GRAVITATIONAL WAVES

By Jeremy Timog and Elena Penner

Motivations, Methods and More!

Motivations - Understanding the properties of black hole mergers.

Methods - Using the gravitational wave python package, and data from the gravitational wave transient catalogue from the gravitational wave open science center.

Assumptions - The black holes orbit each other at the speed of light just before merging

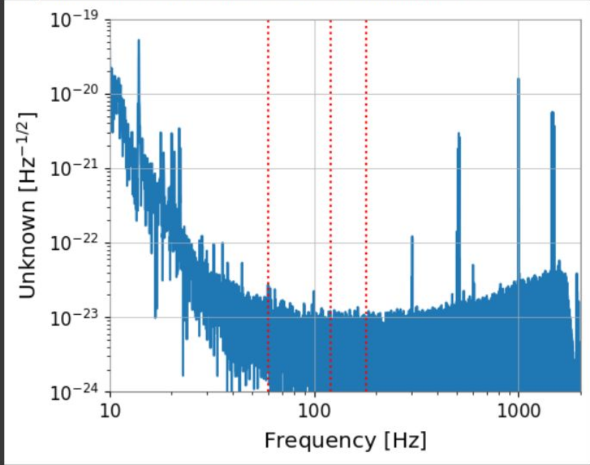
```
! pip install gwpy # first installing packages needed to go through the LIGO data
```

```
Collecting gwpy
  Downloading gwpy-3.0.10-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: astropy>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from gwpy) (6.1.4)
Collecting dateparser>=1.1.4 (from gwpy)
  Downloading dateparser-1.2.0-py2.py3-none-any.whl.metadata (28 kB)
Collecting dqsegdb2 (from gwpy)
  Downloading dqsegdb2-1.2.1-py3-none-any.whl.metadata (3.6 kB)
Collecting gwdatafind>=1.1.0 (from gwpy)
  Downloading gwdatafind-1.2.0-py3-none-any.whl.metadata (3.9 kB)
Collecting gwosc>=0.5.3 (from gwpy)
  Downloading gwosc-0.7.1-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: h5py>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from gwpy) (3.12.1)
Collecting ligo-segments>=1.0.0 (from gwpy)
  Downloading ligo-segments-1.4.0.tar.gz (51 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.0/51.0 kB 3.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ligotimegps>=1.2.1 (from gwpy)
```

```python
t0 = 1264316116.4 #GPS time here GW200129_065458

from gwpy.timeseries import TimeSeries #importing raw data
hdata = TimeSeries.fetch_open_data('H1', 1264316100.4, 1264316132.4) # taking hanford data around the event
#Times listed after H1 are 16 seconds before and after the GPS time
```

```python
import matplotlib.pyplot as plt # importing matplotlib so we can plot a graph

# -- Plot ASD
fig2 = hdata.asd().plot()
plt.xlim(10,2000) # upper and lower bounds for the x axis
ymin = 1e-24
ymax = 1e-19
plt.ylim(ymin, ymax) # upper and lower bounds for the y axis
plt.vlines(60,  ymin, ymax, linestyle="dotted", color="red") # dotted lines to show where the notches will go
plt.vlines(120, ymin, ymax, linestyle="dotted", color="red")
plt.vlines(180, ymin, ymax, linestyle="dotted", color="red")
```

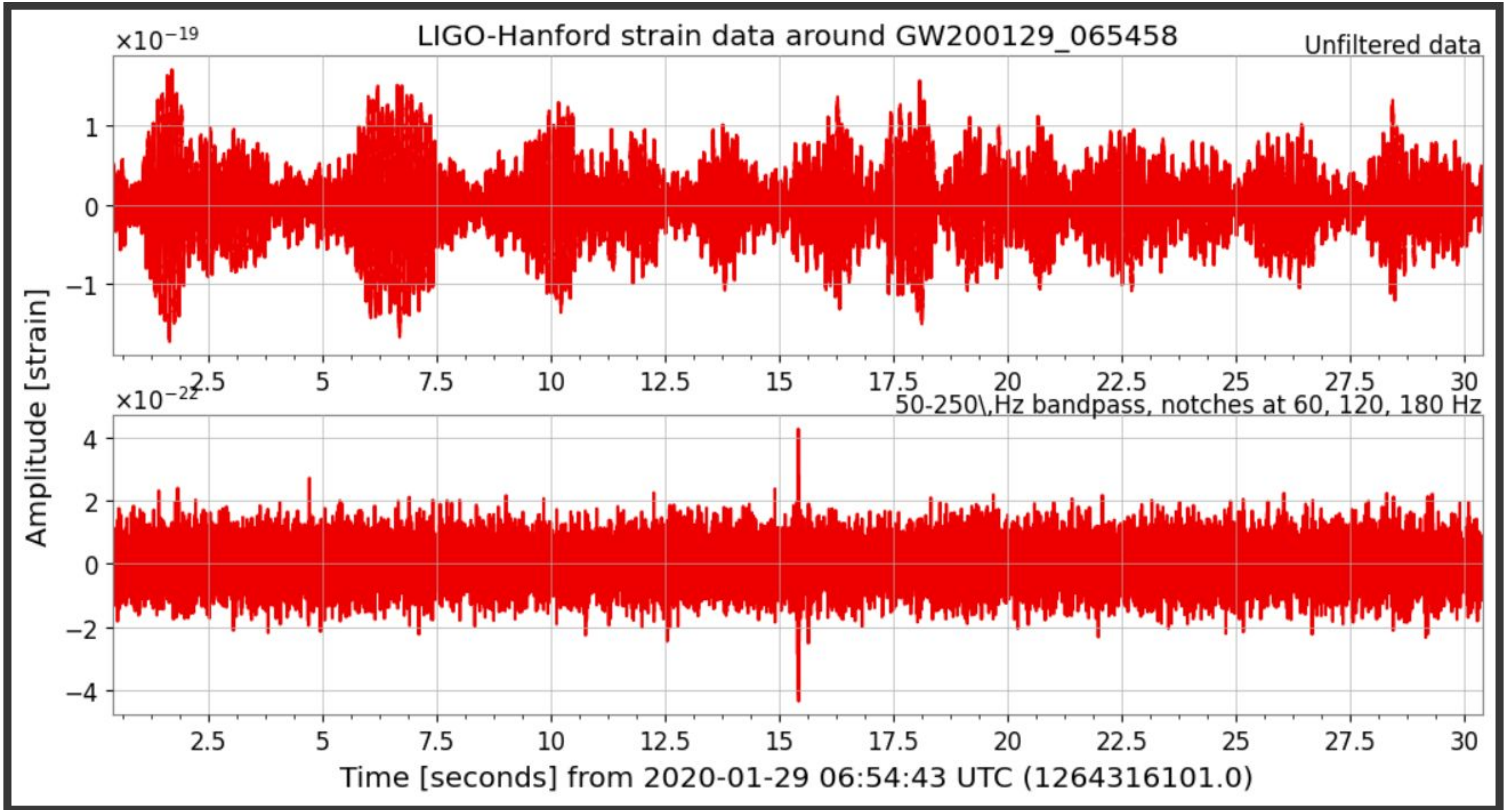
```
<matplotlib.collections.LineCollection at 0x7d5c6f8cf340>
```



```python
from gwpy.signal import filter_design # Import a way to filter out noise from the frequency we want
bp = filter_design.bandpass(50, 250, hdata.sample_rate) # frequencies below 50 and higher than 250 will be filtered out

notches = [filter_design.notch(line, hdata.sample_rate) for
           line in (60, 120, 180)] # these are peaks we can account for and will filter out separately

zpk = filter_design.concatenate_zpks(bp, *notches) #putting the two filters together

hfilt = hdata.filter(zpk, filtfilt=True) # applying the filters
```

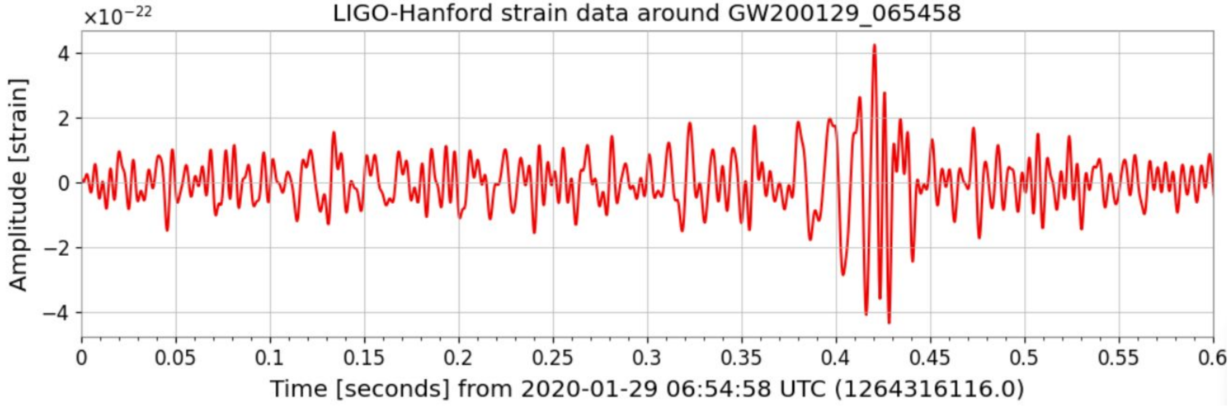
```python
hdata = hdata.crop(*hdata.span.contract(1)) # crop data on both ends that has been messed up by the filter
hfilt = hfilt.crop(*hfilt.span.contract(1))
```

```python
from gwpy.plot import Plot #import plot functions from gravitational wave python package
plot = Plot(hdata, hfilt, figsize=[12, 6], separate=True, sharex=True,
            color='gwpy:ligo-hanford') # formatting and creating the plot
ax1, ax2 = plot.axes # setting the axes of both plots equal
ax1.set_title('LIGO-Hanford strain data around GW200129_065458')
ax1.text(1.0, 1.01, 'Unfiltered data', transform=ax1.transAxes, ha='right')
ax1.set_ylabel('Amplitude [strain]', y=-0.2)
ax2.set_ylabel('')#titling the axes
ax2.text(1.0, 1.01, r'50-250\,Hz bandpass, notches at 60, 120, 180 Hz',
         transform=ax2.transAxes, ha='right')
plot.show() #showing plot
```



LIGO-Hanford strain data around GW200129_065458
Unfiltered data
×10⁻¹⁹
50-250\,Hz bandpass, notches at 60, 120, 180 Hz
×10⁻²²
Amplitude [strain]
Time [seconds] from 2020-01-29 06:54:43 UTC (1264316101.0)

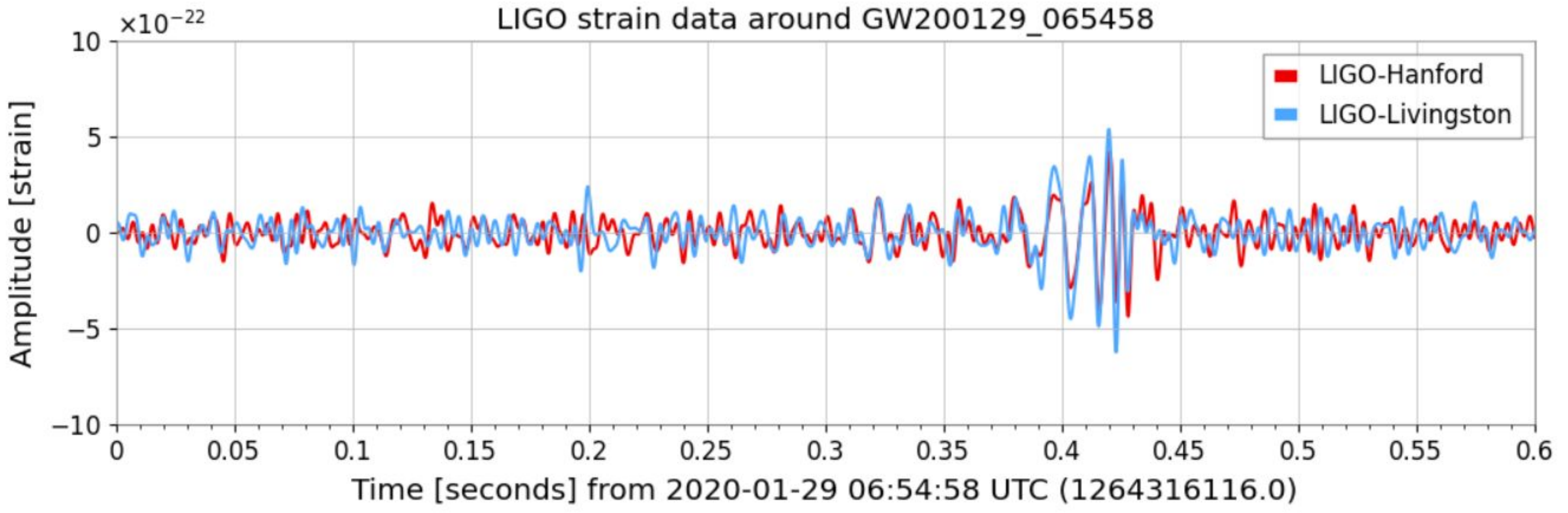
```
import matplotlib.pyplot as plt # importing the package that lets us plot

plot = hfilt.plot(color='gwpy:ligo-hanford')
ax = plot.gca()
ax.set_title('LIGO-Hanford strain data around GW200129_065458') # titling the set
ax.set_ylabel('Amplitude [strain]') # titling the y axis
ax.set_xlim(1264316116,1264316116.6) #limiting the x axis
ax.set_xscale('seconds', epoch=1264316116) #scaling the x axis
plot.show() #showing the data plot

x_val = plt.gca().lines[0].get_xdata()
y_val = plt.gca().lines[0].get_ydata()# stores the x and y values for later
```



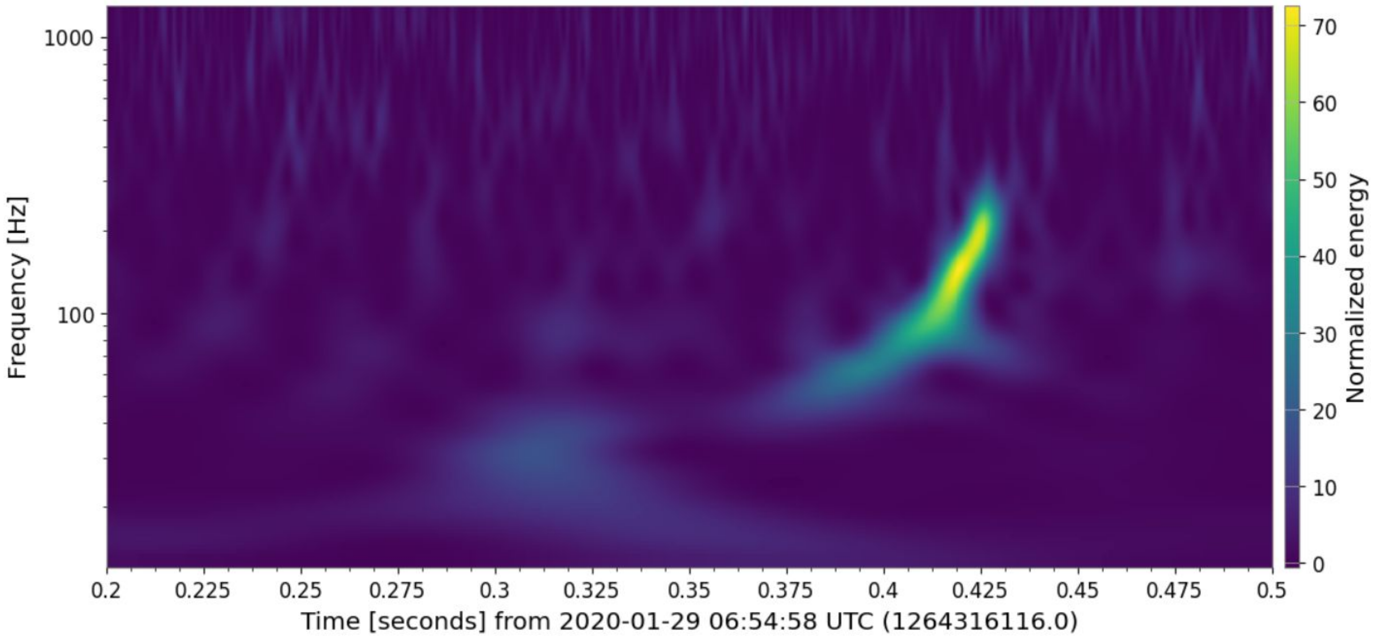
```
#repeating the process with the Livingston data
ldata = TimeSeries.fetch_open_data('L1', 1264316100.4, 1264316132.4)
lfilt = ldata.filter(zpk, filtfilt=True)
```

```
lfilt.shift('-5.5ms') #shifting doesn't help
lfilt *= -1 #flipping the data since the ligo detectors are not angled the same way
```

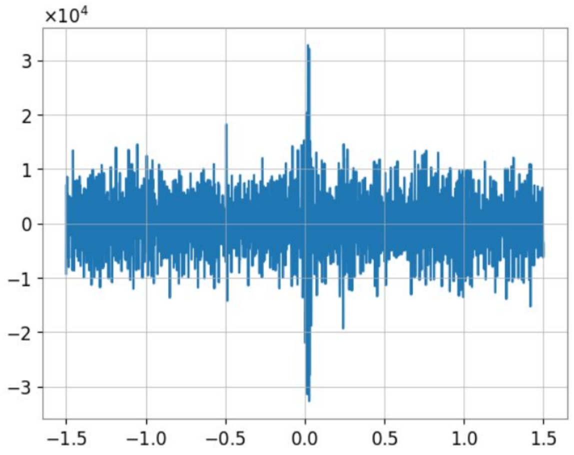
```
plot = Plot(figsize=[12, 4]) #sizing the plot
ax = plot.gca()
ax.plot(hfilt, label='LIGO-Hanford', color='gwpy:ligo-hanford')
ax.plot(lfilt, label='LIGO-Livingston', color='gwpy:ligo-livingston') #putting the hanford and livingston data into the plot
ax.set_title('LIGO strain data around GW200129_065458') #titling the plot
ax.set_xlim(1264316116, 1264316116.6) #x axis limits
ax.set_xscale('seconds', epoch=1264316116) #epoch =t0
ax.set_ylabel('Amplitude [strain]') # y axis label
ax.set_ylim(-1e-21, 1e-21) #y axis limits
ax.legend()
plot.show() # show the plot
```



```python
dt = 0.2  #-- Set width of q-transform plot, in seconds
hq = hfilt.q_transform(outseg=(t0-dt, t0+0.1))
fig4 = hq.plot()
ax = fig4.gca()
fig4.colorbar(label="Normalized energy") # show what colors represent what normalized energies
ax.grid(False)
ax.set_yscale('log') #y axis log scale
```



```python
from scipy.io.wavfile import write # import part of a package that lets us listen to the data
import numpy as np #importing numpy for plots

amplitude = np.iinfo(np.int16).max #Putting a max limit on the amplitude

ind = np.where((x_val < (t0+1.5)) & (x_val > (t0-1.5))) #cropping the x values
y = y_val[ind] #only having y values according to the limited x values
# y = y**3
y = y / np.max(y)
plt.plot(x_val[ind] - t0, (np.array(y) * amplitude).astype(np.int16)) #plotting the data
```



```python
fs = int(1 / np.median(np.diff(np.array(x_val[ind] - t0))))
print("fs = ", fs) #print out frequencies
write("example.wav", fs, (np.array(y) * amplitude).astype(np.int16))
```

```
fs =  4096
```

```python
from google.colab import files #import files from google colab
files.download("example.wav") #showing what the data sounds like
```

```python
#mass estimate
import astropy.constants as const #import astronomy constants
import astropy.units as u #import astronomy units
import numpy as np
dt = 0.004*u.s
ratio = dt * const.c / ( 2 * np.pi * 3e3 * u.m )
print(ratio)
```

```
63.61793545649256
```

```python
wavemass = (ratio - 60.2) * u.solMass
E = wavemass * const.c**2
print(E.to(u.J))
```

```
6.1081708144641836e+47 J
```

Results and Conclusions!

- Our calculated total mass of the black holes before merging was 63.6 solar masses.
- The database listed the final merged black hole mass as 60.2 solar masses.
- We found that the estimation of released energy due to mass lost in the merging event was 6e47 Joules. That's a lot of energy!
- The distance was 191 km between the black holes.
- Our conclusion based on these results is that there was 3.4 solar masses lost in the process of the two black holes merging.