Template Bank Thinning based on Zeroth Order Threshold over Chirp Time

(DCC Reference: LIGO-T1600311)

Aravind Pazhayath Ravi, Ofek Birnholtz, Alex H. Nitz

September 13, 2016

Abstract

We present here a modification to the search pipeline PyCBC [1] (pycbc_inspiral) to accommodate for the usage of injection specific template bank which allows for thinning/trimming the template bank used for matched filtering, thereby throwing away the extra computational hours needed to look over templates in the bank completely unrelated to the injection in question. This relation is quantified by implementing a threshold on chirp time(τ_0) of the compact binary system such that only if the template falls within the window defined by the threshold, would it be filtered to get to the trigger list. The inspiral code forms only the first part of the entire search protocol and threshold is decided or rather reverse engineered by looking at the triggers of the existing runs in O1. The modified template bank consists of sets of concatenated mini-template banks whose sizes are dependent on the threshold. This approach is only a zeroth order approximation, albeit good enough as a first step in reducing the size of the bank searched over. This approach can be extended in the future to include higher order corrections.

1 Introduction

We are trying to implement a threshold on the error in the chirp time comparing the template and injection parameters. The expression for chirp time can be obtained through chirp mass which is a function of the component masses given by

$$\mathcal{M} = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1 + m_2)^{\frac{1}{5}}} \tag{1}$$

Chirp Mass $\mathcal{M}$ for the lowest order post newtonian approximation, relates the frequency and frequency derivative to the emitted gravitational waves by [2]

$$\mathcal{M} = \frac{c^3}{G}\left(\Big(\frac{5}{96}\Big)^3 \pi^{-8} (\omega(t))^{-11} (\dot{\omega(t)})^3\right)^{\frac{1}{5}} \tag{2}$$

Here, $\omega(t)$ is the frequency of the emitted gravitational radiation. We need to integrate this equation to obtain chirp time (τ_0) as a function of $\mathcal{M}$.

$$\int_{\omega_{ref}}^{\infty} \omega^{\frac{-11}{3}} d\omega = \int_0^{\tau_0} dt \left(\frac{G\mathcal{M}}{c^3}\right)^{\frac{5}{3}} \left(\frac{96}{5}\right) \pi^{\frac{8}{3}} \tag{3}$$

Thus evaluating the integral and plugging in the limit,

$$(\omega_{ref})^{\frac{-8}{3}} = \frac{256}{5}\left(\frac{G\mathcal{M}}{c^3}\right)^{\frac{5}{3}} \pi^{\frac{8}{3}} \tau_0 = \frac{(8\pi)^{\frac{8}{3}}}{5}\left(\frac{G\mathcal{M}}{c^3}\right)^{\frac{5}{3}} \tau_0 \tag{4}$$

Thus the chirp time (τ_0) would be

$$\tau_0 = \frac{5}{(8\pi)^{\frac{8}{3}}}\left(\frac{c^3}{G\mathcal{M}}\right)^{\frac{5}{3}} (\omega_{ref})^{\frac{-8}{3}} \tag{5}$$

Plugging equation (1) into equation(5), we get

$$\tau_0 = \frac{5}{(8\pi)^{\frac{8}{3}}}\left(\frac{c}{G^{\frac{1}{3}}}\right)^{5} \frac{(m_1+m_2)^{\frac{1}{3}}}{m_1 m_2} (\omega_{ref})^{\frac{-8}{3}} \tag{6}$$

2 Modifications

The following codes were modified to include the new checks. Modification for strain.py and inject.py was with an aim to return the injection parameters used in the PyCBC[1] inspiral pipeline.

1. inject.py
   Path **/pycbc_dev/src/pycbc/pycbc/inject.py**
   The purpose of this code is to define the function that applies the injection. Modifications to the code return the parameters of these injections as in the original code, the parameters of these applied injections are not stored. Also, not all the injections from the injection file are passed on. Some checks are performed by the original code and only a subset of the injections are made. Thus inject.py was modified such that the function that applies the injection now returns an array with its parameters while making the injections.

2. strain.py
   Path **/pycbc_dev/src/pycbc/pycbc/strain.py** Purpose of this code is to return the strain data necessary for the analysis. However our modifications return the injection parameters along with it under the right conditions. From client, originally there was no option for returning the injection parameters. Thus we first included a boolean flag named return_injections for this and the control was vested within pycbc_inspiral. If the flag were set to true, we return the strain data along with the injection parameters else (as is the default state), we just return the strain.

3. bank.py
   Path **/pycbc_dev/src/pycbc/pycbc/waveform/bank.py**
   This module provides classes that describe banks of waveforms. The aim of modifying was to incorporate a method to the class named TemplateBank which provides some basic helper functions and information about elements of an xml template bank. The method takes in two inputs namely, the injection parameters as returned from strain.py and a new option called the threshold which we define by looking at the plots described later in this report. The method then calculates the template chirp time and compares it with the injection chirp time for each injection. If this difference is within the threshold window we choose, then we include that particular index of the template bank into the new smaller bank. It is then ensured that only unique indices remain.This subset bank is all we need for the injection in question. So many such mini banks are created, one for each injection. Once all the mini banks are created they are concatenated and depending on the injection parameter, only some of the mini banks would be used for the analysis.

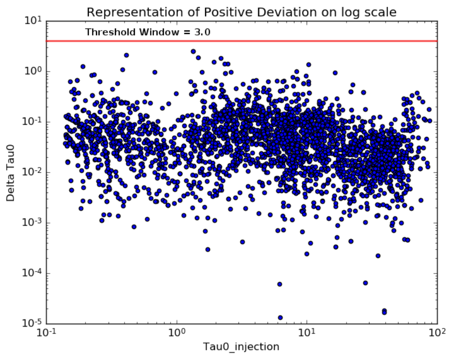
4. pycbc_inspiral
   Path **/pycbc_dev/src/pycbc/bin/pycbc_inspiral**
   This is the main code which incorporates all the changes made thus far in the previous three codes. The code has many input options and using a template bank, injection file and frame file it churns out the trigger list of a single detector. This is only one part of the multi step process of the PyCBC analysis[1]. As we said earlier, the control of the boolean flag controlling the return of injection parameters was implemented in the code so that we can have both the strain and the injection parameters returned. This gives one out of the two inputs needed to call upon the method we defined for the class TemplateBank in bank.py. The other input is the already talked about threshold window which we empirically decided from the plot of the error of chirp time v/s the injection chirp time. An input option was parsed into the code and if the user provides a threshold value then, we call upon the method to thin our template bank and return the smaller bank after the cut. Thus depending on the set of injections provided, we have different number of templates filtered. We record the number of accepted and rejected templates out of the total number of templates in the template bank.

3 Analysis and Plots

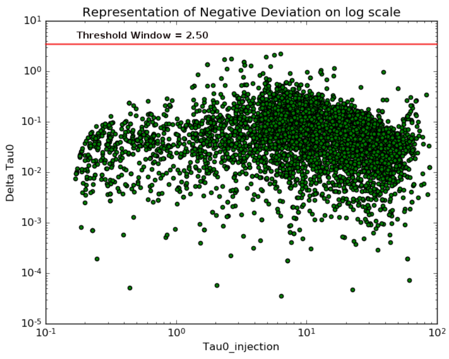
A part of an existing analysis (Analysis 9, carried out for a few weeks of the O1 data from Sep 12, 2015 to Oct 20, 2015) had an snr threshold of 5.5. We have looked at the results of this analysis to empirically get a threshold value on the chirp time, that we implement at an earlier stage of the analysis. We first look at the positive deviation and then the negative deviation to get a conservative threshold value below which all the errors in chirp time lie. The motivation here is that if we consider any injection in the template parameter space, a fixed window as decided should contain all the templates that can give a match with the injection. Thus we should not be missing triggers. Both axes in the plots are in seconds. Deviation (Δ_{τ_0}) is defined as

$$\Delta_{\tau_0} = trigger_{\tau_0} - injection_{\tau_0} \tag{7}$$

which is basically a difference between trigger and injection chirp times. Positive deviation in chirp time (τ_0) occurs when the difference between trigger and injection chirp times becomes positive. Similarly if this difference is negative, we have a negative deviation.



(a) Positive deviation in Chirp Time



(b) Negative deviation in Chirp Time

Figure(2) is the plot of the deviation in chirp mass compared to the injection chirp mass. Chirp mass is directly related to the component masses and we calculate chirp time from chirp mass using the pnutils module of pycbc which exactly matches with the equation (6)

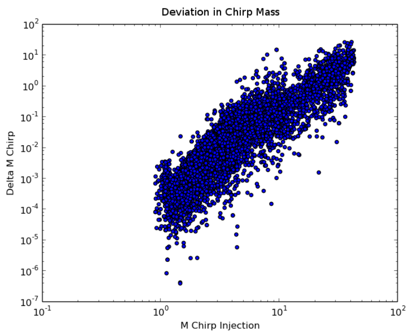


Figure 2: Deviation in Chirp Mass

Once the threshold value is available, we run the modified pycbc_inspiral over three kinds of injection files to check for how many templates were accepted and how many were rejected. Consider a run over the entire bank of the data segment we are looking at. It had 249077 templates before the implementation of the cut. After the cut, the number of rejected and accepted templates were different for different kinds of injection.

Type	Accepted	Rejected
BBH	104302	144775
BNS	114459	134618
NSBH	119492	129585

In order to check that we don't miss triggers, a preliminary test on highest values of snr was done. The top six values of snr in the trigger lists both before and after the cut match. Here we use the template duration to confirm our matches for the six entries.

SNR Original	SNR After Cut	Template Duration	Template Duration
10.930367	10.930367	1.59989	1.59989
11.0635195	11.0635195	1.60776	1.60776
11.12573	11.12573	1.58739	1.58739
11.391197	11.391197	1.61392	1.61392
11.468006	11.468006	1.61526	1.61526
11.528804	11.528804	1.58343	1.58343

Similarly for the lower SNRs we see that

SNR Original	SNR After Cut	Template Duration	Template Duration
5.500166	5.500166	0.618471	0.618471
5.5006704	5.5006704	0.624299	0.624299
5.501088	5.501088	1.94443	1.94443
5.501197	5.501197	0.909629	0.909629
5.501283	5.501283	0.508037	0.508037
5.501386	5.501386	1.74373	1.74373

From this analysis if we have a look at the individual injections, we find that

1. Injection using maximum number of templates (58354)
   (a) Mass1: 10.7736
   (b) Mass2: 2.813951
2. Injection using minimum number of templates (14157)
   (a) Mass1: 55.80185
   (b) Mass2: 37.43661

As we can see lower mass systems use larger number of templates while higher mass systems use lesser number of templates partly because of the Inspiral phase duration and partly because the former are much more denser compared to the latter thus not violating consistency.

To check for more consistency, consider a template bank with 2554 templates over about 1 hour around the detection time of GW150914[3].

Applying different values of threshold, we plot the snr values retained and those missed.

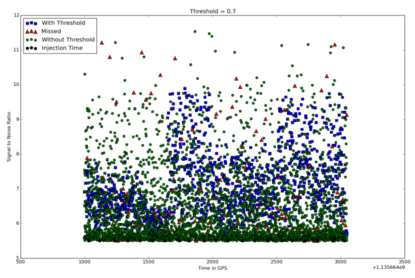


Figure 3: Threshold = 0.7

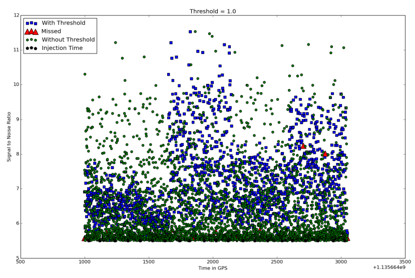


Figure 4: Threshold = 1.0

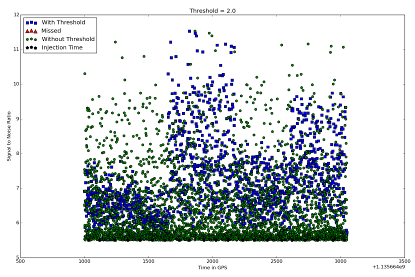


Figure 5: Threshold = 2.0

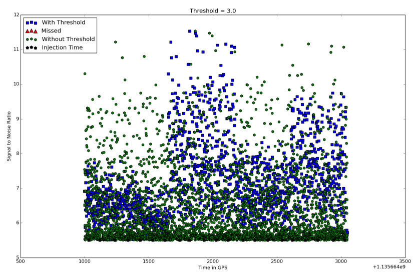


Figure 6: Threshold = 3.0

Consider a similar analysis on a much larger template bank with millions of templates . Our analysis should ensure that we don't miss template matchings with high snr due to our modifications. By comparing the gps end times of the injections with the trigger gps times, we have plotted a set of values recovered with different thresholds. As the threshold increases, the significance of the missed triggers become less and less important. The X axis in the figures 7,8 and 9 pertain to the gps end times for all the injections. In our case we look at 21 injections over the entire dataset. The threshold values were 0.5, 2 and 3.

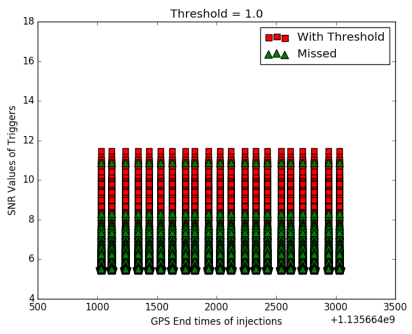


Figure 7: Threshold = 1.0

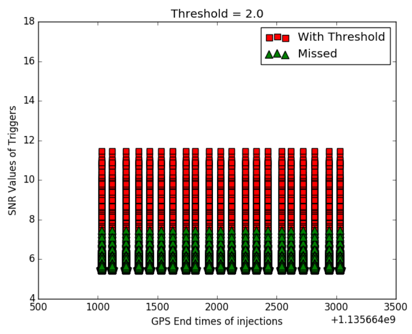


Figure 8: Threshold = 2.0

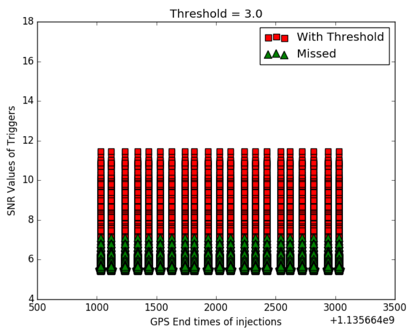


Figure 9: Threshold = 3.0

4 Conclusion

We see how with increasing the threshold values, the number of missed values decrease drastically. Thus we have for higher thresholds, pertaining to each missed case (if any) a higher SNR in the recovered trigger list, thereby not missing out on any injection. We had made an empirical guess of a window of 3.0 around the chirp time, the first step towards optimising the template matching algorithm. The plot over large and small data for this value of threshold reassures us that it was indeed a sober decision to make as the snr of the missed triggers are low and hence we don't miss out on any high snr triggers.

References

[1] B.P.Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration.) GW150914: First results from the search for binary black hole coalescence with Advanced LIGO.

Phys. Rev. D 93, 122003 Published 7 June 2016.

[2] B.P.Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration.) The basic physics of the binary black hole merger GW150914 arXiv:1608.01940 - August 5, 2016

[3] B.P.Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration.) Observation of Gravitational Waves from a Binary Black Hole Merger Phys. Rev. Lett. 116, 061102 Published 11 February 2016.