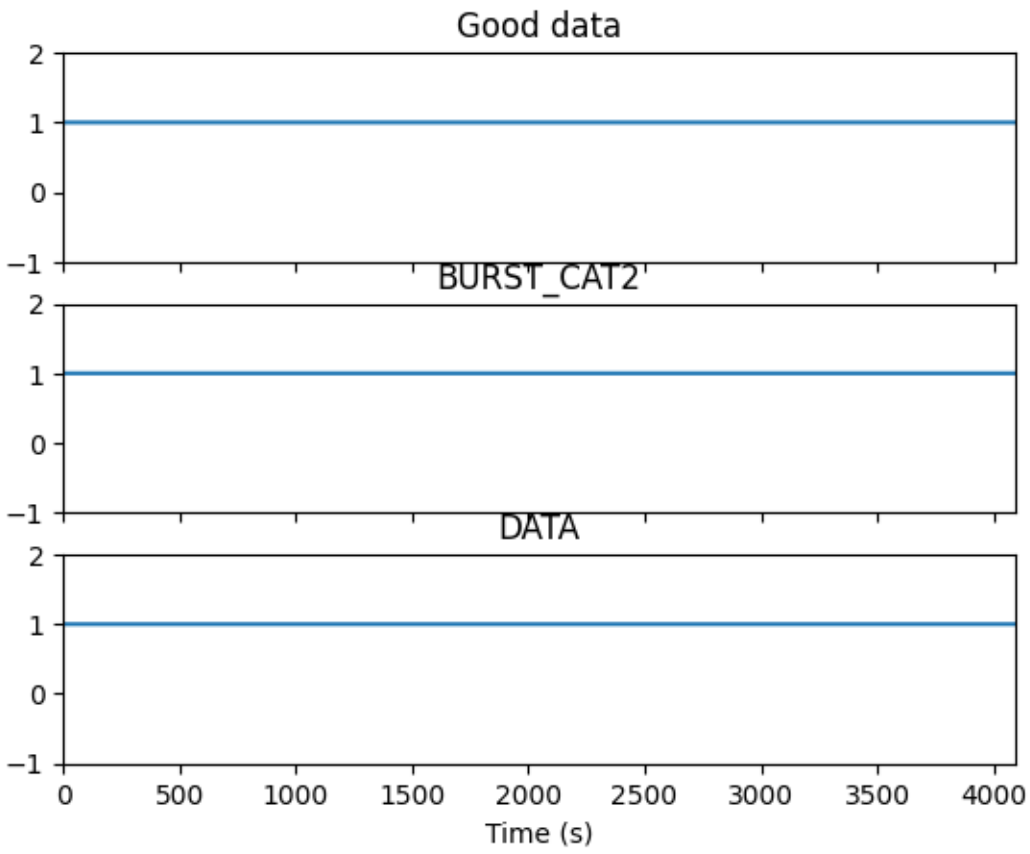
[42]:
```
data_channel = 0
data_mask = (dqmask >> data_channel) & 1 # 0 everywhere the data fails
                                         # and 1 wherever it pases

burst_cat2_channel = 5
burst_cat2_mask = (dqmask >> burst_cat2_channel) & 1
```

[30]:
```
goodData_mask_1hz = data_mask & burst_cat2_mask
```

[49]:
```
fig, (ax0, ax1, ax2) = plt.subplots(3, sharex=True, sharey=True)
ax0.plot(goodData_mask_1hz)
ax0.set_title('Good data')
ax1.plot(burst_cat2_mask)
ax1.set_title('BURST_CAT2')
ax2.plot(data_mask)
ax2.set_title('DATA')
ax2.axis([0, 4096, -1, 2])
ax2.set_xlabel('Time (s)')
# bad data would have spikes to the 0 along the plot
```

[49]:
```
Text(0.5, 0, 'Time (s)')
```

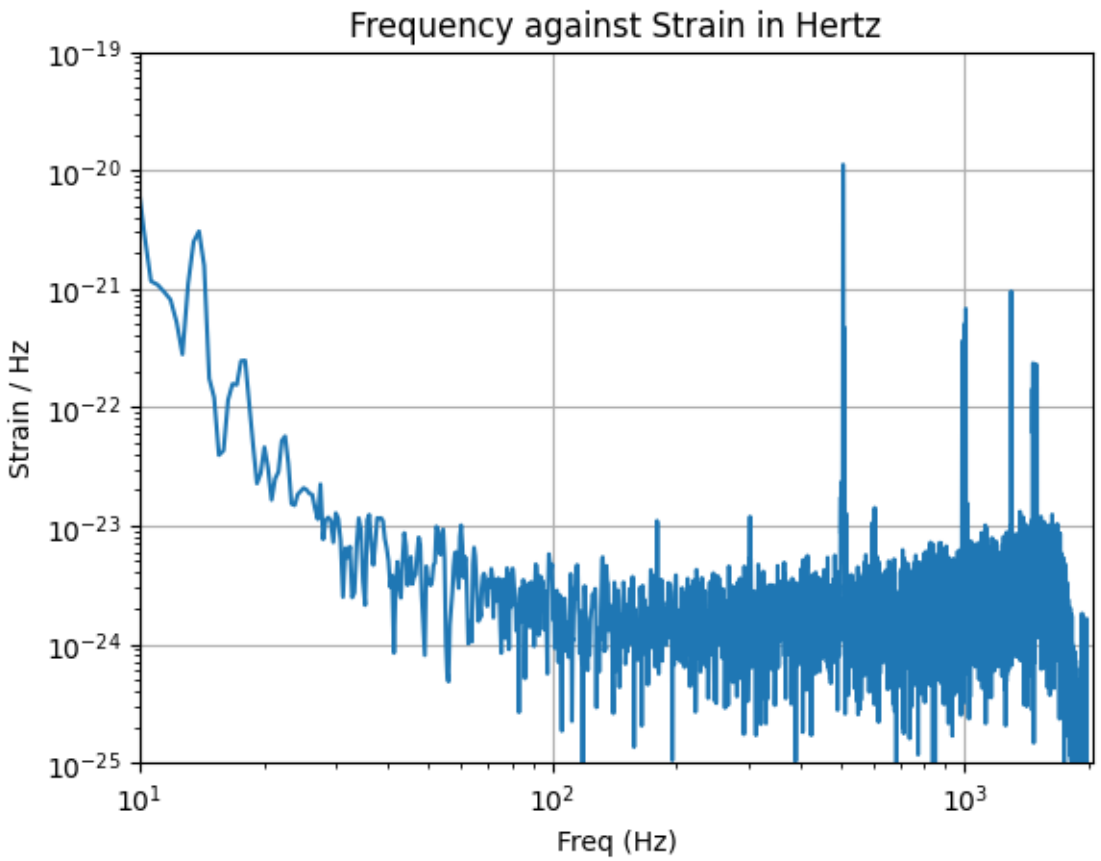


The following tables show that all data is good to use in analysis.

2.1 Analyzing frequency

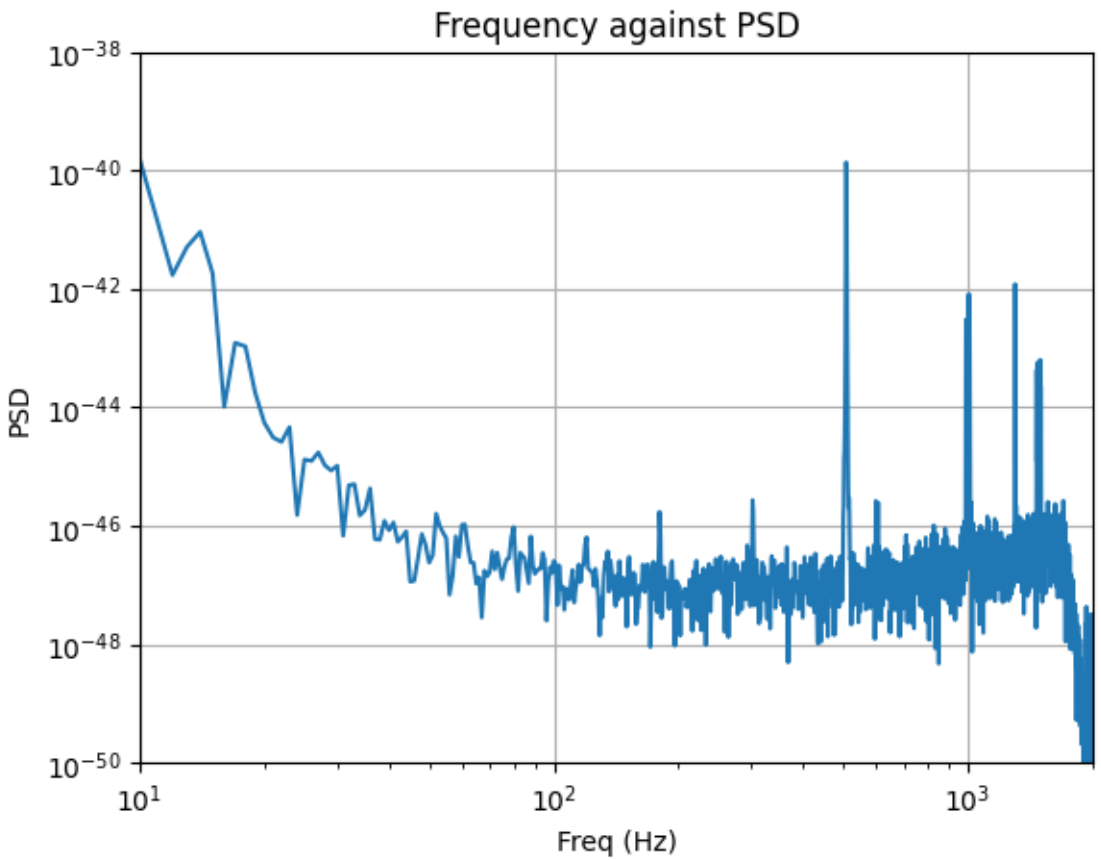
[31]:
```
# sampling frequency
fs = int(1.0 / ts)
```

[59]:
```
# using a 'blackman window' to reduce any leakages in data
window          = np.blackman(strain_seg.size)
windowed_strain = strain_seg * window
freq_domain     = np.fft.rfft(windowed_strain) / fs
freq            = np.fft.rfftfreq(len(windowed_strain)) * fs
plt.loglog(freq, abs(freq_domain))
plt.axis([10, fs/2.0, 1e-25, 1e-19])
plt.grid('on')
plt.xlabel('Freq (Hz)')
plt.ylabel('Strain / Hz')
plt.title("Frequency against Strain in Hertz")
plt.show()
```

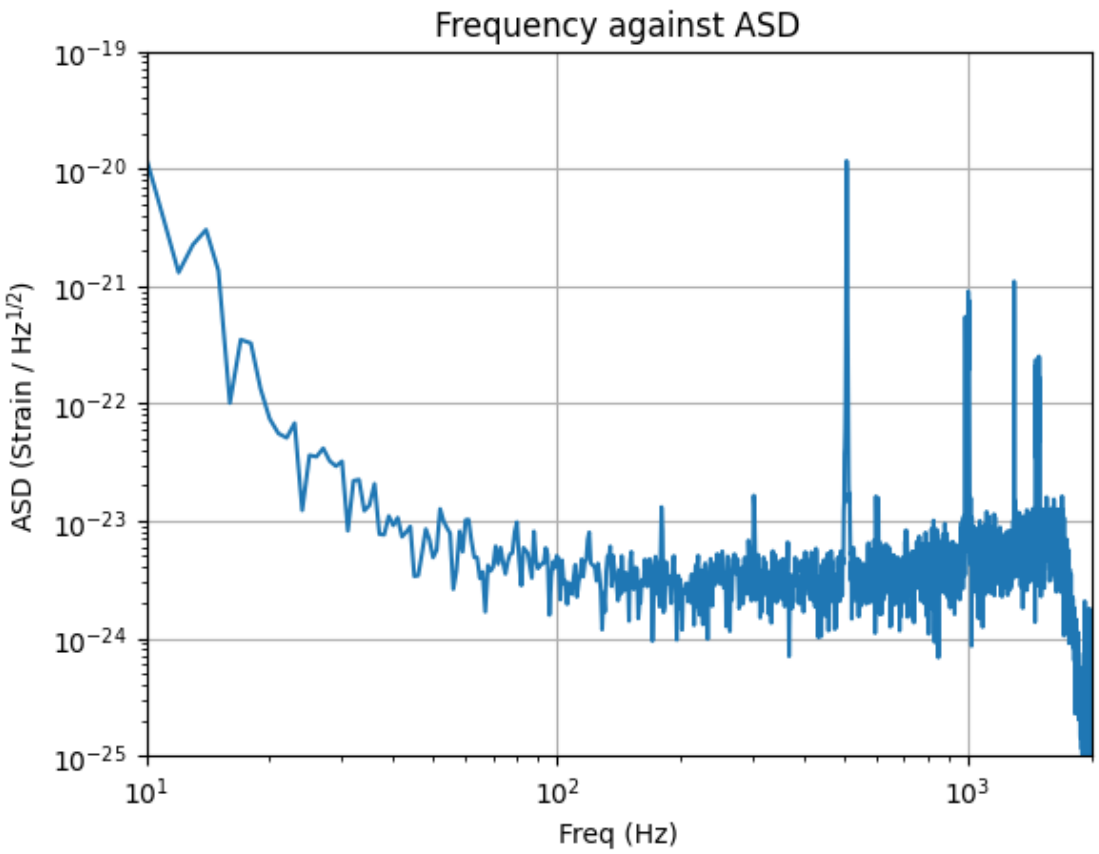


[60]:
```
# power spectral density to see how power is distributed
# available in matplotlib
Pxx, freqs = mlab.psd(strain_seg, Fs=fs, NFFT=fs)
plt.loglog(freqs, Pxx)
plt.axis([10, 2000, 1e-50, 1e-38])
plt.grid('on')
plt.xlabel('Freq (Hz)')
plt.ylabel('PSD')
plt.title("Frequency against PSD")
```

[60]:
```
Text(0.5, 1.0, 'Frequency against PSD')
```

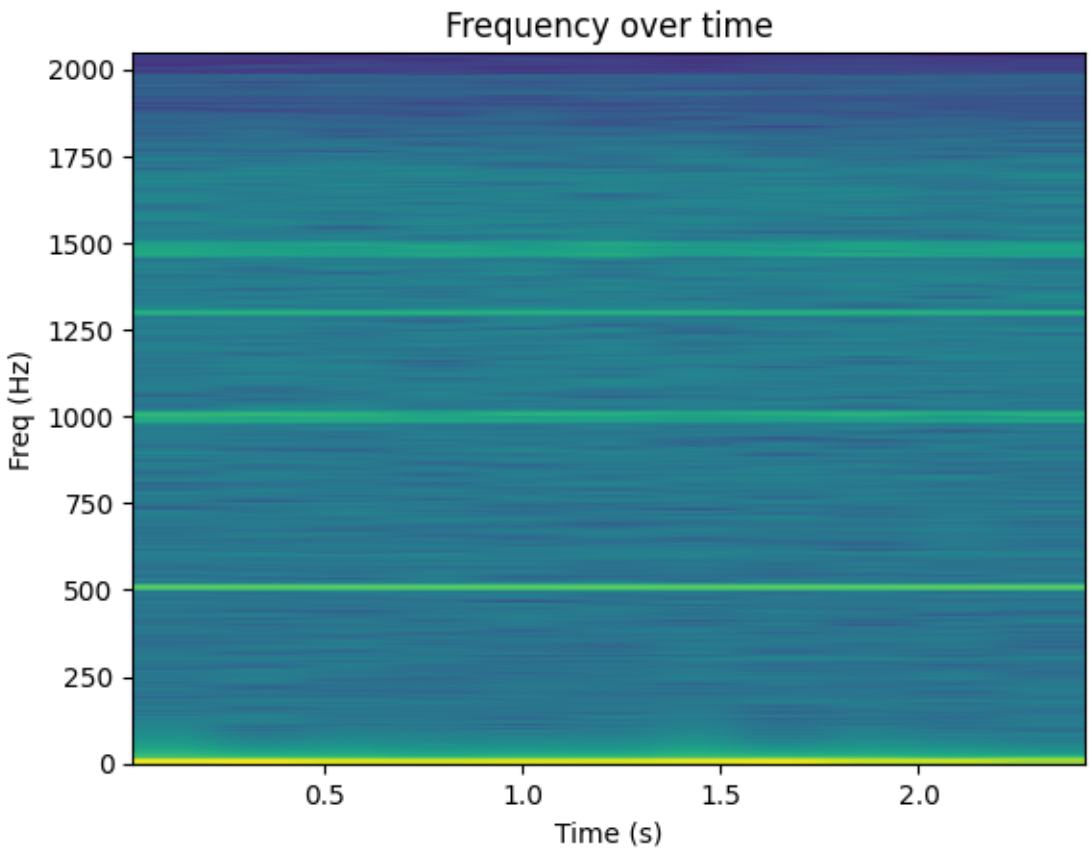


[62]:
```
plt.loglog(freqs, np.sqrt(Pxx)) # creates a plot with log scale on both axes
plt.axis([10, 2000, 1e-25, 1e-19])
plt.grid('on')
plt.xlabel('Freq (Hz)')
plt.ylabel('ASD (Strain / Hz$^{1/2})$')
plt.title("Frequency against ASD")
plt.show()
```



2.1.1 Creating a spectogram to see how frequency varies over time

[64]:
```
NFFT = 1024 # fourier transformation
short_window = np.blackman(NFFT)
spec_power, freqs, bins, im = plt.specgram(
    strain_seg, NFFT=NFFT, Fs=fs,
    window=short_window
)
plt.xlabel('Time (s)')
plt.ylabel('Freq (Hz)')
plt.title("Frequency over time")
plt.show()
```



[37]:
```
import warnings
warnings.filterwarnings("ignore", "Wswiglal-redir-stdio")

from pycbc.catalog import Merger
from pycbc.filter import highpass

# Load GW231123 event
event = Merger("GW231123")

strain = event.strain("H1")

print("Detector:", strain.detector)
print("Sample Rate:", strain.sample_rate)
print("Duration (s):", len(strain) / strain.sample_rate)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[37], line 8
      5 from pycbc.filter import highpass
      7 # Load GW231123 event
```

```
----> 8 event = Merger(           )
     10 strain = event.strain("H1")
     12 print("Detector:", strain.detector)

File /srv/conda/lib/python3.11/site-packages/pycbc/catalog/__init__.py:84, in␣
 ↪Merger.__init__(self, name, source)
     76 def __init__(self, name, source=None):
     77     """ Return the information of a merger
     78
     79     Parameters
   (…)     82         The name (GW prefixed date) of the merger event.
     83     """
---> 84     self.data = find_event_in_catalog(name, source=source)
     86     # Set some basic params from the dataset
     87     for key in self.data:

File /srv/conda/lib/python3.11/site-packages/pycbc/catalog/__init__.py:72, in␣
 ↪find_event_in_catalog(name, source)
     70                 return data
     71 else:
---> 72     raise ValueError(f'Did not find merger matching name: {name}')

ValueError: Did not find merger matching name: GW231123
```

[43]:
```
from pycbc.catalog import Merger
from pycbc.catalog import Catalog

cat = Catalog()
print(cat.names)     # list of events
print(len(cat.names))
```

```
dict_keys(['GW200322_091133-v1', 'GW200316_215756-v1', 'GW200311_115853-v1',
'GW200308_173609-v1', 'GW200306_093714-v1', 'GW200302_015811-v1',
'GW200225_060421-v1', 'GW200224_222234-v1', 'GW200220_124850-v1',
'GW200220_061928-v1', 'GW200219_094415-v1', 'GW200216_220804-v1',
'GW200210_092254-v1', 'GW200209_085452-v1', 'GW200208_222617-v1',
'GW200208_130117-v1', 'GW200202_154313-v1', 'GW200129_065458-v1',
'GW200128_022011-v1', 'GW200115_042309-v2', 'GW200112_155838-v1',
'GW191230_180458-v1', 'GW191222_033537-v1', 'GW191219_163120-v1',
'GW191216_213338-v1', 'GW191215_223052-v1', 'GW191204_171526-v1',
'GW191204_110529-v1', 'GW191129_134029-v1', 'GW191127_050227-v1',
'GW191126_115259-v1', 'GW191113_071753-v1', 'GW191109_010717-v1',
'GW191105_143521-v1', 'GW191103_012549-v1'])
35
```

[33]:
```
!pip install pycbc
```

```
Collecting pycbc
  Downloading pycbc-2.10.0-cp311-cp311-
manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (4.6 kB)
Requirement already satisfied: numpy>=1.16.0 in /srv/conda/lib/python3.11/site-
packages (from pycbc) (2.2.6)
Collecting cython>=0.29 (from pycbc)
  Downloading cython-3.2.2-cp311-cp311-
manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata
(5.0 kB)
Requirement already satisfied: scipy>=0.16.0 in /srv/conda/lib/python3.11/site-
packages (from pycbc) (1.14.1)
Collecting astropy!=4.0.5,!=4.2.1,>=2.0.3 (from pycbc)
  Using cached astropy-7.2.0-cp311-abi3-
manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata
(10 kB)
Requirement already satisfied: matplotlib>=1.5.1 in
/srv/conda/lib/python3.11/site-packages (from pycbc) (3.10.7)
Collecting mpld3>=0.3 (from pycbc)
  Downloading mpld3-0.5.12-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: pillow in /srv/conda/lib/python3.11/site-packages
(from pycbc) (11.1.0)
Requirement already satisfied: h5py!=3.7.0,>=3.0.0 in
/srv/conda/lib/python3.11/site-packages (from pycbc) (3.15.1)
Requirement already satisfied: jinja2 in /srv/conda/lib/python3.11/site-packages
(from pycbc) (3.1.6)
Requirement already satisfied: Mako>=1.0.1 in /srv/conda/lib/python3.11/site-
packages (from pycbc) (1.3.10)
Requirement already satisfied: beautifulsoup4>=4.6.0 in
/srv/conda/lib/python3.11/site-packages (from pycbc) (4.12.3)
Requirement already satisfied: tqdm in /srv/conda/lib/python3.11/site-packages
(from pycbc) (4.67.1)
Requirement already satisfied: setuptools in /srv/conda/lib/python3.11/site-
packages (from pycbc) (75.6.0)
Collecting gwdatafind (from pycbc)
  Downloading gwdatafind-2.1.1-py3-none-any.whl.metadata (3.6 kB)
Collecting pegasus-wms.api>=5.1.1 (from pycbc)
  Downloading pegasus_wms_api-5.1.1.tar.gz (73 kB)
  Installing build dependencies … done
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Collecting igwn-ligolw (from pycbc)
  Downloading igwn_ligolw-2.1.0-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_28_x86_64.whl.metadata (2.3
kB)
Collecting igwn-segments (from pycbc)
  Downloading igwn_segments-2.1.0-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_28_x86_64.whl.metadata (2.0
kB)
```

```
Collecting lalsuite!=7.2 (from pycbc)
  Downloading lalsuite-7.26.1-cp311-cp311-manylinux_2_28_x86_64.whl.metadata
(3.2 kB)
Collecting lscsoft-glue>=1.59.3 (from pycbc)
  Downloading lscsoft_glue-4.1.1-py3-none-any.whl.metadata (3.1 kB)
Collecting pykerr (from pycbc)
  Downloading pykerr-0.2.0-py3-none-any.whl.metadata (6.1 kB)
Collecting pyerfa>=2.0.1.1 (from astropy!=4.0.5,!=4.2.1,>=2.0.3->pycbc)
  Using cached
pyerfa-2.0.1.5-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(5.7 kB)
Collecting astropy-iers-data>=0.2025.10.27.0.39.10 (from
astropy!=4.0.5,!=4.2.1,>=2.0.3->pycbc)
  Using cached astropy_iers_data-0.2025.12.1.0.45.12-py3-none-any.whl.metadata
(3.4 kB)
Requirement already satisfied: PyYAML>=6.0.0 in /srv/conda/lib/python3.11/site-
packages (from astropy!=4.0.5,!=4.2.1,>=2.0.3->pycbc) (6.0.3)
Requirement already satisfied: packaging>=22.0.0 in
/srv/conda/lib/python3.11/site-packages (from
astropy!=4.0.5,!=4.2.1,>=2.0.3->pycbc) (24.2)
Requirement already satisfied: soupsieve>1.2 in /srv/conda/lib/python3.11/site-
packages (from beautifulsoup4>=4.6.0->pycbc) (2.8)
Requirement already satisfied: python-dateutil in
/srv/conda/lib/python3.11/site-packages (from lalsuite!=7.2->pycbc)
(2.9.0.post0)
Requirement already satisfied: MarkupSafe>=0.9.2 in
/srv/conda/lib/python3.11/site-packages (from Mako>=1.0.1->pycbc) (3.0.3)
Requirement already satisfied: contourpy>=1.0.1 in
/srv/conda/lib/python3.11/site-packages (from matplotlib>=1.5.1->pycbc) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /srv/conda/lib/python3.11/site-
packages (from matplotlib>=1.5.1->pycbc) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/srv/conda/lib/python3.11/site-packages (from matplotlib>=1.5.1->pycbc) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/srv/conda/lib/python3.11/site-packages (from matplotlib>=1.5.1->pycbc) (1.4.9)
Requirement already satisfied: pyparsing>=3 in /srv/conda/lib/python3.11/site-
packages (from matplotlib>=1.5.1->pycbc) (3.2.5)
Collecting pegasus-wms.common (from pegasus-wms.api>=5.1.1->pycbc)
  Downloading pegasus_wms_common-5.1.1.tar.gz (103 kB)
  Installing build dependencies … done
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: six>=1.5 in /srv/conda/lib/python3.11/site-
packages (from python-dateutil->lalsuite!=7.2->pycbc) (1.17.0)
Collecting igwn-auth-utils>=0.3.1 (from gwdatafind->pycbc)
  Downloading igwn_auth_utils-1.4.0-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: cryptography>=44.0.1 in
/srv/conda/lib/python3.11/site-packages (from igwn-auth-
```

```
utils>=0.3.1->gwdatafind->pycbc) (46.0.3)
Requirement already satisfied: requests>=2.32.0 in
/srv/conda/lib/python3.11/site-packages (from igwn-auth-
utils>=0.3.1->gwdatafind->pycbc) (2.32.3)
Collecting safe-netrc>=1.0 (from igwn-auth-utils>=0.3.1->gwdatafind->pycbc)
  Downloading safe_netrc-1.0.1-py3-none-any.whl.metadata (1.9 kB)
Collecting scitokens>=1.8 (from igwn-auth-utils>=0.3.1->gwdatafind->pycbc)
  Downloading scitokens-1.8.1-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: cffi>=1.14 in /srv/conda/lib/python3.11/site-
packages (from cryptography>=44.0.1->igwn-auth-utils>=0.3.1->gwdatafind->pycbc)
(2.0.0)
Requirement already satisfied: pycparser in /srv/conda/lib/python3.11/site-
packages (from cffi>=1.14->cryptography>=44.0.1->igwn-auth-
utils>=0.3.1->gwdatafind->pycbc) (2.22)
Requirement already satisfied: charset_normalizer<4,>=2 in
/srv/conda/lib/python3.11/site-packages (from requests>=2.32.0->igwn-auth-
utils>=0.3.1->gwdatafind->pycbc) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /srv/conda/lib/python3.11/site-
packages (from requests>=2.32.0->igwn-auth-utils>=0.3.1->gwdatafind->pycbc)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/srv/conda/lib/python3.11/site-packages (from requests>=2.32.0->igwn-auth-
utils>=0.3.1->gwdatafind->pycbc) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/srv/conda/lib/python3.11/site-packages (from requests>=2.32.0->igwn-auth-
utils>=0.3.1->gwdatafind->pycbc) (2025.10.5)
Requirement already satisfied: PyJWT>=1.6.1 in /srv/conda/lib/python3.11/site-
packages (from scitokens>=1.8->igwn-auth-utils>=0.3.1->gwdatafind->pycbc)
(2.10.1)
Downloading
pycbc-2.10.0-cp311-cp311-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (9.4
MB)
                         9.4/9.4 MB
60.6 MB/s eta 0:00:00
Using cached astropy-7.2.0-cp311-abi3-
manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (9.8 MB)
Using cached astropy_iers_data-0.2025.12.1.0.45.12-py3-none-any.whl (2.0 MB)
Downloading cython-3.2.2-cp311-cp311-
manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (3.4 MB)
                         3.4/3.4 MB
37.7 MB/s eta 0:00:00
Downloading lalsuite-7.26.1-cp311-cp311-manylinux_2_28_x86_64.whl (40.3
MB)
                         40.3/40.3 MB
78.1 MB/s eta 0:00:00:00:0100:01
Downloading lscsoft_glue-4.1.1-py3-none-any.whl (51 kB)
Downloading mpld3-0.5.12-py3-none-any.whl (203 kB)
Using cached
```

```
pyerfa-2.0.1.5-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (738 kB)
Downloading gwdatafind-2.1.1-py3-none-any.whl (45 kB)
Downloading igwn_auth_utils-1.4.0-py3-none-any.whl (34 kB)
Downloading safe_netrc-1.0.1-py3-none-any.whl (10 kB)
Downloading scitokens-1.8.1-py3-none-any.whl (31 kB)
Downloading igwn_ligolw-2.1.0-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_28_x86_64.whl (179 kB)
Downloading igwn_segments-2.1.0-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_28_x86_64.whl (128 kB)
Downloading pykerr-0.2.0-py3-none-any.whl (48.1 MB)
                         48.1/48.1 MB
67.9 MB/s eta 0:00:00:00:0100:01
Building wheels for collected packages: pegasus-wms.api, pegasus-
wms.common
  Building wheel for pegasus-wms.api (pyproject.toml) … done
  Created wheel for pegasus-wms.api:
filename=pegasus_wms.api-5.1.1-py3-none-any.whl size=51367
sha256=ba2a9578b57d7c249f73e3da9ad33e57ea79294737c6efa942b59a5e3f1cb37f
  Stored in directory: /home/jovyan/.cache/pip/wheels/b9/3c/4f/a90e429fd07c03623
19b52d72c9ae1b5d8322adad87fd3ba66
  Building wheel for pegasus-wms.common (pyproject.toml) … done
  Created wheel for pegasus-wms.common:
filename=pegasus_wms.common-5.1.1-py3-none-any.whl size=25197
sha256=38c60ca483342049717958723dd06fda3e7327d7470af077f596a6d02a7feb4d
  Stored in directory: /home/jovyan/.cache/pip/wheels/fa/33/00/9879afc011a472e74
26e6bf709612695d782f83e98e9e1090a
Successfully built pegasus-wms.api pegasus-wms.common
Installing collected packages: safe-netrc, pyerfa, pegasus-wms.common, igwn-
segments, cython, astropy-iers-data, pykerr, pegasus-wms.api, lscsoft-glue,
igwn-ligolw, astropy, scitokens, mpld3, lalsuite, igwn-auth-utils, gwdatafind,
pycbc
                         17/17
[pycbc]m16/17 [pycbc]te]lue]pi]a]
ERROR: pip's dependency resolver does not currently take into

account all the packages that are installed. This behaviour is the source of the

following dependency conflicts.

linearmodels 0.0.0 requires setuptools_scm[toml]<9.0.0,>=8.0.0, which is not

installed.

Successfully installed astropy-7.2.0 astropy-iers-data-0.2025.12.1.0.45.12
cython-3.2.2 gwdatafind-2.1.1 igwn-auth-utils-1.4.0 igwn-ligolw-2.1.0 igwn-
segments-2.1.0 lalsuite-7.26.1 lscsoft-glue-4.1.1 mpld3-0.5.12 pegasus-
wms.api-5.1.1 pegasus-wms.common-5.1.1 pycbc-2.10.0 pyerfa-2.0.1.5 pykerr-0.2.0
safe-netrc-1.0.1 scitokens-1.8.1
```

[36]:
```
# removing low-freq noise
strain_hp = highpass(strain, 15.0) # 15. is the standard cutoff for
                                   # low frequencies in BBH analyses
white = strain_hp.whiten(4, 4) # 4 second long enough for noise estimate

t = white.sample_times - event.time
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 2
      1 # removing low-freq noise
----> 2 strain_hp = highpass(strain, 15.0) # 15. is the standard cutoff for
      3                                    # low frequencies in BBH analyses
      4 white = strain_hp.whiten(4, 4) # 4 second long enough for noise estimat

File /srv/conda/lib/python3.11/site-packages/pycbc/filter/resample.py:342, in␣
 ↪highpass(timeseries, frequency, filter_order, attenuation)
    312 """Return a new timeseries that is highpassed.
    313
    314 Return a new time series that is highpassed above the `frequency`.
   (…)    338
    339 """
    341 if not isinstance(timeseries, TimeSeries):
--> 342     raise TypeError("Can only resample time series")
    344 if timeseries.kind != 'real':
    345     raise TypeError("Time series must be real")

TypeError: Can only resample time series
```

[ ]:
```
plt.figure(figsize=(12, 4))
plt.plot(t, white, linewidth=0.7)
plt.xlim(-1, 0.2)
plt.xlabel("Time (s)")
plt.ylabel("Whitened strain")
plt.title("Whitened strain (H1)")
plt.show()
```

[ ]:
```
print("Mass 1:", event.m1_source, "solar masses")   # primary BH mass␣
 ↪(source-frame)
print("Mass 2:", event.m2_source, "solar masses")   # secondary BH mass
print("Final Mass:", event.final_mass_source, "solar masses")  # post-merger BH␣
 ↪mass
print("Final Spin:", event.final_spin)              # dimensionless spin␣
 ↪parameter (0-1)
print("Luminosity Distance:", event.distance, "Mpc")
```

[ ]:
```
from pycb.waveform import get_td_waveform
hp, hc = get_td_waveform( # accessing theoreitical waveform from pymc catalog
    approximant="IMRPhenomD",
    mass1=event.m1_source,
    mass2=event.m2_source,

    delta_t=1 / strain.sample_rate
    f_lower=15
    )

plt.figure(figsize=(12, 4))
plt.plot(hp.sample_times, hp, label='h_plus')
plt.plot(hc.sample_times, hc, label='h_cross')
plt.legend()

plt.title("Best-Fit Waveform for GW231123")
plt.xlabel("Time (s)")
plt.ylabel("Strain")
plt.show()
```

[ ]: