Glitch Estimation and Removal Using Adaptive Spline Fitting and Wavelet Shrinkage on the Gravitational Wave Data

Mohammad Abu Thaher Chowdhury
The University of Texas Rio Grande Valley

GLITCH ESTIMATION AND REMOVAL USING ADAPTIVE SPLINE FITTING AND WAVELET SHRINKAGE ON THE GRAVITATIONAL WAVE DATA

A Thesis

by

MOHAMMAD ABU THAHER CHOWDHURY

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Physics

The University of Texas Rio Grande Valley

December 2022

GLITCH ESTIMATION AND REMOVAL USING ADAPTIVE SPLINE FITTING AND WAVELET SHRINKAGE ON THE GRAVITATIONAL WAVE DATA

A Thesis
by
MOHAMMAD ABU THAHER CHOWDHURY

COMMITTEE MEMBERS

Dr. SOUMYA D. MOHANTY
Chair of Committee

Dr. MALIK RAKHMANOV
Committee Member

Dr. SOMA MUKHERJEE
Committee Member

December 2022

ABSTRACT

Chowdhury, Mohammad Abu Thaher, GLITCH_ESTIMATION_AND_REMOVAL_USING_ADAPTIVE SPLINE_FITTING_AND_WAVELET_SHRINKAGE_ON_THE_GRAVITATIONAL_WAVE_DATA. Master of Science (MS), December, 2022, 121 pp., 3 tables, 36 figures, references, 57 titles.

The false alarm rate and reduced sensitivity of searches for astrophysical signals are caused by transient signals of earthly origin, or "glitches," in gravitational wave strain data from ground-based detectors. The greater number of observable astrophysical signals will increase the likelihood of glitch overlaps and exacerbate their negative impact for future detectors with higher sensitivities. The wide morphological diversity and unpredictable waveforms of glitches, and with the vast majority of cases lacking supplemental data present the main obstacles to their mitigation. Thus, nonparametric glitch mitigation techniques are required, which should operate for a wide range of glitches and, in the case of overlaps, have little impact on astrophysical signals. The arrangement of free knots is improved to estimate both smooth and non-smooth curves, and wavelet-based shrinkage is added for specific types of glitches in our method for glitch estimation and removal utilizing adaptive spline curve fitting. The effectiveness of the technique is evaluated for seven different kinds of LIGO detector glitch types. In the specific instance of a loud glitch in data from LIGO, Livingston that coincides with the event GW170817, the glitch is evaluated and eliminated without adversely altering the gravitational wave signal. For injected signals overlapped with other kinds of glitches, similar results are observed.

DEDICATION

Dedicated to my parents, Mohammad Abdul Latif Chowdhury and Syeda Shamsun Nahar, for giving me support and believing me during my ups and downs.

ACKNOWLEDGMENTS

I want to express my gratitude, first, to my advisor Dr. Soumya D. Mohanty, for guiding me through this work and helping me to understand the concepts that I needed most; To Dr. Malik Rakhmanov, for sharing his knowledge of physics and others, time, and advice during my study in UTRGV; To Dr. Soma Mukherjee for explaining physics and guidance. I also want to thank UTRGV's Physics and Astronomy department for giving financial assistance to my graduate education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Gravitational Wave

According to General relativity, mass bends space-time. The heavier the mass is, the more bending it will create in space-time. And if this mass is starting to move, then it will create a wave in the space-time. This wave is known as a gravitational wave (GW). The amplitude of this wave, even with a high mass, is so small that detecting it is a hard task. Moreover, GW does not interact with other things (at least no interaction is known till now), except mass, which made it even more difficult to detect. But because of less interaction, it is the most undisturbed wave which can give us much more information about the universe, other than any other form of electromagnetic radiation, and the phenomena happening/occurred in the universe Hartle (2003). That is why it is very important to detect and extract information from a gravitational wave.

The theory of general relativity illustrate gravitational force by metric tensor, $g_{\mu\nu}$. Metric tensor $g_{\mu\nu}$ can be written as,

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \tag{1.1}$$

where, $\eta_{\mu\nu}$ is a constant metric of Minkowski space and $h_{\mu\nu}$ is the metric for small perturbation in the space time caused by the mass. When $h_{\mu\nu}$ is zero i.e. there is no mass, $g_{\mu\nu}$ is equal to $h_{\mu\nu}$. Thus, the line element for the space-time of this case will be,

$$ds_1^2 = -c^2dt^2 + dx^2 + dy^2 + dz^2 \tag{1.2}$$

If there is a mass object, then it will have some perturbation. Let us consider the value of the perturbation is,

$$h_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_{xx} & h_{xy} & 0 \\ 0 & h_{xy} & -h_{xx} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{1.3}$$

Here, the value of h_{xx} and h_{xy} is small. Also, h_{xx} is known as plus polarization (h_+) of GW and (h_{xy}) is known as cross polarization $(h_\times)$ of GW. In this case, the line element will be

$$\begin{aligned} ds^2 &= -c^2dt^2 + dx^2 + dy^2 + dz^2 + 0^2 + h_{xx}dx^2 - h_{xx}dy^2 + 0^2 + h_{xy}dxdy + h_{xy}dydx \\ &= -c^2dt^2 + (1+h_{xx})\,dx^2 + (1-h_{xx})\,dy^2 + dz^2 + 2h_{xy}dxdy \end{aligned} \tag{1.4}$$

1.2 GW Detection parameter

When gravitational wave passes through a space-time, it changes the metric between two test masses due to the perturbation $(h_{\mu\nu})$. For example, let us consider a gravitational wave is traveling toward the z-direction. In this system, there are two test masses, one is in the origin and another one is in x-axis with L distance away from the first test mass. In the space-time of gravitational wave, the distance between these two masses will be,

$$\begin{aligned} L(t) &= \int_0^L (1+h_{xx})^{\frac{1}{2}}\,dx \\ &\approx \int_0^L \left(1+\frac{1}{2}h_{xx}\right)dx \\ &\approx \left(1+\frac{1}{2}h_{xx}\right)\int_0^L dx \\ &\approx L\left(1+\frac{1}{2}h_{xx}\right) \end{aligned} \tag{1.5}$$

Therefore, the change in distance will be,

$$\begin{aligned}\delta L(t) &= L(t) - L(0) \\ &\approx L\left(1 + \frac{1}{2}h_{xx}\right) - L \\ &\approx \frac{1}{2}h_{xx}L \\ \frac{\delta L(t)}{L} &\approx \frac{1}{2}h_{xx} \end{aligned} \tag{1.6}$$

Now, in this example the calculated value of perturbation is in x axis. The perturbation can happen in any direction along any vectors. Therefore, the general case, for a vector 'a' can be written as,

$$\frac{\delta L(t)}{L} \approx \frac{1}{2}h_{\mu\nu}a^{\mu}a^{\nu} \tag{1.7}$$

Moreover, the left hand side of equation 1.6/ 1.7 has a term with the ratio of the change of length (δL) and the length (L) itself, which is known as strain. Gravitational wave detection device measure this strain quantity to detect the gravitational wave.

1.3 Laser Interferometer Gravitational-wave Observatory (LIGO)

There are two types of GW detectors: beam detectors and resonant mass detectors Sathyaprakash & Schutz (2009). Laser Interferometer Gravitational-wave Observatory (LIGO) is one of the beam detectors around the globe. The current LIGO is mentioned as advanced LIGO (aLIGO) J Aasi (2015), because of the change in instruments to increase sensitivity and reduce noise. aLIGO is an enhanced version of the Michelson interferometer with a power recycling mirror and two arms of 4km length with Febry-Perot cavities in it. In every arm, there are two mirrors: the input mirror and the end mirror, which work as a test mass. These two mirrors are suspended from above with the help of glass fibers. If there is an event that changes the distance between these two mirrors symmetrically in both arms then it will not produce any signal as there will be no change in the phase of the incident lights.

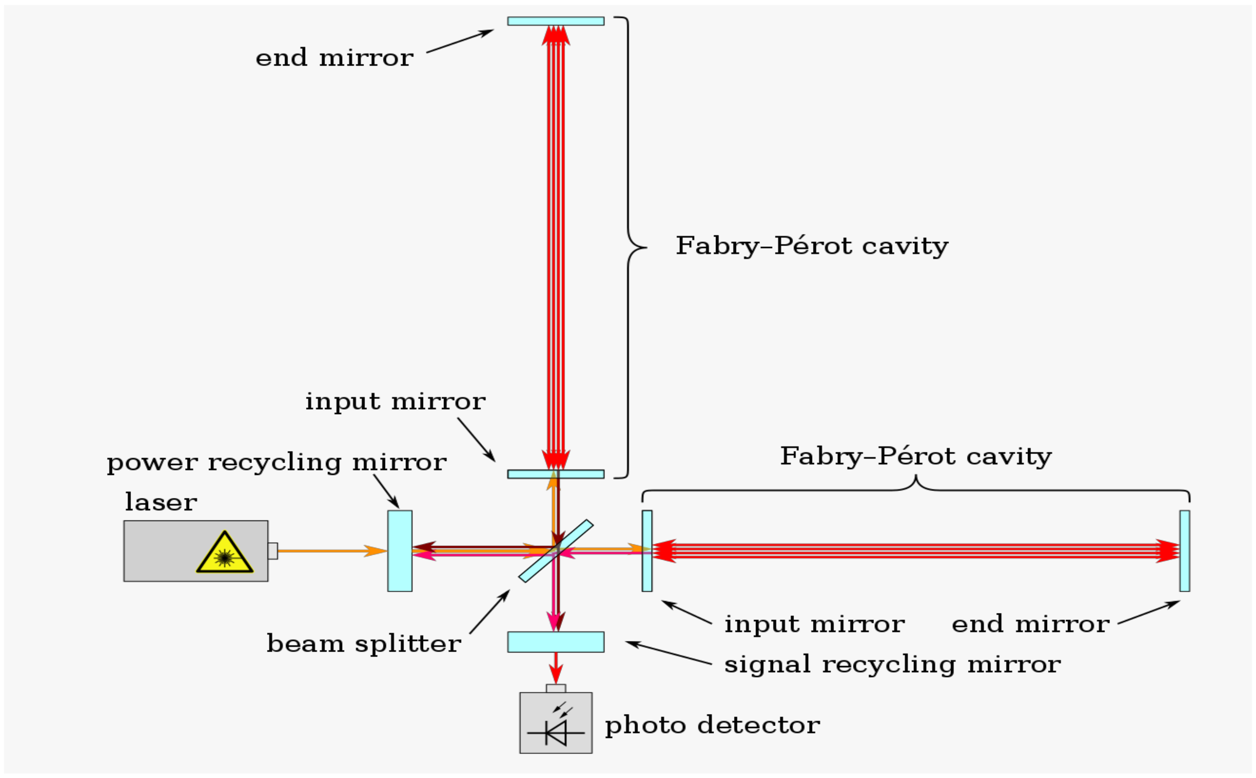


Figure 1.1: Schematic diagram of LIGO (Image credit:https://www.ligo.caltech.edu)

But if the gravitational wave passes through the arms asymmetrically, it will stretch one arm while it will compress the other arm. Thus, it will cause a change in the length of both arms, which will cause the phase difference of light in the different arms. This phenomenon is known as 'Differential Arm'. Thus, when these lights become incident with each other, it will produce interference in the photodetector as long as GW pass through these two arms. As explained in the previous section, this 'Differential Arm' will produce a value for strain.

1.4 GW sources

Any moving object will create gravitational waves (GW). For example, if someone moves his/her hand or takes a step during a walk, it would create a gravitational wave. But it is so weak that it is not detectable. For detecting a gravitational wave signal, the sources should produce a big enough signal so that the detector can detect it. Depending on the condition, we can understand that the only detectable gravitational wave signal comes from big objects like stars, planets, black holes. There are four types of gravitational wave signal sources Riles (2013): Binary inspirals, Continuous gravitational waves, Bursts, and Stochastic.

1.4.1 Binary Inspirals signal

Binary inspiral gravitational waves are produced by pairs of dense and enormous objects such as neutron stars, and black holes. There are two stages for this gravitational wave source. At first, when two compact objects are relatively far from each other, these objects revolve around each other. During this stage, it continues to give a monochromatic gravitational wave signal i.e. it has no noticeable change in frequency and amplitude. Those dense objects start to lose energy due to the rotation and their orbit gets smaller and smaller, which eventually leads to the second stage. During this stage, these compact objects start to rotate faster as these objects are near to each other and coalesce. As the inspirals are moving faster and finally merged, it changes the amplitude and frequency rapidly during this time. The signal, binary inspirals, emanates during the second stage and is known as a chirp signal. Binary inspirals have different waveforms depending on compact objects and their orbits. Binary neutron star - binary neutron star (BNS) pair, binary black hole - binary black hole (BBH) pair, and binary neutron star - binary black hole (BNSBH) pair are an example of binary inspirals. BBH and BNSBH pair emit stronger gravitational wave signals than the BNS pair. Currently, the signal detected by aLIGO is mostly this type of signal.

1.4.2 Continuous gravitational waves signal

According to general relativity, any accelerating object will produce gravitational waves. Due to the detection limitation, we understand that the accelerating object has to be big enough so that we can detect the gravitational wave signal. Both of these conditions are met by the rotating neutron stars (Pulsars). As the pulsar's spin continuously, they should emit a gravitational wave if there is a deformity on the crust of the pulsar. If the spin is constant, the gravitational wave will have equal amplitude and frequency. As this type of gravitational wave come from a continuous spinning source, it is known as a continuous gravitational wave. The continuous gravitational wave signal will change amplitude and frequency over a longer duration for two reasons [1]. The first reason is

[1] https://www.ligo.org/science/GW-Continuous.php

that it will lose energy over time and will rotate slowly. Hence, the frequency and amplitude will change. The second reason is that the earth is rotating and if the continuous gravitational wave is detected by a ground detector then the distance between the continuous wave source and the earth will change. Thus, the frequency and amplitude of the gravitational wave signal will change. There are at least three types of pulsars, which will produce continuous gravitational waves. These are relatively young and isolated pulsars; with spin frequencies, below 50 Hz, actively accreting stars in the binary system, and recycled 'millisecond" stars: with spin frequencies above 100 Hz. From the spin frequency value, we can see that the gravitational wave signal is weak. For detecting continuous gravitational waves, aLIGO sensitivity should increase more and the ground noise should be removed more accurately as the ground noise governs in this frequency range. Detection of continuous gravitational waves along with their electromagnetic counterparts can lead us to understand the information of neutron stars' structure and the equation of state of the nuclear matter at extreme pressures more accurately.

1.4.3 Burst signal

An asymmetric explosion or implosion of the stars can also produce gravitational waves. This type of GW signal is known as burst gravitational waves. The source for this type of GW signal is a short-duration unknown or unanticipated source. The possible example of this type can be Type I supernovae, Type II supernovae, sudden release of energy from a magnetar (highly magnetized neutron star), black hole accretion disk fragmentation, pulsar glitches, and cosmic strings. Various models have been built for supernovae burst signals Szczepańczyk et al. (2021). But the burst signal is yet to be detected by aLIGO. The detection of a burst gravitational wave signal can explain the explosion mechanism of stars. It also can give the range of the energy emitted by supernovae. Moreover, it can give us information about the spot formation of black holes at their birth, asymmetric explosion, magnetar's angular momentum, and rotational state.

1.4.4 Stochastic signal

A stochastic gravitational wave signal is more like a relic gravitational wave signal. A stochastic background of gravitational waves can be produced by random, independent events during the earlier time of the universe. Mass density fluctuation during Big Bang caused gravitational waves. Just like cosmic microwave background (CMB), which was produced approximately after 300,000 years of the Big Bang, stochastic gravitational wave background was produced approximately in between 10^{-36} s and 10^{-32} s [2]. Stochastic gravitational wave background signal will have much more information about the early universe than CMB as it was produced way earlier than CMB. Stochastic gravitational wave backgrounds have stretched during the expansion of the universe and if ever detected, they can give us information about the early universe. Also, the stochastic signal can be produced by the binary inspirals during the history of the universe. Detection of these signals can give us information about that time. The model, that has been created for stochastic signals, has a very low strain value, which is well below the current sensitivity of aLIGO.

1.5 GW data and Noise

aLIGO store the recorded strain data, $h(t)$, as a time series through a calibration procedure Abbott & et al. (2020). The data is recorded by the main GW channel and sampled at 4096 Hz and 16384 Hz. There are auxiliary channels, which record the time series in addition to the strain signal. The standard time series of aLIGO data is using the Global Positioning System (GPS). Nowadays, aLIGO data is publicly available in Gravitational Wave Open Science Center (GWOSC) [3]. For saving space in the data storage system, the data has been compacted in 'hdf5' and 'frame' format.

The auxiliary channels, that have been attached to the main GW channel, not only record the strain data but also monitor environmental disturbances. Also, there are instruments in aLIGO that produce noise of various types. Therefore, the stored strain data has been impacted by many types of noises, including environmental and instrumental noises. In general, the strain data can be written as,

[2]https://www.ligo.org/science/GW-Stochastic.php

[3]https://www.gw-openscience.org/data/

$$h(t) = s(t) + n(t) \tag{1.8}$$

where $s(t)$ is the GW signal and $n(t)$ is the noise.

There are stationary and non-stationary noises in the data. These stationary noises can be removed by using data conditioning. But there are non-stationary transient noises, which will be discussed in the next chapter, that can not be removed using data conditioning. Moreover, these transient noises produce a problem to detect GW signals as these transient noises mimic GW signals.

CHAPTER II

GLITCH AND NOISE REMOVAL METHODS

2.1 Glitch

Glitch is a type of non-stationary transient noise, which mimics the GW signal. In other words, a glitch is a non-GW signal, produced by non-astrophysical sources. As it mimics GW signals, it causes problems with the detection of GW signals. Also, due to the presence of glitch, the sensitivity of detectors can not be enhanced Buikema & et al (2020), Powell (2018). The glitch also causes errors in parameter estimation Powell (2018). The sensitivity of the GW detectors is increasing day by day. Thus, the probability of the number of glitches overlapping with GW signals in the GW data is rising with the progress of detectors. Despite the overlap of the glitches with GW signals, detectors are able to detect GW signals in the case of long-duration signals (for example GW170817), But for the short-duration GW signals, it will be difficult, in some cases not possible, to distinct glitches and GW signals. Therefore, removing glitches from GW data became very important.

A relatively recent approach is that of estimating the waveform of a glitch and subtracting it out from the data. Every glitch has unique waveforms. In addition, the waveform of glitches changes slightly with the detector's design. As the glitches do not have known waveforms, it is difficult to detect, estimate and remove.

2.2 Classification of glitches

Despite having different waveforms, glitches frequently belong to a number of different broad morphological classifications. The morphology of glitches in the time-frequency domain has been used to categorize glitches, which has helped researchers understand the origins of some classes and develop successful mitigation techniques for others. For the classification of glitches, a variety of methods—from fully automated to manual—have been suggested. The Gravity Spy project Zevin et al. (2017) employs a citizen science strategy to include members of the general public in categorizing glitches through visual inspection of their Q-transform Chatterji et al. (2004) photographs. As a result, several major glitch classes with descriptive names like Blip, Tomte, and Koi fish have been identified in the observation runs of the LIGO detectors thus far. Automated classification methods based on machine learning techniques have been proposed, including support vector machines Biswas et al. (2013), t-Sne Bahaadini et al. (2018), random forests Biswas et al. (2013), S-means Mukherjee et al. (2010), and deep convolutional neural networks Biswas et al. (2013). Two automated methods have been developed to detect errors using the Q-transform in addition to machine learning techniques: Omicron Robinet et al. (2020) and a method employing the p-value of the Q-transform Vazsonyi & Davis (2022). Changes in a detector's state, its couplings to the environment, and modifications to its hardware can affect the rates of glitches within a class as well as the emergence and disappearance of the classes themselves.

2.3 Methods for removing noises

As removing glitches from GW data has paramount importance for detecting GW signals and increasing the sensitivity of detectors, various methods have been developed and applied to remove glitches. These methods have been discussed briefly in the next sub-sections.

2.3.1 Wiener Filter and Regression with Wilson-Daubechies-Meyer (WDM) transformation

aLIGO has hundreds of auxiliary channels along with the GW channel to collect data. Some of the auxiliary channels record the data from the environment and some others collect data from instruments. Seismic noise is recorded using seismometers and accelerometers. Using a seismometer and accelerometer for seismic noise, the Wiener filter (the expectation value of the square of the error signal) can be built Driggers et al. (2012). The error signal (e_s) can be defined as,

$$\mathbf{e}_s = \mathbf{n} - \mathbf{w}_y \tag{2.1}$$

where $\mathbf{n}$ is the noise and $\mathbf{w}_y = bf\omega^T\mathbf{x}$ is the approximation of the noise, which can be found using auxiliary channels. Also, ω is the tap weights of the filter, and $\mathbf{x}$ is the measurement of the external disturbance of the witness mirror. Therefore, the figure of merit for calculating the Wiener filter can be expressed as Driggers et al. (2012),

$$E[\mathbf{e}_s^2] = E[\mathbf{n}^2] - 2\omega^T p + \omega^T R\omega \tag{2.2}$$

where p is the cross-correlation vector between the witness channel and target noise, and R represents the auto-correlation matrix for the witness channels. They also applied an online adaptive filtering method (which is based on Least mean square methods), that gives the same result as wiener filter Driggers et al. (2012). This feed-forward method can be applied to higher frequency using an upgraded version of the filter Tiwari et al. (2015), using data from auxiliary channels/ physical environment monitors (PEM). This method is built by modifying the Wiener-Kolmogorov (WK) filters Wiener (1964) with regression in the time-frequency domain. The noise (n_w) in the GW strain data (h) can be predicted, using auxiliary channels (w), as Tiwari et al. (2015)

$$n[i] = \sum_{j=-L}^{L} a_j w_{i+j} \tag{2.3}$$

where the filter length is $2L+1$, and a_j is the filter coefficients. The filter coefficient can be determined by solving Tiwari et al. (2015)

$$R_{xx}a = p_{tx} \tag{2.4}$$

which is Wiener-Hopf (WH) equation. Here, R_{xx} is the auto-correlation matrix with $(2L+1)\times(2L+1)$ components, and p_{tx} is the cross-correlation matrix in between the witness channels and the target noise with $(2L+1)$ components. Also, the filter can be obtained by

$$\chi^2 = \sum_{i=L}^{N+L}\left(h[i] - \sum_{j=-L}^{L} a_j w_{i+j}\right)^2 \tag{2.5}$$

minimizing the mean square error. Regression with the WK filter has two major problems: WK filter requires long filters and computational complexity associated with the inversion of the matrix. Moreover, the filter may fail to capture all details of p_{tx} due to spectral leakage. These problems can be solved by using Wilson-Daubechies-Meyer (WDM) Necula et al. (2012) transformation. But the regression with a single channel is not effective. Thus, multiple channels have been used to do the regression, and equation 2.5 becomes

$$\chi^2 = \sum_{i=L}^{N+L} \left(h[i] - \sum_{j=-L}^{L} a_j w_{i+j} - \sum_{k=-L}^{L} b_k w_{i+k} - \sum_{m=-L}^{L} c_m w_{i+m} - \dots \right)^2 \tag{2.6}$$

The introduction of multiple channel analysis illustrated two shortcomings: the matrix can be ranked deficient if auxiliary channels are highly correlated, and it can add noise if a significant fraction of the auxiliary channels are not correlated with target channels. For solving this problem, regulators have been used Tiwari et al. (2015).

2.3.2 Method for mitigating bilinear noise and scattering glitch using test mass

Estimating bilinear noise by constructing a coherent bilinear noise filter using narrow-band noise in the signal recycling test mass Mukund et al. (2020), can be mitigated as well. For improving the accuracy of the estimation, the adaptive Bayesian approach has been applied. Using the photodiodes in the end benches of VIRGO detectors, the scattered light glitch has been modeled and subtracted from the data Wąs et al. (2021).In the case of modeling, it used the sine and cosine function but in practice, it used the tanh function which gives better results and flexibility.

2.3.3 *Bayeswave*

According to the assumption of the *BayesWave*, the GW strain data $[h(t)]$ can be divided into three parts: these are Gaussian noise $[n(t)]$, signal $[s(t)]$, and glitch-transient noise $[g(t)]$. Mathematically, $h(t)$ can be written as,

$$h(t) = n(t) + s(t) + g(t) \tag{2.7}$$

Using this notion, the *BayesWave* algorithm estimates non-Gaussian features using Morlet-Gabor wavelets Cornish & Littenberg (2015), Pankow et al. (2018), which is a sum of sine-Gaussian. The wavelets, for time series, can be written as

$$\psi(t;\alpha) = Ae^{-\frac{(t-t_0)^2}{\tau^2}} cos\left[2\pi f_0 (t-t_0) + \phi_0\right] \tag{2.8}$$

Here, $\alpha = (A, t_0, q, f_0, \phi_0)$ (amplitude, central time, quality factor $[q = 2\pi f_0 \tau]$, central frequency, phase respectively) are the wavelet parameters. *BayesWave* uses Bayesian inference to model non-stationary data, also the number and parameters of wavelets are not pre-determined. The number and parameter of wavelets were marginalized using Markov chain Monte Carlo (MCMC) Cornish & Littenberg (2015). Using the coherence in between detectors, the glitch and signal, non-stationary sources, can be determined. If the non-stationary source is coherent in between the detectors, than the model for this part considered as signal $[s(t)]$ while the incoherent part is considered as glitch $[g(t)]$ Pankow et al. (2018). In the case of a single detector signal, the glitch modeling from auxiliary channels can be used to remove the glitches around the signal Davis et al. (2022).

2.3.4 *gwsubtract*

gwsubtract is a linear subtraction method Davis et al. (2019). It uses auxiliary channels to determine transfer function $\left[T_f\right]$. Using this transfer function into the strain data, it estimates a glitch and other noises. After that, the glitch is removed by subtracting the estimate from the data. In case of *gwsubtract* method, the assumption is that the linear combination of timeseries from various sources $\left[n_j(t)\right]$ produce the GW strain data $[h(t)]$. Or the assumption can be considered as $h(t)$ consists a signal $[s(t)]$, which is not correlated with the noise Allen et al. (1999), in the GW strain data. Also, another important assumption is that the multiplication of the convolution of a witness time series $[w(t)]$ and an unknown transfer function $\left[T_{fwh}(t)\right]$ can model the noise, at least one of the noises Allen et al. (1999), Davis et al. (2022). Therefore, $h(t)$ can be written as,

$$\begin{aligned} h(t) &= n_1(t) + n_2(t) + n_3(t) + \cdots + n_{j-1}(t) + w(t) \times T_{fwh}(t) \\ &= h^{'}(t) + w(t) \times T_{fwh}(t) \end{aligned} \tag{2.9}$$

In equation 2.9, $h^{'}(t)$ has all of the property of strain data, except one noise sources modeled by the $w(t)$ and the $T_{fwh}(t)$. In frequency domain, the time domain equation 2.9 can be expressed as

$$\tilde{h}(f) = \tilde{h}^{'}(f) + \tilde{w}(f) \times \tilde{T}_{fwh}(f) \tag{2.10}$$

From the derivation from Allen et al. (1999), Davis et al. (2019) , the transfer function $\tilde{T}_{fwh}(f)$ for frequency band $\left[f_{j-1}, f_j\right]$ can be written as,

$$\tilde{T}_{fwh}(f') = \frac{df}{f_j - f_{j-1}} \sum_{f_{j-1}}^{f_j} \tilde{Y}_h(f) \tilde{Y}_w^*(f) \tag{2.11}$$

where $\tilde{Y}_h(f)$ is the discrete Fourier transform of $h(t)$, $\tilde{Y}_w^*(f)$ is the discrete Fourier transform of $w(t)$, and df is the frequency resolution of the data. The chance correlations measurement has been reduced by averaging nearby frequencies during the transfer function calculation Davis et al. (2022), Allen et al. (1999). Note that the inner product in the transfer function equation can be determined if $h(f)$ and $w(f)$ are sampled quickly.

2.3.5 *glitschen*

glitschen is a data-driven, parametric glitch mitigation model Merritt et al. (2021), which uses probabilistic principal component analysis (PPCA) methodTipping & Bishop (1999). In *glitschen*, an isotropic Gaussian noise model with a $d-$ dimensional observation vector $\left(\tilde{d}\right)$ can be expressed as,

$$\tilde{\mathbf{d}}|\mathbf{X}_t \sim N\left(\mathbf{W}\mathbf{X}_t + \mu, \sigma^2\mathbf{I}\right) \tag{2.12}$$

where the mean of isotropic Gaussian noise ($\mu = 0$)is replaced by $\mathbf{W}\mathbf{X}_t + \mu$, σ^2 is variance, and $\mathbf{I}$ is the identity matrix. Also, $\mathbf{X}_t \sim N(0, \mathbf{I})$ is marginalized by r latent variables of the training set, and $\mathbf{W}$ has dimensions $d \times r$ with r training eigenvectors.

Following the derivation of Merritt et al. (2021), Tipping & Bishop (1999), a new observation vector $\left(\tilde{\mathbf{d}}_o\right)$ can be written as

$$\mathbf{X}_t|\tilde{\mathbf{d}}_0 \sim N\left(\mathbf{M}^{-1}\mathbf{W}^T\left(\tilde{\mathbf{d}}_o - \mu\right), \sigma^2\mathbf{M}^{-1}\right) \tag{2.13}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$, with size $r \times r$.

Using this observation vector, the glitch can be modeled by

$$\tilde{\mathbf{g}}_m = \mathbf{W}\mathbf{X}_m + \mu \tag{2.14}$$

where $\mathbf{X}_m \equiv \mathbf{M}^{-1}\mathbf{W}^T\left(\tilde{\mathbf{d}}_o - \mu\right)$. The quality of the glitch model is determined by the standard Gaussian noise likelihood.

2.3.6 *DeepClean*

DeepClean is a noise removal method Ormiston et al. (2020), which uses deep learning and one dimensional convolutional neural network (CNN). For this method, the GW strain data $[h(t)]$ is considered to be the combination of fundamental noise $\left[n_f(t)\right]$, signal $[s(t)]$, and other noise $[n_w(t)]$. Therefore, $h(t)$ can be expressed as

$$h(t) = s(t) + n_f(t) + n_w(t) \tag{2.15}$$

Here, $n_f(t)$ is not desired to be subtracted, while $n_w(t)$, which couples into the witness/auxiliary channels $\left[w_j(t)\right]$, is needed to be subtracted. In this method, the data from auxiliary channels, after preprocessing, is sent to CNN. For estimating $n_w(t)$, CNN work as a function $\mathscr{F}\left(w_j(t);\overrightarrow{\beta}\right)$ on $w_j(t)$. Here, $\overrightarrow{\beta}$ is a set of parameters,that can be determined by

$$\overrightarrow{\beta} = argmin_{\overrightarrow{\beta}'} \mathscr{J}\left[h(t), \mathscr{F}\left(w_j(t);\overrightarrow{\beta}'\right)\right] \tag{2.16}$$

where $\mathscr{J}$ is a loss function, which is determined by the sum of the loss function $\mathscr{J}_{asd}$ (the weighted average of the amplitude spectral density (ASD) of the residual strain $[r(t)]$ [where $\left[r(t) = h(t) - \mathscr{F}\left(w_j(t);\overrightarrow{\beta}\right)\right]$] and the time domain loss function $\mathscr{J}_{mse}$ (the mean square error (MSE) across the time series). Therefore, $\mathscr{J}$ can be written as,

$$\mathscr{J} = m\mathscr{J}_{asd} + (1-m)\mathscr{J}_{mse} \tag{2.17}$$

where m is a weighting factor ranging from 0 to 1.

Using this similar method, a CNN method has been built to remove glitches Mogushi (2021). The difference between *DeepClean* and this CNN method is that it has two-dimensional CNN. The non-linear activation, *ReLU*, has been used for this CNN method.

2.3.7 Method for mitigating angular noise

A CNN method Yu & Adhikari (2021) has been built for mitigating angular noise using the code *Keras* Chollet et al. (2018)(a deep learning code written in python), which is running on *TensorFlow* Abadi et al. (2015). Angular noise is caused by a geometrical effect of the rotational pivot of aLIGO test mass. Also this nonlinear noise couples with the GW readout. The angular motion in the test mass can be expressed as Yu & Adhikari (2021),

$$\delta x^{(mir)}(t) = x^{(mir)}_{spot}(t)\,\theta^{(mir)}(t) \tag{2.18}$$

where $\delta x^{(mir)}(t)$ is a linear length fluctuation, $x_{spot}^{(mir)}(t)$ is a slow ($\lesssim 1\ Hz$) motion of the beam spot on the test mass, and $\theta^{(mir)}(t)$ is a fast ($\gtrsim 10\ Hz$) motion of the beam spot on the test mass. The slow motion induced by the seismic motion and the fast motion, angular perturbation of the mirror, induced by the sensing noise in the angular control system Yu & Adhikari (2021).

In this method, there are two types of CNN structure: General CNN and Specific CNN. General CNN consists of sufficiently many convolutional layers. These layers are densely connected and at least some of these layers have a nonlinear activation function. For the layers with nonlinear activations, the coupling mechanism can be expressed with series expansion and the convolutional layer will act as a finite-impulse-response (FIR) filter. However, using the knowledge of angular noise, a more specific CNN structure can be built, which consists of the auxiliary channels of a slow CNN and the auxiliary channels of a fast CNN. These sets, slow and fast, require only linear activation. This method also works as an FIR filter to convert the linear outputs of auxiliary channels into digital counts. The nonlinearity of the noise can be determined by using equation (2.12), i.e. by multiplying layers. In both CNN structures, a loss function has been applied which is similar to the loss function of *DeepClean*. For getting better noise mitigation, CL training and techniques George & Huerta (2018*a*,*b*) can be used with this method.

2.4 Discussion on the methods

Feed-forward methods can increase the sensitivity of aLIGO detectors, along with it can enhance several percent of detectable inspiral range Meadors et al. (2014). Wiener filter, a feed-forward method, is applied to only seismic noise below 10 Hz, more specifically, it was successful to reduce noise in between $0-7$ Hz Driggers et al. (2012). Feed-forward methods can be applied up to 150 Hz, and using auxiliary channels also increases the sensitivity of the detectors Tiwari et al. (2015). The system, built with the Wiener filter, is applied so that the seismic noise below 10 Hz can be reduced. It will not be applied above 10 Hz as it can remove gravitational waves along with the noise. From the results, certainly it reduced the seismic noise and smooth the data, but it can not smooth all the parts. Moreover, it is applied at a very low frequency. Also, it is not successful to remove narrow-band and broad-band noises. The method, build with regression, is also successful in removing bilinear noise, which is produced by linear coupling of seismic noise and resides near the power lines and calibration lines Tiwari et al. (2015). The nonlinear cases are a challenge for this system. They had to input the data from multiple channels to confirm the estimation. In this case, it overfits the data, along with the system having a problem with rank deficiency of linear equations. Also, it is applied for only low-frequency noise, below 150 Hz to be exact. The regulator solves the problem related to multiple channel analysis. But the value of regulators can not be generalized. The value of the regulator and the regression parameters should be changed with the noise structure.

The bilinear noise removal method depends on how well the system makes the model of narrow-band noise for the success of this method. It works on bilinear noise Mukund et al. (2020), but it is not sure that it will work on glitch as glitch has various waveforms. The scattered glitch that has been mitigated is in between $30-55$ Hz Wąs et al. (2021). If a scattered light comes in a higher frequency, it can not remove completely. Also, what is the effect of this mitigation on the GW signal, is not explained.

The success of the glitch removal, using *BayesWave*, depends on how well the signal is modeled Cornish & Littenberg (2015). The coherence method Pankow et al. (2018) can not model a glitch when the signal is only detected in one detector (for example, GW190424). For the success of the glitch removal method from a single detector, an auxiliary channel should model the glitch very well or it will affect the signal. *BayesWave* can not model a glitch accurately if the glitch duration is > 1s Davis et al. (2022). Hence, it can not accurately model scattering glitches Davis et al. (2022) or any other glitch that is longer than 1s. Although *BayesWave* can model a glitch more accurately when the glitch duration is $\lesssim 1$s, it has an error percentage depending on the glitch model from auxiliary channels and may bias the result slightly.

The GW strain data are non-stationary and non-Gaussian due to noise artifacts of various duration Abbott et al. (2018). Therefore, *gwsubtract* is not a suitable choice for removing glitches in maximum cases. If the glitch system is linear, *gwsubtract* can remove the glitches Davis et al. (2022). But in this case, it will depend on the noise model made by the auxiliary channels. Though the statistical error and systematic uncertainties from *BayesWave* and *gwsubtract* are low Hourihane et al. (2022), it has a larger impact on the measured source properties Payne et al. (2022).

The method explained in *glitschen* is fast and the training for this method is computationally cheap. The method can model glitches (Blip, Tomte) and subtract them from the data successfully. Also, the method can differ between astrophysical signals and glitches in some cases. The problem is that the method requires a lot of training data and training model Merritt et al. (2021). The glitch modeling depends on how high the SNR of the glitch in the data is Merritt et al. (2021). Also, as the sensitivity of aLIGO is increasing, the waveforms of the glitches are changing. So, the method will require identifying some glitches in the selected observing run at first and then training the system to estimate and remove the glitches. Moreover, the method can not differentiate between an unknown astrophysical signal and a new form of glitch, which may cause confusion in the system and it can remove the astrophysical signals. *glitschen* performs better in the case of tomte than a blip. The tomte glitch, that has been described in the work Merritt et al. (2021), has a very low frequency. Thus, it can be said that the method may work better below 100 Hz but above that, this system does not give its best performance. *BayesWave* is also applied for modeling the glitches and the astrophysical signals Chatziioannou et al. (2021). *BayesWave* can successfully distinguish the glitches and astrophysical signals in the case of binary black hole (BBH) signals, but the system needs to be trained with various signals before applying it Chatziioannou et al. (2021). This is a problem as many astrophysical signals are still not modeled and known. Therefore, the performance of *BayesWave* depends on how well the method is trained. That means *BayesWave* needs a plethora of data to train itself first. Also, the limitations, previously discussed, persist in this case as well.

DeepClean is mainly applied to remove seismic noise, jitter noise, and sidebands. It works as a Weiner filter but it has some advantages as well. It is not strictly used for linear couplings, instead, it uses the $tanh$ function, and thus, it can discern some nonlinear features. One of the main advantages of *DeepClean* is that it can determine linear, nonlinear, and nonstationary couplings without any prior knowledge of these noises i.e. without prior training of the particular noise type Ormiston et al. (2020). Though a fraction of the scattering glitch can be removed using two dimensional CNN method, it could not remove it completely or the maximum part of the glitch Mogushi (2021). The method does not perform well on Extremely loud glitches as scattering glitches Mogushi (2021). That means the estimation using the CNN method is a small fraction of the extremely loud glitch. One of the positive things about this method is that this method does not take out too much power from the data. For the training of the system, it uses auxiliary/witness channels. But adding auxiliary channels, which are irrelevant in case of a particular glitch, can cause uncertainty to the subtraction Ormiston et al. (2020), Mogushi (2021). Hence, the selection of auxiliary channels for training is very crucial for these methods. Moreover, the length of training data depends on the complexity of the particular noises. In general, CNN needs a lot of data to train itself. Also, the computational cost for CNN techniques is relatively high.

For the simple structure of general CNN in the method for angular noise mitigation, it needs small knowledge of noises to mitigate noise even with unknown couplings Yu & Adhikari (2021). Conversely, detailed knowledge of noise is needed to build specific CNN. Also, the challenging part for specific CNN is that the reconstruction of the spot motion $x_{spot}^{(mir)}(t)$ on test mass. The performance of general CNN and specific CNN in noise subtraction is decent and comparable in between $10-20$ Hz Yu & Adhikari (2021). But the performance is better for specific CNN in between $6-10$ Hz. Also, after 30 Hz, which is the end of the training band, general CNN starts to add noise while specific CNN continues to remove noises from data. On the real aLIGO data, the broadband reduction of nonlinearity is not that significant Yu & Adhikari (2021). Another significant condition for these methods is the value of the signal-to-noise ratio (SNR). A slight change in the SNR has negatively affected the noise subtraction significantly.

From the discussion above, the methods explained above are effective on linear noises in various frequencies. In aLIGO, the linear component of noises has been mitigated successfully Davis et al. (2019), Driggers et al. (2019). However, in case of glitches, *BayesWave*, *gwsubtract* and a two dimensional CNN structure similar to *DeepClean* can be applied. But none of these methods can work without the model of glitches from auxiliary channels. Also, in some cases, auxiliary channels can not catch glitch waveform accurately and in these cases, these methods will fail. It requires a lot of training to train the model for CNN base methods. *gwsubtract* can not remove glitches which is not linearly dependent. Though *glitschen* is successful for removing glitches at a certain level, it requires prior knowledge of glitches to remove the glitch effectively. Moreover, it has uncertainties for estimating and mitigating the wing of the glitches. That is why we are inspired to apply a non-parametric regression method and a wavelet-based method (not the *BayesWave*), which will be explained in the next chapter, to mitigate glitches.

CHAPTER III

GLITCH SUBTRACTION USING ADAPTIVE SPLINE FITTING AND WAVESHRINK

3.1 Adaptive spline fitting

The fundamental algorithm behind our glitch subtraction technique, SHAPES Mohanty & Fahnestock (2020), is derived using the following models for the signal and noisy data, $\overline{s}(\theta)$ and $\overline{y}$, respectively.

$$\overline{y} \;=\; \overline{s}(\theta)+\overline{\varepsilon}\,, \tag{3.1}$$

where $y_i = y(t_i)$ and $s_i(\theta) = s(t_i;\theta)$, $i = 0,1,\ldots,N-1$, are sampled values at t_i with $t_0 = 0$, $t_{N-1} = 1$, $t_{i+1} > t_i$, and θ signifies the set of signal parameters that need to be estimated from the data, $\overline{y}$, $\overline{s}$, and $\overline{\varepsilon}$ are row vectors with N elements. The noise samples, ε_i, are randomly selected from the normal (Gaussian) probability density function with zero mean and unit variance, $N(0,1)$. Since GW data is always whitened using the estimated noise power spectral density (PSD), this assumption—that of a white Gaussian noise process—does not result in a loss of generalization.

Since it is assumed that the signal $s(t;\theta)$ is a spline of order k, it can be represented by a linear combination of B-spline functions de Boor (2001).

$$s\,(t;\theta=\{\overline{\alpha},\overline{\tau}\}) \;=\; \sum_{j=0}^{P-k-1}\alpha_j B_{j,k}(t;\overline{\tau})\,, \tag{3.2}$$

where $\overline{\alpha} = (\alpha_0,\alpha_1,\ldots,\alpha_{P-k-1})$, and $\overline{\tau} = (\tau_0,\tau_1,\ldots,\tau_{p-1})$, $tau_{i+1} \geq \tau_i$ is a series of P knots that identifies the end points of the continuous intervals holding the cubic polynomial components composing the spline.

The set of B-spline functions, $\{B_{i,k}(x;\overline{\tau})\}$, for any given order can be efficiently computed using the Cox-de Boor recursion relations in de Boor (1972). The recursions begin with piecewise constant functions of order 1 B-splines as

$$B_{j,1}(t;\overline{\tau}) = \begin{cases} 1, & \tau_j \leq x < \tau_{j+1} \\ 0 & \text{else} \end{cases} . \tag{3.3}$$

For $2 \leq k' \leq k$,

$$\begin{aligned} B_{j,k'}(t) &= \omega_{j,k'}(t)B_{j,k'-1}(t) + \gamma_{j+1,k'}(t)B_{j+1,k'-1}(t) , && (3.4)\\ \omega_{j,k'}(x) &= \begin{cases} \frac{t-\tau_j}{\tau_{j+k'-1}-\tau_j} , & \tau_{j+k'-1} \neq \tau_j \\ 0 , & \tau_{j+k'-1} = \tau_j \end{cases} , && (3.5)\\ \gamma_{j,k'}(x) &= \begin{cases} 1-\omega_{j,k'}(t) , & \tau_{j+k'-1} \neq \tau_j \\ 0 , & \tau_{j+k'-1} = \tau_j \end{cases} . && (3.6) \end{aligned}$$

In the aforementioned recursion, $0 \leq j \leq P-k'-1$. Cubic ($k=4$) B-spline functions are shown in Fig. 3.1. Noting that knots may repeat up to k times, it should be noted that the spline is continuous across the knot up to derivatives of order $k-2$ in the absence of repetition and becomes discontinuous with each repetition, ending in a jump discontinuity in the spline itself.

$\widehat{\alpha}$ and $\widehat{\tau}$ are the spline parameters that minimize a penalized least-squares function, and they provide the best fit,

$$\begin{aligned} L_\lambda(\overline{\alpha},\overline{\tau}) &= L(\overline{\alpha},\overline{\tau}) + \lambda R(\overline{\alpha}) , && (3.7)\\ L(\overline{\alpha},\overline{\tau}) &= \sum_{i=0}^{N-1} (y_i - s_i(\overline{\alpha},\overline{\tau}))^2 , && (3.8) \end{aligned}$$

where the penalty term,

$$R(\overline{\alpha}) = \sum_{j=0}^{P-k-1} \alpha_j^2 , \tag{3.9}$$

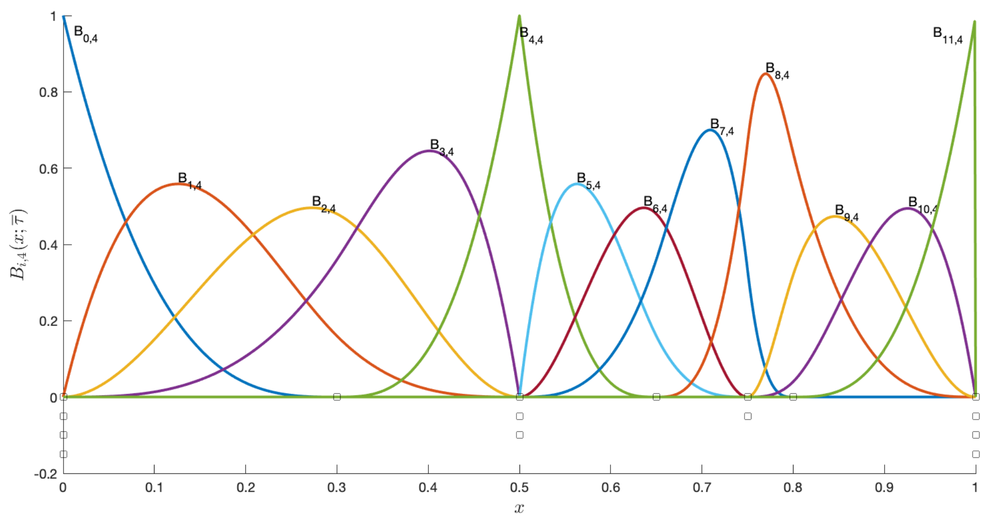


Figure 3.1: Cubic B-spline functions $\{B_{i,4}(t;\overline{\tau})\}$, $i = 0, 1, \ldots, 11$, for an arbitrary choice of 16 knots ($\overline{\tau}$) marked by squares. Knots with multiplicity > 1 result in B-splines that are discontinuous in value or derivatives.

is discovered to be effective in the inhibition of erroneous knot clustering. When the best-fit spline attempts to reduce $L_\lambda(\overline{\alpha}, \overline{\tau})$ by fitting out anomalous data points caused by noise alone, clusters like these are seen. In the present version of SHAPES , the penalty gain factor, λ, is user-specified: a greater value of λ forces the best suited spline to be smoother.

Given that they exist linearly in the signal model, optimizing over $L_\lambda(\overline{\alpha}, \overline{\tau})$ over $\overline{\alpha}$ is simple. However, optimizing over $\overline{\tau}$ has long been a hurdle Wold (1974), Burchard (1974), Jupp (1978), Luo & Wahba (1997) when implementing adaptive spline fitting. The advantages of optimizing knot location have also been amply established at the same time. This has inspired a variety of approaches for the knot optimization step in the literature. It is evident from Gálvez & Iglesias (2011), Mohanty (2012) that Particle Swarm Optimization (PSO) Kennedy & Eberhart (1995), Mohanty (2018), a well-liked nature-inspired metaheuristic for global optimization of high-dimensional non-linear and non-convex functions, provides for significant advancement on this topic. Additionally, because PSO is a continuous optimization approach, it is able to investigate all knot arrangements, including those in which many knots are sufficiently close to one another to be combined into a single knot with a larger multiplicity. As a result, functions with a mixture of smooth and non-smooth elements can be fitted. Adaptive spline fitting, which is a multi-resolution analytic technique akin to the wavelet transform, uses basis functions that are data-driven rather than fixed.

We give a brief summary of how SHAPES implements the salient concepts mentioned above and direct the reader to Mohanty & Fahnestock (2020) for more information. In the nested form, $L_\lambda(\overline{\alpha}, \overline{\tau})$ minimization is carried out as,

$$\begin{aligned} \min_{\overline{\tau}, \overline{\alpha}} L_\lambda(\overline{\alpha}, \overline{\tau}) &= \min_{\overline{\tau}} F_\lambda(\overline{\tau}) \,, && (3.10) \\ F_\lambda(\overline{\tau}) &= L_\lambda(\widehat{\alpha}(\overline{\tau}), \overline{\tau}) \,, && (3.11) \end{aligned}$$

where the solution, $\widehat{\alpha}(\overline{\tau})$, of the inner minimization is expressed in terms of the $(P-k)$-by-N matrix $\mathbf{B}(\overline{\tau})$, with elements

$$[\mathbf{B}(\overline{\tau})]_{m,n} = B_{m,k}(t_n; \overline{\tau}) \,, \tag{3.12}$$

as

$$\begin{aligned} \widehat{\alpha}(\overline{\tau}) &= \overline{y}\mathbf{B}^T \mathbf{G}^{-1} \,, && (3.13) \\ \mathbf{G} &= \mathbf{B}\mathbf{B}^T + \lambda \mathbf{I} \,, && (3.14) \end{aligned}$$

where $\mathbf{I}$ is the identity matrix of size $(P-k)$-by-$(P-k)$. The best-of-M runs method, which involves running many independent PSO runs in parallel and selecting the one that yields the lowest final value to produce $\widehat{\tau}$, is used to minimize $F\lambda(\overline{\tau})$. The two hyperparameters P and λ determine the estimations of $\widehat{\alpha}$ and $\widehat{\tau}$ that are produced at the conclusion. The optimal value of P is chosen through model selection using the Akaike Information Criterion (AIC) Akaike (1998). Although λ is still a user-specified parameter in the current version of "SHAPES," it is discovered to be fairly robust across a number of benchmark functions.

3.2 Wavelet shrinkage

The wavelet transform Donoho & Johnstone (1995) is a wavelet shrinkage system, which is based on multivariate normal decision theory, and it provides a multiresolution analysis of data. For analyzing nonstationary features such as glitches, multiresolution analysis of data performs better than the Fourier transform. The wavelet transform is a decomposition of the data on an orthogonal basis set, just like the Fourier transform. Even so, Fourier basis functions are only localized in frequency whereas the basis functions in a wavelet transform are localized in both time and frequency. That means the wavelet transform is scaled more precisely than the Fourier transform.

In Donoho & Johnstone (1995), the wavelet transform was studied thoroughly as the basis for a family of nonparametric curve fitting methods. In wavelet transform methods, the starting point is the set of wavelet coefficients $\overline{w}$ obtained as,

$$\overline{w}^T = \mathbf{W}\overline{y}^T , \tag{3.15}$$

where $\mathbf{W}$ is a matrix which contains the wavelet basis functions sampled at t_i, $i = 0, 1, \ldots, N-1$ as rows, and $\overline{y}$ represents the data as assumed to the model in Eq. 3.1. A dyadic scheme, $N = 2^J$, is used for labeling the wavelet basis functions. Mathematically, it can be expressed as

$$W_{jk}(t_i) \propto \psi\left(2^j\left(t_i - k2^{-j}\right)\right) , \tag{3.16}$$

$j = 0, 1, \ldots, J-1$, $k = 0, 1, \ldots, 2^j - 1$, where $\psi(t)$ is the mother wavelet that defines a given family of wavelet basis functions.

Next, a non-linear function $\mathscr{T}_\eta(x)$ is used to transform each wavelet coefficient,

$$\mathscr{T}_\eta(x) = \operatorname{sgn}(x)(|x| - \eta)_+ , \tag{3.17}$$

which applies a threshold η to its argument x. $\mathscr{T}_\eta(x)$ returns $x - \eta$ if $|x| > \eta$ and 0 otherwise.

Lastly, the estimate $\widehat{s}$ of the signal $\bar{s}$ is acquired by inverting the transformed coefficients $\widetilde{w}$,

$$\widehat{s} \; = \; \widetilde{w}W \, . \tag{3.18}$$

The function $\mathscr{T}_\eta(x)$ is known as a soft threshold. $\mathscr{T}_\eta(x)$ shrinks threshold-crossing coefficients towards the threshold. Introducing a small bias, the shrinkage of coefficients in large linear models is one of the standard forms of regularization that reduces variance in the estimate $\widehat{s}$. In this case, strong features of the signal are preserved well due to the shrinkage is relatively smaller for larger wavelet coefficients.

Different members in the family of wavelet shrinkage methods can be achieved by setting the value of η. In this work, the `WaveShrink` function provided in the Matlab package `WaveLab` Huo et al. (2000) is used to implement wavelet shrinkage with the “Hybrid" method for setting the threshold. Another parameter the coarsest resolution level, L, also needs to be specified in this code for the wavelet transform method such that $j \geq L$ in Eq. 3.16. There is no significant difference in the results for $L \in \{1,2,4\}$, hence set $L = 1$.

3.3 Glitch estimation and subtraction method

The conditioning of the GW strain data using the subsequent stages in a sequential fashion is the initial step in our glitch estimation and subtraction method. Suppression of seismic noise below 10Hz, robust estimation of the noise floor power spectral density (PSD), whitening of the noise floor using the estimated PSD Mukherjee (2003), automated identification of high-power narrowband noise ("lines") features and their suppression using notch filters, and downsampling of the data to a sampling frequency of 2048Hz are all examples of noise reduction techniques. The CBC and CWb burst searches, the current flagship GW searches, may be negatively impacted by the glitches that may be preserved by reducing the sample frequency while still processing the data quickly. We won’t spend space on their description here because all of these procedures are common to GW search pipelines.

We have two approaches that we have devised to locate the problem. This section’s explana-

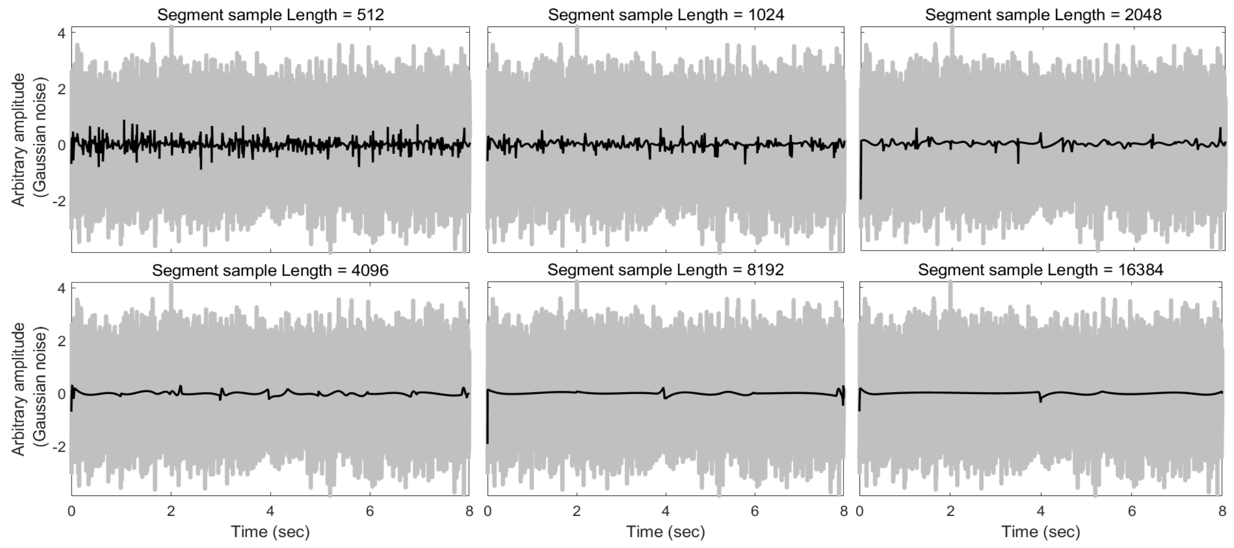


Figure 3.2: Performance of SHAPES using different segment length on Gaussian noise

tion of the terms amplitude threshold and spectrum threshold will follow. The data is separated into overlapping sub-segments of lengths ranging from 8192 to 16384 samples after the location of a glitch has been determined (except the glitch part of the data). According to Fig. 3.2, fitting takes longer the longer the sub-segment length. With the exception of the glitch, 8192 sub-segments are suitable in this situation for a shorter period of time, and the fitting is flat (there are only little ups and downs where one segment joins the next). Despite taking longer to fit the data using 16384 sub-segments, they are more flat on the data overall (apart from the glitch) than the 8192 segment length (see Fig. 3.2). Currently, the specific selection of segment lengths in the glitch part of the data is modified manually according to the glitch class being studied. After that, the code receives the GPS timing for the problem. After being received the data and identification of the data, complete data is upsampled from 4096 samples to 16384 samples. Then, the data will be separated into sub-segments of 16384 samples, excluding the GPS time for the glitch. The sub-segment length will get shorter as the GPS time for the glitch gets closer—more specifically, 2^i o $\frac{2^i + 2^{i+1}}{2}$ r samples where $i = 8, 9, 10, 11, 12$. More knots are tied in the data by smaller sub-segments. As a result, it provides an estimate of the glitch waveform by fitting the glitch portion of the data with more knots. Additionally, depending on the selection of samples, it will return to the segment length of 16384 samples when it approaches the end of the added GPS time for the glitch.

Applying SHAPES (c.f., Sec. 3.1) on each of the segments mentioned above is the next step. A weighted average is used to combine the estimates obtained in two neighboring windows in the overlap region. Let the overlap interval's borders be $[t_L, t_R]$ and the SHAPES estimates from the segments that finish and start at t_R and t_L, respectively, be $\widehat{s}_L(t)$ and $\widehat{s}_R(t)$. Define the weights $w_L(t)$ and $w_R(t)$ such that $w_{L,R}(t) = 0$ for $t \notin [t_L, t_R]$ and for $t \in [t_L, t_R]$, respectively,

$$w_L(t) = \exp\left(-d\frac{t - t_L}{t_R - t_L}\right), \tag{3.19}$$

$$w_L(t) = \exp\left(d\frac{t - t_R}{t_R - t_L}\right), \tag{3.20}$$

where d is a user-specified parameter that determines the weights' lowest value. After that, the estimated function over the combined neighboring segments is given by,

$$\widehat{s}(t) = \begin{cases} \widehat{s}_L(t) & t < t_L \\ \frac{\widehat{s}_L(t)w_L(t)+\widehat{s}_R(t)w_R(t)}{w_L(t)+w_R(t)} & t_L \le t \le t_R \\ \widehat{s}_R(t) & t > t_R \end{cases}. \tag{3.21}$$

It should be noted that the $\widehat{s}(t)$ at $t = t_{R,L}$ are not guaranteed to be continuous or differentiable by weighted averaging. But in actual use, we see that the discontinuities are barely perceptible. Given that both $s_L(t)$ and $s_R(t)$ are splines and naturally admit constraints of continuity and differentiability at defined connection sites, more complex methods for connecting $s_L(t)$ and $s_R(t)$ are possible. Future research will look into these, but they are not very significant in the context of this paper.

3.3.1 Amplitude Threshold

In Fig. 3.3, GW170817 exhibits this loud blip glitch between 2047s and 2048s. The GW170817 data also has a small glitch between the 4080s and 4081s, which is almost the end of the data. In the case of fast scattering, it blends in with the background noise. So, in order for the data to be released, we must filter it. After filtering, we can once more notice that the repeated pattern

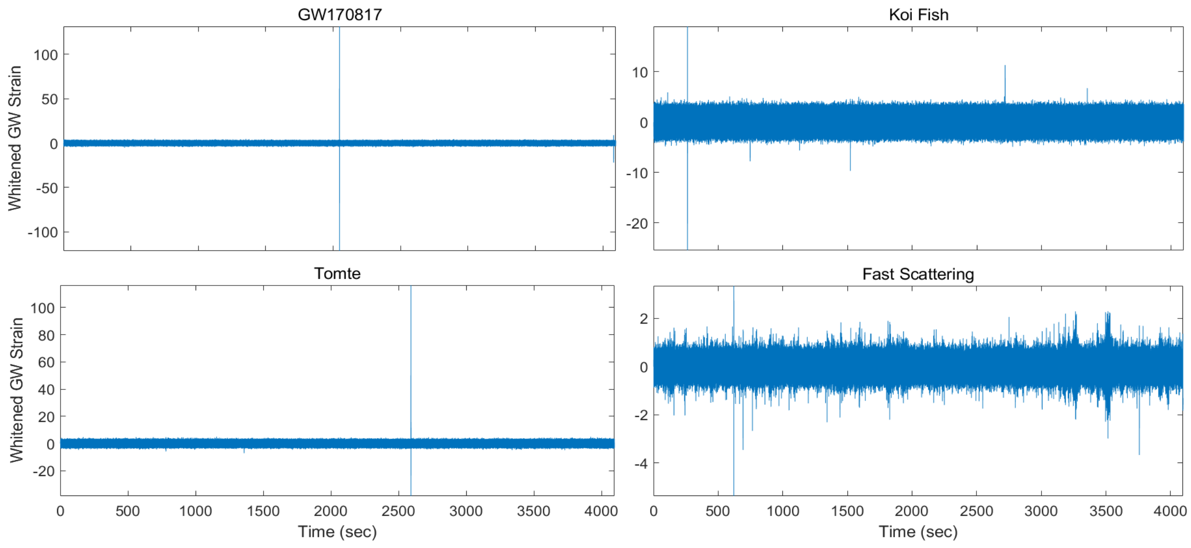


Figure 3.3: Amplitude of glitches

is quite similar to the glitch between the 3490s and 3540s. This data has numerous fast-scattering glitches in various locations. Moreover, we have tomte glitch in 2591 s and 2593s, and koi fish glitch in between 262s and 263s. Similarly, all other glitches, that we have dealt with, have higher amplitudes compared to the general noise amplitude. Thus, we have concluded that glitches have high amplitude than the general noise of the data. The absolute values of the data samples are subjected to a threshold η in order to pinpoint the locations of glitches in a given segment of data. After exceeding the threshold, samples are separated into clusters. Such that each cluster has a distinct start time and end time and that the samples within each cluster are chronologically sequential. The distinction between these clusters is a crucial factor in bringing the clusters together and forming the complete glitch. For one glitch , there is not much of a difference between these clusters. The difference between one cluster and another cluster has a big value, however, if it is a separate glitch. As a result, we can separate various glitches using the difference between clusters [figure 3.4]. A section of the data is taken for each glitch that contains the specific cluster of the glitch and two 100 consecutive samples on either side of it. If the threshold is applied to the conditioned data alone for particular classes, such as slow and fast scattering glitches, the glitches may be overlooked because they are not very loud. In these situations, the conditioned data is low pass filtered one more before the threshold is applied. Even though it is straightforward, this method effectively locates glitches in all the classes we have taken into consideration.

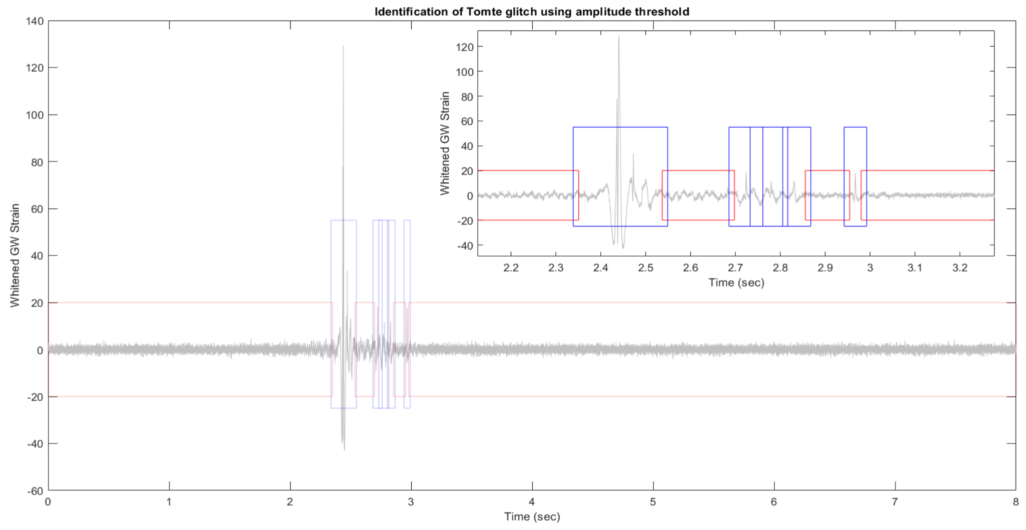


Figure 3.4: Glitch identification using amplitude threshold

3.3.2 Bestfitness-spectrogram Threshold

This glitch identifying method use SHAPES to identify the glitch. During the estimation made by SHAPES, SHAPES chooses the estimation based on the best fitness value of arbitrary runs. During the estimation, SHAPES always has a high value of fitness at a segment [Figure 3.5], where the glitch is present. It has the nearly same value for the Gaussian/general noise all over the data with a little deviation, as we can see from the top panels between approximately 10 and 2000, in the bottom left panel before 4 and after 6, in the bottom right panel up to 2 and after 4 in fig. 3.5. Even with the added chirp signal, the value for Gaussian/general noise remains nearly the same. Thus, taking a mode of all the best fitness values give us the best fitness value for Gaussian/general noise, let us call it the general best fitness value. Adding 5/10 with this value will exceed the standard deviation from the best fitness value determined by the mode. The value that we added with the general best fitness value depends on the sampling frequency of the data. In practice, adding 5 works fine in this algorithm.

The minimum value we get at the last segment is due to the fewer samples left at the end of

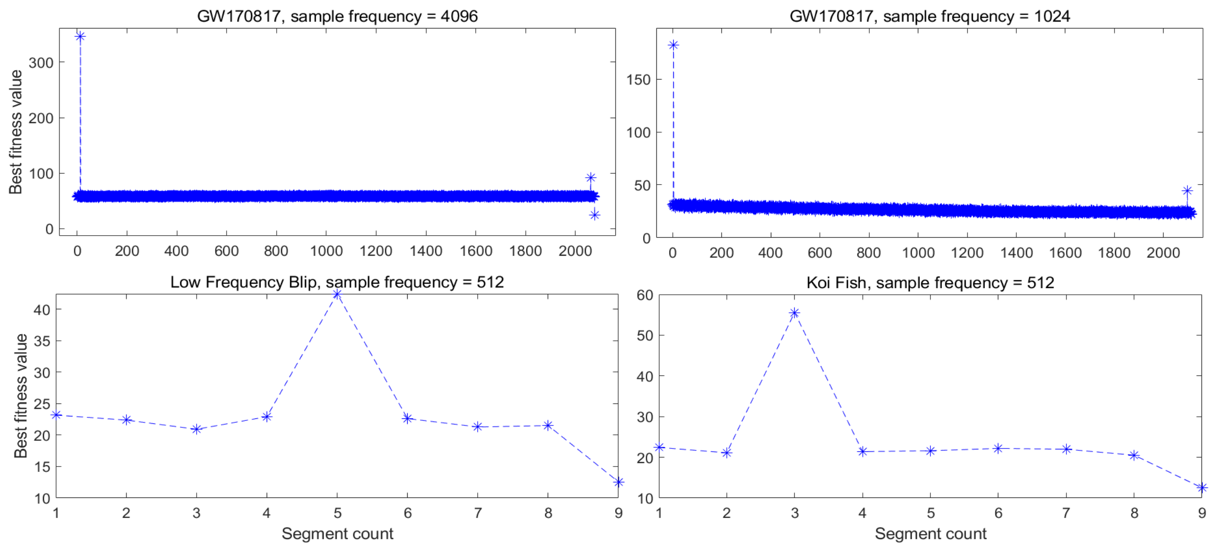


Figure 3.5: Best fitness values of data

the data. As the samples are small, SHAPES fits this part of the data well. Thus, the best fitness value gets small in the last part. Also, the value of sampling frequency plays a very important role in saving time. SHAPES takes time to fit the data and hence, it will take a lot of time if we use the sampling frequency of data as 4096 [fig. 3.5. This problem is solved by downsampling the data. The top right panel of fig. 3.5 has a sampling frequency of 1024 and we still can see it has a higher value in the same places as the best fitness value has a higher value in the top left panel of fig. 3.5. Every high value is associated with a glitch segment. By collecting the segment counts using these high best fitness values in various places, we can detect multiple or all of the glitches that are present in the data. The bottom panels in fig. 3.5 have considerably less segment than the top panels as those figures have been built for only 8 s data. Also, the bottom panels in figure 3.5 have been built using the sampling frequency 512. It takes only 1 minute to estimate and find the glitch segment using the best fitness value. The higher the sampling frequency, the more accurate the position of the segment. But downsampling the data also works fine for identifying glitch positions.

Once the segment count is selected, the segment boundary goes back to the original sampling frequency. Though we selected the segment with glitches with the best fitness value, this method does not decide the segment boundary of the glitches. For determining the segment boundary of

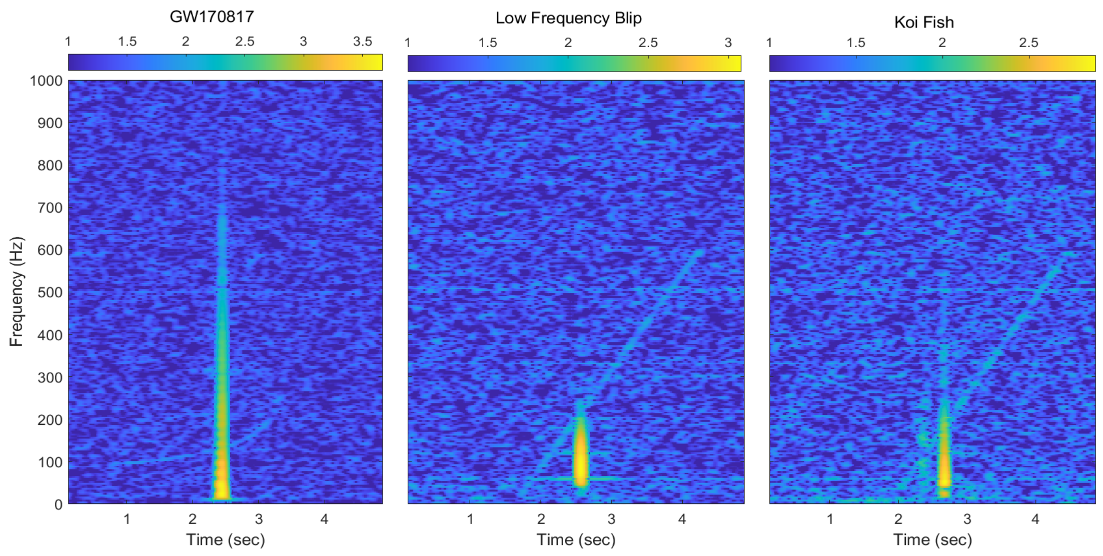


Figure 3.6: Spectrogrom of glitches

the glitch, we use a spectrogram. In figure 3.6, we have added a chirp signal into the second and third panels of the figure. From figure 3.6, we can see that the power of glitch is higher than any other part of the data including the signal. Making all other parts of the data zero except 60% of the maximum power of the spectrogram, we can determine the boundary of the glitch and the time associated with the glitch boundary. This method determines glitch boundaries more accurately than the amplitude threshold method. Therefore, it removes less power around the glitch and keeps more signal parts in the data.

3.4 Measurement of performance

While removing glitches from the data, one of the major questions is how SHAPES, waveshrink, and the combination approach of SHAPES and waveshrink affect the gravitational wave signal or injected signal. For measuring the effect of the glitch removal methods, we consider measuring the signal-to-noise ratio (SNR) of the signal after removing the glitches. For measuring the SNR of the signal in the respective residuals of the respective method, at first, we removed the glitch using the SHAPES, waveshrink, and the combination of SHAPES and waveshrink. From all of the residuals, we choose the best residual (chosen by looking at the spectrogram). Then, we inject the signal into the data around the glitch time. For measuring SNR, either from the glitch-removed data or from the residual of the methods, we use the below equation

$$SNR = \frac{\mathbf{d}(t) \cdot \mathbf{s}(t)}{||\mathbf{s}(t)||} \tag{3.22}$$

where $d(t)$ is representing data, $s(t)$ is representing signal, and $||s(t)||$ is representing norm of the signal. For removing the glitch, aLIGO make the data zeros around the glitch part. The glitch removal method, SHAPES, waveshrink, and the combination approach of SHAPES and waveshrink affect the signal. But the loss of the SNR in the injected signal is far less than the current zeroing procedure that aLIGO use, which we will see in the chapter IV for the respective glitches. The loss percentage has been calculated using the below equation

$$Loss = \frac{SNR(grd) - SNR(method)}{SNR(grd)} \times 100\% \tag{3.23}$$

Here, $SNR(grd)$ is representing the SNR from glitch-removed data, and $SNR(method)$ is representing the SNR from the residuals, which is acquired after removing the glitch using the particular methods. Hence, the $SNR(method)$ can be calculated from $Loss$ by,

$$\begin{aligned} SNR(method) &= SNR(grd) \times (1 - Loss(\%)), \\ &= SNR(grd) \times \left(1 - \frac{Loss}{100}\right) \end{aligned} \tag{3.24}$$

The SNR and $Loss$ calculation shows us the quantitative measurement of the performance of the methods. For measuring the performance qualitatively, the spectrogram and adaptive spectrogram have been used. A spectrogram is a time-frequency domain diagram, which also shows the power difference of various events in the data. For example, glitches have more power than signal and general/Gaussian noise, which can be seen from the spectrogram. An adaptive spectrogram is different than a normal spectrogram. Adaptive spectrogram generates multiple time-frequency diagrams, selects the best part from all of these diagrams, and combines them in one time-frequency diagram Lukin & Todd (2006).

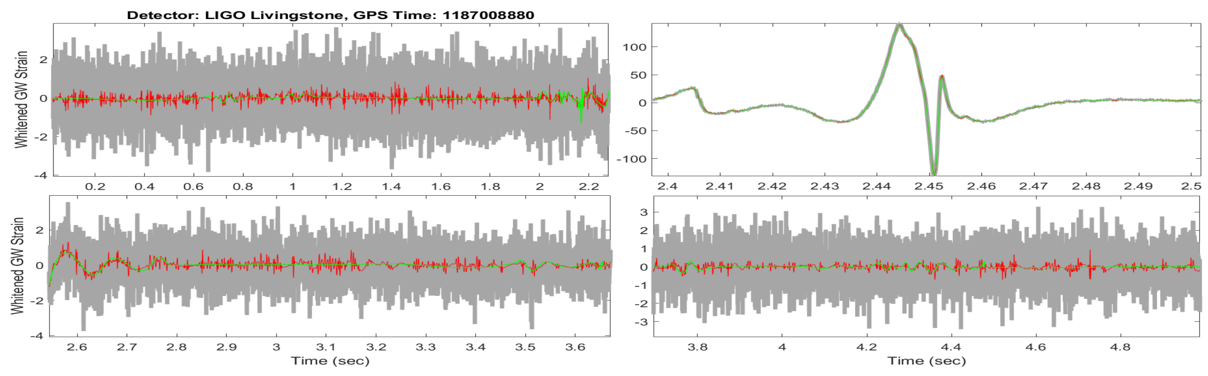
CHAPTER IV

RESULTS AND DISCUSSION

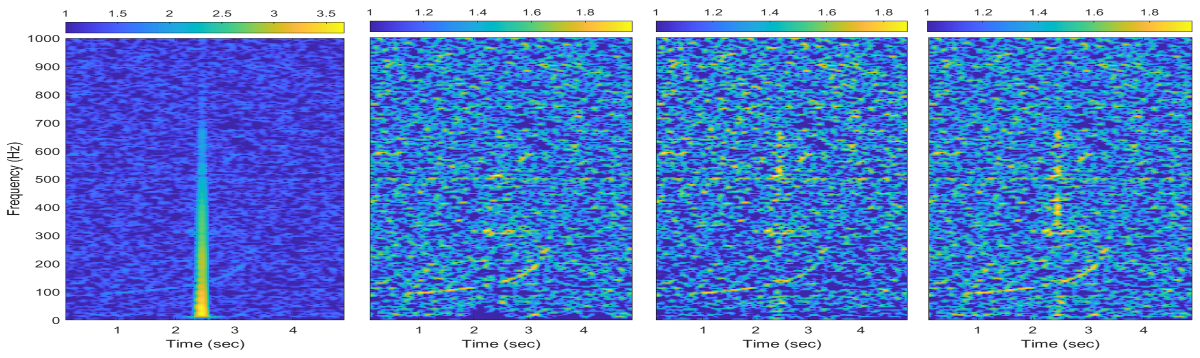
4.1 GW170817

4.1.1 Results using amplitude threshold method

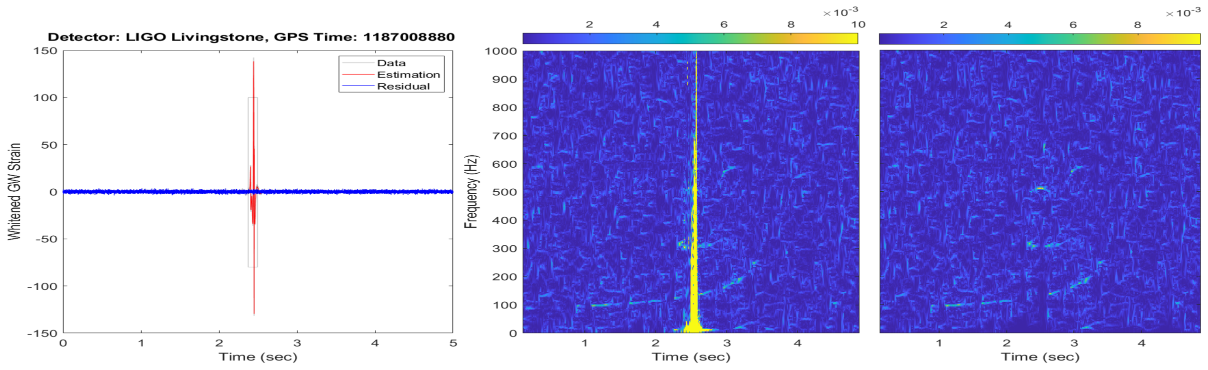
Fig. 4.1 (a) is the comparison of timeseries, from where we can see that at the top left panel, the bottom left panel, and the bottom right panel, the estimation made by SHAPES is smoother than the waveshrink. In the case of SHAPES, the segment length 16384 has been applied for getting the estimation of the data, other than the glitch part. In the glitch part of the data, the segment length applied is 1024. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power. The power difference is visible from the third panel of the general spectrogram figure [figure 4.1 (b)]. In the third panel of figure 4.1 (b), there is less power below 300 Hz, which is caused by the overfitting of the waveshrink. From figure 4.1 (a), the top right panel is the estimation of the glitch by SHAPES and waveshrink. In this case, both systems estimated the glitch pretty well, but SHAPES is following the glitch more closely to the sharp changes in the glitch. That is why we can see from the figure 4.1 (b), that SHAPES is taking out the glitch and more power around the glitch below 50 Hz. Also, we see that there is a trace of the glitch left behind in the third panel of figure 4.1 (b), which is a result of glitch removal by waveshrink estimation. The combination of the estimation of SHAPES and waveshrink failed in this case as we can see from figure 4.1 (b) right panel. Therefore, we can assume that the combination approach of SHAPES and waveshrink will not work well on high frequency glitches. Also, there is a part of a linear noise, which is not removed during data conditioning, is around 300 Hz, which is left behind in all methods.



(a)



(b)



(c)

Figure 4.1: Estimation and removal of the glitch in GW170817 Data using 1024 as a segment length for fitting data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES, black boxes around the glitch is the glitch boundary identified by the amplitude threshold method. Also, the spectrogram in the figure (c) is adaptive spectrogram.

For removing glitches using 1024 segment length, SHAPES have to be applied twice. Using 1024 segment length once, it kept a trace of glitch around 400 Hz as shown in figure 4.2. Now, SHAPES started to keep the trace of glitch from 512 segment length.

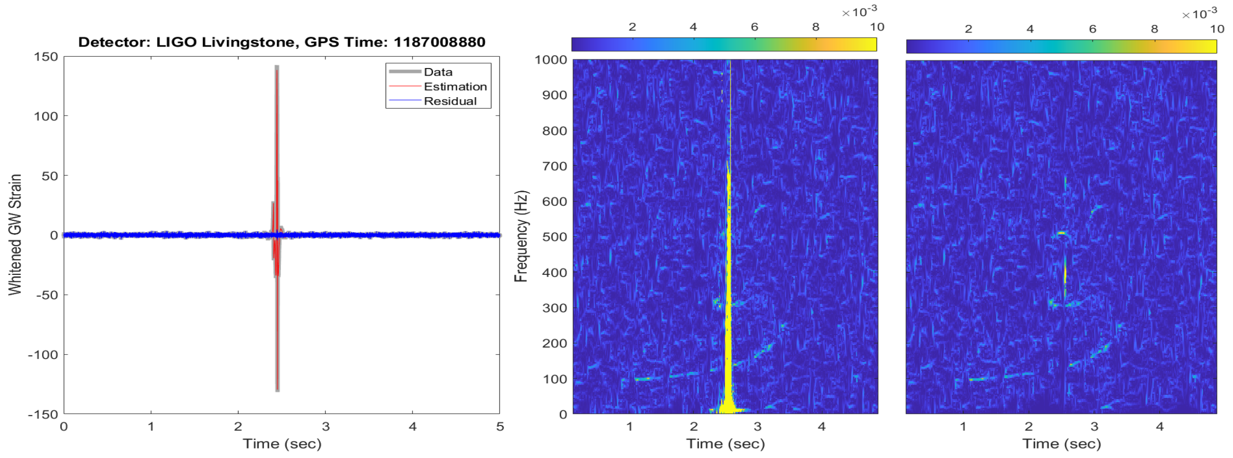


Figure 4.2: Glitch removal with applying 1024 segment length ones on the glitch part of the data

Also, the glitch can be removed using the segment length 2048 at the first run and for the second estimation, 1024 segment length has been used. It kept some more power around the glitch part in fig. 4.3 but it is not that significant change comparing to the figure of 1024 segment length (run = 2).

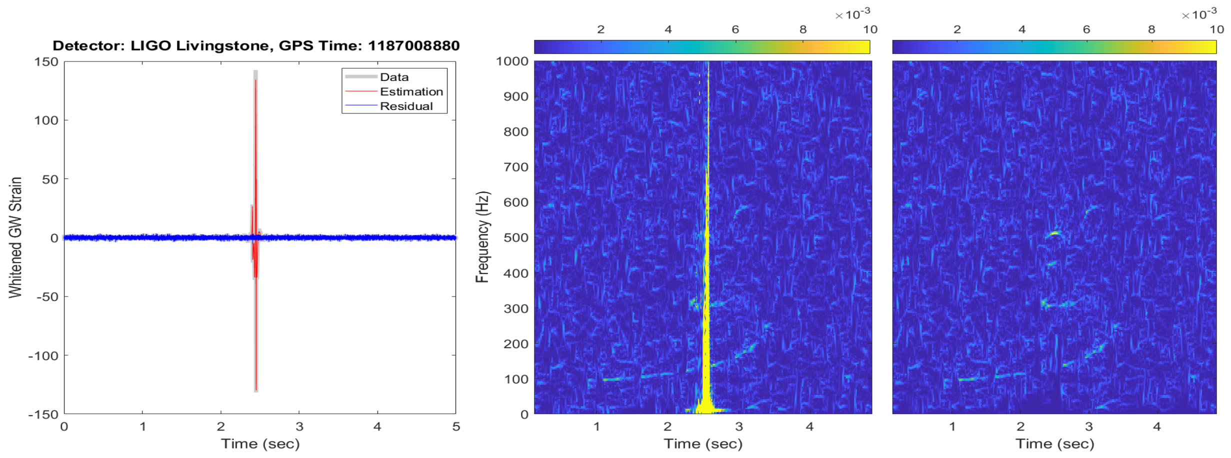
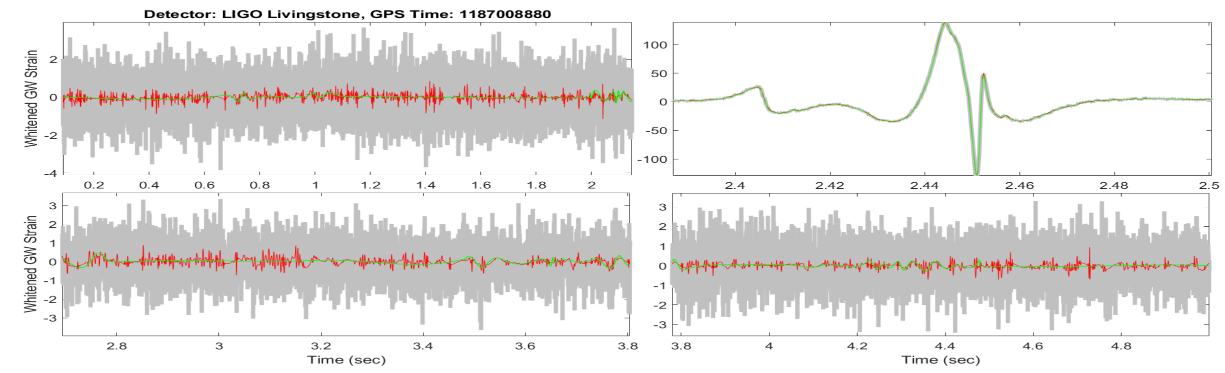
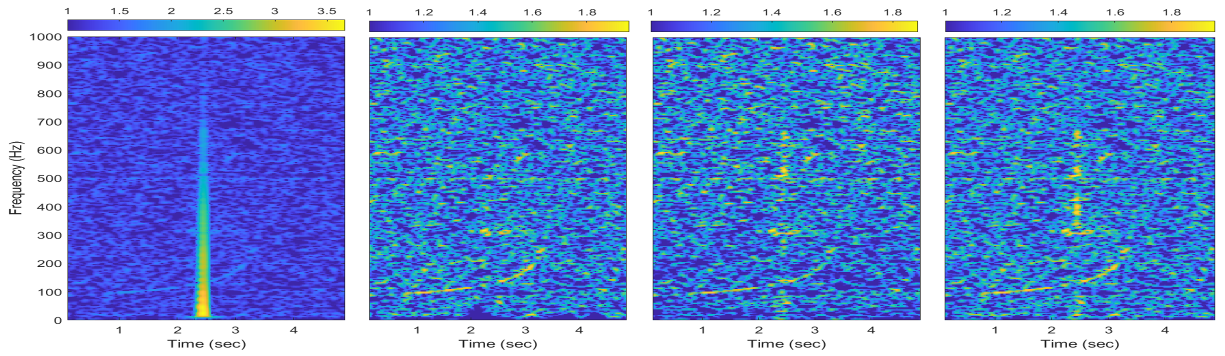


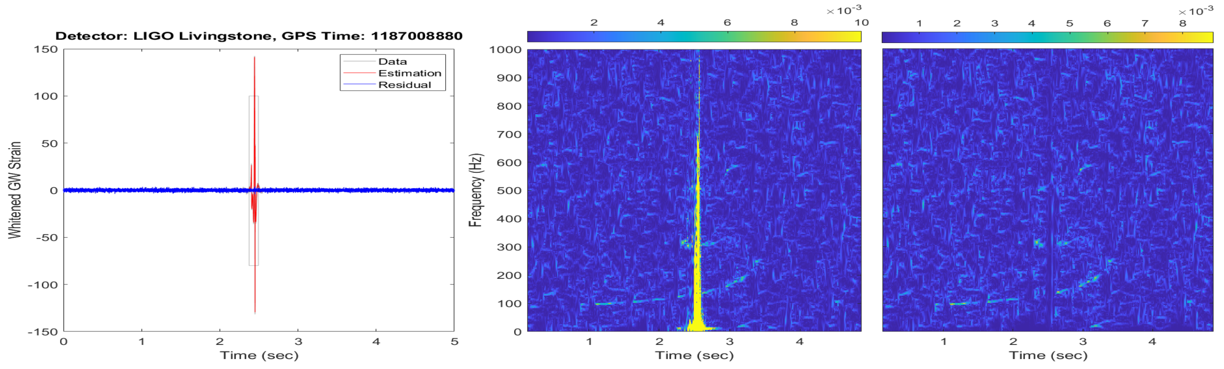
Figure 4.3: Glitch removal of GW170817 using 2048 as a segment length at first run of SHAPES and 1024 as a segment length for the second run



(a)



(b)



(c)

Figure 4.4: Estimation and removal of the glitch in GW170817 Data using 384 as a segment length, on the glitch part of the data, for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES, black boxes around the glitch is the glitch boundary identified by the amplitude threshold method. Also, the spectrogram in the figure (c) is adaptive spectrogram.

If the segment length, 384 = (256+512)/2, is applied to the main glitch part in the case of the application of SHAPES on the full data, the glitch can be removed in the first run [fig. 4.4. Using the threshold technique, we can apply a method that applies SHAPES only on the part of the glitch. For the rest of the data, it does not estimate anything i.e. the estimation is zero for the data, except for the glitch part. In this system, the threshold technique will detect the main glitch part (as before) and select the data around the glitch. SHAPES will only estimate the glitch using the segment length 384, and the rest of the data will get zero estimation. The figure for 384 segment length, for the latter method, is given below. Hence, as long as we apply 384 segment length in the main glitch part, whether the method is SHAPES on full data or the method is SHAPES on the glitch-only part of the data, the glitch will be completely removed in the first run.

The code can remove the glitch using one run up to segment length 410. With a larger segment length value i.e. greater than 410, it keeps a trace of the glitch in the spectrogram. Any segment length below 410 can remove the glitch but it takes out more power around the glitch. Thus, we can use 256 to remove the glitch but it will take out more general power around the glitch. The glitch removal using 410 as segment length is given below [fig. 4.5].

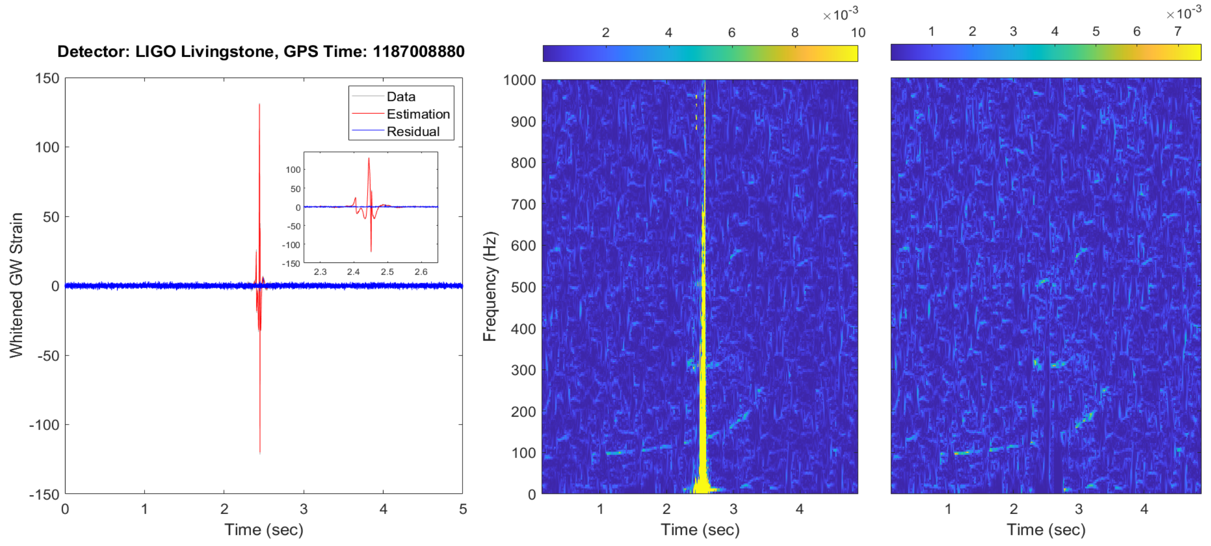


Figure 4.5: Glitch removal of GW170817 using 410 as a segement length, on the glitch part of the data, for fitting the data

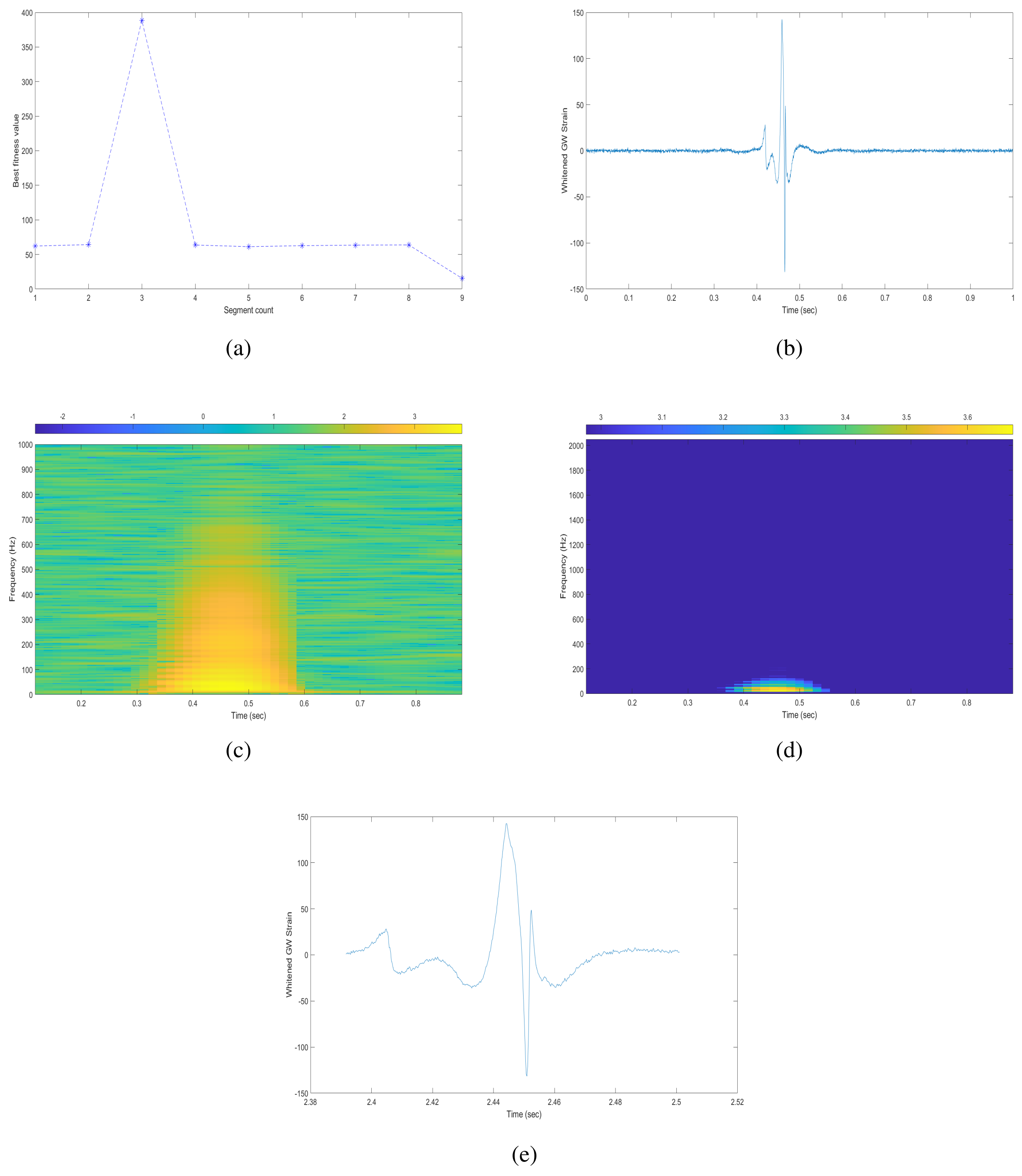


Figure 4.6: Identifying the glitch in GW170817 data. (a) Best fitness value of the data, (b) Data segment containing glitch; selected by using best fitness value, (c) Spectrogram of the selected data segment, (d) Glitch boundary selected by spectrogram threshold, (e) Timeseries of the glitch boundary

4.1.2 Results using bestfitness-spectrogram threshold method

Figure 4.6 (a) illustrating the values of best fitness and segment count. By selecting the higher value than a threshold of best fitness value, we select the segment number. Using this segment number, the time domain data of the glitch is selected as shown in figure 4.6 (b). Then a spectrogram is formed in figure 4.6 (c). From figure 4.6 (c), we can see that glitch has higher power than any other part of the data. Using the spectrogram threshold as mentioned in 3.3.2, we get figure 4.6 (d). In figure 4.6 (d), only glitch has been kept. Using the glitch boundary from figure 4.6 (d), we get the start and end of the glitch i.e. the glitch boundary in time domain figure 4.6 (e). In this way, the determination of glitch boundary is more accurate than the amplitude threshold method.

Fig. 4.8 (a) is the comparison of timeseries, there is not much of a change from the timeseries of fig. 4.4 (a). In the case of SHAPES, the segment length 16384 has been applied to the data, other than the glitch part. In the glitch part of the data, the segment length is 384. The difference it made is clearly visible in spectrogram [figure 4.6 (b) and figure 4.6 (c)]. In the figure 4.6 (b) second panel or figure 4.6 (c) third panel, there is a part of the signal is present in 2.2 s, which is not present in fig. 4.4 (b) second panel or fig. 4.4 (c) third panel. Other than that, we can say that waveshrink is still removing more power below 300 Hz than SHAPES as in fig. 4.4 (b). Also, we see that there is a trace of the glitch left behind in the third panel of figure 4.8 (b) as before, which is a result of glitch removal by waveshrink estimation. The combination of the estimation of SHAPES and waveshrink failed, as before, in this case as we can see from figure 4.8 (b) right panel. In this method as well, the glitch can be removed if we use 410 as a segment length for fitting the data [figure 4.7]. But if we go higher than 410, it starts to keep a trace of glitches left behind in the spectrogram.

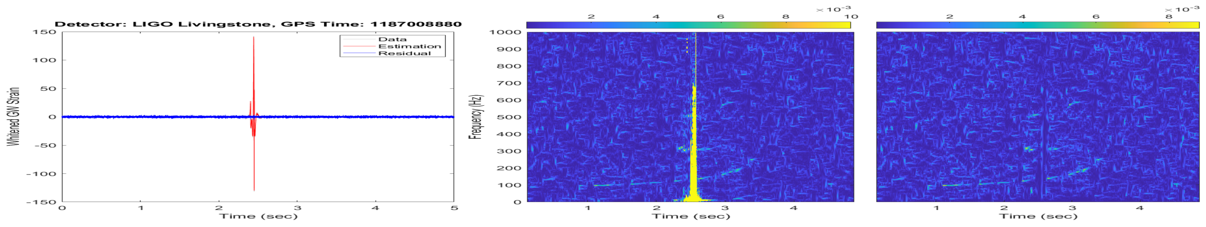
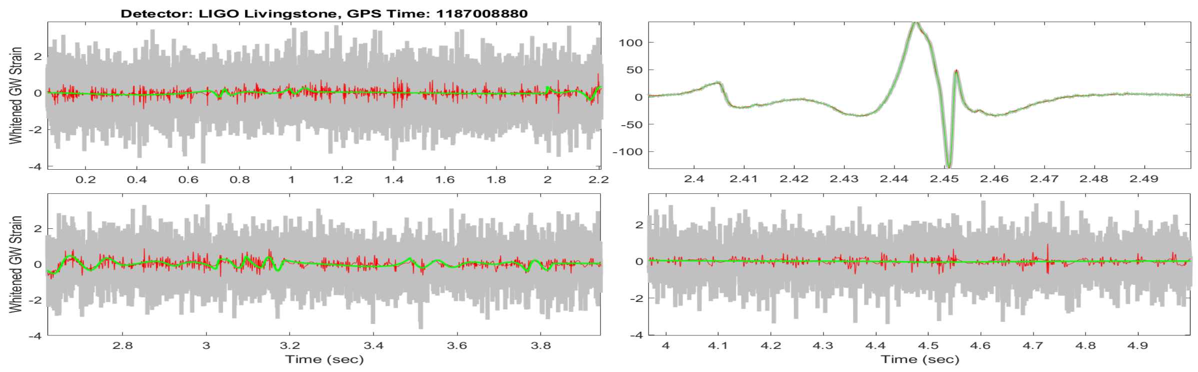
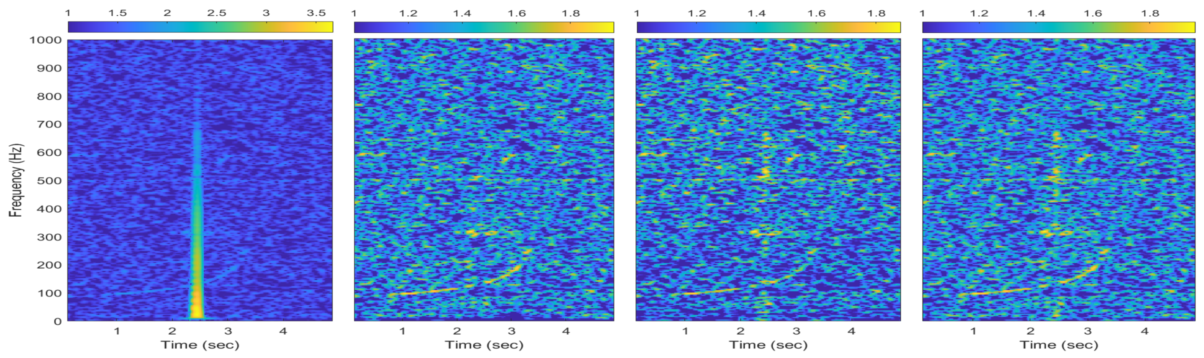


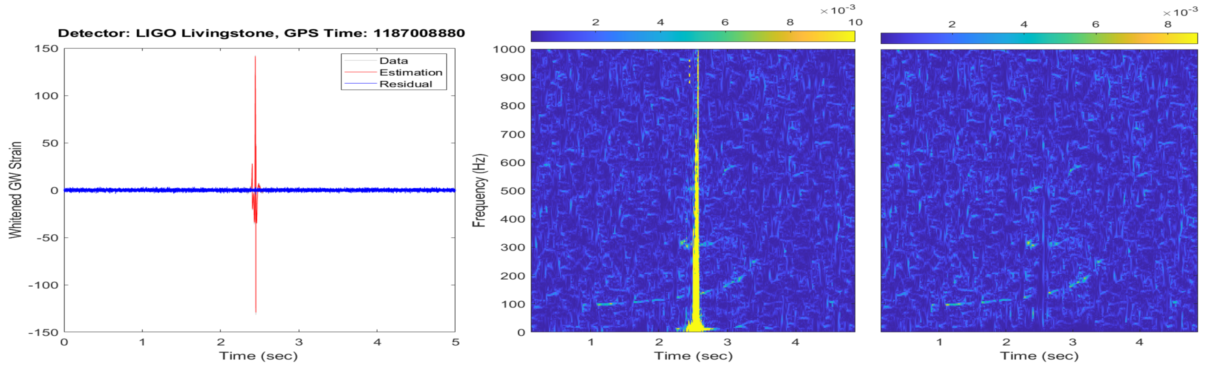
Figure 4.7: Glitch removal of GW170817 using 410 as a segement length in the glitch part of the data for fitting the data



(a)



(b)



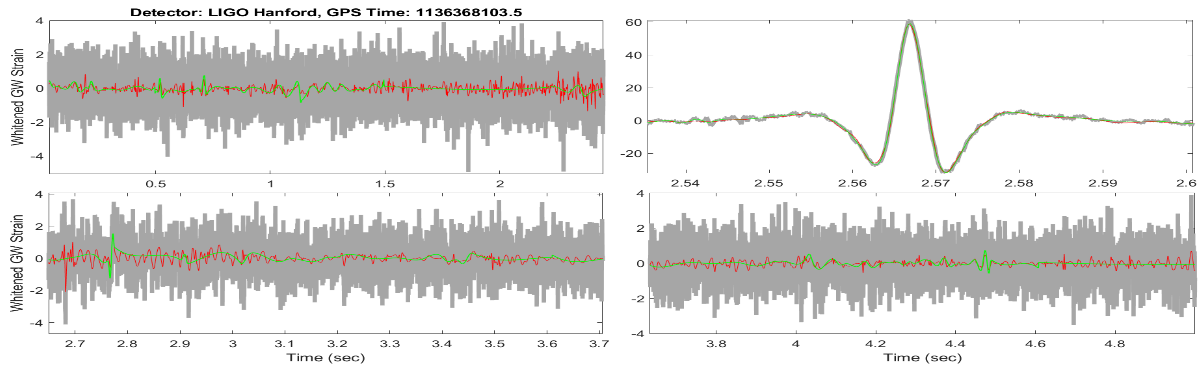
(c)

Figure 4.8: Estimation and removal of the glitch in GW170817 Data using 384 as a segment length, on the glitch part of the data, for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES. Also, the spectrogram in the figure (c) is adaptive spectrogram.

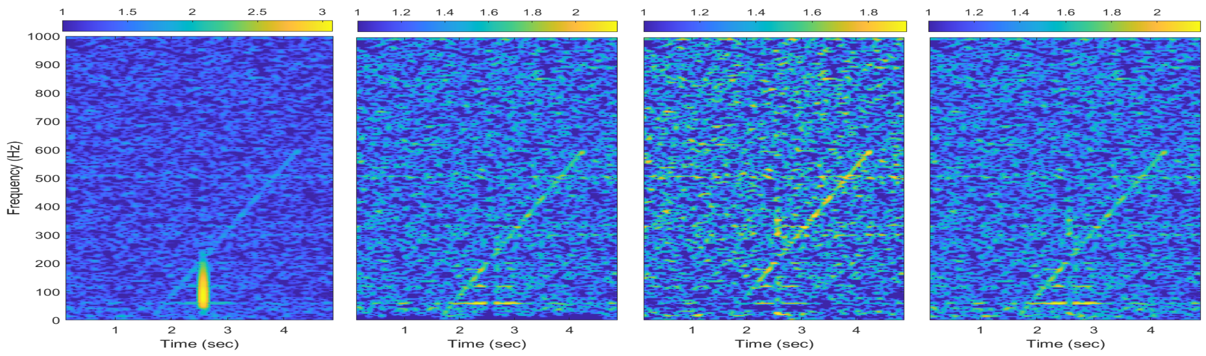
4.2 Low Frequency Blip

4.2.1 Results using amplitude threshold method

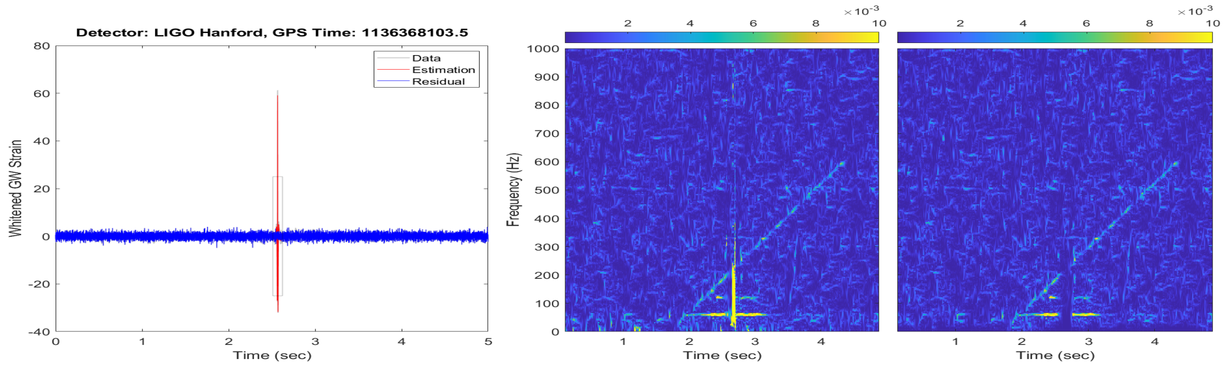
Figure 4.9 (a) is the comparison of timeseries, from where we can see that at the top left panel, the bottom left panel, and the bottom right panel, the estimation made by SHAPES is smoother than the waveshrink. In the case of SHAPES, the segment length 16384 has been applied to the rest of the data, other than the glitch part. In the glitch part of the data, the segment length is 2048. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power. The power difference is visible from the general spectrogram figure [figure 4.9 (b)]. In the third panel of the figure 4.9 (b), there is less power below 300 Hz (as before), which is caused by the overfitting of the waveshrink. From figure 4.9 (a), the top right panel is the estimation of the glitch by SHAPES and waveshrink. In this case, both systems estimated the glitch pretty well, but in some cases, SHAPES is following data closely, which is evident from the maximum and minimum peak of the glitch [figure 4.9 (a)]. That is why we can see from figure 4.9 (b), that SHAPES is taking out the glitch and more power around the glitch below 50 Hz as before. Also, we see that there is a trace of the glitch left behind in the third panel of figure 4.9 (b), which can be clearly seen using an adaptive spectrogram. But the trace is very negligible for this case. Amongst SHAPES and waveshrink, SHAPES performs better for preserving the signal. The combination of the estimation of SHAPES and waveshrink, figure 4.9 (b) right panel, did better than the Blip glitch in GW170817. From this result, we can say that it is an indication that the combination of SHAPES and waveshrink will work better in low-frequency glitches, which will be shown in Scattering glitches. A linear chirp signal with SNR 25 is injected into the data around the glitch. Using equation 3.22 and equation 3.23, we get SNR values and loss percentage for low frequency blip as written in Table 4.1. From the loss percentage, we can say that SHAPES and waveshrink performs way better than the current aLIGO method i.e. gating method. Amongst the SHAPES and waveshrink, we can see that SHAPES performing better than waveshrink.



(a)

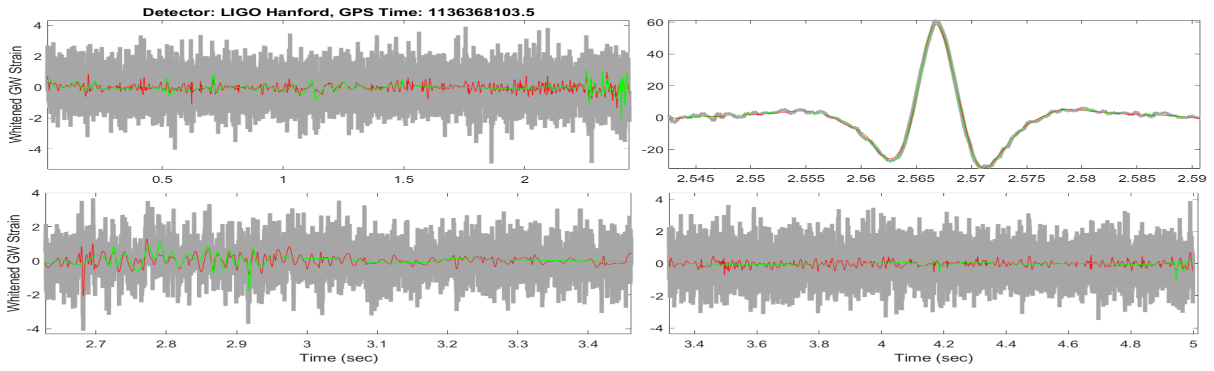


(b)

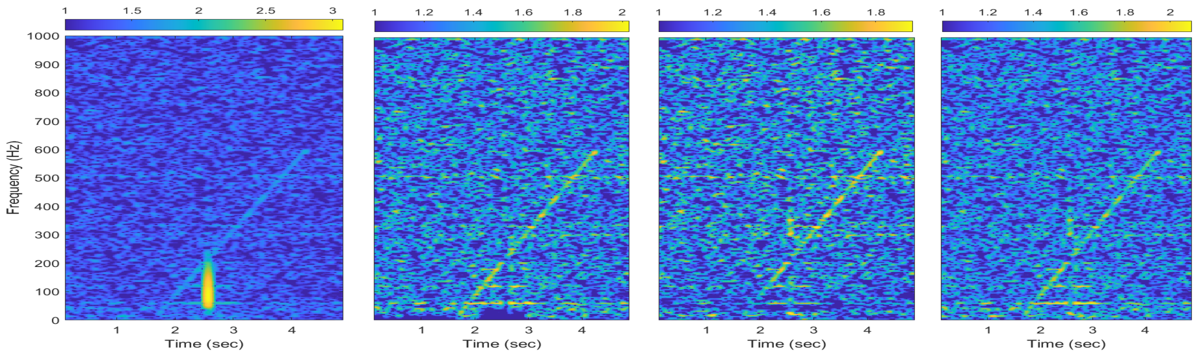


(c)

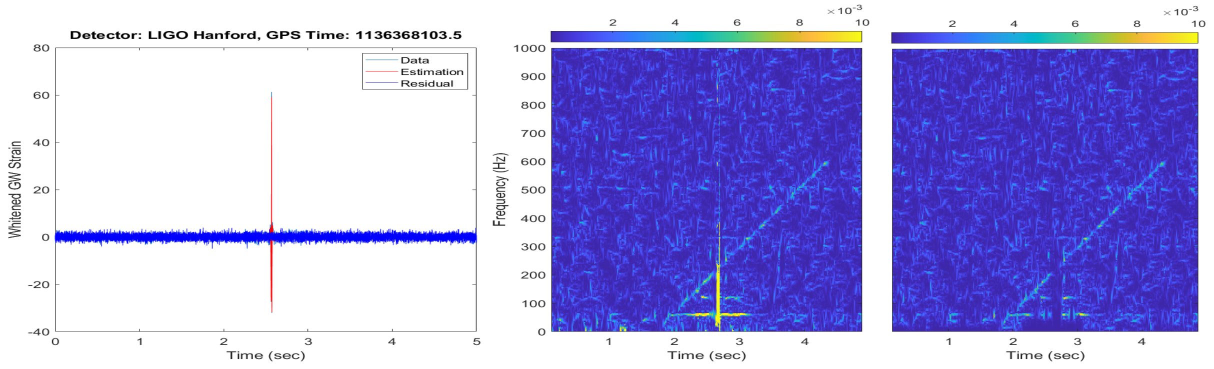
Figure 4.9: Estimation and removal of the low frequency blip glitch using 2048 as a segment length on the glitch part for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES, black boxes around the glitch is the glitch boundary identified by the amplitude threshold method. Also, the spectrogram in the figure (c) is adaptive spectrogram.



(a)



(b)



(c)

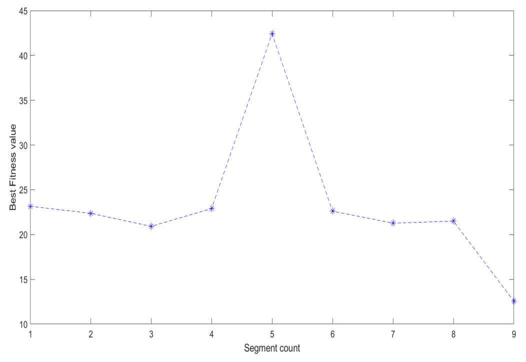
Figure 4.10: Estimation and removal of the low frequency blip glitch using 2048 as a segment length on the glitch and 4096 as a segment length on the linear noise part for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES. Also, the spectrogram in the figure (c) is adaptive spectrogram.

We assumed that as long as the segment length is bigger than the segment boundary of the glitch for the low-frequency glitches, it can remove the glitch successfully. Using this assumption, we applied SHAPES onto the glitch with a code, where the code will decide the length of the segment boundary and select a bigger segment length for the estimation in the glitch part. In the case of the rest of the data, it will apply 16384 Hz. And the results are exactly same as shown in the figures above. This assumption is also successful in the case of the Koi Fish glitch, which will be discussed later.

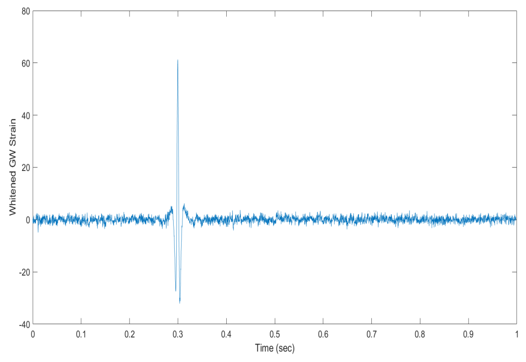
We saw from the low-frequency blip spectrogram, which is explained in the above figure [fig. 4.9], has a linear noise at the bottom part of the glitch. This is a part of a linear noise, which is not removed during whitening process. In these figures, we consider this part of the noise as a secondary glitch and estimated it using SHAPES. For this secondary glitch, we have used 4096 as the segment length. The estimation is changed from the previous fig. 4.10 (a) of low-frequency blip after 2s in the top left panel and in between 2.7s and 2.9s in the bottom left panel. Also, the effect of the estimation can be seen in fig. 4.10 (c), where the linear noise is removed from the right panel of the fig. 4.10 (c).

4.2.2 Results using bestfitness-spectrogram threshold method

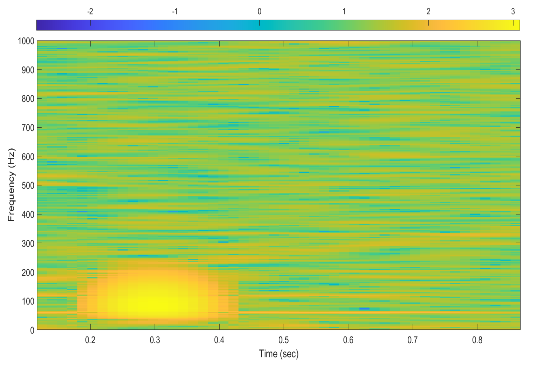
Figure 4.11 (a) illustrating the values of best fitness and segment count. Comparing figure 4.11 (a) with figure 4.6 (a), we can see that the best fitness value differs significantly. From that, we can assume that the change in best fitness values depends on the frequency. As GW170817 has a high-frequency glitch, the best fitness value for the glitch is also much higher than the low-frequency blip glitch. The segment number is selected by using the higher best fitness value in fig. 4.11 (a) than the threshold of best fitness value. Using this segment number, the time domain data of the glitch is selected as shown in figure 4.11 (b). By making a spectrogram for the glitch segment [figure 4.11 (c)] and applying spectrogram threshold as mentioned in 3.3.2, we get figure 4.11 (d) which has only the glitch part. Using the glitch boundary from figure 4.11 (d), we get the start and end of the glitch i.e. the glitch boundary in time domain figure 4.11 (e). As in sub-section 4.1.2, the determination of glitch boundary is more accurate than the amplitude threshold method.



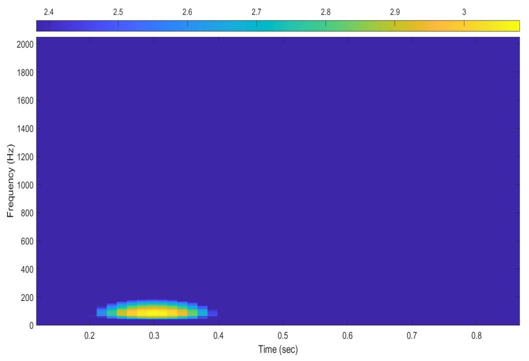
(a)



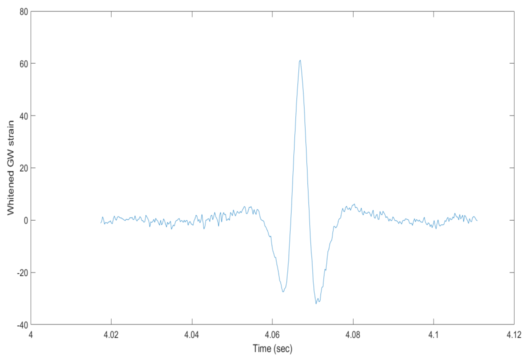
(b)



(c)

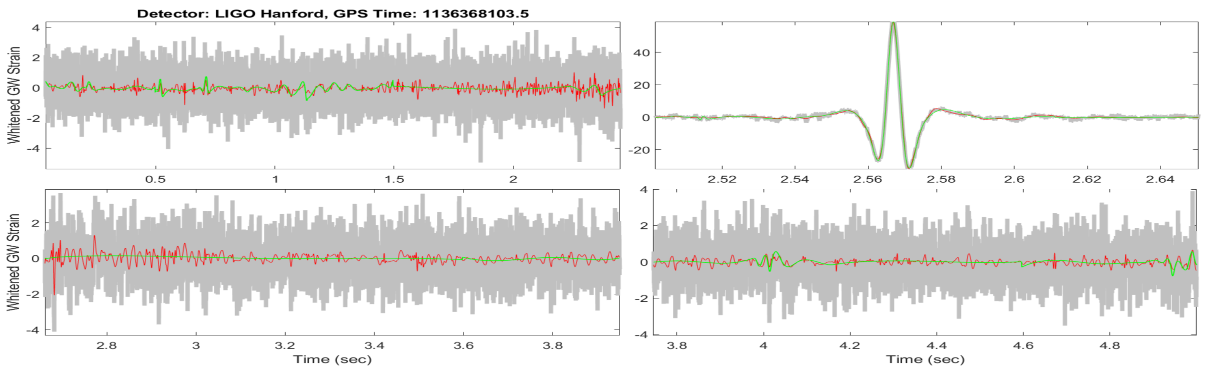


(d)

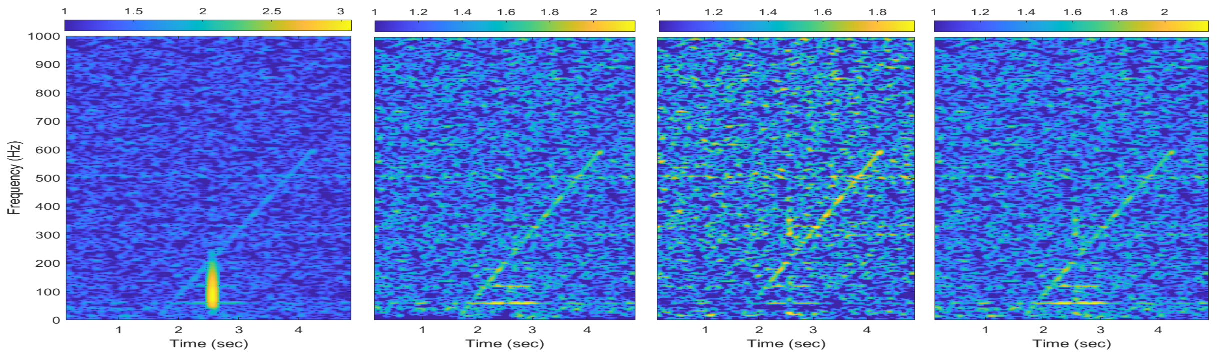


(e)

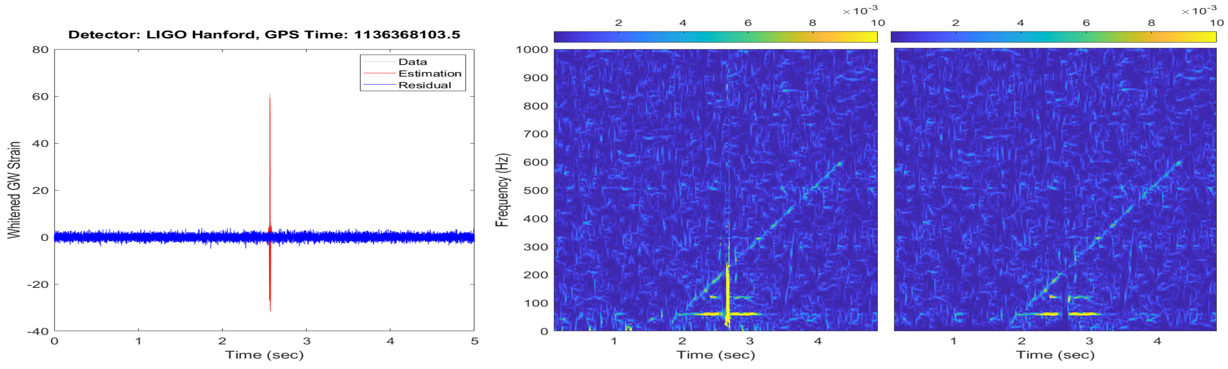
Figure 4.11: Identifying the low frequency blip glitch in the data. (a) Best fitness value of the data, (b) Data segment containing glitch; selected by using best fitness value, (c) Spectrogram of the selected data segment, (d) Glitch boundary selected by spectrogram threshold, (e) Timeseries of the glitch boundary



(a)



(b)



(c)

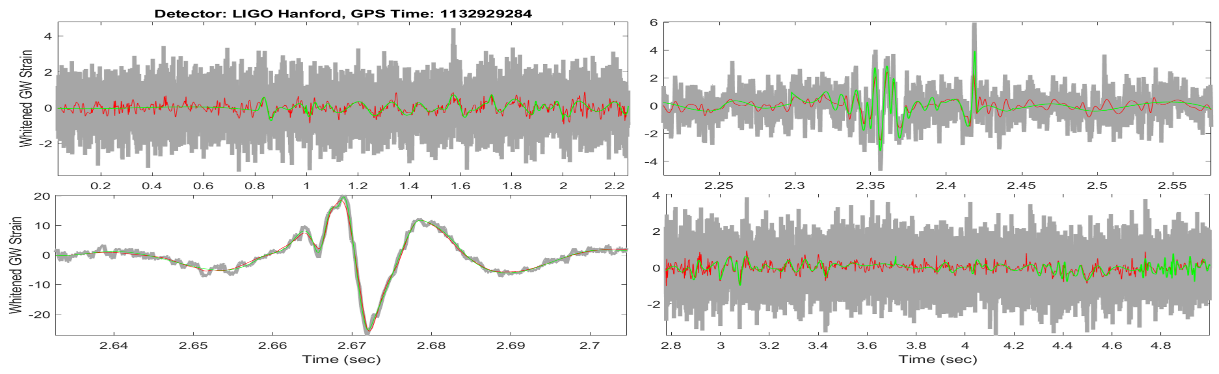
Figure 4.12: Estimation and removal of the low frequency blip glitch using 2048 as a segment length on the glitch part of the data for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES. Also, the spectrogram in the figure (c) is adaptive spectrogram.

In the case of SHAPES, the segment length 16384 has been applied to the data, other than the glitch part. In the glitch part of the data, the segment length is 2048. Fig. 4.12 (a) is the comparison of timeseries, there is a noticeable change in the top right panel and bottom left panel of the time series from the top right panel and bottom left panel timeseries of fig. 4.12 (a) around 2.58 s and in between 2.7 s and 3.0 s, respectively. The difference it made is clearly visible in spectrogram [figure 4.12 (b) and figure 4.11 (c)], where we can see more power in between 2 s and 3 s in the second panel of figure 4.12 (b) than the second panel of figure 4.9 (b). Other than that, we can say that waveshrink is still removing more power below 300 Hz than SHAPES as in the third panel of fig. 4.9 (b). Also, we see that there is a trace of the glitch left behind in the third panel of figure 4.12 (b) as before, which is a result of glitch removal by waveshrink estimation. The combination of the estimation of SHAPES and waveshrink failed, as before, but did better than figure 4.8 (b) right panel as before. A linear chirp signal with SNR 25 is injected exactly in the same position as for the amplitude threshold method of the low-frequency blip. As it removed less power between 2 s and 3 s than the amplitude threshold method, it kept more signal in that time than the amplitude threshold method. Therefore, the loss percentage is also decreased as we can see from table 4.2. Thus, we can say that the best fitness-spectrum threshold is performing better than the amplitude threshold method.

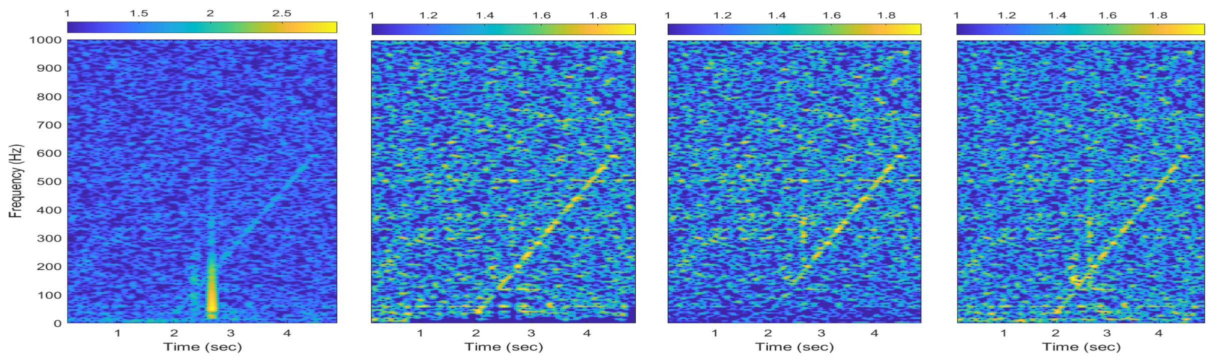
4.3 Koi Fish

4.3.1 Results using amplitude threshold method

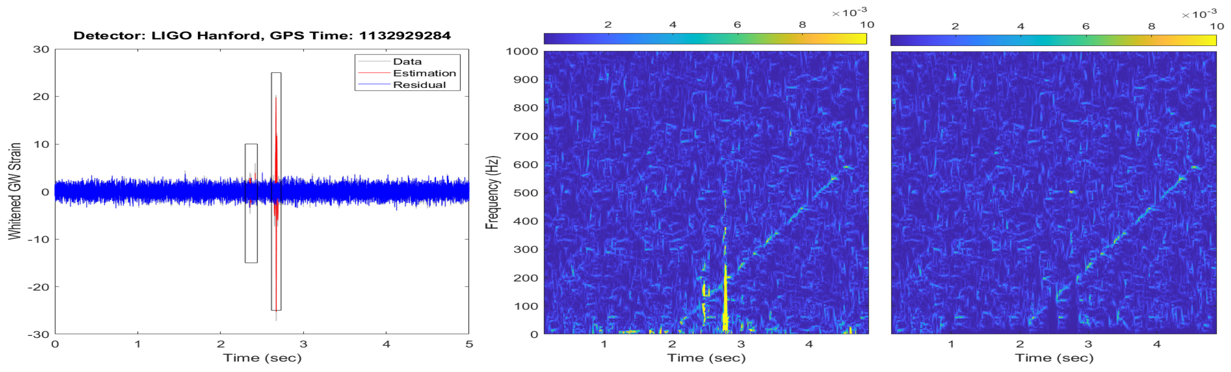
The assumption that we made in the Low-Frequency Blip section works very well in the case of the Koi Fish glitch. Fig. 4.13 (a) is the comparison of timeseries, from where we can see that at the top left panel, and the bottom right panel, the estimation made by SHAPES is smoother than the waveshrink. In the case of SHAPES, the segment length 16384 has been applied to the rest of the data, other than the main glitch (koi Fish) and the secondary glitch (a small glitch near to koi fish glitch) part of the data. In the secondary glitch part (top right panel), a 4096 segment length is applied. In the glitch part of the data, the segment length is 2048. As one can see from the



(a)



(b)



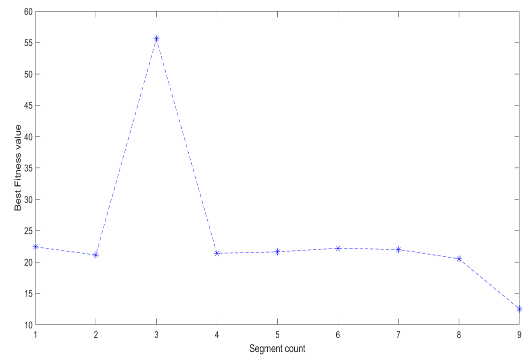
(c)

Figure 4.13: Estimation and removal of the Koi Fish glitch using 2048 as a segment length on the main glitch and 4096 as a segment length for secondary noise for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES, black boxes around the glitch is the glitch boundary identified by the amplitude threshold method. Also, the spectrogram in the figure (c) is adaptive spectrogram.

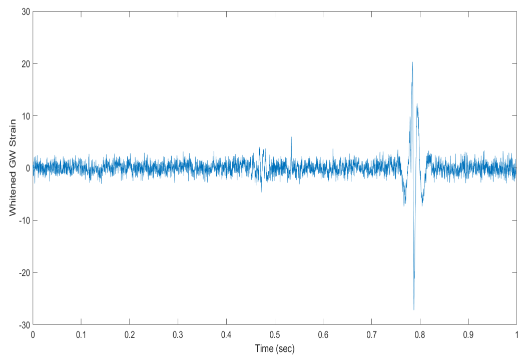
fig. 4.13 (b), the glitch is completely removed from the second panel of the spectrogram, which is a spectrogram for glitch removal by SHAPES. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power. The power difference is visible from the general spectrogram figure [Fig. 4.13 (b)]. In the third panel of the fig. 4.13 (b), there is less power below 300 Hz (as before), which is caused by the overfitting of the waveshrink. From fig. 4.13 (a), the bottom left panel is the estimation of the glitch by SHAPES and waveshrink. In this case, both systems estimated the glitch pretty well, but in some cases, SHAPES is following data closely, which is evident from the maximum peak of the glitch around 2.67s [Fig. 4.13 (a)]. That is why we can see from the fig. 4.13 (a), that SHAPES is taking out the glitch and more power around the glitch. Also, we see that there is a clear trace of the glitch left behind in the third panel of fig. 4.13 (b), which can be seen between 300-400 Hz. The trace is visible in this case. It is more visible in the adaptive spectrogram. Amongst SHAPES and waveshrink, SHAPES performs better for preserving the signal. The combination of the estimation of SHAPES and waveshrink, fig. 4.13 (b) right panel, did not perform well in this case. From this result, we can confirm that the combination of SHAPES and waveshrink will work better in low-frequency glitches. A linear chirp signal with SNR 25 is injected into the data around the glitch i.e. around 2 s. Using equation 3.22 and equation 3.23, we get SNR values and loss percentage for koi fish glitch as written in Table 4.1. From the loss percentage, we can say that SHAPES and gating method performs way better than the wavelet base method i.e. waveshrink. Amongst the SHAPES and gating methods, we can see that SHAPES performing better than gating.

4.3.2 Results using bestfitness-spectrogram threshold method

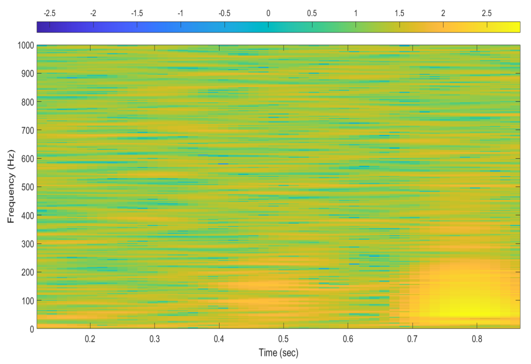
Figure 4.14 (a) illustrating the values of best fitness and segment count. Comparing figure 4.11 (a) with figure 4.14 (a), we can see that the best fitness value differs very slightly. As koi fish has a slightly high-frequency glitch, the best fitness value for the glitch is also slightly higher than the low-frequency blip glitch. Therefore, we can confirm that the best fitness value depends on the frequency of the glitch. Following the same procedure as mentioned in 4.1.2 and 4.2.2, we get



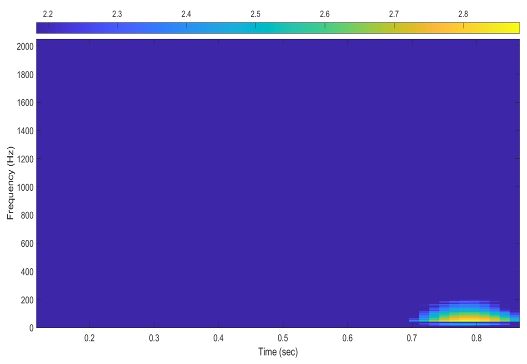
(a)



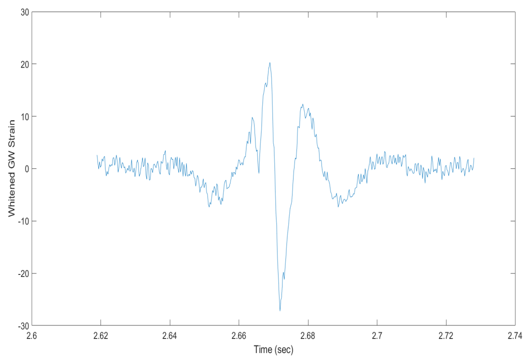
(b)



(c)

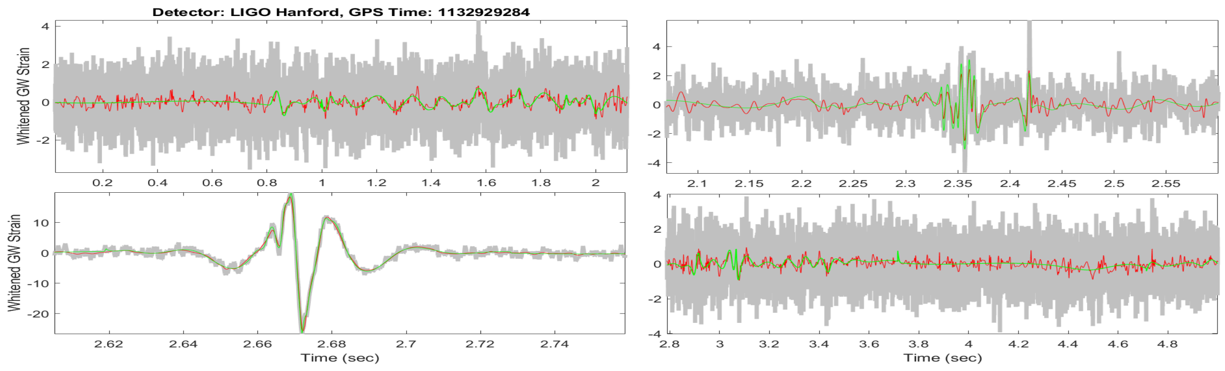


(d)

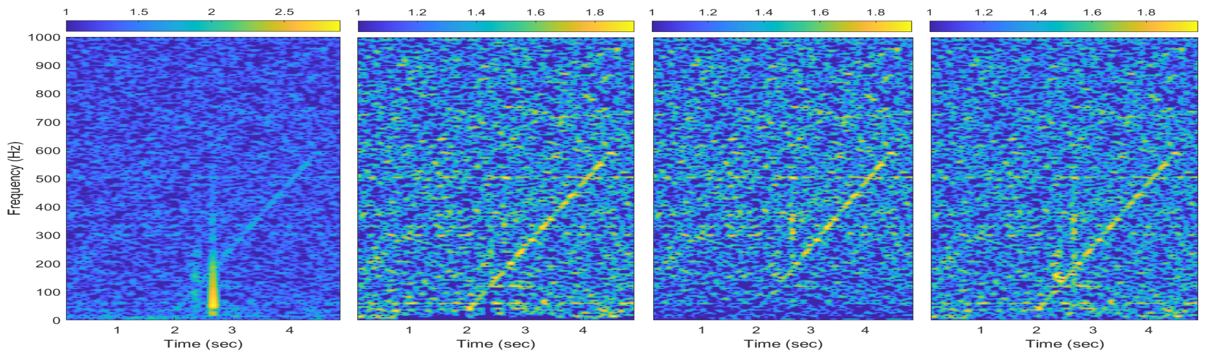


(e)

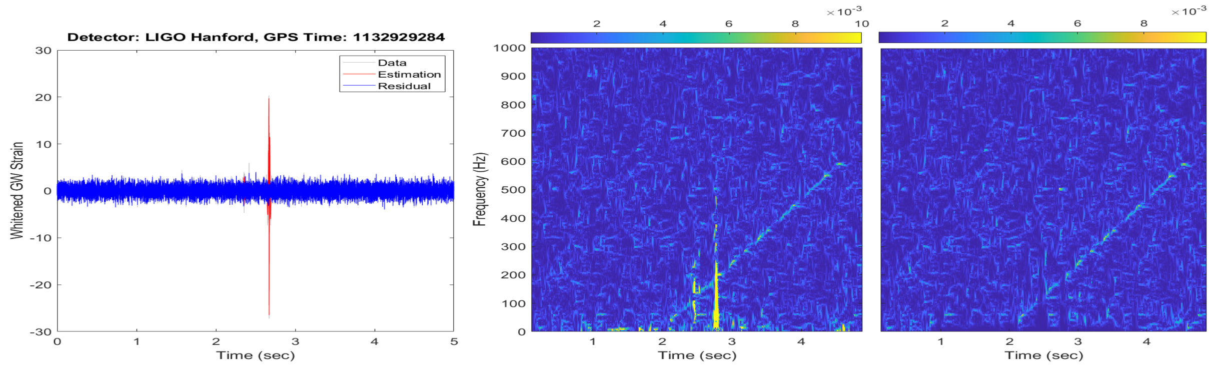
Figure 4.14: Identifying the koi fish glitch in the data. (a) Best fitness value of the data, (b) Data segment containing glitch; selected by using best fitness value, (c) Spectrogram of the selected data segment, (d) Glitch boundary selected by spectrogram threshold, (e) Timeseries of the glitch boundary



(a)



(b)



(c)

Figure 4.15: Estimation and removal of the Koi Fish glitch using 2048 as a segment length on the main glitch and 4096 as a segment length on the secondary noise for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES. Also, the spectrogram in the figure (c) is adaptive spectrogram.

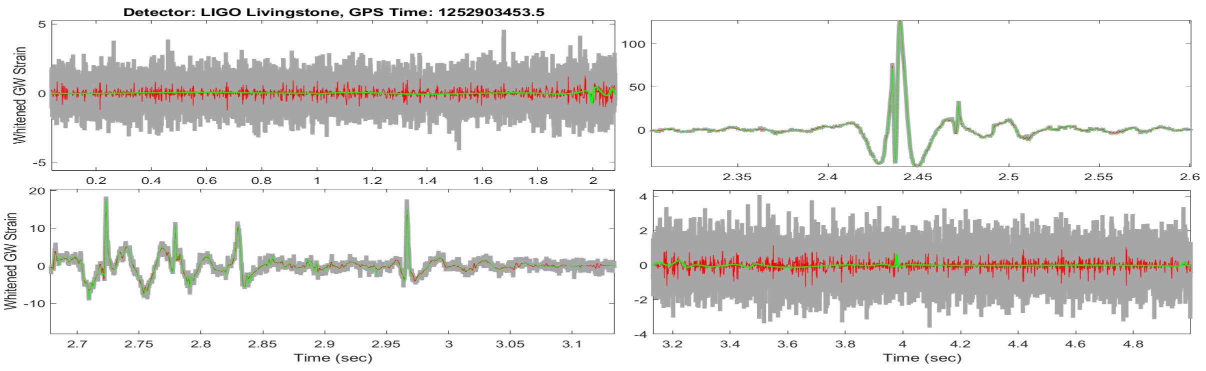
figure 4.14 (b), figure 4.14 (c) and figure 4.14 (d), which has only the glitch part. Using the glitch boundary from figure 4.14 (d), we get the start and end of the glitch i.e. the glitch boundary in time domain figure 4.14 (e). The determination of glitch boundary is more accurate than the amplitude threshold method as before.

In the case of SHAPES, the segment length 16384 has been applied to the data, other than the glitch part. In the glitch part of the data, the segment length is 2048. Fig. 4.15 (a) is the comparison of timeseries, there is a noticeable change in the top right panel and bottom left panel of the time series from the bottom left panel and bottom right panel timeseries of fig. 4.15 (a) in between 2.66 s and 2.68 s, and in between 3.60 s and 5.0 s, respectively. In the bottom right panel, we can see that the estimation is smoother than the bottom right panel of figure 4.13 (a). The difference it made is clearly visible in spectrogram [figure 4.15 (b) and figure 4.15 (c)], where we can see more power in between 2 s and 5 s in the second panel of figure 4.15 (b) or the right panel of figure 4.15 (c) than the second panel of figure 4.13 (b) or the right panel of figure 4.13 (c). The estimation of waveshrink did not change i.e. it is still removing more power below 300 Hz than SHAPES as in the third panel of fig. 4.13 (b). Also, we see that there is a trace of the glitch left behind in the third panel of figure 4.15 (b) as before, which is a result of glitch removal by waveshrink estimation. The combination of the estimation of SHAPES and waveshrink failed, as before, but did better than figure 4.8 (b) right panel as before. A linear chirp signal with SNR 25 is injected exactly in the same position as for the amplitude threshold method of the koi fish. As it removed less power between 2 s and 5 s than the amplitude threshold method, it kept more signal during that time than the amplitude threshold method. Therefore, the loss percentage is also decreased as we can see from table 4.2.

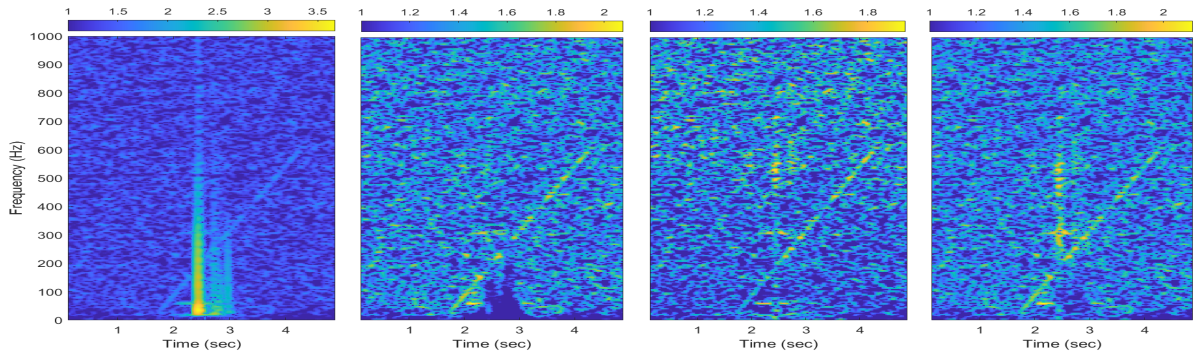
4.4 Tomte

4.4.1 Results using amplitude threshold method

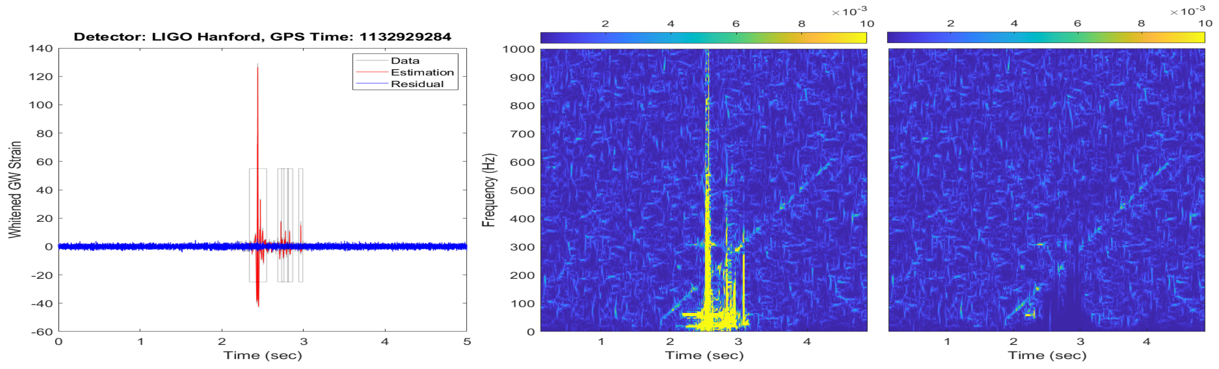
The assumption that we made in the low-frequency blip, which works on koi fish, also works on the tomte glitch part for the small frequency glitches between 2.6s and 3.1s in the bottom left panel of fig. 4.16 (a). But this assumption does not work with a high-frequency glitch which is



(a)



(b)



(c)

Figure 4.16: Estimation and removal of the Tomte glitch using 1024 as a segment length on the glitch part and 16384 on the rest of the data for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of SHAPES, red = estimation of waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using SHAPES, black boxes around the data is the glitch boundary identified by the amplitude threshold method. Also, the spectrogram in the figure (c) is adaptive spectrogram.

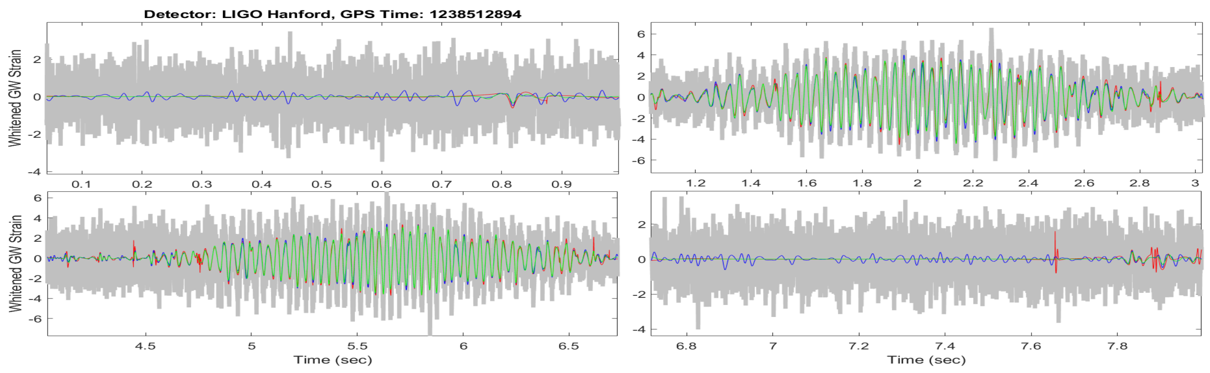
between 2.35 s and 2.55s in the top right panel of the fig. 4.16 (a). Therefore, we can say that for the high-frequency glitch part we need to use shorter segment lengths, which is also evident from the case of the loud blip glitch in GW170817 [section 4.1].

We used 1024 segment lengths in the glitch part and 16384 segment lengths for the rest of the data to remove the tomte glitch. Fig. 4.16 (a) is the comparison of timeseries, from where we can see that at the top left panel, and the bottom right panel, the estimation made by SHAPES is smoother than the waveshrink. In the case of SHAPES, the segment length 16384 has been applied to the rest of the data, other than the glitch part. In the glitch part of the data, the segment length is 1024. As one can see from fig. 4.16 (b), the glitch is completely removed from the second left panel of the spectrogram, which is a spectrogram for glitch removal by SHAPES. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power. The power difference is visible from the general spectrogram figure [Fig. 4.16 (b)]. In the third panel of fig. 4.16 (b), there is less power below 400 Hz (as before), which is caused by the overfitting of the waveshrink. From fig. 4.16 (a), the top right panel and the bottom left panel are the estimations of the glitch by SHAPES and waveshrink. In the case of the high-frequency glitch part at the top right panel of fig. 4.16 (a), w can see the estimation and waveshrink overlapped on each other. Moreover, both systems estimated the glitch pretty well, but in some cases, SHAPES is following data closely, which is evident from the peaks of the glitch in the bottom left panel [Fig. 4.16 (a)]. That is why we can see from fig. 4.16 (b), that SHAPES is taking out the glitch and more power around the glitch. Also, we see that there is a clear trace of the glitch left behind in the third panel of fig. 4.16 (b), which can be seen at around 500 Hz. The trace is visible in this case. It is more visible in the adaptive spectrogram. The combination of the estimation of SHAPES and waveshrink, fig. 4.16 (b) right panel, failed to remove the glitch in this case. From the result in the right panel of the fig. 4.16 (b), we can confirm that the combination of SHAPES and waveshrink will not work on the high-frequency glitch. A linear chirp signal with SNR 25 is injected into the data around the glitch i.e. around 1.9 s. Using equation 3.22 and equation 3.23, we get SNR values and loss percentage for tomte glitch as written in Table 4.1. From the loss percentage, we can say that SHAPES and waveshrink performs way better than the current aLIGO method i.e. gating. Amongst the SHAPES and waveshrink, we can see that SHAPES performing better than waveshrink. Also, we can see that the higher the frequency is, the worse the estimation of the combination approach of SHAPES and waveshrink becomes.

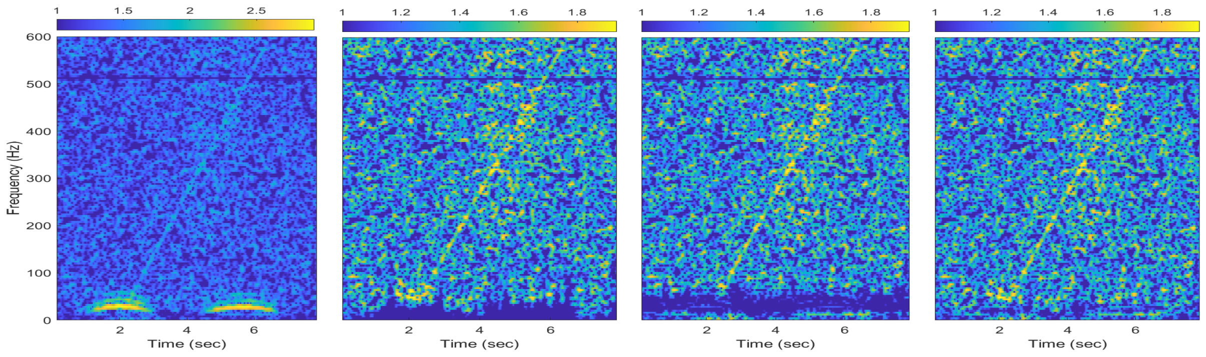
4.5 Slow Scattering Glitch

4.5.1 Results using amplitude threshold method

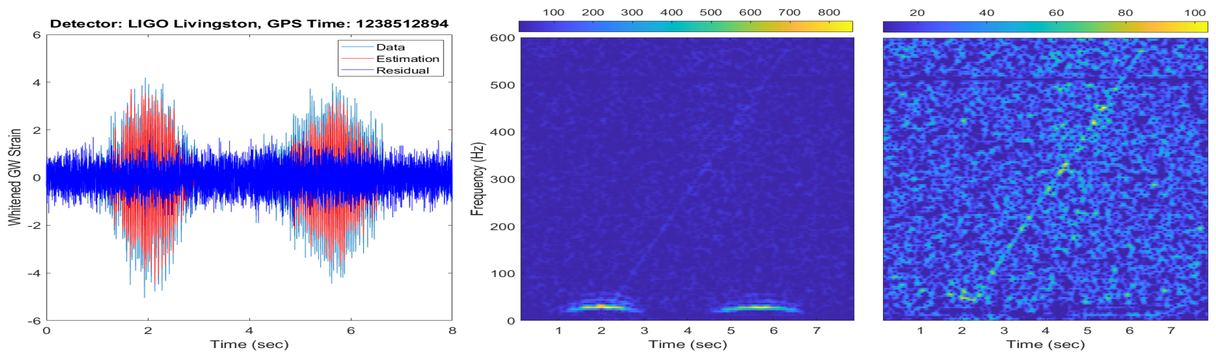
Fig. 4.17 (a) is the comparison of timeseries, from where we can see that at the top left panel, and the bottom right panel, the estimation made by the combination of SHAPES and waveshrink is smoother than the estimation of SHAPES and the estimation of waveshrink individually. In the case of SHAPES, the segment length 16384 has been applied to the rest of the data, other than the glitch part of the data. In the main glitch part of the data, the segment length is 6144. As the glitch is long in this case, SHAPES take out a lot of power below 50 Hz. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power from the general noise part. In the third panel of the fig. 4.17 (b), there is less power below between 10-70 Hz, which is caused by the overfitting of the waveshrink. From the right panel of the fig. 4.17 (b), we see that there is very less power taken out from the general noise, and the glitch is removed from the data. From fig. 4.17 (a), the top right panel and the bottom left panel are the estimations of the glitch by SHAPES, waveshrink, and a combination of SHAPES and waveshrink. In this case, the individual estimation made by SHAPES and Waveshrink followed the glitch very closely, but in some cases, SHAPES is following data closely, which is evident from the maximum and minimum peak of the glitch [Fig. 4.17 (a)]. That is why we can see from fig. 4.17 (b), that SHAPES is taking out the glitch and more power around the glitch below 50 Hz as before. The estimation made by the combination of SHAPES and waveshrink, fig. 4.17 (b) right panel, followed the glitch part of the data lesser than the individual estimation of SHAPES and Waveshrink. Thus, it takes out very less power from the glitch part of



(a)



(b)



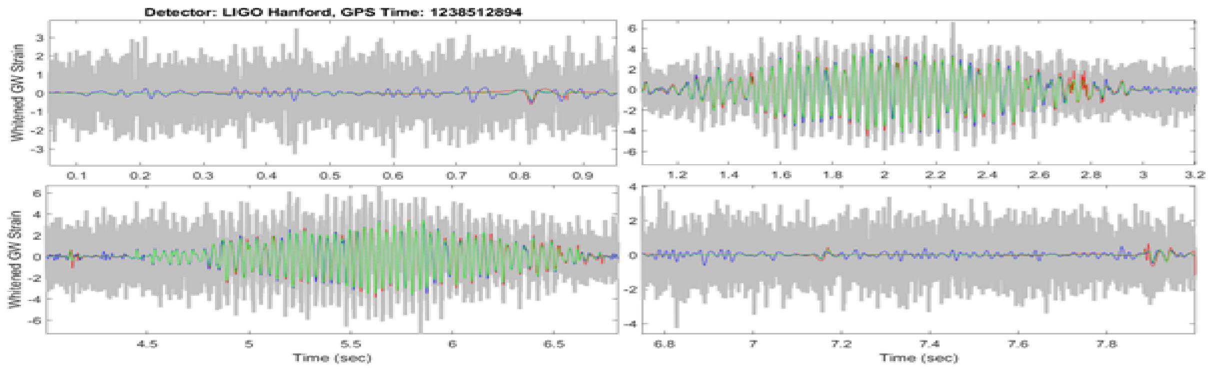
(c)

Figure 4.17: Estimation and removal of the slow scattering glitch using 6144 as a segment length on the glitch with signal I for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.

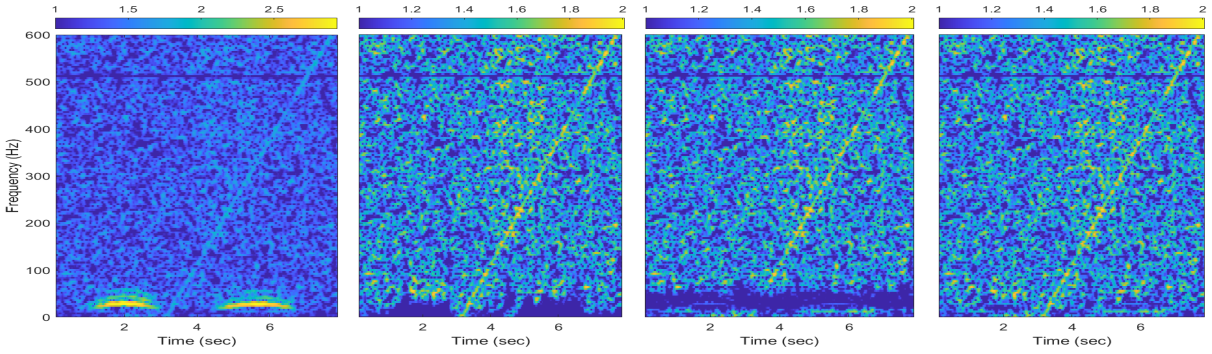
the data compared to the general spectrogram of the individual glitch removal of SHAPES and Waveshrink. From this result, we can confirm that the combination of SHAPES and waveshrink performs better in low-frequency glitches. It is also confirmed by the SNR values in table 4.1. The first linear chirp signal (sig. I in table 4.1) is injected around 2s i.e. in the middle of one of the glitch islands. From the loss percentage and values of SNR, we can say that SHAPES and waveshrink both performed better than the gating procedure of LIGO. Amongst SHAPES and waveshrink, SHAPES performs similarly to waveshrink for preserving the signal. The percentage difference is very small. But the combination of SHAPES and waveshrink in this case works better than the individual systems by considering the loss percentage and SNR values, which is also evident from figure 4.17 (b). Therefore, it also confirms from the values of SNR and loss percentage in table 4.1 that the combination of SHAPES and waveshrink works better in low-frequency glitches (the assumption we made in section 4.5.1).

Also, in the previous fig. 4.17, the signal started from the middle of the glitch. But if the signal (sig. II in table 4.1) started in between the glitch islands i.e. the low frequency of glitch will start from the part where is no glitch present, SHAPES preserve the full signal approximately. As SHAPES preserve the signal, the combination of SHAPES and waveshrink also preserves the signal. However, waveshrink removed the low-frequency part of the signal. These statements are illustrated in the below fig. 4.18. Thus, from the loss percentage and SNR in the table 4.1, we can say that SHAPES and waveshrink both performed better than the gating procedure of LIGO, as before. Amongst SHAPES and waveshrink, SHAPES preserved more signal parts than waveshrink. But the combination of SHAPES and waveshrink in this case works better than the individual systems, as it preserves signal and general noise more than both systems.

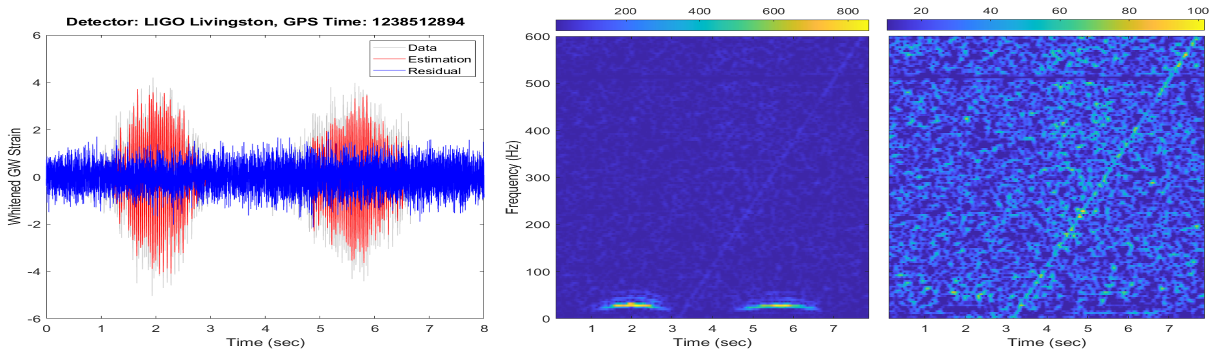
For getting figure 4.19, we have applied 6144 segment length in the glitch part and 16384 segment length in the rest of the data for making the estimation of SHAPES. Also, figure 4.19 has two signals (sig. I and sig. II) in the same data as we can see from the spectrogram [figure 4.19 (b)]. That means we have added two signals in the same data and checked whether it affects the estimate of the SHAPES, which gives a result with no change i.e. no extra effect on the signals.



(a)

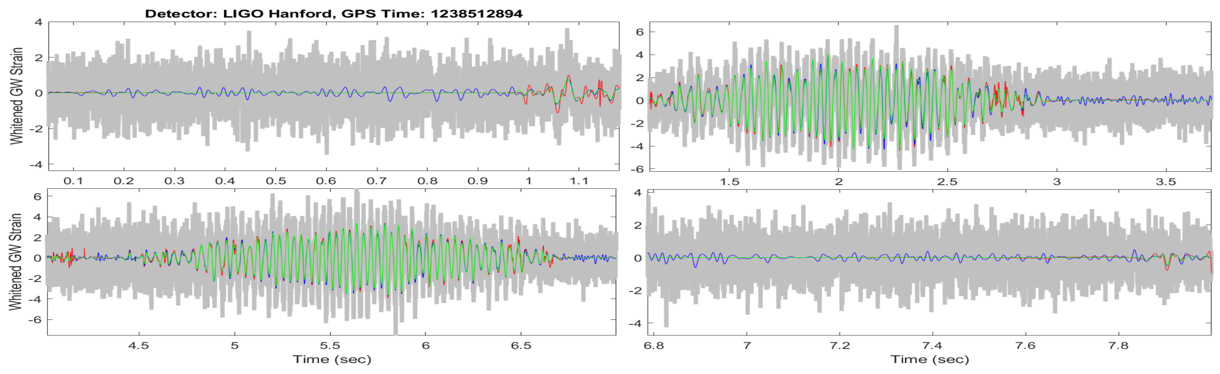


(b)

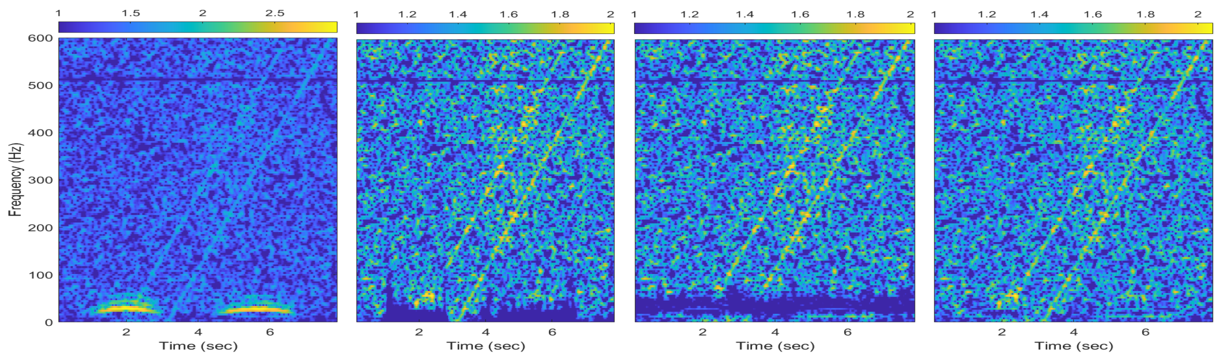


(c)

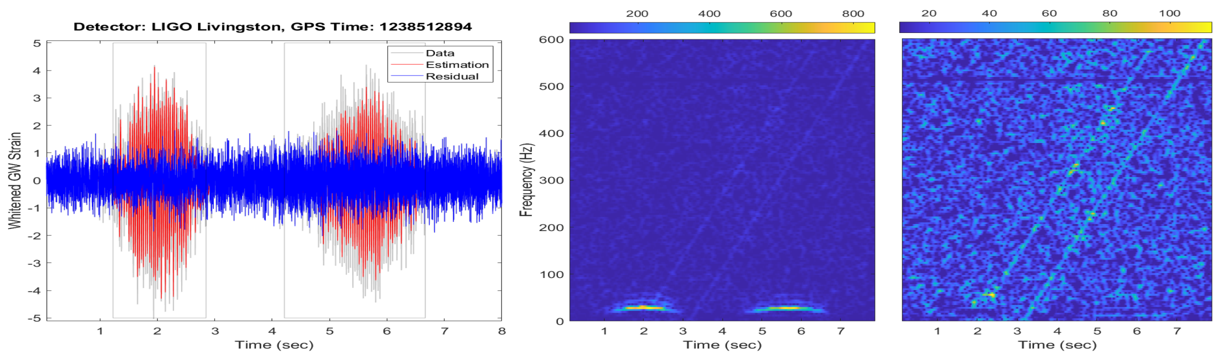
Figure 4.18: Estimation and removal of the slow scattering glitch using 6144 as a segment length on the glitch with signal II for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.



(a)

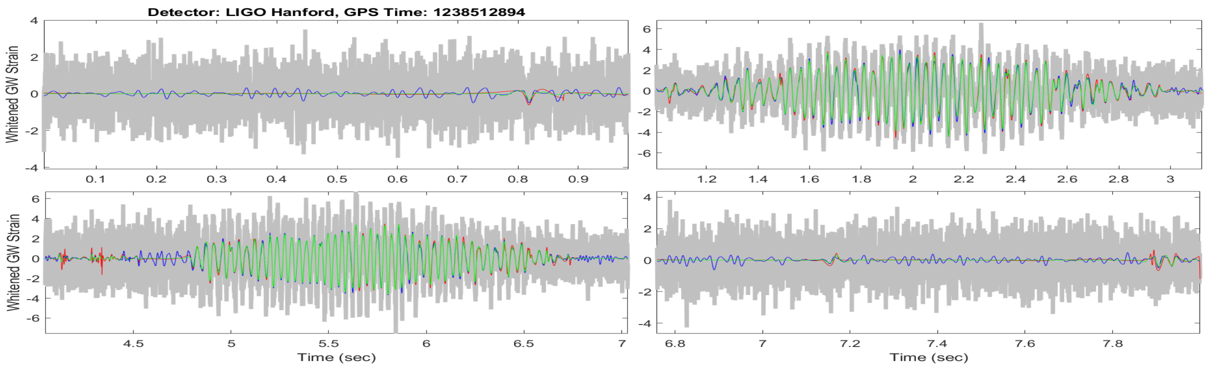


(b)

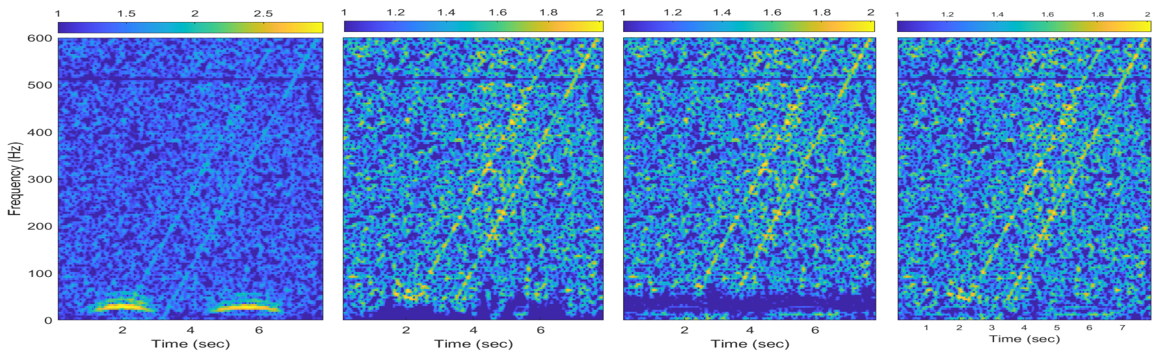


(c)

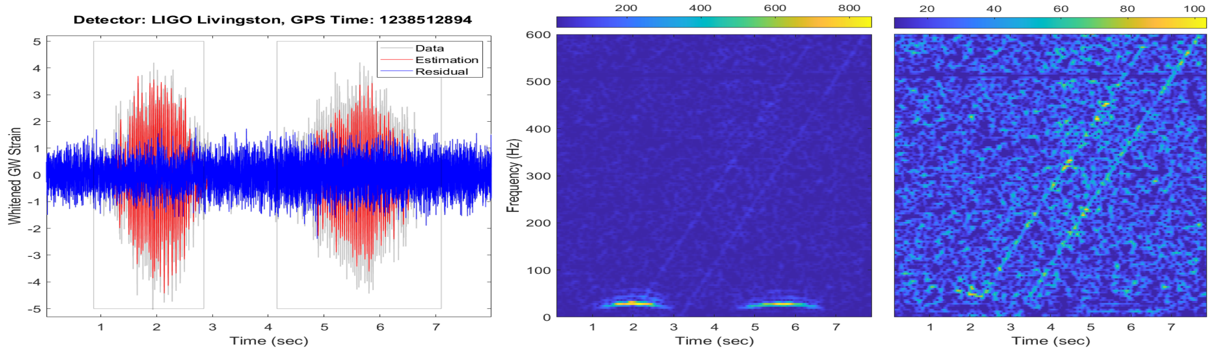
Figure 4.19: Estimation and removal of the slow scattering glitch using 6144 as a segment length on the glitch with two signals for fitting the data. These figures have regulator gain 0.3. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink, black boxes around the glitch is the glitch boundary identified by amplitude threshold method.



(a)



(b)



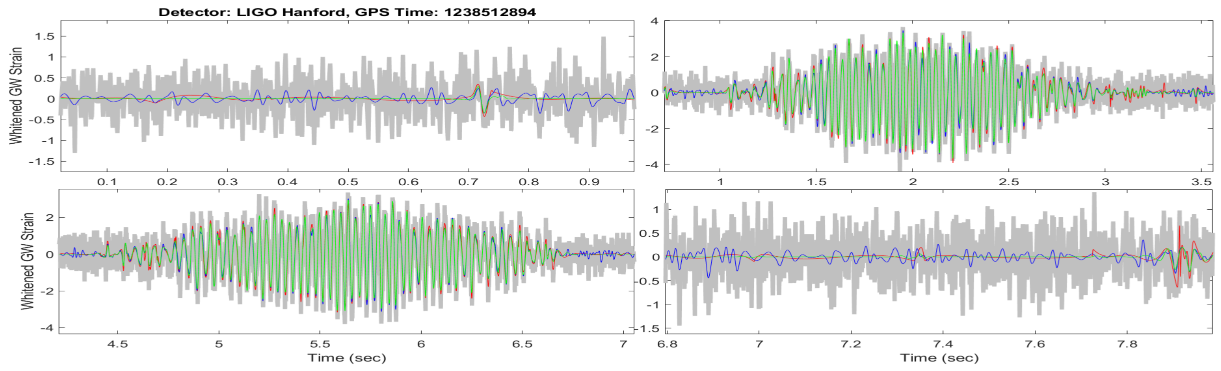
(c)

Figure 4.20: Estimation and removal of the slow scattering glitch using 6144 as a segment length on the glitch with two signals for fitting the data.These figures have regulator gain 0.1. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink, black boxes around the glitch is the glitch boundary identified by amplitude threshold method.

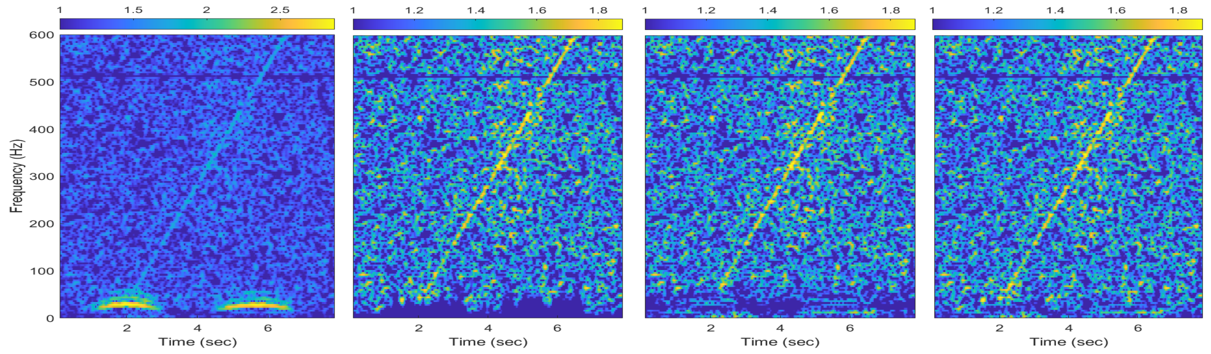
Figure 4.17, figure 4.18, figure 4.19 has regulator gain 0.3, unlike the other figures in the whole chapter IV. Though it is preserving more signal (not significantly different), changing the regulator gain a little does not affect much in case of glitches and signal as we can see from comparing fig. 4.20 (b) with fig. 4.19 (b). Changing the regulator gain, by 0.01 - 0.15, does not have a noticeable effect in case of glitches, which has more frequencies than 100 Hz. If the regulator gain is increased after a certain value, which varies with glitches (for example, 0.15 for GW170817), it starts to keep parts of the glitch. But it has a significant effect on signal II which is in between the glitch islands and buried behind the gaussian noise. In the case of figure 4.19 (b), we can see that it nearly preserved the whole signal II whereas it is not preserved equally in fig. 4.20 (b). The difference is also apparent in the SNR values and loss percentage from table 4.1. So, we can say that by making the estimation smoother, by changing regulator gain, in the Gaussian noise (general noise) part of the data, more power can be saved.

All of the previous graphs are for normalized data i.e. with a standard deviation value of 1. With a standard deviation of the noise is 1, waveshrink performs better to remove high-frequency glitches. But due to this good performance, it takes out more power around the glitch which we can see from the third panel of the fig. 4.17 (b) or the fig. 4.18 (b) or the fig. 4.19 (b). After data conditioning of GW strain data in our case, the standard deviation is a little less than 1, to be exact 0.8699 for slow scattering glitch. If we do not make standard deviation 1, then the combination of SHAPES and waveshrink performs even better than the previous graphs of slow scattering glitches. We get figure 4.21 by applying segment length 6144 in the glitch part and 16384 in the rest of the data for fitting the data by SHAPES.

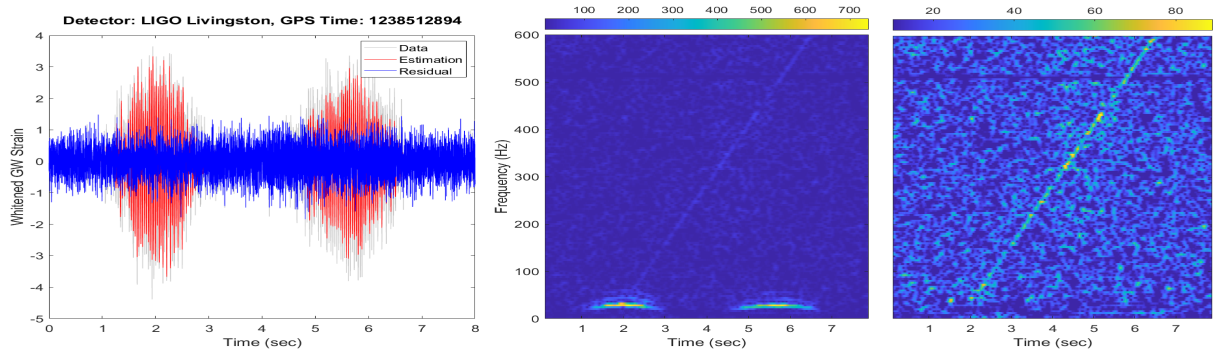
From the right panel of the figure 4.21 (b), we can see that there is more general noise than in the fig. 4.17 (b) or the fig. 4.18 (b) or the fig. 4.19 (b) of the slow scattering glitch. From the second and third panel of fig.4.21 (b), we can see that both SHAPES and waveshrink is removing less power than in the previous graph. Even the SNR values recovered by the combination of SHAPES and waveshrink, including individual systems (SHAPES, and waveshrink), is higher than the normalized data. When we injected the signal into glitch-removed data, the SNR of the signal



(a)

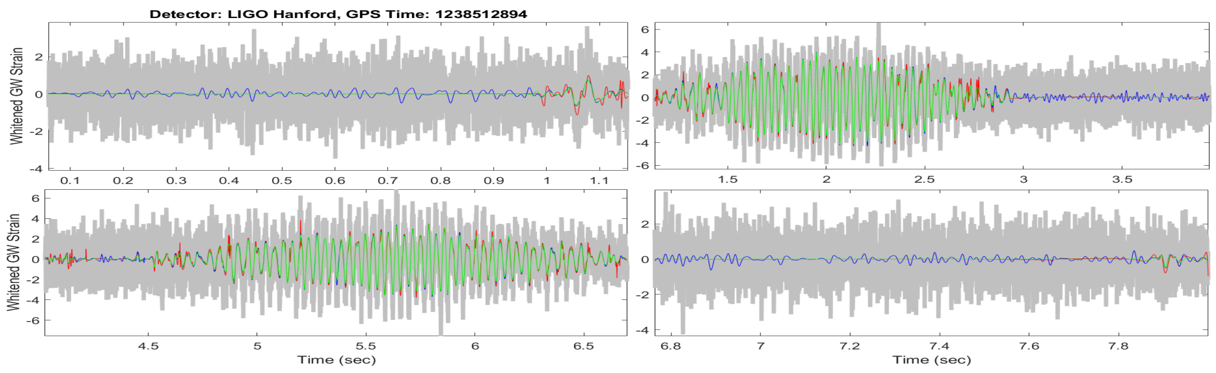


(b)

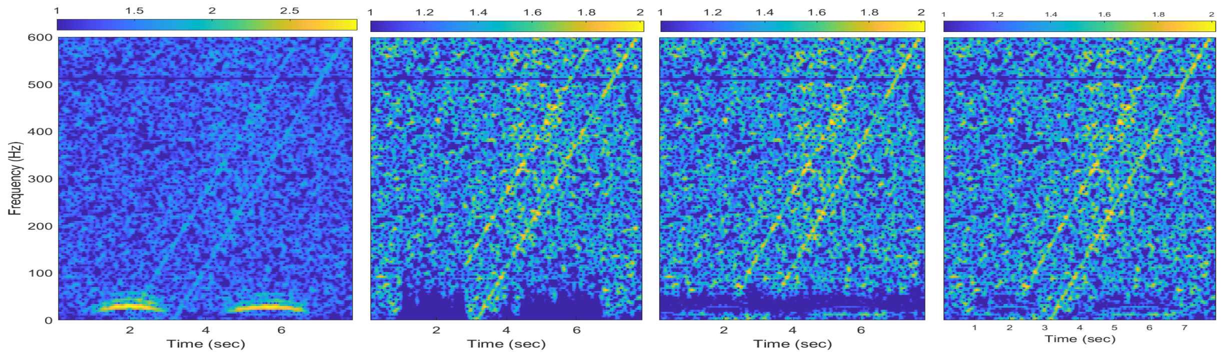


(c)

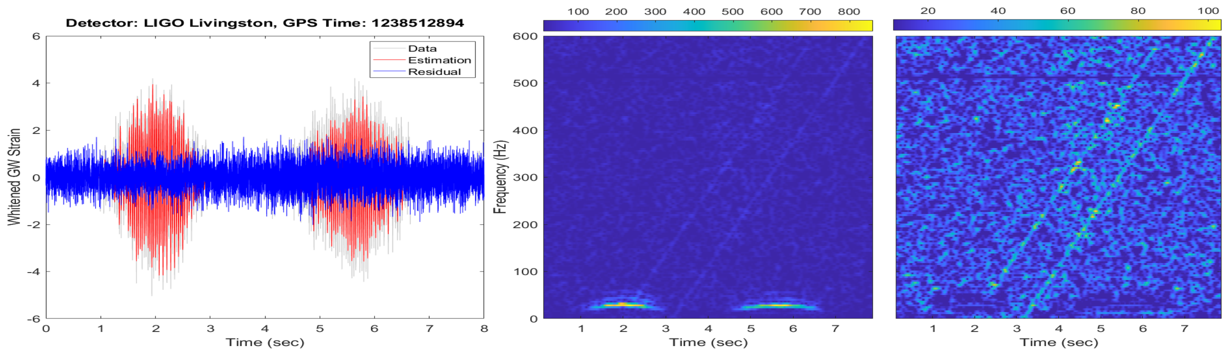
Figure 4.21: Estimation and removal of the slow scattering glitch using 6144 as a segment length on the glitch for fitting the data. In this case, standard deviation of the data is not normalized. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.



(a)



(b)



(c)

Figure 4.22: Estimation and removal of the slow scattering glitch using 4096 as a segment length on the glitch for fitting the data. In this case, standard deviation of the data is not normalized. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.

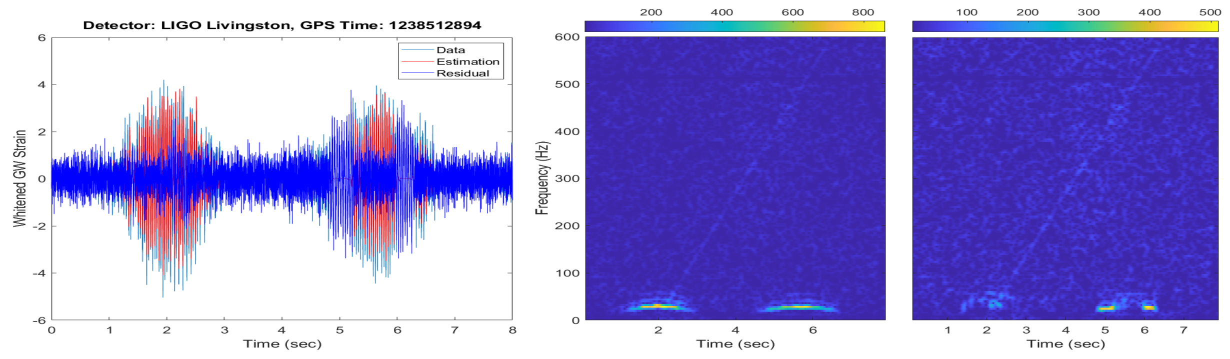


Figure 4.23: Glitch removal with applying 8192 segment length in the glitch part of the data for fitting the data

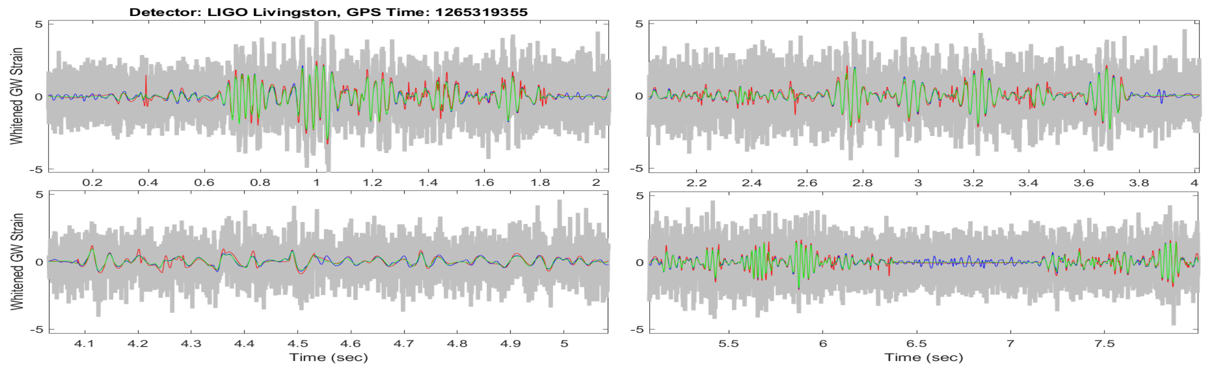
became 25.8172. After glitch removal, the SNR recovered by SHAPES is 24.8348 (which is 3.80% less than SNR in glitch removed data) and the SNR recovered by waveshrink is 24.4710 (which is 5.21% less than SNR in glitch removed data). With the current procedure of LIGO (gating the data around the glitch), the amount of SNR of the chirp signal recovered is 10.5899 (which is 58.98% less than SNR in the glitch removed data). From the percentage and values of SNR, we can say that SHAPES and waveshrink both performed better than the gating procedure of LIGO. Amongst SHAPES and waveshrink, SHAPES performs similarly to waveshrink for preserving the signal. Again, the combination of SHAPES and waveshrink in this case works better than the individual systems. In the case of the glitch removal, the SNR recovered by the combination of SHAPES and waveshrink is 25.2234 (which is 2.30% less than SNR in the glitch removed data). So, in case of a slow scattering glitch, the combination of SHAPES and waveshrink performs better. This result also confirms that the combination of SHAPES and waveshrink works better in low-frequency glitches.

The slow scattering glitch can be subtracted using a lower segment length value. But if we apply a higher segment length value than 6144 i.e. 8192 for fitting the data by SHAPES, it can not remove the full glitch [Fig. 4.23. The below fig. 4.22 of slow scattering has 4096 segment length in the glitch part and 16384 segment length in the rest of the data for fitting the data by SHAPES. Also, figure 4.22 have two signal (sig.I and sig.II) as we can see from spectrogram [figure 4.22 (b)].

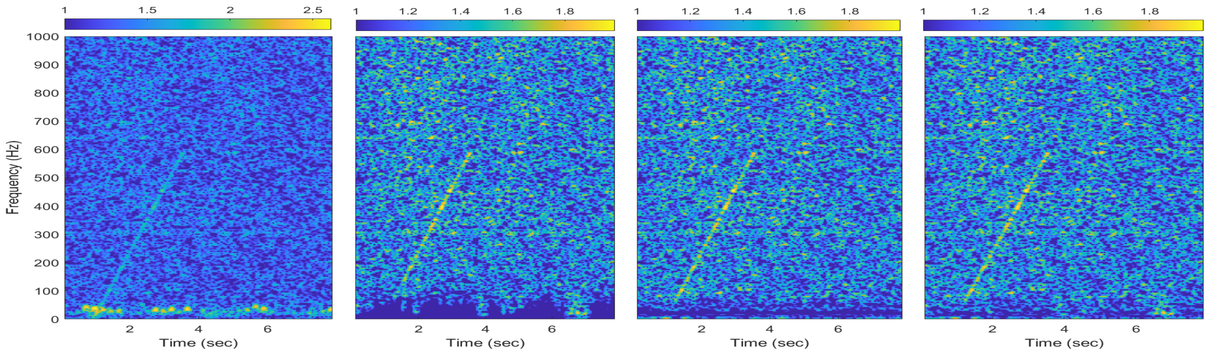
4.6 Fast Scattering Glitch

4.6.1 Results using amplitude threshold method

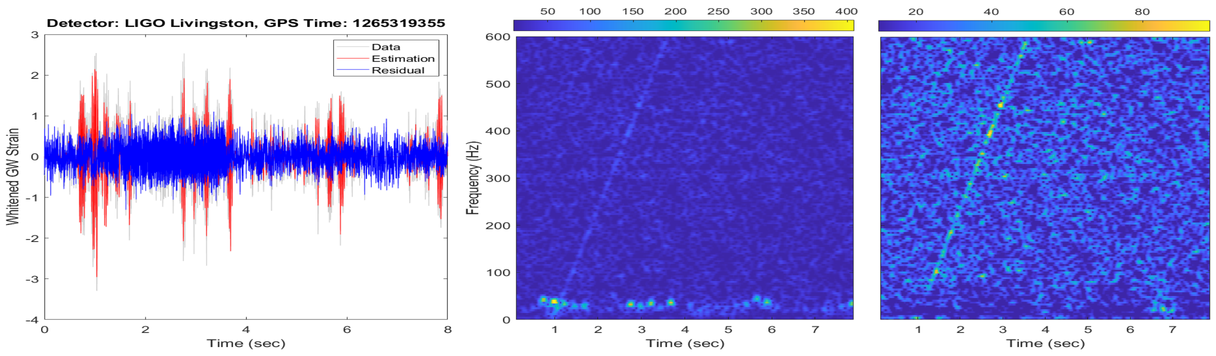
Figure 4.24 (a) is the comparison of timeseries, from where we can see that at the top left panel in between 0-0.6s and 1.8-2.1s, the bottom left panel and the bottom right panel in between 6.5-7.1s, the estimation made by the combination of SHAPES and waveshrink is smoother than the estimation of SHAPES and the estimation of waveshrink individually. In the case of SHAPES, the segment length 16384 has been applied to the rest of the data, other than the glitch part of the data. In the glitch part of the data, the segment length is 4096. As the glitch is long in this case, SHAPES take out a lot of power below 90 Hz. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES, it takes out more power from the general noise part. In the third panel of the fig. 4.24 (b), there is less power below between 07-40 Hz, which is caused by the overfitting of the waveshrink. But in this case, it takes out less power than SHAPES. It is caused by the small gap (less time interval) between the islands of glitch. From the right panel of the fig. 4.24 (b), we see that there is very less power taken out from the general noise, and the glitch is removed from the data. From fig. 4.24 (a), the top right panel and the bottom left panel are the estimations of the glitch by SHAPES, waveshrink, and a combination of SHAPES and waveshrink. In this case, the individual estimation made by SHAPES and Waveshrink followed the glitch very closely, but in some cases, SHAPES is following data closely, which is evident from the maximum and minimum peak of the glitch [Fig. 4.24 (a)]. That is why we can see from fig. 4.24 (b), that SHAPES is taking out the glitch and more power around the glitch below 50 Hz as before. The estimation made by the combination of SHAPES and waveshrink, fig. 4.24 (b) right panel, followed the glitch part of the data lesser than the individual estimation of SHAPES and Waveshrink. Thus, it takes out very less power from the glitch part of the data compared to the general spectrogram of the individual glitch removal of SHAPES and Waveshrink. From this result, we can confirm that the combination of SHAPES and waveshrink performs better in low-frequency glitches.



(a)



(b)



(c)

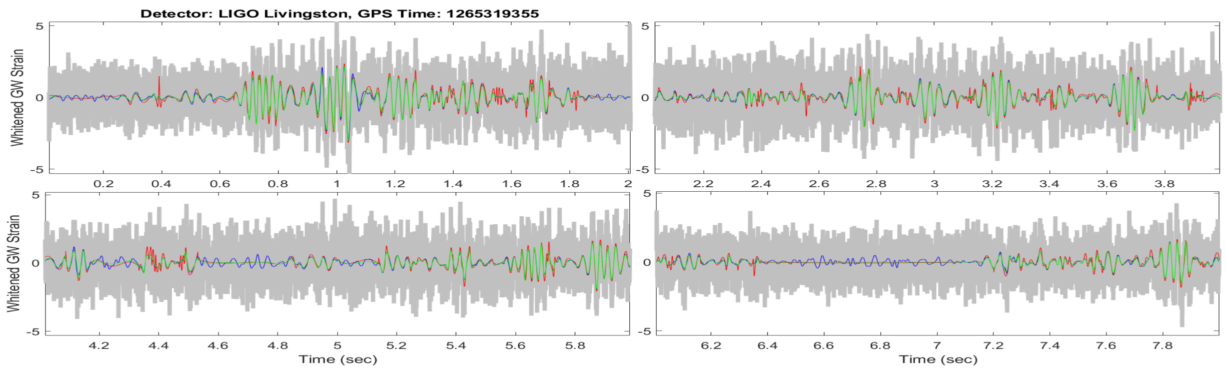
Figure 4.24: Estimation and removal of the fast scattering glitch using 4096 as a segment length on the glitch with signal I for fitting the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.

It is also confirmed by the SNR values in table 4.1. The first linear chirp signal (sig. I in table 4.1) is injected around 1 s. From the loss percentage and values of SNR, we can say that SHAPES and waveshrink both performed better than the gating procedure of LIGO, as before. Amongst SHAPES and waveshrink, SHAPES performs worse to waveshrink for preserving the signal. But the combination of SHAPES and waveshrink in this case works better than the individual systems by considering the loss percentage and SNR values in table 4.1, which is slightly lower and slightly higher (respectively) than waveshrink. Also, the combination method of SHAPES and waveshrink preserves more gaussian (general) noise than SHAPES and waveshrink individually. Therefore, it also confirms that the combination of SHAPES and waveshrink works better in low-frequency glitches.

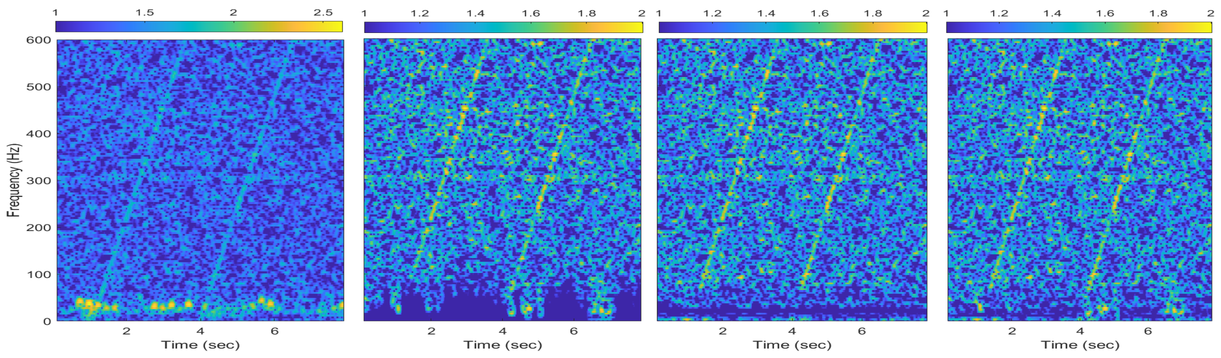
We have added a second signal (sig. II in table 4.1 in the data, where SHAPES is taking out less power to see the effect of estimation (as we have done in section 4.5.1). In the case of fast scattering, it preserves the signal more [Fig. 4.25 (b)] if the signal started from a position where is no glitch present. It is more evident in SNR values and loss percentage of sig. II in table 4.1, that starts from 4 s. In case of fast scattering, the loss percentage and SNR sig.II is lower and higher than sig.I. But the preservation of signal is not as good as sig. II of slow scattering case. That means, the combination of SHAPES and waveshrink still performing better than individual methods. But it did not perform as good as slow scattering cases, where it preserved more parts of the signal [Fig. 4.19].

The fast scattering glitch can also be removed using 6144 as a segment length for fitting the data by SHAPES [Fig. 4.26. It keeps more power around the glitch in the data as we can see from figure 4.26 (b). But above 6144 like 8192 as a segment length, will keep glitch in the data [Fig. 4.28].

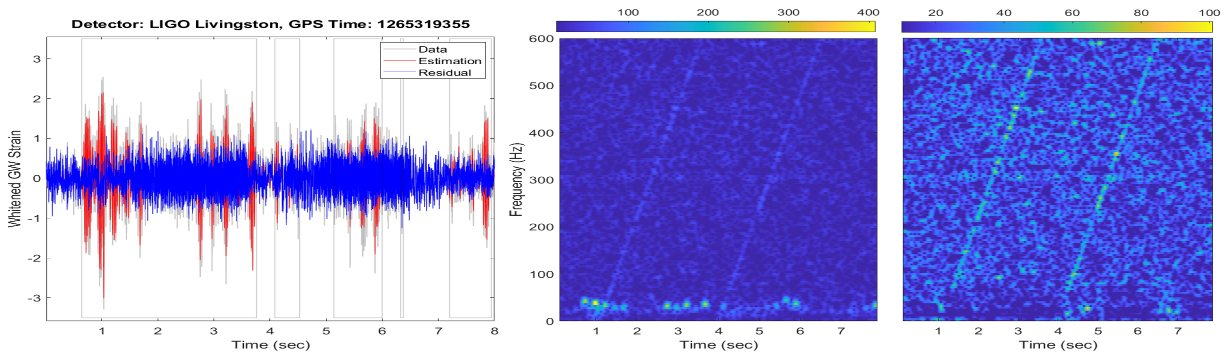
Similar to the slow scattering glitch, the performance of the combination of SHAPES and waveshrink is better if the data is not normalized. In case of fast scattering glitch, the standard deviation is 0.9052. The below figure shows the application of the code on GW strain data which is not normalized.



(a)

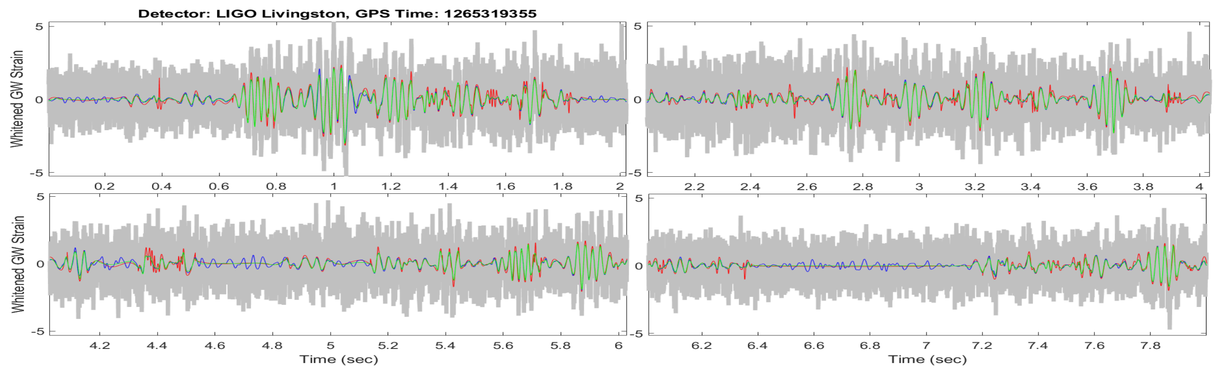


(b)

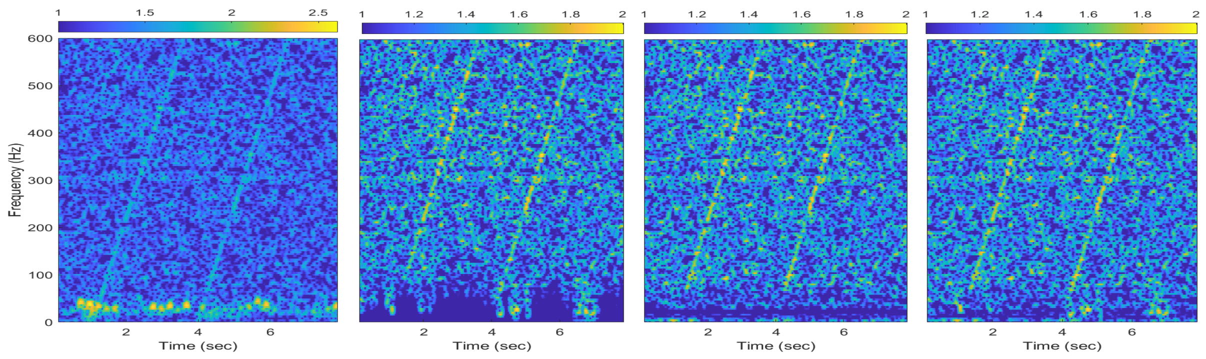


(c)

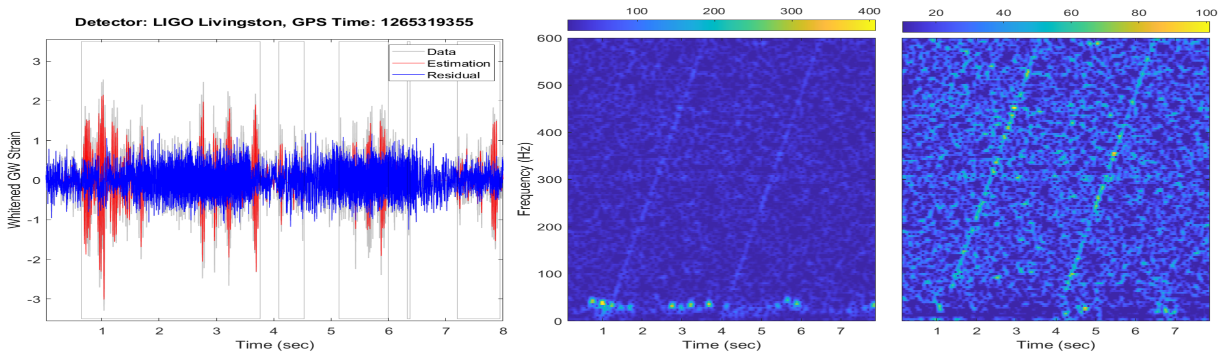
Figure 4.25: Estimation and removal of the fast scattering glitch using 4096 as a segment length on the glitch for fitting the data. These figures have two signal in the data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink, black boxes around the glitch is the glitch boundary identified by amplitude threshold method.



(a)

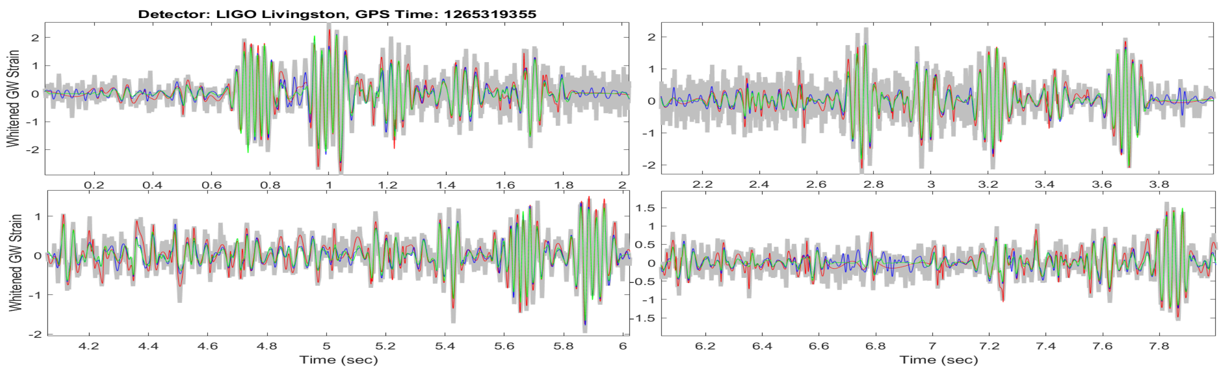


(b)

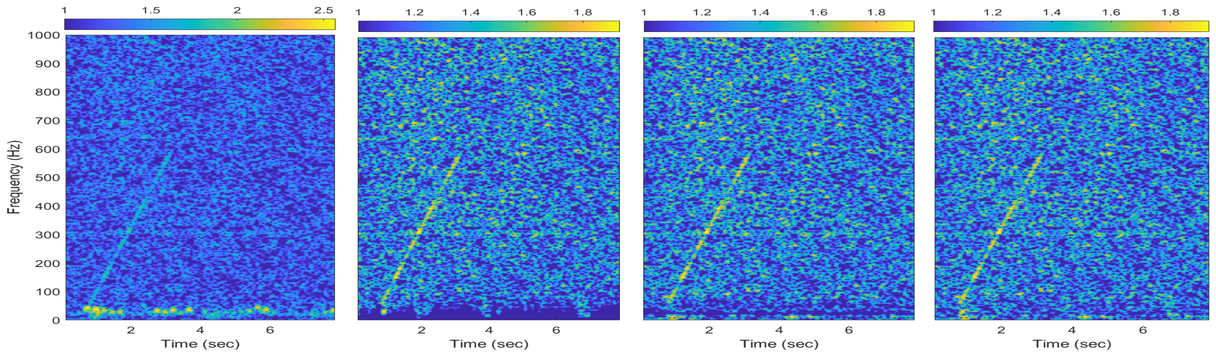


(c)

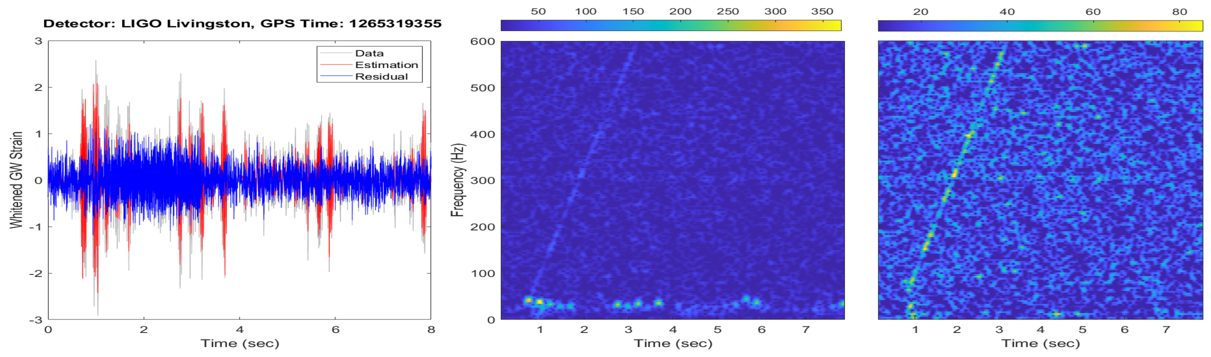
Figure 4.26: Estimation and removal of the fast scattering glitch using 6144 as a segment length on the glitch for fitting the data. These figures have two signal. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink, black boxes around the glitch is the glitch boundary identified by amplitude threshold method.



(a)



(b)



(c)

Figure 4.27: Estimation and removal of the fast scattering glitch using 6144 as a segment length on the glitch for fitting the data. These figures have been produced without normalized data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, red = estimation of SHAPES, blue = estimation of waveshrink, green = estimation of the combination of SHAPES and waveshrink). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using the combination approach of SHAPES and waveshrink.

From the right panel of the fig. 4.27 (b), we can see that there is more general noise around 10-30 Hz than in the previous graph of the glitch. From the second and third panel of fig. 4.27 (b), we can see that both SHAPES and waveshrink is removing less power than in the previous graph. When we injected the signal into glitch-removed data, the SNR of the signal became 24.8070. After glitch removal, the SNR recovered by SHAPES is 21.6444 (which is 12.74% less than SNR in glitch removed data) and the SNR recovered by waveshrink is 22.4160 (which is 9.63% less than SNR in glitch removed data). With the current procedure of LIGO (gating the data around the glitch), the amount of SNR of the chirp signal recovered is 7.9085 (which is 68.12% less than SNR in the glitch removed data). Thus, from the percentage and values of SNR, we can say that SHAPES and waveshrink both performed better than the gating procedure of LIGO. Amongst SHAPES and waveshrink, waveshrink performs better than SHAPES for preserving the signal. Again, the combination of SHAPES and waveshrink in this case works better than the individual systems. In the case of the glitch removal, the SNR recovered by the combination of SHAPES and waveshrink is 22.6431 (which is 8.72% less than SNR in the glitch removed data). So, in case of the fast scattering glitch, the combination of SHAPES and waveshrink performs better.

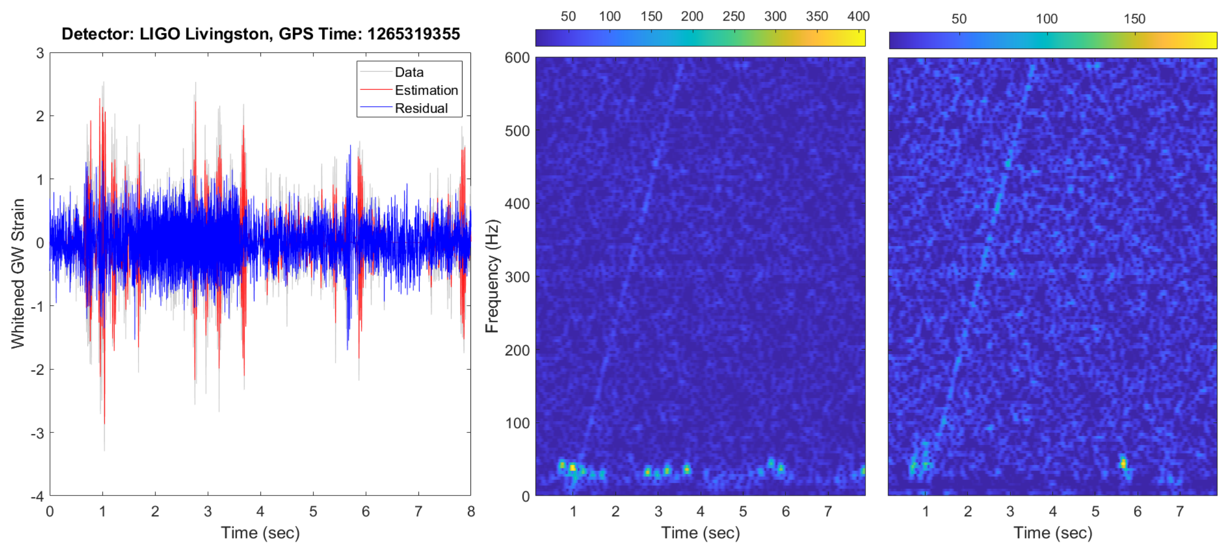
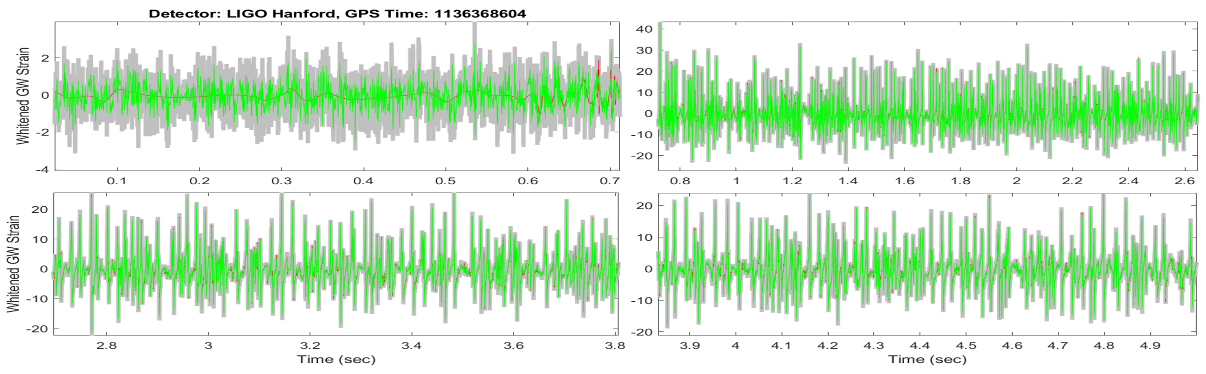


Figure 4.28: Glitch removal with applying 8192 segment length, on the glitch part of the data, for fitting the data

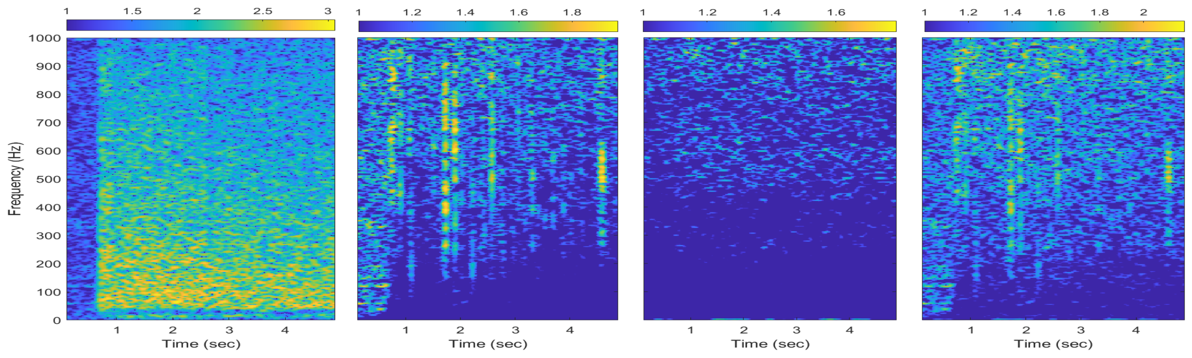
4.7 Extremely Loud Glitch

4.7.1 Results using amplitude threshold method

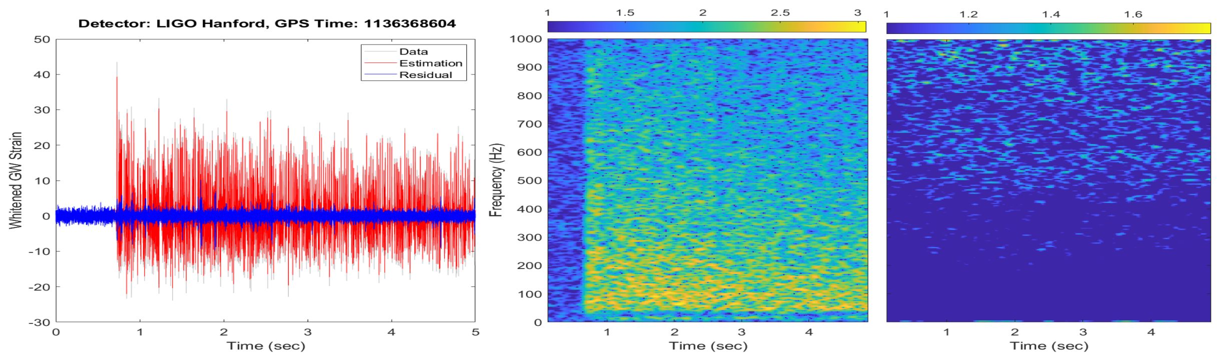
Fig. 4.29 (a) is the comparison of timeseries, from where we can see that at the top left panel, the estimation made by SHAPES is smoother than the waveshrink. In all other panels of fig. 4.29 (a), both methods, SHAPES, and waveshrink estimate the glitch very closely. In some cases, we can see that the estimation of the SHAPES (red color estimation in fig.4.29 (a)) is following the data more closely as the peak is seen behind the estimation of waveshrink (green color estimation fig.4.29 (a)), for example, 0.65s - 0.7s at the top left panel. In the case of SHAPES, the segment length 16384 has been applied for getting the estimation of the data, other than the glitch part. In the glitch part of the data, the segment length applied is 256. When the glitch is removed using waveshrink, as the estimation of waveshrink is following the general noise more closely than the estimation of SHAPES (what can be seen in the top left panel of fig. 4.29 (a) from 0s to 0.65s), it takes out more power in case of general/Gaussian noise. The power difference is visible from the third panel of the general spectrogram figure [figure 4.29 (b)]. In the third panel of figure 4.29 (b), there is less power between 0s and 0.6s approximately, which is caused by the overfitting of the waveshrink. From figure 4.29 (a), the top right panel and the bottom panels are the estimations of the glitch by SHAPES and waveshrink. The glitch is very long, approximately 900s long disturbance of the data. In normal cases, probably this data will be vetoed by data quality. In this case, both systems estimated the glitch pretty well [fig. 4.29 (a)]. But from the spectrogram of figure 4.29 (b), we see that there is a trace of the glitch left behind in the second panel, which is a result of glitch removal by SHAPES estimation. The combination of the estimation of SHAPES and waveshrink failed in this case as we can see from figure 4.29 (b) right panel. From section 4.1, 4.2, 4.3, 4.4, and this section, we can conclude that the combination approach of SHAPES and waveshrink will fail if either of the individual systems fails in any glitch cases. A linear chirp signal is injected into data around 0.7s. But neither of the methods kept a trace of the signal in this glitch case. Also, these methods did not affect the general/Gaussian noise much after 700 Hz.



(a)



(b)



(c)

Figure 4.29: Estimation and removal of the extremely loud glitch using 256 as a segment length on the glitch part of the data for fitting the data. These figures have been produced without normalized data. Figure (a) is the comparison of the estimation of methods in time series (Gray = data, green = estimation of waveshrink, red = estimation of SHAPES). Figure (b) is the comparison of glitch removal (first panel = data, second panel = Glitch Removal by SHAPES, third panel = Glitch removal by WaveShrink, fourth panel = Glitch removal by the combination of SHAPES and Waveshrink). Figure (c) is the data and residual of the data using waveshrink.

Table 4.1: SNR Values, using amplitude threshold, for linear chirp signal in the data. The loss for every method is a percentage. In the first column of the table, glitch type is written in short form. Full forms of glitches are LFB = Low Frequency Blip, KF = Koi Fish, Tm = Tomte, SS = Slow Scattering, and FS = Fast Scattering. Also, in this table, SS (sig.I) (6144) (rg-0.3) means that it is the slow scattering glitch, the sig.I is the injected signal in the first position in the case of two signal injections, 6144 means that the segment length 6144 is applied for estimating the glitch part of the data, and rg -0.3 means that the regulator gain is 0.3. Similarly, other glitch naming is written for slow scattering (SS) and fast scattering (FS). Other than slow scattering glitch (6144) cases, all of the glitches estimations are done with regulator gain 0.1.

Glitch Name	SNR (in residual)	SNR (from SHAPES residual)	SNR (from Waveshrink residual)	SNR (from SHAPES and waveshrink residual)	SNR (from gating procedure)
		Loss	Loss	Loss	Loss
LFB	24.49	2.11	15.22	2.28	24.00
KF	25.06	5.38	17.12	2.72	24.62
Tm	25.60	14.29	33.79	6.45	45.42
SS (sig.I) (6144) (rg-0.3)	25.43	9.56	9.84	6.11	69.61
SS (sig.II) (6144) (rg-0.3)	26.00	0.01	5.68	0.16	53.27
SS (sig.I) (6144) (rg – 0.1)	25.43	8.75	9.85	6.08	71.07
SS (sig.II) (6144) (rg-0.1)	26.00	5.91	5.67	2.79	61.06
SS (sig.I) (4096)	25.34	16.01	9.56	7.63	69.52
SS (sig.II) (4096)	25.96	3.06	5.26	0.10	70.24
FS (sig.I) (4096)	23.73	16.69	8.88	8.64	100
FS (sig.II) (4096)	23.92	13.16	8.68	7.20	100
FS (sig.I) (6144)	23.73	16.69	8.88	8.64	100
FS (sig.II) (6144)	23.92	13.16	8.68	7.20	100

Table 4.2: SNR Values, using bestfitness-spectrogram threshold, for linear chirp signal in the data. The loss for every method is a percentage. In the first column of table, glitch type is written in short form. Full forms of glitches are: LFB = Low Frequency Blip, KF = Koi Fish

Glitch	SNR (in residual)	SNR (from SHAPES residual)		SNR (from Waveshrink residual)		SNR (from SHAPES and waveshrink residual)		SNR (from gating procedure)	
		SNR	Loss	SNR	Loss	SNR	Loss	SNR	Loss
LFB	24.49	24.03	1.84	20.76	15.21	23.94	2.22	23.57	24.00
KF	25.06	24.23	3.3	20.76	17.13	24.61	1.77	22.94	24.62

Table 4.3: Glitch Timeseries. This table contains all the GPS timeseries that we have dealt in this research

Glitch Name	GPS start time (timeseries)	Glitch start time (around)	Detector
GW170817 (Loud Blip)	1187006835	1187008882.4	Livingstone
Low Frequency Blip	1136365568	1136368106	Hanford
Tomte	1252900864	1252903456	Livingstone
Koi Fish	1132929024	1132939287	Hanford
Slow Scattering	1238511616	1238512894	Livingstone
Fast Scattering	1265315840	1265319354	Livingstone
Extremely Loud Glitch	1136365568	1136368605	Hanford

CHAPTER V

CONCLUSION

Noise is a problem in every type of data including GW data. The linear noise in GW data has been dealt with by aLIGO. Glitch is also a type of non-stationary noise, which is better known as non-astrophysical signals or transient signals. Glitch is unexpected and also the waveform of glitch varies with the type of glitch. It even varies with the difference in detector systems. Thus, estimating and removing a glitch from the GW data is a very challenging task. It needs to be removed from the data, as glitch increases the false alarm rate, causes a problem for parameter estimation of the signal, and limits the aLIGO sensitivity. Classifying the glitches helps, but it is not a solution. As the aLIGO sensitivity is increasing, a new glitch may be introduced to data. As it is a new glitch, it would not fall into any old classification.

Using the classification of the glitch and auxiliary channels to model the glitch, some algorithms have been built as discussed in section 2.3. But these methods will fail to detect new glitches without any previous knowledge of the glitch. Even with the training of specific glitches in these algorithms, it could not detect the glitches from other detectors as the waveform varies. The new algorithm, SHAPES , that we applied to the GW data, does not need any previous knowledge of the glitches to identify, estimate, and remove the glitches. Therefore, SHAPES can detect new glitches, along with the classified glitches, without any training or previous knowledge of the glitches. *Bayeswave*, which is mostly used in GW data analysis in aLIGO, takes too much time to estimate and model a glitch. Compared to *Bayeswave*, SHAPES takes much less time to identify, and estimate the glitch. If SHAPES is applied to the glitch-only part, it can identify, estimate and remove glitches in 4096 s data in 5-30 minutes (depending on how noisy the data is). In case of two noisy data i.e. repeated slow scattering glitch, fast scattering glitch, it will take longer time than the time mentioned before. If SHAPES is applied on the full data as in the chapter IV, it will take longer time i.e. 2-3 hours. Waveshrink takes time in seconds to build an estimation. Thus, the combination of SHAPES and waveshrink takes similar time to SHAPES to estimate the data. Still SHAPES is faster than *Bayeswave* in all cases. Moreover, SHAPES and the combination of SHAPES and waveshrink work on a wide variety of glitches and various frequencies, while the methods mentioned in 2.3 work on a particular type of glitches.

The amplitude threshold method for identification of the glitches can identify the glitch very fast i.e. approximately in 1μs. But in the case of the best fitness-spectrum threshold method, it takes more time than the amplitude threshold method. For acquiring a good estimation of the data including glitch by SHAPES , glitch boundary needs to be defined accurately. From section 4.1, section 4.2, and section 4.3, it is clear that the best fitness-spectrum threshold method works more accurately to define glitch boundary than the amplitude threshold method. Also, as SHAPES is applied repeatedly in the case of the best fitness-spectrum threshold method, it needs more computational power than the amplitude threshold method.

The segment length for fitting the glitch part of the data is shorter than the Gaussian/general noise of the data. Also, the segment length varies with the frequency of the glitches. The higher the frequency, the shorter the segment length should be. As discussed in 4.2 , 4.3 ,and 4.4, the glitch is completely subtracted by using a longer segment length than the glitch boundary if the glitch is a low-frequency or mid-frequency i.e. around 10-600 Hz, and short duration glitch. For high-frequency glitches, a shorter segment length can subtract the glitches from the data. But the segment length varies with different high-frequency glitches i.e. the value or a pattern is still not generalized. Thus, there is some manual tuning needed for high-frequency glitches. Therefore, the system is not fully automatic, and more work needed to be done in case of finding a pattern in high-frequency glitches. . Furthermore, the methods take out more power, when it is applied to shorter segment length (e.g. glitch part of the data), which is a problem for short duration signals. In future work, it should be investigated how the subtraction of more power can be reduced. One of the possible solutions is to increase the regulator gain value [sec. 4.5].

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems'.
URL: *http://download.tensorflow.org/paper/whitepaper2015.pdf*

Abbott, B. P., Abbott, R., Abbott, T. D., Abernathy, M. R., Acernese, F., Ackley, K., Adams, C., Adams, T. & et al., A. (2018), 'Effects of data quality vetoes on a search for compact binary coalescences in Advanced LIGO's first observing run', *Classical and Quantum Gravity* **35**(6), 065010.

Abbott, B. P. & et al., R. A. (2020), 'A guide to LIGO–virgo detector noise and extraction of transient gravitational-wave signals', *Classical and Quantum Gravity* **37**(5), 055002.
URL: *https://doi.org/10.1088/1361-6382/ab685e*

Akaike, H. (1998), Information theory and an extension of the maximum likelihood principle, *in* 'Selected Papers of Hirotugu Akaike', Springer, pp. 199–213.

Allen, B., Hua, W. & Ottewill, A. (1999), 'Automatic cross-talk removal from multi-channel data', *arXiv: General Relativity and Quantum Cosmology* .

Bahaadini, S., Noroozi, V., Rohani, N., Coughlin, S., Zevin, M., Smith, J., Kalogera, V. & Kat-saggelos, A. (2018), 'Machine learning for gravity spy: Glitch classification and dataset', Information Sciences **444**, 172–186.
URL: https://www.sciencedirect.com/science/article/pii/S0020025518301634

Biswas, R., Blackburn, L., Cao, J., Essick, R., Hodge, K. A., Katsavounidis, E., Kim, K., Kim, Y.-M., Le Bigot, E.-O., Lee, C.-H., Oh, J. J., Oh, S. H., Son, E. J., Tao, Y., Vaulin, R. & Wang, X. (2013), 'Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data', *Phys. Rev. D* **88**, 062003.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.88.062003*

Buikema, A. & et al, C. (2020), 'Sensitivity and performance of the advanced ligo detectors in the third observing run', *Phys. Rev. D* **102**, 062003.
URL: *https://link.aps.org/doi/10.1103/
PhysRevD.102.062003*

Burchard, H. G. (1974), 'Splines (with optimal knots) are better', *Applicable Analysis* **3**(4), 309–319.

Chatterji, S., Blackburn, L., Martin, G. & Katsavounidis, E. (2004), 'Multiresolution techniques for the detection of gravitational-wave bursts', *Classical and Quantum Gravity* **21**(20), S1809–S1818.
URL: *https://doi.org/10.1088/0264-9381/21/20/024*

Chatziioannou, K., Cornish, N., Wijngaarden, M. & Littenberg, T. B. (2021), 'Modeling compact binary signals and instrumental glitches in gravitational wave data', *Phys. Rev. D* **103**, 044013.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.103.044013*

Chollet, F. et al. (2018), 'Keras: The python deep learning library', *Astrophysics source code library* pp. ascl–1806.

Cornish, N. J. & Littenberg, T. B. (2015), 'Bayeswave: Bayesian inference for gravitational wave bursts and instrument glitches', *Classical and Quantum Gravity* **32**(13), 135012.
URL: *https://doi.org/10.1088/0264-9381/32/13/135012*

Davis, D., Littenberg, T., Romero-Shaw, I., Millhouse, M., McIver, J., Di Renzo, F. & Ashton, G. (2022), 'Subtracting glitches from gravitational-wave detector data during the third observing run', *arXiv preprint arXiv:2207.03429* .

Davis, D., Massinger, T., Lundgren, A., Driggers, J. C., Urban, A. L. & Nuttall, L. (2019), 'Improving the sensitivity of advanced LIGO using noise subtraction', *Classical and Quantum Gravity* **36**(5), 055011.
URL: *https://doi.org/10.1088/1361-6382/ab01c5*

de Boor, C. (1972), 'On calculating with b-splines', *Journal of Approximation Theory* **6**(1), 50 – 62.

de Boor, C. (2001), *A Practical Guide to Splines (Applied Mathematical Sciences)*, Springer.

Donoho, D. L. & Johnstone, I. M. (1995), 'Adapting to unknown smoothness via wavelet shrinkage', *Journal of the American Statistical Association* **90**(432), 1200–1224.
URL: *https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476626*

Driggers, J. C., Evans, M., Pepper, K. & Adhikari, R. (2012), 'Active noise cancellation in a suspended interferometer', *Review of Scientific Instruments* **83**(2), 024501.
URL: *https://doi.org/10.1063/1.3675891*

Driggers, J. C., Vitale, S., Lundgren, A. P., Evans, M., Kawabe, K., Dwyer, S. E., Izumi, K., Schofield, R. M. S., Effler, A., Sigg, D., Fritschel, P., Drago, M. & et. al. (2019), 'Improving astrophysical parameter estimation via offline noise subtraction for advanced ligo', *Phys. Rev. D* **99**, 042001.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.99.042001*

Gálvez, A. & Iglesias, A. (2011), 'Efficient particle swarm optimization approach for data fitting with free knot b-splines', *Computer-Aided Design* **43**(12), 1683–1692.

George, D. & Huerta, E. (2018*a*), 'Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced ligo data', *Physics Letters B* **778**, 64–70.
URL: *https://www.sciencedirect.com/science/article/pii/S0370269317310390*

George, D. & Huerta, E. A. (2018*b*), 'Deep neural networks to enable real-time multimessenger astrophysics', *Phys. Rev. D* **97**, 044039.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.97.044039*

Hartle, J. B. (2003), *Gravity: An Introduction to Einstein's General Relativity*, illustrate edn, Benjamin Cummings.
URL: *http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/0805386629 http://www.amazon.de/exec/obidos/redirect?tag=citeulike01-21&amp;path=ASIN/0805386629 http://www.amazon.fr/exec/obidos/redirect?tag=citeulike06-21&amp;path=ASIN/0805386629 http://www.amazon.jp/exec/obidos/ASIN/0805386629 http://www.amazon.co.uk/exec/obidos/ASIN/0805386629/citeulike00-21 http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805386629 http://www.worldcat.org/isbn/0805386629 htt*

Hourihane, S., Chatziioannou, K., Wijngaarden, M., Davis, D., Littenberg, T. & Cornish, N. (2022), 'Accurate modeling and mitigation of overlapping signals and glitches in gravitational-wave data', *Phys. Rev. D* **106**, 042006.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.106.042006*

Huo, X., Duncan, M., Levi, O., Buckheit, J., Chen, S., Donoho, D. & Johnstone, I. (2000), 'About wavelab'.
URL: *http://statweb.stanford.edu/~wavelab/Wavelab_850/AboutWaveLab.pdf*

J Aasi, e. a. (2015), 'Advanced LIGO', *Classical and Quantum Gravity* **32**(7), 074001.
URL: *https://doi.org/10.1088/0264-9381/32/7/074001*

Jupp, D. L. (1978), 'Approximation to data by splines with free knots', *SIAM Journal on Numerical Analysis* **15**(2), 328–343.

Kennedy, J. & Eberhart, R. C. (1995), Particle swarm optimization, *in* 'Proceedings of the IEEE International Conference on Neural Networks: Perth, WA, Australia', Vol. 4, IEEE, p. 1942.

Lukin, A. & Todd, J. G. (2006), 'Adaptive time-frequency resolution for analysis and processing of audio', *Journal of The Audio Engineering Society* .

Luo, Z. & Wahba, G. (1997), 'Hybrid adaptive splines', *Journal of the American Statistical Association* **92**(437), 107–116.

Meadors, G. D., Kawabe, K. & Riles, K. (2014), 'Increasing LIGO sensitivity by feedforward subtraction of auxiliary length control noise', *Classical and Quantum Gravity* **31**(10), 105014.
URL: *https://doi.org/10.1088/0264-9381/31/10/105014*

Merritt, J. D., Farr, B., Hur, R., Edelman, B. & Doctor, Z. (2021), 'Transient glitch mitigation in advanced ligo data', *Phys. Rev. D* **104**, 102004.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.104.102004*

Mogushi, K. (2021), 'Reduction of transient noise artifacts in gravitational-wave data using deep learning'.
URL: *https://arxiv.org/abs/2105.10522*

Mohanty, S. D. (2012), 'Particle swarm optimization and regression analysis I', *Astronomical Review* **7**(2), 29–35.

Mohanty, S. D. (2018), *Swarm Intelligence Methods for Statistical Regression*, Chapman and Hall/CRC.

Mohanty, S. D. & Fahnestock, E. (2020), 'Adaptive spline fitting with particle swarm optimization', *Computational Statistics* pp. 1–37.

Mukherjee, S. (2003), 'Median-based noise floor tracker (MNFT): robust estimation of noise floor drifts in interferometric data', *Classical and Quantum Gravity* **20**(17), S925.

Mukherjee, S., Obaid, R. & Matkarimov, B. (2010), 'Classification of glitch waveforms in gravitational wave detector characterization', *Journal of Physics: Conference Series* **243**, 012006.
URL: *https://doi.org/10.1088/1742-6596/243/1/012006*

Mukund, N., Lough, J., Affeldt, C., Bergamin, F., Bisht, A., Brinkmann, M., Kringel, V., Lück, H., Nadji, S., Weinert, M. & Danzmann, K. (2020), 'Bilinear noise subtraction at the geo 600 observatory', *Phys. Rev. D* **101**, 102006.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.101.102006*

Necula, V., Klimenko, S. & Mitselmakher, G. (2012), 'Transient analysis with fast wilson-daubechies time-frequency transform', *Journal of Physics: Conference Series* **363**, 012032.
URL: *https://doi.org/10.1088/1742-6596/363/1/012032*

Ormiston, R., Nguyen, T., Coughlin, M., Adhikari, R. X. & Katsavounidis, E. (2020), 'Noise reduction in gravitational-wave data via deep learning', *Phys. Rev. Research* **2**, 033066.
URL: *https://link.aps.org/doi/10.1103/PhysRevResearch.2.033066*

Pankow, C., Chatziioannou, K., Chase, E. A., Littenberg, T. B., Evans, M., McIver, J., Cornish, N. J., Haster, C.-J., Kanner, J., Raymond, V., Vitale, S. & Zimmerman, A. (2018), 'Mitigation of the instrumental noise transient in gravitational-wave data surrounding gw170817', *Phys. Rev. D* **98**, 084016.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.98.084016*

Payne, E., Hourihane, S., Golomb, J., Udall, R., Davis, D. & Chatziioannou, K. (2022), 'The curious case of GW200129: interplay between spin-precession inference and data-quality issues', *arXiv e-prints* p. arXiv:2206.11932.

Powell, J. (2018), 'Parameter estimation and model selection of gravitational wave signals contaminated by transient detector noise glitches', *Classical and Quantum Gravity* **35**(15), 155017.
URL: *https://doi.org/10.1088/1361-6382/aacf18*

Riles, K. (2013), ‘Gravitational waves: Sources, detectors and searches’, *Progress in Particle and Nuclear Physics* **68**, 1–54.
URL: *https://www.sciencedirect.com/science/article/pii/S0146641012001093*

Robinet, F., Arnaud, N., Leroy, N., Lundgren, A., Macleod, D. & McIver, J. (2020), ‘Omicron: A tool to characterize transient noise in gravitational-wave detectors’, *SoftwareX* **12**, 100620.
URL: *https://www.sciencedirect.com/science/article/pii/S2352711020303332*

Sathyaprakash, B. S. & Schutz, B. F. (2009), ‘Physics, astrophysics and cosmology with gravitational waves’, *Living Reviews in Relativity* **12**(1), 2.
URL: *https://doi.org/10.12942/lrr-2009-2*

Szczepańczyk, M. J., Antelis, J. M., Benjamin, M., Cavaglià, M., Gondek-Rosińska, D., Hansen, T., Klimenko, S., Morales, M. D., Moreno, C., Mukherjee, S., Nurbek, G., Powell, J., Singh, N., Sitmukhambetov, S., Szewczyk, P., Valdez, O., Vedovato, G., Westhouse, J., Zanolin, M. & Zheng, Y. (2021), ‘Detecting and reconstructing gravitational waves from the next galactic core-collapse supernova in the advanced detector era’, *Phys. Rev. D* **104**, 102002.
URL: *https://link.aps.org/doi/10.1103/PhysRevD.104.102002*

Tipping, M. E. & Bishop, C. M. (1999), ‘Probabilistic principal component analysis’, *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **61**(3), 611–622. Full publication date: 1999.
URL: *http://www.jstor.org/stable/2680726*

Tiwari, V., Drago, M., Frolov, V., Klimenko, S., Mitselmakher, G., Necula, V., Prodi, G., Re, V., Salemi, F., Vedovato, G. & Yakushin, I. (2015), ‘Regression of environmental noise in LIGO data’, *Classical and Quantum Gravity* **32**(16), 165014.
URL: *https://doi.org/10.1088/0264-9381/32/16/165014*

Vazsonyi, L. & Davis, D. (2022), ‘Identifying glitches near gravitational-wave events using the q-transform’.
URL: *https://arxiv.org/abs/2208.12338*

Wąs, M., Gouaty, R. & Bonnand, R. (2021), ‘End benches scattered light modelling and subtraction in advanced virgo’, *Classical and Quantum Gravity* **38**(7), 075020.
URL: *https://doi.org/10.1088/1361-6382/abe759*

Wiener, N. (1964), *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, The MIT Press.

Wold, S. (1974), ‘Spline functions in data analysis’, *Technometrics* **16**(1), 1–11.

Yu, H. & Adhikari, R. X. (2021), ‘Nonlinear noise regression in gravitational-wave detectors with convolutional neural networks’, *arXiv preprint arXiv:2111.03295* .

Zevin, M., Coughlin, S., Bahaadini, S., Besler, E., Rohani, N., Allen, S., Cabero, M., Crowston, K., Katsaggelos, A. K., Larson, S. L., Lee, T. K., Lintott, C., Littenberg, T. B., Lundgren, A., Østerlund, C., Smith, J. R., Trouille, L. & Kalogera, V. (2017), ‘Gravity spy: integrating advanced LIGO detector characterization, machine learning, and citizen science’, *Classical and Quantum Gravity* **34**(6), 064003.
URL: *https://doi.org/10.1088/1361-6382/aa5cea*

APPENDIX A

APPENDIX A

GLITCH REMOVAL CODES

The code, that is given here, is an example of glitch removal method by amplitude threshold method. Also, this code is for koi fish glitch.

```
% Script for glitch removal - Koi Fish Glitch
% SHPS runs the SHAPES algorithm on given data.
% addpath 'F:\UTRGV_Spring_2021\Statistical_Methods\SDMBIGDAT19\
    CODES'
% addpath 'D:\MS_Research\SHAPES'
% addpath 'D:\MS_Research\Codes\Edited_codes\Main_codes'
% addpath 'D:\MS_Research\Codes\Edited_codes\Automation_work'

%Input data file (Collect the data from GWOSC, whiten the data,
    and keep it
% in a folder. Add the whitened data here as a input file.)
inDataFile = fullfile('D:','MS_Research','Codes','Edited_codes','
    Conditioned_data','Koi_Fish','H-H1_LOSC_4_V1-1132929024-4096
    _dtrndWhtnBndpss.hdf5');

% Regulator gain
rGain = 0.1;
% Number of knots over which to perform model selection
nKnts = {[5,6],[7,8,9],[10,12,14,16,18],...
```

```
              [20,25,30,35,40]}; %Base set
% Number of independent PSO runs
nRuns = 4;
% PSO parameters
psoParams = struct(...
                    'popsize',40,...
                    'maxSteps', 100,...
                    'c1',2,...
                    'c2',2,...
                    'maxVelocity',0.5,...
                    'startInertia',0.9,...
                    'endInertia', 0.4,...
                    'boundaryCond','',...
                    'nbrhdSz',3);
% Upsampling factor
upSmplMltplr = 2; %Set to [] to have no upsampling

% Nominal Segment length
nSegSmpls = [256,512,1024,2048,4096,8192,16384,32768];%256*
    upSmplMltplr;
% Minimum number of samples in a segment
minNSegSmpls = 35;
% Segment overlap
nOvrlp = 30;
% Decay rate of exponential in overlap-average
dkRate = 16; %End value = exp(-dkRate)
% Window length for averaging the data
```

```
winLen = []; %12; %Set to [] to have no averaging


%% Reading files
% In this case, only '*.mat' and '*.hdf5' file is used. Using
    more 'case',
% we can add any format we want to add her, for example '*.gwf',
    '*.xlsx.',
% '*.csv', etc. The functions 'mattodata' and 'hdftodata' reads
    the data
% from the file.
[fPath, fName, fExt] = fileparts(inDataFile);
switch lower(fExt)
  case '.mat'
    [data1,data2] = mattodata(inDataFile);
  case '.hdf5'
    [data1,data2] = hdftodata(inDataFile);
  otherwise
    error('Unexpected file extension: %s', fExt);
end

%% Measuring standard deviation
%Start and stop time (set to [] for default values of 1 and end
    of data,
%respectively). Use a part of data, where no glitch or major
    change is
%present.
```

```
strtDataTime = [1132931039];
stpDataTime = [1132931044];

% If the start and stop time is blank, then it will continue to
    follow
% these steps. Not recomended for standrad deviation measurement.
if isempty(strtDataTime)
     %idx1  =1;
      strtDataTime =data1(1);
end
if isempty(stpDataTime)
     %idx2 = length(data1);
      stpDataTime = data1(end);
end

% finding the index values of the samples in between the start
    and stop
% time for selecting the data
idx = find(data1>=strtDataTime & data1<=stpDataTime);

%datax = data1(idx); % datax is not needed for measuring standard
      deviation
datay = data2(idx);
std_data = std(datay);

%% Data selection; amplitude threshold selection
```

```
% finding the index values of data for measuring maximum and
    minimum values
% of general/Gaussian noise. Select the quietest part of the data
     for
% measuring these values.
idx = find(data1 >= 1132929200 & data1 <= 1132929250); % for koi
    fish
% data selection using the index values
%datax = data1(idx); % timeseries data is not needed for
    measuring
%threshold
datay = data2(idx);

% finding minimum and maximum of general/gaussian noise
minDatVal = min(datay);
maxDatVal = max(datay);
% creating the threshold values by adding some values t0 it. In
    case of
% scattering cases, the values should be very low, for example,
    0.10 or o.15
maxDatValAdd = maxDatVal + 5;
minDatValAdd = minDatVal - 5;

%Start and stop time (set to [] for default values of 1 and end
    of data,
%respectively). Selection of the data in between the start and
    stop time
```

```
%for getting an estimation.
strtDataTime = [1132929284];
stpDataTime = [1132929292];

% If the start and stop time is blank, then it will continue to
    follow
% these steps. Therefore, it will insert the full data.
if isempty(strtDataTime)
    %idx1 =1;
    strtDataTime =data1(1);
end
if isempty(stpDataTime)
    %idx2 = length(data1);
    stpDataTime = data1(end);
end
% finding the index values of the samples in between the start
    and stop
% time for selecting the data
idx = find(data1>=strtDataTime & data1<=stpDataTime);

% Data selection: estimation will be produced for this data. It
    can be the
% full data or a part of it.
datax = data1(idx);
datay = data2(idx);
datay = datay./std_data;

```

```
%% Signal injection
% % Signal parameters
t_a = datax(1) + 2; % start time of the signal
L = 2.5;            % length of the signal
% frequency, phase and SNR value selection
sgnlParams = struct('Freq1', 30, 'Freq2', 600, 'phase', 0, 'snr',
    25);

% A linear chirp signal is produced using 'atcsmgenltcsig.m'.
sigVec = atcsmgenltcsig(datax,[t_a, t_a + L], sgnlParams.snr,[
   sgnlParams.Freq1,sgnlParams.Freq2], sgnlParams.phase);

% Injecting the signal into data
datay = datay + sigVec;

%% Identifying glitch and dividing the data in sub-segments
% Finding the data that surpass the amplitude threshold value
f = find(datay>maxDatValAdd | datay <minDatValAdd);

% There is a functional approach for this method. 'spartnV2.m' is
    a
% function for identifying glitch. Calling the function for
   finding
% index of glitches
%[n,~] = separtnV2(h); % column numbers can be decided by n+1
%[~, idx1, idx2] = separtnV2(h);

```

```
%finding the difference between the indexes
m = diff(f);

% creating the table for differences
ht = table(f');
m = [0,m];
ht.m = m';
% renaming the variables
ht = renamevars(ht, ["Var1", "m"], ["Index_Values", "Differences
    "]);

% Small 'm' values belong to the same glitch. Higher 'm' values
    are for
% different glitches. Thus, using one parameter we can identify
    different
% glitches. Finding 'm' values that are higher than a threshold
n = find(ht.Differences>100);
idxVal = ht.Index_Values(n);

% Selection of glitch boundary
idxVals1 = ht.Index_Values(1) - 200;
idxVals2 = idxVal - 100;
idxVale1 = ht.Index_Values(n-1)+200;
idxVale2 = ht.Index_Values(end) + 200;

idxVals = [idxVals1;idxVals2];
idxVale = [idxVale1;idxVale2];
```

```
idxValf = [idxVals, idxVale];


% Selection of the data other than the glitch part i.e. general/
    Gaussian
% noise.
 idxValfW1s = find(datax==datax(1));
 idxValfW1e = idxValf(1,1) + 50;
 idxValfW2s = idxValf(1,2) - 50;
 idxValfW2e = find(datax <= datax(end));
 idxValfW2e = idxValfW2e(end);

 idxValWs = [idxValfW1s;idxValfW2s];
 idxValWe = [idxValfW1e;idxValfW2e];
 idxValW = [idxValWs,idxValWe];

% manual data selection for secondary glitch which is visible in
    the data,
% specific for koifish or this data
 idxValfP1s = idxValf(1,1) - 1300;
 idxValfP1e = idxValf(1,1) - 700;


 idxValPs = [idxValfP1s];
 idxValPe = [idxValfP1e];
 idxValP = [idxValPs,idxValPe];

```

```
% Sometimes the data start with a glitch. Thus, the method will
% get some negative values for the index. If the glitch is at the
    end of the
% data, the index will be more than the length of the data. Both
    of these
% cases will produce an error. This problem/bug is fixed in this
    part of the
% code.
if idxValf(1,1) <= 0
    idxValf(1,1) = 1;
elseif idxValf(end,2) > length(datax)
    idxValf(end,2) = length(datax);
end

% estimation parameter for SHAPES. Allocating places for the
    estimation
% will make the process faster.
estSig = zeros(1,length(datay));

%% Segment analysis
% This section will produce the figure for sub-segment, that is
    produced by
% amplitude threshold.
figure;
datax1 = datax-datax(1);
plot(datax1(idxValW(1,1):idxValW(end,2)),datay(idxValW(1,1):
    idxValW(end,2)))
```

```
for i = 1:length(idxValW(:,1))
    rectangle('Position', [datax1(idxValW(i,1)), -20, datax1(
       idxValW(i,2)) - datax1(idxValW(i,1)), 40], 'EdgeColor',
       [1, 0, 0, 0.3]);
end
for i = 1:length(idxValf(:,1))
    rectangle('Position', [datax1(idxValf(i,1)), -25, datax1(
       idxValf(i,2)) - datax1(idxValf(i,1)), 50], 'EdgeColor',
       [0, 0, 1, 0.3]);
end
for i = 1:length(idxValP(:,1))
    rectangle('Position', [datax1(idxValP(i,1)), -15, datax1(
       idxValP(i,2)) - datax1(idxValP(i,1)), 25], 'EdgeColor',
       [0, 1, 0, 0.3]);
end

xlabel('Time (sec)')
ylabel('Whitened GW Strain')

%% Working section of SHAPES : Gaussian noise
% this section will produce the estimation for the general/
    Gaussian noise
for i = 1:length(idxValW(:,1))
    datax1 = datax(idxValW(i,1):idxValW(i,2));
    datay1 = datay(idxValW(i,1):idxValW(i,2));
    estsig = estSig(idxValW(i,1):idxValW(i,2));

```

```
%Moving average
if ~isempty(winLen)
    winVec = blackmanharris(winLen);
    winVec = winVec(:)';
    resdltmp = conv(winVec,datay)/sum(winVec);
    datay = resdltmp(1:length(datay));
end

%Upsampling
if ~isempty(upSmplMltplr)
    smplIntrvl = min(diff(datax1));
    [dataY,dataX] = resample(datay1,datax1,2*upSmplMltplr*(1/
       smplIntrvl),...
                                  upSmplMltplr,1);
    [estsig, ~] = resample(estsig, datax1, 2*upSmplMltplr*(1/
       smplIntrvl),...
                                 upSmplMltplr,1);

end

% Selection of the segment length that we should apply on the
    data
idxSegSmpls = find(nSegSmpls>=length(dataX));
if isempty(idxSegSmpls)
    idxSegSmpls = 1:length(nSegSmpls);
    nSegSmpls(idxSegSmpls(1)) = nSegSmpls(end-1);
end
```

```
    % selection of sub-segment of the upsampled data
    nSamples = length(dataX);
    segBndsStrt = 1:(nSegSmpls(idxSegSmpls(1))-nOvrlp): nSamples;
    segBndsEnd = segBndsStrt+ (nSegSmpls(idxSegSmpls(1)))-1;
    nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    while nSmplsEndSeg < (2*upSmplMltplr*minNSegSmpls)
        segBndsStrt(end) = [];
        segBndsEnd(end) = [];
      nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    end
    segBndsEnd(end) = min(nSamples,segBndsEnd(end));
    segBnds = [segBndsStrt(:), segBndsEnd(:)];

    % Parameters for SHPS
    if ~iscell(nKnts)
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',nKnts,'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    else
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',{nKnts},'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    end


    psoP = struct('nRuns',nRuns,...
                  'psoParams',psoParams);
    % Calling the SHAPES function and getting the estimation
    [estsig, bestMdlResultsAllSeg,allMdlResultsAllSeg] =
       shpslong_core(params, psoP);

    % down-sampling the estimation. It also can be downsampled
       using the
    %technique in upsampling factor.
     estsig = resample(estsig, 1, 4);
    % replacing the zeros of estSig with the estimation made by
       SHAPES
    estSig(idxValW(i,1):idxValW(i,2)) = estsig;
end

%% Working section of SHAPES : Glitch
% this section will produce the estimation for the glitches
for i = 1:length(idxValf(:,1))
    datax1 = datax(idxValf(i,1):idxValf(i,2));
    datay1 = datay(idxValf(i,1):idxValf(i,2));
    estsig = estSig(idxValf(i,1):idxValf(i,2));

    %Moving average
    if ~isempty(winLen)
        winVec = blackmanharris(winLen);
        winVec = winVec(:)';
        resdltmp = conv(winVec,datay)/sum(winVec);
        datay = resdltmp(1:length(datay));
    end

    %Upsampling
    if ~isempty(upSmplMltplr)
        smplIntrvl = min(diff(datax1));
        [dataY,dataX] = resample(datay1,datax1,2*upSmplMltplr*(1/
           smplIntrvl),...
                                     upSmplMltplr,1);
        [estsig, ~] = resample(estsig,datax1,2*upSmplMltplr*(1/
           smplIntrvl),...
                                    upSmplMltplr,1);

    end

    % Selection of the segment length that we should apply on the
        data
    idxSegSmpls = find(nSegSmpls>=length(dataX));
    %mltply_seg = nSegSmpls(idxSegSmpls(1))/nSegSmpls;
    % selection of sub-segment of the upsampled data
    nSamples = length(dataX);
    segBndsStrt = 1:(nSegSmpls(idxSegSmpls(1))-nOvrlp): nSamples;
    segBndsEnd = segBndsStrt+ (nSegSmpls(idxSegSmpls(1)))-1;
    nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    while nSmplsEndSeg < (2*upSmplMltplr*minNSegSmpls)
       segBndsStrt(end) = [];
       segBndsEnd(end) = [];
      nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    end
    segBndsEnd(end) = min(nSamples,segBndsEnd(end));
    segBnds = [segBndsStrt(:), segBndsEnd(:)];

    % Parameters for SHPS
    if ~iscell(nKnts)
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',nKnts,'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    else
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',{nKnts},'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    end


    psoP = struct('nRuns',nRuns,...
                  'psoParams',psoParams);
    % Calling the SHAPES function and getting the estimation
    [estsig, bestMdlResultsAllSeg,allMdlResultsAllSeg] =
        shpslong_core(params, psoP);

    % down-sampling the estimation. It also can be downsampled
        using the
    %technique in upsampling factor.
    estsig = resample(estsig,1,4);
    % replacing the zeros of estSig with the estimation made by
        SHAPES
    estSig(idxValf(i,1):idxValf(i,2)) = estsig;

end

%% Working section of SHAPES : secondary Glitch
% this section will produce the estimation for the glitches. It
    is not
% needed for maximum data. It is uncommon to have this type of
% secondary glitch in the data
for i = 1:length(idxValP(:,1))
    datax1 = datax(idxValP(i,1):idxValP(i,2));
    datay1 = datay(idxValP(i,1):idxValP(i,2));
    estsig = estSig(idxValP(i,1):idxValP(i,2));

    %Moving average
    if ~isempty(winLen)
        winVec = blackmanharris(winLen);
        winVec = winVec(:)';
        resdltmp = conv(winVec,datay)/sum(winVec);
        datay = resdltmp(1:length(datay));
    end

    %Upsampling
    if ~isempty(upSmplMltplr)
        smplIntrvl = min(diff(datax1));
        [dataY,dataX] = resample(datay1,datax1,2*upSmplMltplr*(1/
            smplIntrvl),...
```

```
                                          upSmplMltplr,1);
        [estsig, ~] = resample(estsig,datax1, 2*upSmplMltplr*(1/
            smplIntrvl),...
                                          upSmplMltplr,1);

    end

    % Selection of the segment length that we should apply on the
        data
    idxSegSmpls = find(nSegSmpls>=length(dataX));
    %mltply_seg = nSegSmpls(idxSegSmpls(1))/nSegSmpls;
    % selection of sub-segment of the upsampled data
    nSamples = length(dataX);
    segBndsStrt = 1:(nSegSmpls(idxSegSmpls(1))-nOvrlp): nSamples;
        % 18 for tomte
    segBndsEnd = segBndsStrt+ (nSegSmpls(idxSegSmpls(1)))-1;
    nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    while nSmplsEndSeg < (2*upSmplMltplr*minNSegSmpls)
        segBndsStrt(end) = [];
        segBndsEnd(end) = [];
      nSmplsEndSeg = nSamples-segBndsStrt(end)+1;
    end
    segBndsEnd(end) = min(nSamples,segBndsEnd(end));
    segBnds = [segBndsStrt(:), segBndsEnd(:)];

    % Parameters for SHPS
    if ~iscell(nKnts)
```

```
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',nKnts,'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    else
        params = struct('dataY',dataY(:)','dataX',dataX(:)',...
            'nBrks',{nKnts},'rGain',rGain,...
            'segBnds',segBnds,'dkRate',dkRate);
    end


    psoP = struct('nRuns',nRuns,...
                 'psoParams',psoParams);
    % Calling the SHAPES function and getting the estimation
    [estsig, bestMdlResultsAllSeg,allMdlResultsAllSeg] =
       shpslong_core(params, psoP);

    % down-sampling the estimation. It also can be downsampled
       using the
    % technique in upsampling factor.
    estsig = resample(estsig,1,4);
    % replacing the zeros of estSig with the estimation made by
       SHAPES
    estSig(idxValP(i,1):idxValP(i,2)) = estsig;

end
%% Work section of waveshrink
% Waveshrink input data requirements:
```

```
%     y        1-d  signal.  length(y)= 2^J
%                    Normalized  to  noise  level  1!  (See  NoiseNorm)

%Waveshrink  parameters
% -- Type  of  shrinkage  applied:
%                    'Visu','SURE','Hybrid','MinMax','MAD'
%                    Optional;  default  ==  'Visu'
wvshrnkTypeParam  =  'Hybrid';
%     L        Low-Frequency  cutoff  for  shrinkage  (e.g.  L=4)
%                    Should  have  L  <<  J!
wvshrnkLParamVec  =  1;
%     qmf      Quadrature  Mirror  Filter  for  Wavelet  Transform
%                    Optional,  Default  =  Symmlet  8.

wvshrnkQMFParam  =  [];

% estimation  parameter  for  Waveshrink.  Allocating  places  for  the
    estimation
% will  make  the  process  faster.
allEstSigDat  =  zeros(1,length(datax));  %estimation  on  data
allEstSigEst  =  zeros(1,length(datax));  % estimation  on  SHAPES
    estimation

% waveshrink  on  whitened  data
if  isempty(wvshrnkQMFParam)
        [allEstSigDat,~]  =  WaveShrink(datay,wvshrnkTypeParam,
            wvshrnkLParamVec);
```

```
else
        [allEstSigDat,~] = WaveShrink(datay,wvshrnkTypeParam,
            wvshrnkLParamVec,wvshrnkQMFParam);
end

% waveshrink on estimate of the SHAPES. The combination approach
    of SHAPES
% and waveshrink.
if isempty(wvshrnkQMFParam)
        [allEstSigEst,~] = WaveShrink(estSig,wvshrnkTypeParam,
            wvshrnkLParamVec);
else
        [allEstSigEst,~] = WaveShrink(estSig,wvshrnkTypeParam,
            wvshrnkLParamVec,wvshrnkQMFParam);
end

%% Finding residual from the data
% residual from SHAPES
rsdlS = datay-estSig;

% Residual From Waveshrink
rsdlW = datay-allEstSigDat;

% Residual From the combination approach of SHAPES and Waveshrink
rsdlSW = datay-allEstSigEst;

%% Determining SNR values in residuals
```

```
% SNR value of data
snrShrtSigDatay = datay*(sigVec'/norm(sigVec));
disp(snrShrtSigDatay)

%SNR value of SHAPES residual
snrShrtSigRsdlS = rsdlS*(sigVec'/norm(sigVec));
disp(snrShrtSigRsdlS)

% SNR value of Waveshrink residual
snrShrtSigRsdlW = rsdlW*(sigVec'/norm(sigVec));
disp(snrShrtSigRsdlW)

% SNR value of the residual from the combination approach pf
    SHAPES and
% waveshrink
snrShrtSigRsdlSW = rsdlSW*(sigVec'/norm(sigVec));
disp(snrShrtSigRsdlSW)

% current procedure of aLIGO. gating procedure
tempidx1 = find(datax>= 2.4 & datax<= 2.9);
rsdlM = datay;
rsdlM(tempidx1) = 0;
% gating can also be done only onto the part of the secondary
% noise and glitch boundary decided by amplitude threshold method
    .
% This way we can reduce the percentage of gating but it miss
% miss linear noise in between koi fish and secondary noise.
% for i = 1:length(idxValf(:,1))
%       rsdlM(idxValf(i,1):idxValf(i,2)) = 0;
% end
% for i = 1:length(idxValP(:,1))
%       rsdlM(idxValP(i,1):idxValP(i,2)) = 0;
% end
% SNR value of the data from the zeroing procedure
snrShrtSigRsdlM = rsdlM*(sigVec'/norm(sigVec));
disp(snrShrtSigRsdlM)

%% Plottings for indinidual estimation in timeseries
% changing time series from UTC to s
datax = datax - datax(1);

% plot of data and estimate
figure;
subplot(1,2,1)
plot(datax,datay)
hold on
plot(datax, estSig)
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
legend("Data", "Estimation by SHAPES")
subtitle('Estimate - Shapes')

% Plot of data and residual
```

```
subplot(1,2,2)
plot(datax, datay)
hold on
plot(datax,rsdlS)
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
legend("Data", "Residual")
subtitle("Residual-Shapes")

% plot of data and estimate
figure;
subplot(1,2,1)
plot(datax,datay)
hold on
plot(datax, allEstSigDat)
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
legend("Data", "Estimation by Waveshrink")
subtitle('Estimate - waveshrink')

% Plot of data and residual
subplot(1,2,2)
plot(datax, datay)
hold on
plot(datax,rsdlW)
```

```
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
legend("Data", "Residual")
subtitle("Residual-waveshrink")

% plot of data and estimate
figure;
subplot(1,2,1)
plot(datax,datay)
hold on
plot(datax, allEstSigEst)
%ah=annotation('arrow',[m, .9], [n, 0.9],'Color','r');
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
legend("Data", "Estimation by SHAPES + waveshrink")
subtitle('Estimate - Shapes + waveshrink')

% Plot of data and residual
subplot(1,2,2)
plot(datax, datay)
hold on
plot(datax,rsdlSW)
hold off
xlabel("Time(s)")
ylabel("Strain (arbitrary units)")
```

```
legend("Data", "Residual")
subtitle("Residual- Shapes + waveshrink")
%% Timeseries : for comparison
figure;
subplot(2,2,1)
plot(datax1(tempidx),datay(tempidx), 'Color', [192,192,192]/255,
    'LineWidth', 4)
hold on
plot(datax1(tempidx), allEstSigDat(tempidx), 'r')
plot(datax1(tempidx), estSig(tempidx), 'g')
axis tight;
ylabel("Whitened GW Strain", "FontSize",14)
title("Detector: LIGO Hanford, GPS Time: 1132929284")

subplot(2,2,2)
plot(datax1(tempidx),datay(tempidx), 'Color', [192,192,192]/255,
    'LineWidth', 4)
hold on
plot(datax1(tempidx), allEstSigDat(tempidx), 'r')
plot(datax1(tempidx), estSig(tempidx), 'g')
axis tight;

subplot(2,2,3)
plot(datax1(tempidx),datay(tempidx), 'Color', [192,192,192]/255,
    'LineWidth', 4)
hold on
plot(datax1(tempidx), allEstSigDat(tempidx), 'r')
plot(datax1(tempidx), estSig(tempidx), 'g')
axis tight;
xlabel("Time (sec)", 'FontSize',14)
ylabel("Whitened GW Strain", "FontSize",14)


subplot(2,2,4)
plot(datax1(tempidx),datay(tempidx), 'Color', [192,192,192]/255,
    'LineWidth', 4)
hold on
plot(datax1(tempidx), allEstSigDat(tempidx), 'r')
plot(datax1(tempidx), estSig(tempidx), 'g')
axis tight;
xlabel("Time (sec)", 'FontSize',14)

%% General spectrogram : for comparison
figure;
% index values for making spectrogram around the glitch
tempidx = idxValW(1,1):idxValW(end,2);
subplot(1,4,1)
[Si,Fi,Ti] = spectrogram(datay(tempidx),1024,960,[],4096);
th = 10; % threshold value
Si(abs(Si)<=th) = th;
imagesc(Ti,Fi,log10(abs(Si))); axis xy;colorbar;
ylim([0 1000])
xlabel('Time (sec)', 'FontSize',14);
ylabel('Frequency (Hz)', 'FontSize',14);
```

```



subplot(1,4,2)
[Sr,Fr,Tr] = spectrogram(rsdlS(tempidx),1024,960,[],4096);
Sr(abs(Sr)<=th) = th;
imagesc(Tr,Fr,log10(abs(Sr))); axis xy;colorbar;
ylim([0 1000])
xlabel('Time (sec)','FontSize',14);
set(gca,'YTickLabel',[]);

subplot(1,4,3)
[Sr1,Fr1,Tr1] = spectrogram(rsdlW(tempidx),1024,960,[],4096);
Sr1(abs(Sr1)<=th) = th;
imagesc(Tr1,Fr1,log10(abs(Sr1))); axis xy;colorbar;
ylim([0 1000])
xlabel('Time (sec)','FontSize',14);
set(gca,'YTickLabel',[]);

subplot(1,4,4)
[Sr2,Fr2,Tr2] = spectrogram(rsdlSW(tempidx),1024,960,[],4096);
Sr2(abs(Sr2)<=th) = th;
imagesc(Tr2,Fr2,log10(abs(Sr2))); axis xy;colorbar;
ylim([0 1000])
xlabel('Time (sec)','FontSize',14);
set(gca,'YTickLabel',[]);

%% turncated/adaptive spectrogram
```

```
[Fd_t,Td_t,Pxx_dat_10_50_coarse] = adaptspecgrm(datay(tempidx)
    ,8,[128,256,512,1024],[10,50],4096);
figure;imagesc(Td_t,Fd_t,Pxx_dat_10_50_coarse); axis xy;colorbar;
ylim([0,1000]); xlabel('Time (sec)'); ylabel('Frequency (Hz)');

% thresholding the spectrogram
Pxx_dat_10_50_coarse_thresh = Pxx_dat_10_50_coarse;
thresh = 0.01;
Pxx_dat_10_50_coarse_thresh(Pxx_dat_10_50_coarse_thresh>=thresh)
    = thresh;
figure;imagesc(Td_t,Fd_t,(Pxx_dat_10_50_coarse_thresh)); axis xy;
    colorbar; ylim([0,1000]); xlabel('Time (sec)'); ylabel('
    Frequency (Hz)');



%
[Fr_t,Tr_t,PxxR_res_10_50_coarse] = adaptspecgrm(rsdlS(tempidx)
    ,8,[128,256,512,1024],[10,50],4096);
PxxR_res_10_50_coarse_thresh = PxxR_res_10_50_coarse;
PxxR_res_10_50_coarse_thresh(PxxR_res_10_50_coarse_thresh>=thresh
    ) = thresh;
figure;imagesc(Tr_t,Fr_t,(PxxR_res_10_50_coarse_thresh)); axis xy
    ;colorbar;
ylim([0,1000]); xlabel('Time (sec)'); ylabel('Frequency (Hz)');


```

```
%% figure of timeseries and spectrogram around the glitch
%index for figure
datax1 = datax;
datax1 = datax1 - datax1(tempidx(1));

figure;
subplot(1,3,1)
plot(datax1(tempidx),datay(tempidx))
hold on
plot(datax1(tempidx), estSig(tempidx), 'r')
plot(datax1(tempidx), rsdlS(tempidx), 'b')
hold off
xlabel("Time(sec)", 'FontSize',14)
ylabel("Whitened GW Strain", "FontSize",14)
legend("Data", "Estimation", "Residual")
title("Detector: LIGO Hanford, GPS Time: 1132929284")
ax.Fontsize = 14;


subplot(1,3,2)
imagesc(Td_t,Fd_t,(Pxx_dat_10_50_coarse_thresh)); axis xy;
    colorbar;
ylim([0 1000])
xlabel('Time (sec)', 'FontSize',14);
ylabel('Frequency (Hz)', 'FontSize', 14);
ax.Fontsize = 14;

```

```

subplot(1,3,3)
imagesc(Tr_t,Fr_t,(PxxR_res_10_50_coarse_thresh)); axis xy;
    colorbar;
ylim([0 1000])
xlabel('Time (sec)', 'FontSize', 14);
ax.Fontsize = 14;
set(gca,'YTickLabel',[]);

%% Saving the data
% Put a name in the place of '*'
save *.mat
```

APPENDIX B

APPENDIX B

FUNCTIONS

The functions mentioned here are used for glitch removing algorithms (amplitude threshold and bestfitness-spectrum threshold). Function 'mattodata' and 'hdftodata' read '*.mat' and '*.hdf5' files, respectively, to give the data. Function 'atcsmgenltcsig' generates a linear chirp signal.

```
function [dataX,dataY] = mattodata(inFile)
% Read the data from mat file
% [X,Y] = mattodata(F)
% X is the timeseries and Y is the strain data that will be
    collected from
% the input file F.
% Mohammad Abu Thaher Chowdhury, 08/2021

matFile = load(inFile);

dataX = matFile.dataX;
dataY = matFile.dataY;

end
```

```
function [dataX,dataY] = hdftodata(inFile)
% Read the data from hdf5 file
% [X,Y] = hdftodata(F)
% X is the timeseries and Y is the strain data that will be
    collected from
% the input file F.
%Mohammad Abu Thaher Chowdhury, 08/2021

% Reading strain data from file
dataY = h5read(inFile, '/strain/Strain');
dataY = dataY';

% Reading attributes from file
gpsStart = h5readatt(inFile, '/strain/Strain', 'Xstart');
duration = h5read(inFile, '/meta/Duration');

% Creating time series
dataX = linspace(double(gpsStart), double(gpsStart + duration),
    length(dataY));
end
```

```
function sigVec = atcsmgenltcsig(timeData,timeSteps,snr,freq,
    phase)
% Generate a linear transient chirp signal
% S = ATCSMGENLTCSIG(T,TS,SNR,FQ,P)
% Generates a linear transient chirp signal S. T is the vector of
% time stamps at which the samples of the signal are to be
    computed. TS is
% the vector of time limits [t_a, t-a + L], signal has zero value
     out of
% this interval. SNR is the matched filtering signal-to-noise
```

```
    ratio of S,
% FQ is the vector of frequencies [f_0, f_1], and P is the
    initial phase.
% FQ and TS components that parametrize the phase of the signal:
% f_0 * (t - t_a) + f_1 * (t - t_a).^2 + (phase / (2 * pi).

%Mohammad Abu Thaher Chowdhury, January 2021

idxt = find(timeData>=timeSteps(1) & timeData <= timeSteps(2));
sigVec = zeros(1, length(timeData));
chirp_slope = (freq(2) - freq(1))/(timeSteps(2)-timeSteps(1));
phaseVec = (freq(1) * (timeData(idxt) - timeSteps(1)) + 0.5*
    chirp_slope.* (timeData(idxt) - timeSteps(1)).^2) + (phase /
    (2 * pi));
trigSin = sin(2 * pi * phaseVec);
sigVec(idxt) = snr * trigSin/norm(trigSin);

end
```

BIOGRAPHICAL SKETCH

Mohammad Abu Thaher Chowdhury was born in Sylhet, Bangladesh. After receiving his high school diploma at M. C. College in 2009 he attended Shahjalal University of Science and Technology where he obtained a bachelor's and master's in Physics in 2013 and 2015 respectively. After graduating he spent 5 years teaching at the Universal College Sylhet and BRAC University. Soon after he acquired a Master of Science in Physics at The University of Texas Rio Grande Valley in December 2022. During his schooling, he shared his passion for physics by working as a research assistant, and a teaching assistant at the UTRGV. Outside of academia, he enjoys traveling, food, and video games. He can be contacted at mohammadabuthaher.chowdhury01@utrgv.edu.