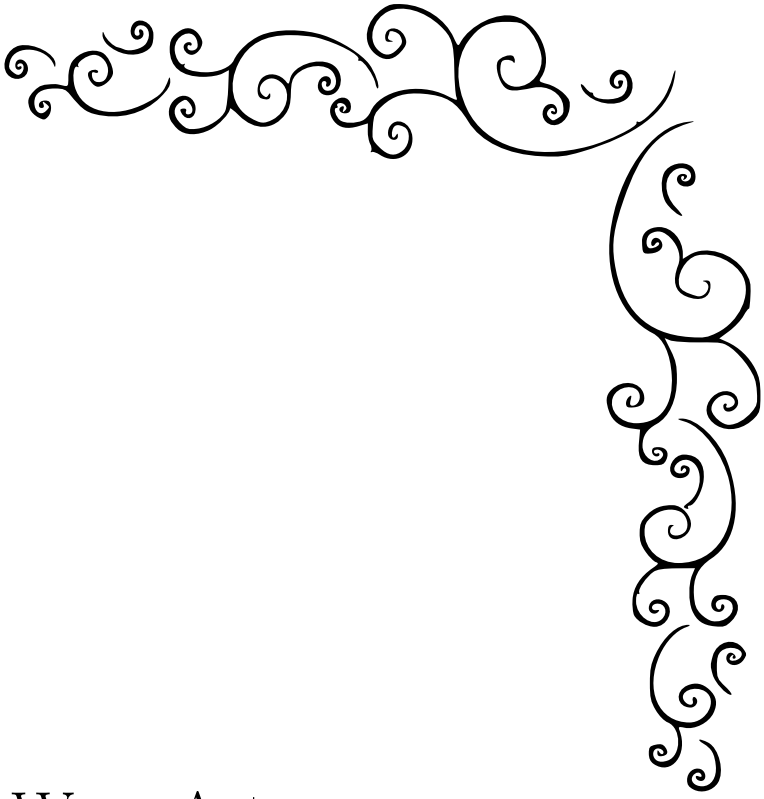


Gravitational Wave Astronomy

Saqlain Afroz

July 2022

Contents

1 Data Quality

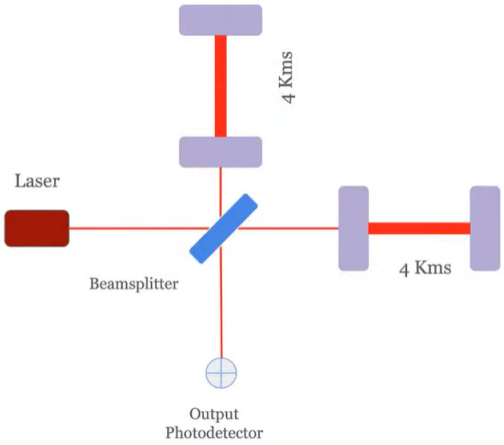
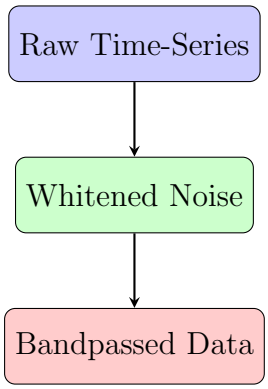


Figure 1: Detector diagram

A gravitational wave interacts with the system, changes the length of the arms and then we read the strain in the gravitational wave channel which is sampled at around 16 kHz.

$$h = \frac{L_x - L_y}{L} \tag{1}$$

1.1 Strain Data: Post Processing



The data we have, i.e. the raw time series, contains a lot more power at lower frequencies than at higher frequencies and that kind of overshadows or dwarfs the feature at higher frequencies, so we whiten the data and by that we mean, we normalize the power in all these frequency bins so that the features at higher frequencies are also better visible. Once we have whiten the data, we bandpass it. So we take a lower frequency cut-off and we say that we are only interested in whatever is going on between these two thresholds and that we are not interested in anything below 10 Hz or anything above 300 Hz. This is done because we already expect within this frequency range.

Q-transform (After Bandpassing)

- Q-transform allows us to visualize the data in time frequency space.
- It helps us in understanding Noise morphology, noise identification and classification, potential correlations with other parts of the instruments.

1.2 LIGO Data Quality

Sensitivity plot tells us how good a job we are doing or how sensitive our detector is and the lower it is the better it is. Then we can discover more GW and so we do a lot of

things to improve the sensitivity both during the observing run and after it, but mainly after the observing run. So once an observing run ends we make a lot of changes that improve the overall sensitivity of the detector and these changes include:

- Increasing the laser power
- New test mass mirrors
- Employing light squeezing - this helps with the quantum noise
- Fixing sources of noise
- A ton of small improvements

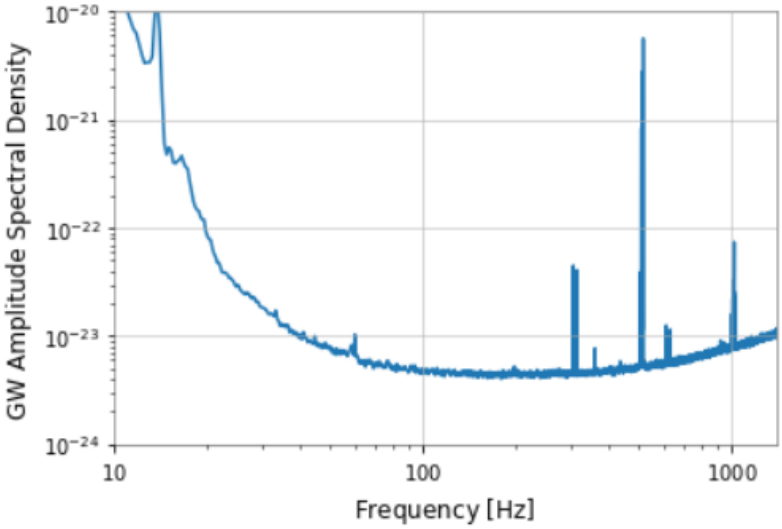


Figure 2: GW Amplitude Spectral Density

In this plot we have seismic noise that dominate at low frequency and that diminishes our ability to get good data at those frequencies and then we have shot noise at higher frequency and shot noise is a type of quantum noise which is due to the uncertainty in the number of photons that hit the photo-detector and that's why we have light squeezing which helps with reducing the shot noise.

Shot noise and Seismic noise are Gaussian noise, these are stationary, they don't change over time.

But we can also have non-Gaussian or non-Stationary noise, which we call Noise Transients.

What are they?
Excess power over a short duration, also known as glitches.

How do we detect them?
Omicron tools - it takes in the strained data from the primary gravitational wave channel but also from other auxiliary channels and it finds these excess power in the strain data and we call them omicron triggers.

What causes them?
Ground motion, thunderstorms, change in humidity etc

- We have different types of transients based on how they look in the time frequency plane.

Why these transients are bad?
These transients can mask a real GW signal, can mimic a signal, reduce our confidence in the detection, reduce the astrophysical range and introduce problems for the parameter estimation.

Where do these transients originate?
Auxiliary channels, so many of them !!!
We have a channel that measures how much the test mass mirror is moving with respect to suspension cage, we have auxiliary channel that looks at beam alignment, we have auxiliary channel that tells us how much the ground is moving at different locations in the XYZ directions. And sometime something goes wrong in one of these channels and if we are really unfortunate, it may permeate to the GW channel and create problems for us.

What to do about the transient noise?

- Identify the noise - figure out its feature, what is its duration, how frequently it happens, or does it have any correlation with trucks coming to the site or with thunderstorms.
- Look for the potential correlation with the auxiliary channels and subsystems.
- Performs tests to simulate the noise - We may shake a component of the detector to see if that would actually create noise in the GW channel and if that happens we are in luck and we can fix that source of noise.
- Fix the source of noise to reduce it or eliminate it.
- Develop Vetoes.

To do all these we have **Detector Characterization Tools (Detchar Tools)** or **Data Quality Tools**

- Gravity Spy
- Q Transform
- Hveto
- Gwdetchar Scattering
- Omicron
- Summary pages

1.3 Q - Transform

- Visualizing the glitch morphology in time - frequency plane.

It starts with taking the data and it projects the data on multi-resolution basis which is parameterized by time, frequency and 'q', which is a constant ratio of duration to bandwidth, so we can also think of 'q' as an aspect ratio.

- So these Q-transform creates these multiple spectrogram with constant q and then it optimizes over these plane and picks a plane which contains the loudest time-frequency tile and once it picks a plane it means it has picked a 'q', so we have what it returns at the end is a high-resolution spectrogram with a specific 'q' value and with a specific event time and frequency.

- gwpy timeseries.q_transform returns a high-resolution spectrogram with a specific 'q' value and with a specific event time and frequency.

1.4 Gravity Spy

It is an image recognition algorithm which is based on convolutional neural network. It helps us classify 23 transient noise in LIGO during 3rd observation run.

Omicron tells us which day we have a lot of transients but gravity spy tells us which day we had a specific type of noise transients. So, for example if there is a day with a lot of trains passing by LIGO Livingston or it had a lot of seismic wave in 1-3 Hz band and that's why probably we had a lot of *fast scattering*. So, it helps us correlate the type of transients that we have with the conditions that may be present at the detector.

1.5 Hardware Injections (HI)

- HI is when we inject a signal that physically actuates one of the test mass.
- A set of sine-Gaussians with different frequency and amplitude.
- Establish safety of vetoes, safe and unsafe auxiliary channels.

Safe auxiliary channels:

- Does not respond to hardware injections in primary GW channel.
- Can be used to veto primary GW channel transients.

2 Physics and Astronomy with Coalescences

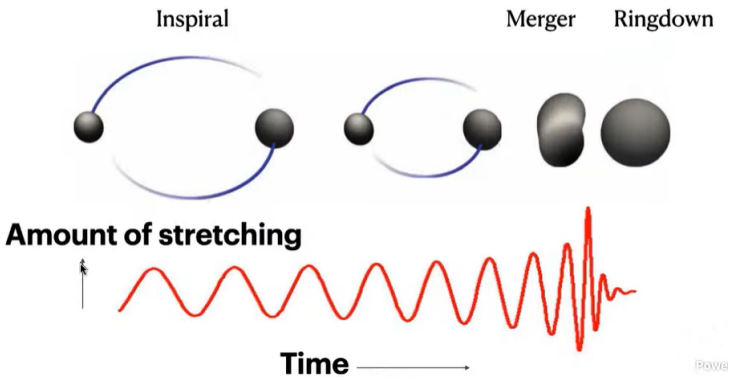


Figure 3: Phases of a merger and dependence of GW

$$f_{\text{gw}} = 2f_{\text{orbital}} \xrightarrow{\text{depends on}} \text{Mass, spin} \tag{2}$$

As these GWs are carrying away this orbital energy and angular momentum, these two compact objects are getting closer together so the f_{orbital} is increasing and so the f_{gw} increases as well. The GW also gets louder and louder as the two compact objects get closer together. Eventually the compact objects merge and then we are left with black hole ringdown with a a characteristic signal that depends on the mass and spin of our final black hole.

2.1 GW encode source properties like

- How big is each black hole or neutron star?
- How fast are they spinning?
- Where and when did they merge?
- How squishy are neutron stars?

2.2 Learning about the full population of compact objects

- A population model describes the distribution of masses, spins etc across multiple events.
- Example : Fit a power law to black hole masses.
- Population parameters : power-law slope, minimum BH mass, maximum BH mass.

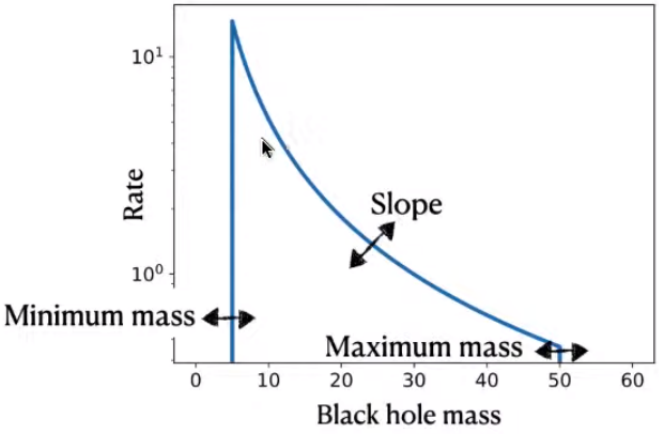


Figure 4: Population of BH

3 CBC searches and Matched Filtering

3.1 CBCs

- As object orbit they lose energy to GWs.
- The orbit shrinks and speeds up, releasing more energy to GWs.
- Frequency and amplitude of GWs increases monotonically.
- Creates a runaway process leading to inspiral merger.

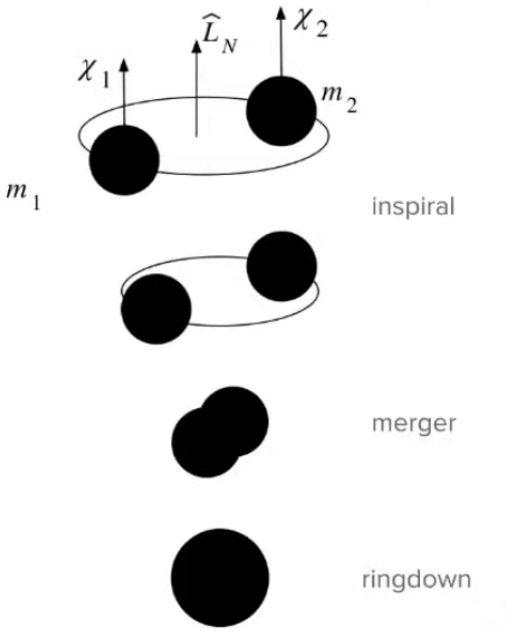


Figure 5: Black Hole Merger

- Waveforms model the inspiral, merger and ringdown of the binary.

- Intrinsic parameters - masses and spins
- Extrinsic parameters - sky position, inclination, time of arrival, phase

3.2 Matched Filtering

3.2.1 An introduction on the LIGO data

We assume that strain data from detectors consists of a possible signal and additive noise. We assume that the noise is stationary and Gaussian.

- **Stationary Noise** - the noise properties are constant in time.
- **Gaussian Noise** - the noise values follows distributions which we can transform to have 0 mean.

$$\underbrace{d[t_i]}_{\text{Strain data}} = \underbrace{n[t_i]}_{noise} + \underbrace{s[t_i]}_{signal} \tag{3}$$

3.2.2 Matched Filtering as cross-correlation

If we know what the signal looks like, we can use matched filtering to find signals in the data. Matched filtering is a correlation of a template waveform with the data. Matched Filtering output is the signal-to-noise ration SNR time-series.

$$\underbrace{\rho(t)}_{\text{SNR time series}} = \underbrace{2\int_{-\infty}^{\infty} df\ \tilde{h}(f)\ \tilde{d}(f)\ \exp 2\pi i f t}_{\text{(whitened) template waveform}} \qquad \text{(Frequency domain)}$$

$$= \underbrace{2\int_{-\infty}^{\infty} d\tau\ h(\tau)\ d(t+\tau)}_{\text{(whitened) data = noise + possible signal}} \qquad \text{(Time domain)}$$

We are taking a template and we slide it along the data and at every point in time we calculate the overlap between the template and the data and the output of that process is SNR or SNR time-series.

Peaks in the SNR time-series are used to identify signals in the data.

A coincident SNR peak in the data from multiple detectors increases the significance.

3.3 Constructing the template banks

- Matched filtering relies on knowing the shape of the signal.
- For CBC waveforms we can model the signals with template waveforms.
- We construct template banks of waveforms that vary over the intrinsic property.

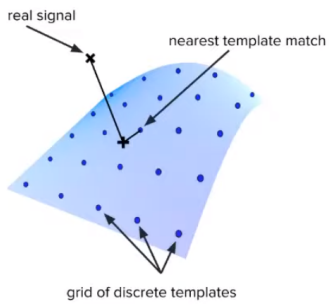


Figure 6: Grid of template bank

Waveforms depends on masses and spins of the source which is at the start unknown, i.e. we don't know what masses and spins we might necessarily be looking for, so we can't just pick one template and match filter the data and be done with it. Instead what we have to do is construct template banks, which means basically we choose a parameter space that we are interested in; so a range of masses and spins we are going to construct templates covering this entire range and then filter all of those templates.

How many templates do we need?

- If the signal perfectly matches the template, we will have an optimal SNR.
- Any mismatch causes an SNR loss.
- Construct banks with a dense grid of templates such that any signal will be close enough to the nearest template.
- $N_{\text{templates}} \sim \mathcal{O}(10^5 - 10^6)$

3.4 Revisiting the assumptions made about the LIGO data

In reality LIGO data is not well-modeled.

- non-stationary over short and long-time scales.
- non-Gaussian

The only characterisation of the LIGO noise is from observations.

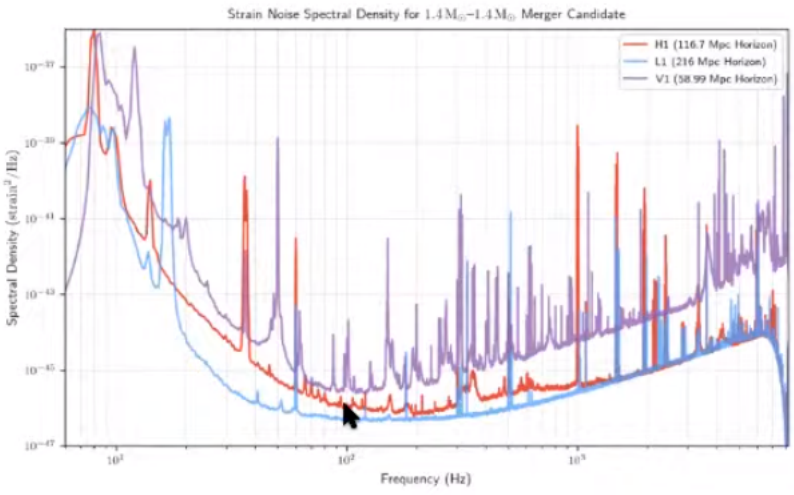


Figure 7: LIGO and VIRGO PSD for O3

Note : The PSD shows a measure of the sensitivity and how the noise varies over frequency bins. The lower the lines $\implies$ more sensitive the detector is.

3.4.1 Colored Noise

LIGO noise is not white, i.e. the power is not evenly distributed across frequencies.

$$\rho(t) = \underbrace{\int_{-\infty}^{\infty} d\tau \; \hat{h}(\tau) \; \hat{d}(t+\tau)}_{\text{whitened template and data}} \tag{4}$$

$$\hat{d}(\tau) = \int_{-\infty}^{\infty} df \frac{\tilde{d}f}{\sqrt{S_n(|f|)}} \exp 2\pi i f t \tag{5}$$

Note : The PSD varies across bins, so the data needs to be whitened before filtering, essentially scaled by the PSD in frequency space.

3.4.2 Non-Stationarity

- Noise properties can vary over long-timescales.
- We must continuously track the noise properties and update our estimate of the PSD.
- The snapshot of the PSD of O3 data may differ at different times.
- Noise properties can vary over short-timescales. And these short duration non-Gaussian artefacts in the data are glitches.
- Glitches can be a major problem for matched filtering searches.

3.5 Complementation in real searches

3.5.1 Matched filtering with real search pipelines

In practise, real search pipelines must come up with solutions to handle non-ideal noise properties. Filtering millions of templates over months of data or in real time is a huge computational burden.

To make this feasible, real search pipelines need to find ways to make the process more efficient. The three searches are - PyCBC, GstLAL and MBTA each use slightly different methods.

3.5.2 Matched Filtering with GstLAL - non-Gaussian data

Solution 1: Gating

- Remove stretches of data with large non-Gaussian features before matched filtering.
- But we have to be careful not to accidentally gate a real data.

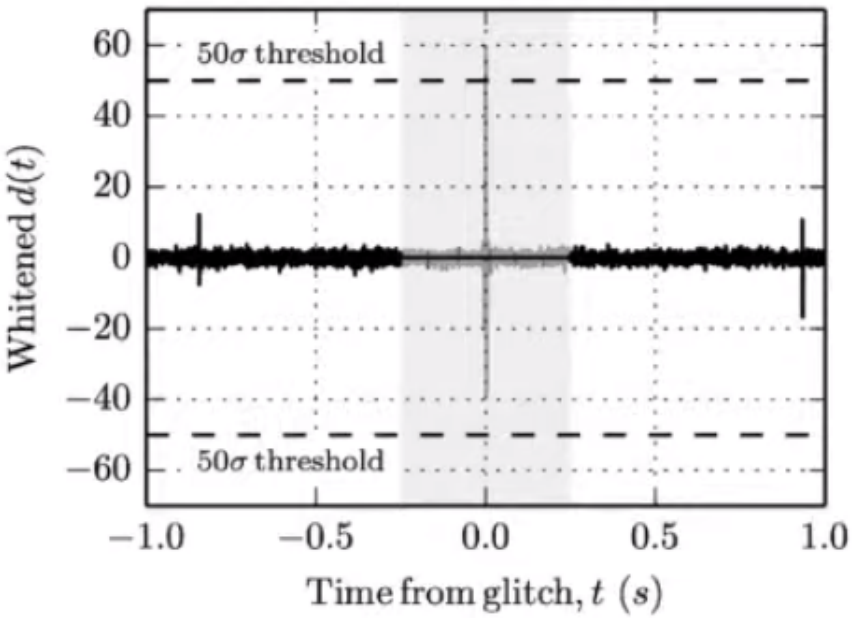


Figure 8: Whitened strain data with a glitch surpassing 50 standard deviations is gated by replacing ± 0.25 seconds around the glitch with zero.

Note : This can be problem for higher mass templates, i.e. very heavy BBH. They can have waveforms that are extremely loud and very short, so these are very easy to mix up with glitches.

Solution 2 : Signal Consistency Tests

- Matched filtering doesn't produce just an SNR peak, but a time-series of SNR data.
- Compare the SNR time-series shape to the predicted shape for a template waveform.

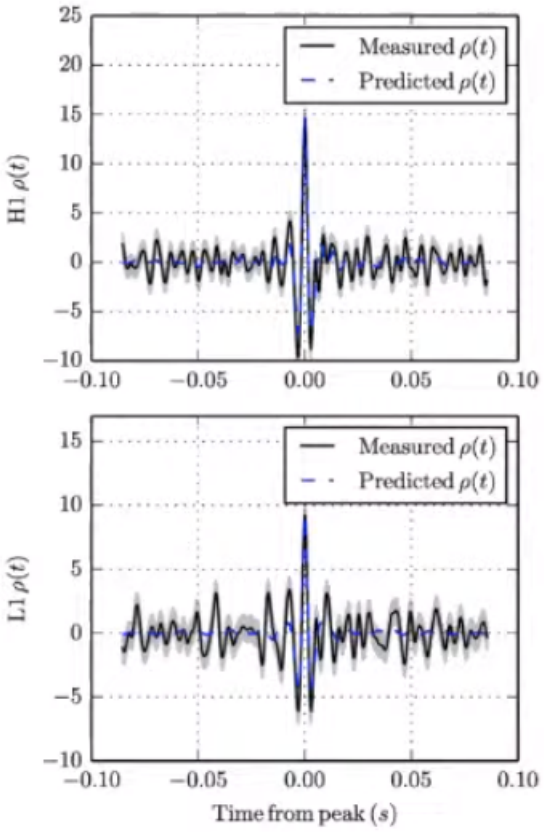


Figure 9: SNR time-series for a simulated signal compared to the predicted SNR from the auto-correlation of a template waveform.

Solution 3 : iDQ

- iDQ = integrated data quality
- Use machine learning and data from auxiliary channels to identify glitches.
- **Clean data** : boost significance of candidates.
- **Glitchy data** : reduce significance of candidates.

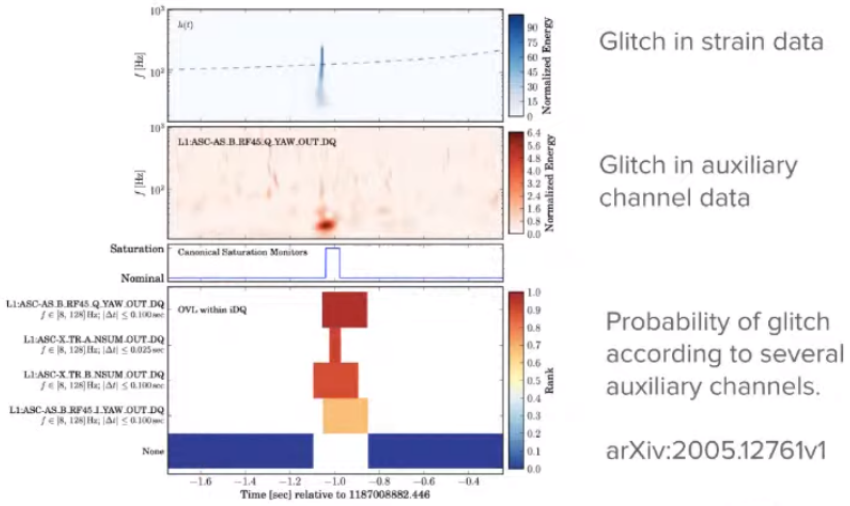


Figure 10: iDQ

3.5.3 How GstLAL handles this computational burden of matched filtering?

LLOID Method → Low Latency Online Inspiral Detection

This method was developed to enable real time matched filtering and even early warning matched filtering, which means that we are able to identify a signal in the data even before the merger occurs or before the coalescence time of the signal and this method consists of two steps, i.e.

$$\text{LLOID} = \underbrace{\boxed{\text{Time Slicing and Down Sampling}}}_{\text{Exploit signal properties}} + \underbrace{\boxed{\text{Singular Value Decomposition (SVD)}}}_{\text{Exploit template bank properties}} \tag{6}$$

The whole idea of this method is to :

- Reduce the computational burden
- Improve efficiency
- Lower the latency of the search

Step 1: Time Slicing and Down Sampling

- Take advantage of monotonic increase in signal frequency.
- Use lower sampling rate at earlier parts of the signal to avoid oversampling.
- In order to accurately represent any waveform we need to sample it at a sampling rate that at least twice as high as the largest frequency component in the waveform.

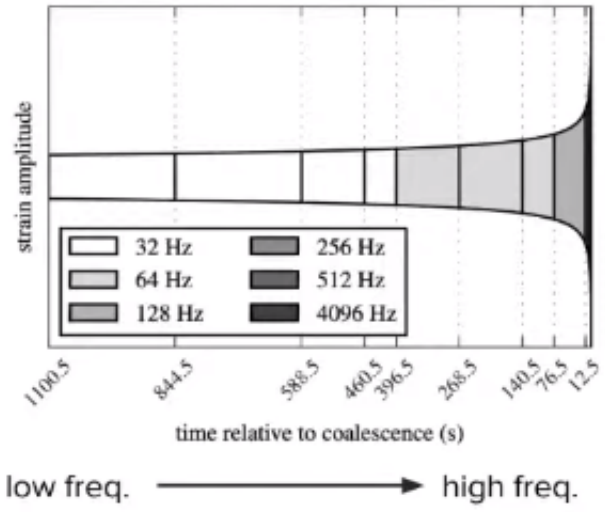


Figure 11: We can slice waveforms into frequency bands and then down-sample each one to save computations.

Step 2: Singular Value Decomposition (SVD)

- Our template banks are highly redundant.
- Transform to a reduced set of basis waveforms.
- Filter with the basis waveforms, then reconstruct the physical templates after filtering.

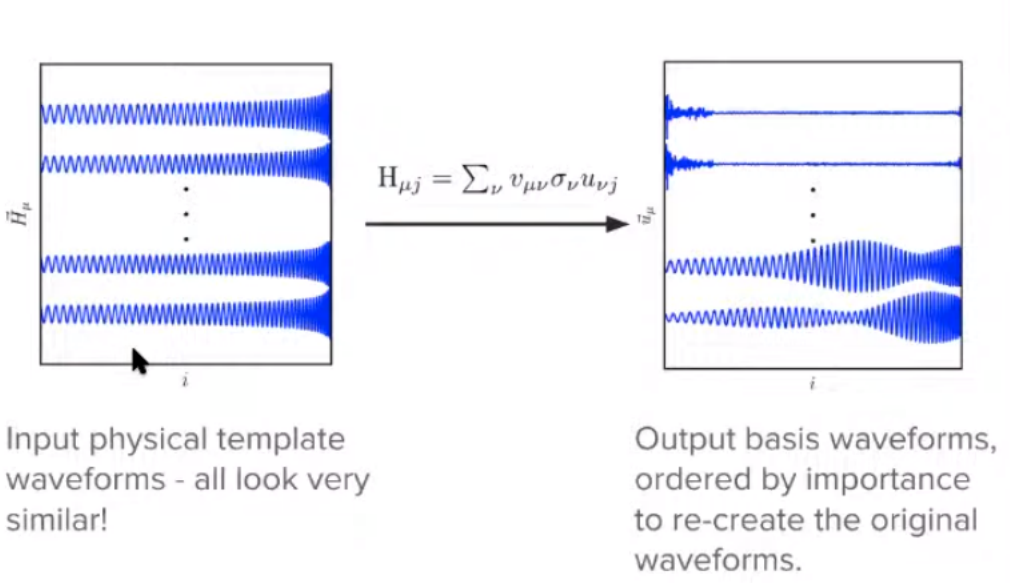


Figure 12: SVD

Remember that we reconstruct these template banks to be extremely dense over the full parameter space so that we can make sure that we are not losing very much SNR, but

because these template banks are so dense we end up with a lot of redundant templates, they all look really similar and maybe from template to template there's just a very small change and so if we filter all of these it would be a pretty redundant process. So, instead what we do is we can think each of these waveforms as a vector and the full set of these waveform vectors cover the space of our parameter space that we chose. What we can do is reduce the set of vectors to a set of basis waveforms, which still covers the space but all of the basis waveforms are linearly independent so there's a much smaller number of them. The set of basis waveforms can maybe at least factor of 10 smaller than original size of the bank. Thus the idea with SVD is that we can perform our matched filtering with these basis waveforms and then at the end we can reconstruct any of the original input templates by just doing a linear combination of the basis waveforms.

3.5.4 Matched Filtering with PyCBC and MBTA

- PyCBC and MBTA both regularly perform matched filtering in the frequency domain, whereas GstLAL does matched filtering in the time domain.
- Perform Fast Fourier Transform on blocks of data, then correlate the data with the template waveform.
- Typical block length = 2048 seconds

$$\tilde{d}[K] = \sum_{j=0}^{N-1} d[j] e^{-2\pi i j k/N} \rightarrow \text{DFFT} \tag{7}$$

$$\underbrace{\rho[j] = d[j] * h[j]}_{\text{Time domain requires convolution}} \quad \rightarrow \quad \underbrace{\tilde{\rho}[k] = \tilde{d}[k] \cdot \tilde{L}[k]}_{\text{Frequency domain is just a multiplication}} \tag{8}$$

So the idea here is that if we are doing matched filtering in the time domain, to get the output or SNR at any point in time, we have to do convolution, i.e. we have to do this integral given in Eq. 7, but if we transform to frequency domain, then that convolution just becomes a multiplication. So what PyCBC and MBTA do is that first take the time-series data from the detector, then they do the FFT on the data to get into the frequency domain. Then we can filter just by doing this multiplication and then FFT back to get the SNR time series out.

How do PyCBC and MBTA handle the non-Gaussian data?
PyCBC and MBTA both use gating and signal consistency tests similar to GstLAL.

- PyCBC and MBTA do not use iDQ, but they do use vetoes to reject triggers from times identified to have poor data quality.

For MBTA

Step 1: Re-filter high mass templates

- Heavy mass BBH systems can be mistaken for glitches due to their short duration.

- MBTA re-filters templates with duration < 3 seconds without gating first.
- This helps to avoid accidentally gating real signals.

Step 2: Multi-Banding Filter with waveforms split into two frequency bands:

- Low frequency band: $f_0 < f < f_c$
- High frequency band: $f > f_c$

Reconstruct the templates after filtering

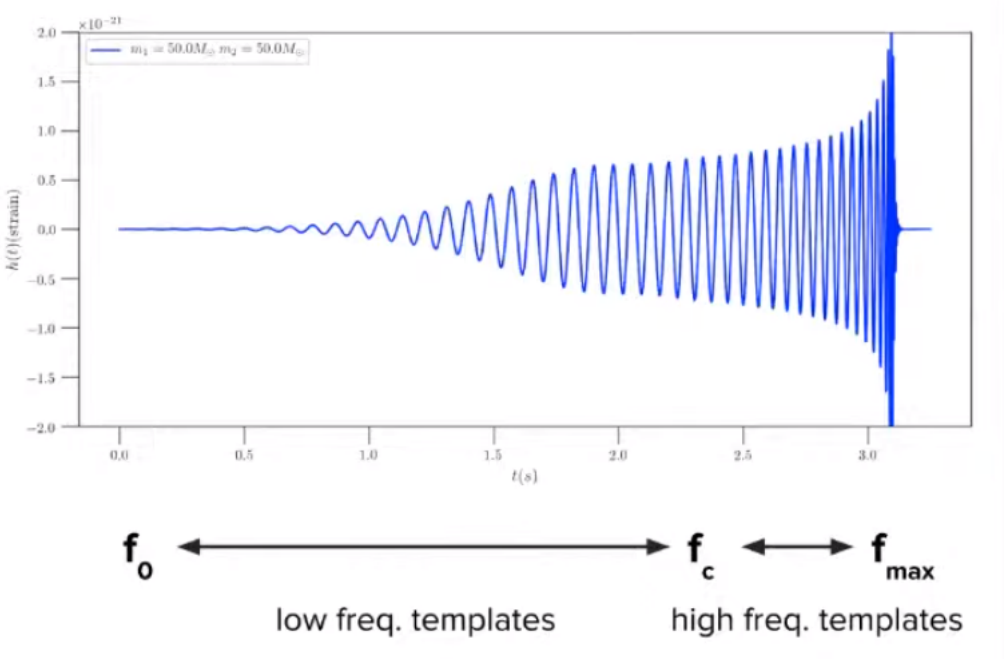


Figure 13: Multi Banding

Summary

CBC searches find GWs from merging black holes and neutron stars in LIGO/VIRGO strain data.

These searches use matched filtering - a correlation between the data and numerical or analytical models of the signal - to identify signals in the data.

Real matched filtering searches - PyCBC, GstLAL and MBTA - each employ similar but distinct methods to:

- implement matched filtering
- improve sensitivity in non-stationary, non-Gaussian data, and
- reduce computational cost

4 Estimating Parameter from Gravitational Wave Signals

The question is: How do we go from wiggles that we see in our detectors to the measurement of parameters of signals, say the mass of the Black Hole in Binary Black Hole System?

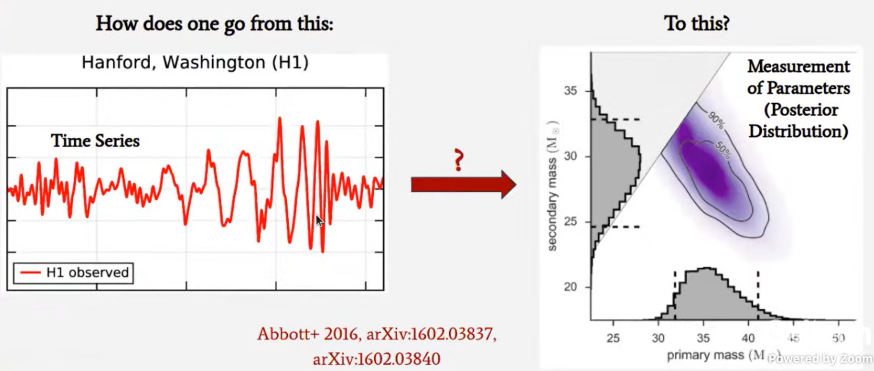


Figure 14: The Problem at hand

But why do we care about estimating parameters?

- Understand the Physics and Astrophysics eg. equation of state of neutron stars.
- Perform tests of general relativity.
-
- ... and for more interesting and impactful science.

4.1 Our Tools

- Models for the signal and noise.
- Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{9}$$

- Stochastic Samplers: emcee, dynesty etc. These are basically the tools that we use to efficiently find probability distributions that we actually want to measure.
- And some nice code to tie them together! ('Bilby', which is a parameter estimation package).

4.2 The Signal and the Noise

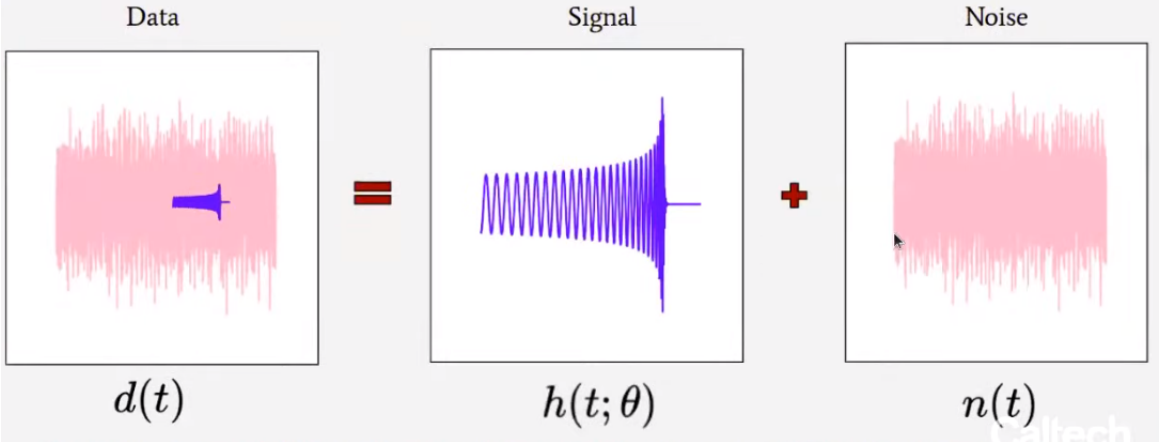


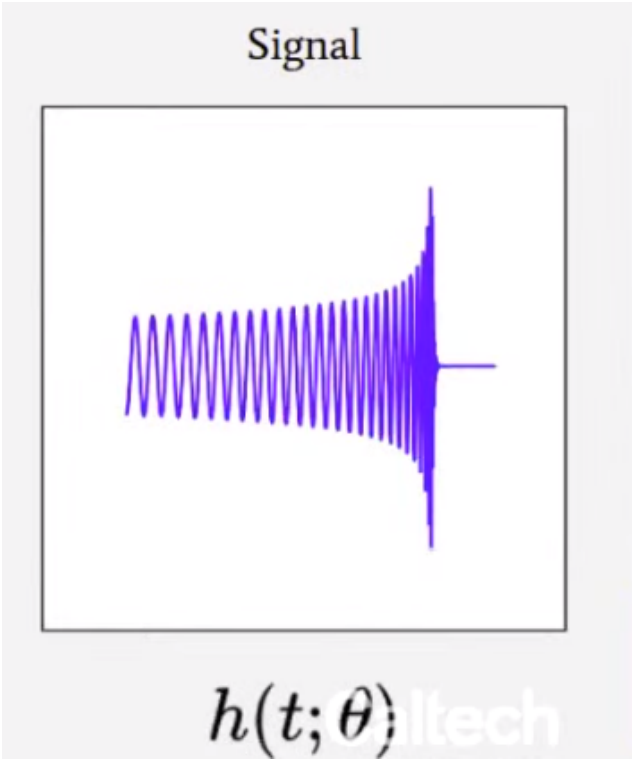
Figure 15: Data = Signal + Noise

4.2.1 The Signal

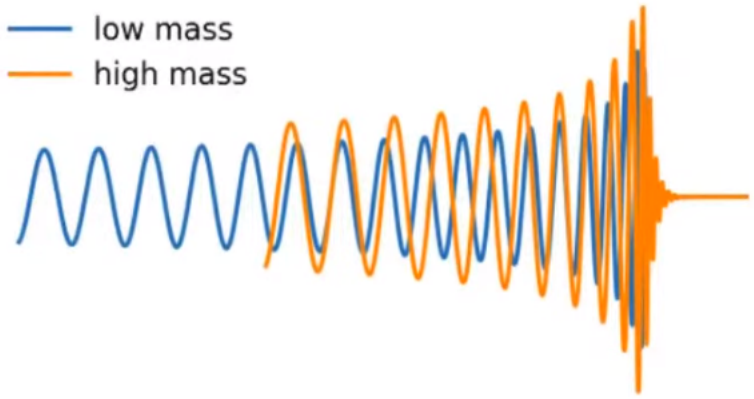
It has two polarizations. Described by 15 parameters in case of a binary black hole.

- Intrinsic parameters: masses, spins etc.
- Extrinsic parameters: distance, sky location (RA, Dec), polarization angle ψ etc.

$$h(\theta;t) = \sum_{\mathrm{p}=+,x} F_p \underbrace{(\mathrm{RA},\ \mathrm{Dec},\psi)}_{\text{these factors also known as Antenna Parameter Function}} h_p(\theta;t) \qquad (10)$$



- Higher mass $\implies$ shorter signal but more amplitude



- The red vector is perpendicular to the plane of the rotation of the binary and it is also the direction of orbital angular momentum of the binary. If the individual angular momentum of these black holes, also known as spins, are actually aligned to the angular momentum of the binary and that too in the positive sense, that actually increases the length of the waveform. If there is no spin at all the length of the waveform will decrease. And if the spins are anti-aligned, it will further decrease the waveform length.

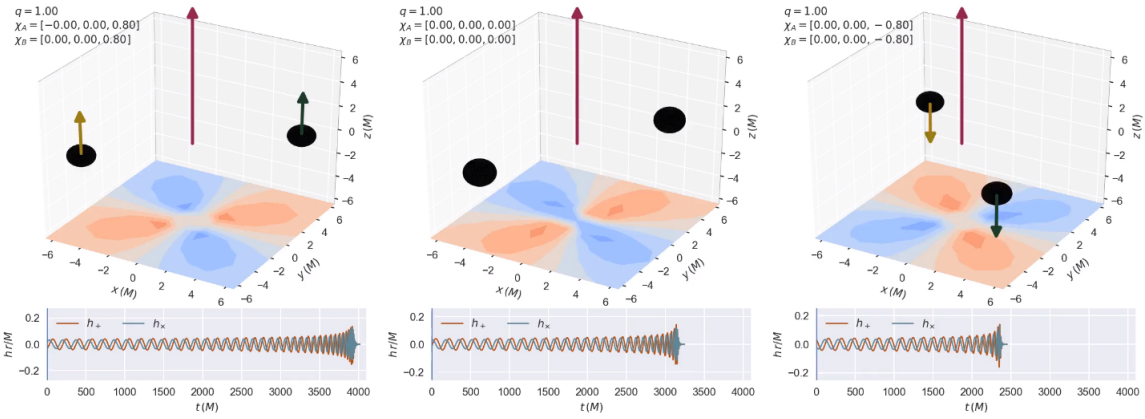
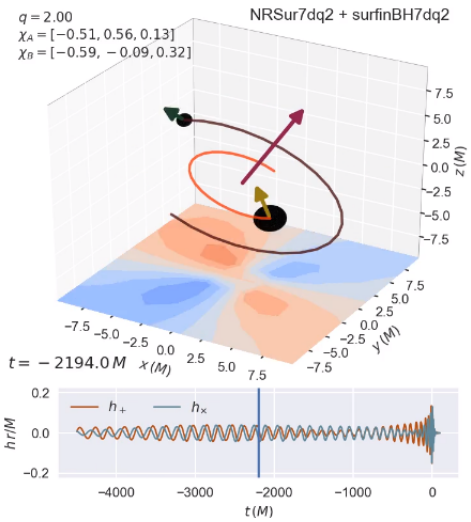


Figure 16: Spin alignment of BBH

So, binary black holes with anti-aligned spins will merge first, followed by those with zero spin.

- Higher aligned spin $\implies$ Longer Signal

One very interesting thing that happens with binaries, where the spin vector is no longer aligned with the orbital angular momentum, is that the orbit starts precessing and when that happens there are these discernible amplitude variations.

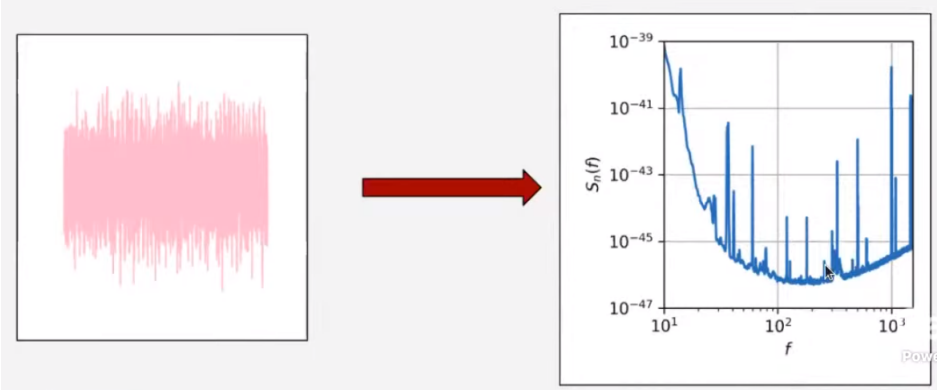


4.2.2 The Noise

- The noise properties are generally unknown and need to be calculated from the data.
- We shall make the following assumptions:
  - Stationary: The "noise properties" do not change with time.
  - Colored Gaussian: The "noise properties" are different at each frequency, but described by a Gaussian process.

$$n(t)$$

Under these assumptions noise properties are described by the Power Spectral Density (PSD), $S_n(f)$ in the frequency domain.



- PSD $\implies$ variance of noise at each frequency

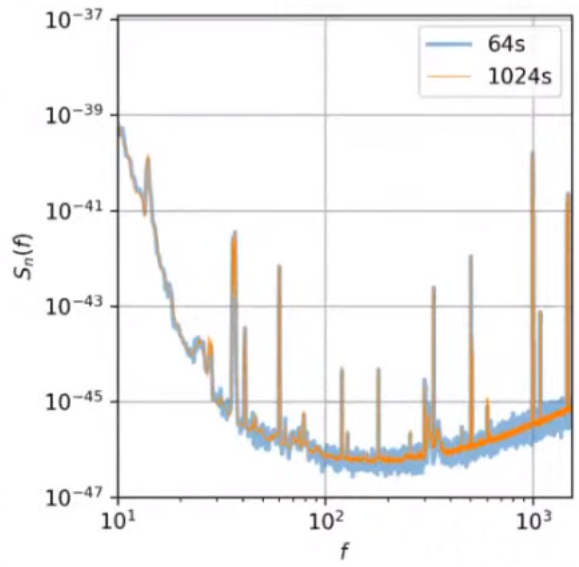
$$P(n_i) \sim \exp -2\delta f \frac{n_i^2}{S_n(f_i)} \tag{11}$$

where, $P(n_i)$ is the probability of getting a noise sample, n_i at any frequency f_i. And by variance we mean it in the sense of Gaussian distribution, so there's a mean of the noise, which is taken to be zero and the variance of the noise is basically given by being proportional to the power spectral density.

How do we estimate PSD from the data?

- To estimate the PSDWelch Method:
  - Divide time-series data into small segments. They could be overlapping segments as well.
  - Fourier Transform and calculate $|d(f)|^2$, which is basically the Fourier transform, for each segment.
  - Calculate the average over all segments.

So, if we have more data to average over, we will have a ore accurate estimation of a PSD, which is shown in the plot



4.3 Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{12}$$

$$\xrightarrow{\text{goes to}} P(\theta|d) = \frac{P(d|\theta)P(\theta)}{p(d)} \tag{13}$$

where, d is data.

We will promote all of these to also a probability distribution, here we were talking about individual probabilities, but now we will think of these P_s as probability distribution and we will also give these probability distribution their own separate symbol.

Just changing the notations a bit more we get,

$$p(\theta|d) = \frac{\mathcal{L}(d|\theta)\pi(\theta)}{Z_s} \tag{14}$$

where, $p(\theta|d)$ is posterior distribution of parameters, $\mathcal{L}(d|\theta)$ is likelihood, $\pi(\theta)$ is prior and $Z_s = \int d\theta\ \mathcal{L}(d|\theta)\pi(\theta)$ is known as evidence.

The problem with parameter estimation is to actually estimate the left hand side of Eq. 14, which is called posterior distribution, the probability that the set of parameters θ describes he signal.

4.3.1 The Likelihood

- At each frequency, the residual after subtracting the best fit signal from the data, should look like noise and hence follow a Gaussian distribution.
- Hence the likelihood function/ distribution can be written as a Gaussian distribution in the residuals. For each frequency:

$$\mathcal{L}(d_i|\theta) = \frac{2\delta f}{\pi S_n(f_i)} \exp\left(-2\delta f \frac{|d(f_i) - h(f_i;\theta)|^2}{S_n(f_i)}\right) \tag{15}$$

  And hence Total Likelihood can be written as:

$$\mathcal{L}(d|\theta) = \prod_{i=1}^{N} \mathcal{L}(d_i|\theta) \tag{16}$$

4.3.2 The Prior Distribution

- A distribution that encodes the prior knowledge/ belief in what parameters of the system ought to be.
- Example:
  - No specific preferred direction in the sky $\implies$ Uniform in sky area
  - Not much information about the mass distribution $\implies$ Uniform in masses.

– Physical limit on the maximum spin on neutron stars $\implies$ ¡ 0.89.

– and so on.

4.3.3 The Evidence

- The Evidence is used to compare different hypothesis for the data.
- Example: Signal vs Noise

  A simple example of where the evidence is used is in comparing signal and noise. So, say we want to find out how significant is the fact that the data contains the signal is compared to the hypothesis that the data contains no signal.

  – The evidence for noise is just the likelihood with no signal.

$$Z_n = \frac{2\delta f}{\pi S_n(f_i)} \exp\left(-2\delta f \frac{|d(f_i)|^2}{S_n(f_i)}\right) \tag{17}$$

  – Bayes Factor (of signal vs noise) is the ratio of evidences:

$$\mathrm{BF}^{\mathrm{S}}_{\mathrm{N}} = \frac{Z_S}{Z_N} \tag{18}$$

  If the Bayes Factor is say above certain cut-off say 3000, we say that there is evidence for the signal being present in the data.

4.4 The Problem

- We have all tools to calculate the posterior. But we have to calculate it over a very high dimensional space.
- Gridding the space and calculating will be inefficient (cost scales exponentially with dimensions.)
- Solution: Use stochastic samplers

4.5 Stochastic Samplers: Markov Chain Monte Carlo

- Populate the prior space with random numbers.
- Take steps based on whether the move will increase probability.
- Note: there exists other sampling methods too, eg nested sampling, used frequently in LVK data analysis.

5 Projects

All these questions I have attempted from a workshop organised by Gravitational Wave Open Science Center

5.1 Data Access

How many months did O2 lasted?

```
from gwosc.datasets import run_segment
O2 = run_segment('O2_4KHZ_R1')
print('O2 start and stop gps: ', O2)
time_seconds = O2[1] - O2[0]
month_seconds = 30 * 24 * 3600
print('months in O2:', time_seconds / month_seconds)
```

Output -
O2 start and stop gps: (1164556817, 1187733618) months in O2: 8.941667052469136

How many GWTC-3-confident events were detected during O3b?

```
from gwosc import datasets
from gwosc.datasets import run_segment
GWTC3_events = datasets.find_datasets(type='events', catalog='GWTC-3-
    ↪ confident', segment=run_segment('O3a_16KHZ_R1'))
print(len(GWTC3_events))
```

Output -
0

What file URL contains data for V1 4096 seconds around GW170817?

```
from gwosc.locate import get_event_urls
url = get_event_urls('GW170817', duration=4096, detector='V1')
print(url)
```

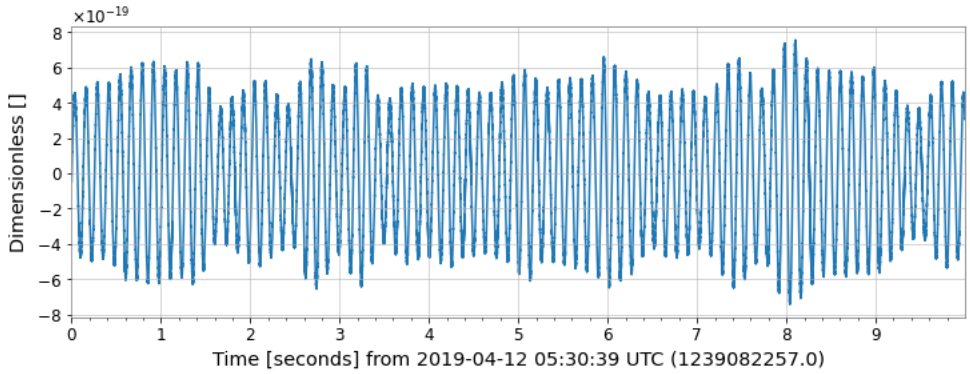
Output -
['https://www.gw-openscience.org/eventapi/json/GWTC-1-confident/GW170817/v3/V-V1_GWOSC_4KHZ_R1-1187006835-4096.hdf5']

5.2 Working with GWpy

Plot the data for the LIGO-Hanford detector around GW190412. Looking at your new LIGO-Handford plot, can your eye identify a signal peak?

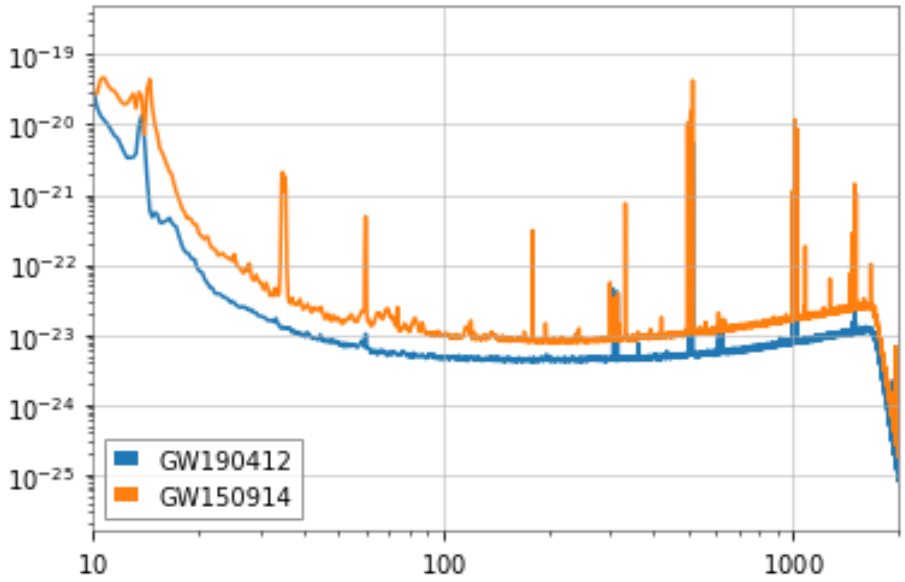
```
from gwpy.timeseries import TimeSeries
from gwosc.datasets import event_gps
```

```
3
4 gps = event_gps('GW190412')
5 segment = (int(gps)-5, int(gps)+5)
6 hdata = TimeSeries.fetch_open_data('H1', *segment, verbose=True)
7 hdata.plot()
```



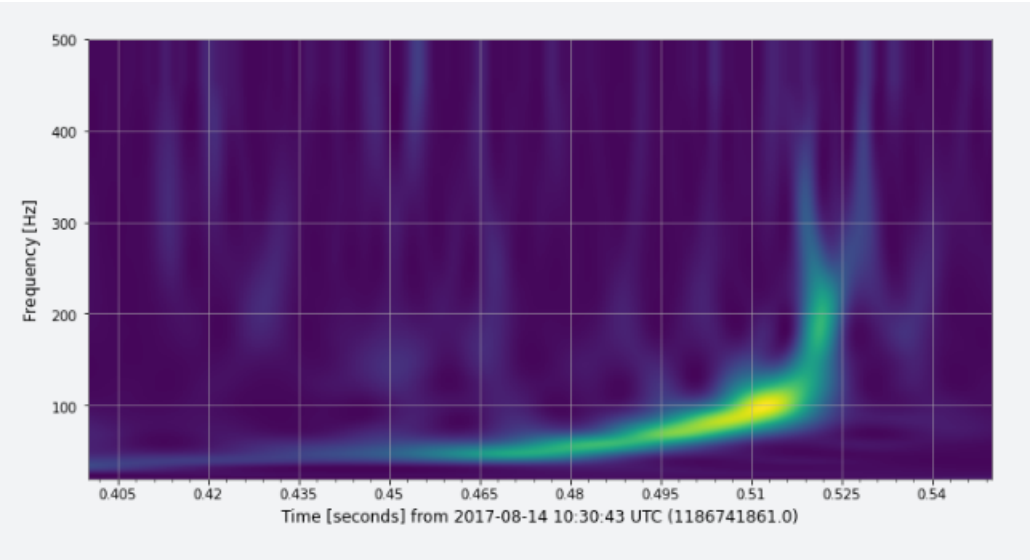
Make an ASD around the time of an O3 event, GW190412 for L1 detector. Compare this with the ASDs around GW150914 for L1 detector. Which data have lower noise - and so are more sensitive - around 100 Hz?

```
1  from gwpy.timeseries import TimeSeries
2  from gwosc.datasets import event_gps
3  import pylab as plt
4
5  gps = event_gps('GW190412')
6  ldata = TimeSeries.fetch_open_data('L1', int(gps)-512, int(gps)+512,
       ↪ cache=True)
7  lasd = ldata.asd(fftlength=4, method="median")
8
9  gps2 = event_gps('GW150914')
10 ldata2 = TimeSeries.fetch_open_data('L1', int(gps2)-512, int(gps2)
       ↪ +512, cache=True)
11 lasd2 = ldata2.asd(fftlength=4, method="median")
12
13 plt.loglog(lasd, label = 'GW190412')
14 plt.loglog(lasd2,label = 'GW150914')
15 plt.legend()
16 plt.xlim(10,2000)
17 plt.plot()
18 plt.show()
```



Download the data for the LIGO-Livingston detector (L1) around the GW170814 detection, and make a Q-scan of the data. The signal is visible for how about how much time?

```
1 from gwosc.datasets import event_gps
2 from gwpy.timeseries import TimeSeries, TimeSeriesDict
3 gps = event_gps('GW170814')
4 print("GW170814 GPS:", gps)
5 data = TimeSeries.fetch_open_data('L1', int(gps)-512, int(gps)+512,
    ↪ cache=True, verbose=True)
6 QT = data.q_transform(frange=(20, 500), qrange=(4, 12), outseg=(gps
    ↪ -.1,gps+.05))
7 QT.plot()
```



5.3 Working with PyCBC

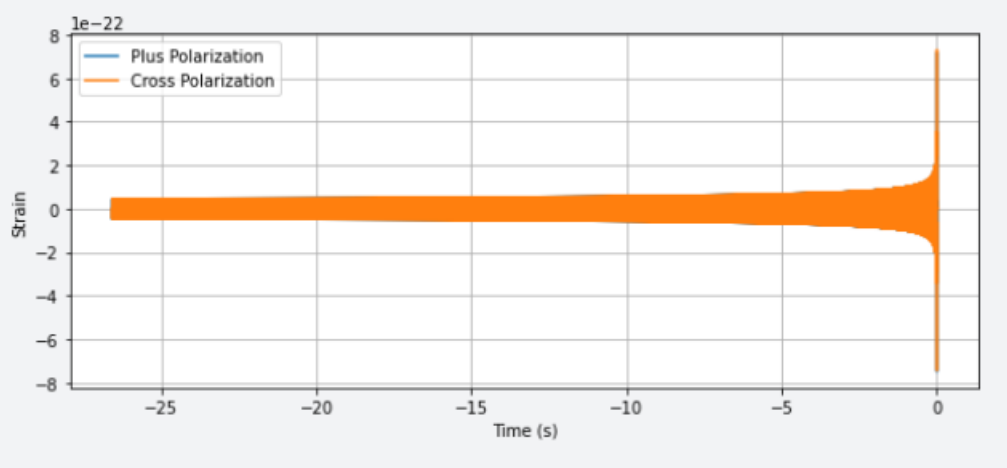
Make a waveform with the mass parameters of GW170817, and a low frequency cut-off (f_lower) of 40 Hz. How long is the waveform?

```
from pycbc.waveform import get_td_waveform
import pylab

m1 = 1.46 # solar mass units
m2 = 1.27 # solar mass units
d = 40 # Megaparsec units
f_low=40 # Hertz

hp, hc = get_td_waveform(approximant="SEOBNRv4_opt",
                mass1 = m1,
                mass2 = m2,
                distance = d,
                delta_t = 1.0/16384,
                f_lower = f_low)

pylab.figure(figsize=pylab.figaspect(0.4))
pylab.plot(hp.sample_times, hp, label='Plus Polarization')
pylab.plot(hp.sample_times, hc, label='Cross Polarization')
pylab.xlabel('Time (s)')
pylab.ylabel('Strain')
pylab.legend()
pylab.grid()
pylab.show()
```



Which template provides the best match (highest SNR) to the binary black hole in file "File PyCBC_T2_0.gwf"?
Information that may be useful:

- **Signals are all placed between 100 and 120 seconds into the frame file.**

- **You may assume mass1 = mass2 (equal mass) and that each component mass is one of 15, 30, or 45.**
- **Each file starts at GPS time 0, and ends at GPS time 128.**
- **The channel name in each file is "H1:TEST-STRAIN".**

```
# Download the challenge set files
from pycbc.frame import read_frame
from pycbc.filter import resample_to_delta_t, highpass
from pycbc.psd import interpolate, inverse_spectrum_truncation
from pycbc.filter import matched_filter
import numpy
import urllib

def get_file(fname):
    url = "https://github.com/gw-odw/odw-2022/raw/main/Tutorials/Day_2/
        ↪ Data/{}"
    url = url.format(fname)
    urllib.request.urlretrieve(url, fname)
    print('Getting : {}'.format(url))

files = ['PyCBC_T2_0.gwf', 'PyCBC_T2_1.gwf', 'PyCBC_T2_2.gwf']

for fname in files:
    get_file(fname)


# Reading the data from these files:
file_name = "PyCBC_T2_0.gwf"

# Strain is typically IFO:LOSC-STRAIN, where IFO can be H1/L1/V1.
channel_name = "H1:TEST-STRAIN"

start = 0
end = start + 128

ts = read_frame(file_name, channel_name, start, end)

# Calculating SNR for each case

ts = highpass(ts,15.0)
ts = resample_to_delta_t(ts, 1/2048)
ts2 = ts.crop(2, 2)

# Using 4 second samples of our time series in Welch method.

ts2_psd = ts2.psd(4)

```

```
ts2_psd = interpolate(ts2_psd, ts2.delta_f)

ts2_psd = inverse_spectrum_truncation(ts2_psd, int(4 * conditioned.
    ↪ sample_rate), low_frequency_cutoff=15)


snrp_list = []
i = 0
required = ''

for m in [15,30,45]:
    hp, hc = get_td_waveform(approximant='SEOBNRv4_opt',
                             mass1=m,
                             mass2=m,
                             delta_t=ts2.delta_t,
                             f_lower=20)


    hp.resize(len(ts2))

    template = hp.cyclic_time_shift(hp.start_time)

    snr = matched_filter(template, ts2, psd=ts2_psd,
        ↪ low_frequency_cutoff=20)

    snr = snr.crop(4 + 4, 4)

    peak = abs(snr).numpy().argmax()
    snrp = snr[peak]
    time = snr.sample_times[peak]
    snrp_list.append(abs(snrp))
    if i > 0 and snrp_list[i] > snrp_list[i-1]:
        required = m
    i+=1
print("We found the best match for {} solar masses".format(required))
```

Output -

We found the best match for 45 solar masses

This is a simple demonstration to loading and viewing data released in association with the publication titled GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs available through DCC and arXiv. This should lead to discussion and interpretation.
The data used in these tutorials will be downloaded from the public DCC page LIGO-P1800370.

Calculate the mean of the chi_eff posterior?

```
import numpy as np
import matplotlib.pyplot as plt
import h5py
import pandas as pd
import corner
import bilby
import astropy.units as u
from astropy.cosmology import Planck15, z_at_value

label = 'GW150914'

! wget https://dcc.ligo.org/LIGO-P1800370/public/{label}_GWTC-1.hdf5

posterior_file = './'+label+'_GWTC-1.hdf5'
posterior = h5py.File(posterior_file, 'r')

print('This datasets this file contains are: ',posterior.keys())

print(posterior['Overall_posterior'].dtype.names)

samples = pd.DataFrame.from_records(np.array(posterior['Overall_posterior']))

# Plotting the corner plot

corner.corner(samples,labels=['costhetajn',
                              'distance [Mpc]',
                              'ra',
                              'dec',
                              'mass1 [Msun]',
                              'mass2 [Msun]',
                              'spin1',
                              'spin2',
                              'costilt1',
                              'costilt2']);


# We need to convert the detector masses into source frame mass

z = np.array([z_at_value(Planck15.luminosity_distance, dist * u.Mpc) for dist in samples['luminosity_distance_Mpc']])

samples['m1_source_frame_Msun']=samples['m1_detector_frame_Msun']/(1.0+z)
```

```
42    samples['m2_source_frame_Msun']=samples['m2_detector_frame_Msun
          ↪ ']/(1.0+z)
43    samples['redshift']=z
44
45    corner.corner(samples[['m1_source_frame_Msun','m2_source_frame_Msun
          ↪ ','redshift']],labels=['m1 (source)','m2 (source)','z']);
46
47
48    # Calculate the detector frame chirp mass
49    mchirp = ((samples['m1_detector_frame_Msun'] * samples['
          ↪ m2_detector_frame_Msun'])**(3./5))/(samples['
          ↪ m1_detector_frame_Msun'] + samples['m2_detector_frame_Msun
          ↪ '])**(1./5)
50
51    # Initializing a SampleSummary object to describe the chirp mass
          ↪ posterior samples
52    chirp_mass_samples_summary = bilby.core.utils.SamplesSummary(
          ↪ samples=mchirp, average='median')
53    print('The median chirp mass = {} Msun'.format(
          ↪ chirp_mass_samples_summary.median))
54    print('The 90% confidence interval for the chirp mass is {} - {}
          ↪ Msun'.format(chirp_mass_samples_summary.
          ↪ lower_absolute_credible_interval,chirp_mass_samples_summary.
          ↪ upper_absolute_credible_interval))
55
56    # Calculating effective spin
57
58    eff_spin = (samples['m1_source_frame_Msun']*samples['spin1']*
          ↪ samples['costilt1'] + samples['m2_source_frame_Msun']*
          ↪ samples['spin2']*samples['costilt2']) / (samples['
          ↪ m1_source_frame_Msun'] + samples['m2_source_frame_Msun'])
59    print('mean', eff_spin.mean())
60    summary = bilby.core.utils.SamplesSummary(samples=eff_spin, average
          ↪ ='median')
61    print('The 90% confidence interval for chi effective is {} - {} '.
          ↪ format(summary.lower_absolute_credible_interval, summary.
          ↪ upper_absolute_credible_interval))
```