Tuto 1.3 Q-transforms with GWpy

November 17, 2020

1 Gravitational Wave Open Data Workshop #3

Tutorial 1.3: Data representations in GWpy This notebook covers some different ways of representing data, including

- the spectrogram
- the Q-transform

with a challenge you can explore for each method.

Click this link to view this tutorial in Google Colaboratory

1.1 Installation (execute only if running on a cloud platform or if you haven't done the installation already!)

Note: we use `pip`, but **it is recommended** to use conda on your own machine, as explained in the installation instructions. This usage might look a little different than normal, simply because we want to do this directly from the notebook.

[1]:
```
# -- Uncomment following line if running in Google Colab
#! pip install -q 'gwpy==1.0.1'
```

Important: With Google Colab, you may need to restart the runtime after running the cell above.

1.2 Initialization

[1]:
```
%matplotlib inline
import gwpy
```

1.3 Showing the time-evolution of FFTs

The FFT, and the ASD, show us a snapshot of the frequency-domain content of our signal, at a single time. It is commonly useful to show how this frequency-domain content evolves over time.

For this we use spectrograms, which show the FFT (or ASD) at each time step on a time-frequency-amplitude axis. The `TimeSeries` in GWpy includes two methods for this:

- `spectrogram()` - show an averaged ASD at each time step, and

- `spectrogram2()` - show a single-FFT ASD at each time step

Which one should I use? The short answer is use `spectrogram2()` for short(ish) chunks of data, less than a minute or so, and `spectrogram()` for longer chunks where the averaging helps remove very short noise bursts.

First, let's reload our data:

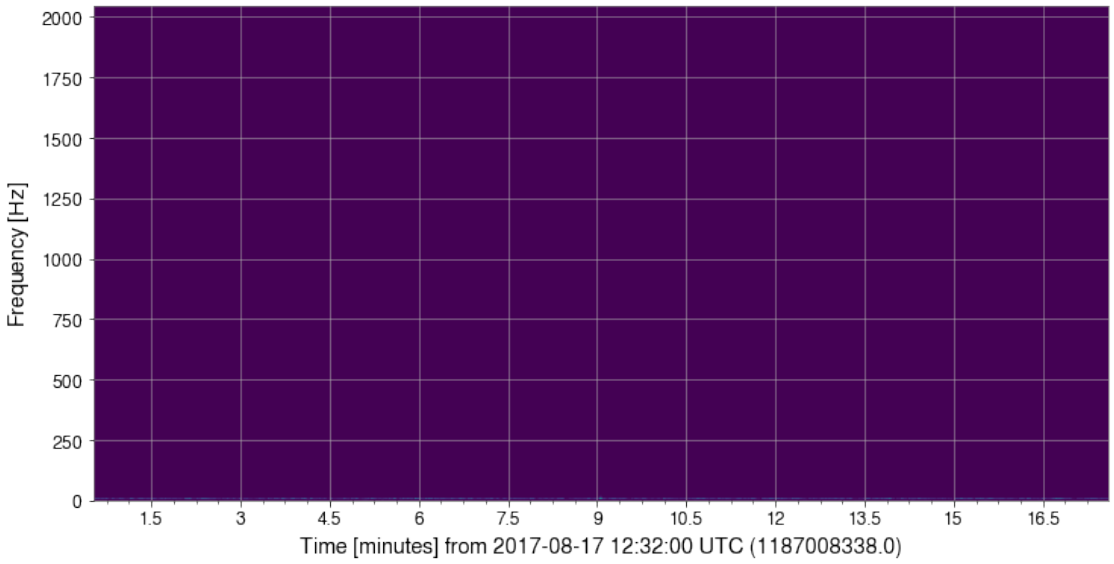
[15]:
```
from gwosc.datasets import event_gps
from gwpy.timeseries import TimeSeries

gps = event_gps('GW170817')
print("GW170817 GPS:", gps)

ldata = TimeSeries.fetch_open_data('L1', int(gps)-512, int(gps)+512, cache=True)
print("GW170817 data")
print(ldata)
```

```
GW170817 GPS: 1187008882.4
GW170817 data
TimeSeries([2.06056010e-20, 1.59181918e-20, 2.18438811e-20, …,
            1.25504332e-19, 1.23976846e-19, 1.22231459e-19]
           unit: dimensionless,
           t0: 1187008370.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
```

Now, we can generate our spectrogram using a specific FFT length (remembering to use a window):

Each of these methods returns the output as stacked power spectral densities, so we take the square root to get back to a familiar amplitude spectral density

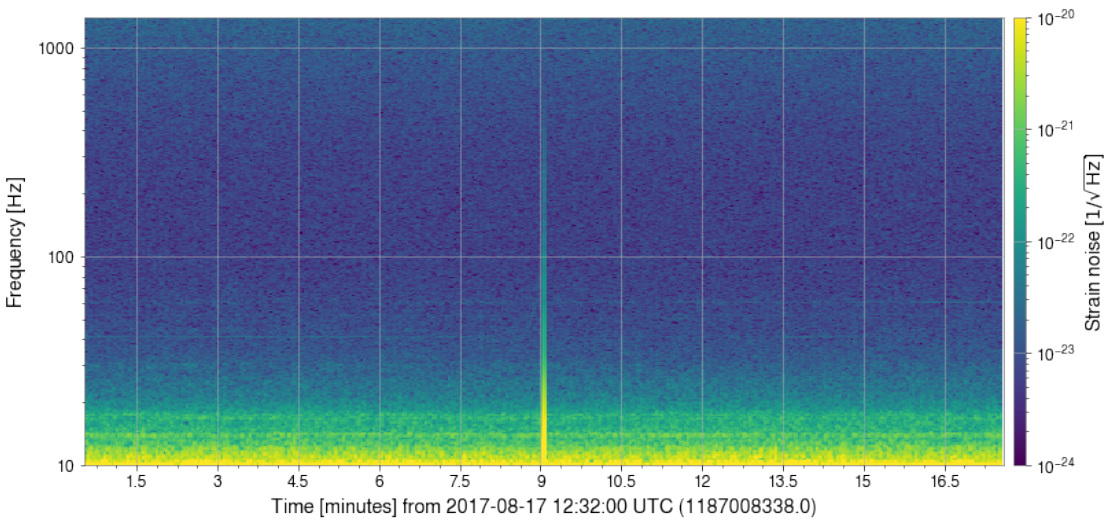
[16]:
```
specgram = ldata.spectrogram2(fftlength=4, overlap=2, window='hann') ** (1/2.)
plot = specgram.plot()
```



Hmmm… something's not right. We need to pass a few more arguments to our plot to control the display (especially the colouring):

[17]:
```
ax = plot.gca()
ax.set_yscale('log')
ax.set_ylim(10, 1400)
ax.colorbar(
    clim=(1e-24, 1e-20),
    norm="log",
    label=r"Strain noise [$1/\sqrt{\mathrm{Hz}}$]",
)
plot  # refresh
```

[17]:



Here we can see how the ASD for LIGO-Livingston evolves over a ~17 minute span around GW170817. We can see that the low-frequency noise (<30 Hz) rumbles along with some variation, but high frequencies (>100 Hz) are relatively stable. Between 30-100 Hz we can see some narrow features appearing and disappearing as non-stationary noise affects the measurement.

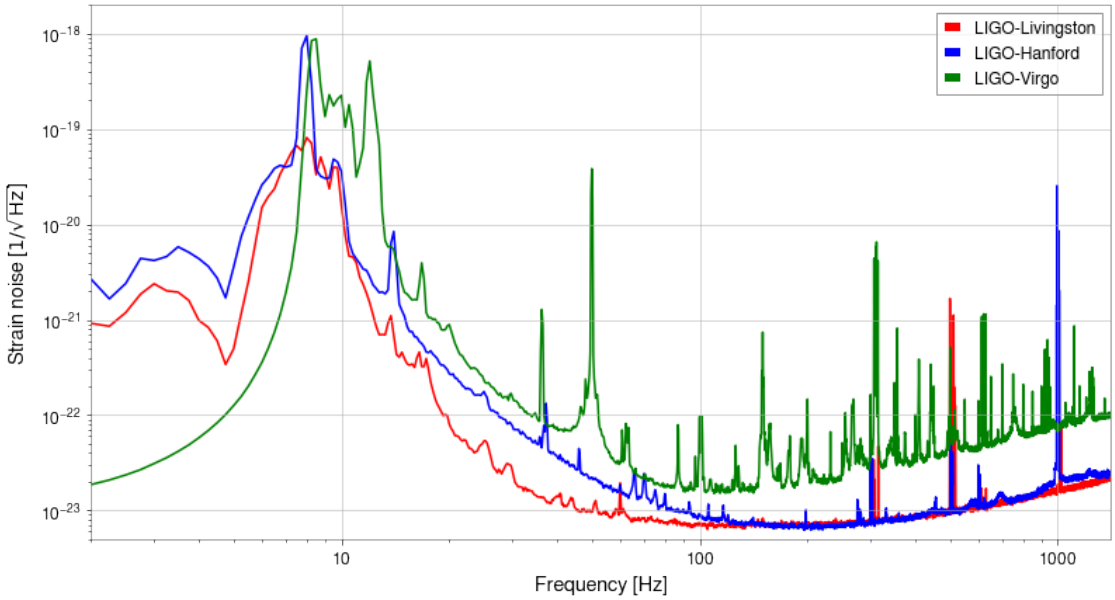
1.4 Spectrogram challenge!

- download the data for all three detectors involved with the GW170814 detection, generate a PSD for each, and make a plot
- make a spectrogram of data for 10 minutes around the GW170817 detection for the LIGO-Livingston detector

[9]:
```
from gwosc.datasets import event_gps
from gwpy.timeseries import TimeSeries

gps_GW170814 = event_gps('GW170814')
print("GW170814 GPS:", gps_GW170814)

data_GW170814_L1 = TimeSeries.fetch_open_data('L1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data L1 =",data_GW170814_L1)
data_GW170814_L1_asd = data_GW170814_L1.asd(fftlength=4, method="median")

data_GW170814_H1 = TimeSeries.fetch_open_data('H1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data H1 =",data_GW170814_H1)
data_GW170814_H1_asd = data_GW170814_H1.asd(fftlength=4, method="median")

data_GW170814_V1 = TimeSeries.fetch_open_data('V1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data V1 =",data_GW170814_V1)
data_GW170814_V1_asd = data_GW170814_V1.asd(fftlength=4, method="median")

import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(13,7))
# and plot using standard colours
ax.plot(data_GW170814_L1_asd, label='LIGO-Livingston', color='red')
ax.plot(data_GW170814_H1_asd, label='LIGO-Hanford', color='blue')
ax.plot(data_GW170814_V1_asd, label='LIGO-Virgo', color='green')

ax.set_xlim(2, 1400)
ax.set_ylim(5e-24, 2e-18)
# update the Livingston line to use standard colour, and have a label
```

```
ax.set_yscale('log')
ax.set_xscale('log')
ax.set_ylabel(r'Strain noise [$1/\sqrt{\mathrm{Hz}}$]')
ax.set_xlabel(r'Frequency [Hz]')

ax.legend()
plt.show()
```

```
GW170814 GPS: 1186741861.5
GW170814 data L1 = TimeSeries([6.71509695e-20, 6.97313000e-20, 6.22758834e-20,
…,
            5.42646307e-20, 6.40301087e-20, 5.42822595e-20]
           unit: dimensionless,
           t0: 1186741261.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
GW170814 data H1 = TimeSeries([-7.38466061e-19, -7.37551069e-19,
-6.95249542e-19,
            …, -7.23194401e-19, -7.48978223e-19,
            -7.05070524e-19]
           unit: dimensionless,
           t0: 1186741261.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
GW170814 data V1 = TimeSeries([-1.48413879e-19, -1.48239654e-19,
-1.47302345e-19,
            …, -2.80828691e-19, -2.90640523e-19,
            -3.14884932e-19]
           unit: dimensionless,
           t0: 1186741261.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
```

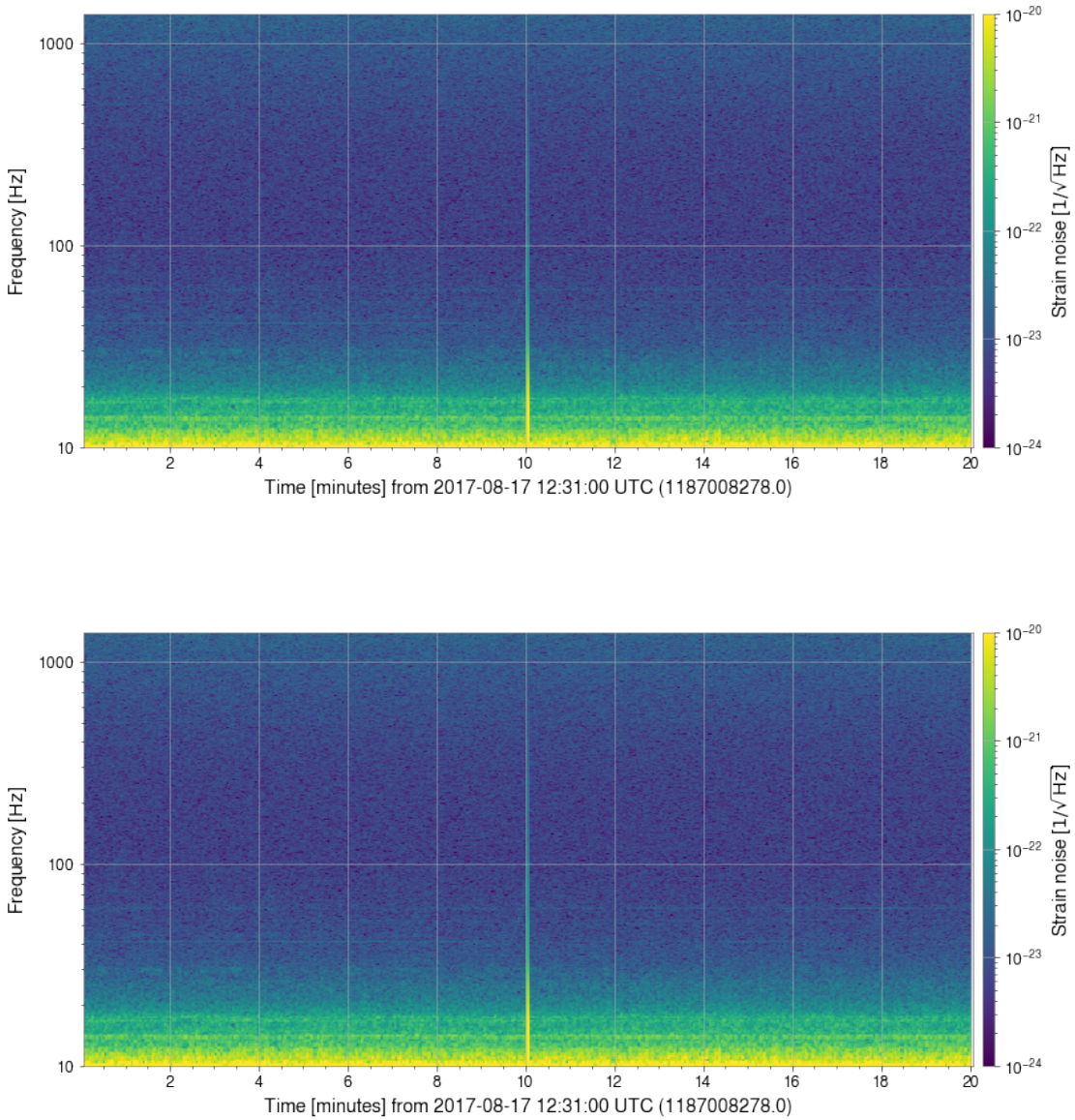


[14]:
```
from gwosc.datasets import event_gps
from gwpy.timeseries import TimeSeries

gps_GW170817 = event_gps('GW170817')
print("GW170817 GPS:", gps_GW170817)

data_GW170817_L1 = TimeSeries.fetch_open_data('L1', int(gps_GW170817)-600,␣
 ↪int(gps_GW170817)+600, cache=True)
print("GW170817 data L1",data_GW170817_L1)

specgram_GW170817_L1 = data_GW170817_L1.spectrogram2(fftlength=4, overlap=2,␣
 ↪window='hann') ** (1/2.)

plot = specgram_GW170817_L1.plot()

ax = plot.gca()
ax.set_yscale('log')
ax.set_ylim(10, 1400)
ax.colorbar(
    clim=(1e-24, 1e-20),
    norm="log",
    label=r"Strain noise [$1/\sqrt{\mathrm{Hz}}$]",
)
plot  # refresh
```

```
GW170817 GPS: 1187008882.4
GW170817 data L1 TimeSeries([ 4.88755885e-20,  6.25354330e-20,  6.57188880e-20,
```

```
 …, -6.44556120e-20, -6.99363912e-20,
 -6.21959184e-20]
unit: dimensionless,
t0: 1187008282.0 s,
dt: 0.000244140625 s,
name: Strain,
channel: None)
```

[14]:



1.5 Q-transforms in GWpy

The spectrogram above is a useful way to show the variation of a power spectral density (PSD) estimate over time. It's best used to see general trends in how the sensitivity of the GW detectors is changing over longish periods (minutes or hours).

In this section, we will see how we can use a special filter called a Q-transform to create a time-frequency representation of our data that allows use to pick out features at different frequencies, and how they evolve over very short times, without much prior knowledge of the signal morphology.

See this article for more details on the Q-transform and its application to gravitational-wave data.

First, lets reload some data from LIGO Hanford around GW170817:

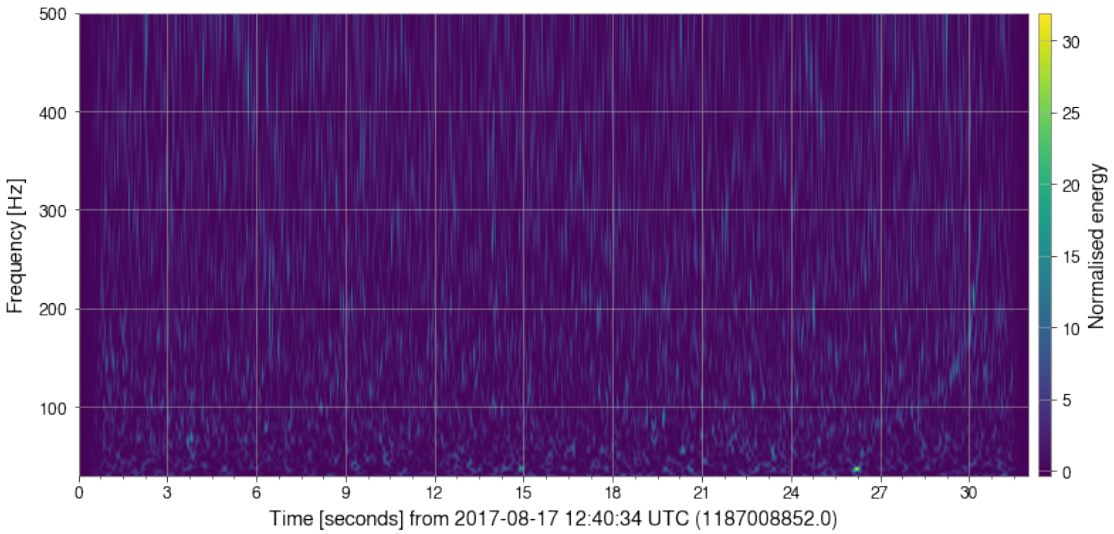
[18]:
```
segment = (int(gps) - 30, int(gps) + 2)
hdata = TimeSeries.fetch_open_data('H1', *segment, verbose=True, cache=True)
```

```
Fetched 1 URLs from www.gw-openscience.org for [1187008852 .. 1187008884))
Reading data… [Done]
```

We can now use the q_transform() method of the hdata TimeSeries to create our time-frequency representation (as a spectrogram).

[19]:
```
hq = hdata.q_transform(frange=(30, 500))
plot = hq.plot()
plot.colorbar(label="Normalised energy")
```

[19]:
```
<matplotlib.colorbar.Colorbar at 0x7fa652e78310>
```

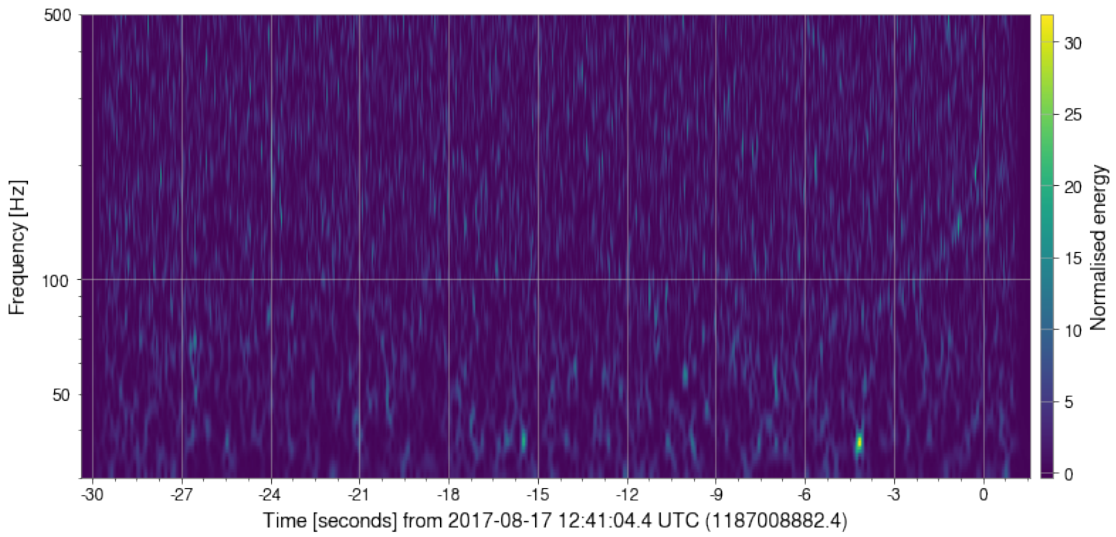


From this we can see a different representation of the data. Because the Q-transform returns (by default) normalised energy, the low-frequency rumbling is now much less obvious, and we can see better some noises at higher frequencies.

But, we can clean up the display to better visualise the data:

[20]:
```
ax = plot.gca()
ax.set_epoch(gps)
ax.set_ylim(30, 500)
ax.set_yscale("log")
plot  # refresh
```

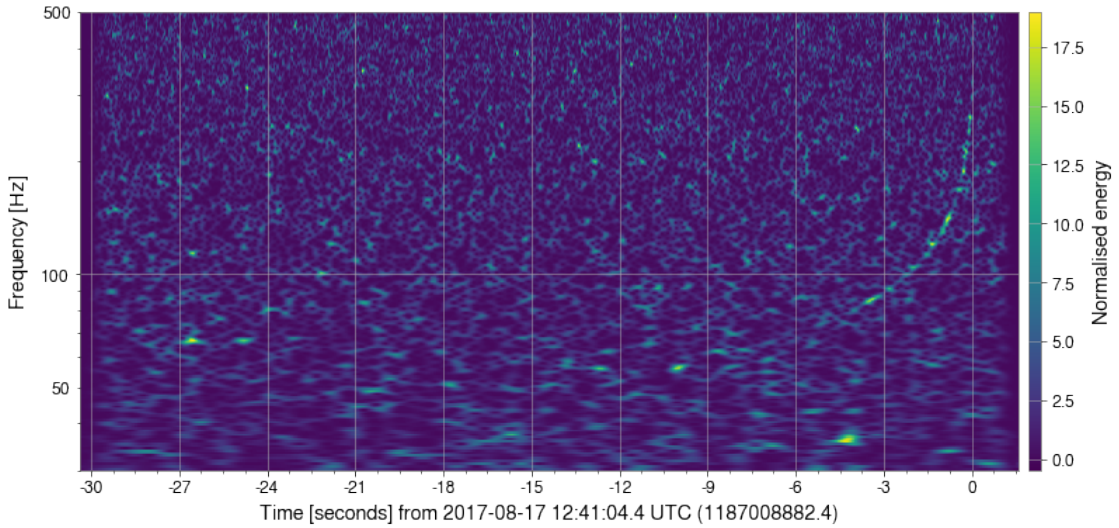
[20]:



Now we can see a more prominent feature starting at ~-6 seconds that looks a little familiar. Here we can use our knowledge of the Q-transform, and our hunch about the origin of the `feature' to choose a more specific range of `Q' for the Q-transform, so as to better resolve the feature:

[21]:
```
hq = hdata.q_transform(frange=(30, 500), qrange=(100, 110))
plot = hq.plot()
ax = plot.gca()
ax.set_epoch(gps)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")
```

[21]:
```
<matplotlib.colorbar.Colorbar at 0x7fa652b004d0>
```

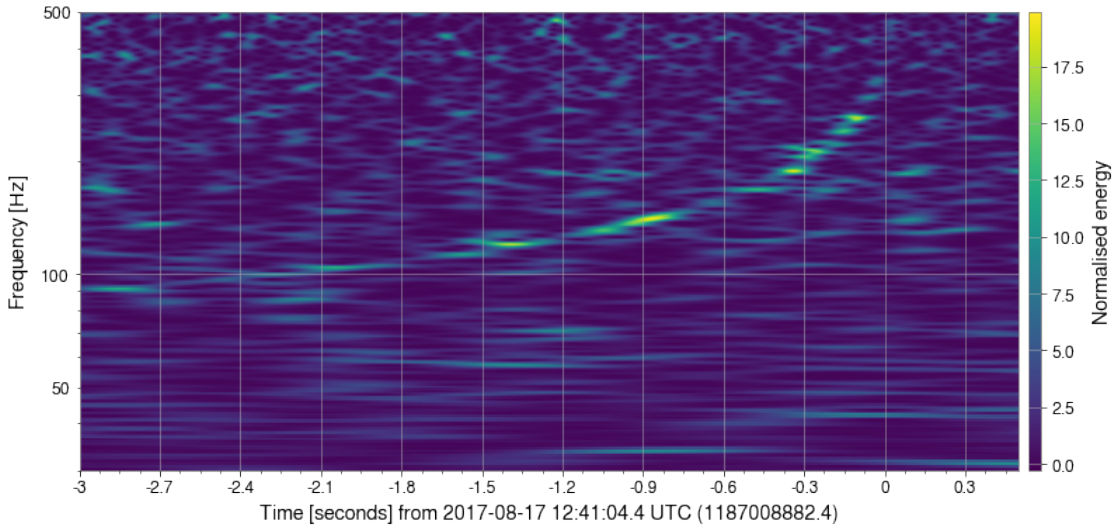


Now we see the beautiful, clear track of a BNS merger, visible from about -4 seconds (maye -10 if you squint), all the way through to the merger at T=0.

HINT: We can also use the outseg option to zoom in around the merger:

[22]:
```
#-- Use OUTSEG for small time range
hq2 = hdata.q_transform(frange=(30, 500), qrange=(80, 110), outseg=(gps-3,gps+0.
↪5))
plot = hq2.plot()
ax = plot.gca()
ax.set_epoch(gps)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")
```

[22]:
```
<matplotlib.colorbar.Colorbar at 0x7fa652bb5ad0>
```



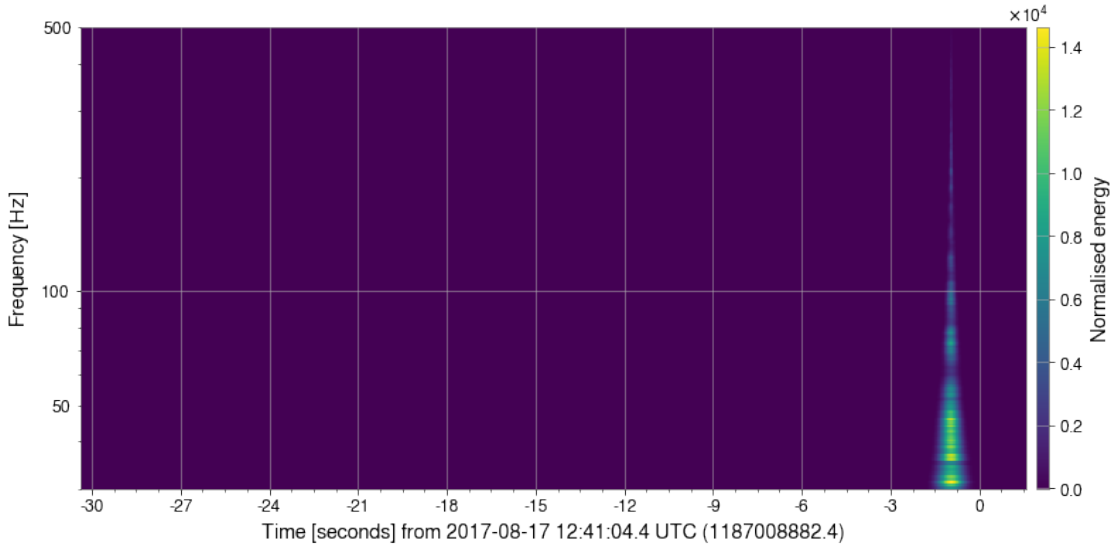
We can repeat the exercise using LIGO-Livingston data to see something even more remarkable. First we download and filter the Livingston data:

[23]:
```
ldata = TimeSeries.fetch_open_data('L1', *segment, verbose=True)
```

```
Fetched 1 URLs from www.gw-openscience.org for [1187008852 .. 1187008884))
Reading data… [Done]
```

[24]:
```
lq = ldata.q_transform(frange=(30, 500), qrange=(100, 110))
plot = lq.plot()
ax = plot.gca()
ax.set_epoch(gps)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")
```

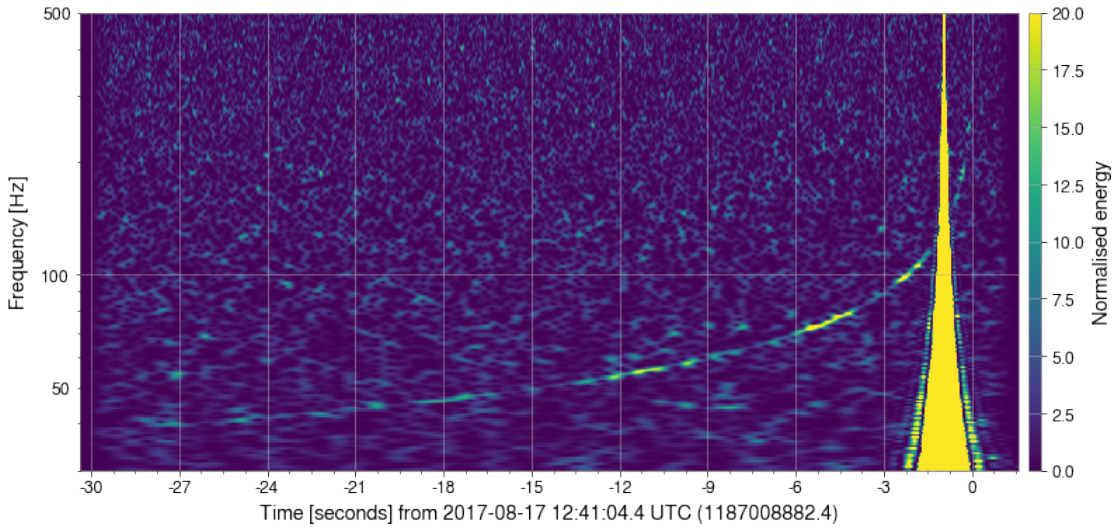
[24]:
```
<matplotlib.colorbar.Colorbar at 0x7fa652a08250>
```



Now we can see a large blob of energy that is 1000 times louder than what we see in the LIGO-Hanford data. As luck would have it, an instrumental glitch almost exactly overlaps the BNS signal in LIGO-Livingston. But, we can rescale things to see the astrophyscal signal better:

[25]:
```
plot.colorbars[0].mappable.set_clim(0,20)
plot.refresh()
plot
```

[25]:



Now we can see the BNS signal track all the way back to T=-28 seconds in LIGO-Livingston data!

This is basically the same procedure (and the same code) that was used to produce Figures 1 and 2 of the BNS discovery article `*Observation of Gravitational Waves from a Binary Neutron Star Inspiral*' [link]

1.6 Q-scan challenge!

- download the data for all detectors involved with the GW170814 detection. Which detectors registered this event?
- generate a Q-scan for the data in each detector. (Hint: try using a much smaller time range and Q-range than you did for GW170817)

[29]:
```
from gwosc.locate import get_event_urls
urls = get_event_urls('GW170814')
print(urls)
```

```
['https://www.gw-openscience.org/eventapi/json/GWTC-1-confident/GW170814/v3/H-H1
_GWOSC_4KHZ_R1-1186741846-32.hdf5', 'https://www.gw-openscience.org/eventapi/jso
n/GWTC-1-confident/GW170814/v3/H-H1_GWOSC_4KHZ_R1-1186739814-4096.hdf5',
'https://www.gw-openscience.org/eventapi/json/GWTC-1-confident/GW170814/v3/L-L1_
GWOSC_4KHZ_R1-1186741846-32.hdf5', 'https://www.gw-openscience.org/eventapi/json
/GWTC-1-confident/GW170814/v3/L-L1_GWOSC_4KHZ_R1-1186739814-4096.hdf5',
'https://www.gw-openscience.org/eventapi/json/GWTC-1-confident/GW170814/v3/V-V1_
GWOSC_4KHZ_R1-1186741846-32.hdf5', 'https://www.gw-openscience.org/eventapi/json
/GWTC-1-confident/GW170814/v3/V-V1_GWOSC_4KHZ_R1-1186739814-4096.hdf5']
```

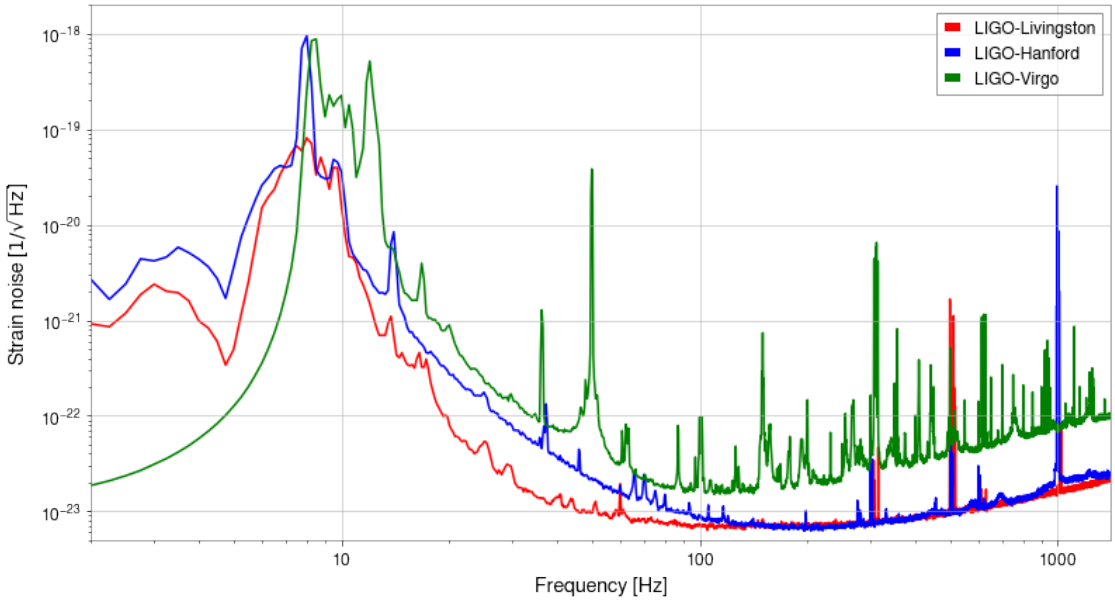
2 The GW170814 event was detected by Livingston, Hanford and Virgo detectors

[57]:
```
from gwosc.datasets import event_gps
from gwpy.timeseries import TimeSeries

gps_GW170814 = event_gps('GW170814')
print("GW170814 GPS:", gps_GW170814)

data_GW170814_L1 = TimeSeries.fetch_open_data('L1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data L1 =",data_GW170814_L1)
data_GW170814_L1_asd = data_GW170814_L1.asd(fftlength=4, method="median")

data_GW170814_H1 = TimeSeries.fetch_open_data('H1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data H1 =",data_GW170814_H1)
data_GW170814_H1_asd = data_GW170814_H1.asd(fftlength=4, method="median")
```

```
data_GW170814_V1 = TimeSeries.fetch_open_data('V1', int(gps_GW170814)-600,␣
 ↪int(gps_GW170814)+600, cache=True)
print("GW170814 data V1 =",data_GW170814_V1)
data_GW170814_V1_asd = data_GW170814_V1.asd(fftlength=4, method="median")

import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(13,7))
# and plot using standard colours
ax.plot(data_GW170814_L1_asd, label='LIGO-Livingston', color='red')
ax.plot(data_GW170814_H1_asd, label='LIGO-Hanford', color='blue')
ax.plot(data_GW170814_V1_asd, label='LIGO-Virgo', color='green')

ax.set_xlim(2, 1400)
ax.set_ylim(5e-24, 2e-18)
# update the Livingston line to use standard colour, and have a label

ax.set_yscale('log')
ax.set_xscale('log')
ax.set_ylabel(r'Strain noise [$1/\sqrt{\mathrm{Hz}}$]')
ax.set_xlabel(r'Frequency [Hz]')

ax.legend()
plt.show()
```

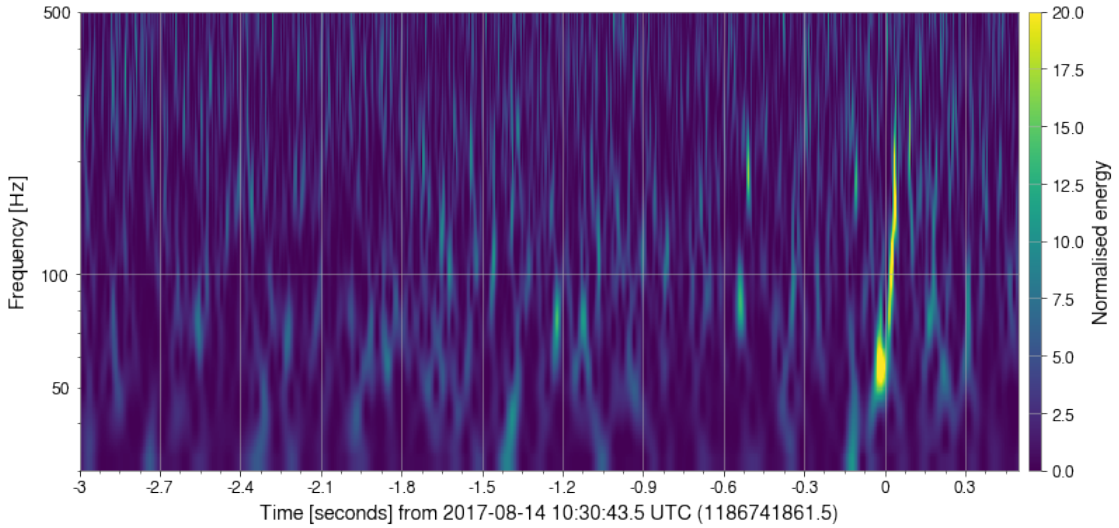
```
GW170814 GPS: 1186741861.5
GW170814 data L1 = TimeSeries([6.71509695e-20, 6.97313000e-20, 6.22758834e-20,
…,
            5.42646307e-20, 6.40301087e-20, 5.42822595e-20]
           unit: dimensionless,
           t0: 1186741261.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
GW170814 data H1 = TimeSeries([-7.38466061e-19, -7.37551069e-19,
-6.95249542e-19,
            …, -7.23194401e-19, -7.48978223e-19,
            -7.05070524e-19]
           unit: dimensionless,
           t0: 1186741261.0 s,
           dt: 0.000244140625 s,
           name: Strain,
           channel: None)
GW170814 data V1 = TimeSeries([-1.48413879e-19, -1.48239654e-19,
-1.47302345e-19,
            …, -2.80828691e-19, -2.90640523e-19,
            -3.14884932e-19]
```

```
unit: dimensionless,
t0: 1186741261.0 s,
dt: 0.000244140625 s,
name: Strain,
channel: None)
```



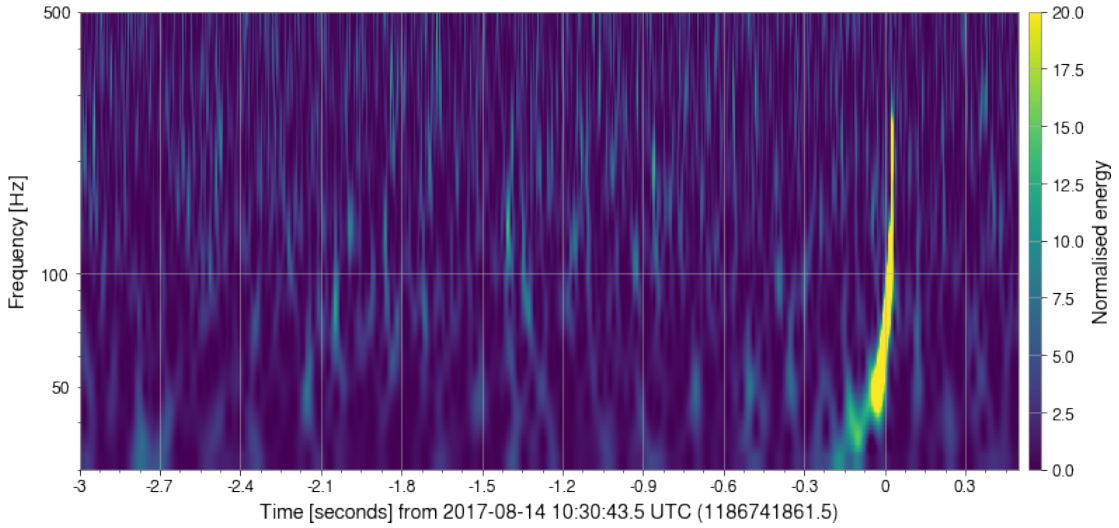
[82]:
```
segment = (int(gps_GW170814) - 10, int(gps_GW170814) + 2)
hdata = TimeSeries.fetch_open_data('H1', *segment, verbose=True, cache=True)

#-- Use OUTSEG for small time range
hq2 = hdata.q_transform(frange=(30, 500), qrange=(5, 10),␣
 ↪outseg=(gps_GW170814-3,gps_GW170814+0.5))
plot = hq2.plot()
ax = plot.gca()
ax.set_epoch(gps_GW170814)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")

plot.colorbars[0].mappable.set_clim(0,20)
```

```
Fetched 1 URLs from www.gw-openscience.org for [1186741851 .. 1186741863))
Reading data… [Done]
```



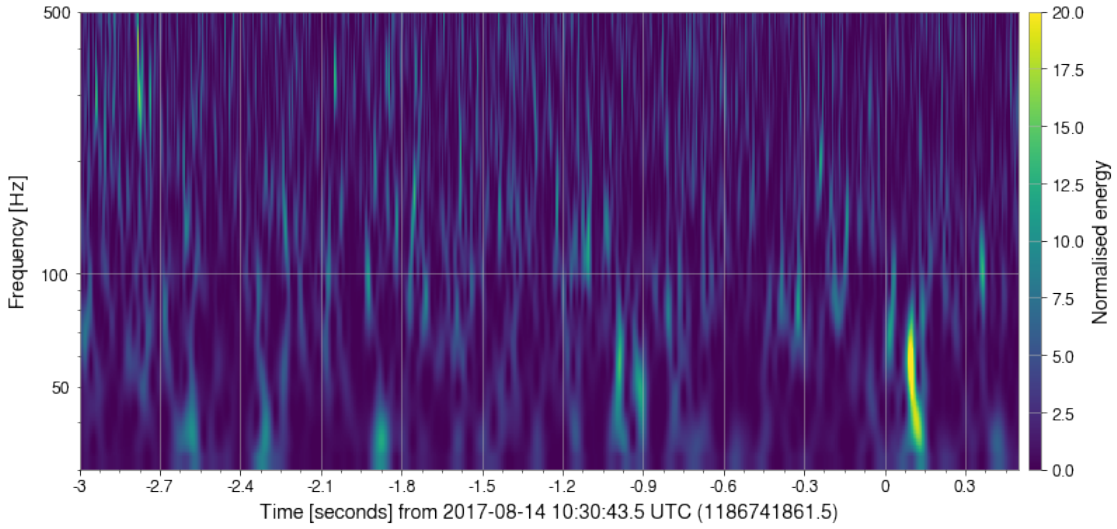
[86]:
```
ldata = TimeSeries.fetch_open_data('L1', *segment, verbose=True, cache=True)

lq2 = ldata.q_transform(frange=(30, 500), qrange=(5, 10),␣
 ↪outseg=(gps_GW170814-3,gps_GW170814+0.5))
plot = lq2.plot()
ax = plot.gca()
ax.set_epoch(gps_GW170814)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")

plot.colorbars[0].mappable.set_clim(0,20)
```

```
Fetched 1 URLs from www.gw-openscience.org for [1186741851 .. 1186741863))
Reading data… [Done]
```



[91]:
```
vdata = TimeSeries.fetch_open_data('V1', *segment, verbose=True, cache=True)

vq2 = vdata.q_transform(frange=(30, 500), qrange=(5, 10),␣
 ↪outseg=(gps_GW170814-3,gps_GW170814+0.5))
plot = vq2.plot()
ax = plot.gca()
ax.set_epoch(gps_GW170814)
ax.set_yscale('log')
ax.colorbar(label="Normalised energy")

plot.colorbars[0].mappable.set_clim(0,20)
```

```
Fetched 1 URLs from www.gw-openscience.org for [1186741851 .. 1186741863))
Reading data… [Done]
```



[ ]: