Methods in Time Domain Analysis of Gravitational Wave Data

Clive Tomlinson

A Thesis Presented for the Degree of
Doctor of Philosophy in Physics

Department of Physics and Astronomy
University of Sheffield
November 2015

Abstract

Definitive confirmation of the detection of gravitational waves will require strong independent detections from at least two sources which may include the opportunity for astronomical observation of associated electromagnetic and/or neutrino bursts. For these events, the most energetic of which may be of short duration, to be confirmed astronomically it is necessary to pass source location information to astronomical instruments with sufficient low latency to supply steering information to the instruments before fading of any counterpart. Furthermore, the limitations on computing resources for searches, and the likelihood that at least some sources are likely to have waveforms poorly localised in a time-frequency basis, leads us to seek a search method that operates primarily in the time domain.

We present a rapid time domain exponential cross-correlation indicator which is capable of generating event triggers with a latency on the order of signal duration whilst providing source resolution. A complete analysis pipeline was developed and trigger method tested using archived data from the LIGO H1 (Hanford, WA) and L1 (Livingston, LA) detectors.

Standard data preprocessing techniques such as linear predictive filtering were combined with novel line removal methods. The events selected by this algorithm were further processed by novel level 2 signal enhancement and parameterisation algorithms. Detection efficiency and false alarm rate results were generated from mock data challenge injection signals added to data from the S6/VSR3 science run of initial LIGO. Our results indicate that this search algorithm is a good candidate for an unmodelled burst pipeline complementary to the existing codes, and having certain advantages over these codes in terms of latency, computational burden and robustness.

Personal Declaration

I declare that no part of this thesis has been accepted, or is currently being submitted, for any degree or diploma or certificate or any other qualification in this University or elsewhere.

This thesis is the result of my own work unless otherwise stated. Other sources are acknowledged by explicit references. The views expressed are my own.

In Section 4.3 I present a simple derivation of the distribution of the RTCC algorithm output when processing Gaussian distributed data and am confident of its validity. For reference purposes I sought confirmation within the existing mathematical literature but could not find reference to this derivation (or the results, Eqs. 4.10 and 4.11) upon posing the literature search as, for example, 'the distribution of exponentially weighted products of Gaussian random variables'. I therefore tentatively claim this part to be my work.

Acknowledgements

I would like to thank my PhD supervisor Edward Daw and the University of Sheffield for offering this research opportunity. Without the support and friendship of my supervisor this work would not have been possible. I am also grateful to Bruce Allen and Karsten Danzmann for their funding of our visit to the GEO600 detector site and Hartmut Grote, Jonathan Leong, and Jacob Slutsky for their kind hospitality and cooperation.

The development of our pipeline was made substantially easier by the use of several signal processing, data access and management features contained in the DMT, and so would like to thank John Zweitzig and contributors for the creation of this resource. Many thanks to Amber Stuver for her help with the LIGO MDC resource.

Finally, my love and gratitude goes to Loretta Cooper for the many sacrifices she has made in support of this endeavour.

Contents

List of Figures

List of Tables

Chapter 1

Gravitational Waves and Astrophysical Sources

Introduction

All modern observational astronomy relies on the processing of received particles: photons, neutrinos and cosmic rays, emitted, reflected or otherwise influenced by the presence of matter and energy distributions in stars, galaxies, and intergalactic media [1]. Early optical observation and analysis of planetary motion by astronomers such as Kepler led Isaac Newton to his Universal Law of Gravitation (1687), which explains gravitational interaction in terms of forces between matter within an absolute space and time [2]. Almost two hundred years later, Albert Einstein, using a geometrical approach to light and energy, would show in his theory of Special Relativity (SR) that space and time are not the passive stage for physical events assumed in the Newtonian view. SR describes the relationship between physical events as observed in reference frames of constant relative motion and occurring within a four dimensional space-time, a continuum space formed by combining the three spatial dimensions and time. Extending SR to include gravity, Einstein developed the theory of General Relativity (GR) which he had fully formulated and published in 1915. GR is a theory of gravitation in which gravity is no longer interpreted as a force, but is an effect due to the curvature of space-time caused by the presence of matter and energy. Einstein's 1916 paper [3] showed that the field equations of GR predict the existence of gravitational waves (GWs), the propagation of space-time curvature variation caused by the interaction between matter, energy and space-time.

Confidence in the validity of GR comes from high precision experiments which verify many of its predictions. Examples are the gravitational redshift of light (Pound-Rebka-Snider experiment [4]), gravitational lensing through the observed double image of quasar 0957+561 [5], and the Shapiro time delay effect, where light is delayed in its travel time close to a large gravitating body due to space-time curvature [6]. The direct detection of GWs remains as one of the last tests of GR.

The first indirect evidence for the existence of GWs followed from the discovery in 1974 of the first known binary pulsar PSR 1913+16 made by Joseph Taylor and Russell Hulse [1]. Subsequent observation of the rate of orbital decay of this system over a six year period

[1] Hulse and Taylor received the 1993 Nobel Prize in Physics "for the discovery of a new type of pulsar, a discovery that has opened up new possibilities for the study of gravitation".

made by Taylor and collaborators was found to be in excellent agreement with the rate of energy loss predicted by GR via the emission of GWs [7]. Fig. 1 shows the results of extended observation of the periastron (closest approach) time shift over a period totaling three decades, from which the departure from GR theory (solid curve) is calculated to be around 0.2% [8].

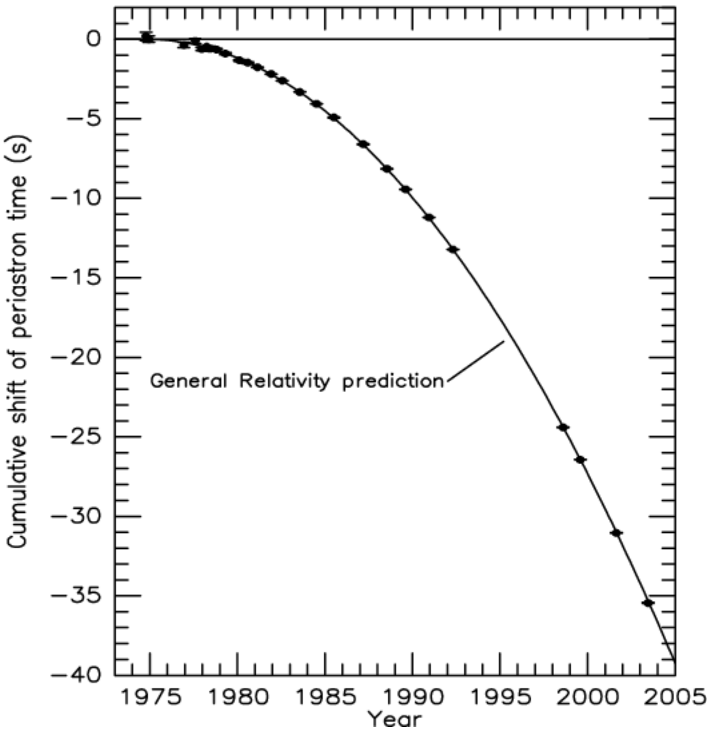


Figure 1.1: Observation of periastron shift of the binary pulsar PSR 1913+16 is plotted along with the GR prediction (solid curve) with energy loss mechanism due to emission of GWs. Deviation from theory was found to be around 0.2%. Reproduced from [8].

We begin with a brief outline of the properties of GWs in Section 1.1, showing that GWs are extremely difficult to detect. Section 1.2 provides an abridged derivation of GWs from linearised GR and in Section 1.3 we review some of the expected astrophysical sources of GWs, the nature of the processes leading to the emission of GWs, and the type of detection analysis required. Finally, Section 1.4 briefly covers the historical developments which led to the current GW detector era.

1.1 Gravitational Waves

The detailed properties of GWs were first derived in Einstein's 1918 paper Über Gravitationswellen [9]. Einstein used a linearised form of the GR field equations to determine that the leading order term contributing to the generation of GWs is the mass quadrupole moment of the system. The quadrupole moment is a measure of the axial asymmetry of the moment of inertia or aspherical distribution. The locally time varying curvature of space-time so produced radiates energy outward at the speed of light, inducing waves of spatial strain in the space traversed. The field strength of GWs was found to be proportional to the second time derivative of the quadrupole moment and inversely proportional to the distance from the source.

GWs, like light, oscillate perpendicularly to the direction of propagation and have two independent polarisations but differ from electromagnetic (EM) radiation in several important respects. EM radiation is dipolar due to the existence of two EM charges whereas GWs result from quadrupolar (at lowest order) motion due to the fact that spherically symmetric motion cannot change the gravitational potential energy. EM waves interact strongly with matter and so are easily absorbed or scattered, making their detection trivial compared to GWs whose coupling with matter is exceedingly weak, and pass through space-time with virtually zero dispersion. For example, the quadrupole formula predicts that a system of two orbiting neutron stars, located at typical distances from Earth, would induce a GW strain minuscule even when compared to nuclear dimensions (see Section 1.3.5). GWs are generated by bulk coherent motions throughout the source in contrast to EM radiation, which is skin deep in the sense that the received signal results from incoherent superposition of emission from the atoms on the outermost surface of the source. For this reason GWs are expected to convey information about the inner dynamics of the source which cannot be obtained through EM observation. Typical wavelengths of visible light are much smaller than the emitting sources and therefore permit both source localisation within the narrow field of view of a single instrument and detailed source images to be constructed. However, the relativistic conditions required for a source of strong GWs place a minimum bound on the wavelength λ_{min} at which emission occurs. The source must be gravitationally compact and so a source of mass M should have minimal radial dimension r close to the Schwarzschild radius r_s. Therefore

$$r \gtrsim r_s = \frac{2GM}{c^2}, \tag{1.1}$$

where G is Newton's gravitational constant and c the speed of light. We should therefore expect λ_{min} to correspond to near relativistic bulk processes occurring around source circumference $\gtrsim 2\pi r$. This gives the estimated minimum wavelength as

$$\lambda_{min} \gtrsim \frac{4\pi GM}{c^2} \sim (10^4 \ \mathrm{m})\frac{M}{M_\odot} \quad \Rightarrow \quad f_{max} \lesssim \frac{c^3}{4\pi GM} \sim (10^4 \ \mathrm{Hz})\frac{M_\odot}{M}. \tag{1.2}$$

When scaled to the Solar mass $M_\odot$ we find the proportionality of λ_{min} to source mass is of order 10^4 m and corresponding maximum frequency f_{max} of order 10^4 Hz. Given that λ_{min} is at least of order the source dimension (Solar $r_s \sim 3 \times 10^3$ m) then GW emission will give poor spatial resolution. The result for f_{max}, in the audio frequency range, provides a useful experimental limit when searching for GW signals.

GWs also differ from EM in that they are non-linear, due to the energy of the wave contributing further to the space-time curvature [10]. These non-linearities complicate the analysis of systems in which very strong gravitational fields are present locally, but in the far field limit, in which GW detection takes place, may be neglected. We now provide an abridged derivation of GWs from the linearised field equations of GR.

1.2 Linearised GR and Gravitational Waves

Before highlighting the main steps involved in showing GW solutions of the Einstein field equations, we first attempt to convey some meaning to the key concepts of GR whilst providing an explanation of the notation in the equations presented. A comprehensive treatment of GR is beyond the scope of this work and for this the reader is referred to the sources on which this section is based, i.e. Gravitation [11] by C.W. Misner, K.S. Thorne and J.A. Wheeler and A First Course in General Relativity [12] by B.F. Schutz.

1.2.1 GR Concepts and Notation

GR is founded upon two principles, the equivalence principle and general covariance. The equivalence principle states that no distinction can be made between a local gravitational field g and, in the absence of gravity, a frame of reference with uniform acceleration $-g$. Thus, in the reference frame of a freely falling body one obtains a local inertial frame (LIF) in which gravity is perceived as absent and which is locally the flat space-time of SR. General covariance is a natural requirement of GR in which the form of physical laws are restricted to those which are valid in all coordinate systems. In mathematical language this means that the quantities (terms) expressed and those resulting from operations performed in GR transform under general coordinate change as tensors.

Without formal definition, we may regard tensors as coordinate independent geometrical objects which may be characterised by their total number of vectorial components (rank). They encompass scalars (rank=0), vectors (1) and more complex objects with multiple vectorial components, each of which is indicated by an index which may take on the range of values corresponding to the dimensions of the relevant space (e.g a Cartesian tensor A_{ij} where $i, j = x, y$ or z). In this notation, the coordinates of space-time points in a LIF may be given by x^μ, where the index μ ranges from 0 to 3. Indices 1 to 3 denote the usual three spatial dimensions (e.g. x, y, z) with x^0 reserved for ct, the product of the speed of light c and time t, giving all coordinates the unit of length. The infinitesimals and partial differentials required for space-time calculus now take the respective forms dx^μ and $\partial/\partial x^\mu = \partial^\mu$.

In general coordinate systems we must make the distinction between two interconvertible forms of vector component (equivalent in Cartesian systems) based upon how they transform with respect to change of the coordinate axes or basis set. Under a rotation of basis the components of a covariant vector transform in the same way as the basis vectors whilst those of a contravariant vector (e.g. displacement vector) transform in a contrary (inverse) manner so as to preserve vector orientation and length. These transformations[1] acting together upon a tensor, which in general may have any number of covariant and contravariant components, maintain the geometric identity of the object. In the following equations, contravariant and covariant components are indicated by upper and lower indices respectively and we indicate the coordinate independence of tensors or covariant operations in boldface.

As a notational convenience Einstein introduced the convention whereby repeated assignment on upper and lower indices on single terms within an expression imply

[1] Covariant and contravariant transformations respectively follow the chain rule for partial derivatives and differentials i.e. $V_i \rightarrow V_i' = (\partial x^j/\partial x^{i'})V_j$ and $V^i \rightarrow V^{i'} = (\partial x^{i'}/\partial x^j)V^j$.

summation over all coordinate index values[2], thus removing the need for explicit representation of the summation sign. The Einstein convention applied to individual tensors, which must be of at least rank 2, is known as the operation of contraction, with the resulting tensor losing 2 orders of rank for each pair of contracted indices[1]. Contraction amongst tensor and vector products is equivalent to the dot product. With this notation we can express in compact form the generalisation of Pythagoras' Theorem known as the space-time interval. This gives the differential line element ds in curved space-time

$$ds^2 = g_{\alpha\beta}\, dx^\alpha dx^\beta. \tag{1.3}$$

The additional ingredient is the metric tensor $g_{\alpha\beta}$, the fundamental geometrical object in GR which encodes the structure of space-time in terms of the local coordinate basis $\vec{e}_\alpha$, the elements of which are given by $g_{\alpha\beta} = \vec{e}_\alpha \cdot \vec{e}_\beta$. In flat space-time one can find a set of coordinates such that the metric tensor is globally identified with the Minkowski metric $\eta_{\alpha\beta}$

$$g_{\alpha\beta} = \eta_{\alpha\beta} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.4}$$

The principles of GR can be applied to a space-time assumed to have the properties of a differentiable manifold, a generally curved and continuous space where finite vectors and derivatives can be defined at all (or nearly all) points. The equivalence principle follows naturally from continuity of the manifold, which permits the construction of local Euclidean coordinate systems at each point in space-time, equivalent to LIFs, much in the same way in which a smooth curved surface can be approximated locally with a sufficiently small flat plane at the single point of contact. In the neighbourhood of each LIF, gravity is then manifest as second order deviations from flatness or space-time curvature. General covariance is somewhat more complex. In the Newtonian description of gravity, the density of matter ρ is the sole source of the gravitational field or potential ϕ (i.e. via $\nabla^2\phi = 4\pi\rho$), but the SR result of mass-energy equivalence means that all forms of energy must contribute, and the covariant source takes the form of the stress-energy tensor $\boldsymbol{T}$. This contains information on all densities of energy, momentum and stress (i.e. pressure). Einstein proposed that the field equations of GR should then be of the form

$$\boldsymbol{G} = \kappa \boldsymbol{T}, \tag{1.5}$$

where $\boldsymbol{G}$ is the Einstein tensor, related to space-time curvature, and κ a proportionality constant. We now illustrate how the application of general covariance to the operation of differentiation, clearly required by a dynamical theory, yields a new form derivative and its important relationship to space-time curvature.

Euclidean space can be defined by a single basis set and so comparison of vectors requires only the partial differentiation of vector components. In curved space-time,

[2] For example, the inner product for Cartesian vectors $\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{3} A_i B^i = A_i B^i$, where the assignment $A^i = A_i$ is valid.

[1] For example contracting A_i^{jk} on the i, k indices $A_i^{jk} \rightarrow A_i^{ji} \rightarrow A^j$.

however, both vector components and basis are position dependent, and the resulting partial derivative is now not covariant. Since, in GR we do not have an independent reference by which we can directly compare vectors in different LIFs, the vector must be transported to the point of comparison whilst maintained parallel to itself in a process known as parallel transport. In the limit that the parallel transported vector is infinitesimally close the differential comparison is the new form of derivative we seek, known as the covariant derivative $\boldsymbol{D}$. In addition to the partial derivative, the covariant derivative includes terms which account for the differential change of basis occurring between infinitesimally separated LIFs. These are given by the Christoffel symbols Γ, defined by

$$\Gamma^{\mu}{}_{\alpha\beta} = \frac{\partial \vec{e_{\alpha}}}{\partial x^{\beta}} \cdot \vec{e_{\mu}}, \tag{1.6}$$

and whilst formally resembling tensors they are coordinate dependant sets of coefficients. The form of the covariant derivative is therefore $\boldsymbol{D} = \partial + \Gamma$, reducing to the partial derivative in flat space (i.e. $\Gamma = 0$).

We are now in a position to make the connection between local space-time curvature, the geometric significance of which we should expect to be characterised by a tensor, and the covariant derivative. Consider parallel transporting a vector V_i which is tangential to the surface of a sphere around the path formed by the quarter circumference sides of a curved triangle covering one octant of the surface as depicted in Fig. 1.2.

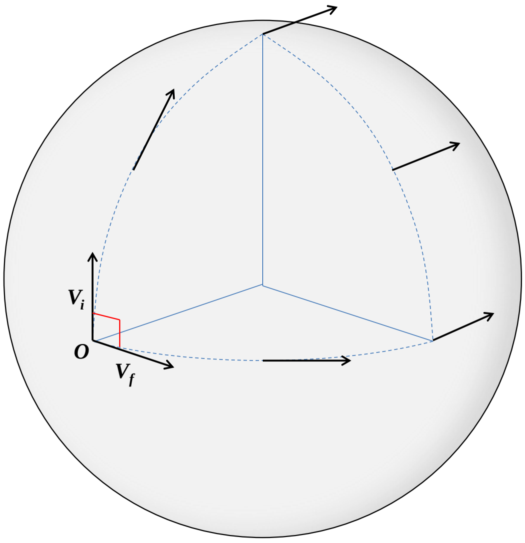


Figure 1.2: Parallel transport of a vector V_i from position O clockwise around the curved triangular path (dotted line) on the surface of a sphere. The final orientation of the transported vector V_f is at 90° with respect to V_i.

On returning to the starting point we will find that the resulting vector V_f has a rotation of 90° (clockwise) with respect to its starting orientation. Curvature induces a change in parallel transported vectors which is in general path dependent. This manifestation of curvature at a point in space-time can be obtained by the parallel transport of a vector around an infinitesimal parallelogram in the directions α and β. This operation gives the commutator of the covariant derivative $[D_\alpha, D_\beta] = D_\alpha D_\beta - D_\beta D_\alpha$ which can be shown to be equivalent to the Riemann curvature tensor

$$R^{\mu}{}_{\nu\rho\sigma} = \partial_\rho \Gamma^{\mu}{}_{\nu\sigma} - \partial_\sigma \Gamma^{\mu}{}_{\nu\rho} + \Gamma^{\mu}{}_{\alpha\rho}\Gamma^{\alpha}{}_{\nu\sigma} - \Gamma^{\mu}{}_{\alpha\sigma}\Gamma^{\alpha}{}_{\nu\rho}, \tag{1.7}$$

which is the standard geometric description of curvature on differentiable manifolds.

In component form the Einstein field equations are

$$G_{\alpha\beta} = \frac{8\pi G}{c^4} T_{\alpha\beta}, \tag{1.8}$$

with κ now determined by correspondence with the Newtonian gravitational field in the non-relativistic limit. The curvature tensor may be contracted to give the Ricci tensor

$$R_{\mu\nu} = R^{\ \alpha}_{\ \mu\alpha\nu} \quad . \tag{1.9}$$

and the Ricci scalar

$$R = g^{\mu\nu} R_{\mu\nu} = R^{\ \nu}_{\ \nu}. \tag{1.10}$$

In terms of these quantities, Einstein found that the requirements of general covariance and local conservation of stress-energy and momentum led to the following form for the Einstein tensor

$$G_{\alpha\beta} = R_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} R. \tag{1.11}$$

The Einstein field equations form a set of non-linear coupled equations which are in general difficult to solve. In the next section we employ the method known as perturbation which enables extraction of the simplest non-trivial solutions to complex sets of equations.

1.2.2 Gravitational Wave Solutions from Linearised GR

We may treat the induced GW strain from a distant and/or weak source of GWs as a small perturbation on an otherwise flat space, i.e.

$$g_{\alpha\beta} = \eta_{\alpha\beta} + h_{\alpha\beta} \quad \text{where} \quad |h_{\alpha\beta}| \ll 1. \tag{1.12}$$

In terms of the perturbed metric, the Christoffel symbols are given by

$$\Gamma^{\rho}{}_{\mu\nu} = \frac{1}{2} g^{\rho\sigma} (\partial_\mu g_{\sigma\nu} + \partial_\nu g_{\sigma\mu} - \partial_\sigma g_{\mu\nu}). \tag{1.13}$$

Note that Eq. 1.13 uses the inverse metric $g^{\alpha\beta} = \eta^{\alpha\beta} - h^{\alpha\beta}$, obtained by the conversion of covariant to contravariant components[1]. For brevity and flow of derivation we omit some of these component raising/lowering manipulations. Now substituting for $g^{\alpha\beta}$ in Eq. 1.13 and neglecting the higher order terms we obtain

$$\Gamma^{\rho}{}_{\mu\nu} = \frac{1}{2} \eta^{\rho\sigma} (\partial_\mu h_{\sigma\nu} + \partial_\nu h_{\sigma\mu} - \partial_\sigma h_{\mu\nu}). \tag{1.14}$$

[1] Consistency of vector representation yields the following relations between contravariant and covariant components $V_\mu = g_{\mu\nu} V^\nu$ and $V^\mu = g^{\mu\nu} V_\nu$.

By inspection of Eq. 1.14, we can see that the Γ square terms of the Riemann tensor only involve terms non-linear in h (i.e. $(\partial h)^2$). Therefore we may first reduce the Riemann tensor to

$$R^{\mu}{}_{\nu\rho\sigma} = \partial_{\rho}\Gamma^{\mu}{}_{\nu\sigma} - \partial_{\sigma}\Gamma^{\mu}{}_{\nu\rho}. \tag{1.15}$$

Insertion of the linearised Christoffel symbols now yields

$$R^{\mu}{}_{\nu\rho\sigma} = \frac{1}{2}(\partial_{\rho}\partial_{\nu}h^{\mu}{}_{\sigma} + \partial^{\mu}\partial_{\sigma}h_{\nu\rho} - \partial^{\mu}\partial_{\rho}h_{\nu\sigma} - \partial_{\sigma}\partial_{\nu}h^{\mu}{}_{\rho}). \tag{1.16}$$

Defining the d‘Alembertian wave operator $\Box = \nabla^2 - \partial_t^2 = \partial^{\mu}\partial_{\mu}$ and the perturbation trace $h = h^{\mu}{}_{\mu}$, upon contraction of the Riemann tensor we obtain the linearised Ricci tensor

$$R_{\nu\sigma} = \frac{1}{2}(\partial_{\mu}\partial_{\nu}h^{\mu}{}_{\sigma} + \partial^{\mu}\partial_{\sigma}h_{\nu\mu} - \Box h_{\nu\sigma} - \partial_{\sigma}\partial_{\nu}h). \tag{1.17}$$

Contraction on the remaining indices now yields the linearised Ricci scalar

$$R = \partial_{\mu}\partial^{\sigma}h^{\mu}{}_{\sigma} - \Box h. \tag{1.18}$$

We may now construct the linearised Einstein tensor

$$G_{\alpha\beta} = \frac{1}{2}(\partial_{\mu}\partial_{\alpha}h^{\mu}{}_{\beta} + \partial^{\mu}\partial_{\beta}h_{\alpha\mu} - \eta_{\alpha\beta}\partial_{\mu}\partial^{\rho}h^{\mu}{}_{\rho} - \Box h_{\alpha\beta} - \partial_{\beta}\partial_{\alpha}h - \eta_{\alpha\beta}\Box h), \tag{1.19}$$

which can be simplified by defining the the trace-reversed perturbation as

$$\bar{h}_{\alpha\beta} = h_{\alpha\beta} - \frac{1}{2}\eta_{\alpha\beta}h \quad \text{where} \quad \bar{h}^{\mu}_{\ \mu} = -h^{\mu}_{\ \mu}. \tag{1.20}$$

This now gives the Einstein tensor

$$G_{\alpha\beta} = \frac{1}{2}(\partial_{\mu}\partial_{\alpha}\bar{h}^{\mu}{}_{\beta} + \partial^{\mu}\partial_{\beta}\bar{h}_{\alpha\mu} - \eta_{\alpha\beta}\partial_{\mu}\partial^{\rho}\bar{h}^{\mu}{}_{\rho} - \Box\bar{h}_{\alpha\beta}). \tag{1.21}$$

The first three terms of Eq. 1.21 all involve contraction of the partial derivative with a component of $\bar{h}_{\alpha\beta}$ and therefore may be eliminated if we are able to set

$$\partial^{\alpha}\bar{h}_{\alpha\beta} = 0. \tag{1.22}$$

This condition, known as the Lorentz gauge, is analogous to that used in electromagnetic theory and it can be shown that selection of this gauge leaves the physical content of the theory unchanged. This now leaves the linearised Einstein field equations as

$$\Box\bar{h}_{\alpha\beta} = -\frac{16\pi G}{c^4}T_{\alpha\beta}. \tag{1.23}$$

In vacuum $T_{\alpha\beta} = 0$ and so we are simply left with the wave equations

$$\Box\bar{h}_{\alpha\beta} = 0. \tag{1.24}$$

1.2.3 Effect of Gravitational Waves on Matter

GWs may be resolved into two independent polarisations h_+ and $h_\times$ whose axes are, unlike light, at 45°. In general, the GW strain $h(t)$ is a linear superposition of both polarisations, the magnitudes of which depend on the orientation of the source in relation to the observer. Fig. 1.3 shows (highly exaggerated) the individual effect of the polarisations of a GW with normal incidence to the plane of a ring of free test particles at quarter phase intervals during one GW cycle. The h_+ component simultaneously increases particle separation on the central vertical axis whilst equally decreasing separation on the horizontal axis. Particle separation on the oblique axes is continuously gradated such that the ring becomes an ellipse, achieving its maximum semi-major axis at $\pi/2$. In the next quarter the deformation proceeds in reverse until regaining circularity at π. Over the remaining half cycle the same deformation now occurs along the reverse axes. The same magnitude $h_\times$ polarisation reproduces the h_+ strain rotated by 45°.

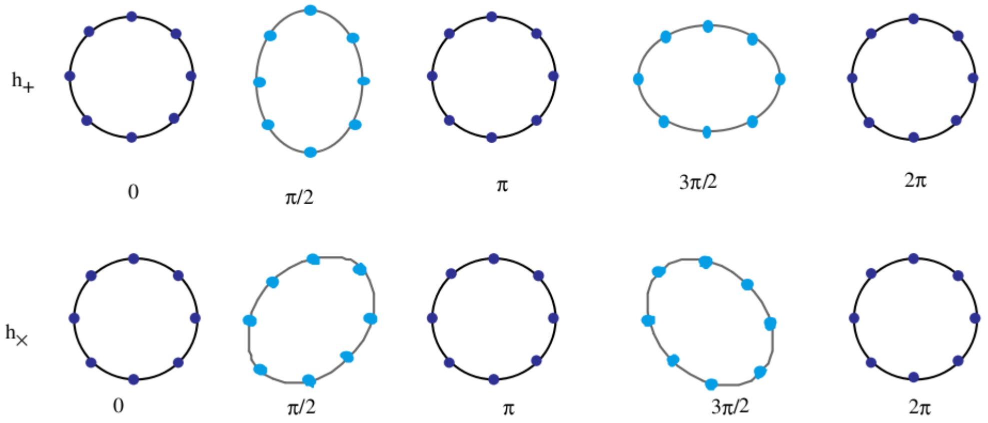


Figure 1.3: Action on the separation of a ring of particles over one cycle of h_+ and $h_\times$ polarisations for GW propagation along ring axis. Reproduced from [13].

1.3 Astrophysical Sources of Gravitational Waves

In this section we review some of the main astrophysical sources of GWs. Except where otherwise stated, this material is based on the relevant sections of Gravitational-Wave Physics and Astronomy, by J.D.E. Creighton and W.G. Anderson [14].

GW sources may be classified by the type of GW signal we expect the source to emit at some point during the evolution of the system and may be categorised as continuous, burst, or stochastic background. Further classification can be made based on the frequency range of GW emission as this determines which type or scale of detector may be sensitive to the source. Depending upon the complexity of GW generation, the waveforms may be mathematically modelled or unknown and this determines the GW data analysis search method for such sources. Compact binary systems, which can produce both continuous and burst signals, whilst contributing to a weak random GW background (stochastic), present the most likely candidate for first detection and so are treated separately.

1.3.1 Continuous Sources

Continuous sources emit GW signals of long duration compared to observation time and are produced by systems undergoing periodic motions with gradual loss of energy. Consequently the GWs emitted are expected to be weak with frequency and amplitude which change very slowly. The frequency of GWs emitted is generally twice the frequency of the GW generation process, which is typically the orbital or rotational frequency of the system.

Rapidly spinning neutron stars (NS) with nonaxisymmetry (pulsars) and compact binary systems far from coalescence are considered sources of continuous GWs. Spherical asymmetry in NSs may be due to surface irregularities such as crustal deformation (mountains), internal fluid oscillation, rotational precession, or slow mass accretion from a companion object. Pulsars have been observed with rotational periods on the order of milliseconds and therefore considered likely sources of high frequency continuous GWs. The strength of GWs emitted by a pulsar is proportional to its ellipticity, which has been estimated to have an upper limit $\epsilon \sim 10^{-6}$ and so substantially weaker by approximately this factor compared to the estimate we obtain in section 1.3.5 for a compact binary system.

Continuous source signals can be modelled to high precision as sine waves of a given frequency f and time evolution $\dot{f}$. Searches for continuous sources permit signal correlation with GW signal models over long periods. This has the advantage that very weak GW signals may be extracted from detector noise with the signal-to-noise ratio increasing as the square root of the observation time. However, the analysis is complicated by the necessary inclusion of source-receiver relative motion due to the Earth's rotation on its axis and orbit around the Sun.

1.3.2 Burst Sources

In contrast to continuous sources, burst sources emit GWs for periods which are short compared to the observation time. These sources are systems or bodies which undergo short-lived catastrophic events. The coalescence of compact binary systems and asymmetric core-collapse supernovae are considered primary burst sources.

Core-collapse supernovae occur when the outward pressures due to nuclear fusion radiation and degeneracy pressure (electron or neutron) within the core can no longer support the star against gravitational collapse. The ensuing highly energetic explosion, classified as either Type II,Ib or Ic supernovae, ultimately leads to neutron star or black hole formation, depending upon the mass of the progenitor and remnant mass distribution.

White dwarfs may undergo accretion induced collapse when its total mass exceeds the Chandrasekhar mass limit of 1.4 solar masses as a result of the transference of matter from a companion star. The thermodynamics of the collapse may lead to nuclear detonation resulting in a Type Ia supernovae event.

The timescale for these events ranges from $\sim$ 1–1000 ms giving these sources an inherent bandwidth of the reciprocal of their duration. As with the compact binary merger phase, these processes are extremely difficult to model, requiring input from magnetohydrodynamics and neutrino physics in addition to GR in order to predict the time varying quadrupole moment of the object. Given these difficulties, the methods used to extract these signals employ unmodelled search methods which make minimal assumptions about the waveform. These highly energetic processes may also be accompanied by strong short-lived bursts of electromagnetic radiation, in particular gamma-rays and/or neutrinos. Combined observations of these counterparts with strong GW detection candidates would provide compelling evidence that GWs had been detected.

1.3.3 Compact Binary Systems

Compact binary systems are considered as the best understood and the most likely sources to be detected with current interferometers. A compact binary system is composed of two gravitationally bound compact stellar mass objects. The objects which form these systems may be any binary combination of neutron star, black hole, or white dwarf.

These systems are characterised by undergoing three stages of evolution. First is the inspiral phase, during which the orbit is quasi-periodic and GW emission is almost monochromatic. As GWs carry away energy from the system, the orbital distance gradually decays and the binary objects spiral in towards each other. This increases the frequency and amplitude at which GWs are emitted and rapidly rises towards the end of the inspiral phase, the waveform of which is referred to as a 'chirp'. This continues until the innermost stable orbit, when phase two, merger, begins. During this phase the coalescence of the two objects begins and an intense burst of GWs is emitted. The coalescence forms an excited Kerr (i.e. rotating) black hole. Stage three is the ringdown phase in which the excited Kerr black hole relaxes by emission of GWs which have waveforms predicted to be exponentially damped sinusoids. The evolution and characterisation of the expected waveform of a compact binary system is depicted in Fig. 1.4.

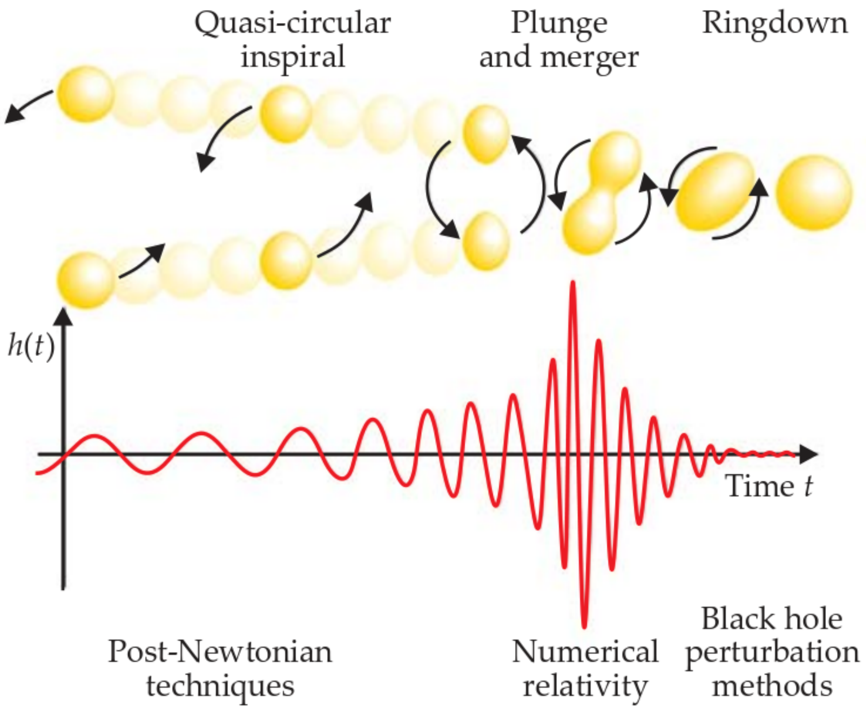


Figure 1.4: Evolution of a compact binary system (top) and expected GW waveform (bottom). Reproduced from [15].

Intense interest in these sources exists because, with some limitations, the waveforms for the inspiral, merger and ringdown phases may be confidently predicted. The complex, highly non-linear, dynamics of the late inspiral and merger phase makes theoretical waveform prediction extremely challenging, employing the techniques of post-Newtonian theory and numerical relativity. Knowledge of these waveforms and the use of matched filter techniques, to be discussed in section 3.3.1, considerably enhance detection probability. Were the inspiral and ringdown waveforms detected, it should then be possible to measure the merger waveforms which may aid the development of better models for coalescence.

White dwarf binaries are expected to emit in a frequency range 10^{-3} – 1 Hz which is far too low to be detectable with ground based detectors but would be within the sensitivity range of detectors such as the proposed space interferometer LISA.

1.3.4 Stochastic Gravitational Wave Background

In addition to independent sources of GWs, a stochastic (random) GW background of astrophysical origin is expected to result from the incoherent superposition of many independent GW sources, intrinsically weak or sufficiently distant such that individual contributions cannot be resolved nor can be localised to a sky position. The independence of these sources are expected to produce a Gaussian distributed random GW background noise, difficult to distinguish from detector noise unless correlated with signals from other detectors over long periods. If this background was successfully detected, this would provide a basis for testing existing astrophysical source population models. Furthermore, underlying the astrophysical background, there may also be a cosmological stochastic background from the epoch of the big bang, when the Universe was 10^{-43} s old.

1.3.5 Estimate of Gravitational Wave Strain from a Compact Binary Source

GW detection poses a technical challenge which can be better appreciated by estimating the measurement precision required to detect a ‘typical’ source. We take the source to be a compact binary system of mass $M \sim 10^{30}$ kg, orbital radius $R \sim 20$ km, orbital frequency $f \sim 400$ Hz, at a distance of $r \sim 10^{23}$ m (3 Mpc). For such a system Einstein's mass quadrupole formula gives

$$h \approx \frac{GMR^2f^2}{c^4r}, \tag{1.25}$$

where G is Newton's gravitation constant, c the speed of light. This gives a strain value of $h \sim 10^{-21}$, corresponding to a change in dimension of $\sim 10^{-6}$ m of a nuclear diameter over 1 m.

One might consider the nearby detection of GWs produced on Earth with a suitable arrangement of rapidly rotating large masses. For example, a dumbbell spinning about the central axis perpendicular to the connecting rod is similar to a binary system. There are two main reasons prohibiting local or remote detection of laboratory generated GWs. Most important is the exceedingly small coupling factor of the dynamics to the GW strain

$G/c^4 \sim 10^{-45}$ N^{-1}. Also, for the radiative character of GWs to be received, detection must take place in the far-field limit, at a distance of at least the GW wavelength, and so $r \sim \lambda_{gw} \sim c/2f$. If we take $h \sim 10^{-21}$ as the order of magnitude of the minimal detectable strain, then our GW generation and detection experiment must now require that $MR^2f^3 \sim 10^{32}$ W. The rotational kinetic energy needed is immense and no practical experiment is conceivable. Therefore the search for GWs must remain an entirely observational endeavour.

1.4 The Search for Gravitational Waves

The search for GWs initially began in the minds of theorists. Although GWs were one of the first predictions of GR and naturally address 'action at a distance' issues, in 1936 Einstein, amongst other relativists, was not convinced that GWs existed. Einstein's conclusions were based on attempts (with Nathan Rosen) to obtain exact solutions for gravitational plane waves but which resulted in singularities that he believed disproved their existence. This debate elicited Arthur Eddington's comment on some GW solutions travelling 'at the speed of thought' [16].

The theoretical basis of GWs as detectable phenomena was established in 1957 at the Chapel Hill Conference (University of North Carolina), held primarily to discuss the relationship between GR, quantum theory and measurement. The close analogy in the derivation of GWs to electrodynamics made the existence and nature of GWs of prime importance. A number of notable contributors, Felix Pirani, Hermann Bondi and (in later correspondence) Richard Feynman, discussed the matter. Pirani contributed a clear physical interpretation of the tidal effects of the Riemann tensor. This led Feynman to propose a thought experiment to determine if a rigid rod threaded with beads which are free to slide (subject to friction) would absorb energy from a passing GW and so dissipate heat to the rod. This argument left the consensus of opinion in favour of GW energy absorption and so provided a physical principle for detection. Also attending was Joseph Weber, who would begin designing the first GW experiment a year after Chapel Hill [17].

The detection principle of Weber's experiment was the measurement of thermal energy absorbed from a GW by resonance of a large aluminium cylindrical bar. His resonant bar detectors were designed to have dimensions which give a sharp resonance at around 1660 Hz, a frequency of GW emission expected to occur during supernova collapse, and achieved a sensitivity to strain of $\sim 10^{-16}$. In a 1969 paper [18], Weber reported coincident GW detection between detectors separated by ∼1000 km, but his results were controversial on energetic grounds and not reproduced by other experimenters. Weber defended his results but by 1975 a consensus had formed that the significance of his reported detections were the result of statistical errors [19]. Although this was not an auspicious beginning for GW astronomy, Weber is generally considered to have pioneered the field of GW experimental methods and his contribution is recognised in the annual Joseph Weber Award for Astronomical Instrumentation, awarded from 2002. Enhanced resonant detectors such as AURIGA are still in operation [20]. Fig. 1.5 shows Weber working on one of his detectors.

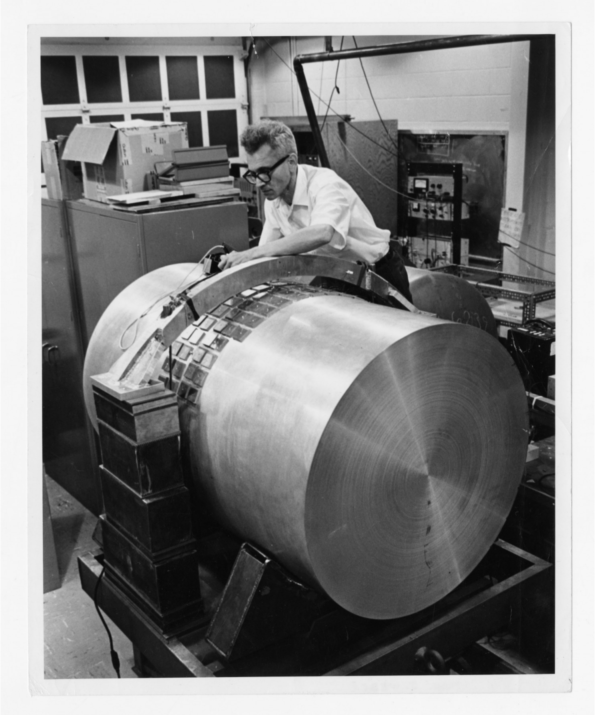


Figure 1.5: Joseph Weber working on one of his resonant bar detectors.
Reproduced from Special Collections University Archives, University of Maryland Libraries.

One major disadvantage of resonant detection is that high sensitivity in a very narrow frequency range, apart from severely limiting which sources could be detected, cannot provide the full waveform description required to characterise the dynamics of the GW source. A detector with high sensitivity over a broad range of frequency is therefore desirable. Broadband strain measurement can be achieved with the detection principle of differential length measurement of the separation of test masses based upon the Michelson interferometer configuration.

The first suggestion of GW detection by interferometry was made by the Russian physicists Michael Gertsenshtein and Vladislav Pustovoit in 1962, in a critique of Weber's bar detection method [21]. Weber's former doctoral student Robert Forward, who had worked with him on the development of resonant bar detectors, was the first to construct a prototype laboratory scale laser interferometric detector at Hughes Research Labs in 1972 [22]. As we will see in Section 2.2, the intrinsic sensitivity to GW strain scales in proportion to the effective test mass separation. Consequently, all prototype interferometers of metre scale constructed by a number of groups during the 1970s and 80s (MIT (Weiss - see below), University of Glasgow [24], and Max Planck Institute (Munich) [25]) were not large enough to detect GWs but these would be of considerable value in developing the technologies necessary for interferometers of much larger scale.

The first detailed study of large-scale GW interferometry was set out by Rainer Weiss in an MIT laboratory report (unpublished) of 1972 [23], in which an extensive analysis of the noise sources limiting detection sensitivity is presented. The 1983 National Science Foundation (NSF) funded study, authored by Paul Linsay, Peter Saulson, and Weiss, comprehensively states the case for the building of (at least) two kilometre-scale interferometers. The document cites the compelling evidence from Taylor and Weisberg's 1982 results on the observation of PSR 1913+16 that GWs exist and states that the primary factor in achieving realistic prospects of GW detection is detector scale rather than the need for technological innovation as motivation for such an endeavour. Now known as the 'blue book', the report also included the practical considerations of potential sites, construction and expense of such a project [26]. In 1984, on the basis of the feasibility of GW detection presented in the blue book, the National Science Board (NSB) endorsed the formation of a joint Caltech/MIT project which came to be known as LIGO (Laser Interferometer Gravitational-Wave Observatory), initially headed by Ronald Drever (Caltech, formerly Glasgow), Weiss and theorist Kip Thorne (Caltech). In 1990, following organisational restructuring with LIGO now under the direction of Rochus Vogt, the NSB accepted the 1989 proposal to build the LIGO observatory: two 4 km GW interferometers at widely separated locations within the US and a third 2 km collocated detector. The proposal envisaged both incremental evolution of the detectors from the initial design by future enhancements anticipated through knowledge gained during operation of the detectors and advanced detectors, essentially new installations incorporating new technical advances which offer substantial improvements in sensitivity [27]. Similar proposals towards establishing large scale GW interferometry were taking place in Europe leading to the joint French-Italian VIRGO and British-German GEO600 projects, and in Japan with the TAMA300 project. In the following, we list a selection of the major developments leading to the first generation of large scale GW detectors based upon an NSF funding report for LIGO [28], a status report for VIRGO [29], and descriptions of the GEO600 [30] and TAMA300 [31] projects.

- **1992** Hanford (Washington) and Livingston (Louisiana) sites selected for LIGO installation.
- **1994-95** Barry Barish (Caltech) appointed as LIGO director and construction begins at Hanford and Livingston. Construction of the 600m GEO600 GW detector in Hannover (Germany) begins. TAMA300 project starts.
- **1997** Formation of the LIGO Scientific Collaboration (LSC), extending LIGO collaboration to GW research groups worldwide.
- **1999-2000** TAMA300 detector (Tokyo) operational and completes first data taking.
- **2002-03** All three LIGO detectors and GEO600 begin scientific operation. Commissioning of newly built 3 km VIRGO detector in Cascina (Italy).
- **2007** Collaborative agreement between LSC and VIRGO project. VIRGO detector starts scientific operation.

Figs. 1.6 and 1.7 show the 4 km detectors sited at Livingston (Louisiana), labeled L1, and at Hanford (Washington state) H1, which houses the third 2 km detector H2. As a collective, these are regarded as the LIGO observatory.

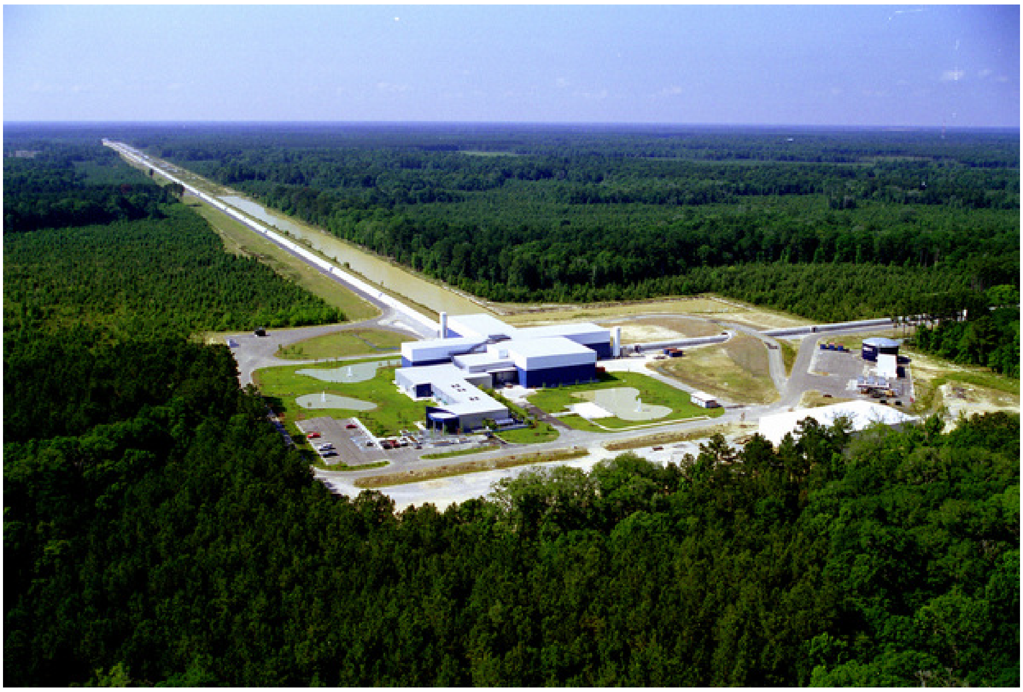


Figure 1.6: LIGO Livingston site (May 19, 2015). Photo credit Caltech/MIT/LIGO Lab (ID ligo20150731c).

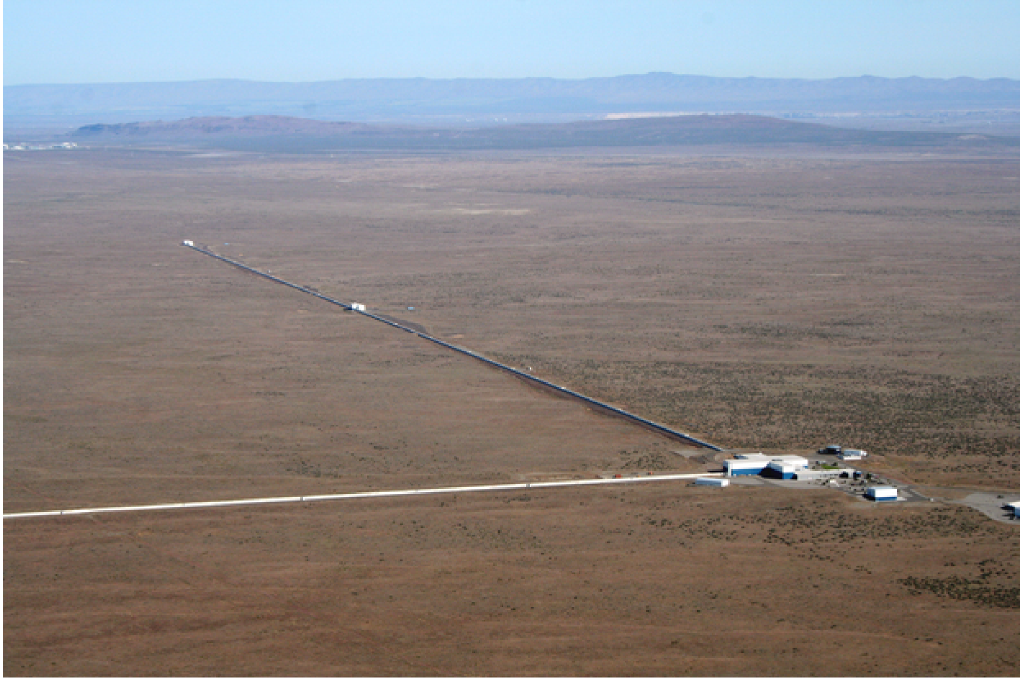


Figure 1.7: LIGO Hanford site (May 2, 2008). Photo credit Caltech/MIT/LIGO Lab (ID ligo20150731f).

Chapter 2

Laser Interferometric Detectors

Introduction

All present GW interferometric detectors are highly evolved enhancements of the Michelson interferometer used by Albert Michelson and Edward Morley in their famous experiments to test for the presence of a 'luminiferous aether' [32]. From this point on, when referring to GW detectors, we mean just this type of detector. GW detectors have each of their mirrors (test masses) suspended as a pendulum, and at frequencies well above the pendulum resonance ($\sim$ 1 Hz) they are free to move along the optical axes in response to a GW. All interferometer optical paths are maintained at high vacuum for acoustic isolation and to reduce beam scattering and variation of refractive index. The majority of content in this chapter is based on Interferometer Techniques for Gravitational-Wave Detection by A. Friese [33] and Gravitational Waves Volume 1 by M. Maggiore [34].

2.1 The Michelson Interferometer

The principle of interferometric length measurement may be illustrated using a simple Michelson interferometer as shown in Fig. 2.1. The input light is provided by a stable Nd:YAG laser emitting at an amplitude ε_0 and wavelength λ=1.064 μm. The beam is incident on a 50:50 beam splitter (BS) which transmits half of the light intensity along the x-axis and reflects the other half along the y-axis. The beams traverse the interferometer arm length distances L_x and L_y before encountering highly reflecting mirrors M_x and M_y which return the light to the BS. On return to the BS, each beam has acquired a phase shift[1] of $\Delta\phi = 2\pi \cdot (2L/\lambda)$ due to the round trip travel time for the given arm length L. The returning beams are each transmitted and reflected by the BS along paths towards the photodiode (PD), which measures the output light intensity, and towards the laser where it exits the interferometer. The PD receives the sum of the transmitted beam returning from M_y and the reflected component from M_x. Overall both beams have been transmitted and reflected by the BS, which introduces a factor of 1/2 in amplitude and a relative phase shift of π due to reflection from opposite sides of the BS.

[1] The common phase factor of π upon reflection from the end mirrors can be ignored as these disappear upon measurement of the field modulus by the PD.

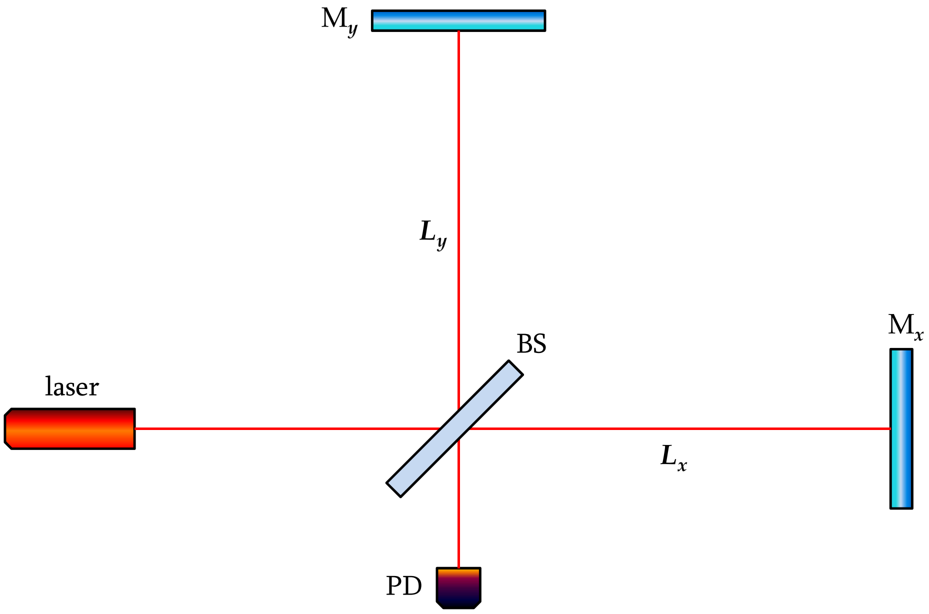


Figure 2.1: Simple Michelson Interferometer. Interferometer components lie in the horizontal plane. A 50:50 beamsplitter (BS) directs the laser light along two perpendicular paths towards the y-arm mirror M$_y$ at distance L_y, and the x-arm mirror M$_x$ at distance L_x. The photodiode (PD) receives the combined components of the light returning from M$_x$ and M$_y$ which respectively have been reflected and transmitted by the BS.

The resulting output field amplitude to the photodiode E_A is antisymmetric with respect to the arm lengths and is given by

$$E_A = \frac{\varepsilon_0}{2}(e^{i2kL_y} - e^{i2kL_x}), \tag{2.1}$$

where the wavenumber $k = 2\pi/\lambda$. If we define the common arm length and differential arm length respectively as

$$\bar{L} = \frac{L_x + L_y}{2} \quad \text{and} \quad \Delta L_d = L_x - L_y, \tag{2.2}$$

then expressing L_x and L_y in terms of these quantities in Eq. 2.1 yields

$$E_A = i\varepsilon_0 e^{i2k\bar{L}} \sin{(k\Delta L_d)}. \tag{2.3}$$

Therefore the output intensity $|E_A|^2$ is proportional to $\sin^2(k\Delta L_d)$.

Now consider the effect of a passing GW with normal incidence to the plane of an interferometer whose arm lengths are L. For simplicity, we consider only the h_+ component and assume the polarisation axes to be parallel with the arms of the interferometer. The GW will produce time dependent separations between the beam splitter and end mirror in each arm, undergoing oscillations similar to those shown in Fig. 1.3. If one arm is lengthened by ΔL then the other is shortened by the same magnitude. Therefore $\Delta L_d = 2\Delta L$ and the output power is now proportional to $\sin^2(2k\Delta L)$. Similarly obtained, or via conservation of energy ($|E_A|^2 + |E_S|^2 = \varepsilon_0^2$), is the symmetric output towards the laser $E_S = \varepsilon_0 e^{i2k\bar{L}} \cos{(2k\Delta L)}$, which shows that in the absence of a GW all laser power is reflected back towards the laser.

Although we require the arm lengths $L_x \approx L_y$ so that the GW strain induced in the arms is of the same magnitude, they do not need to be exactly the same since the output is identical modulo λ and so satisfied when $|L_x - L_y| = n\lambda$ for integer n. This freedom is referred to as the Schnupp asymmetry and will be of use when we consider an optimal GW detection strategy to be described in Section 2.3.2.

2.2 Gravitational Wave Interferometry

In this section we motivate the development of the simple Michelson interferometer towards a GW detector capable of measuring strain amplitudes of order $h \sim 10^{-21}$ with a few simple calculations that will give some insight into the technical and practical issues to be addressed in the design of GW detectors.

Since the output of the interferometer is a measurement of $\Delta L = hL$, then we can improve sensitivity to h by making L as large as practically possible. For a GW of a given frequency, the optimal response of the detector occurs when the time taken for the light to travel from the BS to the end mirrors and back is equal to half the period of the GW. After this time the GW changes sign, producing cancellation which degrades the output signal. For example, to optimally detect GWs of frequency 100 Hz we require arm lengths of $L \sim \lambda_{GW}/4 = c/4f_{gw} \sim 750$ km. The provision of a detector site on this scale (squared) is clearly impractical for ground based interferometers. The simplest method to achieve much longer optical paths within each arm is to reflect the beam multiple times along separate paths between an additional mirror and the end mirror before the light returns to the BS. The disadvantage of this method is that very large mirrors would be required to accommodate the hundreds of paths required for a detector on the more realistic scale of a few kilometres. The more compact solution, first suggested by Ronald Drever and adopted into the design of all subsequent GW detectors, is to make each arm a Fabry-Perot cavity. This important modification is discussed in Section 2.3.1.

Regardless of the optical path configuration, the interferometer must measure a phase difference of order $\Delta\phi \sim 2\pi Lh/\lambda = 2\pi(750 \times 10^3\,\mathrm{m})(1 \times 10^{-21})/(1 \times 10^{-6}\,\mathrm{m}) \sim 10^{-9}$ rad. The intensity measured at the output is a photon average, but associated with this is an error due to fundamental fluctuations in photon number known as shot noise. The uncertainty in photon number N for coherent state[1] light is given by $\Delta N = \sqrt{N}$. This introduces an uncertainty in the energy

$$\Delta E = \Delta N \hbar\omega = \sqrt{N}\hbar\omega, \tag{2.4}$$

where ω is the angular frequency of the light and $2\pi\hbar$ is Planck's constant. The Heisenberg Uncertainty Principle in the energy and time form $\Delta E \Delta t \geq \hbar$ can be used to relate the uncertainty in the phase $\Delta\phi = \omega\Delta t$ to ΔN, to give

$$\Delta\phi \geq \frac{1}{\sqrt{N}} \quad \Rightarrow \quad N \geq \frac{1}{|\Delta\phi|^2} = 10^{18}. \tag{2.5}$$

The laser must then supply N photons in a time half the period of the GW, therefore the minimum power needed is $P = 2N\hbar\omega f_{\mathrm{GW}} \sim 30$ W. At the time the initial LIGO detectors were being designed, stable laser power was limited to ~ 10 W and to overcome

[1] Waveform is that of a classical harmonic oscillator, or far from quantum behaviour.

this deficit it was suggested, again by Drever, that the light which is lost by reflection back through the symmetric output be reused in a method known as power recycling. This is accomplished by means of an additional mirror situated between the laser and the BS, which is highly transmissive to the input laser light and highly reflective to the light received from the symmetric output.

2.3 Detector Optical Techniques

The design of a GW detector of practical proportion and required sensitivity relies upon a number of optical techniques which reduce the extreme values of arm length and laser power that otherwise would be needed. We begin this section by considering a simple model of the Fabry-Perot Cavity. In the following treatment of light propagation we neglect transverse beam diffraction, but in practice the beam wavefront must be profiled by lenses and mirror curvature to confine the maximum beam diameter within the arm cavity.

2.3.1 The Fabry-Perot Cavity

A simple modification of the interferometer arms can increase the effective arm length considerably, allowing much shorter arms. Originally proposed by Ronald Drever [35], this was the inclusion of one partially reflective mirror in each arm, close to the BS, to form Fabry-Perot cavities between the additional and end mirrors.

The Fabry-Perot cavity is a two mirror laser resonator which is highly sensitive to both the frequency of input light and the mirror separation. We can see how light is stored in the cavity by modelling the cavity with two mirrors M_1 and M_2 with reflection and transmission coefficients r_1, t_1 and r_2, t_2 respectively. The squared values of the mirror coefficients are proportional to the probability of photon reflection and transmission at the mirror boundary. Optical losses due to scattering and mirror absorption are ignored in this discussion, therefore $r^2 + t^2 = 1$ for each mirror. Laser light of wavelength λ and field amplitude ε_0 is incident on M_1 at the origin of plane wave propagation and M_2 is the highly reflecting end mirror of an interferometer arm with $r_2 \sim 1$ and $t_2 \sim 0$, separated from M_1 by the arm length distance L. The cavity model along with the steady state circulating and reflected field amplitudes are shown in Fig. 2.2.

The circulating field E_{circ} and corresponding power P_{circ} can be determined from the field amplitudes a_i at the mirror surfaces as follows:

$$a_4 = a_3 e^{-ikL} = a_2 r_2 e^{-ikL} = a_1 r_2 e^{-i2kL} = r_2 e^{-i2kL}(r_1 a_4 + i t_1 \varepsilon_0), \tag{2.6}$$

and $E_{\mathrm{circ}} = i t_1 \varepsilon_0 + r_1 a_4$ which gives

$$E_{\mathrm{circ}} = \varepsilon_0 \frac{i t_1 e^{-i2kL}}{1 - r_1 r_2 e^{-i2kL}} \quad \Rightarrow \quad P_{\mathrm{circ}} = |\varepsilon_0|^2 \frac{t_1^2}{1 + r_1^2 r_2^2 - 2 r_1 r_2 \cos(2kL)}. \tag{2.7}$$

Resonance occurs when the length of the cavity is an integer multiple of $\lambda/2$ (i.e. $kL = n\pi$) and when $r_1 \sim 1$ the circulating power in the cavity can become very large. The ratio $P_{\mathrm{circ}}/|\varepsilon_0|^2$ is defined as the cavity gain and at resonance given by

$$\frac{P_{\text{circ}}}{|\varepsilon_0|^2} = \frac{t_1^2}{(1 - r_1 r_2)^2}. \tag{2.8}$$

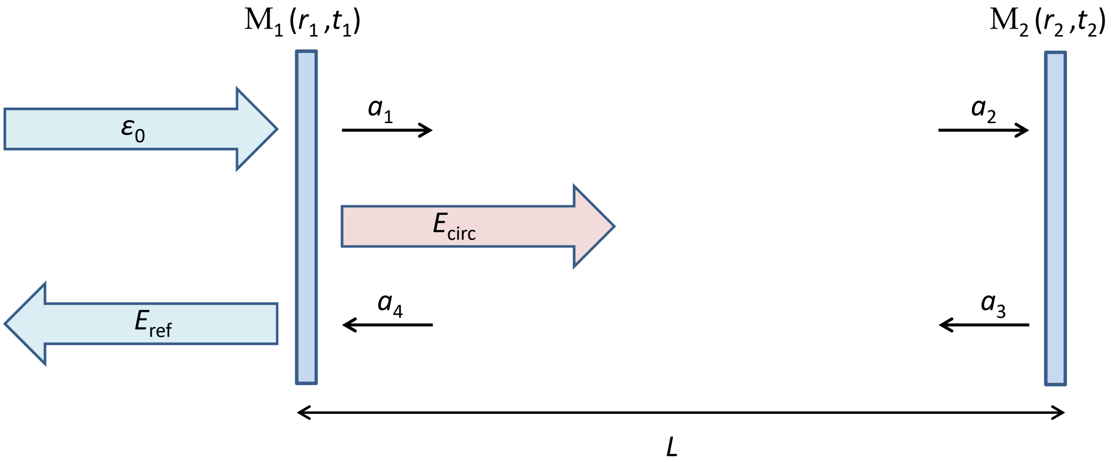


Figure 2.2: Simple model of a Fabry-Perot interferometer arm cavity of length L. M_1 is situated close to the BS and M_2 is the end mirror. For incident light of amplitude ε_0, the circulating and reflected fields E_{circ} and E_{ref} are determined from the mirror boundary amplitudes a_i.

In Fig. 2.3 we show that when the frequency of light is gradually increased (or decreased) from resonance the power inside the cavity falls to a minimum at anti-resonance and new resonances occur at integer values of the free spectral range (FSR) given by FSR=$c/2L$. We can equally regard Fig. 2.3 as showing how stored power changes with cavity length by replacing the frequency axis values $nc/2L$ with $n\lambda/2$.

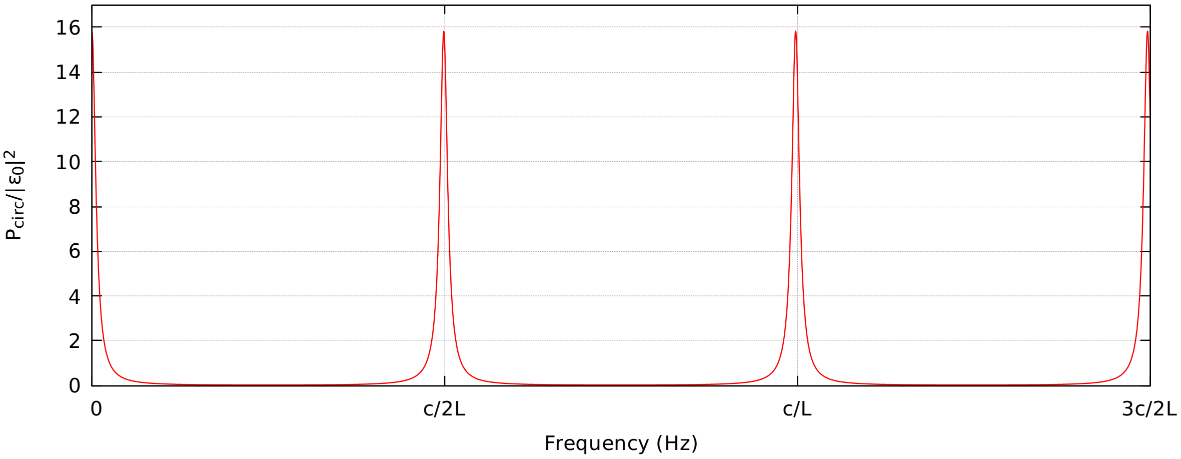


Figure 2.3: Ratio of circulating power to input power as a function of frequency.

The sharpness of resonance is characterised by the finesse $\mathcal{F}$ of the cavity, defined as the ratio of FSR and the full width at half maximum. For a high finesse cavity, with $r_1 \sim r_2 \sim 1$, this can be approximated as

$$\mathcal{F} \approx \frac{\pi\sqrt{r_1 r_2}}{1 - r_1 r_2} \approx \begin{cases} \dfrac{\pi}{2(1-r_1)} & \text{when} \quad r_1 = r_2. \\ \\ \dfrac{\pi}{1-r_1} & \text{when} \quad r_1 < r_2 \simeq 1. \end{cases} \tag{2.9}$$

The first limit applies to the condition known as critical coupling, which is shown below to be of no practical use for GW interferometry. A third limit is obtained from the second by interchange of the coefficients (i.e. $r_1 \rightleftarrows r_2$) and we also show this to be suboptimal.

As with the simple Michelson interferometer, measurement of GW strain depends upon the (average) phase acquired by photons during transit within the Fabry-Perot cavities. The average number of round trips photons make before exiting the cavity is a statistical quantity which is proportional to the cavity finesse. If we take $r_1 \sim 1$ and $r_2 = 1$, on each round trip an initial photon number N is depleted by a factor $(1 - r_1^2)$ in time $2L/c$. Therefore

$$\frac{dN}{dt} = -\frac{c(1-r_1^2)}{2L}N, \tag{2.10}$$

which gives the average lifetime of the photon in the cavity τ as

$$\tau = \frac{2L}{c(1-r_1^2)} \;\Rightarrow\; \text{No. of round trips} = \frac{c\tau}{2L} = \frac{1}{1-r_1^2} \approx \frac{1}{2(1-r_1)}. \tag{2.11}$$

Comparing this with Eq. 2.9 for the case ($r_1 < r_2 \simeq 1$) implies the approximate relation $\mathcal{F} \approx 2\pi\times$(No. of round trips).

Phase information is carried in the reflected field E_{ref}, which is the sum of the reflected incident field and transmitted circulating field at M1, and is given by

$$E_{\text{ref}} = \varepsilon_0 \frac{r_1 - r_2 e^{i2kL}}{1 - r_1 r_2 e^{i2kL}}. \tag{2.12}$$

Considering the numerator of Eq. 2.12, the relative values of the reflection coefficients determine which of three distinct conditions apply, $r_1 = r_2$, $r_1 < r_2$ or $r_1 > r_2$, referred to as critical coupling, overcoupling and undercoupling respectively. For GW detection or control purposes we require a phase response around resonance which is sensitive to change in cavity length or laser frequency and so critical coupling (i.e. $E_{\text{ref}} = 0$) is clearly of no value. Expressing the reflected field in the form $E_{\text{ref}} = |E_{\text{ref}}|e^{i\phi}$, the tangent of the phase ϕ is found to be

$$\tan(\phi) = \frac{r_2(r_1^2 - 1)\sin(2kL)}{r_1(r_2^2+1) - r_2(r_1^2+1)\cos(2kL)}. \tag{2.13}$$

We characterise the phase response $\phi(2kL)$ for instances of overcoupling ($r_1 = 0.95$ $r_2 = 0.99$) and undercoupling ($r_1 = 0.99$ $r_2 = 0.95$) in Fig. 2.4. In both cases the phase response is virtually zero over a wide range around anti-resonance but significant differences can be seen on approaching resonance. Compared to undercoupling, the overcoupling phase response is prominent, almost linear near resonance and monotonic between resonances. Overcoupled cavities are therefore the optimal choice for interferometer arms.

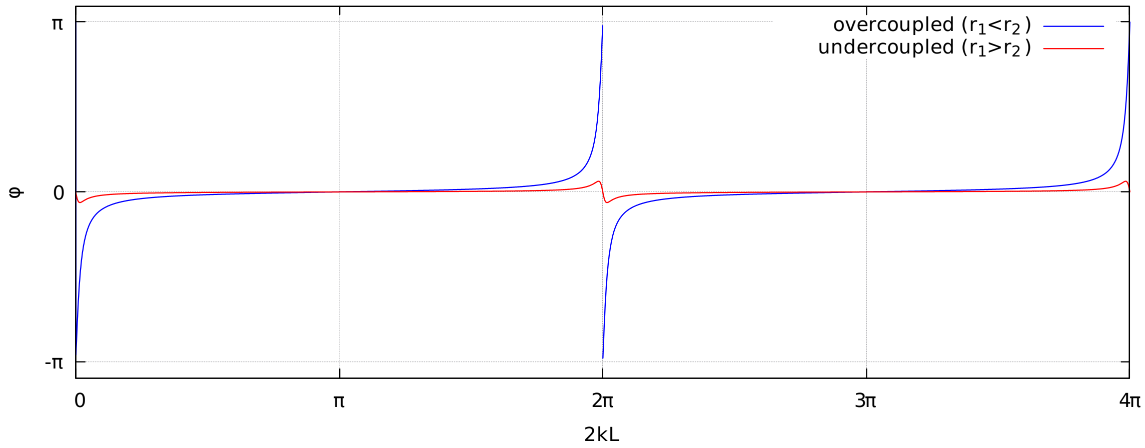


Figure 2.4: Phase response $\phi(2kL)$ of overcoupled ($r_1 = 0.95$ $r_2 = 0.99$) and undercoupled ($r_1 = 0.99$ $r_2 = 0.95$) Fabry-Perot cavities.

We now estimate the linear phase response for an overcoupled cavity near resonance. The gradient of the phase can be obtained by expanding Eq. 2.13, where $\phi \to \phi_{res} + \delta\phi$ and $2kL \to 2n\pi + \delta(2kL)$, to give

$$\frac{d\phi}{d(2kL)} = \frac{(1+r_1)^2}{1-r_1^2} \sim \frac{4}{1-r_1^2} \sim \frac{2\mathcal{F}}{\pi}. \tag{2.14}$$

Thus, compared to the simple Michelson interferometer with phase response $\Delta\phi_{\mathrm{M}} = 2kLh$, high finesse cavities give a large amplification factor of $2\mathcal{F}/\pi$ in phase sensitivity. The magnitude of the total phase change in a Fabry-Perot interferometer is therefore

$$\Delta\phi_{\mathrm{FP}} = \frac{4\mathcal{F}}{\pi}\Delta\phi_{\mathrm{M}}. \tag{2.15}$$

A Fabry-Perot interferometer can then achieve the required sensitivity of $\Delta\phi_{\mathrm{FP}} \sim 10^{-9}$ rad with arm lengths of a few kilometres ($\Delta\phi_{\mathrm{M}} \sim 10^{-11}$) if the arm cavity finesse $\mathcal{F} \sim 10^2$.

Later, it will be useful to consider the cavity input mirror to have a frequency/cavity length dependent reflectivity $r(2kL) = E_{\mathrm{ref}}/\varepsilon_0 = |r(2kL)|e^{i\phi}$. The magnitude of the cavity reflection coefficient for the overcoupled case above is plotted in Fig. 2.5.

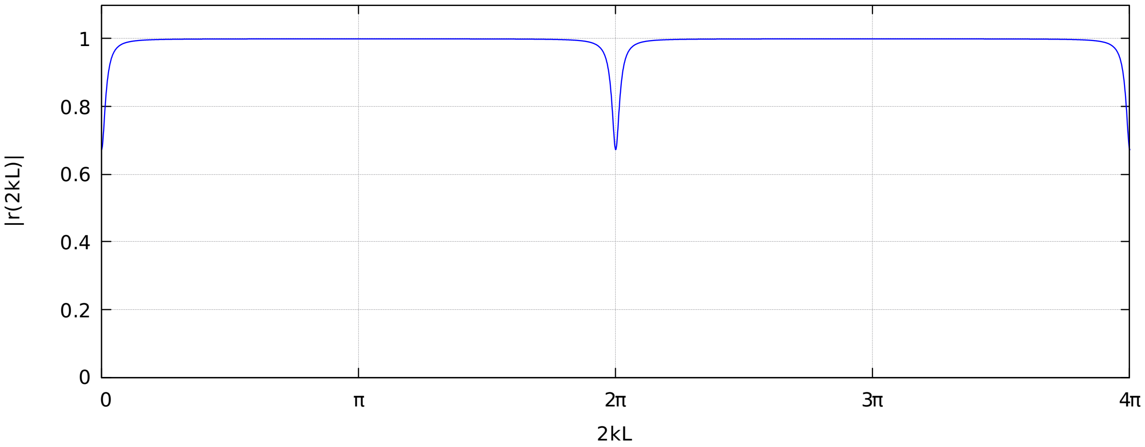


Figure 2.5: Magnitude of cavity reflection coefficient $|r(2kL)|$ ($r_1 = 0.95$ $r_2 = 0.99$).

This shows that the magnitude at resonance $r_0 = |(r_1 - r_2)|/(1 - r_1 r_2)$ rapidly rises to a constant magnitude of $(r_1 + r_2)/(1 + r_1 r_2) \sim 1$. It follows that for light at frequencies which are not close to resonance, where $\phi \sim 0$, the reflection coefficient is insensitive to both frequency and cavity length and so are entirely reflected. This difference between the reflection coefficient sensitivity on resonance and insensitivity around anti-resonance can be exploited in a powerful technique (or variants) for both GW detection and interferometer control purposes.

2.3.2 Interferometer Operation and Gravitational Wave Signal Extraction

In Section 2.1 we found that the output power P measured by the photodiode in a simple Michelson interferometer is proportional to $\sin^2(2k\Delta L)$. A plot of the ratio of output power to incident intensity $|\varepsilon_0|^2$ is shown in Fig. 2.6, where we have indicated two possible operating points, each achieved with the appropriate phase offset. Point 1 would seem to be the natural choice to obtain maximum sensitivity to change in length since the gradient is maximal and has linear response. However, the output is also maximally sensitive to laser power noise and the small signal we wish to detect must compete with noise within a relatively large DC offset.

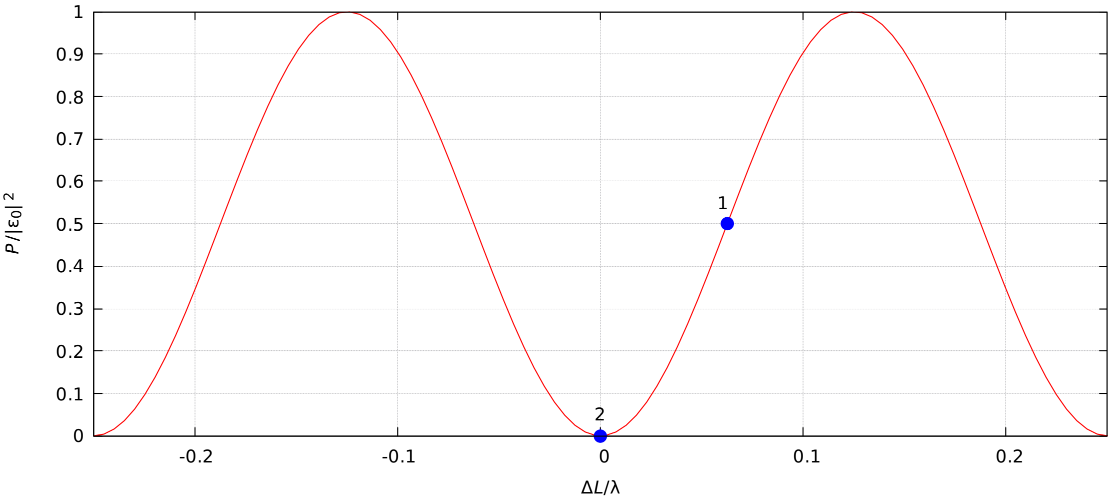


Figure 2.6: Ratio $P/|\varepsilon_0|^2$ as function of ΔL showing possible operating points located at points 1 and 2.

The DC offset can be eliminated by operation at point 2, achieved by ensuring that the light returning to the BS from the arm cavities destructively interferes on the output side of the BS. This operating point is referred to as the ‘dark fringe’, where the output signal is now entirely due to the induced phase change. The cost of operation at the dark fringe is that the response is now quadratic about the null point since $\sin^2(2k\Delta L) \sim (2k\Delta L)^2$, making detection practically impossible since $(\Delta L)^2 \sim 10^{-36}$ m^2. Recovery of a linear response is possible by means of a modulation-demodulation technique which we now discuss.

The input laser light is phase modulated by passing it through an electro-optic modulator (EOM), a device whose index of refraction can be varied with an applied voltage. If the EOM is supplied with a sinusoidal signal at a radio frequency (RF) $\Omega \sim 10^7$ Hz, chosen so that $\omega \pm \Omega$ fall between resonances, and modulation depth m, then the input light to the interferometer will be of the form

$$E = \varepsilon_0 \exp(i(\omega t + m\cos(\Omega t))). \tag{2.16}$$

Eq. 2.16 may be expressed in terms of Bessel functions of the first kind $J_k(m)$ using the Jacobi-Anger expansion [36], from which we obtain

$$E = \varepsilon_0 e^{i\omega t} \sum_{k=-\infty}^{\infty} i^k J_k(m) e^{ik\Omega t}, \tag{2.17}$$

where

$$J_k(m) = \left(\frac{m}{2}\right)^k \sum_{n=0}^{\infty} \frac{\left(\frac{-m^2}{4}\right)^n}{n!(n+k)!} \quad \text{and} \quad J_{-k}(m) = (-1)^k J_k(m). \tag{2.18}$$

A large number of sidebands will be created but for $m \ll 1$ the Bessel functions fall off rapidly with increasing k, and so a good approximation is obtained by only considering the $k = \pm 1$ contributions. With $J_{-1}(m) = -J_1(m)$ we then have

$$E \approx \varepsilon_0 e^{i\omega t}(J_0(m) + iJ_1(m)e^{i\Omega t} + iJ_1(m)e^{-i\Omega t}), \tag{2.19}$$

which to second order in m reduces to

$$E = \varepsilon_0 e^{i\omega t}\big(1 - \frac{m^2}{4} + i\frac{m}{2}(e^{i\Omega t} + e^{-i\Omega t})\big). \tag{2.20}$$

This shows that a small proportion of the energy from the carrier has been transferred to the sidebands. The field reflected from the cavities is the linear superposition of each field component multiplied by the corresponding reflection coefficient. This gives

$$E_{\text{ref}} = \varepsilon_0 e^{i\omega t}(r(\omega)J_0 + r(\omega+\Omega)J_1 e^{i(\Omega)t} + r(\omega-\Omega)J_1 e^{i(-\Omega)t}), \tag{2.21}$$

where for brevity we retain the J_k and the factor of i is ignored as this changes the phase of both sidebands by $\pi/2$.

In the presence of a GW phase changes of magnitude $\Delta\phi = 2\mathcal{F}kLh/\pi$ but opposite sign are induced in each arm, thereby changing the reflection coefficients at the carrier frequency of the x and y arm cavities to $r_x = r(\omega)e^{+i\Delta\phi}$ and $r_y = r(\omega)e^{-i\Delta\phi}$ respectively. As discussed in Section 2.3.1, the reflection coefficients for the sideband components remain unchanged at $r(\omega \pm \Omega) \sim 1$. Following some algebra and making the approximation $\sin(\Delta\phi) \approx \Delta\phi$, we find the output field to the photodiode is then of the form

$$E_A = i\frac{\varepsilon_0}{2}e^{i2k\bar{L}}(r(\omega)J_0\mathcal{F}kLh/\pi + J_1\sin(2\pi\Delta L/\lambda_{mod})\cos(\Omega t)), \tag{2.22}$$

where $\lambda_{mod} = 2\pi c/\Omega$ is the modulation wavelength. In Eq. 2.22, the first term comes from the carrier whilst the second is from the sidebands. With $L_x = L_y$ (i.e $\Delta L = 0$) both carrier and sidebands lie on the dark fringe, and our measured signal is again proportional to h^2. If we now introduce a Schnupp asymmetry by setting the cavity length in both arms equal but the difference in distance between the arm cavities and the beam splitter to $n\lambda$, then the sidebands no longer lie on the dark fringe even when the carrier is, and $|E_A|^2$ now contains a cross term linear in h, oscillating at the modulation frequency. Demodulation of the output at the modulation frequency, followed by low-pass filtering to remove the 2Ω component, enables recovery of the linear term in h.

2.3.3 Interferometer Control

Maintaining the conditions for maximum sensitivity requires precision control systems which hold the relative positions of the mirrors accurate to a very small fraction of the wavelength of the laser. Feedback control systems must sense deviations from the required operating point and then respond with an appropriate signal from actuators which reduce the deviation.

We can see from Fig. 2.3 that the power circulating in a Fabry-Perot cavity is symmetric on either side of resonance and so this signal would not be useful in determining whether, for example, a mirror position along the optical axis should be increased or decreased. The gradient, however, is antisymmetric around the control point and in Fig. 2.7 we show the derivative of the circulating power. This has the ideal form of a control signal near the required operating point; the signal is bipolar where its sign depends on which side of the operating point the system has moved; it has zero crossing at the operating point; and it also has high gain and linearity.

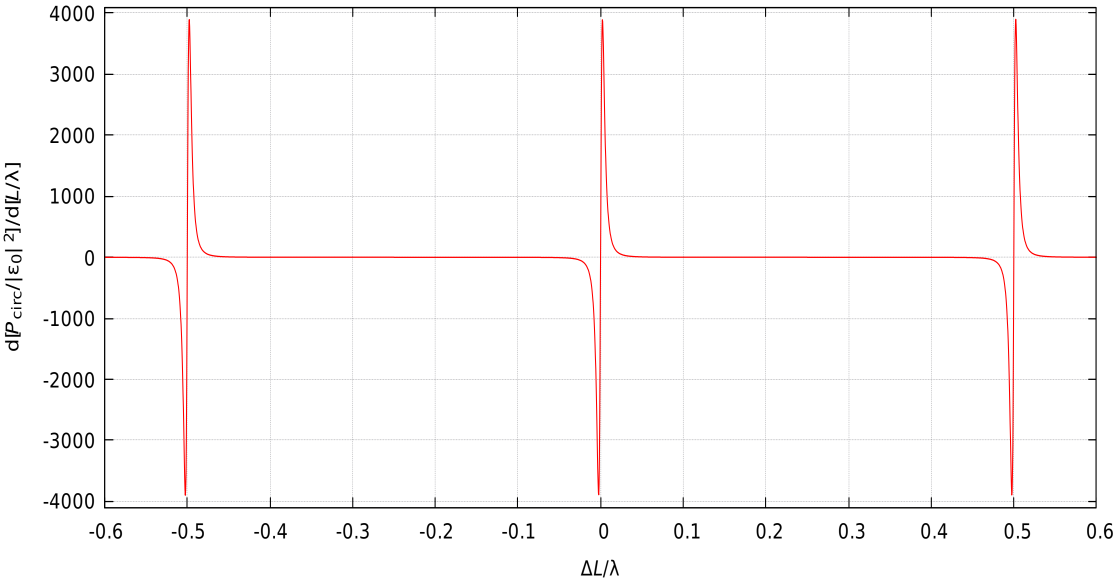


Figure 2.7: Gradient of $P_{circ}/|E_0|^2$ as a function of $\Delta L/\lambda$ showing the bipolar behaviour around resonance required for a control signal.

In order to obtain a control signal similar to that shown in Fig. 2.7 the Pound-Drever-Hall (PDH) [37] locking technique may be used. The laser light is phase modulated with a modulation frequency, usually in the radio frequency (RF) band, which is far from an integer multiple of the FSR of the cavities. Modulation imposes sidebands on the carrier whose transmission into the cavities will differ from that of the carrier.

Most of the sideband amplitude will be reflected from the cavities which beat with the reflected carrier. Demodulation of the output at the modulation frequency and low-pass filtering of the signal yields a signal proportional to ΔL.

2.4 Detector Noise

In this section we briefly review some of the main sources of noise present in GW detectors. Whilst our discussion is quite general, our focus will be on the noise characteristics of the LIGO detectors.

In the absence of a GW signal all detector output is the summation of noise originating from various sources. Individually or in combination, these sources may dominate or mimic a GW signal, severely hindering the prospect of detecting the presence of relatively weak GW signals. The sources of noise may be internal or external to the detector, examples of the latter being gravity gradient noise, caused by local changes in mass distribution which induce gravitational forces on the mirrors, and anthropogenic noise. Internal sources of noise may be classified as either technical or fundamental.

Technical noise sources, generated by the various subsystems within the interferometer, result from the design implementation, experimental practice, or unexpected behaviour of instrumental components. Examples of technical noise are the 60 Hz powerline harmonics, and vibrational modes of the pendulum suspensions. Predictable noise, produced at well-defined frequencies, may be monitored and removed with signal processing techniques [38]. Fundamental forms of noise such as photon shot noise and thermal noise are unavoidable as they are inherent in the physics of the detection principle used or necessary operating conditions. Fundamental noise represents the theoretical limit of experimental design sensitivity.

With the exception of some forms of technical noise shared by detectors with common design (e.g. powerline harmonics), all of the noise sources mentioned so far generally produce noise which is uncorrelated to noise in other distantly separated detectors. However, there are some natural phenomena such as earthquakes, solar storms and Schumann resonances[1] [39] which, due to their extensive or global effects, can produce correlation. The impact of such noise sources on data can be reduced by subtraction of noise detected by seismometers and magnetometers.

2.4.1 Seismic Noise

The natural choice of location for high precision instruments such as GW detectors are areas of low seismic activity and distanced from anthropogenic disturbances. Unfortunately, natural seismic vibrations are a constant presence and some environmental noise input from human activity, such as local vehicular motion, and atmospheric conditions (e.g. wind) is unavoidable. Seismic noise couples into the alignment and suspension systems of the mirrors and dominates all noise input in the low frequency range $\lesssim$ 40 Hz (initial LIGO). Microseismic motion of the ground, induced by coupling of ocean waves to land or sea floor, produces peaks in the range 0.1–1 Hz [40].

[1] Lightning discharge excitations of the cavity formed between the Earth's surface and ionosphere which induce magnetic forces on susceptible detector components.

The detectors employ seismic vibration isolation systems which fall into two categories, passive and active. Passive isolation of the mirrors utilises the filtering properties of the pendulum suspension at frequencies above the pendulum resonance ($\sim$ 1 Hz) and spring-mass isolation stacks for optical benches. Active isolation is used to precisely control the mirror positions by means of ground motion sensors and actuators which provide correctional feedback to the suspensions.

2.4.2 Thermal Noise

Thermal noise originates from the fact that at a given operating temperature T the molecules or atoms of which the detector is composed must have a random motion with a kinetic energy of order $\sim k_bT$ for each degree of freedom by equipartion.

There are two main types of thermal noise which affect the detector: off-resonance thermal excitation of the mirror suspension wires and mirror mode vibration. The normal modes of the suspensions produce clusters of many sharp resonances around 300 Hz and at kHz frequencies, referred to as ‘Violin modes’. Mirror thermal noise arises from Brownian motion of the mirror surface and within the bulk (substrate). Thermal noise limits detector sensitivity in the frequency range of the detector between approximately 35 – 100 Hz.

2.4.3 Quantum Noise

Shot noise results from the fact that photons obey Poisson statistics. Fluctuations in the number of photons arriving at the photodiode of the detector are related to the average number of photons sampled N as $\Delta N = \sqrt{N}$, so the relative error in the photodiode measurement varies as $N^{-1/2}$. Shot noise limits the sensitivity of the detector in the high frequency band of the detection range above $\sim$ 100 Hz.

The detector is subject to another fundamental quantum noise due to photon momentum transfer to the mirrors which gives rise to radiation pressure. Whereas shot noise decreases with increasing laser power, radiation pressure increases. Radiation pressure limits the effective number of bounces desirable in the interferometer arms and is therefore the reason that laboratory scale GW detectors with very high finesse cavities are not feasible. The sum of shot and radiation pressure noise defines a fundamental restriction to detector sensitivity known as the standard quantum limit.

2.4.4 Characterisation of Detector Noise

The noise characteristics of a GW detector may be represented by its strain amplitude spectral density (ASD) in units of $\mathrm{Hz}^{-1/2}$ over a broad range of frequencies. Fig. 2.8 shows the design noise budget ASD for the initial LIGO configuration, with major contributing sources of noise shown individually and total noise (red). The best sensitivity or lowest noise occurs around 160 Hz with an ASD value $\sim 4\times10^{-23}$ $\mathrm{Hz}^{-1/2}$. In general, the sources of noise which limit the sensitivity in the interferometer (e.g. in the initial LIGO design, seismic, thermal, and shot noise) are those which touch the sum noise curve, and those lying just below are sources which present the next noise issues to be addressed in future proposed design improvements.

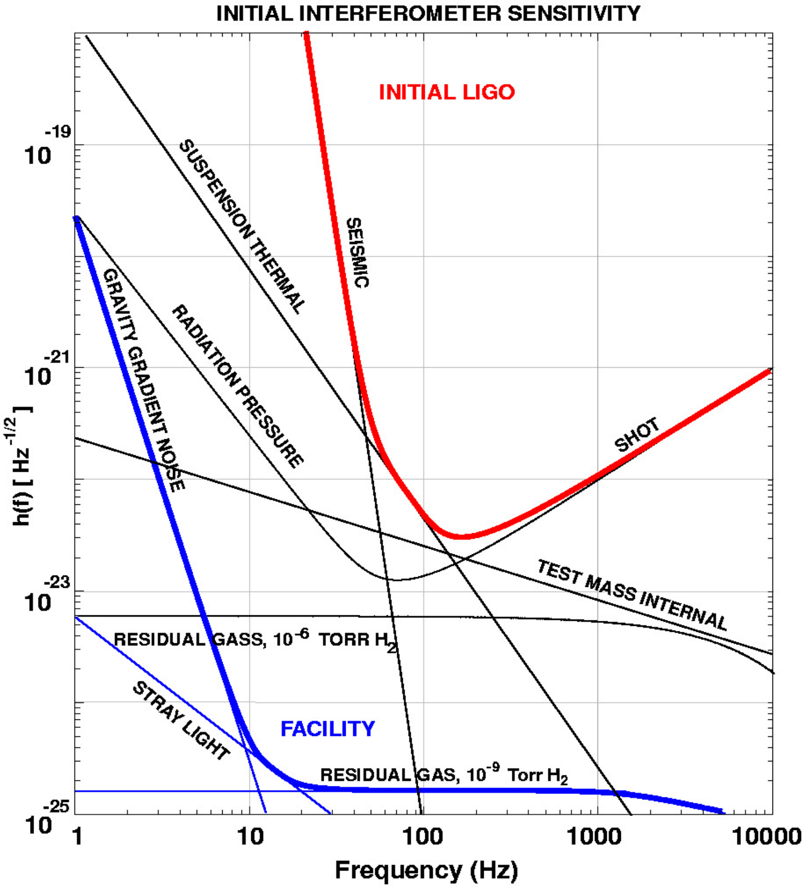


Figure 2.8: Design sensitivity for initial LIGO. The main noise contributions are shown individually along with the total noise (red curve). Reproduced from [41].

Following approximately two years of operation at design sensitivity, in late 2007 the 4 km LIGO detectors H1 and L1 underwent a program of upgrades referred to as Enhanced LIGO [42] which included higher laser power (10 → 35 W). In Fig. 2.9 we show the typical ASD of the Enhanced detectors during the S6 science period (July 2009 – October 2010). The general features of noise shown in the ASD are trends in the noise spectrum over wide ranges of frequency (broadband noise), similar in shape to the noise design curve in Fig. 2.8, and high amplitude narrowband sinusoidal backgrounds which we henceforth refer to as lines. We can also see that the best sensitivity at around 200 Hz of $\sim 2 \times 10^{-23}$ $\mathrm{Hz}^{-1/2}$ is an improvement beyond the initial design by a factor of ~ 2, in agreement with the anticipated enhancement goal.

The second generation of GW detectors employ techniques which reduce the limiting noise sources described in the previous section by varying degrees. For example, the Advanced LIGO detectors use signal recycling for better (and tunable) frequency response, higher laser power which reduces shot noise, and improvements in seismic isolation. Overall, these enhancements give Advanced LIGO (ALigo) detectors an order of magnitude improvement in sensitivity. For a detailed description of the second generation detectors the reader is referred the following; for Advanced LIGO [45], Advanced VIRGO [46] and GEO-HF [47].

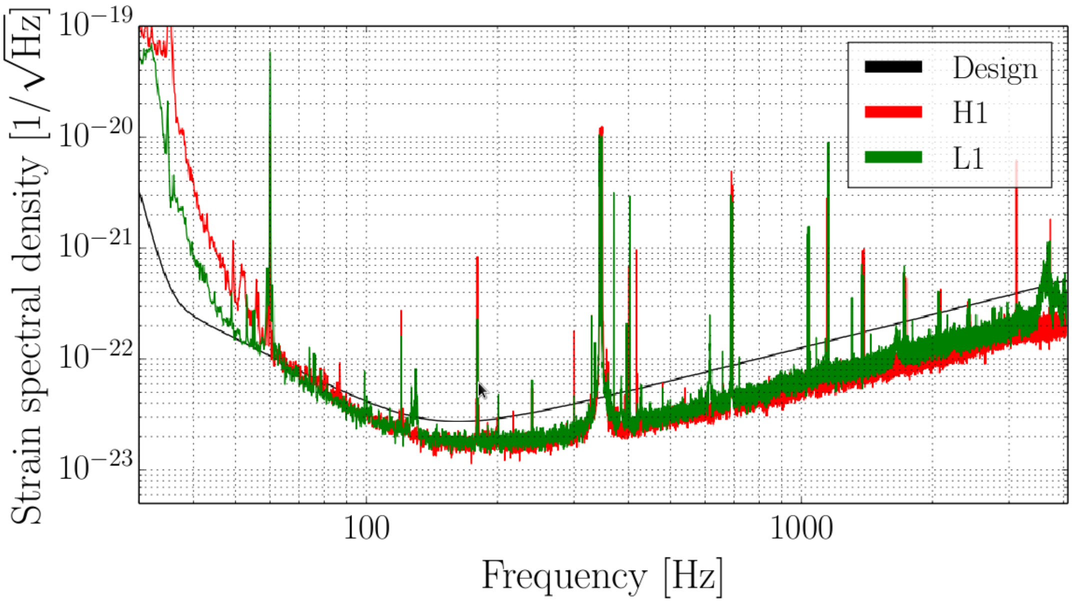


Figure 2.9: Typical noise characteristics for LIGO detectors H1 (red) and L1 (green) during S6. The noise design curve is shown for comparison (black). Reproduced from [43].

2.5 Gravitational Wave Detector Response

The sensitivity of an interferometer to GWs is not only a function of source strength and distance. The response of the detector to incident GWs is a linear combination of the h_+ and h_- polarisations which is given by the projection of each polarisation onto the arms of the interferometer. For a source direction with sky position given by the spherical polar angle θ and azimuthal angle ϕ with polarisation angle ψ, the response of a detector whose arms lie along the x and y axes is given by

$$\frac{\Delta L(t)}{L} = F_+(\theta,\phi,\psi)h_+(t) + F_\times(\theta,\phi,\psi)h_\times(t), \tag{2.23}$$

where F_+ and $F_\times$ are the antenna response functions. These are given by

$$F_+(\theta,\phi,\psi) = \frac{1}{2}(1+\cos^2(\theta))\cos(2\phi)\cos(2\psi) - \cos(\theta)\sin(2\phi)\sin(2\psi), \tag{2.24}$$

$$F_\times(\theta,\phi,\psi) = \frac{1}{2}(1+\cos^2(\theta))\sin(2\phi)\cos(2\psi) + \cos(\theta)\sin(2\phi)\cos(2\psi). \tag{2.25}$$

Fig. 2.10 shows the interferometer response function. Although the response drops to a minimum in the plane of the detector, the root mean squared sensitivity to both polarisations is significant over a large proportion of the sky. Therefore, unlike most other astronomical instruments such as telescopes, GW detectors have an almost omnidirectional sensitivity but due to the long wavelengths of GWs are non-imaging. Consequently, GW source sky location can only be inferred from a single detector by long period observation taking into account Earth's orbit and rotation, thereby limiting localisation to continuous sources. With a minimum of three GW detectors, each separated by long baselines, short-lived or transient GW sources can be localised on the sky by triangulation of source signal delay times between the detectors. As with many other experiments, reproducibility in the form of independent coincident observation also gives greater confidence and statistical significance.

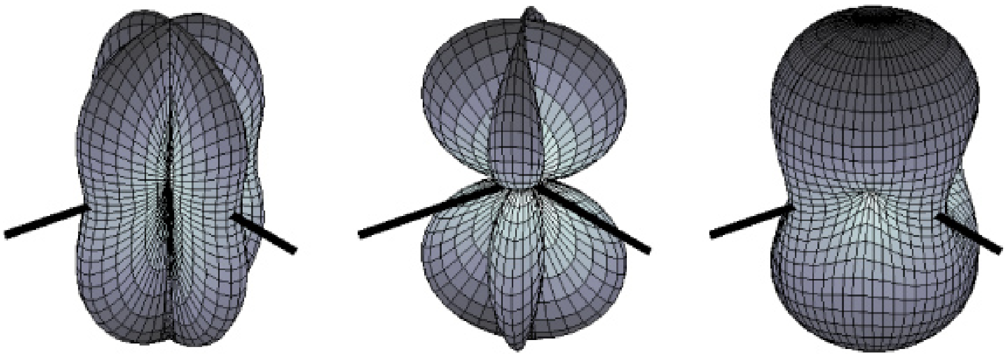


Figure 2.10: Interferometer antenna response for a detector whose beam splitter is situated at centre of each figure and arm orientation indicated by black lines. Left: + polarisation, middle: × polarisation and right: zero polarised (i.e. $\sqrt{(|F_+|^2 + |F_-|^2)}$). Reproduced from [44].

Chapter 3

Gravitational Wave Data Analysis

In this chapter we present the key concepts underlying many of the current approaches to GW detector data analysis. The chain of analysis leading to GW candidate selection is referred to as an ‘analysis pipeline’, of which there are three main components: detector data characterisation and preprocessing, the GW search algorithm, and assessment of the search sensitivity and reliability. For simplicity we follow this chain in order of presentation but the development of a mature analysis pipeline involves continual evolution and/or feedback from all these components.

In the following sections we illustrate some data analysis procedures using a sample of LIGO detector strain data and with figures drawn mainly from early development of the work presented later. Our treatment of the Fourier transform (FT) and time domain filtering closely follow the relevant material in Numerical Recipes by W.H. Press et al. [48].

3.1 Data Characterisation

The amplitude spectral density (ASD) of the detector strain data is a convenient form of characterisation for GW search purposes, from which we can assess the potential sensitivity of the GW search frequency band and consider our approach to data conditioning or preprocessing methods.

Most GW search pipelines use some form of discrete function transform to perform spectral analysis and comparison between signals. The discrete Fourier transform (DFT) is used for sampled data and so we begin with a brief discussion of the FT and its relation to data spectral characterisation by the ASD.

3.1.1 The Fourier Transform

The FT forms the basis for both preprocessing spectral analysis and some GW search algorithms by its utility in determining autocorrelation for signal power estimation and cross-correlation for signal comparison. FTs decompose a time domain signal into an equivalent representation in the time-frequency domain in an orthonormal basis of sines and cosines. The DFT is implemented as a fast Fourier transform (FFT) since, for sample size N, the computational effort scales as $N \log_2 N$ compared to N^2 for the time domain computation of cross-correlation related quantities. Eq. 3.1 shows the FT for continuous data and Eq. 3.2, the discrete form for data sampled at the rate f_s. For the remainder of this discussion we use the continuous form. Theorems stated in this section are to be found in [49].

$$\mathcal{F}[h(t)] = H(f) = \int_{-\infty}^{\infty} h(t)e^{-2\pi i f t}dt. \tag{3.1}$$

$$\mathcal{F}[h_n] = H_n = \sum_{n=0}^{n=\infty} h_n e^{-2\pi i f n \Delta t} \quad \text{where } \Delta t = 1/f_s. \tag{3.2}$$

For two signals $h(t)$ and $g(t)$ the cross-correlation $C(\tau)$ at time lag τ is given by

$$C(\tau) = \int_{-\infty}^{\infty} g^*(t)h(t+\tau)dt. \tag{3.3}$$

The correlation theorem relates the FT of the correlation to the FTs of the functions as

$$\mathcal{F}[C(\tau)] = G(f)H^*(f). \tag{3.4}$$

When $g(t) = h(t)$ in Eq. 3.3 we obtain the autocorrelation function and Eq. 3.4 is then equivalent to the Wiener-Khinchin theorem. Assigning zero lag $\tau = 0$ we then obtain Parseval's theorem Eq. 3.5, which equates the energy of the signal calculated in both time and frequency domains:

$$\int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(f)|^2 df. \tag{3.5}$$

We may then regard $|H(f)|^2$ as an energy density per unit Hz. In practice one evaluates the time average ($\langle .. \rangle$) of the integrands in Eq. 3.5 over some period T and finite frequency range $2\Delta f$ (i.e. $\pm\Delta f$), where Δf is the highest positive frequency. Since $h(t)$ is real, $|H(-f)|^2 = |H(f)|^2$ and so negative frequencies can now be included when we evaluate the ASD over the positive frequency range Δf with a prefactor of 2. This yields the power spectral density $S(f)$, again in units Hz^{-1} with a frequency resolution $f_{res} = 1/T$,

$$\langle |h(t)|^2 \rangle = 2\int_{0}^{\Delta f} S(f)df. \tag{3.6}$$

Since $h(t)$ is dimensionless the ASD has units of Hz$^{-1/2}$. We can directly relate the ASD at a particular frequency to the value of $h(t)$ by[1]

$$h_f(t) = \sqrt{2\Delta f S(f)}. \tag{3.7}$$

[1] N.B. the output of a single sided spectrum analysis already includes the factor of 2 and so this may be omitted.

3.1.2 Detector Noise ASD

The sampling frequency of data imposes a general restriction upon the highest frequency representation of amplitude in the ASD to $\Delta f \lesssim f_s/2 = N_f$, where N_f is known as the Nyquist frequency. This is a consequence of the Nyquist-Shannon sampling theorem [50] which may be stated: complete representation of a signal with a maximum frequency component f_{max} requires a sampling rate of at least $2f_{max}$. Higher spectral components than N_f cause the power at those frequencies to be attributed to lower frequencies. This type of misassignment is known as aliasing[1], and may be avoided by the removal of signal components higher than N_f by low-pass filtering prior to ASD determination.

The ASD may be obtained from the DFT of a large data set, or for ongoing analysis and/or data presentation purposes, in fixed short segments when processing incoming data. For a given sampling rate f_s and DFT block length n, corresponding to a time interval $\Delta t = n/f_s$, the frequency resolution of the ASD is equal to $1/\Delta t$. This is the width of frequency or bin size over which each ASD amplitude is evaluated over the range $0 - N_f$ Hz.

Unfortunately, the block treatment of data causes another form of aliasing known as spectral leakage, where signal power which should be allocated to a particular frequency bin is distributed amongst neighbouring bins. This may be reduced by applying an appropriate window function to the data segments, which taper the time series at the beginning and end of the data block prior to the DFT. Examples of commonly used window functions include the Hann, which is a raised cosine function, and the triangular Bartlett window. Suppression of segment ends requires that the ASD should be obtained from overlapping data segments rather than contiguous blocks to minimise window edge effects.

If we wish to characterise detector data spectrally over long periods of time, then the average of many ASD spectra is required. In Fig. 3.1 we show characterisation of 128 s of LIGO H1 detector data[2] from the strain output channel H1:LDAS-STRAIN, sampled at $f_s = 16384$ Hz. We have averaged the ASD over 4 s windows with applied Hann function, giving a frequency resolution of 0.25 Hz over a frequency range 0 – 8 kHz.

The figure shows that low frequency noise ($\lesssim$ 40 Hz) dominates the strain data. Maximum sensitivity (i.e. lowest noise) is achieved at around 200 Hz and at higher frequencies follows the shot noise trend seen in Fig. 2.8. Apart from general trends in the ASD, we can also see many instrumental lines due to suspension noise and those at the mains powerline frequency 60 Hz and harmonics (120 and 180 Hz).

Before turning to the next link in the analysis chain, data preprocessing, we may use Eq. 3.7 to estimate the ASD sensitivity required to detect a source with the value of characteristic strain calculated in Section 1.3.5 $h_c = 10^{-21}$. This gives the required sensitivity as $\sqrt{S(f)} = h_c/\sqrt{f_s} \sim 8{\times}10^{-24}$ Hz$^{-1/2}$, less than one order of magnitude lower than the ASD minimum in Fig. 3.1.

[1] The potential severity of aliasing can be seen in the following. Consider sampling the position of an oscillator at the rate of its frequency of motion. From this single sample the motion appears stationary and so assign the oscillation frequency to zero.

[2] Data used is 128 s from 2010-09-16 06:00:17 UTC.

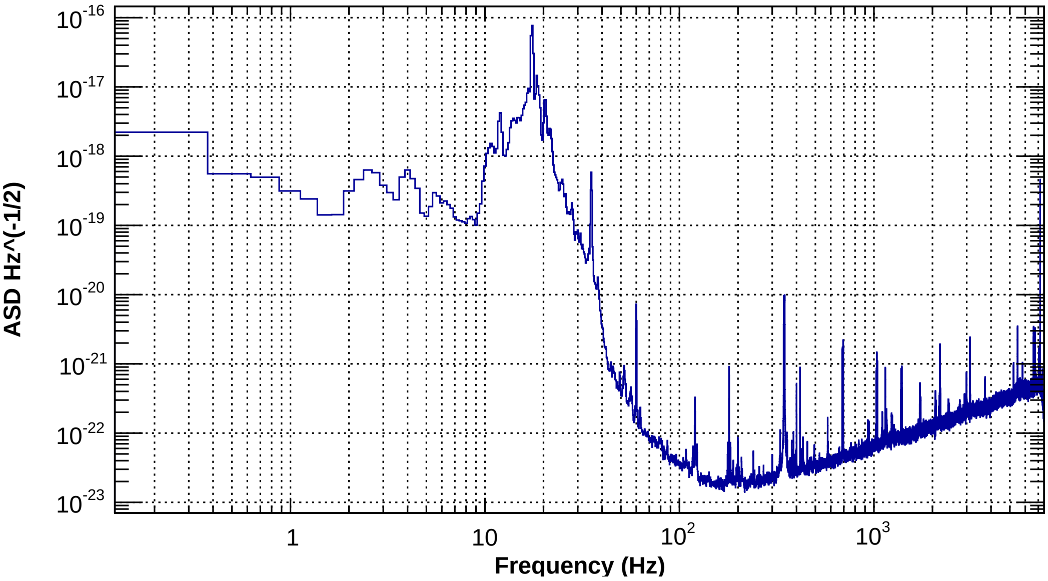


Figure 3.1: Short term characterisation of LIGO H1 strain data.

3.2 Data Preprocessing

The objective of data preprocessing is to achieve the best detection sensitivity in the GW search frequency range and to give the resulting data suitable statistical properties for the search analysis. Gaussian noise power is distributed equally at all frequencies (spectrally white) and thus gives the simplest non-trivial spectrum we could obtain, a constant ASD value of $\sigma/\sqrt{N_f}$, where σ is the noise standard deviation. Moreover, Gaussian noise is easily characterised and makes theoretical analysis of search methods considerably more tractable. It is therefore desirable, whether conducting time or frequency domain analysis, to condition the data such that it is close to Gaussian within the search frequency range.

The characterisation of detector data by ASD shows the features, both general trends and lines, which divide our preprocessing tasks into two distinct types: broadband whitening and narrowband line removal. We have at our disposal a number of standard signal processing techniques which assist in these tasks and so we now begin with a general description of time domain filtering.

3.2.1 Time Domain Filtering

The general form of the output of a time domain filter y_n of a time series x_n is given by a linear sum of $M+1$ previous inputs and N previous outputs as

$$y_n = \sum_{k=0}^{M} b_k x_{n-k} + \sum_{j=0}^{N-1} a_{j+1} y_{n-j-1}. \tag{3.8}$$

If $N \neq 0$ then Eq. 3.8 represents an infinite impulse response (IIR) filter and when

$N = 0$ we have a finite impulse response filter (FIR). The IIR filter is so called because the feedback from previous outputs results in non-zero output for extended periods, even when the input has zero value. In contrast, the FIR filter output must reach zero after a finite number of steps upon zero-valued input. IIR filters have the advantage of requiring fewer coefficients to achieve the desired response and therefore computationally faster. For this reason we now confine our discussion of time domain filtering to the IIR filter.

The frequency response of the IIR filter may be obtained by taking the z-transform of Eq. 3.8. The z-transform of a discrete time series x_n is defined as [51]

$$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}, \tag{3.9}$$

where z is a complex number. Taking the z-transform of Eq. 3.8 and rearranging to give the z-domain response, $\mathcal{H}(z) = Y(z)/X(z)$, gives

$$\mathcal{H}(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{j=0}^{N-1} a_{j+1} z^{-j}}. \tag{3.10}$$

Comparing the discrete FT (Eq. 3.2) with the z-transform when z is expressed in polar form $z = re^{2\pi i f/f_s}$, reveals that the z-transform is equivalent to the discrete FT when $r = 1$. Therefore the FT lies on the unit circle in the z-plane. In terms of the filter coefficients and sampling frequency f_s, the frequency response of the filter, $\mathcal{H}(f)$ is given, upon making the substitution $z = e^{2\pi i f/f_s}$ in Eq. 3.10, by

$$\mathcal{H}(f) = \frac{\sum_{k=0}^{M} c_k e^{-2\pi i k f/f_s}}{1 - \sum_{j=0}^{N-1} d_j e^{-2\pi i (j+1) f/f_s}} \quad \text{for} \quad -f_s/2 < f \leq f_s/2. \tag{3.11}$$

Designing a time domain filter first involves determining the required filter response function, the complexity of which depends upon the data conditioning required. For example, we might wish to attenuate signal frequencies below and/or above particular values with high-pass and low-pass filters respectively. The difficulty resides in finding an economic set of coefficients which give a sufficiently accurate approximation. Fortunately the problem can be recast by reducing the forms which $\mathcal{H}(z)$ can take to products of functions which are the ratio of two second-degree polynomials in z. These functions are known as second order sections (SOS). The recurrence relation for a SOS is given by[1]

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + a_1 y_{n-1} + a_2 y_{n-2}. \tag{3.12}$$

This yields the general form of the z response of a SOS as a rational function of quadratic polynomials in z^{-1} given by

$$\mathcal{H}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}. \tag{3.13}$$

[1] N.B. the signs of the feedback coefficients (i.e. $a_{1/2}$) are often reversed.

The frequency response depends on the locations of the roots of the quadratic forms. Roots in the numerator, referred to as 'zeros', suppress the response at the corresponding frequency. Roots in the denominator (poles) amplify the response and these must lie within the unit circle of the z-plane for stability. We have calculated the frequency response of the SOS as

$$|\mathcal{H}(f)| = \sqrt{\frac{b_0^2 + b_1^2 + b_2^2 + 2b_1(b_0 + b_2)\cos(2\pi f/f_s) + 2b_0 b_2 \cos(4\pi f/f_s)}{1 + a_1^2 + a_2^2 - 2a_1(1 - a_2)\cos(2\pi f/f_s) - 2a_2\cos(4\pi f/f_s)}}. \tag{3.14}$$

A simple time domain line removal filter such as a notch filter works by weighting the previous inputs and outputs such that the frequency response is equivalent to a band-reject filter. The ASD of Gaussian noise which has been filtered with a single SOS notch filter designed for line removal at 60 Hz is shown in Fig. 3.2. The figure shows clear attenuation within a bandwidth $\sim$ 6 Hz around 60 Hz.

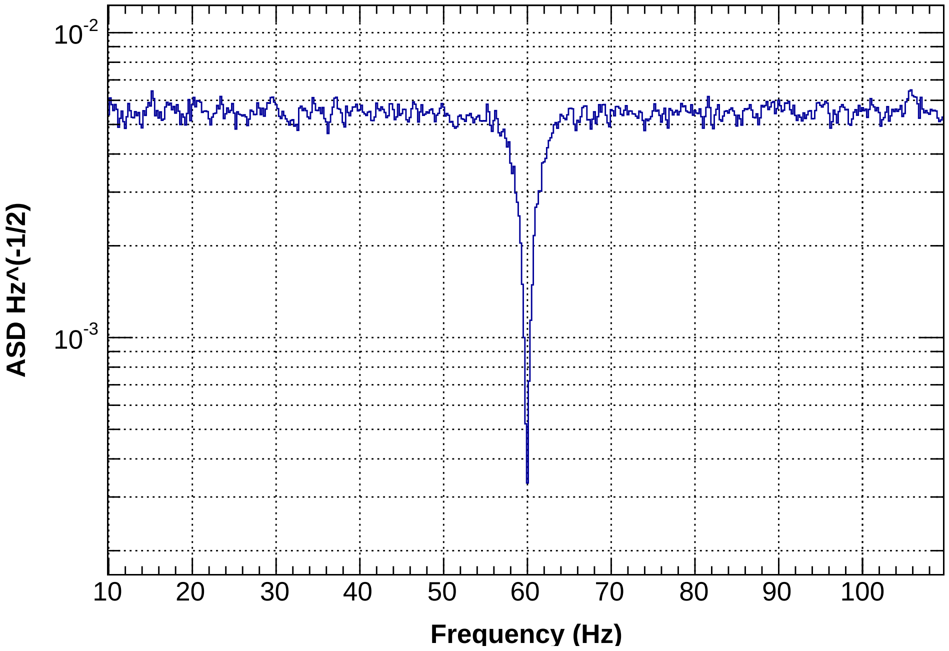


Figure 3.2: 60 Hz single SOS notch filter response. The SOS coefficients for this filter are $b_0 = 1$, $b_1 = -1.99947$, $b_2 = 1$, $a_1 = -1.99794$, $a_2 = 0.998467$.

More complex IIR filters which achieve a level of whitening can also be designed. A Butterworth filter can be constructed as the product of 12 SOSs with a low frequency 120 Hz cutoff multiplied by a second order section with a pole at 1000 Hz. Fig. 3.3 shows this filter applied to the same data used to produce Fig. 3.1.

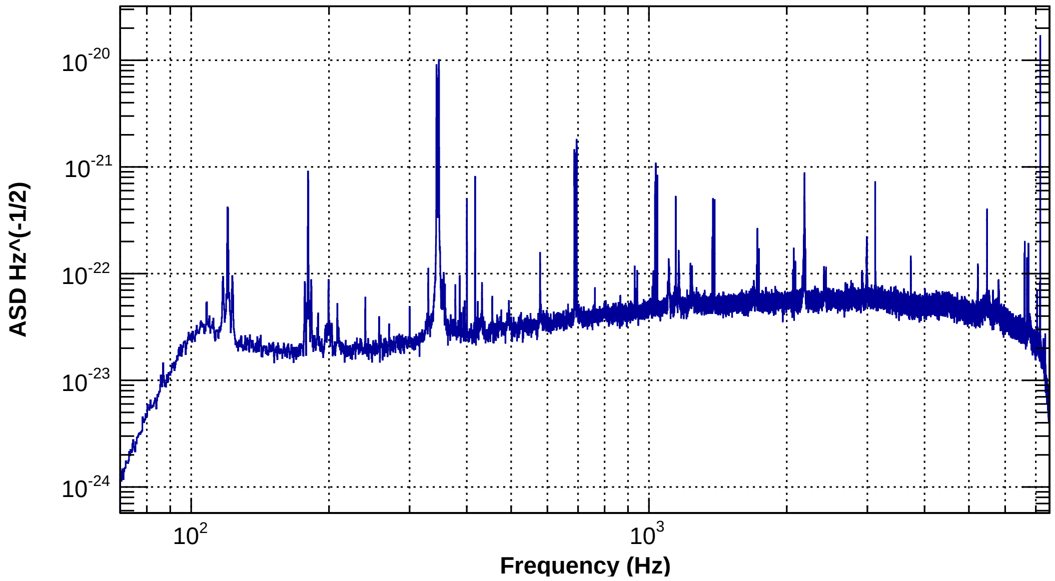


Figure 3.3: ASD on applying 12th order Butterworth filter.

The effect is dramatic, confining the general ASD trend to within one order of magnitude in the frequency range 120 – 7000 Hz, with a rapid drop in amplitude either side. We are still left with lines in the ASD which are up to about two orders of magnitude greater than the average ASD.

A convenient way to view the contribution to the noise amplitude at each frequency is by the cumulative amplitude spectral density (CASD). The amplitude of the CASD for a particular frequency bin is the sum of all ASD amplitudes in frequency bins less than or equal to that frequency. The resulting curve is monotonic and shows clear steps when the amplitude contribution is significant, corresponding with large peaks in the ASD.

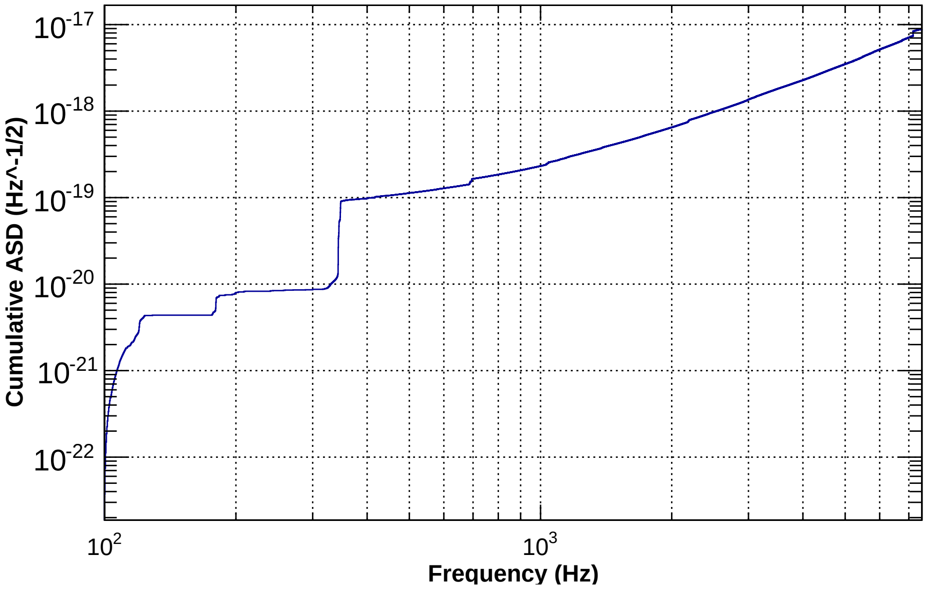


Figure 3.4: Cumulative ASD representation of Fig. 3.3.

In Fig. 3.4 we show the CASD representation of the ASD in Fig. 3.3 where we can see that the largest step at $\sim$ 350 Hz, due to violin modes, contributes approximately one order of magnitude to the total noise amplitude. The CASD of Gaussian noise would show a linear increase with frequency.

Filters often introduce large amplitude signals into the output during initialisation of the filter, known as filter transients. The transient time of the filter may be determined by applying it to Gaussian distributed data and observing how long it takes for the RMS of the filtered time series to stabilise. The filtered samples covering this transient time must not be used in subsequent analysis and are therefore removed from the data before proceeding. Alternatively, the filter may be preconditioned by prepending the data with a segment of length equal to the transient time from the beginning of the data. Again we discard the initial transient time filtered output but have now filtered all of the data.

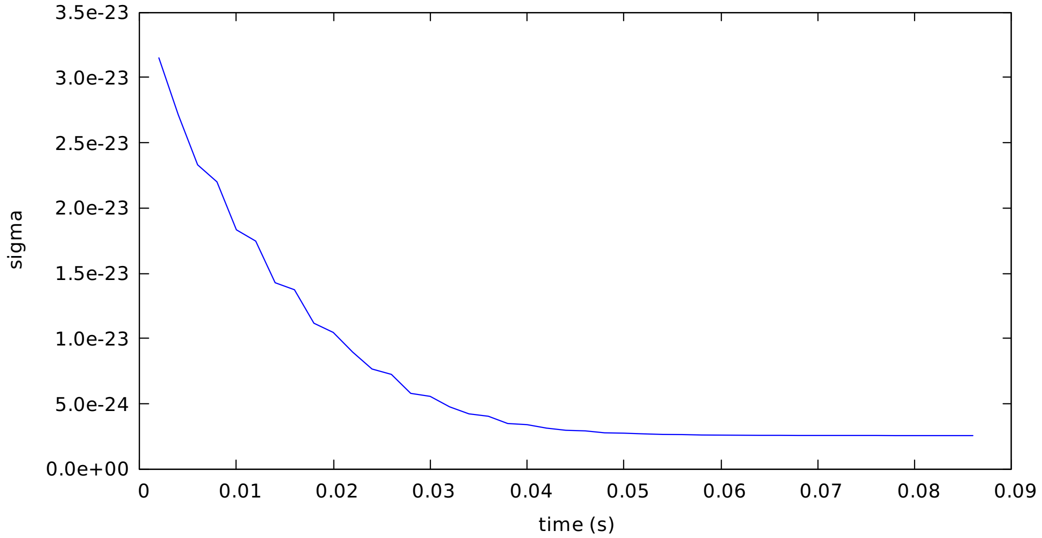


Figure 3.5: Filter transient determination of Butterworth 12th order filter.

3.2.2 Broadband Whitening by Linear Prediction

For GW search purposes, cross-correlation (CC) in particular, it is required that the ASD is not a strong function of frequency. A very powerful method used for speech processing and recognition is the linear prediction error filter (LPEF) which can be used for broadband whitening purposes. Our mathematical description of the LPEF essentially follows that contained in Advanced Digital Signal Processing and Noise Reduction by S.V. Vaseghi [52].

The LPEF attempts to predict future values of the time series from a linear combination of prior inputs by measuring correlated input via the autocorrelation matrix. For a time series x_n the predicted value x_n^p from M previous values is

$$x_n^p = \sum_{i=1}^{M} a_i x_{n-i}, \tag{3.15}$$

where a_i are the filter coefficients. The prediction error e_n^p is the difference between the actual sample value and that predicted:

$$e_n^p = x_n - x_n^p = x_n - \sum_{i=1}^{M} a_i x_{n-i}. \tag{3.16}$$

The prediction error sequence is the original signal minus predictable content, which is the filtered output we require.

Obtaining the coefficients for the LPEF is achieved by minimising the expectation value of the prediction error squared with respect to all coefficients. Ultimately this leads to

$$A = R^{-1} r, \tag{3.17}$$

where A is the column vector of filter coefficients, R is the autocorrelation matrix of x having components $R_{jk} = \mathbb{E}(\sum_{n=1}^{M} x_{n-j} x_{n-k})$, where $\mathbb{E}(..)$ denotes the expectation value, and r is the column matrix formed from the autocorrelation components, $r_j = \mathbb{E}(\sum_{n=1}^{M} x_n x_{n-j})$, excepting r_0. Explicitly,

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{M-1} \\ r_1 & r_0 & r_1 & \cdots & r_{M-2} \\ r_2 & r_1 & r_0 & \cdots & r_{M-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{M-1} & r_{M-2} & r_{M-3} & \cdots & r_0 \end{pmatrix}^{-1} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_M \end{pmatrix}. \tag{3.18}$$

The matrix R^{-1} has the form of a Toeplitz matrix, i.e is square and has constant left to right descending diagonal elements. The solution of Toeplitz matrix equations, especially for large matrices, is most efficiently obtained by use of the Levinson-Durbin algorithm (see pages 238–240 in [52]). The computational speed of the Levinson-Durbin algorithm facilitates the use of the LPEF as an adaptive filter, where the filter coefficients may be re-evaluated at chosen intervals.

The effect of the filter is to remove signal components correlated over M samples. If the sampling frequency of the time series is f_s, then the LPEF achieves spectral whitening over bandwidths [53]

$$\Delta f \gtrsim \frac{f_s}{M}. \tag{3.19}$$

Given the simplicity of the LPEF configuration, we decided to use the LPEF for broadband whitening purposes and during early preprocessing development we wrote our own LPEF code for testing. Fig. 3.6 shows the ASD of prefiltered LIGO H1 strain data where the noise ranges over three orders of magnitude. Fig. 3.7 shows the result of applying the LPEF configured with $M = 2000$, giving whitening over frequency ranges of $\Delta f = 16384/2000 \gtrsim 8$ Hz. The spectrum is now considerably flatter but has left many of the lines above or below the main horizontal trend. Our LPEF was written without the Levinson-Durbin optimisation and found to be unacceptably slow. Fortunately, much faster LPEFs were available in the LIGO software library, so we decided to use those. On comparison with the LIGO LPEF, we observed the same behaviour, strong subtraction occurring where the line bandwidth is close to Δf, whilst leaving those lines with less bandwidth above the median trend. The LPEF could also be used to remove the lines but would require a significantly larger value of M and therefore more LPEF coefficients to be computed.

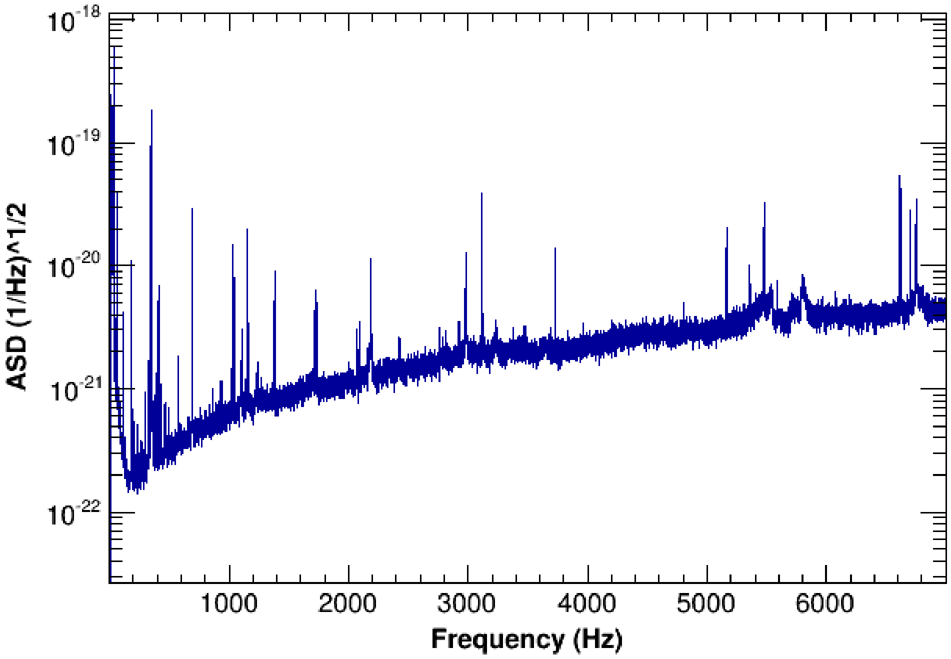


Figure 3.6: Spectrum of bandpassed H1 data prior to LPEF filtering.

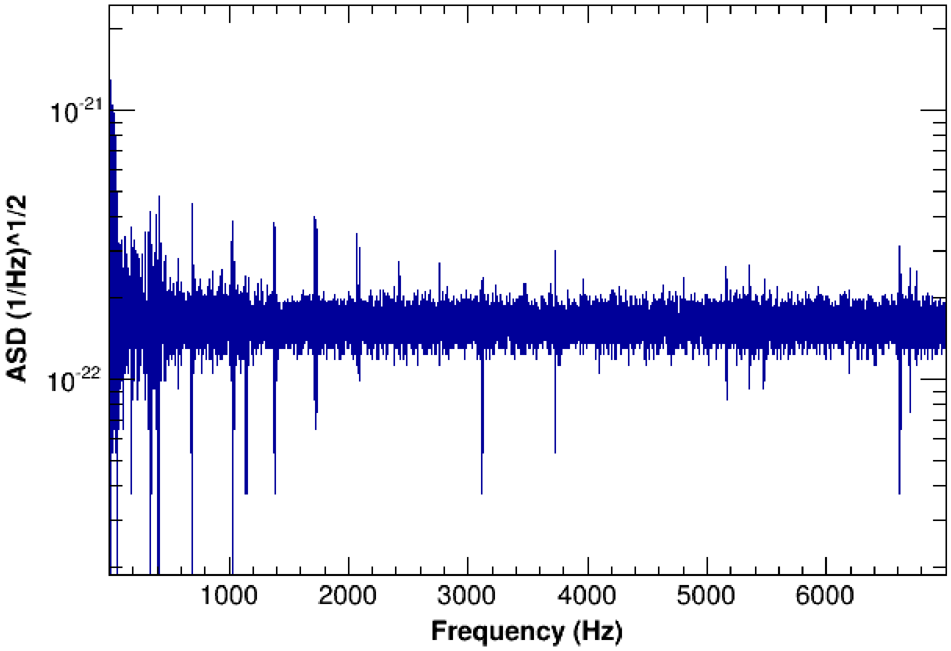


Figure 3.7: Spectrum of data used to produce Fig. 3.6 following LPEF broadband whitening.

3.3 GW Searches

The GW search analysis selects signals of interest or candidate GWs by setting thresholds or conditions on some statistically characterised properties of the data (e.g. signal power), which when satisfied signal the need for deeper analysis and, if not already included, evidence of corroborative events in other detectors. Confidence in the significance of the event trigger is based upon selecting threshold conditions such that detector noise is highly unlikely to result in the issue of an event trigger over long periods (e.g. months or years).

In order to determine the nature of the GW source and facilitate potential optical searches, the search analysis must be able to parameterise the event data in terms of variables intrinsic to the source, such as signal duration and characteristic frequency, and extrinsic parameters such as detection time lag between detectors from which the sky source location may be determined.

The best forms of discriminant between noise events and causally related triggers, whose origin may be a GW source, are coincidence measures. These may be the results of our source parameterisation, such as time coincidence windows, duration, frequency, waveform and any other means of showing consistency between the signals of interest.

3.3.1 Time-Frequency Domain Analysis

Short time Fourier transforms (STFTs) may be used to produce time-ordered power spectra to give a 'three-dimensional' representation known as a spectrogram. Spectrograms plot the power, indicated by a range of colours, at each time and frequency bin to give a pixel. Algorithms may be used to detect clusters of pixels where the power exceeds background noise by some predefined threshold for the generation of event triggers, or when used as a data quality monitor, to indicate times when the data may be unsuitable for analysis.

The Hierarchical Algorithm for Curves and Ridges (HACR) is a time-frequency cluster analysis developed by GEO600 for the detection of burst GW signals [54]. Fig. 3.8 shows an example of HACR output when processing simulated data with test waveforms added to Gaussian noise.

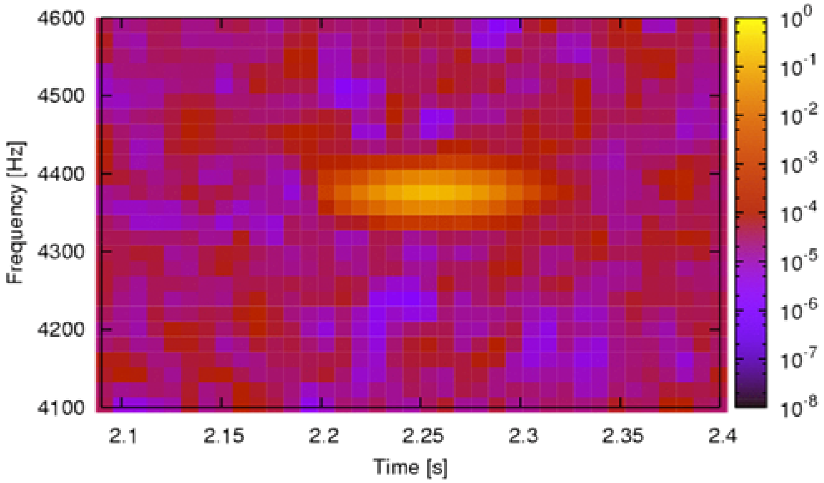


Figure 3.8: HACR time-frequency cluster analysis reproduced from [55].

HACR processes the data in short time segments $\sim$ 1 s. These segments are further divided into smaller equal time segments, the STFTs of which produce the spectrogram for the whole segment. Extremes in the distribution of power in each time-frequency bin or pixel are removed before computing a mean power and root mean squared (RMS) power. The mean is then subtracted from each pixel and divided by the RMS to obtain its signal-to-noise ratio (SNR). Event triggers are generated by the clustering of pixels which satisfy two thresholds on pixel SNR. If two or more neighbouring (i.e. adjacent or closest diagonal) pixels have SNR above the first lower value SNR threshold and at least one these pixels has SNR above the second upper SNR threshold then these are regarded as a time-frequency cluster. Event parameterisation such as event time and dominant frequency are determined by the SNR weighted average of the cluster in time-frequency coordinates.

An important class of search algorithms for modelled GW sources employ the matched filter technique. Theoretical waveforms for a well modelled source type, such as the inspiral phase of a compact binary coalescence (CBC), are used to generate a template bank from the parameter space of the generic waveform. Each template is a Fourier transform of the theoretical waveform $h_T(t)$. If the detector output is given by $s(t) = h(t) + n(t)$, where $h(t)$ is the GW signal and $n(t)$ the noise, the matched filter output is

$$M(t) = 4 \int_0^\infty \frac{H_T^*(f)S(f)}{S_n(f)} e^{2\pi i f t} df, \tag{3.20}$$

where $S_n(f)$ is the power spectral density of the detector output.

Whilst the matched template method has been shown theoretically to have optimum detection sensitivity when when searching for known waveforms amidst Gaussian noise, the exact waveform is never known in advance[1]. The expected signal from any modelled source may depend upon many parameters. Therefore the matched template search is often computationally demanding, involving the generation of thousands of waveform templates which must sample the parameter space with sufficient density to give some acceptable threshold probability of detection [56]. This method also requires that the noise should be accurately modelled and, ideally, Gaussian.

Regardless of whether the pipeline operates in the time or frequency domain, power or SNR alone is not a sufficiently reliable discriminator between noise transients and signals of interest, especially when, as is most often the case, the noise is neither Gaussian nor stationary. Therefore, most analysis pipelines will use at least one more consistency measure or 'statistic'. For example, the χ^2 time-frequency test [57] can be used as a secondary statistic for pipelines employing the matched filter technique. In essence, the GW search frequency band is subdivided and the time-frequency response of the signal in each frequency band is compared to that computed for the waveform template.

3.3.2 Time Domain Search Analysis

Time domain analysis permits the data to be streamed into the analysis algorithm one sample at a time and, given adequate computing power, processed faster than the sampling rate of the detector. Therefore, these methods are suitable for rapid low latency

[1] Whist some waveforms can be computed to high precision, the scale and phase remain unknown.

analysis. Unlike time-frequency methods, which process data in segments, the sensitivity of the analysis is independent of time, and the only edge effects introduced are by data conditioning filters. The signals we seek may be of very short duration, so it is an important advantage that time domain analysis preserves the time locality of such signals, rather than spreading these over windows as in the time-frequency approach which must always make a compromise between time and frequency resolution.

An example of an early LIGO time domain GW burst search algorithm is SLOPE [58]. SLOPE uses a sliding window of N samples of the time series data to generate gradient and intercept (or offset) line fitting parameters. If X_N and T_N denote the set of N time series samples and sample times ($t_i = i/f_s$) respectively, the gradient a and offset b are given by

$$a = \frac{\mathbb{E}(T_N X_N) - \mathbb{E}(T_N)\mathbb{E}(X_N)}{\mathbb{E}(T_N^2) - \mathbb{E}(T_N)^2}, \tag{3.21}$$

$$b = \mathbb{E}(X_N) - a\mathbb{E}(T_N), \tag{3.22}$$

where for example $\mathbb{E}(T_N^2) = (1/N)\Sigma_{i=0}^{N-1} t_i^2$. When processing Gaussian zero mean noise with variance σ^2, a and b are themselves Gaussian zero mean random variables, the respective variances of which are found to be

$$\sigma_a^2 = \frac{12 f_s^2}{N(N^2 - 1)}\sigma^2, \tag{3.23}$$

$$\sigma_b^2 = \frac{4N + 2}{N(N - 1)}\sigma^2. \tag{3.24}$$

The SNR of the SLOPE output gradient and offset parameters can now be expressed as $A = a/\sigma_a$ and $B = b/\sigma_b$ respectively. A threshold on the magnitude (or SNR) of the gradient output is set based upon the probability of obtaining output of this value or greater from processing noise. If the threshold and N values are appropriately selected, the presence of a burst signal is then much more likely than noise to result in an event trigger.

3.4 Detection Sensitivity and Reliability

The sensitivity of an analysis pipeline is measured by the efficiency with which it detects simulated GWs or signals with characteristic temporal/spectral properties added (injected) to the detector strain data. The injections are performed periodically and scaled over a range of strain amplitudes for the duration of the study. The detection efficiency is the ratio of true detections to the number of potentially detectable signals within the observation time.

Fig. 3.9 shows example[1] detection sensitivity curves for injected sine-Gaussian waveforms (described in Section 5.3) at four RTCC pipeline threshold values. The sensitivity curves plot on the x-axis (usually logarithmic) the injection amplitude expressed as the root-sum-square of the strain h_{rss} in units of $\mathrm{Hz}^{-1/2}$ and detection

[1] From development of the RTCC pipeline to be described later.

efficiency on the y-axis. The curves are monotonic, tending to be sigmoid in shape, and often show that detection efficiency rapidly increases above a particular h_{rss}.

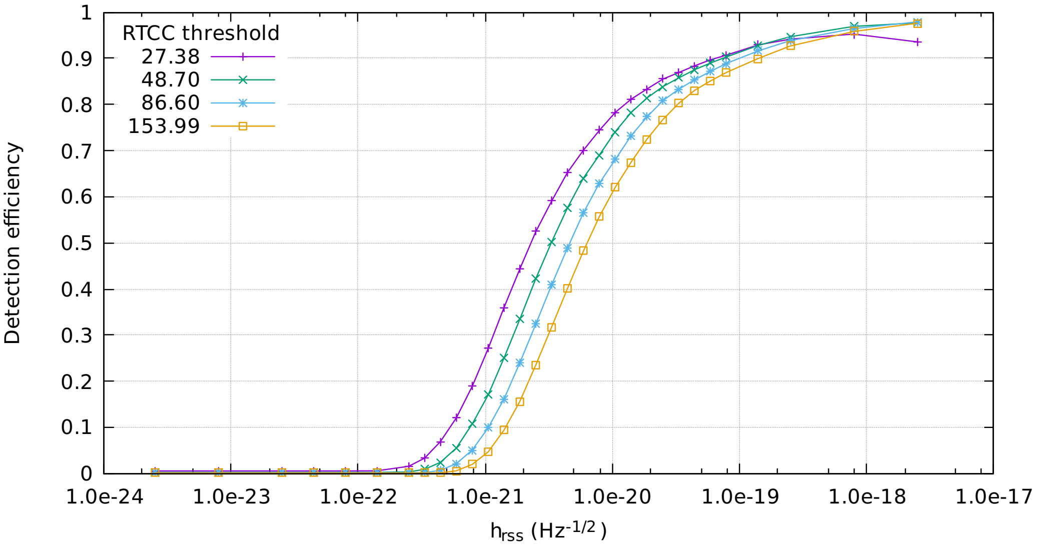


Figure 3.9: Example detection sensitivity curves of sine-Gaussian waveforms for four RTCC pipeline threshold values. The curves show the typical sigmoid shape with a rapid rise in detection efficiency above a particular injection amplitude h_{rss}.

Sensitivity curves are useful for both threshold configuration and comparison of analysis pipelines. The figure shows improvement in sensitivity with decreasing threshold but this also results in increased sensitivity to noise. The ability of the pipeline to distinguish noise events from injections or GW candidates is measured by the pipeline reliability.

The reliability of a pipeline trigger event is in inverse proportion to the false alarm rate (FAR). This is the rate at which noise backgrounds alone pass search pipeline thresholds to issue an event trigger, thereby indicating a false detection. Since detector noise characteristics are not stationary, it is necessary, especially when evaluating potentially strong GW candidates, to assess the FAR within a short time window close to the trigger event.

For pipelines which use data from at least two detectors the method used to estimate the FAR is known as a time-slide analysis. The data (excluding GW candidates) from each detector is first time shifted with respect to each other to eliminate all causal relationship. For example, time-slide data for the LIGO Livingston and Hanford detectors, separated by 3000 km, would require a minimal time offset equal to the light travel time between the detectors of $\sim$ 10 ms. The pipeline then processes this data to measure the rate of triggers to determine the FAR. The advantage of this method is that, apart from the causal constraint, the time shift is arbitrary and so a modest amount of data with different time offsets can generate very long period data for accurate FAR determination [59]. In the case where the pipeline input data is from a single detector, the FAR estimation is limited to one short dataset and is therefore subject to significantly more uncertainty than time-slide analysis. In this case we might consider increasing the size of the dataset by substantially extending the time window around the event, but this increases the probability that the determined FAR is no longer representative of the noise characteristics relevant to the event.

Chapter 4

Real Time Cross-Correlation Indicator

In this chapter we introduce the Real Time Cross-Correlation (RTCC) algorithm which we propose as a viable event trigger generator. We refer to the algorithm as an indicator of the standard time domain cross-correlation (CC). This means that in the presence of a signal of interest we require a high degree of correspondence between the significance of RTCC output above background noise and that obtained from standard CC. Close comparison will be made between the output from the RTCC algorithm and standard time domain CC using test data.

Event triggers are generated when a search algorithm indicates that a statistically significant signal is present in the data. Therefore, the trigger threshold criteria depend upon the statistical properties of the output of the algorithm when processing background noise. In particular, we are interested in the extremes (tails) of the output distribution as these will determine the probability that thresholds set in this region will be exceeded by noise alone, and so determine the false alarm rate. We make the assumption that the background noise output of the RTCC algorithm will be approximately Gaussian distributed. This is justified on the basis that many systems whose signal results from the summation of many independent random variables exhibit or tend towards Gaussian distributed output as a consequence of the Central Limit Theorem (CLT) [60]. The power of the CLT for analytic purposes lies in the fact that the distributions of the variables need not be Gaussian in order to produce Gaussian distributed output. Gaussian input and output is highly desirable since it is easily modelled, being completely described by its first and second central moments (i.e. mean and variance), and often makes analysis more tractable.

4.1 The RTCC Algorithm

The RTCC algorithm is an exponential average applied to the product of samples within two data streams for a range of sample lags. The algorithm is given in Eq. 4.1 and for comparison we give the standard time domain CC algorithm for real-valued data in Eq. 4.2. C_n^l is RTCC output at sample n and lag l between the time series X and Y, which could be the output from two detectors. The weighting factor w $(0 < w < 1)$ determines the relative contributions to the output made by past RTCC output and present data products.

$$C_n^l = (1-w)C_{n-1}^l + w x_n y_{n-l} \text{ where } w = 1 - e^{-1/N}. \tag{4.1}$$

$$(X \star Y)_n^l = \sum_{i=0}^{L-1} x_{n-i} y_{n-l-i}. \tag{4.2}$$

Recasting the $(1-w)$ factor in Eq. 4.1 into the exponential decay term $e^{-1/N}$, with N a positive integer, shows that N is the number of samples processed after which the previous C_{n-N}^l output contributes w/e of its value to the current output. Therefore, N plays a role somewhat analogous to the correlation length L in standard time domain CC. The value of N may be chosen to correspond to the signal duration or signal features and for $N \gg 1$ $w \sim 1/N$.

Notice that the number of operations required to compute C_n^l is independent of N, involving at most 5 operations, but since $(X \star Y)_n^l$ is a dot product of length L, standard CC requires $2L-1$ operations. We should therefore expect the RTCC algorithm to operate faster than standard CC for correlation lengths $L > 3$.

In order to illustrate and characterise RTCC output we will use two time series X and Y as input. In the following descriptions of data we use the standard notation $\mathcal{N}(\mu, \sigma^2)$ to describe a Gaussian distribution, where μ and σ^2 are the mean and variance respectively. In Fig. 4.1 we show the two 1 Hz sampled time series X and Y, which are the sum of Gaussian noise $\mathcal{N}(0,1)$ and Gaussian pulses of the form $g(t) = A\exp(-(t-t_c)^2/2\sigma^2)$. Both pulses have amplitude $A = 5$ and width parameter $\sigma = 40$ s, but have time centre values t_c which centre the X and Y pulse maxima at 600 s and 1400 s respectively. Although we cannot represent a general signal, we suggest that a suitable series of Gaussian pulses might be considered a reasonable approximation to more interesting signals such as sinusoidal pulses.

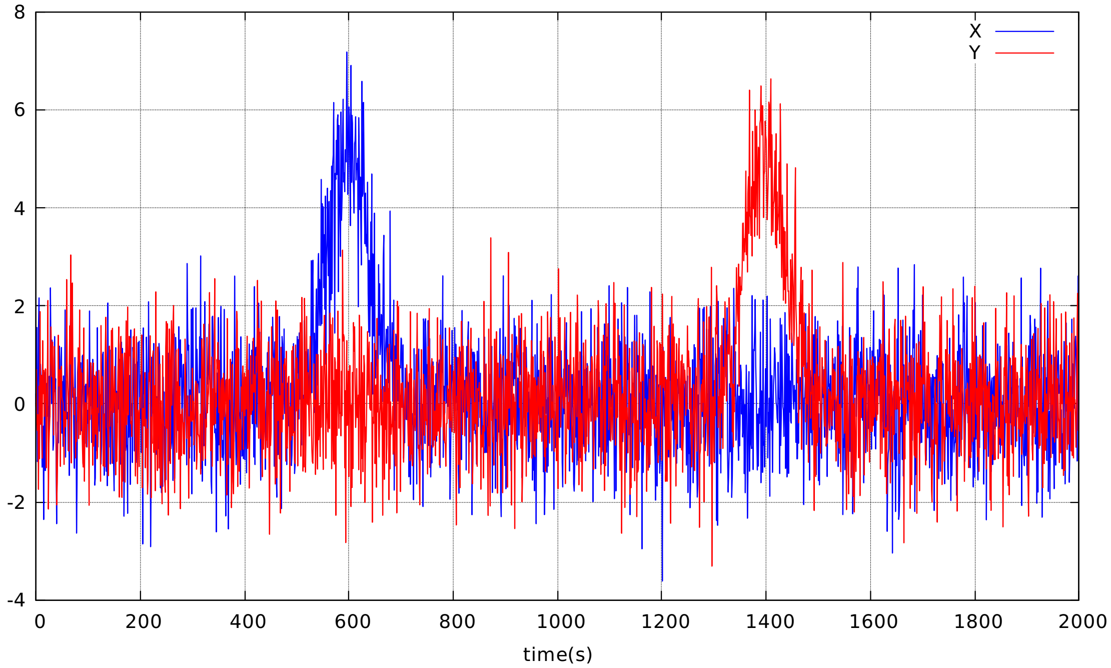


Figure 4.1: Time series X and Y. Both are the sum of Gaussian noise and a Gaussian pulse.

The RTCC output in Fig. 4.2 shows, as one would expect from standard CC, the time when the Gaussian pulse in the X time series strongly correlates with the pulse in the Y time series with a delay equal to the lag (we define positive (negative) lag to mean before (after) X). We have also marked the maximum RTCC value with a green cross.

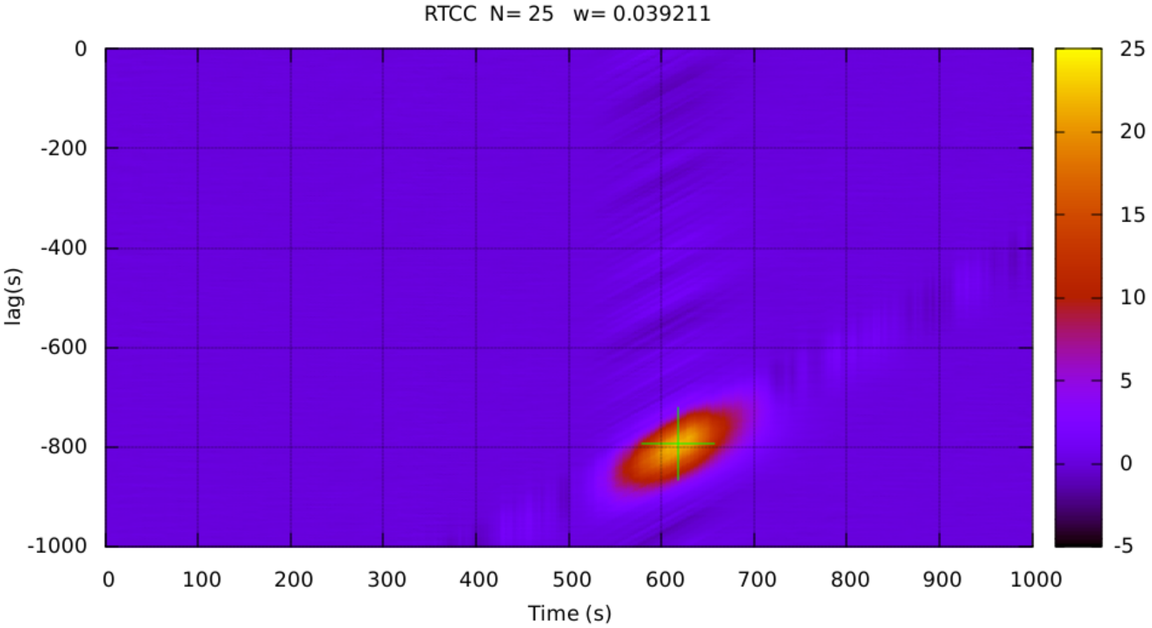


Figure 4.2: RTCC output for time series X and Y in Fig. 4.1. Maximum RTCC correlation of 22.5 is indicated by the green cross with time and lag coordinates (618,−792).

The (time,lag) coordinate of the maximum at (618,−792) corresponds closely to the expected location (600,−800) in the absence of noise. The difference between the lag and pulse central times indicated by the RTCC algorithm and the known lag and pulse timing we refer to as errors. In this case the timing and lag errors are 18 s and 8 s respectively.

The timing and lag errors occur for two reasons: the presence of noise and the exponential nature of the algorithm. Since noise determines the exact location of peaks within the time series, the lag error is dependent upon the particular instances of noise occurring in the neighbourhood of the pulse maxima. The timing error persists in the absence of noise and is then entirely due to the exponential decay rate of the algorithm. This effect is replicated for selected values of N in Fig. 4.3, which plots the RTCC C_t^{-800} output for the product of the X and Y Gaussian pulses (minus noise). The pulses are now effectively aligned and for comparison the −800 lag time-series product $XY = A^2e^{-t/\sigma^2}$ is shown. The figure shows that as N increases so does the timing error of the maximum, the spread of the curve, and asymmetry about the output maximum. The RTCC output maxima all lie on the XY product curve. We can see why this occurs by defining the change in RTCC output at sample n as

$$\Delta C_n^l = C_{n+1}^l - C_n^l = -wC_n^l + wx_{n+1}y_{n+1-l}. \tag{4.3}$$

Therefore, RTCC maxima occur when the output is equal to the next XY product.

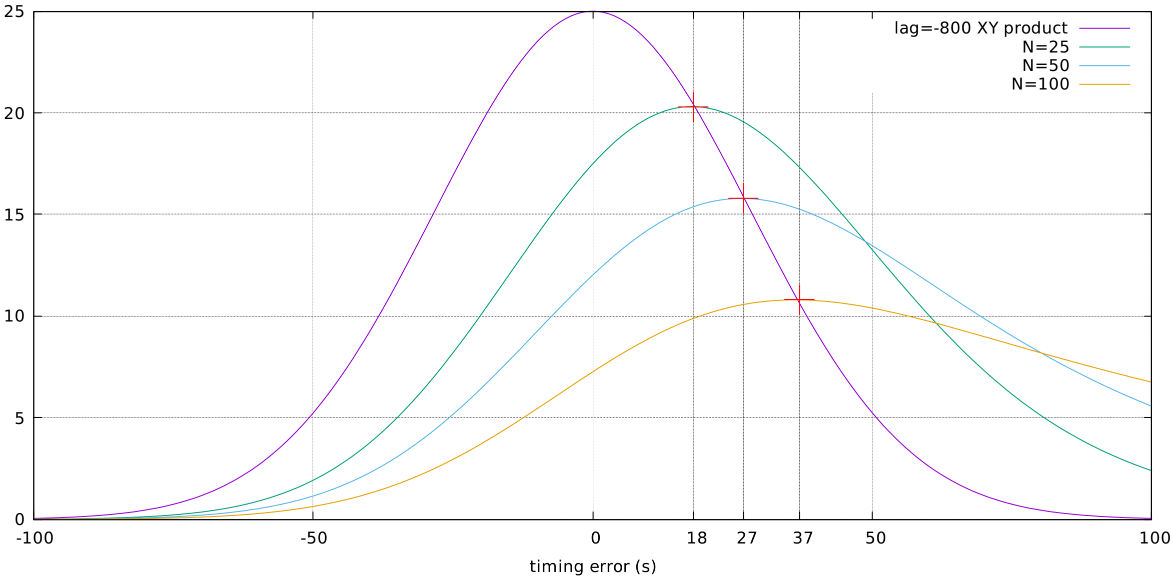


Figure 4.3: RTCC timing error for $N = 25, 50, 100$. Maxima are indicated by red crosses which are found to coincide with the XY product curve.

The asymmetry seen in Fig. 4.3 illustrates, perhaps, the main difference between RTCC and standard CC: standard CC has a memory only as long as its correlation length, whereas RTCC output, most noticeably after a maximum (or minimum), retains past contributions for all time. Once the maximum amplitude is reached then the RTCC output decays exponentially with time constant N. The timing error may appear to be a serious issue: however, in practice it is quite trivial to monitor both the X and Y time series to accurately establish maximum amplitude times.

Our main focus will be on the detection of burst type signals, but we conclude this section with a brief demonstration of RTCC applied to the detection of low signal-to-noise ratio (SNR) continuous signals.

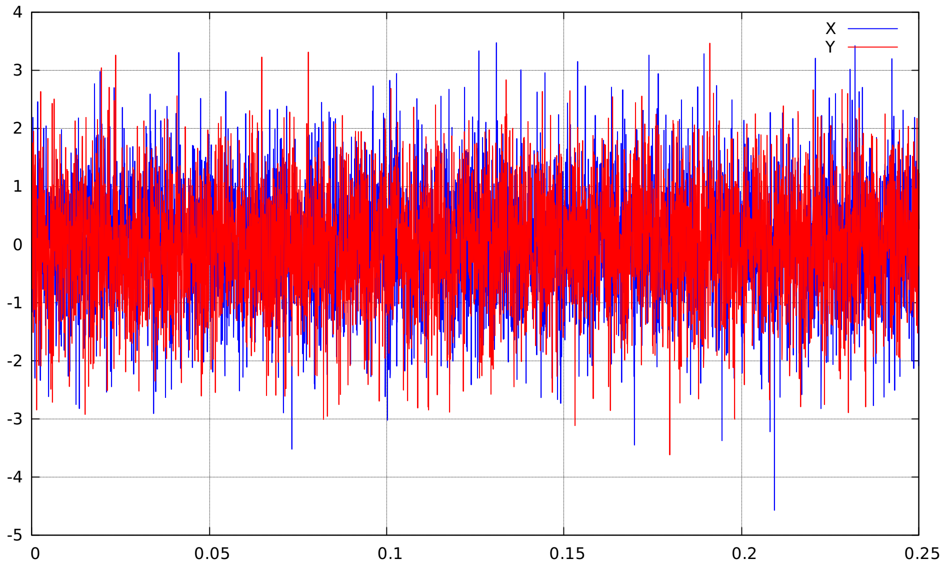


Figure 4.4: Time series X and Y, which are both the sum of a 100 Hz sine wave of amplitude 0.1 (SNR $\sim$ 0.07) and $\mathcal{N}(0, 1)$ noise.

Fig. 4.4 shows a 0.25 s excerpt of two time series X and Y sampled at 16384 Hz, both of which are the sum of a 0.1 amplitude sine wave of frequency 100 Hz and $\mathcal{N}(0,1)$ noise. The RTCC output for 0.25 s of processing the time series is given in Fig. 4.5 and shows no clear presence of of an underlying signal. If we now sum the output at each lag value we find that small correlations build faster than those due to noise alone. The accumulated output for the extended period 10 s is shown in Fig. 4.6, which clearly indicates the presence of a sinusoidal wave of period 0.01 s.

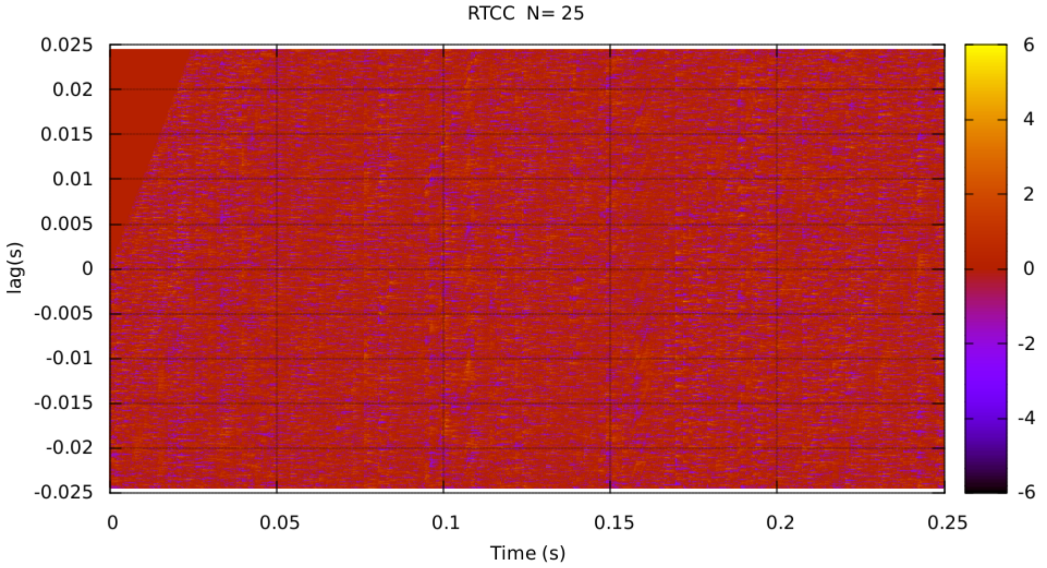


Figure 4.5: RTCC output for 0.25 s of time series X and Y showing no obvious presence of a continuous underlying signal.

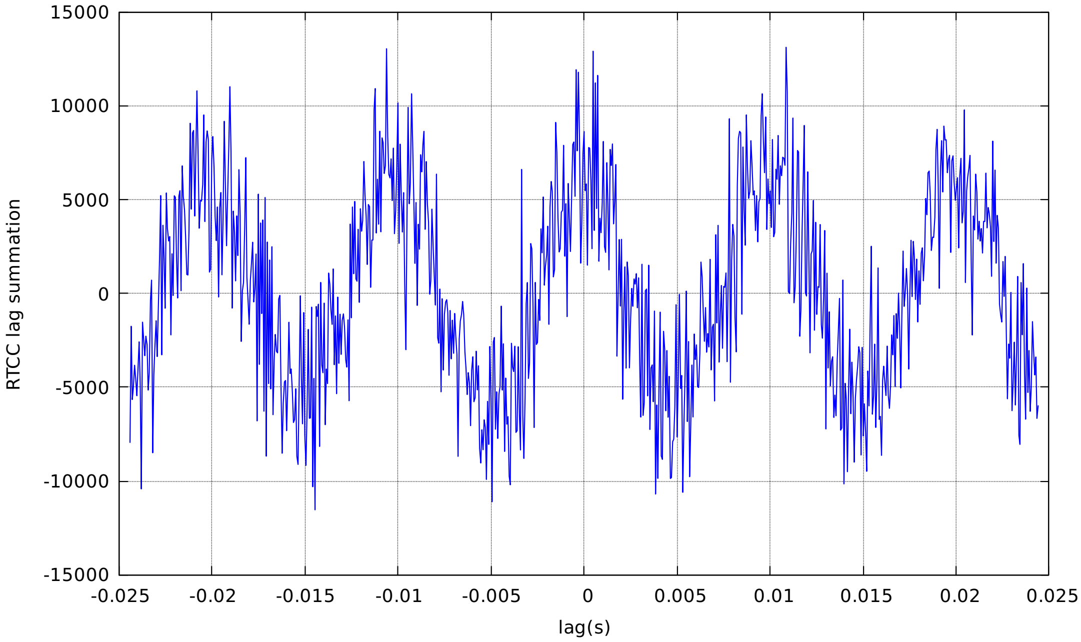


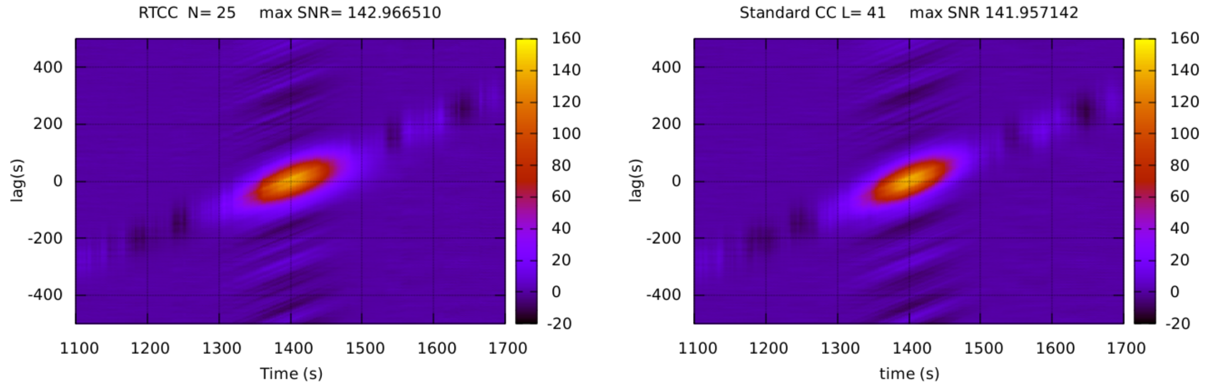
Figure 4.6: Summation of 10 s of RTCC output at each lag for X and Y. This shows the presence of a low SNR 100 Hz sine wave.

4.2 Cross-correlation Indication and Sensitivity

To see how RTCC performs as a CC indicator we now compare the SNR of the output obtained from both RTCC and standard CC upon processing the X and Y Gaussian pulse time series represented in Fig. 4.1. For ease of display we have modified the X time series so that its pulse is now coincident in time with that in Y.

SNR may be expressed in terms of signal power or amplitude; we adopt the latter and so scale the RTCC and standard CC outputs by the expected standard deviation of output for each when processing uncorrelated noise. In Section 4.3 we derive the required expressions for σ_{RTCC} and σ_{STD}.

Given the differences between RTCC and standard CC, we should not expect the output for both to correspond closely when $N = L$, and so for comparison we have chosen the value of $L = 41$ which gives the closest standard CC maximum SNR to that for RTCC when $N = 25$. The scaled outputs for both are shown in Fig. 4.7, where we have made the time correction of the RTCC output maxima referred to in Section 4.1 (see Fig. 4.3). Visually, these look very similar, indicating that the RTCC algorithm is capable of achieving a good approximation to standard CC and demonstrates that the RTCC noise output is constrained to a similar range as that for standard CC.



(a) RTCC output scaled by σ_{RTCC}.

(b) Standard CC scaled by σ_{STD}.

Figure 4.7: Comparison of SNR output between the RTCC algorithm and standard CC.

Our principal interest is in the degree of correspondence between the central maximum correlation regions (SNR $\gtrsim$ 60) which we will refer to as discs. We may compare the outputs above quantitatively by plotting the absolute value of the ratio of RTCC to standard CC SNR output as shown in Fig. 4.8. The absolute value is chosen to flatten the noise output range (i.e make all output positive). We also restrict the ratio plot range to $0 - 2$ for better contrast since the ratio of RTCC and standard CC noise output leads to extreme values. White areas indicate that the ratio was higher than the selected and occur primarily at the lower right hand edge of the disc. We suspect here that relatively high RTCC output extends beyond lower standard CC output, in part due to the asymmetry effect observed in Fig. 4.3 and due to disc non-alignment. The figure shows that the ratio is quite constant and close to unity in the disc region which now indicates a very close correspondence between RTCC and standard CC output.

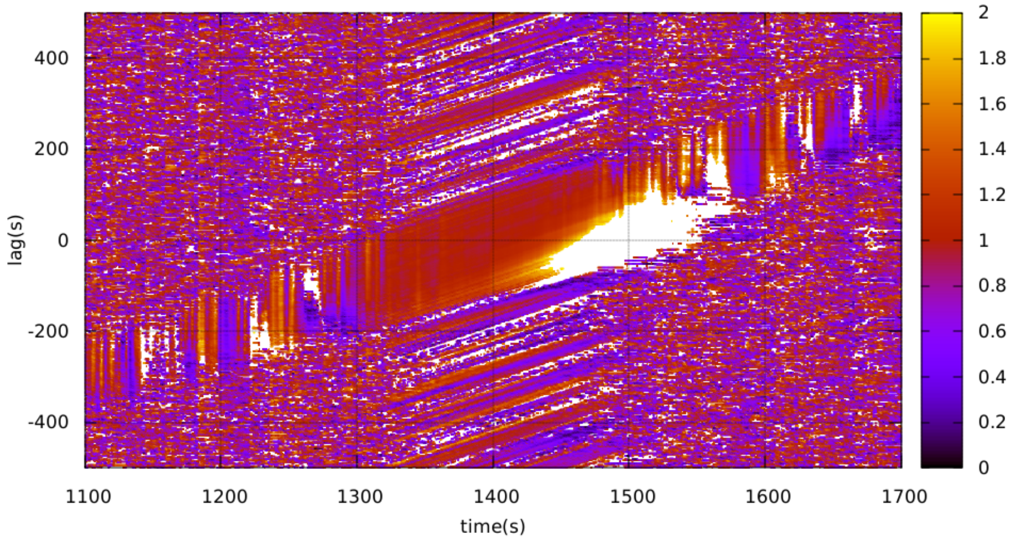


Figure 4.8: Absolute value of the ratio of the RTCC to Standard CC SNR output shown in Fig. 4.7.

We now investigate and compare which values of N/L optimise RTCC and standard CC SNR for the X and Y time series. It would be impractical to produce figures similar to Fig. 4.7 for many values of N and L, so we take the approach used to produce Fig. 4.3, where we consider only the zero lag output. Even though the correlation maximum was in fact 8 s earlier, we still find the results given in Fig. 4.9 to be consistent with output such as Fig. 4.7 for all other values of N/L we compared.

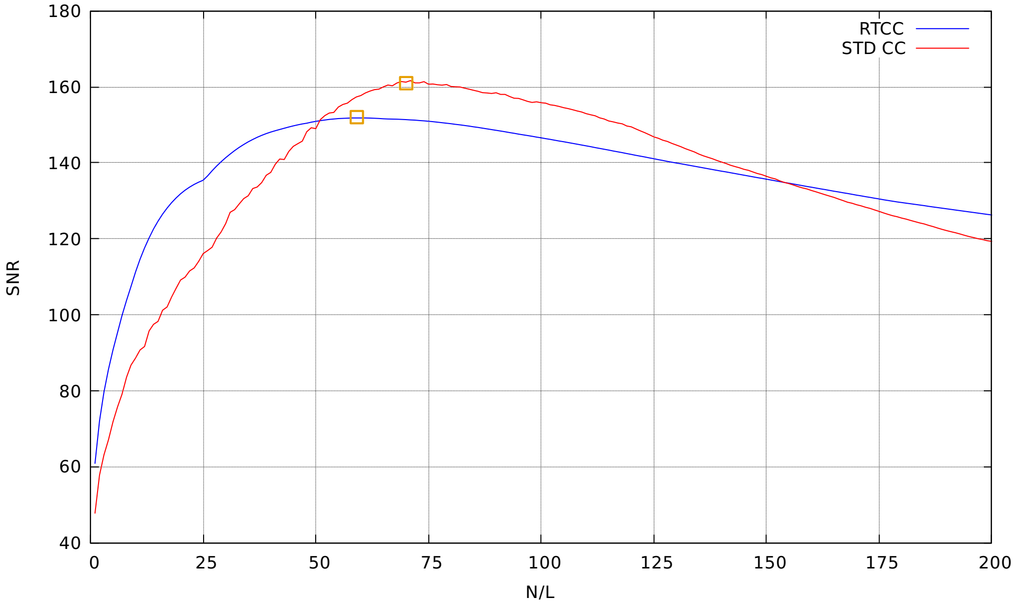


Figure 4.9: RTCC and standard CC SNR on input of time series X and Y for a range of N/L. Maxima for RTCC and standard CC are indicated by the orange boxes at 59,152 and 70,161 respectively.

The figure shows that the RTCC maximum occurs at an N value lower than the optimal

L and RTCC SNR falls off much slower than for standard CC. We find this behaviour to be general and due again to the exponential 'memory' of the RTCC algorithm. The flatter peak for RTCC SNR shows that a broader range of N values give a higher SNR in proportion to the optimal SNR. For standard CC we would expect the optimal choice of L would be proportional to the signal duration. We find that this depends upon the SNR of the signal, which determines the effective duration of the signal amplitude above $\sim 3\sigma$ of the noise Δt_{eff}. An excerpt of the X time series in the neighbourhood of the pulse is shown in Fig. 4.10 which shows that $\Delta t_{\text{eff}} \sim 70$ s close in value to the optimal L value.

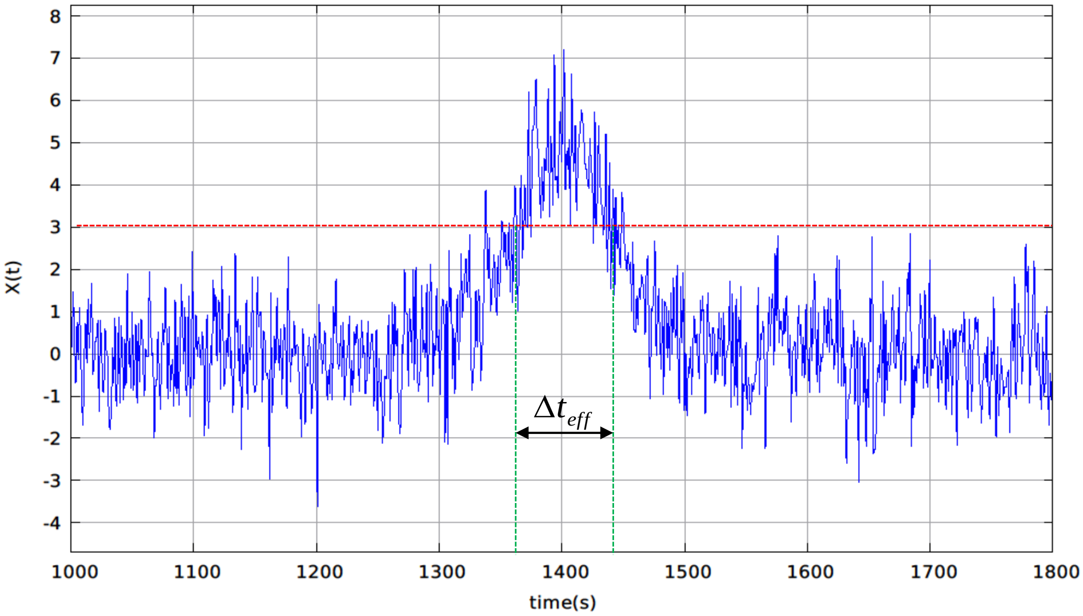


Figure 4.10: Effective length of time series X pulse as number of samples above $3\sigma_x$ of the noise. Δt_{eff} =70 s corresponds to the optimal correlation length L for standard CC shown in Fig. 4.9.

For a signal composed of a train of pulses we should expect the optimal correlation length to correspond to the minimal sample length which covers most of the signal power.

4.3 Gaussianity

In this section we attempt to derive an analytic model of the RTCC output distribution when the input time series are drawn from two separate Gaussian distributions. Our motivation is driven by the convenience of being able to theoretically predict the false alarm rate (FAR) as a function of threshold. We assume that the RTCC output will be approximated by a zero mean Gaussian distribution (via the CLT) and therefore need only model the RTCC output variance. The model obtained will be compared to the normalised histogram distribution of RTCC output on input of pseudo-randomly generated time series X and Y, with distributions $\mathcal{N}(0, \sigma_x^2)$ and $\mathcal{N}(0, \sigma_y^2)$ respectively.

Inspection of Eq. 4.1 shows that the output values are weighted sums of sample products $R = xy$. The probability distribution function (PDF) of the product of two Gaussian distributed random variables is given by (see page 49 of [61])

$$p(R) = \frac{1}{\pi\sigma_x\sigma_y} K_0\left(\frac{|R|}{\sigma_x\sigma_y}\right), \tag{4.4}$$

where K_0 is a modified Bessel function of the second kind. From this one obtains the variance of the products, σ_R^2, as

$$\sigma_R^2 = \sigma_x^2\sigma_y^2. \tag{4.5}$$

Fig. 4.11 shows the distribution $p(R)$ (red) and a normalised histogram of the products of $\sim$ 20 million Gaussian random numbers (blue). $p(R) \to \infty$ as $R \to 0$ and so we curtail the distribution close to $R = 0$.

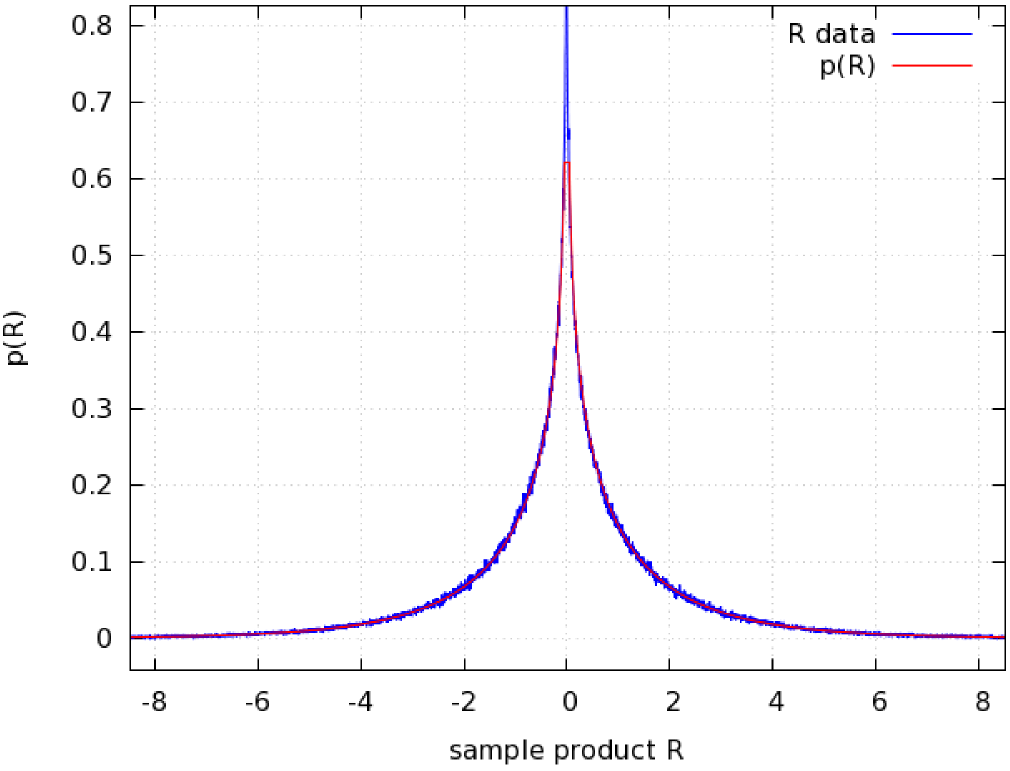


Figure 4.11: Probability density function $p(R)$ (red) and normalised histogram of products of Gaussian data with $\sigma_x = 1$ and $\sigma_y = 2$ (blue).

We make the assumption that the input Gaussian time series are uncorrelated. Therefore, the expected RTCC output distribution should be lag independent and so we ignore lag in the following (i.e. omit the l index). Each iteration of the RTCC algorithm introduces a new sample product and we can therefore determine the distribution of the output by evaluating how the sample products $R_i = x_i y_i$ are weighted following n iterations of the algorithm. We set the initial RTCC output value C_0 to zero and proceed as follows:

$$C_0 = 0 \tag{4.6}$$

$$C_1 = (1-w)C_0 + wR_1 = wR_1 \tag{4.7}$$

$$C_2 = (1-w)C_1 + wR_2 = (1-w)wR_1 + wR_2 \tag{4.8}$$

$$\vdots$$

$$C_n = (1-w)^{n-1}wR_1 + (1-w)^{n-2}wR_2 + ... + (1-w)wR_{n-1} + wR_n. \tag{4.9}$$

In Eq. 4.9 we can see that iterative application of the RTCC algorithm (Eq. 4.1) gives C_n values which are the sum of the products R_i weighted by factors $w(1-w)^{n-i}$, where $i = 0, 1, 2, \cdots, n-1$. Since the products R_i are all from the same distribution, the variance of the resulting distribution should be the product of the variance σ_R^2 and the sum of the square of the weights. The variance of RTCC output as a function of w is then given by

$$\sigma^2_{\mathrm{RTCC}}(w,n) = w^2 \left(\sum_{i=0}^{n-1} (1-w)^{2i} \right) \sigma_R^2 = w^2 \frac{\left(1-(1-w)^{2(n-1)}\right)}{1-(1-w)^2} \sigma_R^2. \tag{4.10}$$

In general the output of the RTCC algorithm will be the result of many iterations of the algorithm over large data sets. This permits simplification of Eq. 4.10 if the condition $(1-w)^{2(n-1)} \ll 1$ is satisfied for large iteration to

$$\sigma_{\mathrm{RTCC}}(w) = \sqrt{\frac{w}{(2-w)}} \sigma_x \sigma_y. \tag{4.11}$$

We can justify this approximation by showing that only modest amounts of input data are required before the $\sigma_{\mathrm{RTCC}}(w,n)$ sample dependence becomes negligible. If we require the relative difference between the variances as given by both expressions to be less than some small arbitrary value $\epsilon \ll 1$, i.e.

$$\frac{\sigma^2_{\mathrm{RTCC}}(w) - \sigma^2_{\mathrm{RTCC}}(w,n)}{\sigma^2_{\mathrm{RTCC}}(w)} = (1-w)^{2(n-1)} < \epsilon, \tag{4.12}$$

in terms of N one can show that n should satisfy

$$n > -\frac{1}{2} N \ln(\epsilon) + 1. \tag{4.13}$$

For example, for $N = 25$ and $\epsilon = 0.01$, Eq. 4.13 gives $n > 58$. In the results that follow, RTCC output for n far in excess of this requirement was used.

As a test of the standard deviation predicted by Eq. 4.11, we now compare this with that obtained from the sample standard deviation of RTCC output for a range of w values. The results given in table 4.1 show agreement to $\sim 99.5\%$.

N	w	σ_{RTCC}	$\sigma_{\mathrm{RTCC}}(w)$
25	0.039210	0.2834	0.2835
50	0.019801	0.2002	0.2005
75	0.013244	0.1633	0.1637
100	0.009950	0.1413	0.1418
125	0.007968	0.1263	0.1268
150	0.006644	0.1152	0.1158
175	0.005697	0.1066	0.1072
200	0.004987	0.0997	0.1002

Table 4.1: Comparison of the measured standard deviation of RTCC output σ_{RTCC} with predicted value $\sigma_{\mathrm{RTCC}}(w)$.

The same argument as above can be used to predict the standard deviation of standard CC output σ_{STD}, where in this case all product weights are unity. For correlation length L we obtain

$$\sigma_{\mathrm{STD}}(L) = \sqrt{\left(\sum_{i=0}^{L-1} 1^2\right)}\sigma_x\sigma_y = \sqrt{L}\sigma_x\sigma_y. \tag{4.14}$$

We omit the results for brevity but state that Eq. 4.14 was found to be in excellent agreement with σ_{STD} observed. This result and $\sigma_{\mathrm{RTCC}}(w)$ (Eq. 4.11) were used to scale the standard CC and RTCC output in Section 4.2.

We now compare the distribution of RTCC output for $N = 25$ with the Gaussian distribution model $\mathcal{N}(0, \sigma^2_{\mathrm{RTCC}})$. The normalised frequency histogram or PDF of the RTCC output for Gaussian distributed random input of two data sets with $\sigma_x = 1$ and $\sigma_y = 2$ and the predicted distribution is shown in Fig. 4.12. We show only the positive half of the distribution and indicate integer multiples of the predicted standard deviation ($\sigma_{\mathrm{RTCC}} = 0.283$) on the correlation axis.

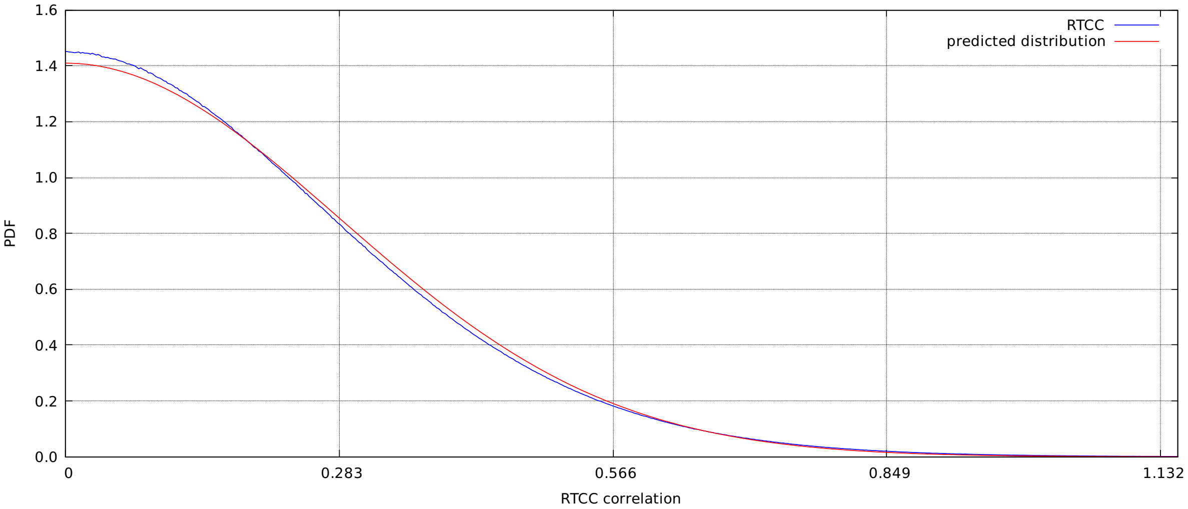


Figure 4.12: . Normalised histogram of RTCC output on processing time series with $\sigma_x = 1$ and $\sigma_y = 2$ compared to model prediction with $\mathcal{N}(0, \sigma^2_{\mathrm{RTCC}})$, where $\sigma_{\mathrm{RTCC}} = 0.283$.

On close examination, Fig. 4.12 reveals two crossover points between the RTCC output distribution and the Gaussian model. In the approximate range $0 - 0.7\ \sigma_{\mathrm{RTCC}}$, the RTCC distribution is significantly higher than the Gaussian model, then is close to but less than the model in the range $0.7 - 2.3\ \sigma_{\mathrm{RTCC}}$ and beyond this again higher. We can emphasise the distribution tail by replotting Fig. 4.12 on a logarithmic scale. This is shown in Fig. 4.13 which shows that the tails in the distribution are poorly modelled. Worse, the tails lie significantly above those predicted and would therefore more frequently reach set thresholds. This means that thresholds would need to be set accordingly higher thereby reducing sensitivity.

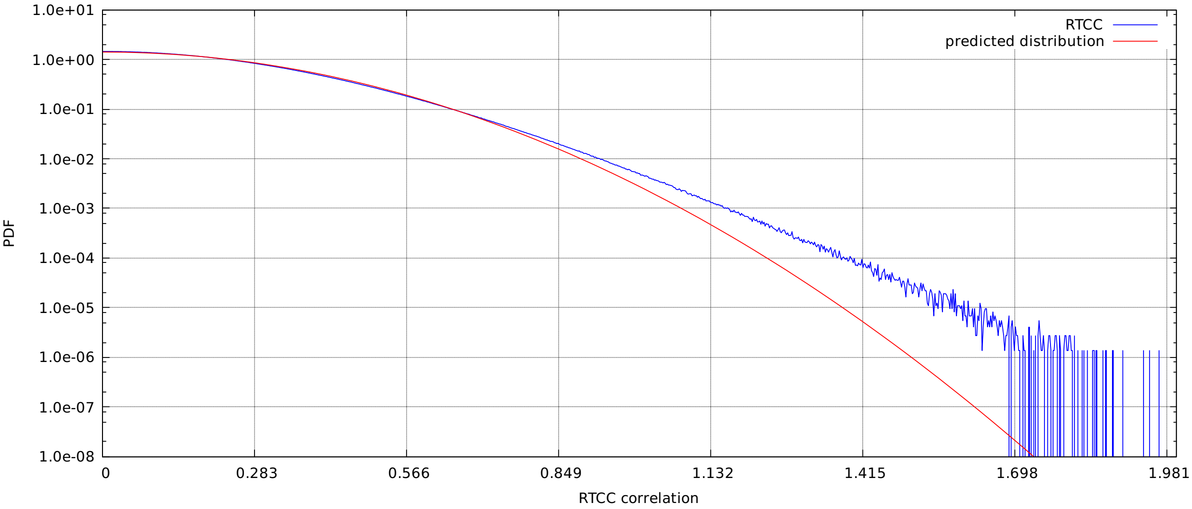
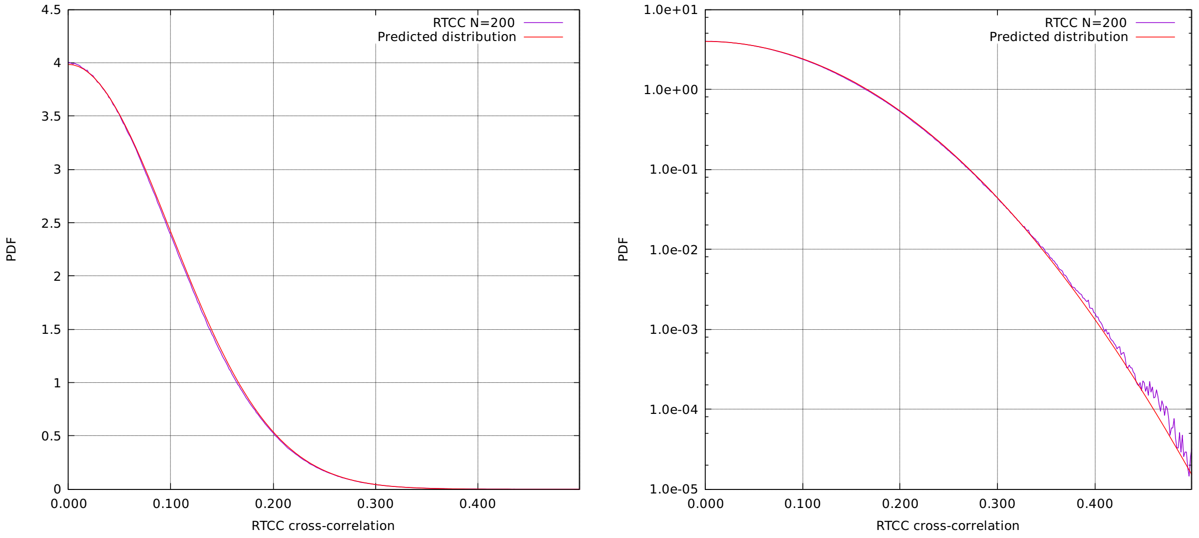


Figure 4.13: Logarithmic plot of Fig. 4.12 to emphasise distribution tails.

We would have liked to predict the false alarm rate (FAR) for a set threshold from a suitable distribution model and so were motivated to seek functions which could provide a better fit to the observed distribution. The most promising functions devised were of the form $f(x) = A\exp(\mathcal{Q}(x))$, where $\mathcal{Q}(x)$ is a symmetric quartic polynomial. It was reasoned that these functions may possess the necessary freedom to produce the excess peak and crossover points observed in Figs. 4.12 and 4.13. We appear to have modelled the standard deviation of the RTCC output successfully but not the exact distribution. Something must be wrong with our assumptions.

On investigation of the distribution of standard CC output for the correlation length $L = 41$ used above, we found that the tails of the distribution were similarly above that of the predicted Gaussian distribution $\mathcal{N}(0, \sigma^2_{\mathrm{STD}})$. In Fig. 4.14 we now make the comparison between the distribution of RTCC output for $N = 200$ and the Gaussian model prediction of $\sigma_{\mathrm{RTCC}} = 0.100$ (see Table 4.1). The agreement is now far better, with elimination of the peak excess and the tails well modelled out to $\sim 4.5\ \sigma_{\mathrm{RTCC}}$. This suggests that the output of RTCC approaches Gaussianity in the limit of large N (i.e longer memory), or in the case of standard CC, large correlation length.

In practice, RTCC will process real data whose noise character is unlikely to be neither stationary nor exactly Gaussian, so we abandoned trying to model the tails of the distribution in favour of measuring the FAR.



(a) RTCC distribution

(b) log scale distribution

Figure 4.14: Comparison of distribution of RTCC output for $N = 200$ (blue) with Gaussian model (red). The RTCC output is well modelled out to $\sim 4.5\ \sigma_{\text{RTCC}}$.

4.4 Frequency Response

Comparison of the second order section (SOS) difference equation (Eq. 3.12) with the RTCC algorithm (Eq. 4.1) shows that we may use the SOS model to obtain the frequency response of the RTCC algorithm. If we set $b_0 = w$, $a_1 = 1 - w$ and $b_1 = b_2 = a_2 = 0$ in the SOS frequency response Eq. 3.14, after some manipulation we obtain

$$|\mathcal{H}(f)| = \frac{w}{\sqrt{w^2 + 2(1-w)(1-\cos(2\pi f/f_s))}}. \tag{4.15}$$

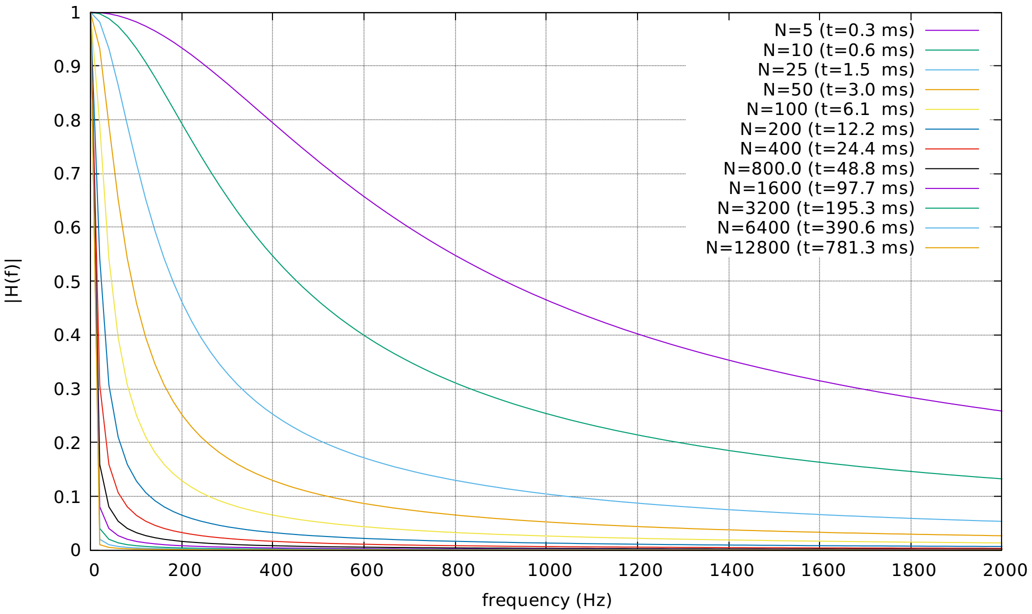


Figure 4.15: Frequency response of RTCC algorithm for selected values of N.

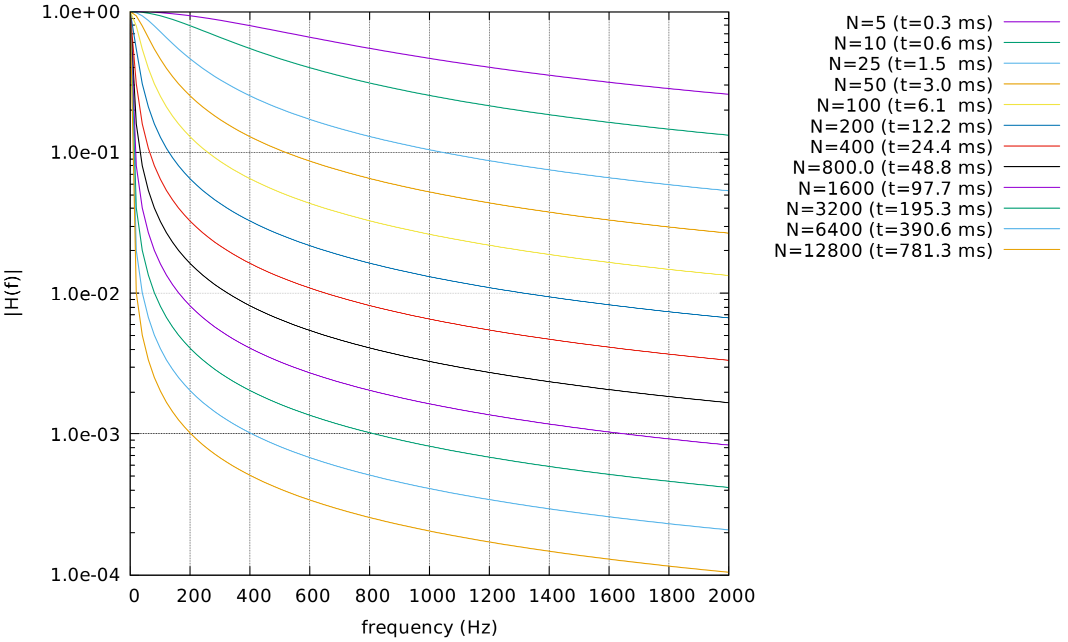


Figure 4.16: Logarithmic plot of Fig. 4.15.

In Fig. 4.15 and the logarithmic plot Fig. 4.16 above is shown the RTCC frequency response for a range of values of N. These figures show that as N increases the RTCC suppression of higher frequency output also increases and becomes broader.

As a test of the $|\mathcal{H}(f)|$ model we may compare the ASD of RTCC processed noise with the predicted frequency response. For ease of comparison the standard deviation of the noise is chosen such that the 16384 Hz sampled noise ASD is unity, i.e. $\sigma = \sqrt{\Delta f}$. Fig. 4.17 shows this comparison for $N = 25$ where the ASD $\Delta f = f_s/2 = 8192$ Hz.

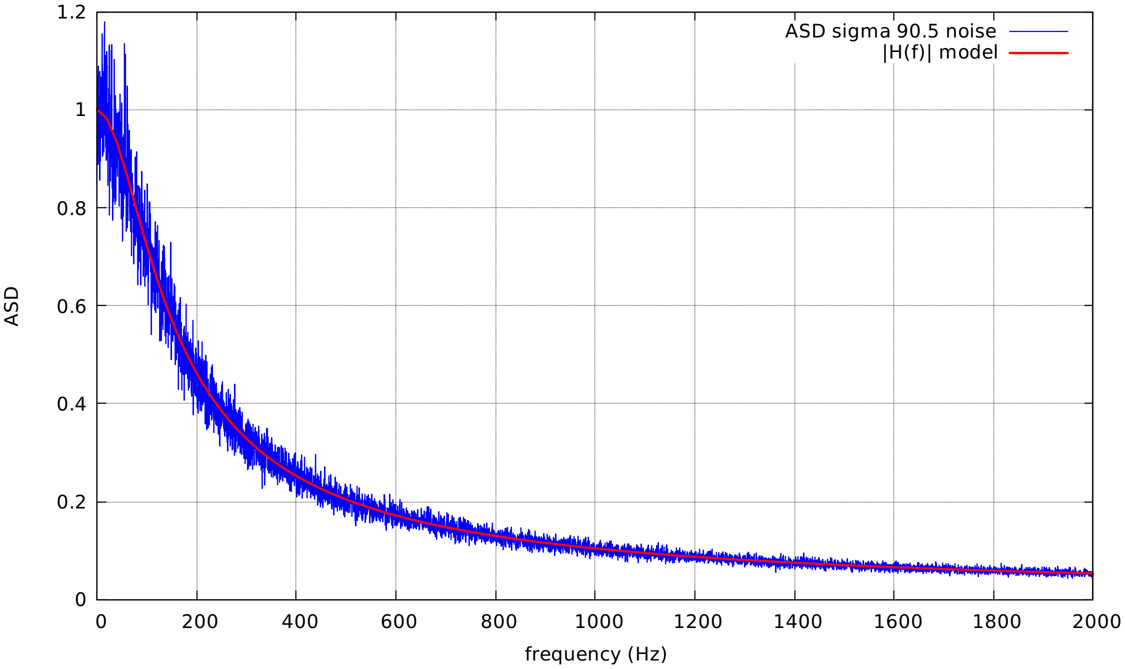


Figure 4.17: Comparison of frequency response model with ASD of RTCC output for $N = 25$ when processing noise $\mathcal{N}(0, 8192)$ scaled to have unfiltered ASD value of 1 $\text{Hz}^{-1/2}$.

The agreement is excellent over the full 0 – 8192 Hz range of the ASD but we show the reduced 0 – 2000 Hz range for clarity.

Performing the inverse scaling on $|H(f)|$, now for general noise input $\mathcal{N}(0, \sigma^2)$, we can express the ASD of RTCC output as

$$\sqrt{S(f)} = \frac{\sigma}{\sqrt{\Delta f}} |H(f)|. \tag{4.16}$$

Eq. 4.16 permits a derivation of the standard deviation of RTCC output σ_{RTCC} independent from the assumptions made in Section 4.3. We spare the reader the details of the calculation but confirm that with $\sigma = \sigma_x \sigma_y$

$$\sigma^2_{\text{RTCC}} = 2\sigma_x^2\sigma_y^2 \int_{-f_s/2}^{f_s/2} \frac{w^2}{w^2 + 2(1-w)(1-\cos(2\pi f/f_s))} df = \frac{w}{(2-w)} \sigma_x^2 \sigma_y^2, \tag{4.17}$$

which is consistent with Eq. 4.11.

Chapter 5

RTCC Pipeline and Mock Data Challenge

Having described the basic processes involved in the development of a GW analysis pipeline and characterised the RTCC output when processing noise and Gaussian pulses, we now present in detail the pipeline developed to test the RTCC algorithm detection sensitivity and reliability.

A common method of testing and comparing pipelines is to measure the detection efficiency of signals added to the detector data. These are referred to as 'injections' and may be performed during detector operation by excitation of detector mirrors via control systems (hardware injection) or added to the detector output by software (software injection) whilst preserving the original output. The injected waveforms may be based on GW emission models or selected for particular characteristics, but in either case the injections simulate the detector response for a given waveform source distance and sky location.

For the development and study of the RTCC pipeline we have used archived data from the LIGO 4 km detectors at Hanford (H1) and Livingston (L1) and simulated burst waveforms from the LIGO Mock Data Challenge (MDC), generated for the purpose of testing LIGO analysis pipelines. The RTCC pipeline will process the sample sum of archived detector and MDC data from which we will measure the detection efficiency and false alarm rate of the pipeline. We begin with a brief description of the main pipeline goals.

5.1 RTCC Pipeline Objectives

The RTCC pipeline employs an unmodelled search method, where the presence of a significant signal in the data from both detectors may be indicated by the RTCC algorithm. The pipeline will monitor the RTCC algorithm output C_n^l for all sample lags corresponding to within plus or minus the light travel time between the H1 and L1 detectors ($\sim$ 10 ms), equivalent to ± 180 samples (at 16384 Hz). Following an event trigger, we attempt to capture the event signals by recording and monitoring both detector data to determine the event lag and define the trigger end. Immediately following the trigger end a post trigger analysis will use the determined lag to align the recorded data for computation of the normalised standard CC of the trigger and estimate the dominant frequency in the trigger data from each detector. These will be our main discriminants when we attempt to reject triggers caused by noise. We also wish to measure the sensitivity of the pipeline at trigger thresholds close to the noise floor and accept the challenge of rejecting false alarm triggers with the aim of achieving false alarm rates of $\sim 1/\text{year}$. Before describing the RTCC pipeline we now address detector data access and the software infrastructure required.

5.2 LIGO Data

The LIGO Data Grid (LDG) provides Ligo Scientific Collaboration (LSC) members access to archived detector data from the collaboration detectors LIGO (H1,H2,L1), VIRGO (V1) and GEO600 (G1). The data are stored in a common format agreed upon by LSC members known as a Gravitational Wave Frame (GWF) [62]. GWFs store data in channels containing output from detector instrumental and environmental monitors. GWF data are timestamped in Global Positioning System (GPS) seconds, which is the time elapsed since 6 Jan 1980 00:00 UTC (Coordinated Universal Time) as measured by atomic clock. Whilst this data is acquired, LIGO operates various data quality monitoring tasks which trigger when there are particular issues which affect the reliability of the data for GW analysis. These periods are marked with data quality flags, the category of which indicates the severity of the issue. The particular analysis being undertaken will determine which of these flags are relevant to the exclusion of detector data (cuts). Further cuts, to remove intervals when either detector or environmental issues degrade the data quality, are made using Veto Definer Files from which one can generate science segment lists for each detector. Periods deemed suitable for GW analysis are referred to as ‘science quality’ data. The remaining science segments for analysis are those periods common to all the detectors belonging to the network analysis.

5.2.1 GWF Data Access

LIGO has developed many GW analysis software tools which are contained in LALSuite (LSC Algorithm Library). The LAL software command `gw_data_find` may be used to return a list of the LDG directory paths of GWFs of specified type containing data from science segments during a specified time interval. The frame types used in this study were the version 2 calibrated frames `H1_LDAS_C02_L2` and `L1_LDAS_C02_L2` for the detectors H1 and L1. Our interest is in the detector strain channels `H1:LDAS-STRAIN` and `L1:LDAS-STRAIN`, sampled at 16384 Hz, where GW signals may reside and into which our MDC injections are to be made.

The initial task is to locate the H1 and L1 GWFs, typically 128 s in length, on the LDG for each run period. We developed code, unimaginatively called `DATAFIND`, incorporating the command `gw_data_find`, to locate the GWFs and write the LDG directory paths in frame time order to file. For example,

```
datafind -o H -t H1_LDAS_C02_L2
        -s 954115215 -e 961977615
        -out H1_954115215_961977615.txt
```

returns in `H1_954115215_961977615.txt` the required list of LDG frame paths for H1. Essentially the same procedure is followed for the MDC GWFs. Occasionally the time ordered GWF paths are not contiguous between the end of a GWF and the start of the next GWF and so `DATAFIND` reports these gaps which must be taken into account.

5.2.2 Data Quality

We must now produce files which contain lists of periods which are of at least the minimal quality acceptable for GW analysis for each detector. We have used version 4 of DMT science segments `H1:DMT-SCIENCE:4`, `L1:DMT-SCIENCE:4` which we obtain using the LAL command `ligolw_segment_query`. The output segment files, returned in `xml` format, may then be converted to plain text files via the `ligolw_print` command.

These science segments may be further refined in quality by removal of periods for which the data quality categories or flags `CAT1`, `CAT2` and `CAT4` applied to each detector. The decision to use these flags was based upon LIGO S5 burst criteria [63]. The flag definitions relevant to the LIGO S6 data are given as follows [64]:

- `CAT1`
  Periods during which severe detector issues exist, including zero detector output.
- `CAT2`
  Noise entering into detector strain and auxiliary channels where the coupling mechanism is understood.
- `CAT3`
  Noise entering into detector strain and auxiliary channels where the coupling mechanism is not understood.
- `CAT4`
  Noise entering into detector strain and auxiliary channels where the coupling mechanism is poorly understood.

Lists of periods when the data quality flags are active for each detector are produced using the LAL command `ligolw_segments_from_cats`, with input from a veto definer file for the pipeline run period concerned, with output in separate files for each category and again in `xml` format. These are then converted to plain text files.

We then subtract the `CAT1,2` and `CAT4` periods from the DMT science 4 data. The final step is to produce the intersection of both files which leaves the run segments. Segmenting the data may leave science segments which are too short for data analysis, especially when filter transient periods must also be subtracted. We have chosen a minimal segment length of 100 s which permits the detection of one injection. Segments shorter in length are removed by the analysis pipeline prior to the run.

LIGO operates ongoing testing of pipelines with both hardware and software signal injection periods. These periods must also be removed either before or after the analysis. We chose to account for these in post processing and the injection times were passed to the post run analysis code in file form.

5.3 MDC GWFs and Injection Waveforms

The GWFs used for injection were the burst MDC data sets configured for injection times during the joint LIGO science run 6 and VIRGO science run 2 (S6/VSR2). The GWF resource is named S6/VSR2 All Sky Sine-Gaussian, Gaussian, White-Noise Linearly Polarised Burst MDC and these frames contain a separate channel for each detector. The MDC injection frames contain short duration burst signals which are separated in time by 100 s plus or minus a random time within 10 s with zero amplitude in-between. The MDC resource also provides injection log files for each period. These are headed with summary information which give the number of injections of each particular waveform and then lists each injection in time order, giving injection parameters such as waveform and injection time for each detector.

The MDC injection waveforms are of various types, Gaussian pulses (GA), sine-Gaussians (SG), and white noise bursts (WNB) (short duration band limited Gaussian noise). These signals have been simulated by the Gravitational-wave Engine (GravEn) software which reproduces the response of a network of detectors to GWs of specified type, distance, sky position and other parameters [65]. The MDC injection set name reflects that they are burst injections (BRST) and cover the particular quarterly period in LIGO science run 6 (S6) shown in Table 5.1. Each analysis run was divided into those periods with the exception that BRST_S6_10Q3 was divided into two runs due to a 62592 s H1 data gap between GPS 965353984-965416576.

MDC GWF set	GPS period	Calendar period (UTC times)
BRST_S6_10Q1	946339215 - 954115215	1 Jan 2010 00:00:00 - 1 Apr 2010 00:00:00
BRST_S6_10Q2	954115215 - 961977615	1 Apr 2010 00:00:00 - 1 Jul 2010 00:00:00
BRST_S6_10Q3	961977615 - 969926415	1 Jul 2010 00:00:00 - 1 Oct 2010 00:00:00
BRST_S6_10Q4	969926415 - 977875215	1 Oct 2010 00:00:00 - 1 Jan 2011 00:00:00

Table 5.1: MDC injection GWF sets and GPS periods.

Our focus will be on the detection efficiency of SGs. The SG waveform is a sine wave with a Gaussian function as amplitude and may be characterised by the central frequency f_c of its bandwidth Δf and quality factor Q. Q is proportional to the number of prominent oscillations in the waveform. In terms of f_c, Q, location t_0 and phase ϕ_0 the generic form of a sine-Gaussian may be expressed as

$$h_{SG}(t) = h_0 \sin(2\pi f_c(t-t_0)+\phi_0)e^{-2(t-t_0)^2(\pi f_c/Q)^2}. \tag{5.1}$$

SGs are not of astrophysical significance but are easy to characterise and define spectrally [66]. The bandwidth Δf of a SG is given by the ratio f_c/Q and after setting both $t_0 = 0$ and $\phi_0 = 0$ the root-sum-square amplitude is given by [67]

$$h_{\text{rss}} \equiv \sqrt{\int_{-\infty}^{\infty} |h(t)|^2 dt} = h_0 \left(\frac{Q}{4f_c}\right)^{1/2} \pi^{-1/4}. \tag{5.2}$$

In Fig. 5.1 is shown a plot of the SG waveform SG361Q8d9 from the MDC set with file names in the format SG(f_c)Q(Q) where d signifies a decimal point.

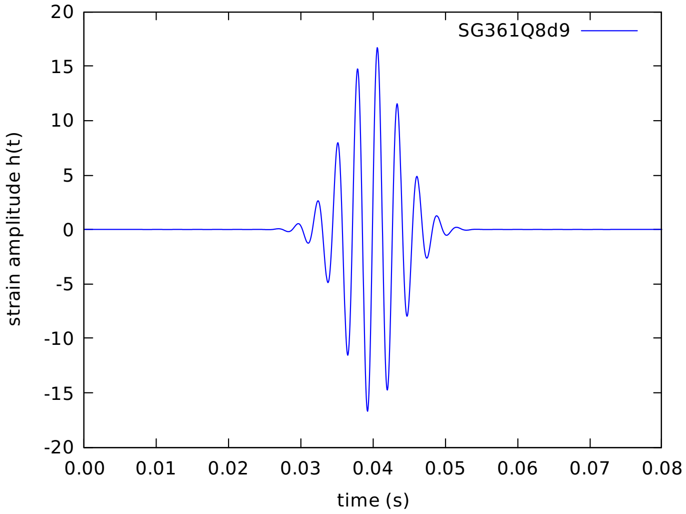


Figure 5.1: Sine Gaussian waveform with f_c=361 Hz and Q=8.9.

All SGs in the injection sets have an h_{rss} of 2.5×10^{-21} $\mathrm{Hz}^{-1/2}$. During each separate run the injection data will be multiplied by one of a range of scale factors prior to adding this to the detector GWF data. Scaling the injection data permits emulation of a range of source strengths and distances to measure detection sensitivity. We have used a total of 31 scale factors ranging from the highest at 1000 and decreasing exponentially to 0.001, which scale the injection h_{rss} from 2.5×10^{-18} to 2.5×10^{-24} $\mathrm{Hz}^{-1/2}$.

SG Waveform	f_c	Q	Δf	f_-	f_+	ΔT	N_s
SG70Q3	70	3	23.33	58.3	81.7	0.04286	702
SG70Q8d9	70	8.9	7.86	66.0	74.0	0.12723	2084
SG70Q100	70	100	0.70	69.7	70.4	1.42857	23405
SG100Q8d9	100	8.9	11.23	94.4	105.6	0.08905	1459
SG153Q8d9	153	8.9	17.19	144.4	161.6	0.05817	953
SG235Q3	235	3	78.33	195.8	274.2	0.01277	209
SG235Q8d9	235	8.9	26.40	221.8	248.2	0.03789	621
SG235Q100	235	100	2.35	233.8	236.2	0.42553	6971
SG361Q8d9	361	8.9	40.56	340.7	381.3	0.02465	404
SG554Q8d9	554	8.9	62.25	522.9	585.1	0.01606	263
SG849Q3	849	3	283.00	707.5	990.5	0.00353	58
SG849Q8d9	849	8.9	95.39	801.33	896.7	0.01048	172
SG849Q100	849	100	8.49	844.8	853.3	0.11778	1930
SG1053Q9	1053	9	117.00	994.5	1111.5	0.00855	140
SG1304Q9	1304	9	144.88	1231.6	1376.4	0.00690	113
SG1615Q3	1615	3	538.33	1345.8	1884.2	0.00186	30
SG1615Q9	1615	9	179.44	1525.3	1704.7	0.00557	91
SG1615Q100	1615	100	16.15	1606.9	1623.0	0.06192	1014
SG2000Q9	2000	9	222.22	1888.9	2111.1	0.00450	74

Table 5.2: All MDC SG injection waveforms and parameters: central frequency f_c, quality factor Q, bandwidth Δf, lower bandwidth frequency f_-, upper bandwidth frequency f_+, time width $\Delta T = 1/\Delta f$, and number of samples corresponding to the latter $N_s = f_s \Delta T$.

In Table 5.2 we list all SG injection types in the MDC sets, giving central frequency, bandwidth, and approximate range of spectral content from lower frequency f_- to higher frequency f_+, where $f_\pm = f_c \pm \Delta f/2$. We also include the estimate of the duration of each SG from its bandwidth, $\Delta T = 1/\Delta f$ and the corresponding number of samples N_s. The minimum f_- and maximum f_+ of the SG injection waveforms determine the bandwidth of our search. This gives our search a frequency range from $\sim$ 60 Hz to 2100 Hz to cover all significant spectral content of the SG waveforms.

Given that the SG injections have a broad range of duration, we could run several different pipelines, each with N selected to optimise SNR for a particular signal duration (see section 4.2). This however, would either considerably multiply the computational time required or necessitate modification of the MDC injection types and times. The minimum number of optimal N we could use is one, and in the next section we determine the detection sensitivity just one choice of N for all pipeline runs can provide.

5.4 RTCC Pipeline Sensitivity Dependence on N

In Section 4.2 we determined the value of N which gave maximum RTCC SNR and observed that the SNR peak was substantially broader than seen in standard CC. It is this behaviour of the RTCC algorithm which may permit selection of only one N value and still retain good sensitivity to all the waveforms considered in this analysis. Figs. 5.2, 5.3 and 5.4 show the ratio of the RTCC output maximum to $\sigma_{\mathrm{RTCC}}(N)$ for all waveforms in Table 5.2 grouped by Q value. We have chosen N ranging from 1 to 10000 to include all RTCC maxima. All waveforms were scaled to have an equal maximum amplitude of 35 and added to Gaussian noise $\mathcal{N}(0,1)$.

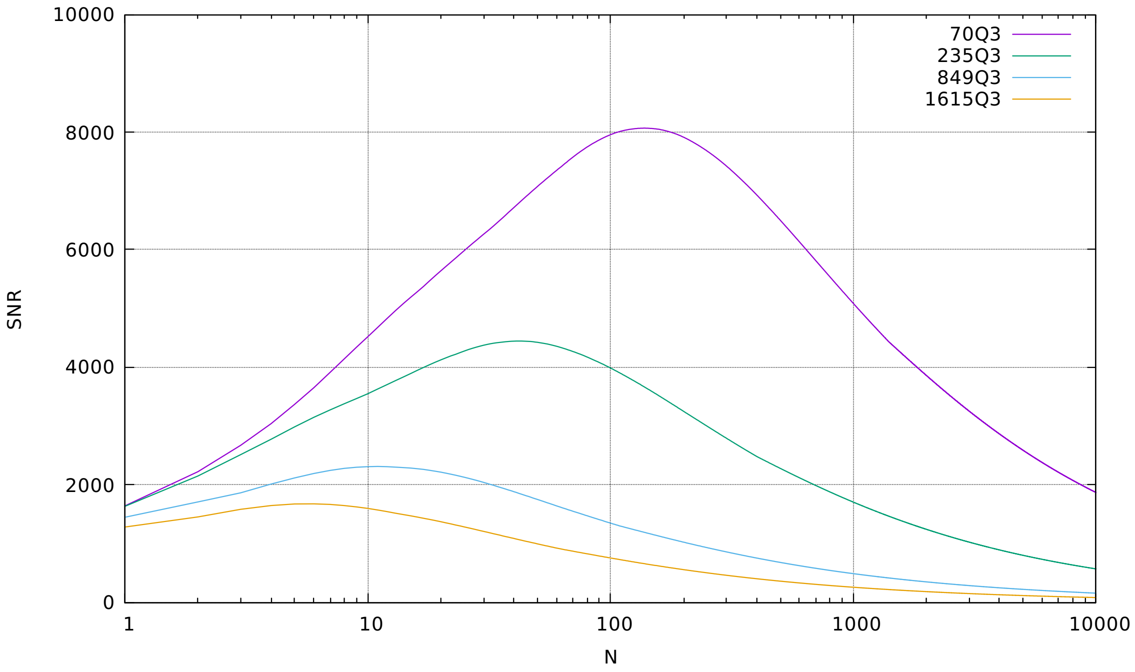


Figure 5.2: RTCC SNR(N) for $Q = 3$ waveforms.

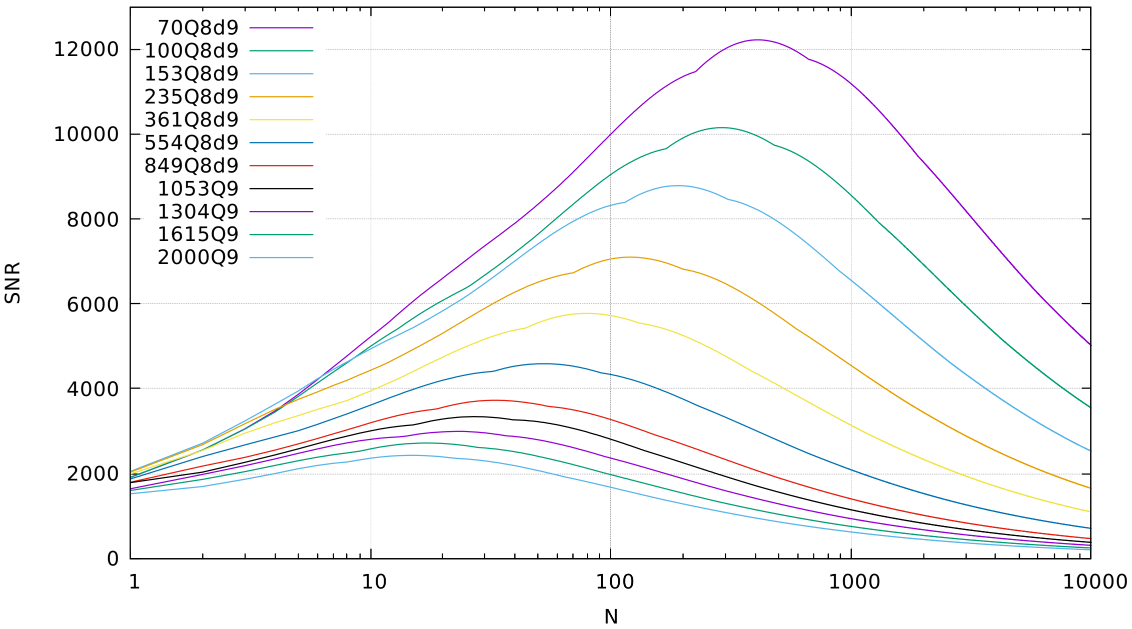


Figure 5.3: RTCC SNR(N) for $Q = 8.9$ and $Q = 9$ waveforms.

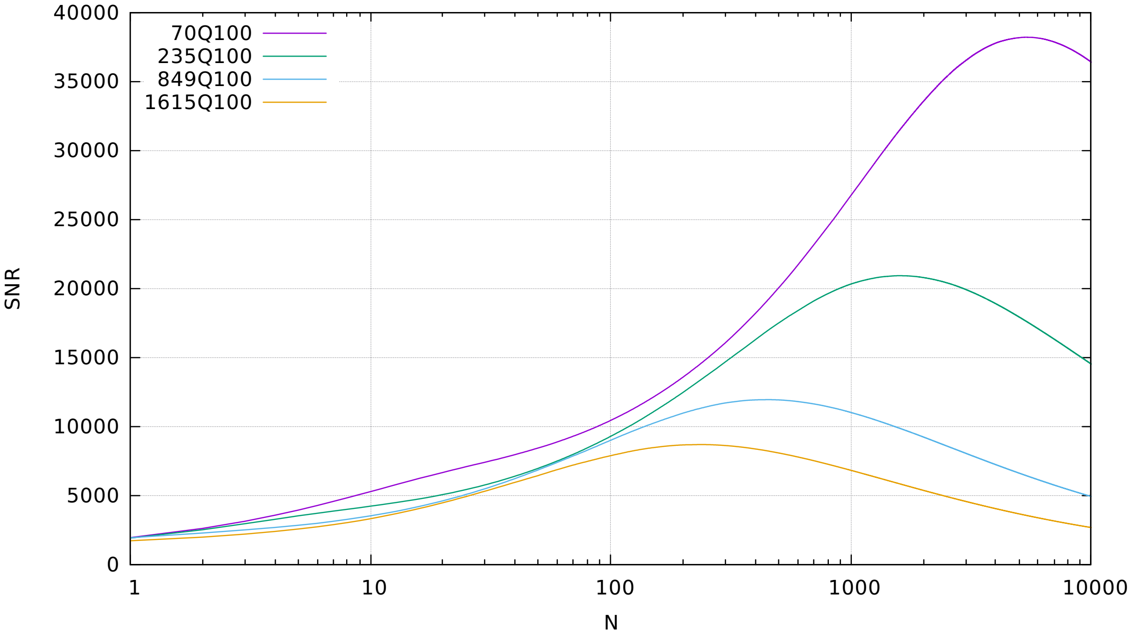


Figure 5.4: RTCC SNR(N) for $Q = 100$ waveforms.

The figures all show the same general trend with increasing SG frequency with both the SNR maxima and corresponding optimal value of N decreasing. Also, as the value of Q increases, the range of SNR and N at the maxima shifts towards higher values. Table 5.3 summarises the results.

SG Waveform	$\mathrm{SNR}_{\mathrm{opt}}$	N_{opt}	N_s
SG70Q3	8067.19	137	702
SG70Q8d9	12219.5	410	2084
SG70Q100	38182.2	5329	23405
SG100Q8d9	10150.6	287	1459
SG153Q8d9	8783.2	190	953
SG235Q3	4446.41	41	209
SG235Q8d9	7098.73	120	621
SG235Q100	20906.2	1629	6971
SG361Q8d9	5773.24	78	404
SG554Q8d9	4587.21	52	263
SG849Q3	2312.16	10	58
SG849Q8d9	3725.52	32	172
SG849Q100	11924.9	452	1930
SG1053Q9	3339.49	26	140
SG1304Q9	2992.4	22	113
SG1615Q3	1677.5	5	30
SG1615Q9	2717.8	16	91
SG1615Q100	8672.52	238	1014
SG2000Q9	2428.13	14	74

Table 5.3: Optimal N values for SG waveforms.

If we define the SNR sensitivity for a particular SG and value of N as the ratio $\mathrm{SNR}(N)/\mathrm{SNR}(N_{\mathrm{opt}})$, then summation of this quantity over all waveforms gives the total sensitivity $\mathrm{SNR}_{\mathrm{SG}}(N)$ with an ideal value equal to the number of waveforms (i.e 19). Fig. 5.5 shows the layered summation of $\mathrm{SNR}(N)/\mathrm{SNR}(N_{\mathrm{opt}})$ for all waveforms, with the last contribution from the SG2000Q9 representing the total $\mathrm{SNR}_{\mathrm{SG}}(N)$.

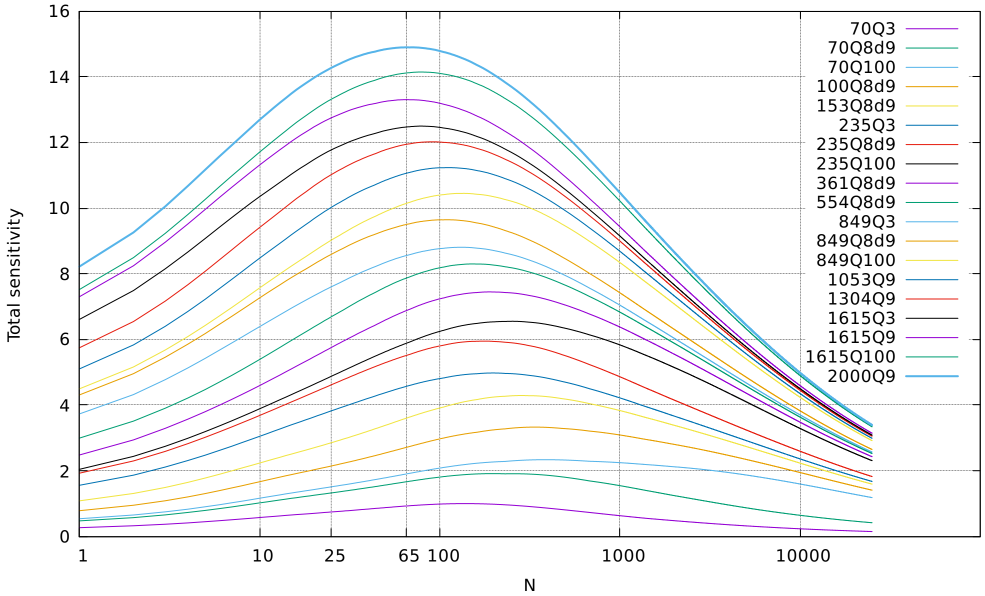


Figure 5.5: Total $\mathrm{SNR}(N)/\mathrm{SNR}(N_{\mathrm{opt}})$ for all waveforms. $\mathrm{SNR}_{\mathrm{SG}}(N)$ represented by the top curve as the last SG (2000Q9) is added. The maximum is located at $N = 65$.

The maximum total sensitivity to all SGs occurs at $N = 65$ and in Table 5.4 we compare this to that obtained when $N = 25$, a value which during pipeline development had already returned promising results prior to undertaking the analysis presented in this section. For $N = 25$ the total sensitivity is 14.3 out of 19 or 75 % and for $N = 65$ we obtain the rather modest improvement of 78 %. For this reason we retained the value $N = 25$ for the pipeline.

	SNR(N)/SNR(N_{opt})	
SG Waveform	$N = 25$	$N = 65$
SG70Q3	0.749	0.926
SG70Q8d9	0.587	0.736
SG70Q100	0.190	0.238
SG100Q8d9	0.633	0.808
SG153Q8d9	0.710	0.889
SG235Q3	0.967	0.969
SG235Q8d9	0.801	0.940
SG235Q100	0.264	0.370
SG361Q8d9	0.871	0.994
SG554Q8d9	0.946	0.990
SG849Q3	0.913	0.688
SG849Q8d9	0.989	0.946
SG849Q100	0.431	0.641
SG1053Q9	0.999	0.917
SG1304Q9	0.997	0.881
SG1615Q3	0.755	0.528
SG1615Q9	0.972	0.832
SG1615Q100	0.575	0.812
SG2000Q9	0.961	0.781
Total	14.317	14.894
Total/19	0.753	0.783

Table 5.4: Comparison of $\mathrm{SNR}_{\mathrm{SG}}(25)$ and $\mathrm{SNR}_{\mathrm{SG}}(65)$ showing a slight improvement in sensitivity for $N = 65$.

5.5 The DMT

The Data Monitoring Tool (DMT) is a suite of software written in C/C++, authored mainly by John Zweizig (Caltech), which provides software tools for signal processing and data monitoring of the detectors [68]. The DMT also provides GWF data access and buffering methods. DMT classes of particular use were GWF data access `Dacc`, time series `TSeries`, `IIRFilter` and `LPEFilter` filters. The software we developed to perform GWF data access and preprocessing techniques was tested by implementing the DMT classes via a ROOT (CERN) [69] interface locally, and once approved, directly on the Caltech cluster. Several mathematical functions available within ROOT have been used for convenience throughout.

So far detector characterisation has been discussed in terms of the ASD; however, the native output of the DMT class used for spectral analysis, `FSpectrum`, is the power spectral density (PSD). The PSD is just the square of the ASD and future discussion will now be in terms of the PSD.

5.6 Preprocessing

The preprocessing procedure we have developed is an adaptation of the broadband whitening method for use with LPEF filters described in [70]. Before discussing our method to achieve broadband whitening, we first briefly describe iWave (Iterative Waveform Action-Angle Variable Estimator), a line monitoring and subtraction tool currently in development at the University of Sheffield [71], our method of choice for line removal.

5.6.1 iWave Line Removal

iWave models the line wave component of the data in terms of the action angle variables (AAV) of the oscillator. In classical mechanics AAVs permit the frequencies of oscillation of systems to be determined without necessitating the solution of the equations of motion [72]. The reconstructed line may then be subtracted from the input data. This technique can be implemented as either a dynamic or static line remover. In the former case, iWave adapts to changes in the frequency, amplitude and phase of the line and in the static case, the mode to be used in our pipeline, iWave accepts as input the data sampling frequency f_s, the frequency of the line for removal f_{line} and the line bandwidth parameter $\tau = 1/(\pi \Delta f)$. Fig. 5.6 shows attenuation of a 180 Hz powerline harmonic by tuning of the τ parameter.

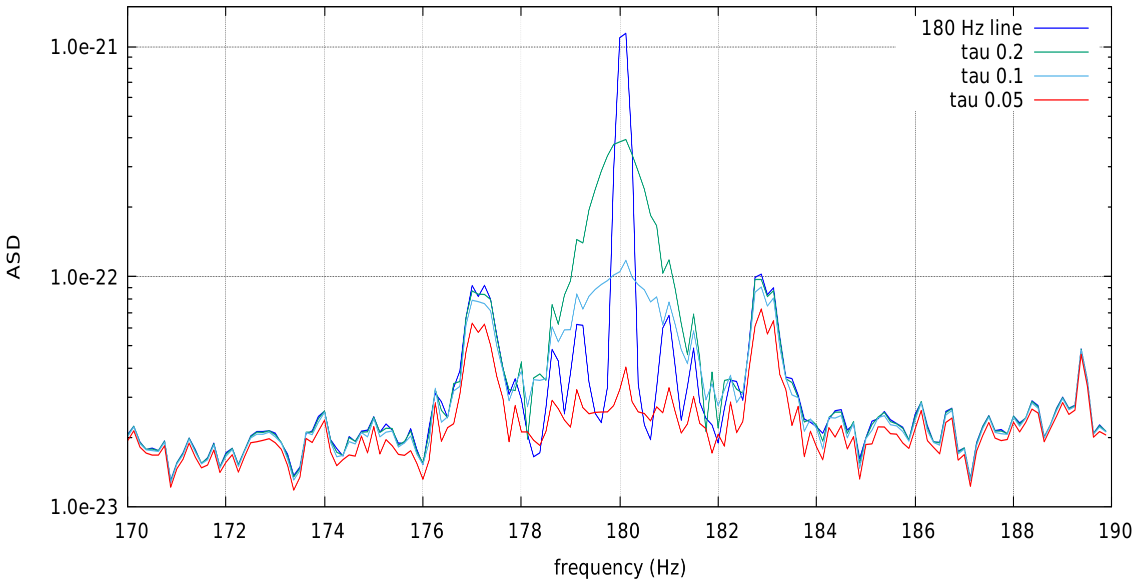


Figure 5.6: Tuning of iWave τ parameter for the removal of 180 Hz mains line harmonic (blue) to give data with significantly reduced amplitude at the line frequency (red).

In addition to line removal to facilitate search analysis, the ability to monitor lines may be useful for detecting causal relationships between lines which are not obvious, as may occur during the commissioning/upgrading of detector instrumentation. iWave was incorporated into the RTCC pipeline line removal procedure to be described in Section 5.6.3.

5.6.2 Initial Preprocessing

In order to limit the spectral content of the detector data to the search frequency range 60 – 2100 Hz we use IIR high-pass and low-pass filters. Having first obtained an IIR filter with the desired frequency response, we can use the `IIRFilter` class to provides access to the filter parameters in terms of the second order section (SOS) coefficients. The pipeline builds the IIR filters from the product of individually created SOSs using the `IIRSos` class and SOS coefficients provided by file input.

Our first preprocessing task will be to high-pass the data at 64 Hz. The PSD of the raw data after high-pass filtering with a 6th order Butterworth filter at 64 Hz is shown in Fig. 5.7.

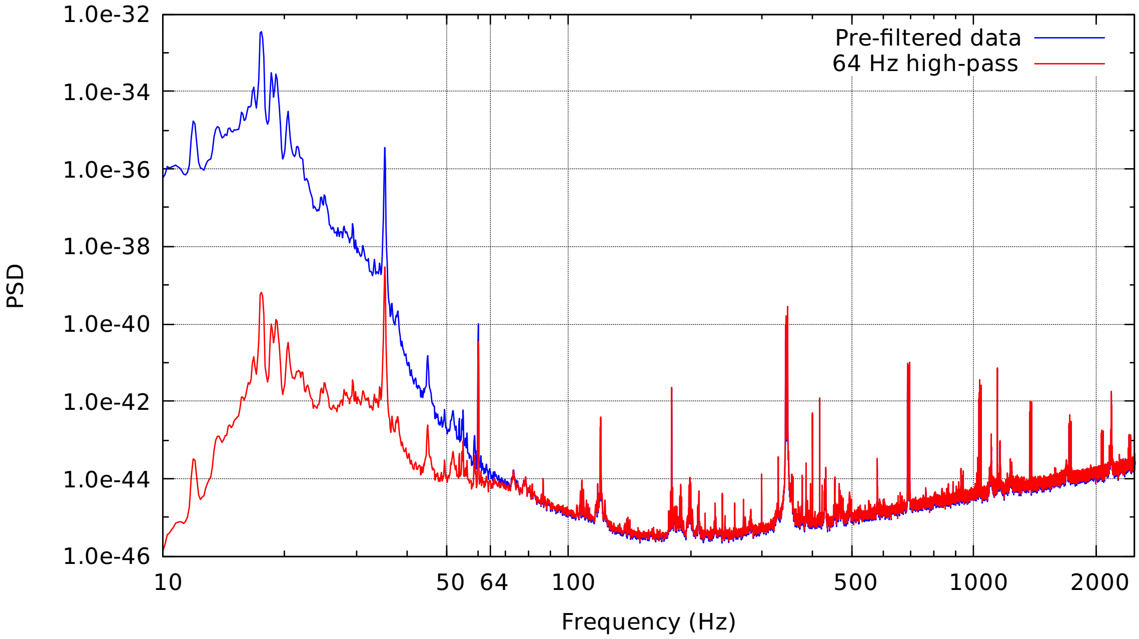


Figure 5.7: PSD of raw H1 strain data (blue) and after high-pass filter at 64 Hz. This levels the power below 64 Hz in preparation for further preprocessing.

In this case we have not cutoff spectral power below 64 Hz but have levelled the spectrum in preparation for the next preprocessing steps.

5.6.3 Line Selection and iWave Line Removal

The PSD of the lines may be compared to the noise power surrounding the line by taking the median baseline average (MBA) of the spectrum. For this we choose a frequency window of sufficient width so that the median value gives a good estimate of the noise floor surrounding the lines. Starting at the low end of the frequency spectrum, lines are detected in the PSD by determining the frequencies at which a negative change of gradient occurs and selected for removal if the ratio of the PSD to MBA power exceeds a particular threshold. The identified lines are then listed in order of descending power relative to the MBA for use by the line subtraction procedure. Fig. 5.8 shows both the MBA (red) and line selection (green crosses) where PSD/MBA exceeds a threshold of 2. The frequency resolution of the PSD was 0.25 Hz and the MBA window chosen for this analysis was 30 Hz. The number of lines identified for removal is 138, with many clustering around the violin modes and harmonics.

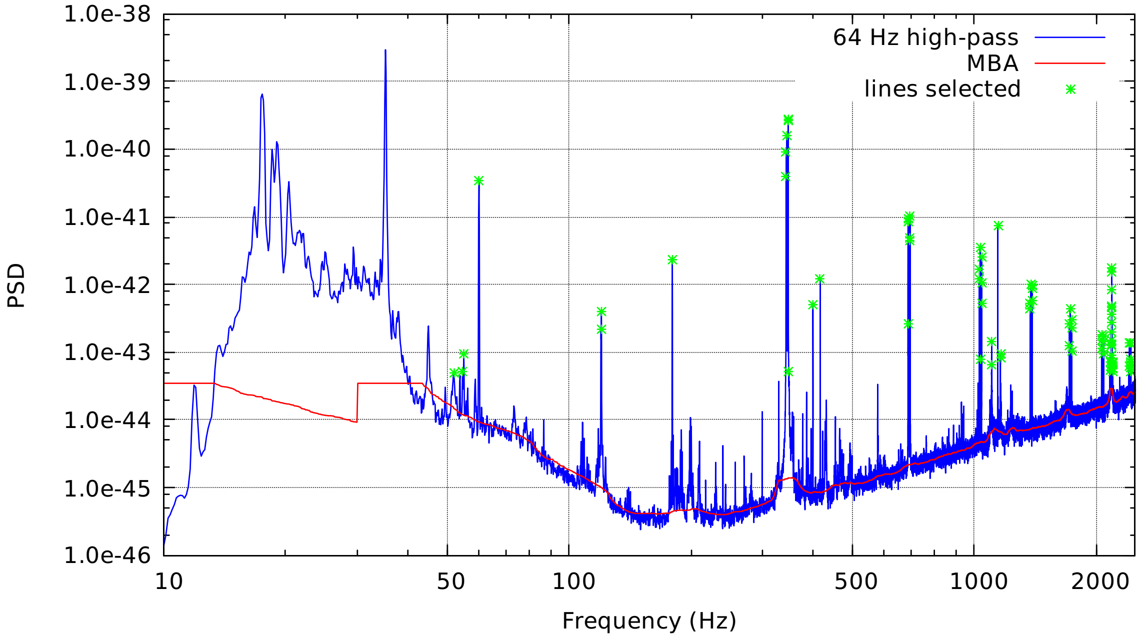


Figure 5.8: 64 Hz high-pass data (blue), MBA (red) and line selection (green crosses).

iWave line removal is preconfigured by passing the line selection list to code which tunes iWave for the removal of lines independently and in list order to produce a list of τ for each line. The list of line frequencies and τ values are then used to create separate instances of the iWave filter for each line. Fig. 5.9 shows the effect of applying all iWave line filters to the data. The figure shows that many of the selected lines have been successfully subtracted, substantially attenuated or when clusters of lines occur, effectively notched. An undesirable degree of notching occurs around the violin modes at around 340 Hz which more than inverts the ASD in a bandwidth of $\sim$ 150 Hz. The degree of notching depends upon the number of lines in the cluster, line power and the bandwidth overlap. The cause of excessive line subtraction is our implementation of iWave. This employed simple independent tuning which uses the original data to be filtered for each line, taking no account of the effect on other lines. Therefore, lines belonging to a cluster can be subject to iWave subtraction multiple times. Our line subtraction procedure is only suitable for data whose lines are bandwidth exclusive. The large violin mode notch is quite a broad feature, and before we considered developing a more adaptive line removal procedure, we were interested to see how the LPEF filter would affect the PSD in this region.

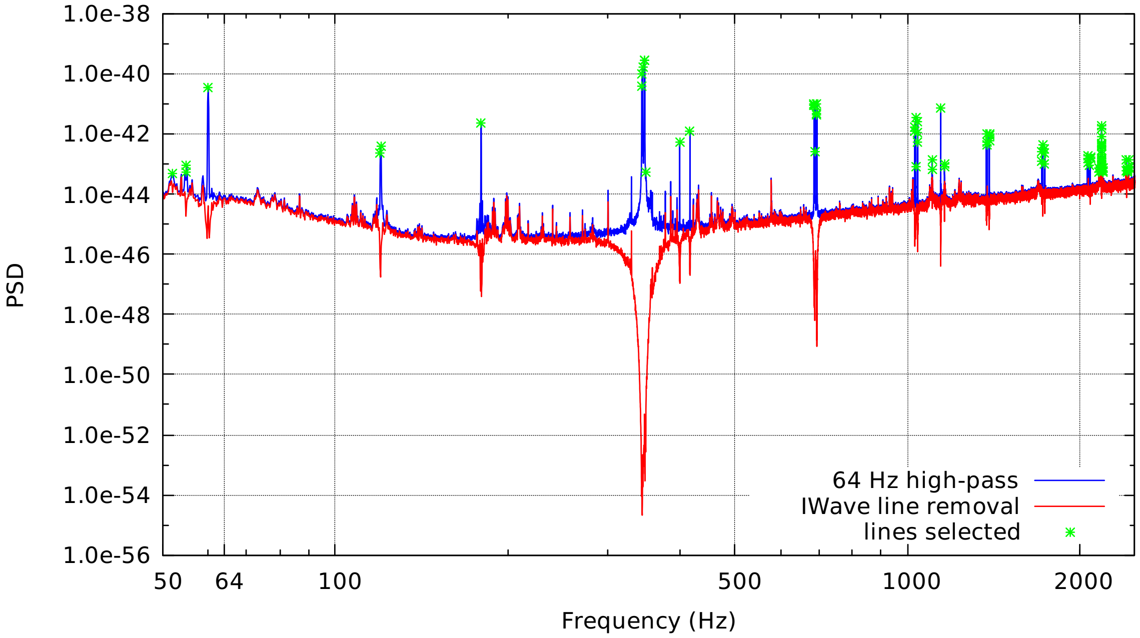


Figure 5.9: Line selection (green crosses). Prior to line removal (blue) and after iWave line removal (red).

The cumulative PSD spectrum following the 64 Hz high-pass filter and after iWave line removal is shown in Fig. 5.10 below.

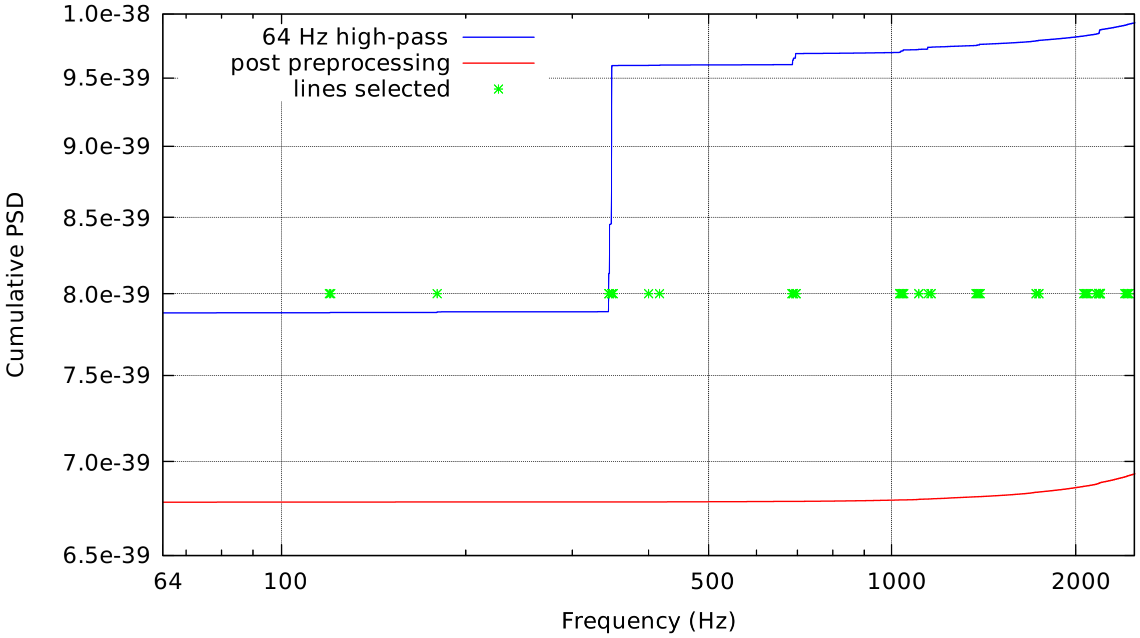


Figure 5.10: Cumulative PSD after 64 Hz high-pass filter (blue) and following iWave removal of selected lines (red).

The large step corresponding to a cluster of lines around 347 Hz has been removed leaving a much flatter noise spectrum.

5.6.4 LPEF Whitening

The LPEF alone is capable of removing lines and whitening the PSD; however, applied on its own this may require a longer filter length and therefore incur a higher computational cost. The strategy adopted was to use iWave to remove many of the most powerful lines in the PSD, thereby reducing the burden on the LPEF to do mainly whitening and adaptive filtering. The LPEF filter used had length 81920 samples (5 s) and a filter training time of 50 s. Re-training of the filters was performed approximately every 2000 s at times when the measured injection signal was zero. Fig. 5.11 shows the PSD following LPEF filtering of the data post line removal. The PSD is now confined to within approximately one decade in power, showing that the data is now considerably whiter. The LPEF has also removed the most prominent effects of aggressive line subtraction caused by our too simplistic implementation of iWave. The LPEF filter also raises the power at frequencies below 64 Hz which we counteract in the final preprocessing step.

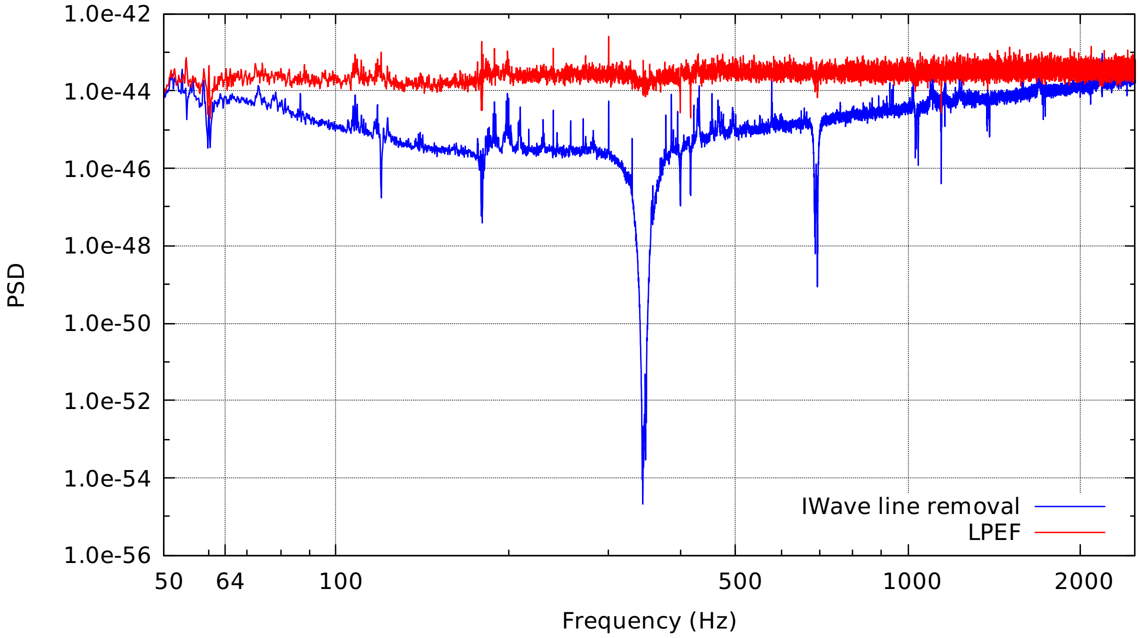


Figure 5.11: PSD following LPEF filter (red). The PSD is now much flatter and has removed most of the effect of excessive line subtraction.

5.6.5 Final Preprocessing

Preprocessing is concluded by first applying a second 64 Hz high-pass filter to suppress the elevation in power below 64 Hz due to LPEF filtering, followed by low-pass filtering at 2500 Hz with a 12th order Butterworth filter. The resulting PSD is shown in Fig. 5.12 (green)[1] where we can see that the spectral content of the data has now been substantially whitened within the pipeline search frequency range and attenuated at either end.

[1] Shown slightly elevated so that the 2nd high-pass step can be seen.

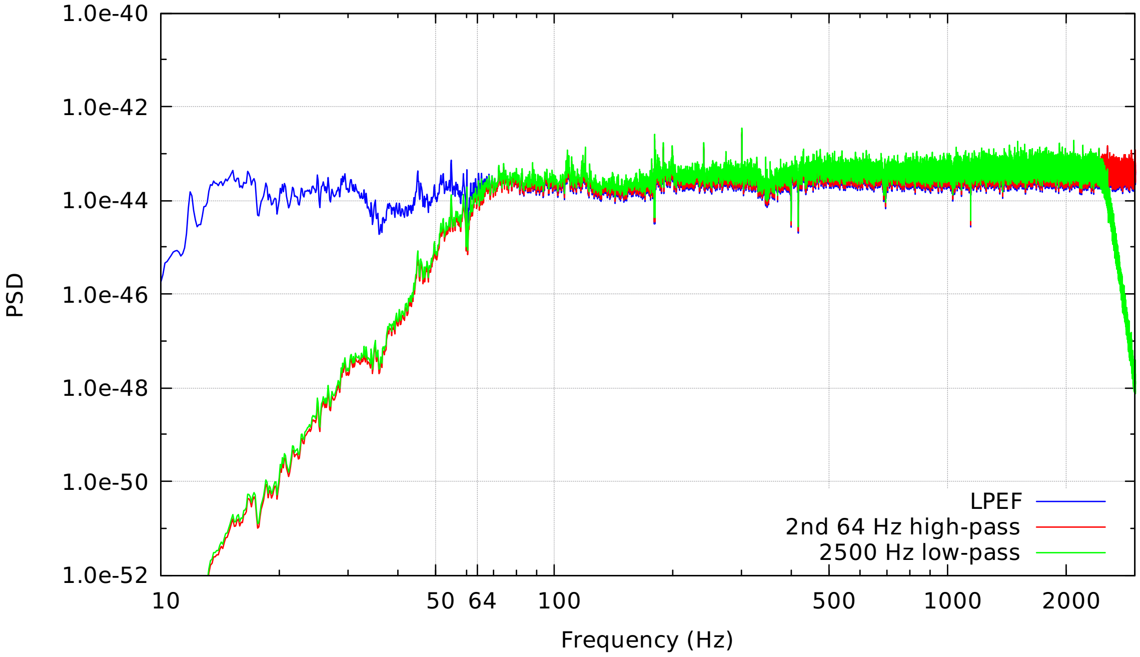


Figure 5.12: 2nd high-pass at 64Hz (red) and low-pass at 2500Hz (green).

In summary, the data preprocessing steps are as follows:

- High-pass filter at 64 Hz
- iWave line removal
- Apply LPEF
- Second high-pass at 64 Hz
- Low-pass filter at 2500 Hz

5.7 Post Trigger Processing

In this section we describe the processing steps performed immediately following an event trigger in the order of pipeline execution.

5.7.1 Trigger Capture

On passing a set RTCC event trigger threshold our objective is to capture the most significant causally related data prior to and after the trigger from both detector data streams for use in post trigger analysis. Relative to the trigger threshold, data prior to the trigger is likely to be less significant and we simply choose to continuously retain a set number of samples from each detector which can be prepended to the data captured following the trigger event. It is highly unlikely that the threshold will be equal to the RTCC output maximum following the trigger; therefore the most significant trigger data almost always occur post trigger and so we take more care in finding a method of defining the trigger end.

The RTCC output on processing SG triggers is not a simple pulse as we show in Fig. 5.13, which gives the C_n^0 output for two SG235Q8d9 signals injected with zero lag. In general we observe approximately twice the number of local maxima as prominent oscillations of the SG, each separated by $1/(2f_c)$ s and distributed over a pulse envelope. Therefore, simply defining the end of the trigger period as the condition when RTCC output again falls below threshold would almost certainly omit important data.

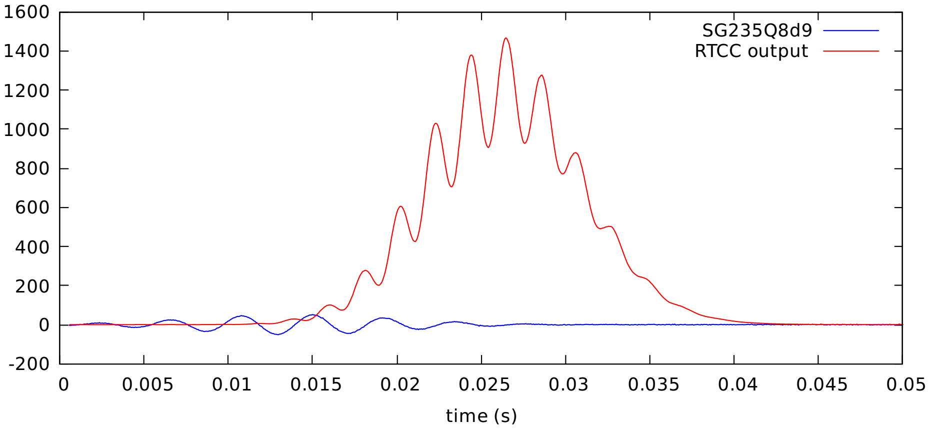


Figure 5.13: RTCC output for a SG235Q8d9 (shown for comparison) trigger.

We considered measuring the convergence of the summed signal power P_n at sample n following the trigger by taking successive power ratios P_n/P_{n-1} for each detector time series. These should converge to a value close to unity on passing of the trigger signal. We therefore set a convergence threshold $T_p \lesssim 1$. Since the power from the injections arrive as a series of pulses, premature termination is avoided by establishing the stability of convergence by setting a threshold on the consecutive number of times T_N that T_p must be satisfied before termination. Since noise may disrupt convergence, especially for low SNR injections, termination is ensured by setting a maximum post trigger recording length of 1 s.

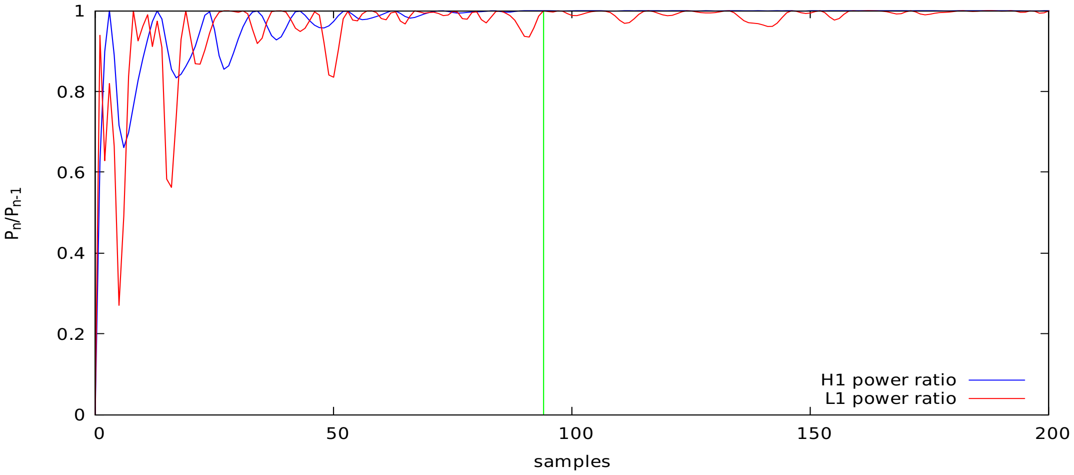


Figure 5.14: Example of SG849Q8d9 trigger capture using sample power ratio. The green vertical line at $n = 94$ marks the first instance where the power ratios for each detector satisfy the T_p threshold.

Fig. 5.14 shows an example portion of the power ratio evolution for a SG849Q8d9 trigger (shown in Fig. 5.16) where the convergence criterion $P_n/P_{n-1} > T_p = 0.999$ is first reached at 94 samples. The stability criterion $T_N = 540$ ultimately leads to a total post trigger sample length of $8084 = 0.49$ s. Comparison with $N_S = 172$ (Table 5.3) shows that in some cases the trigger sample length will far exceed the SG injection waveform length.

This method was found to be good at defining the trigger end for medium to high SNR injections, but convergence was often slow for low SNR injections. In some cases convergence was prolonged by the presence of long LPEF filter transients which were observed to remain for times longer than the trigger signal duration. To illustrate the above points, in Fig. 5.15 we show 8 consecutive captured trigger time series, each separated by the vertical lines reaching the top edge of the plot. Triggers (left to right) 1, 2, 4, 6, 7, and 8 are all from SG injections. Trigger 8, resulting from a comparatively low SNR SG, takes an excessive length of time to converge in proportion to the waveform duration. We can mitigate the effect of poor convergence by the signal enhancement methods described in Section 5.7.4.

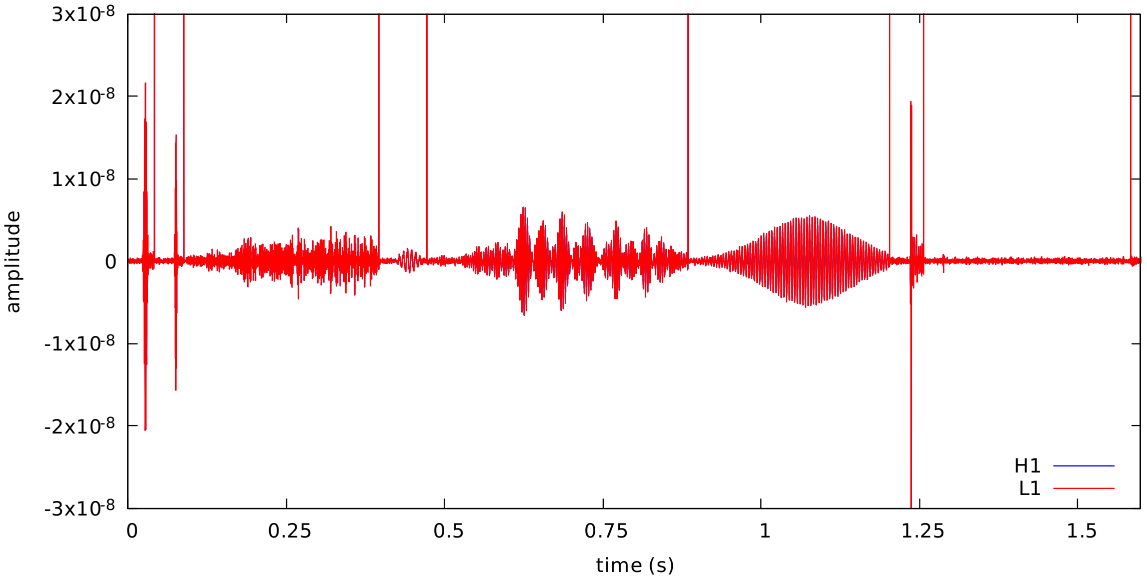


Figure 5.15: Series of 8 triggers captured by power ratio convergence. Convergence is good for medium/high SNR triggers but poor for low SNR triggers.

5.7.2 Lag Alignment of Trigger Data

During trigger capture, the lag between the SG injections in the detector data is determined by finding the sample lag $l_{\max}$ at which the RTCC output achieves its maximum over the trigger duration. The accuracy of lag determination depends on the contrast in amplitude in the neighbourhood of the the SG waveform maximum and SNR. When multiple amplitudes are similar, as occurs with SG waveforms with large Q factor, noise is more likely to cause RTCC maxima to occur at peaks neighbouring the waveform maximum. This introduces lag errors in multiples of $1/(2F_c)$. All subsequent analysis now uses the determined lag to align the waveforms in the captured detector trigger data. Fig. 5.16 shows the lag alignment of SG849Q8d9 waveforms.

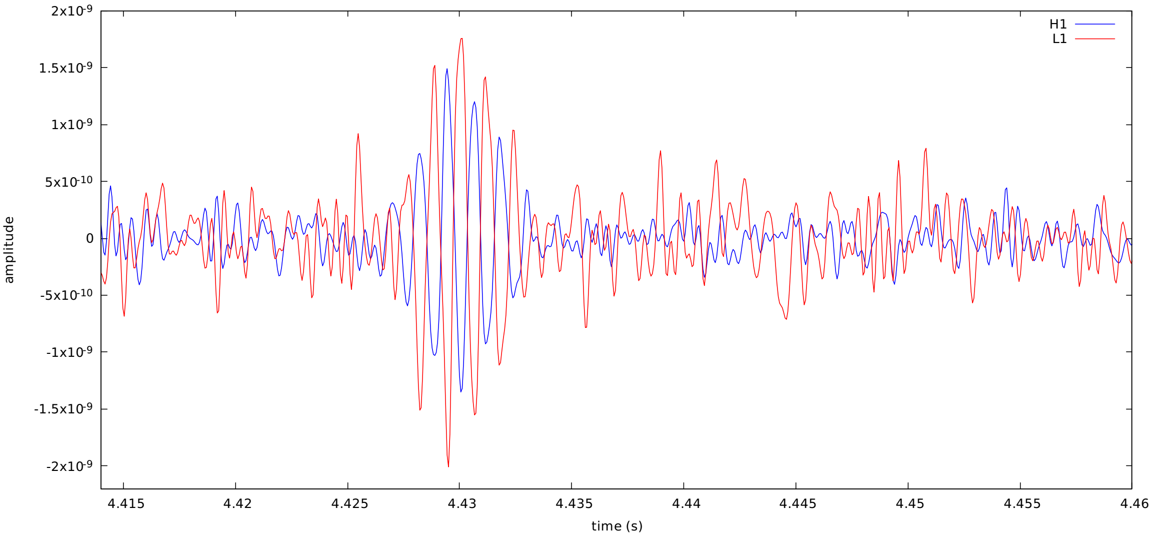


Figure 5.16: H1 and L1 SG849Q8d9 trigger signal aligned by detected lag.

5.7.3 Normalised Standard Cross-correlation

Although the RTCC algorithm has indicated a correlation between the two input time series, our primary discriminant will be the standard cross-correlation of the captured data at the determined lag $l_{\max}$. It is convenient to scale the output such that its magnitude lies between 0 and 1 by normalising the cross-correlation output by the product of the measured standard deviations of both time series. For the cross-correlation of N samples we therefore use

$$(X \star Y)^{l_{\max}} = \frac{\sum_{i=0}^{N-1} x_i y_{i-l_{\max}}}{\sigma_x \sigma_y}. \tag{5.3}$$

5.7.4 Probabilistic Filtering and Spike Removal

In the next section we describe a method of estimating the dominant frequency in the captured trigger data. The results of applying this method directly to data containing SG injections were promising but not deemed sufficiently reliable, especially for low SNR injections. We often observed instances where the method would have given a close correspondence to the SG injection frequency, but noise led to frequency selection outside the expected bandwidth. We were therefore motivated to find a method which reduces noise whilst retaining most of the injected waveform. The first method of signal enhancement we considered was the singular value decomposition (SVD) [73]. We developed trial code and the result of applying our SVD filter to trigger data containing a low SNR SG is shown in Fig. 5.17. The filtered signal closely resembles the injected SG waveform and so represents an almost ideal output. Unfortunately, this method is computationally expensive as it requires the solution of large matrices.

Having seen how effectively the SVD filter eliminates noise, with practically null output either side of the wave form, we considered it probable that frequency estimation would benefit by effectively zeroing all data lying below approximately three standard deviations of the background noise.

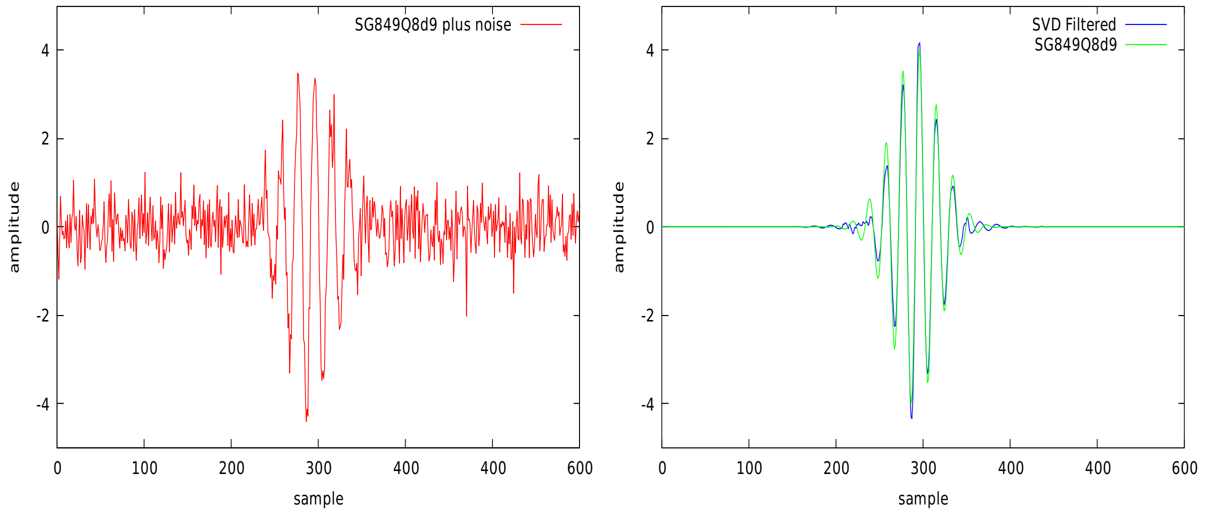


Figure 5.17: SVD recovery of SG849Q8d9 waveform. Left: SG plus noise. Right: SVD filter output (blue) and original injected SG849Q8d9 waveform (green).

This would remove a significant fraction of the noise surrounding low SNR SG waveforms in the trigger data. For example, in the above figure (left) approximately two thirds of the noise surrounding the waveform would be removed. We achieved a primitive approximation to the SVD filter output by applying what we will refer to as a probabilistic Gaussian filter followed by a spike removal filter.

The probabilistic Gaussian filter weights each sample x_i in the time series by a measure of its probability $P_i(x_i)$ not to occur in a Gaussian distribution with variance equal to the noise background σ_n^2. The error function could have been used for this purpose, but we found that the proportional term $P_i(x_i) \propto 1 - \exp(-x_i^2/2\sigma_n^2)$ sufficed. In order to obtain weights for any given data set which do not suppress maximum amplitudes, we normalise by the same expression as the proportional factor but with the maximum observed excursion x_{max} as argument. The Gaussian filter function is then given by

$$P_i(x_i) = \frac{1 - e^{(-x_i^2/2\sigma_n^2)}}{1 - e^{(-x_{max}^2/2\sigma_n^2)}}. \tag{5.4}$$

We then set a threshold on the value of $P_i(x_i)$, below which we set its value to zero. The threshold value is chosen so as to retain most of the injection signal information whilst removing most of the noise content.

Clearly, we inevitably lose some of the waveform data at either extreme, and even the SVD filter noticeably attenuates , but since our primary interest in this case is to obtain a satisfactory estimate of the frequency we only require that a sufficient portion of the waveform remains. The RTCC pipeline uses a threshold of 0.8 and in Fig. 5.18 (left) we show an example of the selectivity of the weighting function for a SG849Q8d9 injection. After Gaussian filtering (Fig. 5.18 (right)) we are often left with isolated spikes which we remove by applying a simple conditional filter which determines the presence of spike features of between one and three samples duration.

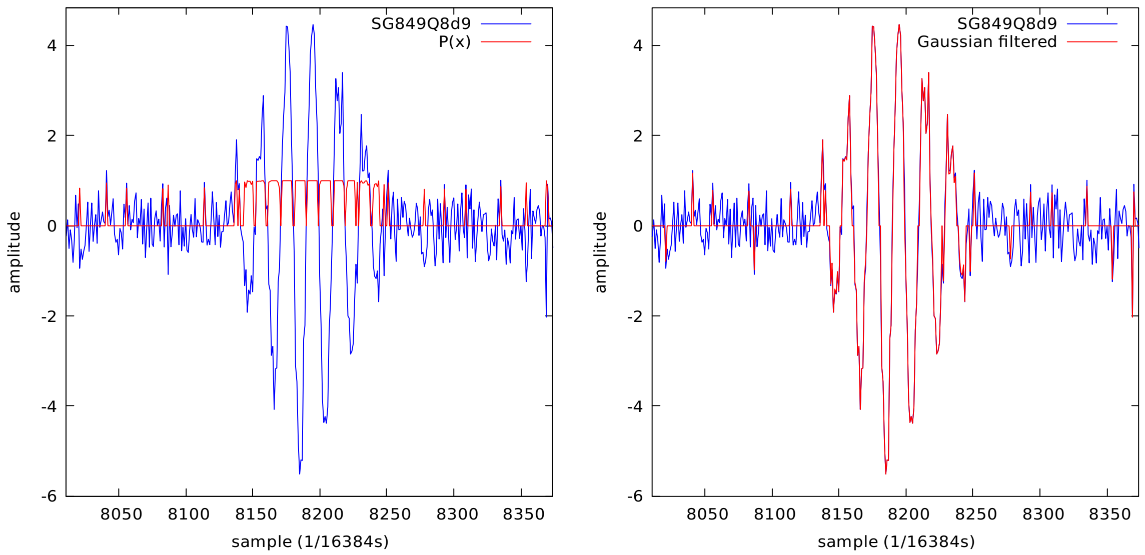


Figure 5.18: Gaussian probabilistic filter selectivity for SG849Q8d9 (left) and Gaussian filtered time series (right).

Fig. 5.19 shows the combined filter process applied to a SG849Q8d9 trigger, better defining where the dominant power in the injection is present and leaving almost zero signal elsewhere. Apart from the potential improvements in frequency estimation, this also has the benefit of making the computation of quantities whose output only relies upon non-zero signal values much faster.

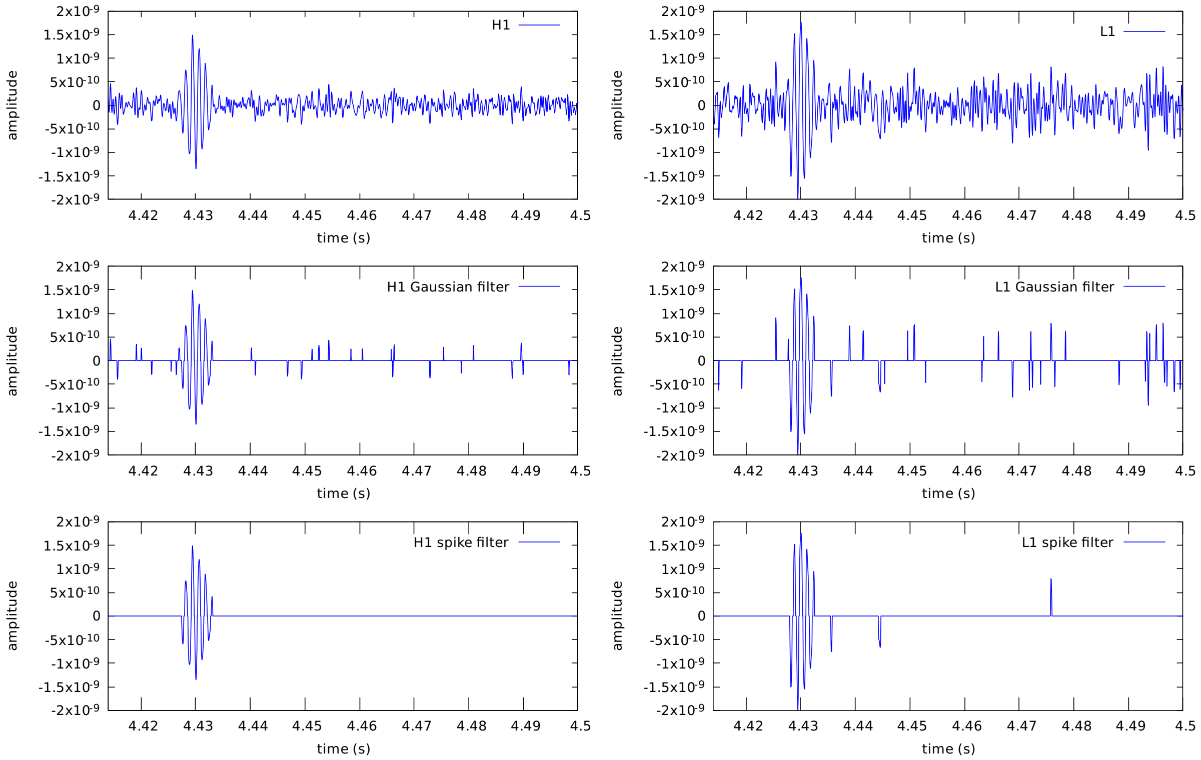


Figure 5.19: Application of Gaussian filter and spike filter to H1 and L1 SG849Q8d9 trigger signals.

Of course, we could have obtained similar output to the Gaussian-spike filter directly by setting to zero all data amplitudes below $\sim 3\sigma_n$, but this would in general still leave spikes and indiscriminately null low amplitude waveform data. The Gaussian-spike filter approach has more scope, we believe, for further development and so we test its use in this pipeline.

5.7.5 Frequency Estimation

The method devised to estimate the dominant frequency present in each of the captured trigger data sets is based on the Fourier decomposition of a signal into a sum of sine and cosine harmonics. To obtain an indication of the power at a particular frequency $A(f)$, we sum the squared correlation of the captured time series with the discretely sampled sine and cosine of corresponding frequency. For a time series x_i of N samples the power amplitude function is given by

$$A(f) = \left(\sum_{i=0}^{N-1} x_i \sin(2\pi f t_i) \right)^2 + \left(\sum_{i=0}^{N-1} x_i \cos(2\pi f t_i) \right)^2 , \tag{5.5}$$

where discrete time $t_i = i/f_s$. We obtain a ‘spectrum’ by calculating $A(f)$ in discrete 1 Hz steps over the pipeline frequency search range. The dominant frequency estimate is the value of f at which $A(f)$ is maximum.

For high SNR SG injections, this method produces frequency estimates which are accurate to within a few Hz of the SG central frequency. However, to be of use in the pipeline, we require at least adequate estimates for low SNR injections. In Fig. 5.20 we show the spectra obtained from a low SNR SG849Q8d9 trigger before (top) and after (bottom) Gaussian-spike filtering (see Section 5.7.4).

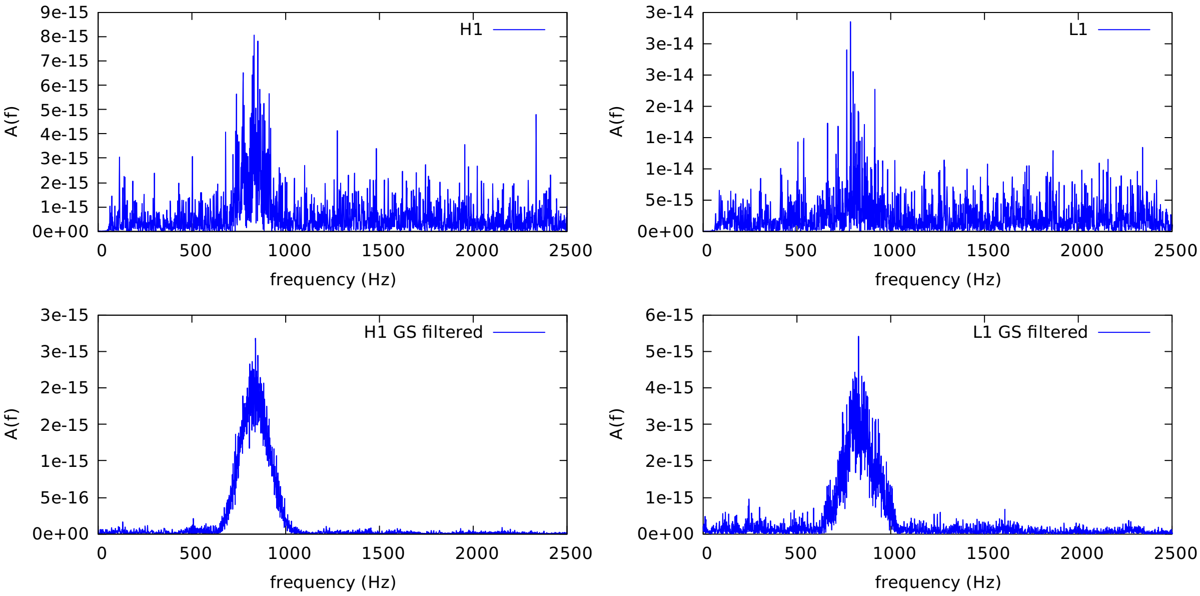


Figure 5.20: Top row: Frequency estimation spectra $A(f)$ for SG849Q8d9 injection trigger data prior to Gaussian-spike filtering with maxima at H1 831 Hz L1 785 Hz. Bottom row: Spectra following Gaussian-spike (GS) filtering giving improved estimates H1 838 Hz and L1 828 Hz.

In the above figure, prior to Gaussian-spike filtering the estimated trigger frequencies for H1 and L1 are 831 Hz and 785 Hz respectively. These estimates would be considered adequate as they are relatively close in value and either lie within or close to the SG bandwidth. Nevertheless, in general we find that for low SNR injections the spectra show many strong spikes across the frequency search range which often force frequency selection far outside the SG bandwidth. We can substantially improve the probability that both H1 and L1 frequency estimates more closely correspond by applying the Gaussian-spike filter described in Section 5.7.4.

The significant noise reduction following Gaussian-spike filtering produces spectra with more prominent maxima and approximate Gaussian[1] spectral profile about the maximum. From this we might consider attempting estimation of the Q value as an additional parameterisation. Taking the definition of Q as the ratio of central frequency f_c to the spectrum standard deviation σ_f, we may estimate Q by relating the full width at half maximum (FWHM) to σ_f by FWHM $= 2\sigma_f\sqrt{2\ln 2}$. This gives

$$Q = \frac{f_c}{\sigma_f} = 2\sqrt{2\ln 2}\frac{f_c}{\mathrm{FWHM}}. \tag{5.6}$$

For example, in the above figure we measure $\mathrm{FWHM}_{\mathrm{H1}} = 196$ Hz and $\mathrm{FWHM}_{\mathrm{L1}} = 178$ Hz which respectively give estimates of $Q_{\mathrm{H1}} = 10.0$ and $Q_{\mathrm{L1}} = 10.9$. Unfortunately, after several Q determination trials we found the method given above too unreliable and reluctantly decided to omit this trigger parameterisation.

5.8 RTCC Thresholds

Each run uses one of five RTCC trigger thresholds. These were chosen such that the highest threshold gave 100 % detection efficiency at the highest injection scale and the lowest as the minimum which still gives an acceptable trigger rate. The trigger rate at the lowest threshold, now primarily due to noise, should not be too high (e.g. 1 Hz) as this will severely impede run progress and lead to excessively large trigger files. The RTCC thresholds are scaled exponentially and given by the relation,

$$\mathrm{threshold}_n = 10^{\frac{23+4(n-1)}{16}} \text{ for } n = 1...5. \tag{5.7}$$

Eq. 5.7 therefore gives thresholds for $n = 1$ to 5 of 27.38, 48.70, 86.60, 153.99 and 273.84 (rounded to two decimal places).

5.9 RTCC Pipeline Execution Timing

Although the RTCC pipeline was designed to process archived data rather than during detector data acquisition in real time, we will present the main pipeline operation times scaled to filter sample length for preprocessing and post-trigger process timing per sample. Additional scalings for some operations are given for generality. In particular, we give the iWave times scaled to the number of lines removed and RTCC algorithm times per ±1

[1] The frequency spectrum of a SG is a Gaussian function centred on f_c.

ms of lag. Processing triggers, whether from injections or noise, clearly takes more time, and therefore the times obtained are upper limits on the pipeline execution speed.

All timing tests were performed on a computer running the Scientific Linux 6.3 operating system with `x86_64` architecture. The hardware consisted of 4 `Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz` processors with 2 cores and 3.3 GiB RAM. The RTCC software (see Section 7.2) was compiled with `gcc/g++` and `c/c++ O2` optimisation.

Table. 5.5 provides average preprocessing timings per sample for high-pass (HP) filtering, iWave line removal (per line), LPEF (pre-trained) and low-pass (LP) filtering. The second high-pass HP2 is equivalent to HP1 and the LP filter takes approximately double the processing time of the HP filter.

	HP1	iWave	LPEF	HP2	LP
Time (μs)	0.026	0.006	0.335	0.025	0.050

Table 5.5: Average preprocessing times per sample for the first high-pass (HP1), iWave per line, LPEF, second high-pass (HP2) and low-pass (LP) filters.

The above table can be used to estimate the total preprocessing time per sample for the total of 201 lines removed during pipeline runs for each detector (H1 - 138, L1 - 63), to give 1.64 μs.

Table. 5.6 gives timing averages per sample for the RTCC algorithm (per ±1 ms lag), Gaussian-spike (GS) filtering, normalised cross-correlation (NCOR), and dominant frequency estimation (F_{est}) in the range 50 – 3000 Hz.

	RTCC (/± 1 ms)	GS	NCOR	F_{est}
Time (μs)	0.14	0.12	0.22	35.35

Table 5.6: Average timing per sample for the RTCC algorithm, Gaussian-spike filter (GS), normalised cross-correlation (NCOR) and frequency estimation (F_{est}).

Our pipeline runs have RTCC lag configured for ±180 sample lags at 16384 Hz (i.e. ±11 ms) and so the total trigger processing time is 37.23 μs, of which the largest portion is consumed by trigger frequency estimation.

The total preprocessing and trigger processing times give an estimated time of ~ 39 μs per sample. Real time data processing would require processing faster than the sampling rate, i.e. within $1/16384 \sim 61$ μs. Therefore, the main pipeline operations are capable of very low latency data processing. Unfortunately, trigger capture power convergence times introduce latencies which were observed to range on average between 1 – 10 waveform lengths, depending upon SNR and the waveform quality factor. In real time processing we should then expect trigger delays of ~ 5 ms for a 1 ms duration waveform.

5.10 RTCC Pipeline Structure and Large Scale Runs

The RTCC pipeline was developed with the intention that it could process GWF formatted data from any detector and meet different preprocessing requirements. In order to facilitate this flexibility we have enabled configuration of the pipeline analysis entirely by the input of files. All preprocessing filters are either built or initialised from input configuration files which can request a wide range of IIR Filters, including the LPEF.

The outer layers of the RTCC pipeline software parse the separate input files for frame paths, science segments, and preprocessing configuration. Inner layers manage coordinated data input for the detector H1/L1 GWFs and injection MDC GWFs (generally not equal in length), perform the software injections and preprocess the resulting data. At the core of the pipeline lies the RTCC algorithm, trigger data analysis and trigger file writing routines.

Before a run commences, the availability of each frame is checked since frame access on the LDG may have changed since production of the frame path files. In the event that a required frame becomes unavailable during a run, the DMT data access class `Dacc` will issue an error message regarding GWF access and the run terminates. The pipeline was written so that if a run stopped due to the unavailability of frame data or any other cause, then the run may be restarted and automatically process data commencing from the last trigger time recorded and proceed by appending the trigger files.

By way of summary, we show in Fig. 5.21 a schematic representation of the RTCC pipeline which outlines the required input to the pipeline and the main processes performed during operation. Terminal input refers to manual keyboard entry of the pipeline arguments (e.g. run start/end times).

Large scale runs utilise the Directed Acyclic Graph (DAG) job management system on the LDG. This permits the allocation of computing resources for multiple computing jobs. These are submitted to the DAG via DAG job submission files. These files detail the job input parameters. For example, one RTCC run requires one of 31 separate injection scales and one of five trigger threshold values. Therefore a typical multiple run will comprise 155 jobs. Processing speed will depend upon available computing resources and load usage.

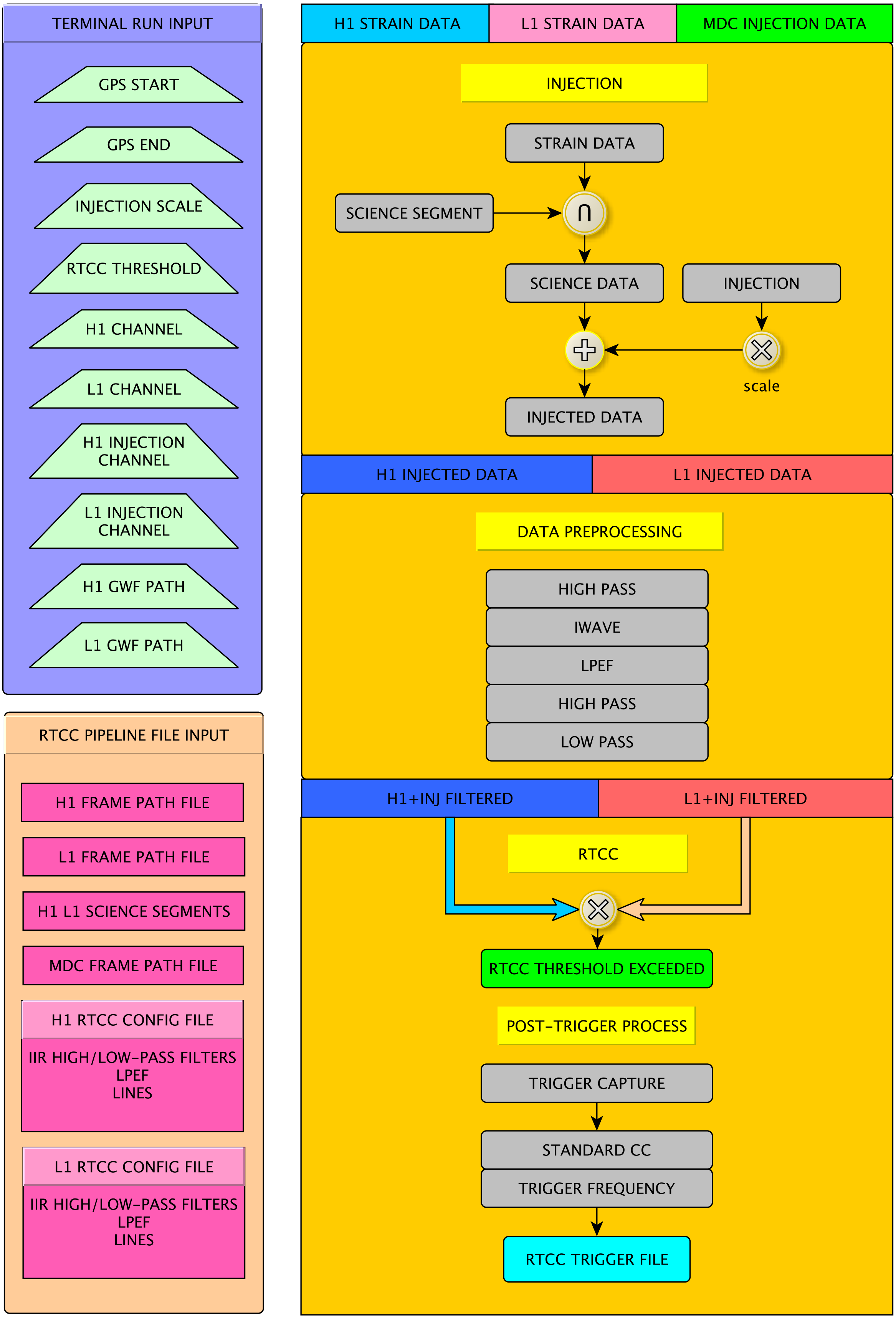


Figure 5.21: RTCC pipeline schematic. This shows both the required input (terminal and files) and main processes during operation.

Chapter 6

RTCC Pipeline MDC Results

We now present the results of our RTCC pipeline runs. Sine-Gaussian (SG) injection triggers are selected based on two coincidence criteria: the post-trigger normalised standard cross-correlation (CC), and a measure of the similarity of the dominant frequencies in both of the captured detector data. In addition, we discard all triggers where the determined signal-to-noise ratio (SNR) is plus or minus infinity or negative (i.e. unphysical). The SNR is determined periodically from the RMS of the data when injections are not present and we reject triggers when this is above the post-trigger RMS. We will consider the measured distributions of the trigger discriminants in order to decide how to apply thresholds or cuts which give close to maximal rejection of time slide (TS) triggers whilst minimising rejection of SG injection triggers.

As in Section 4.3, the probability distribution functions (PDFs) shown are obtained from the frequency histogram by normalisation relative to area.

6.1 Normalised Cross-Correlation Distribution

We show in Fig. 6.1 the distribution of normalised CC for the SG and TS triggers from run 954115215-961977615 for the five RTCC output thresholds used.

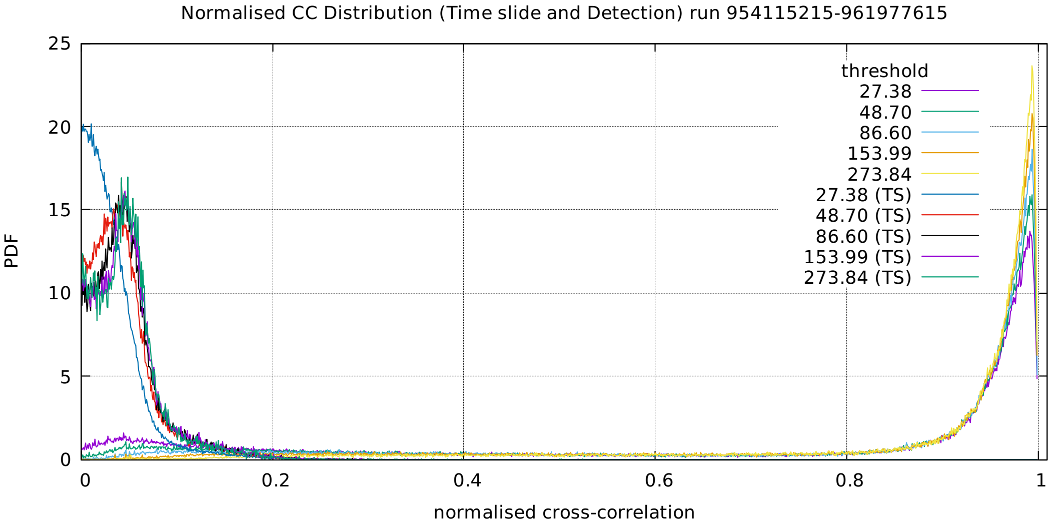


Figure 6.1: Distribution of normalised CC for SG triggers and time slide (TS) triggers.

The distributions show marked separation of maxima: most of the SG triggers have normalised CC greater than 0.8 whilst most of the TS triggers fall below 0.2. The tail of the TS distribution is better viewed on the logarithmic plot of Fig. 6.1 shown in Fig. 6.2.

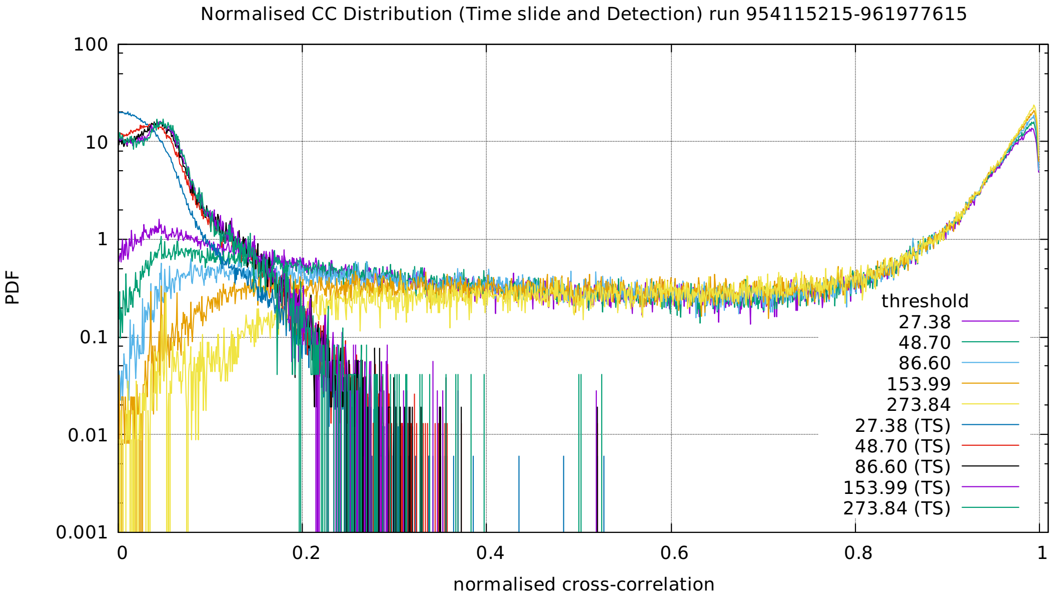


Figure 6.2: Logarithmic normalised distribution of normalised CC for SG triggers and time slide (TS) triggers.

It is now clear that some TS triggers have normalised CC values far above 0.2. We can see how our choice of rejecting TS triggers which fall below a particular normalised CC threshold C_{thr} will affect our SG trigger acceptance in the cumulative distribution of Fig. 6.3.

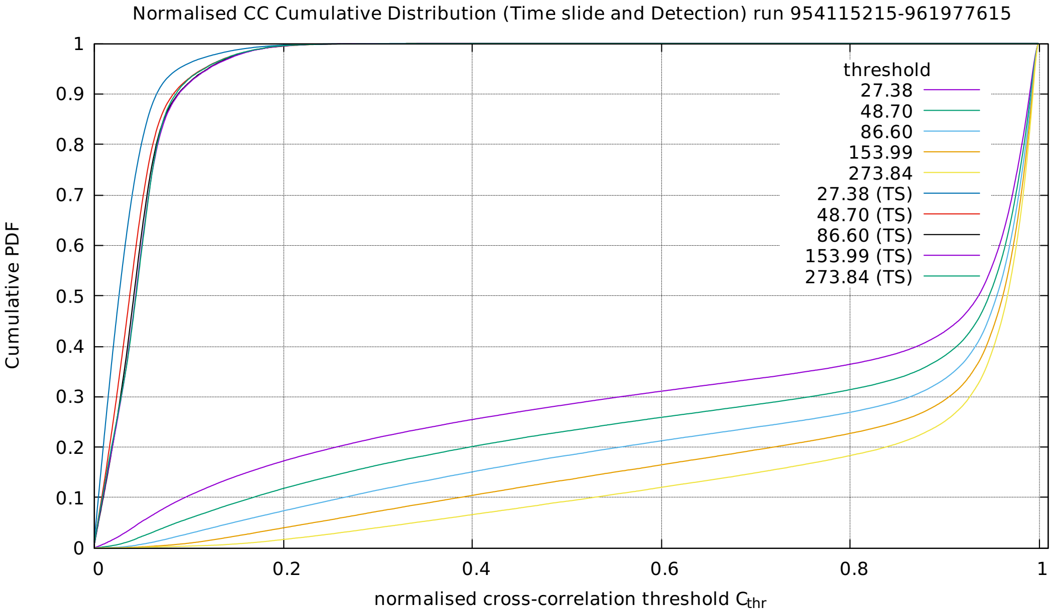


Figure 6.3: Cumulative distribution of TS and SG triggers for threshold selection C_{thr}.

If we set $C_{\mathrm{thr}} = 0.6$ this would eliminate all TS triggers from this run but also reject from between around $10 - 30$ % of SG triggers, depending on the detection threshold. The maximum distance between TS and SG trigger curves occurs in range of $C_{thr} = 0.98 - 2.2$. We chose a value of $C_{thr} = 0.25$ which rejects 99.8% of TS triggers and retains 97, 94, 90, 86 and 80% of SG triggers from highest to lowest trigger threshold.

6.2 Trigger Frequency Discriminant F_{rat}

We assume that the probability that the difference between the determined dominant frequencies from both detector trigger time series f_{det1} and f_{det2} will be large is much greater for noise triggers than for triggers of SG injection origin. Therefore, when the trigger frequencies are plotted as points on a graph with axes spanning the search range $0 - F$ for each detector, we expect SG triggers to lie close to the line $f_{\mathrm{det2}} = f_{\mathrm{det1}}$ and that TS trigger points will be approximately uniformly randomly distributed within the search frequency space F^2.

As a measure of the proximity of the trigger frequencies we define F_{rat} as the sum of the ratio of trigger frequencies and the inverse ratio. Thus,

$$F_{\mathrm{rat}} = \frac{f_{\mathrm{det1}}}{f_{\mathrm{det2}}} + \frac{f_{\mathrm{det2}}}{f_{\mathrm{det1}}}, \tag{6.1}$$

which is symmetric with respect to interchange of detector frequencies and has a value always greater than or equal to 2. For frequencies very similar in value $F_{\mathrm{rat}} \gtrapprox 2$, and for very different frequencies F_{rat} gives the approximate ratio of the largest frequency to the smallest. If we choose to accept triggers with $F_{\mathrm{rat}} \leq R$ where $R > 2$, this defines a trigger acceptance zone A_{ac} within the frequency space which is symmetric about the $f_{\mathrm{det2}} = f_{\mathrm{det1}}$ line. The frequency space is shown in Fig. 6.4 with A_{ac} in grey and the remaining area A_{rej}, in which triggers are rejected, in white.

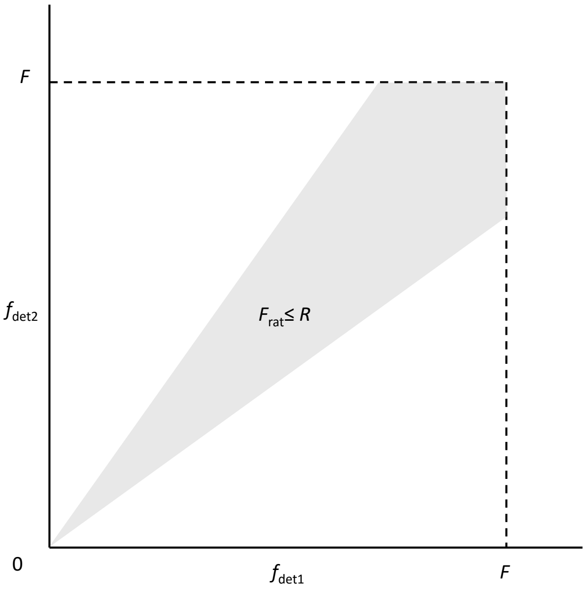


Figure 6.4: Frequency space for plotting (f_{det1},f_{det2}) where triggers are accepted if $F_{\mathrm{rat}} \leq R$ (grey zone) and otherwise rejected (white areas).

Given our naive assumption on the distribution of TS trigger frequencies, we expect the proportion of TS triggers occurring over a sufficiently long period of time within any defined area A of the total F^2 space to be $P = A/F^2$. A little algebra yields the proportion of triggers rejected as a function of the threshold R as

$$P_{\mathrm{rej}}(R) = \frac{(R - \sqrt{R^2 - 4})}{2}. \tag{6.2}$$

Clearly, we have accepted the proportion $P_{\mathrm{ac}}(R) = 1 - P_{\mathrm{rej}}(R)$ of TS triggers and these must contribute to the FAR unless rejected by another criterion. Since the bandwidth of SG waveforms is proportional to frequency, our intuition in devising this form of discriminant was that the tolerance for acceptance based on difference in estimated frequency should also increase with frequency.

In order to find the threshold value of R which maximises TS rejection whist minimising SG trigger rejection, we investigate the distributions of F_{rat} for both trigger types. Fig. 6.5 shows a logarithmic plot of the normalised distribution of F_{rat} where, as expected, the TS triggers show a much higher probability that the frequencies are significantly different, only approaching the SG probability when $F_{\mathrm{rat}} \gtrsim 30$.

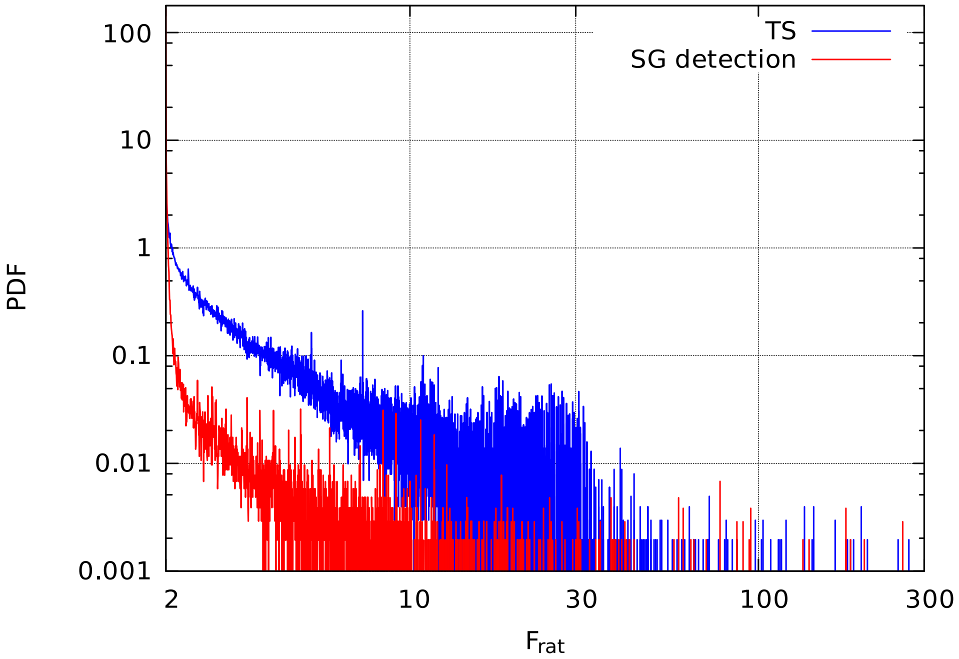


Figure 6.5: Normalised distribution (PDF) of F_{rat} for TS and SG triggers.

The 1-cumulative distribution shown in Fig. 6.6 shows the proportion of TS and SG triggers rejected upon setting the threshold. From $F_{\mathrm{rat}} = 2$ onward, the distance between the TS and SG trigger rejection curves increases and then decreases. Therefore, a maximum distance between the curves occurs at some point in-between, which we take to be the optimal R threshold. This maximum was found to occur at $F_{\mathrm{rat}} = 2.05$, giving a TS rejection of 83%, close to the 80% value expected from Eq. 6.2, and a SG trigger rejection of 7%. This value was subsequently used for our post-trigger cuts.

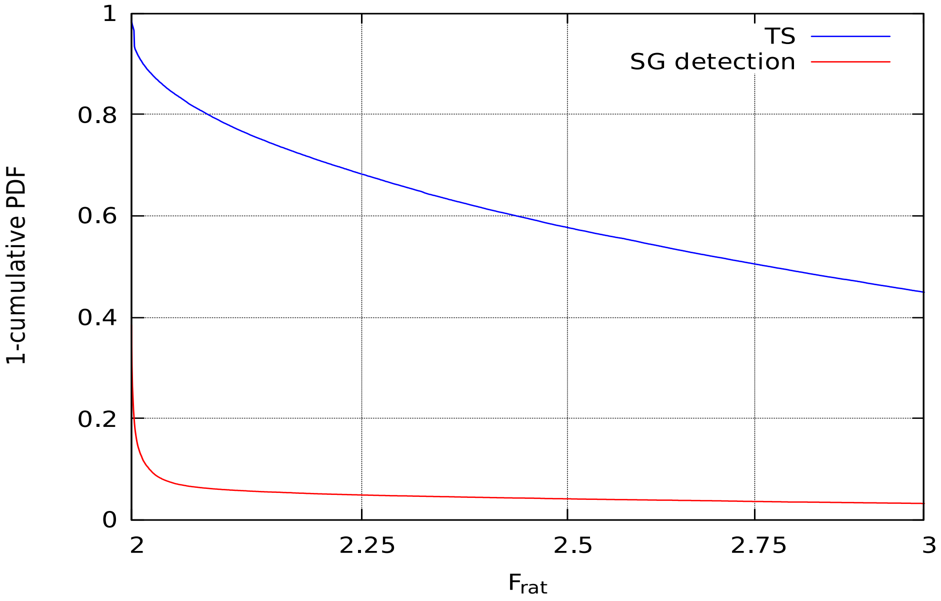


Figure 6.6: 1-cumulative $F_{\rm rat}$ distribution.

6.3 Detection Sensitivity and False Alarm Rate

Fig. 6.7 shows the sensitivity curves obtained prior to threshold cuts from detection of all SG injection types at each RTCC trigger threshold. The sensitivity curves are separated at 50% detection efficiency by $h_{\rm rss}^{50\%}$ values of 2.8, 4.0, 5.5, 7.6 and 10.2×10^{-21} Hz$^{-1/2}$.

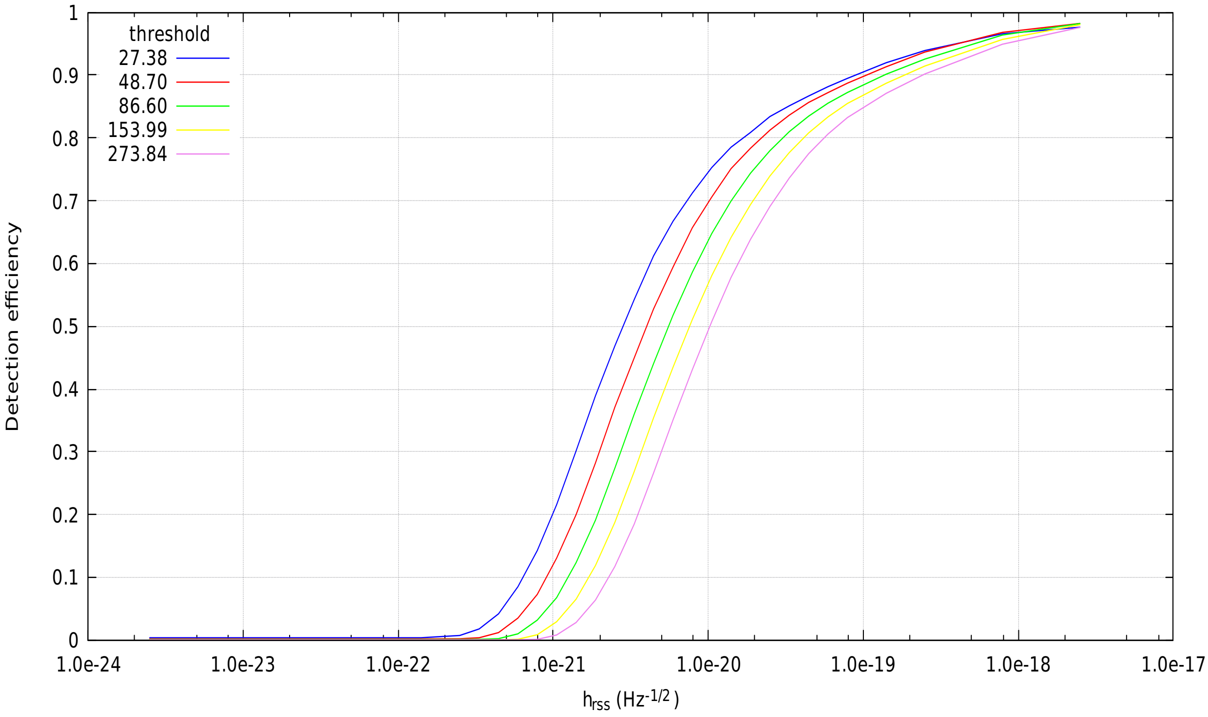


Figure 6.7: SG detection sensitivity before applying cuts.

Fig. 6.8 shows the sensitivity curves obtained following the threshold cuts on normalised CC and F_{rat} from detection of all SG injection types at each trigger threshold. Comparing this with Fig. 6.7, we can see that the curves much closer together, with detection at the two lowest thresholds now barely distinguishable. At 50 % detection efficiency the sensitivity curves are separated by $h_{\mathrm{rss}}^{50\%}$ values of 6.9, 7.1, 7.5, 8.6 and 10.4×10^{-21} Hz$^{-1/2}$.

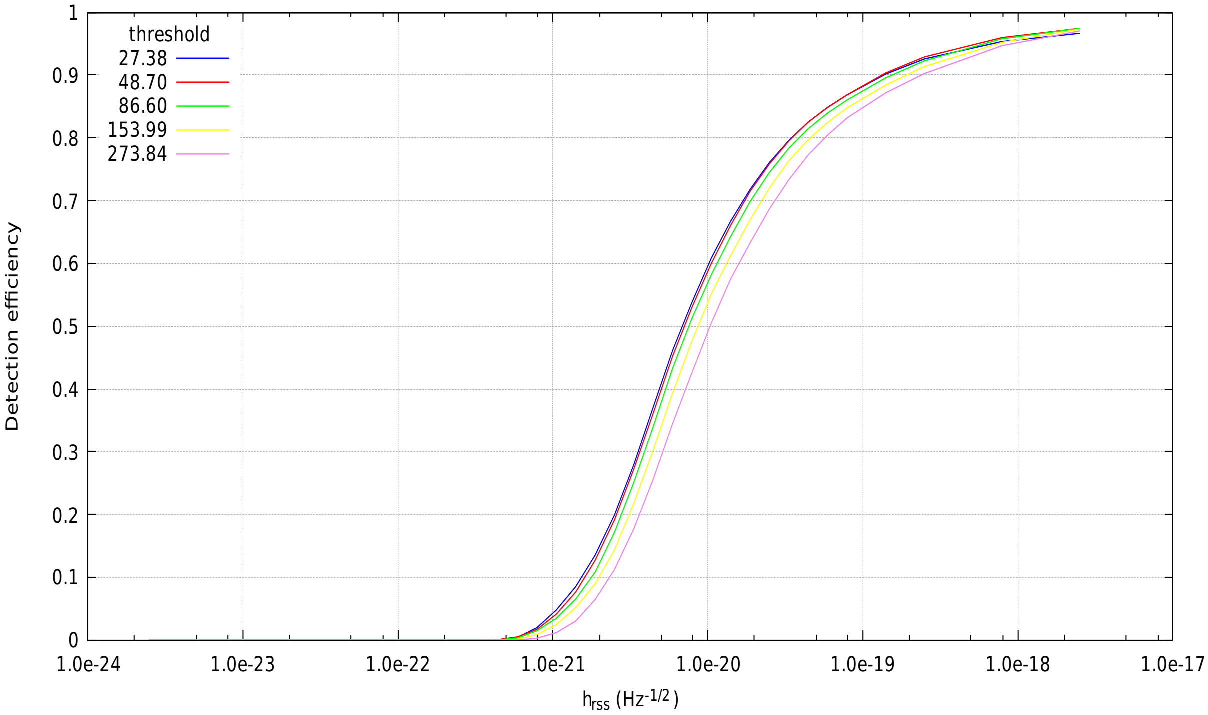


Figure 6.8: SG Detection sensitivity after cuts applied

Figs. 6.9 – 6.12 show the post cut sensitivity for all SG injections individually. All the sensitivity curves for SGs of frequency 235 Hz or less show a marked inflection at around 70 % detection efficiency with a reduction in the rate at which detection efficiency increases with injection h_{rss} and then the rate recovers. These results were not as expected, given the overall detection efficiency curves in the above figures. We expended a considerable amount of time and effort in an attempt to explain these results, but without resolution.

The remaining results for SGs of frequency 361 Hz, whilst not perfectly smooth, are much closer to the expected sigmoid shape. All results show that the sensitivity curves of $Q = 3$, 8.9, and 9 converge with increasing SG frequency whilst those with $Q = 100$ have a higher dependence on RTCC threshold and maintain distinct separation. This can be explained simply by the fact that $Q = 100$ SGs have much more waveform for RTCC correlation than the lower valued Q waveforms.

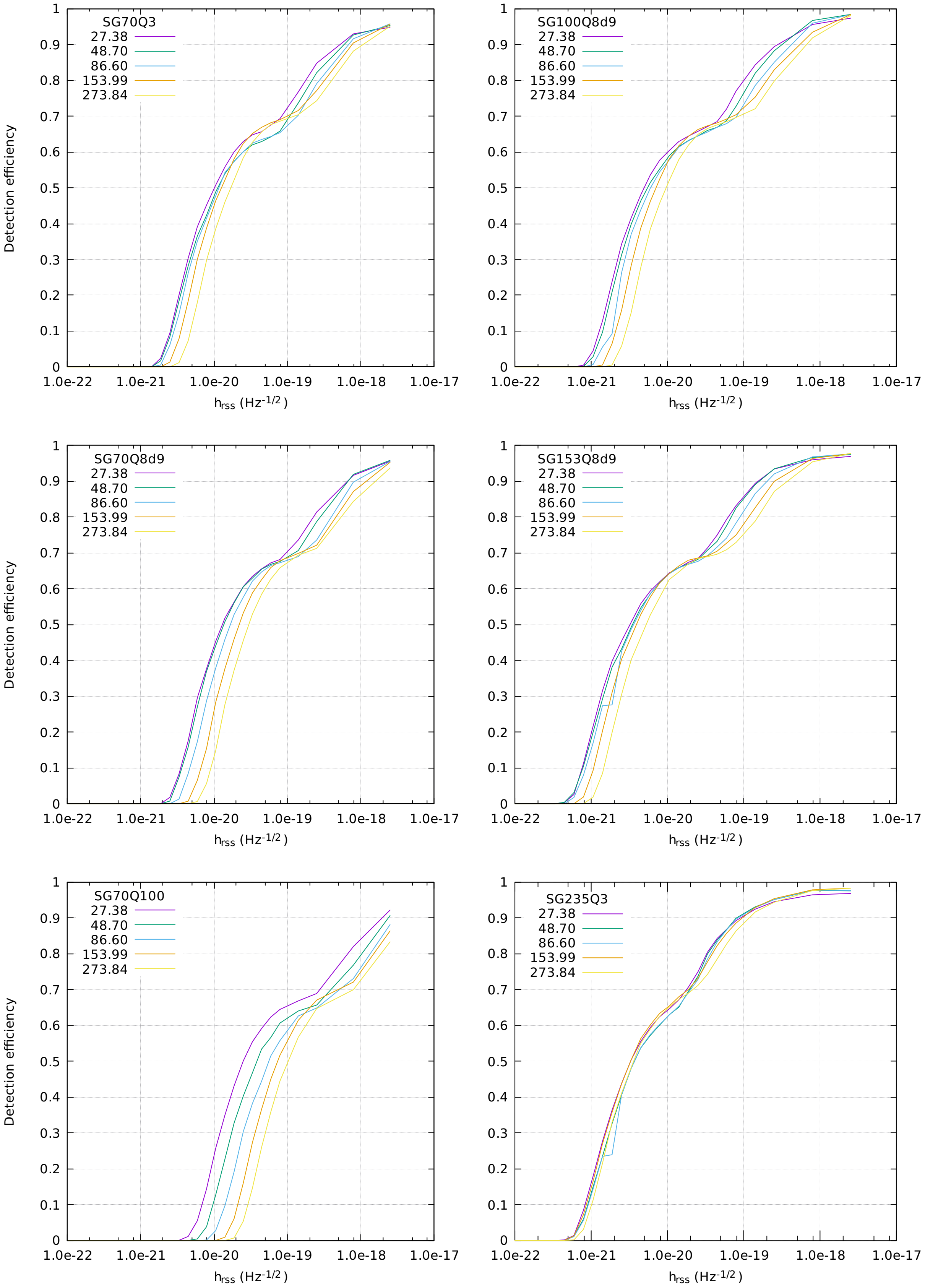


Figure 6.9: Sine-Gaussian sensitivity curves SG 70Q3/8d9/100 100Q8d9 153Q8d9 235Q3.

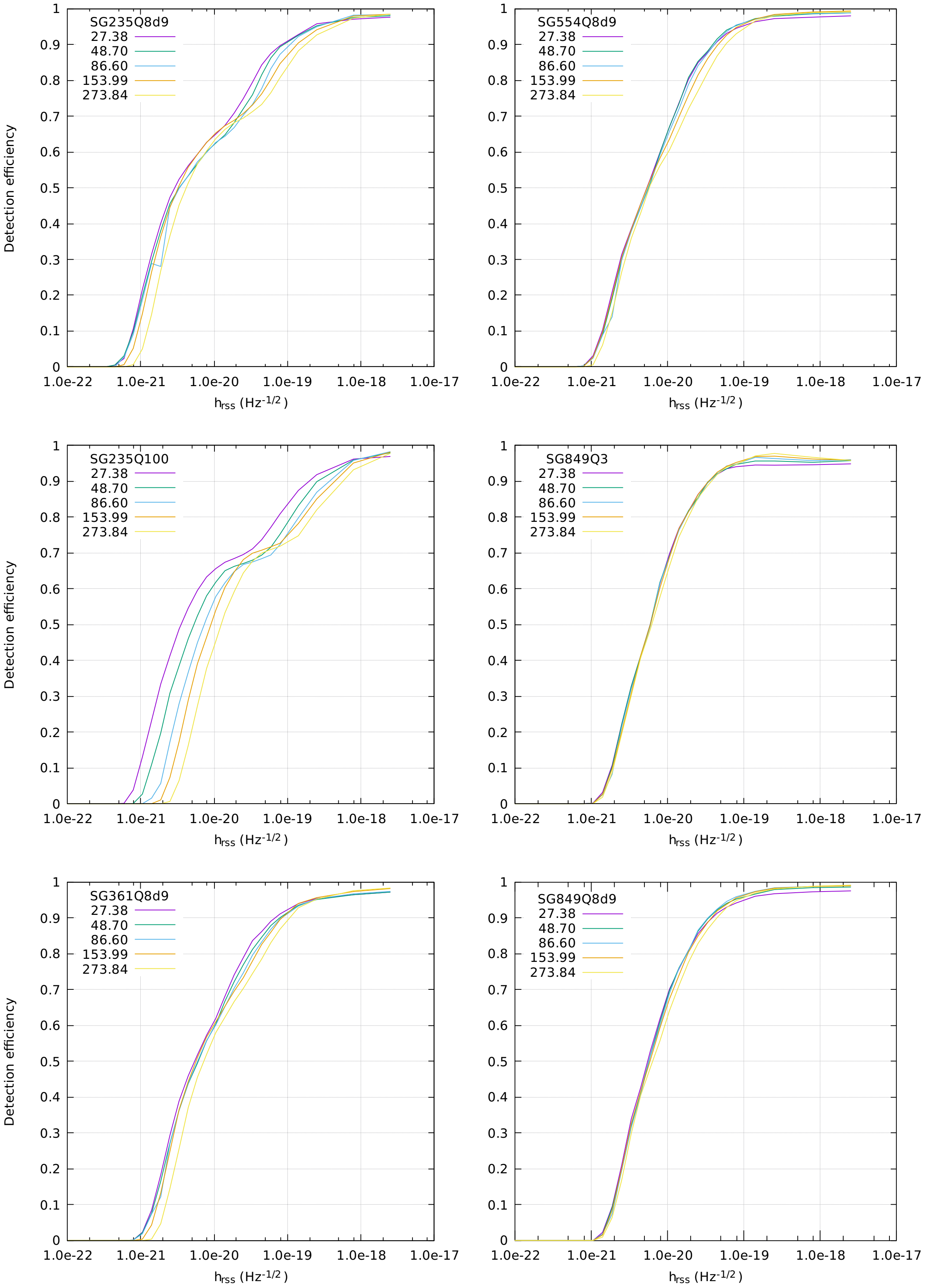


Figure 6.10: Sine-Gaussian sensitivity curves SG 235Q8d9/100 361Q8d9 554Q8d9 849Q3/8d9.

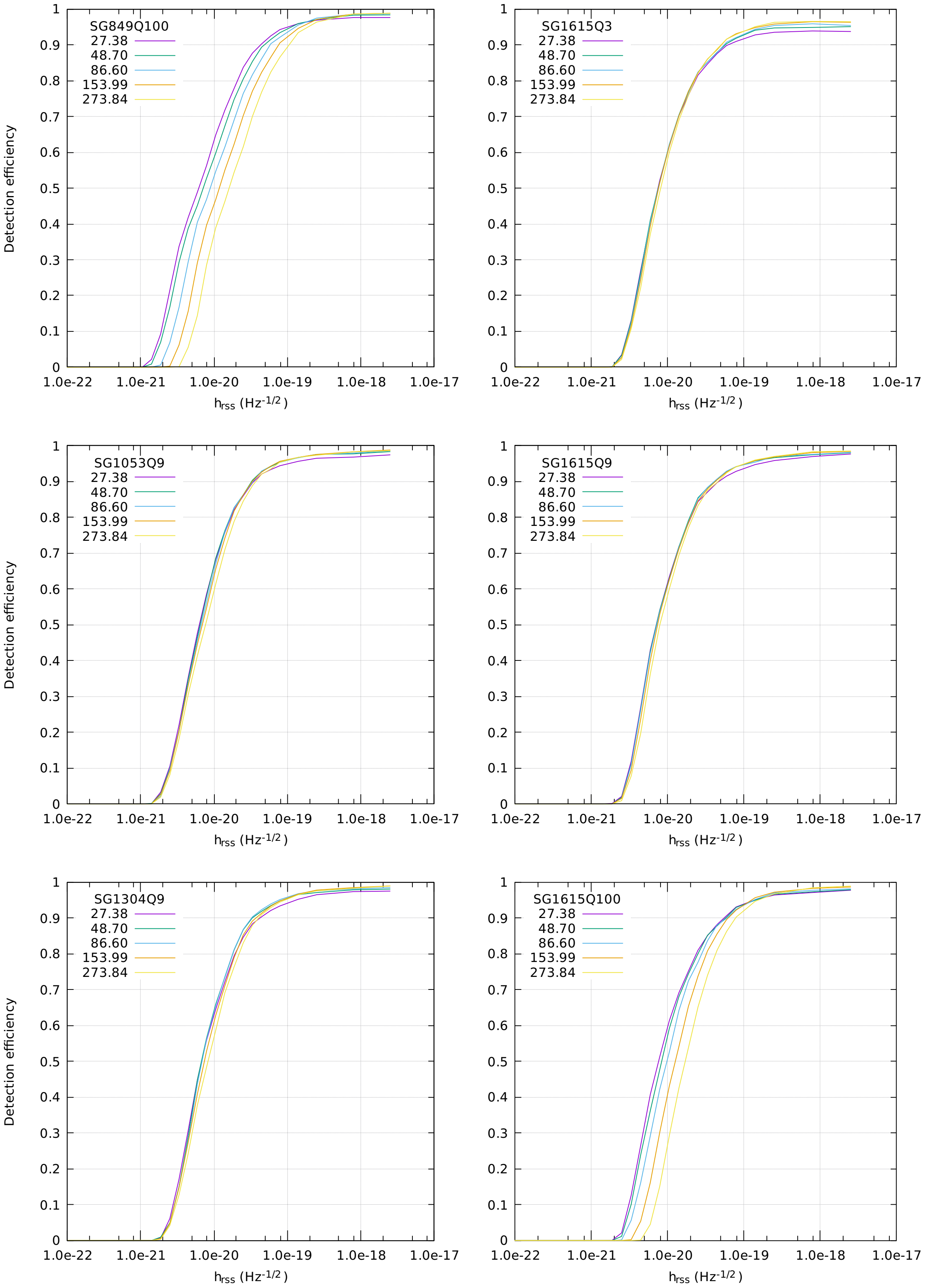


Figure 6.11: Sine-Gaussian sensitivity curves SG 849Q100 1053Q9 1304Q9 1615Q3/9/100.

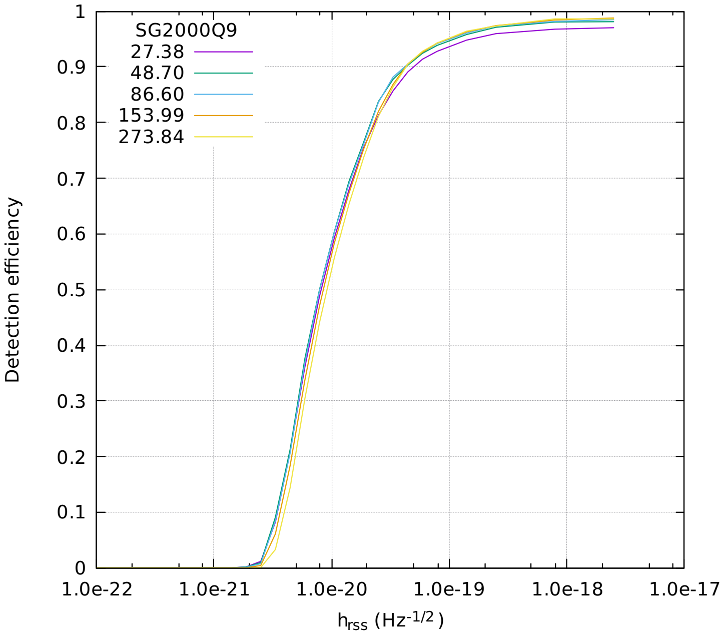


Figure 6.12: Sine-Gaussian sensitivity curve SG2000Q9.

After applying the normalised CC and detector frequency thresholds to the TS trigger set we obtain the false alarm rate dependence on RTCC trigger threshold shown in Fig. 6.13, which we summarise in Table 6.1. From these we can see that the false alarm rate rapidly decreases upon increasing the RTCC threshold, where it achieves a limiting minimum value of $\sim$ 48 nHz above 154.

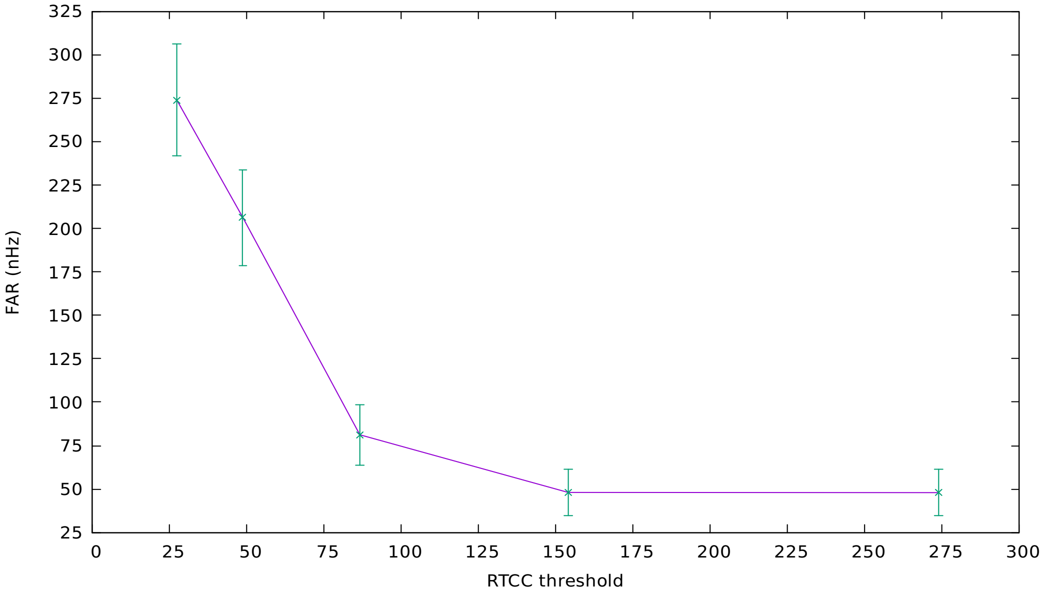


Figure 6.13: False alarm rate dependence on RTCC trigger threshold.

Threshold	FAR (nHz)	(days)
27.38	273 ± 32	42
48.70	206 ± 28	56
86.60	81 ± 17	142
153.99	48 ± 13	241
273.84	48 ± 13	241

Table 6.1: FAR dependence on RTCC threshold as plotted in Fig. 6.13.

In Table 6.2 we collate the results shown in Figs. 6.9 – 6.12 at 50 % detection efficiency $h_{\mathrm{rss}}^{50\%}$ and for comparison those obtained during the 'All-sky search for gravitational-wave bursts in the second joint LIGO-Virgo run' conducted jointly by the LIGO and VIRGO collaborations [78]. Their study uses a network of three detectors (H1, L1, and V1 - the Virgo detector) and the pipeline coherent WaveBurst (cWB) [79]. Coherent WaveBurst has its thresholds configured to achieve a FAR of 1/8 year^{-1}. We cannot make a direct comparison since their pipeline has the advantage of triple coincidence from three detectors which vastly improves upon the SNR obtained from two detectors. However, their study uses LIGO data from the same period (S6) and the same family of SG waveforms.

	RTCC					cWB
	(H1,L1)					(H1,L1,V1)
	Threshold $h_{\mathrm{rss}}^{50\%}$					FAR 1/8 yr^{-1}
Injection	27	48	86	153	273	$h_{\mathrm{rss}}^{50\%}$
SG70Q3	101.7	112.4	116.6	126.1	169.2	18.9
SG70Q8d9	129.9	135.7	167.0	219.7	297.6	21.5
SG70Q100	248.3	382.6	555.9	733.0	1023.0	24.2
SG100Q8d9	49.3	54.6	59.6	70.5	96.0	10.5
SG153Q8d9	32.1	34.5	35.9	39.0	52.3	6.7
SG235Q3	32.8	36.7	36.9	32.8	35.8	5.7
SG235Q8d9	29.1	33.3	33.8	32.3	41.6	5.2
SG235Q100	35.5	53.1	73.5	90.5	124.4	4.6
SG361Q8d9	54.3	60.8	59.5	56.7	72.3	8.6
SG554Q8d9	53.3	55.8	56.6	53.8	57.5	8.9
SG849Q3	59.0	59.2	59.8	59.3	61.8	15.1
SG849Q8d9	54.6	56.5	57.0	58.6	63.5	14.1
SG849Q100	54.5	56.5	57.0	58.5	63.5	12.3
SG1053Q9	63.1	64.6	66.7	69.2	76.6	16.9
SG1304Q9	67.8	67.3	68.6	73.8	82.7	21.1
SG1615Q3	74.9	75.4	76.1	76.2	81.3	41.6
SG1615Q9	71.0	71.0	72.2	73.9	79.0	35.2
SG1615Q100	76.2	83.8	98.1	126.1	170.4	28.3
SG2000Q9	81.6	79.0	79.3	85.2	91.5	30.8

Table 6.2: Sensitivity of the RTCC pipeline at each threshold and results of the LIGO-VIRGO all-sky burst search study at 50 % detection efficiency. Units of $h_{\mathrm{rss}}^{50\%}$ are in 10^{-22} Hz$^{-1/2}$.

In Figs. 6.14 and 6.15 we plot the $h_{\mathrm{rss}}^{50\%}$ values for each SG along with the LIGO-VIRGO results in order to compare the sensitivity trends with increasing SG frequency and Q.

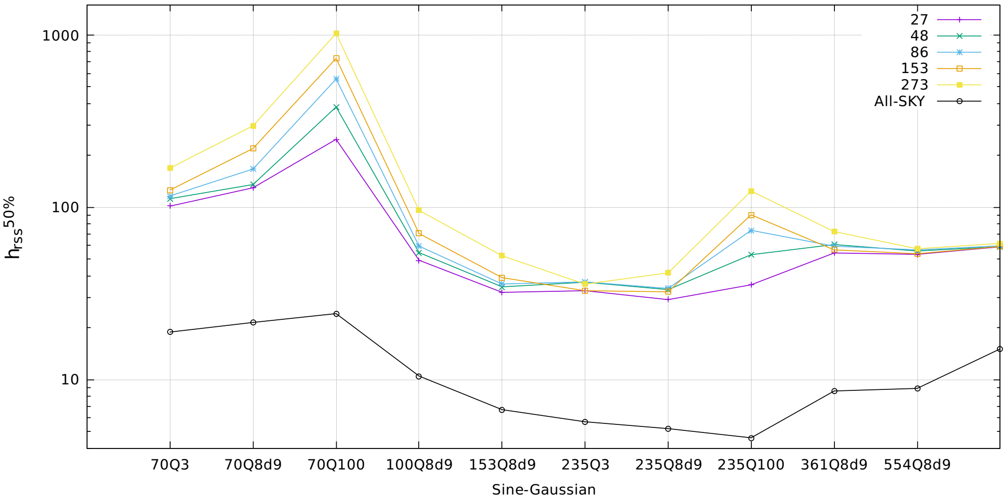


Figure 6.14: Comparison of the trends in detection sensitivity at $h_{\rm rss}^{50\%}$ $(10^{-22}\ {\rm Hz}^{-1/2})$ between RTCC and the All-sky study results for SG injections in the range 70 – 554 Hz.

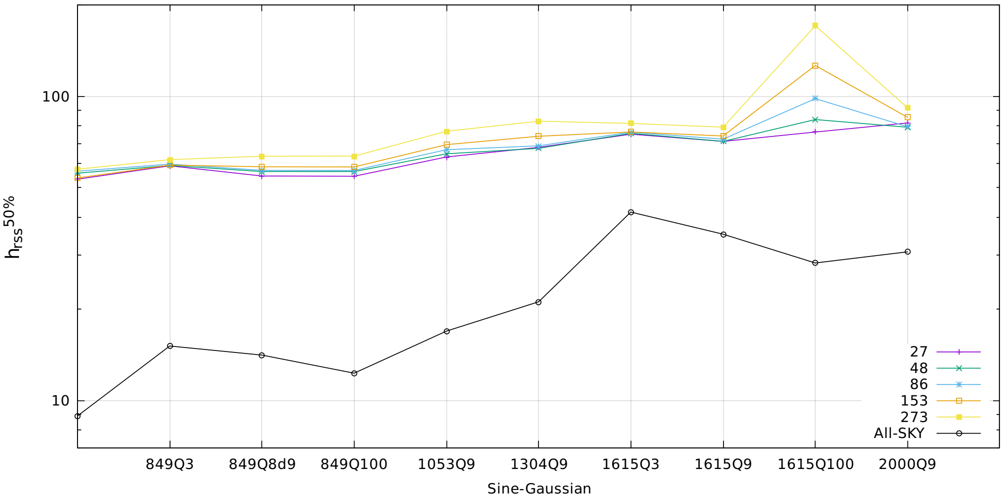


Figure 6.15: Comparison of the trends in detection sensitivity at $h_{\rm rss}^{50\%}$ $(10^{-22}\ {\rm Hz}^{-1/2})$ between RTCC and the All-sky study results for SG injections in the range 849 – 2000 Hz.

These figures show a similar average trend in sensitivity across the injection frequency and that the ratio of RTCC pipeline $h_{\rm rss}^{50\%}$ values to those of the 'All-sky' study is decreasing with increasing frequency, at best approaching a value of ~ 2 for the SG 1615Q3.

6.4 Trigger Lag Error

The difference between the determined trigger lag and the corresponding lag computed from the injection log files is the lag error. The injection lag is taken to be the difference between the H1 and L1 MDC injection times given in the log files. We show the resulting normalised histograms for each SG injection type in figures 6.16 – 6.19 below.

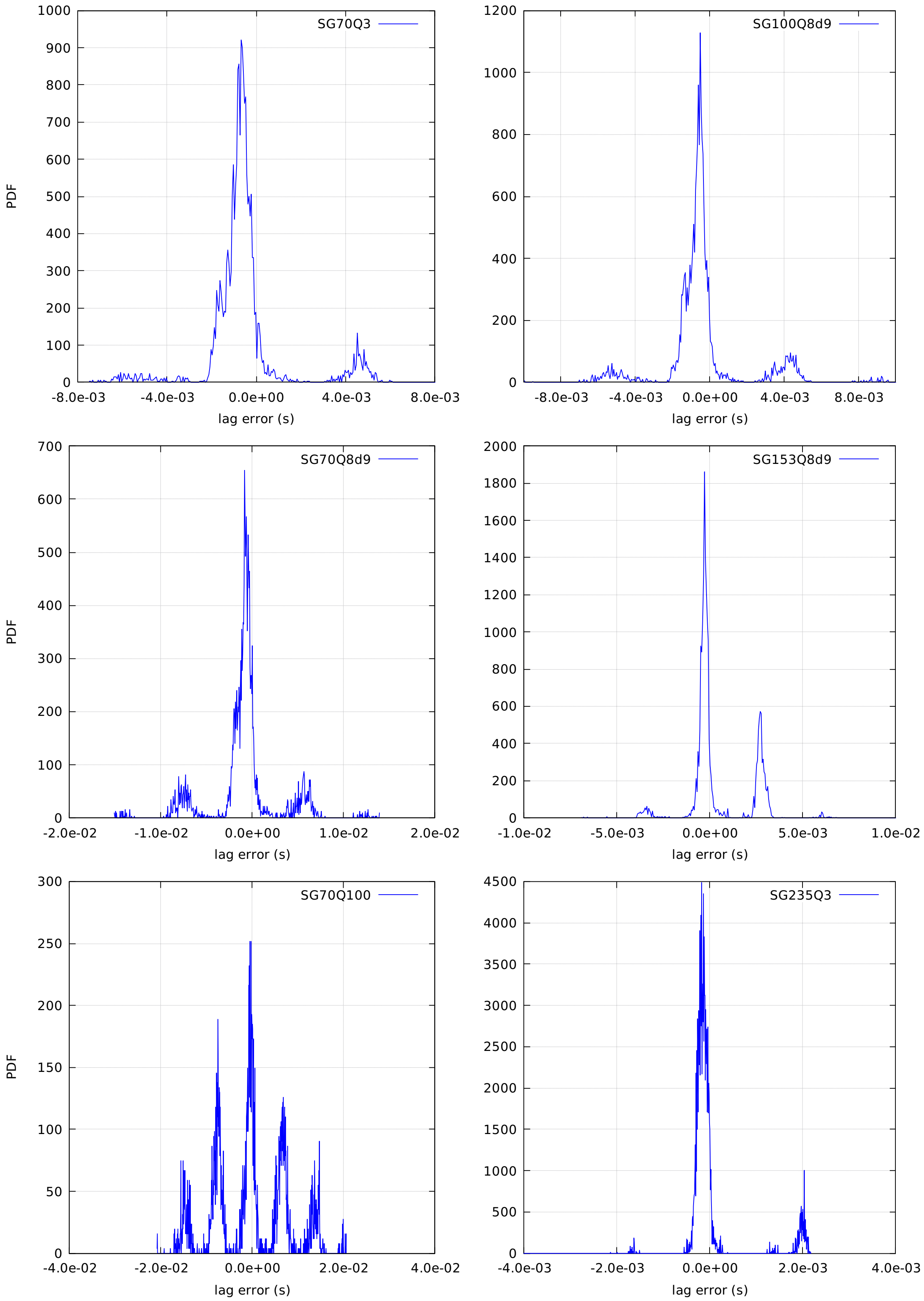


Figure 6.16: SG lag determination error SG 70Q3/8d9/100 100Q8d9 153Q8d9 235Q3.

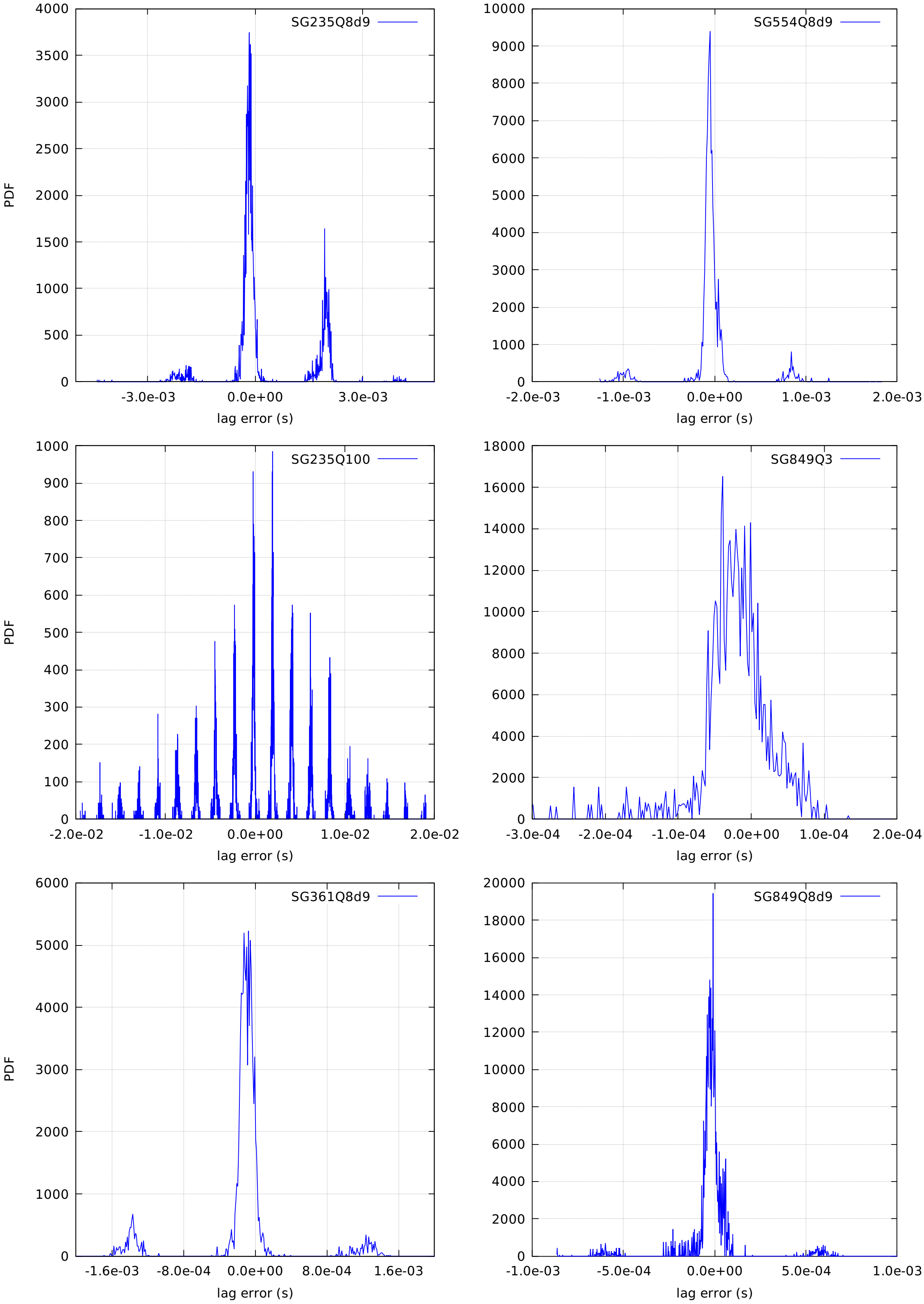


Figure 6.17: SG lag determination error SG 235Q8d9/100 361Q8d9 554Q8d9 849Q3/8d9.

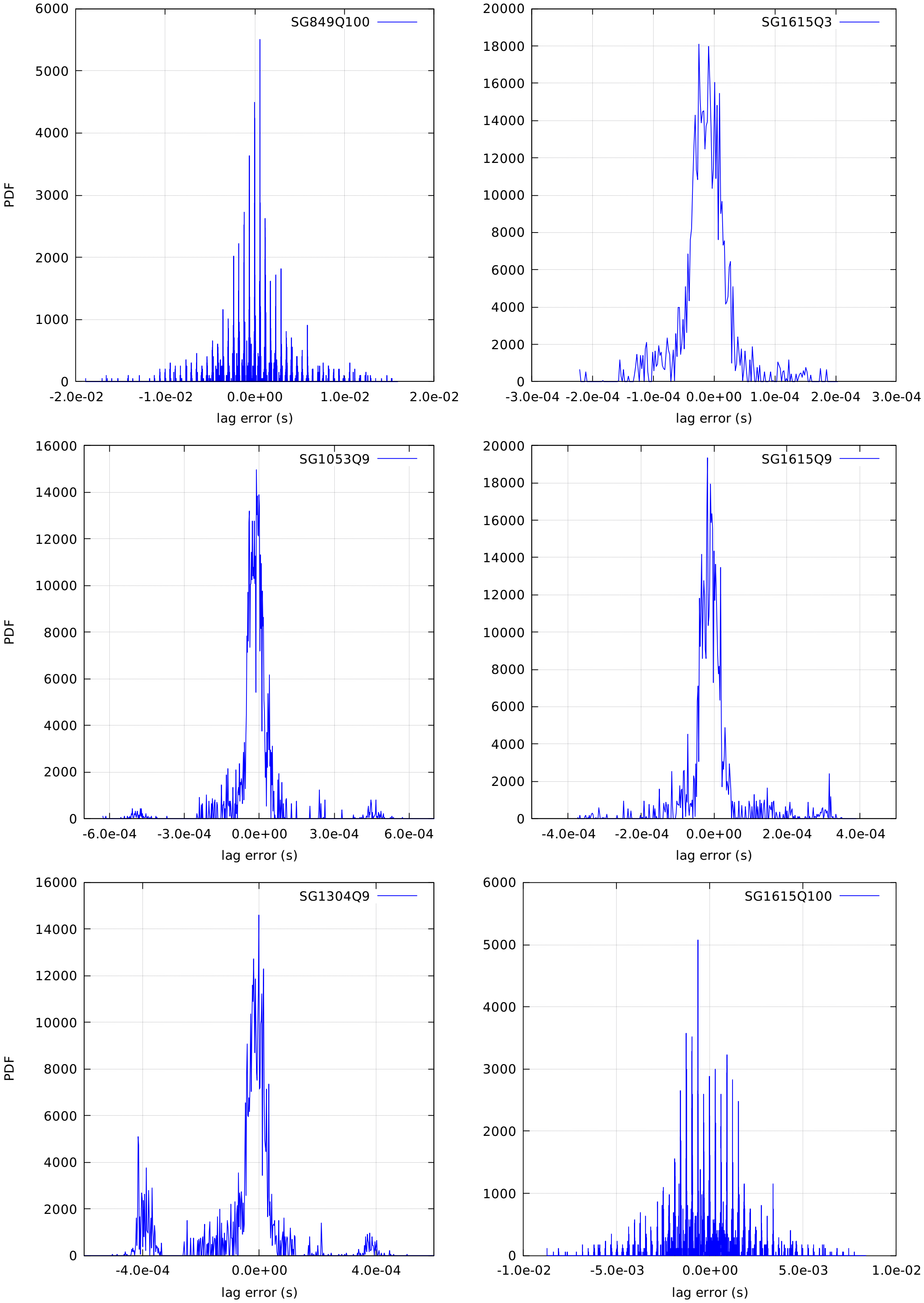


Figure 6.18: SG lag determination error SG 849Q100 1053Q9 1304Q9 1615Q3/9/100.

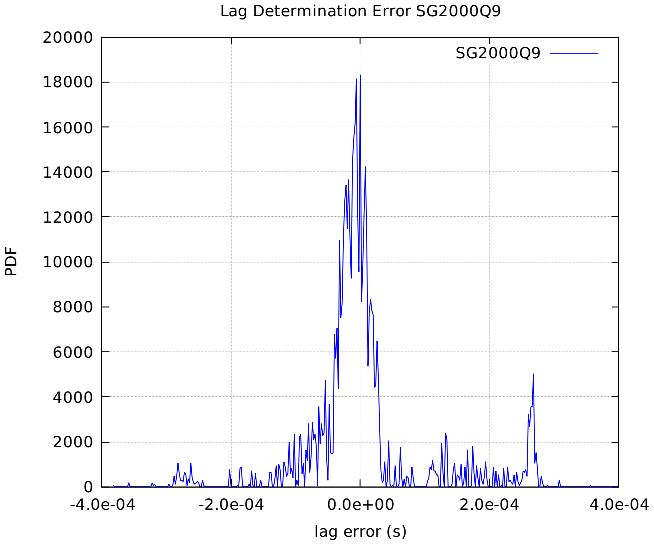


Figure 6.19: Lag determination error SG2000Q9.

Table. 6.3 summarises the above results, giving the maxima of the distributions and where possible some description of general features.

Injection	max (ms)	sepn (ms)	$1/(2f_c)$	description
SG70Q3	-0.66	5.16	7.14	2
SG70Q8d9	-0.84	6.50		5
SG70Q100	-0.46	7.04		small comb 5
SG100Q8d9	-0.53	4.62	5.00	3
SG153Q8d9	-0.29	3.06	3.26	3
SG235Q3	-0.18	2.14	2.13	2
SG235Q8d9	-0.16	1.99		3
SG235Q100	1.94	2.13		comb 19
SG361Q8d9	-0.08	1.27	1.39	3
SG554Q8d9	-0.06	0.92	0.90	3
SG849Q3	-0.04		0.59	
SG849Q8d9	-0.01	0.58		3
SG849Q100	0.56	0.59		comb 50 (est)
SG1053Q9	-0.01	0.47	0.47	3
SG1304Q9	-0.00	0.38	0.38	3
SG1615Q3	-0.02		0.31	
SG1615Q9	-0.02			
SG1615Q100	-0.64	0.31		comb 44 (est)
SG2000Q9	-0.00		0.25	

Table 6.3: Summary of lag error. The distribution maximum (max) and approximate peak separation (sepn) are given. The value of $1/(2f_c)$ is given at each waveform central frequency for comparison with the peak separation. When multiple peaks are present, a brief description of the distribution is given. The number of peaks have been estimated (est) in some cases.

The determination of trigger lag depends upon locating the peaks in the trigger data which produce a maximum RTCC correlation, and because RTCC maxima can be positive for crest or trough then neighbouring maxima may occur at twice the SG waveform frequency. This can most clearly be seen for the higher Q waveforms which display a comb of peaks separated in time by $1/(2f_c)$. Lower SG frequency results in poorer lag determination at low frequency as shown in the result for SG70Q100.

6.5 Trigger Timing Error

Here we compare the H1 peak RTCC trigger times with the H1 SG injection time from the MDC injection log files. Figures 6.20 – 6.23 show the normalised distribution of the difference between trigger time and log injection time for each SG waveform. The results are summarised in Table. 6.4 where we give the location(s) of the distribution maxima, average separation of local maxima and general description.

As with lag determination, timing again depends upon locating the trigger signal maxima and the results show large timing errors for Q100 signals and improvement with increasing frequency.

Injection	max (ms)	sepn (ms)	$1/(2f_c)$	description
SG70Q3	17.43	4.46	7.14	4
SG70Q8d9	24.36	6.79		5
SG70Q100	17.33	7.15		comb 40
SG100Q8d9	12.45	4.87	5.00	3
SG153Q8d9	7.91	3.11	3.26	4
SG235Q3	4.49	2.70	2.13	2
SG235Q8d9	4.63			
SG235Q100	4.58	2.11		comb 42
SG361Q8d9	0.86	0.70	1.39	4
SG554Q8d9	1.17	0.43	0.90	4
SG849Q3	1.06	0.31	0.59	2
SG849Q8d9	1.15	0.27		4
SG849Q100	1.08	0.59		comb 26
SG1053Q9	1.05		0.47	
SG1304Q9	1.06		0.38	
SG1615Q3	1.06		0.31	
SG1615Q9	1.04			
SG1615Q100	1.07	0.30		comb 26
SG2000Q9	1.38	0.30	0.25	2

Table 6.4: Summary of timing error. The distribution maximum (max) and approximate peak separation (sepn) are given. The value of $1/(2f_c)$ is given at each waveform central frequency for comparison with the peak separation. When multiple peaks are present, a brief description of the distribution is given. The number of peaks have been estimated (est) in some cases.

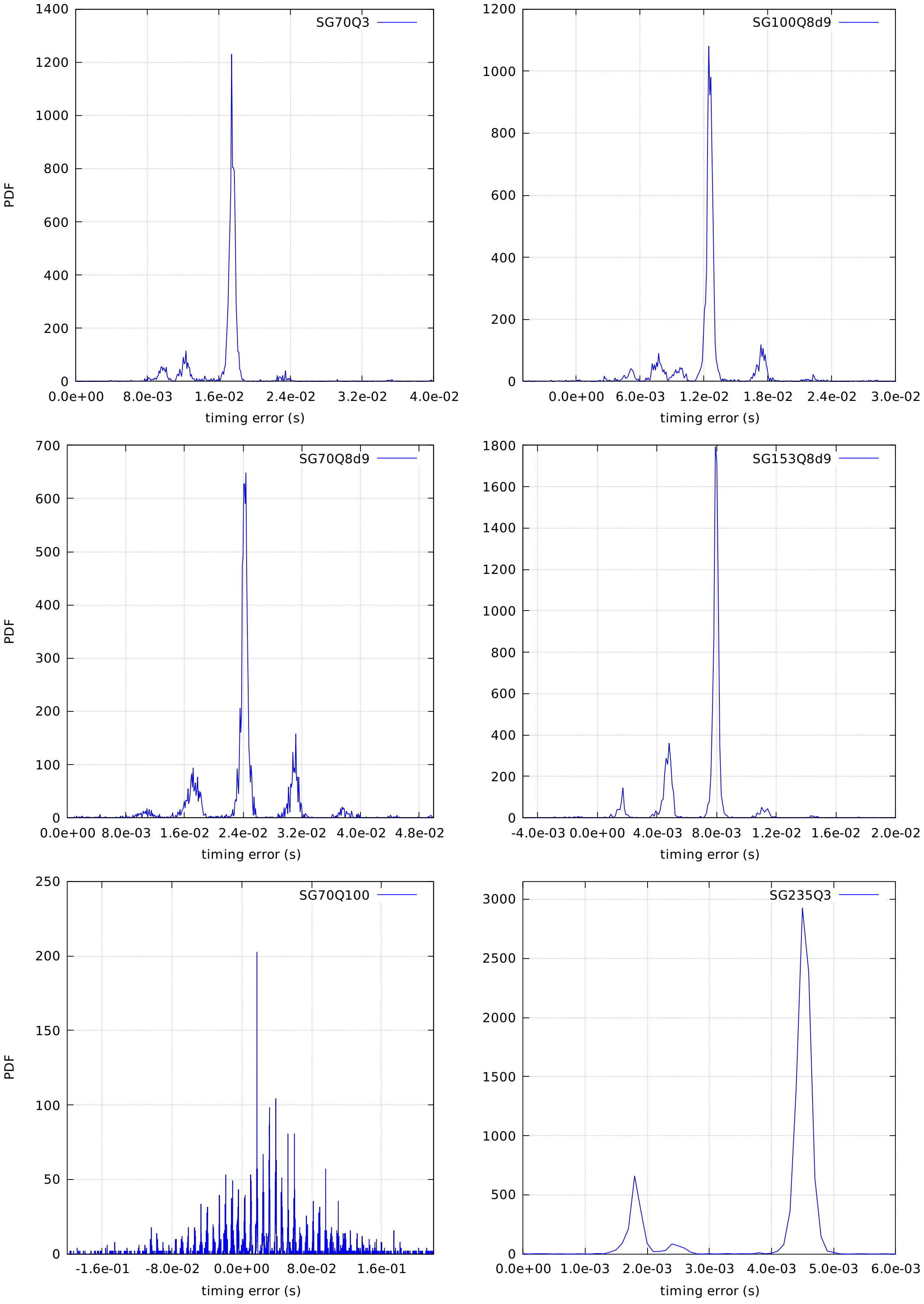


Figure 6.20: Timing error SG 70Q3/8d9/100 100Q8d9 153Q8d9 235Q3.

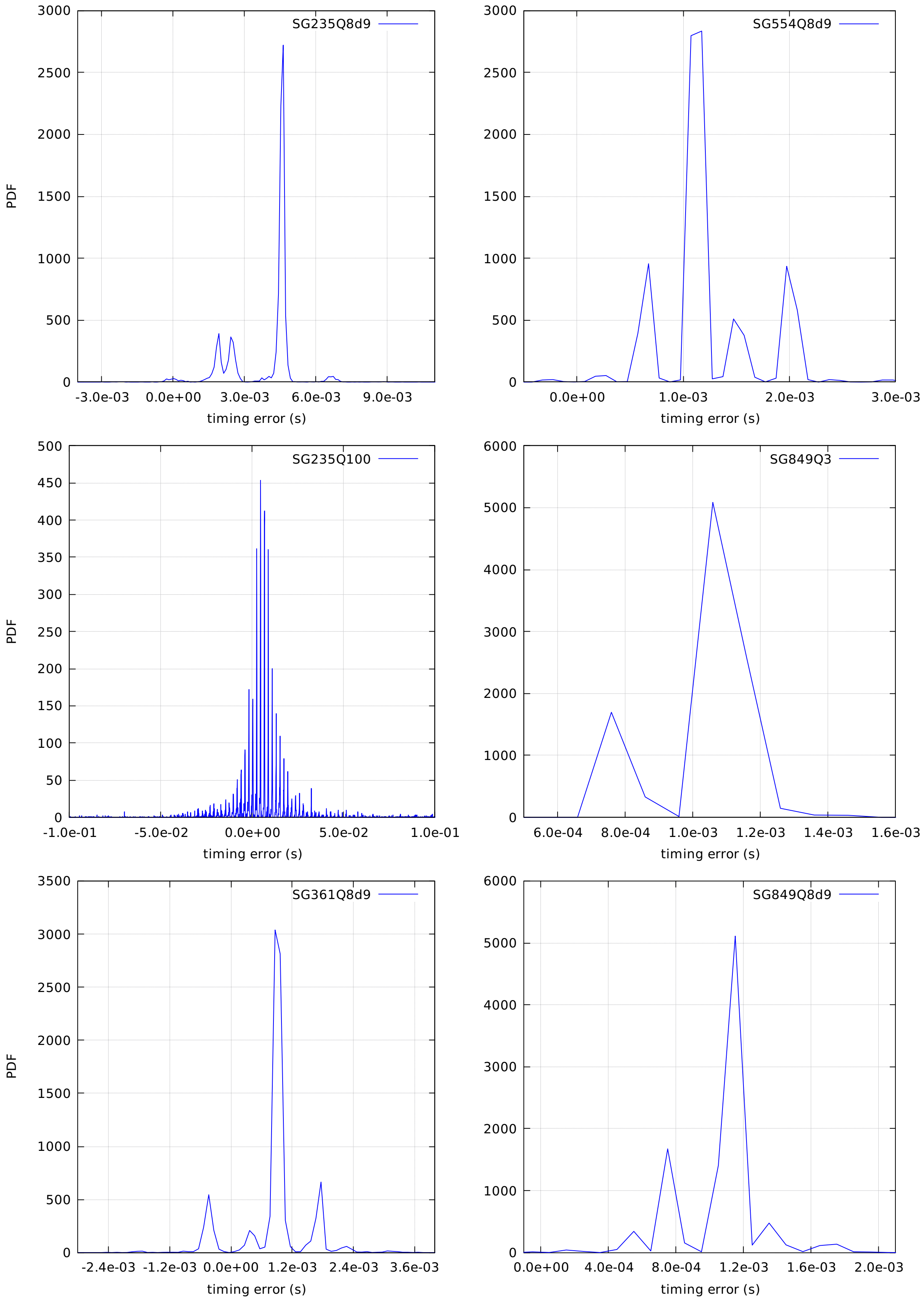


Figure 6.21: Timing error 235Q8d9/100 361Q8d9 554Q8d9 849Q3/8d9.

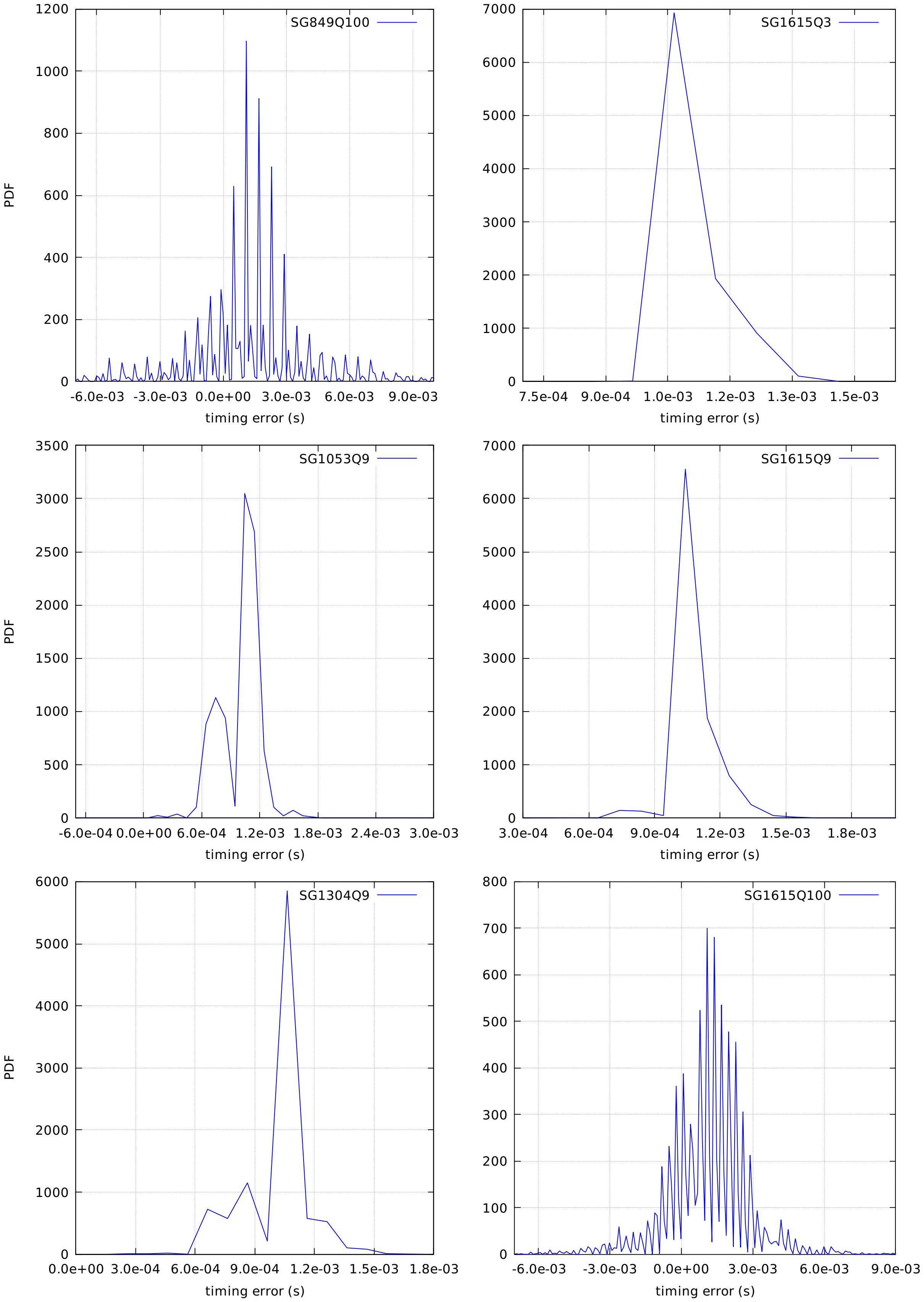


Figure 6.22: Timing error SG 849Q100 1053Q9 1304Q9 1615Q3/9/100.

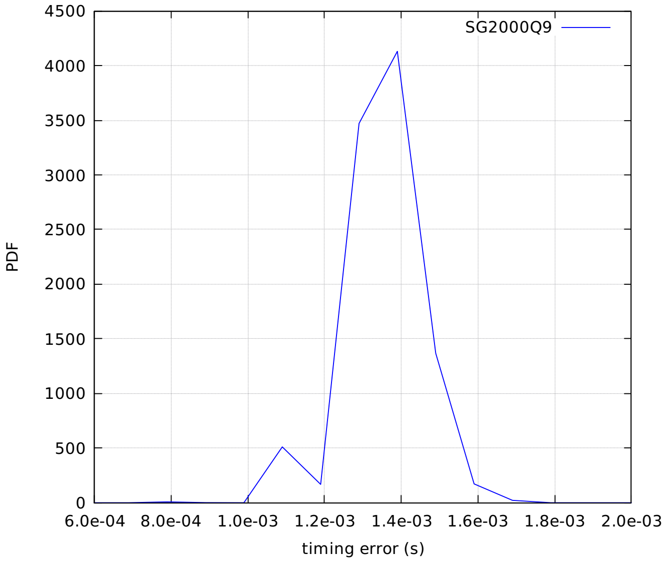


Figure 6.23: Timing error SG2000Q9.

6.6 Frequency Estimation

In this section we provide the results of frequency estimation of the captured trigger data from each detector. The $F_{\rm rat}$ cut ensures that all accepted SG trigger frequencies must lie within the defined proximity.

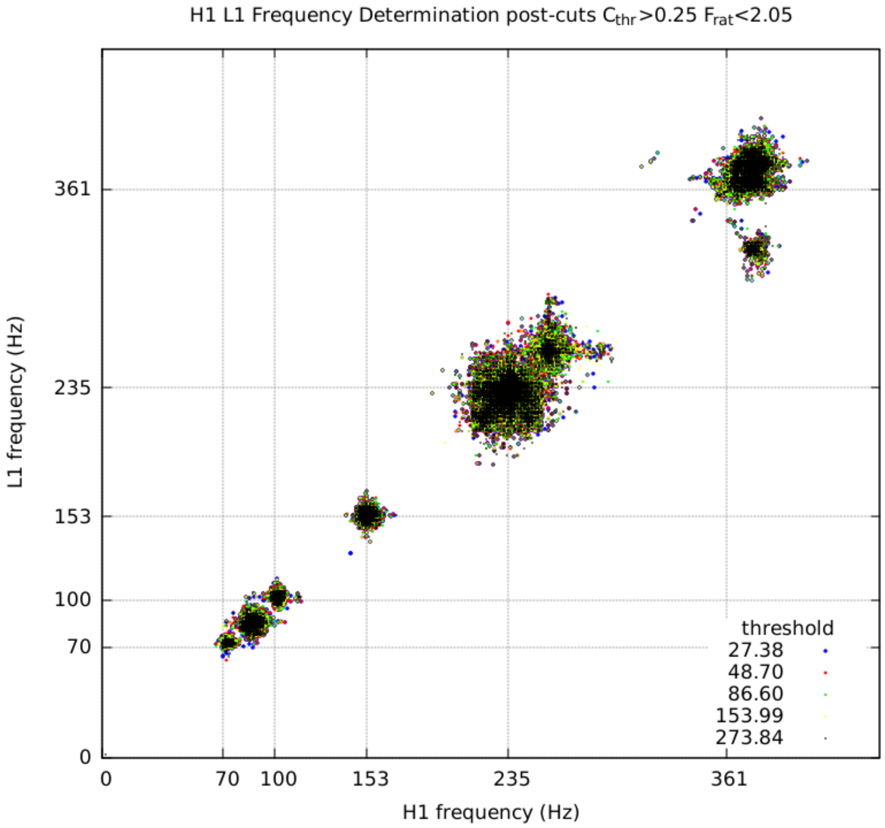


Figure 6.24: H1 L1 trigger frequency estimation scatterplot 0 – 450 Hz.

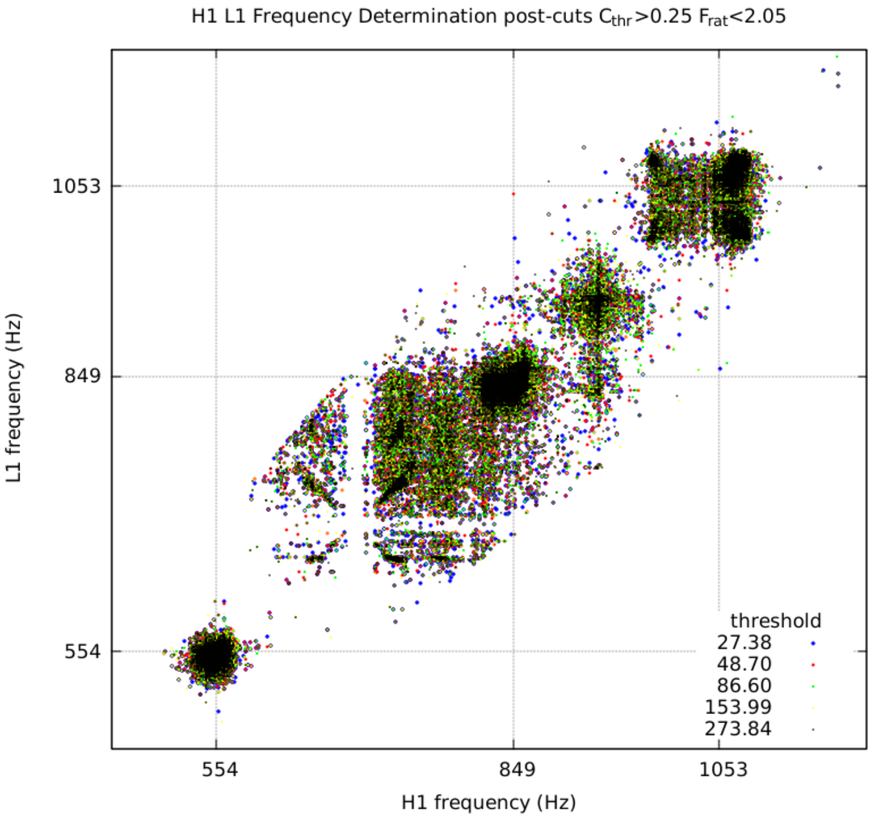


Figure 6.25: H1 L1 trigger frequency estimation scatterplot 450 – 1200 Hz..

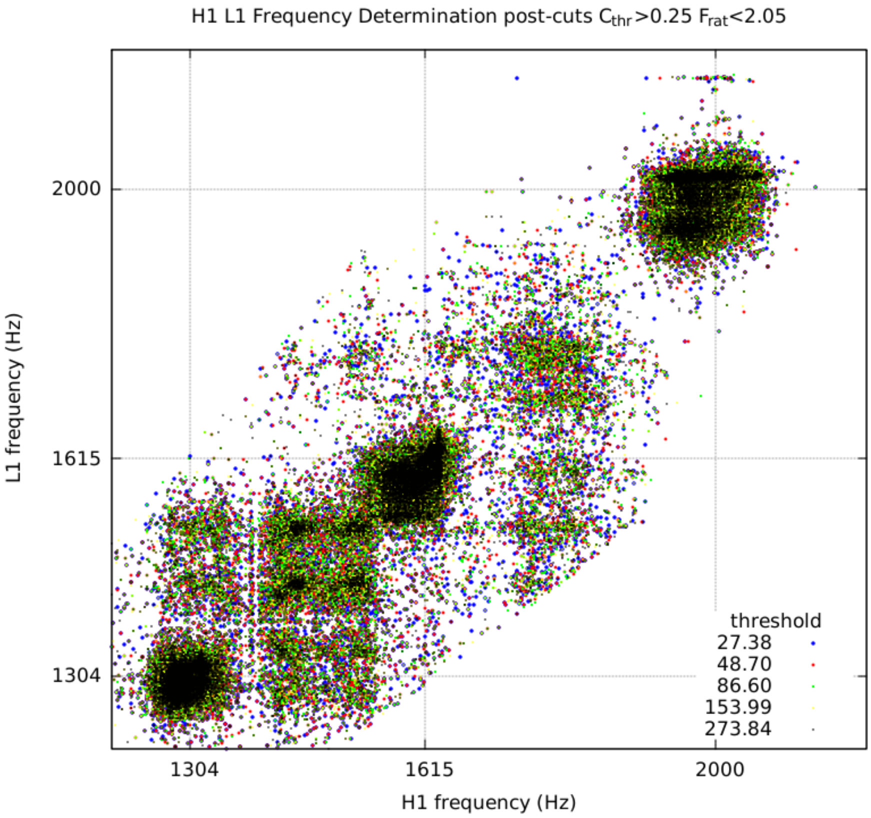


Figure 6.26: H1 L1 trigger frequency estimation scatterplot 1200 – 2200 Hz..

Prior to the F_{rat} cut there were clear indications of cases where frequency estimation was correct (or acceptable) for one detector signal but poorly determined in the other, resulting in the rejection of the trigger. For a given SG injection we expect the frequency estimation to be distributed across the SG bandwidth $\Delta f = f_c/Q$ and therefore good frequency identification at low injection frequency or high Q values.

In Figs. 6.24 – 6.26 above we plot trigger frequency determination at each RTCC threshold with axis labels at the injection frequencies. These show definite structure and correspondence to the SG injection frequencies. Frequency determination is better for high amplitude injections which can be seen by the closer clustering at higher thresholds. However, some trigger frequencies are systematically misassigned. For example, in Fig. 6.24 there is a cluster of triggers between 70 Hz and 100 Hz and an additional higher frequency cluster near to the 235 Hz group.

In Fig. 6.25 we can see a zone around $\sim$ 690 Hz in which frequency selection appears to be excluded. This effect can also be seen to a lesser extent in Fig. 6.26 between 1304 – 1615 Hz. We believe that this is a boundary effect between frequency selection for different injections rather than direct exclusion.

The above plots show only the spread of trigger frequencies and in some cases leaves ambiguity as to which injection frequency the trigger pixel or cluster belongs. In Figures 6.27 – 6.30 we show the two dimensional histogram distribution of trigger frequency for each SG waveform. In general the distributions are spread in approximate agreement with the SG bandwidth and show some symmetry about the $f_{\mathrm{L1}} = f_{\mathrm{H1}}$ axis. All SG $Q = 100$ distributions display simple maxima and give accurate frequency selection. The distributions for $Q = 3, 8.9$ injections less than 235 Hz are again simple but show a bias towards overestimation of the injection frequency. For $Q = 3, 8.9$ injections of frequency 235 Hz and above the distributions are no longer simple. These distributions are quite complex in some cases, showing sparsely dispersed frequency selection along with clear singular or multiple maxima. We cannot explain the observed structure of these distributions but expect that non-stationarity of the detector noise plays some part.

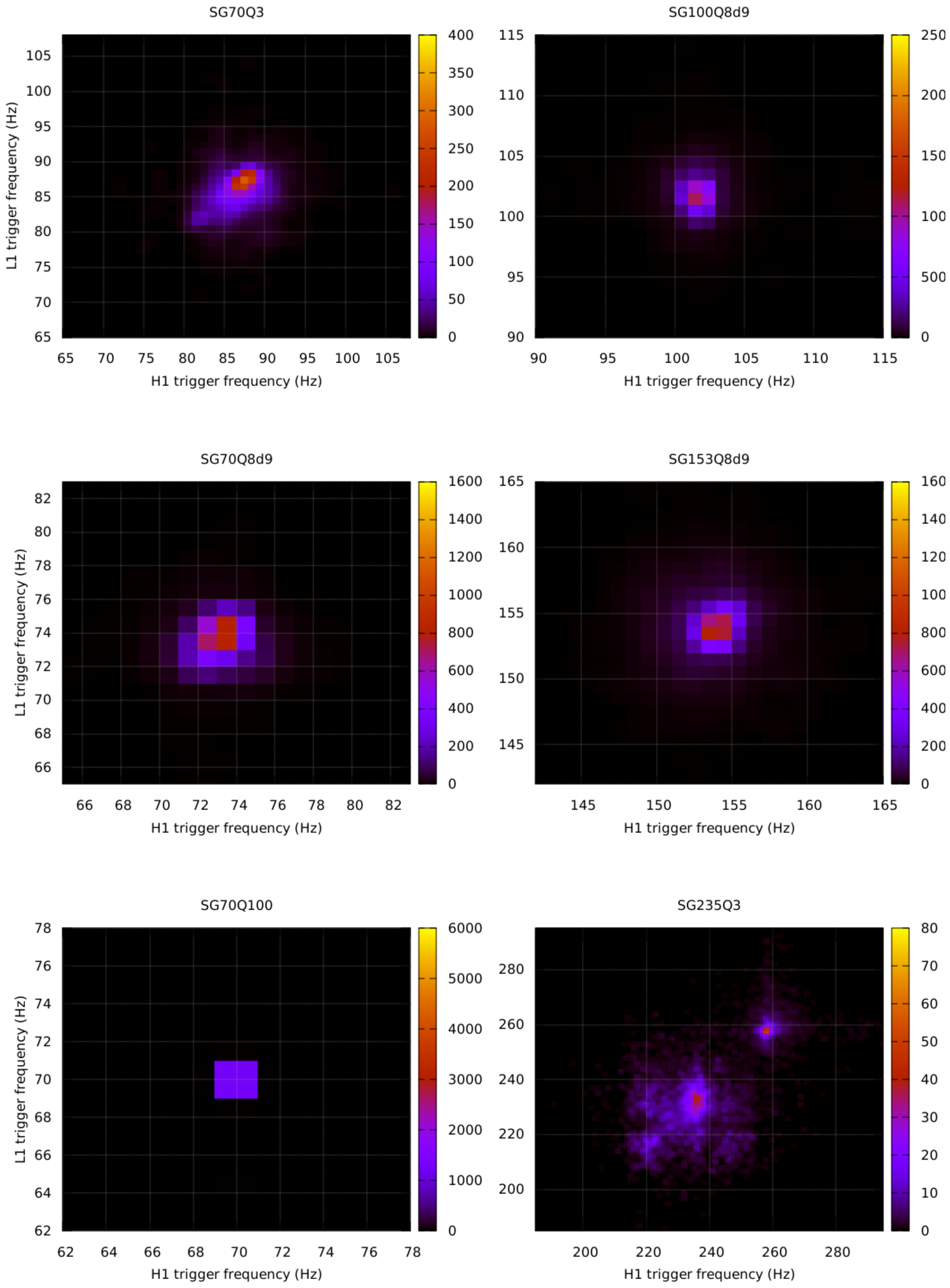


Figure 6.27: Frequency distribution SG 70Q3/8d9/100 100Q8d9 153Q8d9 235Q3.

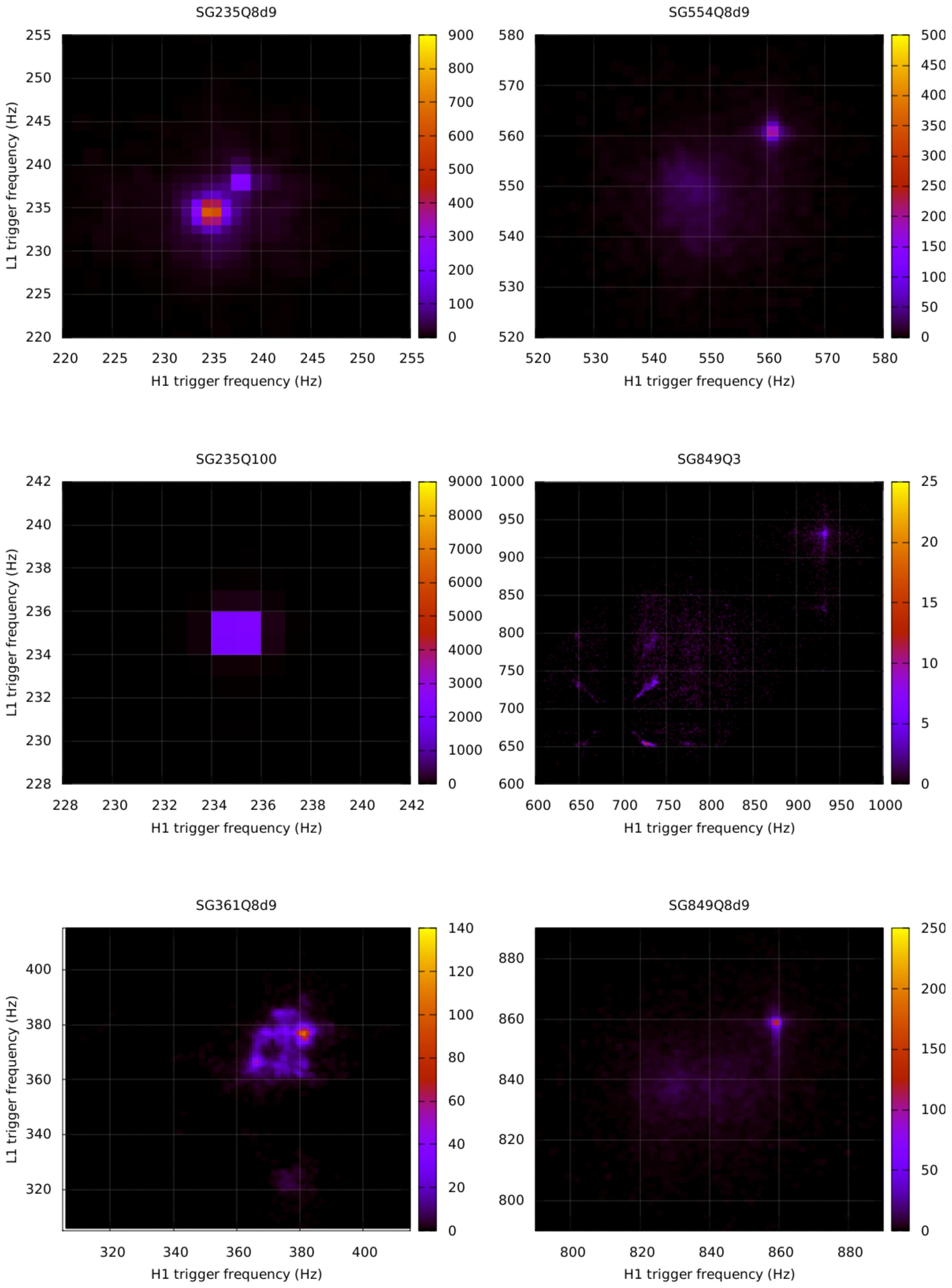


Figure 6.28: Frequency distribution SG 235Q8d9/100 361Q8d9 554Q8d9 849Q3/8d9.

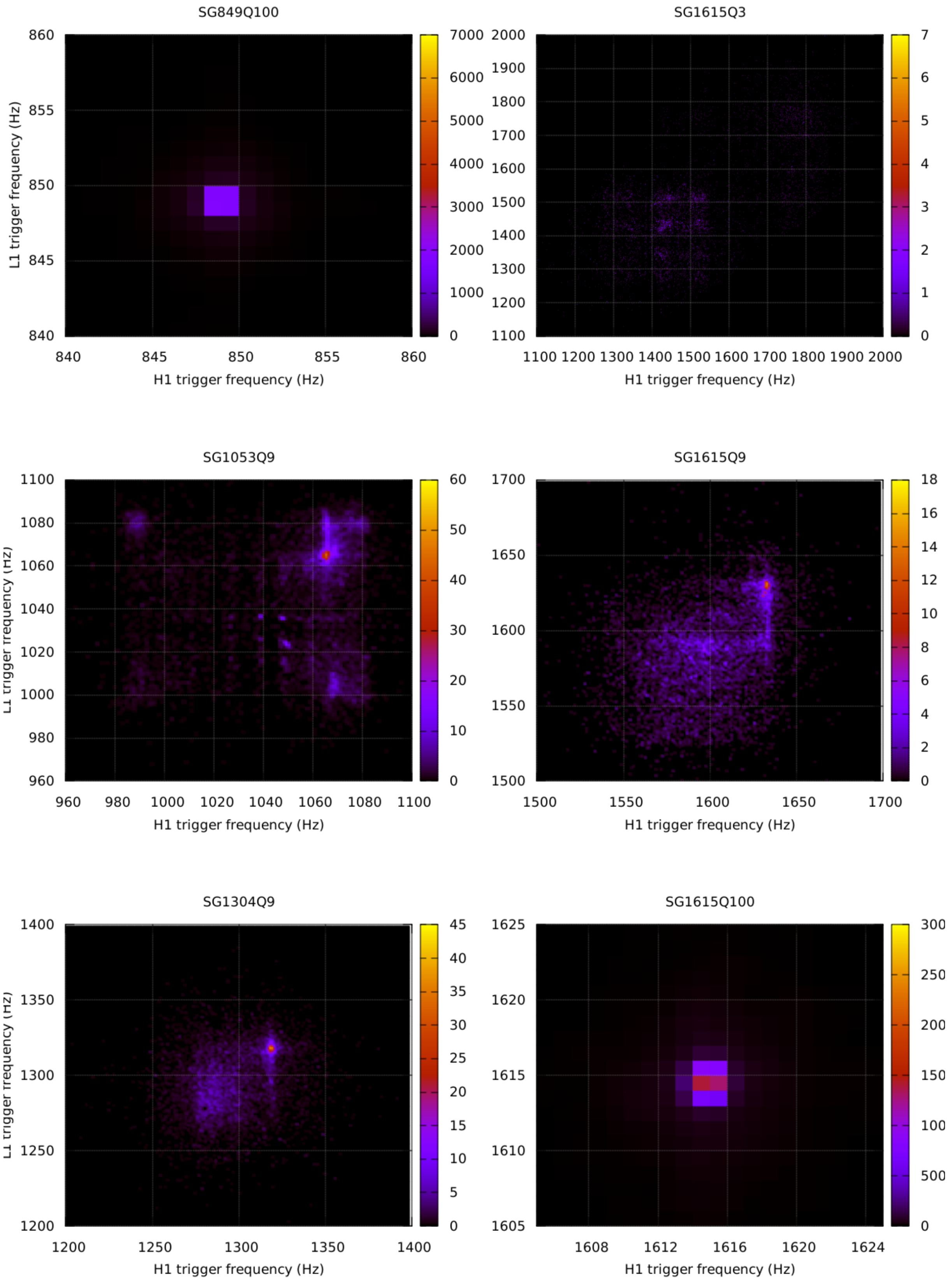


Figure 6.29: Frequency distribution SG 849Q100 1053Q9 1304Q9 1615Q3/9/100.

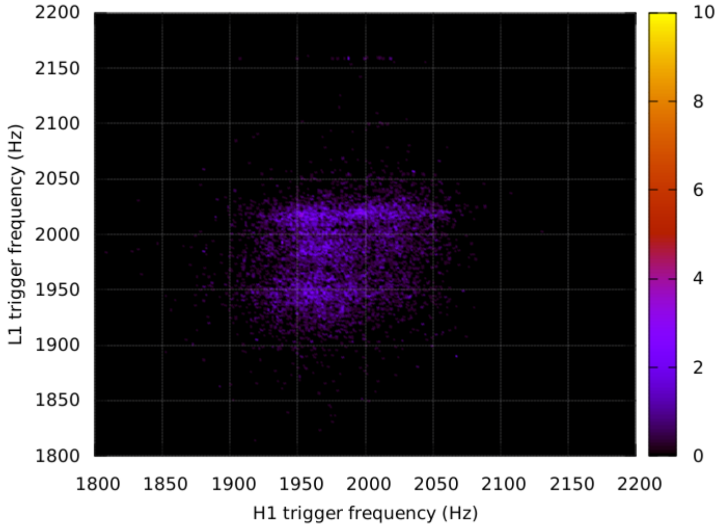


Figure 6.30: Frequency distribution SG2000Q9.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

The nucleus for the research presented in this work was the RTCC algorithm, a cross-correlation indicator which uses exponential memory rather than the sliding dot product used in standard cross-correlation and we were motivated by its potential use as a low latency trigger generator for GW searches.

We have shown that the RTCC algorithm is capable of close replication of standard cross-correlation output at a lower computational cost. Rather than adapt an existing pipeline to accommodate the RTCC algorithm output, we have developed a complete pipeline relying entirely upon time domain methods. Although the method of trigger frequency determination used was relatively slow, we have demonstrated that the main pipeline operations are capable of execution in real time or with very low latency.

We found direct comparison of the accuracy our trigger parameterisations (times of arrival, lag, and frequency) difficult, as often, when presented at all in a given study, these tended to be summary statistics (see comments below).

The primary limiting factors affecting the performance of the RTCC pipeline in terms of the sensitivities and false alarm rates obtained were, in our opinion, due to the signal processing procedure adopted. We almost certainly could have achieved better sensitivity if the LPEF filter was not in part negating the effects of excessive line subtraction. Nevertheless, at the highest RTCC threshold with a false alarm rate of 1/241 days the measured 50 % detection sensitivities are generally within an order of magnitude of those obtained in the comparison study.

For the above reasons we believe that the embryonic RTCC pipeline is worthy of further development.

7.2 Future Work

We were, perhaps, a little too pragmatic in our acceptance of the LPEF filter to remedy our poor implementation of iWave line removal. Therefore, a more adaptive line removal procedure should be developed. This would, as was originally intended, leave the LPEF filter to perform whitening and should give higher sensitivity. Further improvement in sensitivity would also be expected by implementing iWave line removal in its adaptive mode rather than in static mode.

A natural development of the pipeline would be to enable processing data input from three or more detectors to perform RTCC correlation on detector pairs. The pipeline would then benefit from an increased ability to reject noise events and the potential to localise sources. Since the common objective of GW burst analysis pipelines is source localisation, we might then find it easier to directly compare the source localisation errors resulting from lag determination errors.

The following comments are suggestions, some of which are speculative.

- Our pipeline used only one instance of the RTCC algorithm and given the rapidity of the algorithm we might consider running the pipeline with multiple instances with a range of values of N.

- Preliminary tests suggest that the Gaussian-spike filter might have use as a general preprocessing filter (after usual filtering) given that the process may significantly improve SNR and has little computational cost.

- The use of the RTCC algorithm is clearly not limited to GW analysis pipelines and its use in other data analysis fields could be considered, for example, in seeking correlations in complex data sets where, for example, standard cross-correlation methods may be computationally expensive or prohibitive.

Appendix

RTCC Pipeline Code Listing

In the following two pages we list the c code implementation of the real time cross-correlation indicator `corest.c`. The remainder of this appendix lists the full RTCC pipeline c++ code `INJTRIG.cpp`. Important sections have been commented.

```
//clive tomlinson
//University of Sheffield ‘‘1/11/2015
//RTCC Correlator
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "corest.h"
#include <string.h>

int init_correlator(cordata* pmydata, int maxlag,double w,int reclength)
{
  pmydata->txdata=calloc(3*maxlag+1,sizeof(double));//det1 data array
  pmydata->tydata=calloc(3*maxlag+1,sizeof(double));//det1 data array
  pmydata->pcorrdata=calloc(2*maxlag+1,sizeof(double));//correlation array
  pmydata->dcount=0;//data count
  pmydata->weight=w;
  pmydata->maxlag=maxlag;//maximum sample lag
  pmydata->reclength=reclength;//set trigger recording length in samples
  pmydata->rec=0;//record flag
  pmydata->reccount=0;//record sample counter
  //allocate recording array
  pmydata->xrecdata=calloc(pmydata->reclength,sizeof(double));
  pmydata->yrecdata=calloc(pmydata->reclength,sizeof(double));
  return 0;
}
int run_correlator(cordata* pmydata, double nx, double ny)
{
  int i;
  //correlate at each lag offset
  pmydata->pcorrdata[pmydata->maxlag]=
    (1-pmydata->weight)*pmydata->pcorrdata[pmydata->maxlag]+
    pmydata->weight*pmydata->txdata[2*pmydata->maxlag]*
    pmydata->tydata[2*pmydata->maxlag];
  pmydata->maxcor=fabs(pmydata->pcorrdata[pmydata->maxlag]);
  pmydata->maxcorlag=0;
  for(i=0;i<pmydata->maxlag;++i)
    {
    pmydata->pcorrdata[pmydata->maxlag+i+1]=
        (1-pmydata->weight)*pmydata->pcorrdata[pmydata->maxlag+i+1]+
        pmydata->weight*pmydata->txdata[2*pmydata->maxlag]*
        pmydata->tydata[2*pmydata->maxlag-i-1];
     pmydata->pcorrdata[pmydata->maxlag-i-1]=
        (1-pmydata->weight)*pmydata->pcorrdata[pmydata->maxlag-i-1]+
        pmydata->weight*pmydata->txdata[2*pmydata->maxlag]*
        pmydata->tydata[2*pmydata->maxlag+i+1];
     if(fabs(pmydata->pcorrdata[pmydata->maxlag+i+1])>pmydata->maxcor)
       {
         pmydata->maxcor=fabs(pmydata->pcorrdata[pmydata->maxlag+i+1]);
         pmydata->maxcorlag=i+1;
       }
     if(fabs(pmydata->pcorrdata[pmydata->maxlag-i-1])>pmydata->maxcor)
       {
         pmydata->maxcor=fabs(pmydata->pcorrdata[pmydata->maxlag-i-1]);
         pmydata->maxcorlag=-i-1;
       }
    }
  if(pmydata->rec==1)
    {//record trigger data
      pmydata->xrecdata[pmydata->reccount]=pmydata->txdata[0];
      pmydata->yrecdata[pmydata->reccount]=pmydata->tydata[0];
      pmydata->reccount++;
      if(pmydata->reccount==pmydata->reclength)
        {
          pmydata->rec=0;
          pmydata->reccount=0;
        }
      pmydata->xpow+=pmydata->txdata[2*pmydata->maxlag]*
        pmydata->txdata[2*pmydata->maxlag];
      pmydata->ypow+=pmydata->tydata[2*pmydata->maxlag]*
        pmydata->tydata[2*pmydata->maxlag];
    }
```

```c
  memcpy(pmydata->txdata,pmydata->txdata+1,(3*pmydata->maxlag)*sizeof(double));
  pmydata->txdata[3*pmydata->maxlag]=nx;
  memcpy(pmydata->tydata,pmydata->tydata+1,(3*pmydata->maxlag)*sizeof(double));
  pmydata->tydata[3*pmydata->maxlag]=ny;
  pmydata->dcount=(pmydata->dcount+1)%pmydata->maxlag;
  return 0;
}
int destroy_correlator(cordata* pmydata)
{
  free(pmydata->txdata);
  free(pmydata->tydata);
  free(pmydata->pcorrdata);
  free(pmydata->xrecdata);
  free(pmydata->yrecdata);
  return 0;
}
int corest_copy(double* destination, cordata* pmydata)
{
  unsigned int nbytes;
  nbytes=(2*pmydata->maxlag+1)*sizeof(double);
  memcpy((void*)destination,(void*)pmydata->pcorrdata,nbytes);
  return 0;
}
int corest_dump(cordata* pmydata)
{
  int i;
  printf("correlation buffer:\n");
  for(i=0;i<(2*(pmydata->maxlag)+1);i++)
    {
      printf("%d\t%lf\t%d\n",i,pmydata->pcorrdata[i],pmydata->dcount);
    }
  return 0;
}
int corest_copystruct(cordata* ptarget, cordata data)
{
  ptarget->maxlag=data.maxlag;
  ptarget->weight=data.weight;
  ptarget->txdata=calloc(3*data.maxlag,sizeof(double));
  ptarget->tydata=calloc(3*data.maxlag,sizeof(double));
  ptarget->xrecdata=calloc(16384,sizeof(double));
  ptarget->yrecdata=calloc(16384,sizeof(double));
  memcpy((void*)ptarget->txdata,(void*)data.txdata,3*data.maxlag*sizeof(double));
  memcpy((void*)ptarget->tydata,(void*)data.tydata,3*data.maxlag*sizeof(double));
  memcpy((void*)ptarget->xrecdata,(void*)data.xrecdata,16384*sizeof(double));
  memcpy((void*)ptarget->yrecdata,(void*)data.yrecdata,16384*sizeof(double));
  ptarget->pcorrdata=calloc((2*data.maxlag+1),sizeof(double));
  memcpy((void*)ptarget->pcorrdata,(void*)data.pcorrdata,
         (2*data.maxlag+1)*sizeof(double));
  ptarget->dcount=data.dcount;
  return 0;
}
```

```
//clive tomlinson
//University of Sheffield 1/11/2015
//RTCC Pipeline
//N.B. RTCC correlator alias Corest, IWave alias PIIR
#include <iostream>
#include <fstream>
#include <sstream>
#include <stdlib.h>
#include "Dacc.hh"
#include "TSeries.hh"
#include "PIIR.hh"
#include "Corest.hh"
#include "corest.h"
#include "IIRFilter.hh"
#include "TMath.h"
#include "LPEFilter.hh"
#define pi 3.14159265

double CORRcon(const char* conffile, double *mean, int *nlin, double *fs)
{//Pipeline configuration returns mean and number lines for removal from
  //configuration file
  std::ifstream config;
  std::string line;
  double sigma;
  int lcount=0;
  int pcs=0;
  int pce=0;
  config.open(conffile);
  while(config.good())
    {
      getline(config,line);
      if(line=="sampling frequency")
        {
          getline(config,line);
          *fs=std::atof(line.c_str());
        }
      if(line=="sigma")
        {
          getline(config,line);
          sigma=std::atof(line.c_str());
        }
      if(line=="mean")
        {
          getline(config,line);
          *mean=std::atof(line.c_str());
        }
      lcount++;
      if(line=="piir")
        {
          pcs=lcount+1;
        }
      if(line[0]==’n’)
        {
          pce=lcount-1;
        }
    }
  *nlin=pce-pcs+1;
  config.close();
  return sigma;
}

IIRFilter CORRcon(const char* conffile,double *ftrans, std::string filtname)
{//return IIR filter from config file and filter transient time
  std::ifstream config;
  std::string line;
  int ccount=0;
  int ec;
  double a0=1;
  double a[5];
  double val;
  double fs;
  config.open(conffile);
```

```
 while(config.good())
  {
   getline(config,line);
   if(line=="sampling frequency")
    {
     getline(config,line);
     fs=std::atof(line.c_str());
    }
   if(line=="filter transient time(s)")
    {
     getline(config,line);
     *ftrans=std::atof(line.c_str());
    }
  }
 config.close();
 IIRFilter IIR(fs);
 config.open(conffile);
 while(config.good())
 {
   getline(config,line);
   if(line==filtname)
    {
     getline(config,line);
     while(line[0]!='t')
      {
       ccount++;
       ec=0;
       while(line[ec]!='=')
        {
         ec++;
        }
       line.erase(line.begin(),line.begin()+ec+1);
       val=std::atof(line.c_str());
       a[(ccount-1)%5]=val;
       if((ccount-1)%5==4)
        {
         IIRSos SOS(a[0],a[1],a[2],a0,a[3],a[4]);
         IIR*=SOS;
        }
       getline(config,line);
      }
    }
 }
 config.close();
 return IIR;
}

int CORRcon(const char* conffile, double* freq, double* tau)
{//return line removal frequency array and IWave tuning parameter tau
 std::ifstream config;
 std::string line,linecopy;
 int lcount=0;
 int pcs,pce;
 config.open(conffile);
 while(config.good())
  {
   getline (config,line);
   lcount++;
   if(line=="piir")
    {
     pcs=lcount+1;
    }
   if(line[0]=='n')
    {
     pce=lcount-1;
    }
  }
 int nlin=pce-pcs+1;
 config.close();
 config.open(conffile);
 lcount=0;
 while(config.good())
```

```
  {
      getline (config,line);
      lcount++;
      if(lcount>=pcs && lcount<=pce)
        {
          int sc=0;
          while(line[sc]!=’\t’)
            {
              sc++;
            }
          linecopy=line;
          line.erase(line.begin(),line.begin()+sc+1);
          linecopy.erase(linecopy.begin()+sc+2,linecopy.end());
          freq[lcount-pcs]=std::atof(linecopy.c_str());
          tau[lcount-pcs]=std::atof(line.c_str());
        }
      }
  config.close();
  return 0;
}


LPEFilter CORRcon(const char* conffile,double *ftrans)
{//return LPEF filter and transient time from config file
  std::ifstream config;
  std::string coefline="    Coefs:";
  std::string line,cline,coefstring;
  std::string delcol=":";
  std::string dels=" ";
  double fs;//sampling frequency
  int coefcount=0;
  int d1,d2;//string delimeter positions
  int M;//filter length
  int TRL;//training length
  int TRP;//retraining length
  config.open(conffile);
  while(config.good())
    {
      getline(config,line);
      if(line=="sampling frequency")
        {
          getline(config,line);
          fs=std::atof(line.c_str());
        }
      if(line=="LPEF")
        {
          getline(config,line);
          getline(config,line);
          M=std::atoi(line.c_str());
        }
      if(line=="training length")
        {
          getline(config,line);
          TRL=std::atoi(line.c_str());
        }
      if(line=="re-training time")
        {
          getline(config,line);
          TRP=std::atoi(line.c_str());
        }
    }
  config.close();
  double* COEFS=new double[M];//array for LPEF coefficients
  config.open(conffile);
  while(config.good())
    {
      getline(config,line);
      if(line.substr(0,10)==coefline)
        {
          d1=line.find(delcol);
          d2=line.find(dels,d1+1);
          cline=line.substr(d2,line.size());
          int wcount=0;
          int s=0; int e=0;
          for(int i=0;i<cline.size();i++)
            {
              if(cline[i]==' ')
                {
                  wcount++;
                }
            }
          int* se=new int[wcount+1];
          wcount=0;
          for(int i=0;i<cline.size();i++)
            {
              if(cline[i]==' ')
                {
                  wcount++;
                  se[wcount-1]=i;
                }
            }
          se[wcount]=cline.size();
          for(int i=0;i<wcount;i++)
            {
              coefcount++;
              coefstring=cline.substr(se[i]+1,se[i+1]-1-se[i]);
              COEFS[coefcount-1]=atof(coefstring.c_str());
            }
        }
    }
  //config LPEF
  LPEFilter LPEF(M,TRP,TRL);
  LPEF.setRate(fs);
  LPEF.setCoefs(M,COEFS);
  *ftrans=LPEF.getTransientTime().GetSecs();
  config.close();
  delete[] COEFS;
  return LPEF;
}

void dumphelp(void) {//print help
  std::cout <<"useage: injtrig -injp\tinjection <file path>\n\t\t";
  std::cout <<"-sc\tinjection scale factor\n\t\t";
  std::cout <<"-xgwf\t<file path>\n\t\t";
  std::cout <<"-xchan\tgwf channel\n\t\t";
  std::cout <<"-xinj\tinjection channel\n\t\t";
  std::cout <<"-xconf\t<file path>\n\t\t";
  std::cout <<"-ygwf\t<file path>\n\t\t";
  std::cout <<"-ychan\tgwf channel\n\t\t";
  std::cout <<"-yinj\tinjection channel\n\t\t";
  std::cout <<"-yconf\t<file path>\n\t\t";
  std::cout <<"-segf\tscience segments file\n\t\t";
  std::cout <<"-sdur\tsigdur\n\t\t";
  std::cout <<"-thr\tthreshold\n\t\t";
  std::cout <<"-s\tstart\n\t\t-e\tend\n\t\t";
  std::cout <<"-o\ttrigger file output file path"<<std::endl;
  return;

}

int main(int argc, char** argv)
 {
  if (argc == 1) { dumphelp(); return 0; }
  std::string clineflag,clinedata,injpathstring,scalestring,xgwfstring,
    xchanstring,xinjchanstring,xconstring,ygwfstring,ychanstring,
    yinjchanstring,yconstring,sigdurstring,thresstring,segfile,startstring,
    endstring,trigfilestring;
  for(int ccount=1; ccount<argc; ++ccount) {
    clineflag = argv[ccount++];
    clinedata = argv[ccount];
    if ( clineflag=="-injp" ) {
      injpathstring=clinedata;
      std::cout << "injection file path set to " <<injpathstring<< std::endl;
    }
    if ( clineflag=="-sc" ) {
      scalestring=clinedata;
      std::cout << "injection scale set to " <<scalestring<< std::endl;
    }
    if ( clineflag=="-xgwf" ) {
      xgwfstring=clinedata;
      std::cout << "x file path set to " <<xgwfstring<< std::endl;
    }
    if ( clineflag=="-xchan" ) {
      xchanstring=clinedata;
      std::cout << "x channel set to " <<xchanstring<< std::endl;
    }
    if ( clineflag=="-xinj" ) {
      xinjchanstring=clinedata;
      std::cout << "x injection channel set to " <<xinjchanstring<< std::endl;
    }
    if ( clineflag=="-xconf" ) {
      xconstring=clinedata;
      std::cout << "x config file path set to " <<xconstring<< std::endl;
    }
    if ( clineflag=="-ygwf" ) {
      ygwfstring=clinedata;
      std::cout << "y file path set to " <<ygwfstring<< std::endl;
    }
    if ( clineflag=="-ychan" ) {
      ychanstring=clinedata;
      std::cout << "y channel set to " <<ychanstring<< std::endl;
    }
    if ( clineflag=="-yinj" ) {
      yinjchanstring=clinedata;
      std::cout << "y injection channel set to " <<yinjchanstring<< std::endl;
    }
    if ( clineflag=="-yconf" ) {
      yconstring=clinedata;
      std::cout << "y config file path set to " <<yconstring<< std::endl;
    }
    if ( clineflag=="-sdur" ) {
      sigdurstring=clinedata;
      std::cout <<"sigdur set to " <<sigdurstring<< std::endl;
    }
    if ( clineflag=="-thr" ) {
      thresstring=clinedata;
      std::cout <<"trigger on threshold " <<thresstring<< std::endl;
    }
    if ( clineflag=="-segf" ) {
      segfile=clinedata;
      std::cout << "science segments from " <<segfile<< std::endl;
    }
    if ( clineflag=="-s" ) {
      startstring=clinedata;
      std::cout << "trigger start >= " <<startstring<< std::endl;
    }
    if ( clineflag=="-e" ) {
      endstring=clinedata;
      std::cout << "trigger end <= " <<endstring<< std::endl;
    }
    if ( clineflag=="-o" ) {
      trigfilestring=clinedata;
      std::cout << "trigger output to " <<trigfilestring<< std::endl;
    }
  }

  std::ifstream input,tempin1,tempin2;
  std::ofstream Trigfile,test,test2;
  std::stringstream ss;
  std::string line,start,end,dur;
  std::string del="-"; // delimeter
  std::string dels=" ";
  std::string delr="\r";
  std::string del2=".";
  std::string del3=":";
  std::string deltab="\t";
```

```
std::string delfs="/";
int d1,d2,d3; //delimeter place variables
int append=0;//trigger file append flag
int Tcount=0;//initialise trigger count
double TRS,TRE;//trigger start/end variables
int oldTcount=0;
Time apptime;
int apps,appn;
input.open(trigfilestring.c_str());
if(input.is_open())
  {//permit restart of run by appending existing trigger files
    append=1;
    std::cout<<"file exists appending "<<std::endl;
   while(getline(input,line))
      {
        if(line[0]!='#')
          {
            oldTcount++;
            d1=line.find(deltab);
            d2=line.find(deltab,d1+1);
            d3=line.find(deltab,d2+1);
            startstring=line.substr(d2,d3-d2);
          }
      }
    TRE=atof(startstring.c_str());
    apps=(int)TRE;
    appn=(int)((TRE-(double)apps)*1e+9);
    std::cout<<std::fixed<<std::setprecision(9)<<TRE<<std::endl;
    std::cout<<apps<<"\t"<<appn<<std::endl;
    apptime.setS(apps); apptime.setN(appn);
  }
else{std::cout<<"new file"<<std::endl;}
input.close();
int trigs=atoi(startstring.c_str());
int trige=atoi(endstring.c_str());
if(append==0){Trigfile.open(trigfilestring.c_str());}
if(append==1)
  {
    Tcount=oldTcount;
    Trigfile.open(trigfilestring.c_str(),std::ios::out | std::ios::app);
  }

int GPSS,GPSE,DUR;
int injfcount=0;
int xfcount=0;
int yfcount=0;
int segcount=0;
int fs[4];//arrays for start/end seg/frames gps, 0-segs,1-inj,2-det1,3-det2
int fe[4];//for overlap test
TSeries xts,yts,xinjts,yinjts; //frame/injection time series
double xfs,yfs; //frame sampling frequencies
Time tS,tE; //TSeries time for cross-correlation time series
int maxlag=180; // max time lag in samples between detectors ~ 0.01s LH
double timeoff=(double)maxlag/16384.0; //correlation time offset
double wtime;
int sigdur=atoi(sigdurstring.c_str()); //signal sample length
double w =1-TMath::Exp(-1/(double)sigdur);
int recl=16384;// recording length after trigger in samples
CORRest Corr(maxlag,w,recl); //inialise correlator
int wincount;
double xhtrans,xltrans,xlpeftrans;//filter transients from config file
double yhtrans,yltrans,ylpeftrans;
double trans;//maximum of above transients
double xsig,ysig,xmean,ymean; //mean and sigma from config file
int xlin,ylin;//number of lines to be removed (config file)
double scale=atof(scalestring.c_str());
double scitime=0;//total time detector in "science" mode
double downtime=0;//total time not in science mode
double DF;//Duty factor DF=scitime/(scitime+downtime)
double time;//start time for CC loop
double SEC,trigtime,lag;//SEC:secs part of trigger time (trigtime),lag(s)
double filttime;//filter time
```

```
int segmin=100;//minimum segment length (s)
Dacc xIn,yIn,injIn;
//open segments file
input.open(segfile.c_str());
while(getline(input,line))
  {
    d1=line.find(dels);
    d2=line.find(delr,d1+1);
    start=line.substr(0,d1);
    end=line.substr(d1+1,line.size()-d1);
    GPSS=atoi(start.c_str());
    GPSE=atoi(end.c_str());
    if(((GPSS<=trigs && GPSE>trigs) ||
       (GPSS>trigs && GPSE<=trige) ||
        (GPSS<trige && GPSE>trige)) &&
       GPSE-GPSS>=segmin)
      {
        segcount++;
        if(segcount==1){fs[0]=GPSS;}
        fe[0]=GPSE;
      }
  }
input.close();

if(fs[0]>trigs){trigs=fs[0];}
if(fe[0]<trige){trige=fe[0];}
int* SEGS=new int[segcount];
int* SEGE=new int[segcount];
int count=0;
input.open(segfile.c_str());
while(getline(input,line))
  {//assign science segment array
    d1=line.find(dels);
    d2=line.find(delr,d1+1);
    start=line.substr(0,d1);
    end=line.substr(d1+1,line.size()-d1);
    GPSS=atoi(start.c_str());
    GPSE=atoi(end.c_str());
    if(((GPSS<=trigs && GPSE>trigs) ||
       (GPSS>trigs && GPSE<=trige) ||
        (GPSS<trige && GPSE>trige)) &&
       GPSE-GPSS>=segmin)
      {
        count++;
        SEGS[count-1]=GPSS;
        SEGE[count-1]=GPSE;
        if(count==1 && (GPSS<=trigs && GPSE>trigs) && GPSE-trigs>=segmin)
          {
            SEGS[0]=trigs;
          }
        if(count==segcount && (GPSS<trige && GPSE>trige)
           && trige-GPSS>=segmin)
          {
            SEGE[segcount-1]=trige;
          }
        scitime+=(double)(SEGE[count-1]-SEGS[count-1]);
        if(count>1 && count<segcount)
          {
            downtime+=(double)(SEGS[count-1]-SEGE[count-2]);
          }
      }
  }
DF=scitime/(scitime+downtime);
input.close();
int oldgpse;
//open injections
input.open(injpathstring.c_str());
while(getline(input,line))
  {//add paths of injection GWFs
    d1=line.find(del);
    d2=line.find(del,d1+1);
    d3=line.find(del,d2+1);
```

```
      d1=line.find(del,d3+1);
      d2=line.find(del,d1+1);
      d3=line.find(del,d2+1);
      d1=line.find(del,d3+1);
      d2=line.find(del2);
      start=line.substr(d3+1,d1-d3-1);
      dur=line.substr(d1+1,d2-d1-1);
      GPSS=atoi(start.c_str());
      DUR=atoi(dur.c_str());
      GPSE=GPSS+DUR;
      if((GPSS<=trigs && GPSE>trigs) ||
         (GPSS>trigs && GPSE<=trige) ||
         (GPSS<trige && GPSE>trige))
        {
          injfcount++;
          if(injfcount==1){fs[1]=GPSS;}
          injIn.addPath(line);
          fe[1]=GPSE;
          if(injfcount>1 && GPSS!=oldgpse)
            {//alert missing GWFs
              std::cout<<"Missing injections ! "<<oldgpse<<" to "<<GPSS<<"\t"
                       <<GPSS-oldgpse<<std::endl;
            }
          oldgpse=GPSE;
        }
    }
  input.close();
  //open det1 gwf
  input.open(xgwfstring.c_str());
  while(getline(input,line))
    {//add GWF paths for detector 1
      d1=1;//1st character is "/"
      int fspos=0;
      while(d1!=-1)
        {
          d1=line.find(delfs,d1+1);
          if(d1!=-1){fspos=d1;}
        }
      d2=line.find(del,fspos);
      d1=line.find(del,d2+1);
      d2=line.find(del,d1+1);
      d3=line.find(del2,d2+1);
      start=line.substr(d1+1,d2-d1-1);
      dur=line.substr(d2+1,d3-d2-1);
      GPSS=atoi(start.c_str());
      DUR=atoi(dur.c_str());
      GPSE=GPSS+DUR;
      if((GPSS<=trigs && GPSE>trigs) ||
         (GPSS>trigs && GPSE<=trige) ||
         (GPSS<trige && GPSE>trige))
        {
          xfcount++;
          if(xfcount==1){fs[2]=GPSS;}
          xIn.addPath(line);
          fe[2]=GPSE;
          if(xfcount>1 && GPSS!=oldgpse)
            {//alert missing GWFs
              std::cout<<"Missing H1 ! "<<oldgpse<<" to "<<GPSS<<"\t"
                       <<GPSS-oldgpse<<std::endl;
            }
          oldgpse=GPSE;
        }
    }
    input.close();
    std::cout<<"x frames "<<xfcount<<std::endl;

  //open det2 gwf
  input.open(ygwfstring.c_str());
  while(getline(input,line))
    {//add GWF paths for detector 2
      d1=1;//1st character is "/"
      int fspos=0;
      while(d1!=-1)
        {
          d1=line.find(delfs,d1+1);
          if(d1!=-1){fspos=d1;}
        }
      d2=line.find(del,fspos);
      d1=line.find(del,d2+1);
      d2=line.find(del,d1+1);
      d3=line.find(del2,d2+1);
      start=line.substr(d1+1,d2-d1-1);
      dur=line.substr(d2+1,d3-d2-1);
      GPSS=atoi(start.c_str());
      DUR=atoi(dur.c_str());
      GPSE=GPSS+DUR;
      if((GPSS<=trigs && GPSE>trigs) ||
         (GPSS>trigs && GPSE<=trige) ||
         (GPSS<trige && GPSE>trige))
        {
          yfcount++;
          if(yfcount==1){fs[3]=GPSS;}
          yIn.addPath(line);
          fe[3]=GPSE;
          if(yfcount>1 && GPSS!=oldgpse)
            {//alert missing GWFs
              std::cout<<"Missing L1 ! "<<oldgpse<<" to "<<GPSS<<"\t"
                       <<GPSS-oldgpse<<std::endl;
            }
          oldgpse=GPSE;
        }
    }
  input.close();
  std::cout<<"y frames "<<yfcount<<std::endl;
  //check frame time overlap
  if(injfcount==0 || xfcount==0 || yfcount==0 || segcount==0)
    {//abort run on zero science data overlap
      std::cout<<"Input frames/science do not overlap. Aborted.\n";
      return -1;
    }

  xIn.nextFrame();
  yIn.nextFrame();
  injIn.nextFrame();
  injIn.addChannel(xinjchanstring);//add det1 injection channel
  injIn.addChannel(yinjchanstring);//add det1 injection channel
  xIn.addChannel(xchanstring);//add det1 channel
  yIn.addChannel(ychanstring);//add det2 channel
  //get det1/2 configuration data standard deviation
  xsig=CORRcon(xconstring.c_str(),&xmean,&xlin,&xfs);
  ysig=CORRcon(yconstring.c_str(),&ymean,&ylin,&yfs);
  //construct filters
  std::string highpass="high pass filter";
  std::string lowpass="low pass filter";
  std::string LPEF="LPEF";
  IIRFilter xhigh=CORRcon(xconstring.c_str(),&xhtrans,highpass);
  IIRFilter xlow=CORRcon(xconstring.c_str(),&xltrans,lowpass);
  IIRFilter yhigh=CORRcon(yconstring.c_str(),&yhtrans,highpass);
  IIRFilter ylow=CORRcon(yconstring.c_str(),&yltrans,lowpass);
  IIRFilter xhpcopy,yhpcopy;//copies of hp filters
  xhpcopy=xhigh; yhpcopy=yhigh;
  PIIR xpfilt[xlin];
  PIIR ypfilt[ylin];
  LPEFilter xLPEF=CORRcon(xconstring.c_str(),&xlpeftrans);
  LPEFilter yLPEF=CORRcon(yconstring.c_str(),&ylpeftrans);
  xLPEF.fm_zero_phase;
  yLPEF.fm_zero_phase;
  int xbins=xLPEF.getTrainLength();
  std::cout<<"LPEF training length "<<xbins<<std::endl;
  int LPEFLEN=xLPEF.getLength();
  std::cout<<"LPEF filter length "<<LPEFLEN<<std::endl;
  wtime=(double)xbins/16384.0;
  Interval frint(wtime);//interval of frame data
  double* xdata=new double[xbins];
```

```
double* ydata=new double[xbins];
double* xfiltereddata=new double[xbins];//det1 array for iwave filtered data
double* yfiltereddata=new double[xbins];//det2 array for iwave filtered data
double* xfv=new double[xlin];//det 1 line freq array for iwave
double* xtau=new double[xlin];//det 1 tau array
double* yfv=new double[ylin];//det 2 line freq array for iwave
double* ytau=new double[ylin];//det 2 tau array
double** xfm=new double*[xlin];
double** yfm=new double*[ylin];
CORRcon(xconstring.c_str(),xfv,xtau);//populate arrays
CORRcon(yconstring.c_str(),yfv,ytau);
double* nx=new double[xbins];
double* ny=new double[xbins];
double TRANS[6]={xhtrans,xltrans,xlpeftrans,yhtrans,yltrans,ylpeftrans};
trans=TMath::MaxElement(6,TRANS);//obtain max transient of filters
Interval textract(trans);//interval of filter transient
//construct piir filters
double* xfstate=new double[xpfilt[0].getStatesize()];
double* yfstate=new double[ypfilt[0].getStatesize()];
//initialise IWave
for(int i=0;i<xlin;i++)
  {
    xfm[i]=xfstate;
    xpfilt[i]=PIIR(xfs,xtau[i],xfv[i],xfm[i]);
  }
for(int i=0;i<ylin;i++)
  {
    yfm[i]=yfstate;
    ypfilt[i]=PIIR(xfs,ytau[i],yfv[i],yfm[i]);
  }

std::cout<<"maximum filter transient "<<trans<<std::endl;
double M=(double)sigdur*log(1000);
double radic=w*(1-pow(1-w,2*M))/(2-w+w*w);
double sigmachar=sqrt(radic)*xsig*ysig;//expected sigma of RTCC output
double sigmult=atof(thresstring.c_str());//threshold value multiplier
double thresh=sigmult*sigmachar;//trigger threshold
std::cout<<std::scientific<<thresh<<std::endl;
double x,y;
int trigstart,trigend;
int s,ns,yc;
Interval ext;
double trigskip=1.5*xlpeftrans;//min time between triggers
if(append==0){//output header to trigger file
  Trigfile<<"#\t mdc injection trigger input info"<<std::endl;
  Trigfile<<"# triggering from science segments :"<<segfile<< std::endl;
  Trigfile<<"# gwf input det1 :"<<xgwfstring<<std::endl;
  Trigfile<<"# channel det1 :"<<xchanstring<<std::endl;
  Trigfile<<"# config file det1 :"<<xconstring<<std::endl;
  Trigfile<<"# gwf input det2 :"<<ygwfstring<<std::endl;
  Trigfile<<"# channel det2 :"<<ychanstring<<std::endl;
  Trigfile<<"# config file det2 :"<<yconstring<<std::endl;
  Trigfile<<"# injection file " <<injpathstring<< std::endl;
  Trigfile<<"#\t mdc trigger parameter info"<<std::endl;
  Trigfile<<"# injection scale :"<<scale<<std::endl;
  Trigfile<<"# maxlag (samples):"<<maxlag<<std::endl;
  Trigfile<<"# sigdur (samples):"<<sigdur<<std::endl;
  Trigfile<<"# minimum time between triggers (s) :"<<trigskip<<std::endl;
  Trigfile<<"# correlation sigma :"<<sigmachar<<std::endl;
  Trigfile<<"# trigger threshold :"<<thresh<<std::endl;
  Trigfile<<"# Science Duty factor :"<<DF<<std::endl;
  Trigfile<<"# Minimum segment length (s) :"<<segmin<<std::endl;}

int correc,reccount;//correc=1 record trigger reccount-samples recorded
double cormax,cormaxmem,cormaxlag,cormaxlagmem,cormaxmemtime;
double* xREC;//array for det1 data upon trigger
double* yREC;//array for det2 data upon trigger
double oldxpow,oldypow;//trigger power variables
double xpowdelta,ypowdelta;
int tscount=0;
int init=0;//flag trigger file inilialised
int cor; //flag for CC
int lpeftrain,lpeftrainwait,lpeftraincancel,getsigma;
int lpefint=40;//wincount%lpefint sets train wait
double xsigma,ysigma;//sigma of filtered signals
double frtime,exttime;
for(int i=0;i<segcount;i++)
  {//loop through science segments
    std::cout<<"segment "<<SEGS[i]<<"\t"<<SEGE[i]<<std::endl;
    cor=1;
    Time triggerstart(SEGS[i],0);
    if(append==1)
      {
        triggerstart=apptime;
        append=0;
      }
    else{TRE=(double)SEGS[i];}//look at this for other error

    xIn.seek(triggerstart);//locate start time in GWF data
    yIn.seek(triggerstart);
    injIn.seek(triggerstart);
    xhigh.reset(); xlow.reset(); xLPEF.reset();
    yhigh.reset(); ylow.reset(); yLPEF.reset();
    xhpcopy.reset(); yhpcopy.reset();
    wincount=0;
    correc=0;
    while(xIn.synch()==0 && yIn.synch()==0 && cor==1 && frtime<SEGE[i])
      {
        wincount++;
        lpeftraincancel=0;
        if(wincount%lpefint==1){lpeftrainwait=1;}
        ext=frint;
        frtime=xIn.getCurrentTime().totalS();
        exttime=frtime+frint.GetSecs();
        if(exttime>SEGE[i])//check when data exceeding segment end
          {
            double diffext=exttime-(double)SEGE[i];
            ext-=(diffext);
            cor=0;
            xhpcopy.reset(); yhpcopy.reset();
            xlow.reset(); ylow.reset();
            lpeftraincancel=1;
          }

        xIn.fillData(ext);
        yIn.fillData(ext);
        injIn.fillData(ext);
        xts = *(xIn.refData(xchanstring));
        xinjts=*(injIn.refData(xinjchanstring));
        xinjts*=(scale);
        yts = *(yIn.refData(ychanstring));
        xbins=xts.getNSample();
        yinjts=*(injIn.refData(yinjchanstring));
        yinjts*=(scale);//scale injection
        if(xinjts.getMaximum()==0 && //determine if injection zero
           yinjts.getMaximum()==0)  //so LPEF can train
          {lpeftrain=1;}
        else{lpeftrain=0;}
        xts+=(xinjts);//inject
        yts+=(yinjts);
        //filter
        TSeries xTSHP(xhigh.apply(xts));//apply high pass
        TSeries yTSHP(yhigh.apply(yts));
        xTSHP.getData(xbins,xdata);
        yTSHP.getData(xbins,ydata);
        for(int j=0;j<xlin;j++)
          {//IWave line filter det1 data
            xpfilt[j].filterline(xdata,xfiltereddata,xbins,xfm[j]);
            xdata=xfiltereddata;
          }
        for(int j=0;j<ylin;j++)
          {//IWave line filter det2 data
            ypfilt[j].filterline(ydata,yfiltereddata,xbins,yfm[j]);
            ydata=yfiltereddata;
          }
        //cast data arrays to TSeries objects
        TSeries xLR(xTSHP.getStartTime(),xTSHP.getTStep()
                     ,xbins,xfiltereddata);
        TSeries yLR(yTSHP.getStartTime(),yTSHP.getTStep()
                     ,xbins,yfiltereddata);
        TSeries xTSLPEF,yTSLPEF;
        if(lpeftrainwait==1 && lpeftrain==1 && lpeftraincancel==0)
          {//train LPEF filter on condition
            xLPEF.train(xLR); yLPEF.train(yLR);
            lpeftrainwait=0;
            getsigma=1;
          }
        xLPEF.apply(xLR,xTSLPEF);//apply LPEF
        yLPEF.apply(yLR,yTSLPEF);
        TSeries xHP2(xhpcopy.apply(xTSLPEF));//apply hp again
        TSeries yHP2(yhpcopy.apply(yTSLPEF));
        TSeries xCor(xlow.apply(xHP2));//apply final low pass
        TSeries yCor(ylow.apply(yHP2));
        //
        if(wincount==1)
          {//subtract maximum filter transient
            Time xt0=xCor.getStartTime();
            xt0+=(textract);
            xCor=xCor.extract(xt0);
            yCor=yCor.extract(xt0);
            trigstart=xCor.getStartTime().totalS();
            if(Tcount==0 && init==0)
              {//output data header line and trigger start
                Trigfile<<"# trigger start :"<<trigstart<<std::endl;
                Trigfile<<"#n\tstart\t\t\tend\t\t\tdet1 max\t\t";
                Trigfile<<"det2 max\t\tlag\t\tnormcorln\t";
                Trigfile<<"det1 freq\tdet2 freq\tdet1*det2 freq\t";
                Trigfile<<"det1 sigma\tdet1 trig sigma\tdet2 sigma\t";
                Trigfile<<"det2 trig sigma\tdet1 SNR\tdet2 SNR\n";
                init=1;
              }
          }
        tS=xCor.getStartTime();
        tE=xCor.getEndTime();
        xCor.getData(xbins,nx);
        yCor.getData(xbins,ny);
        if(getsigma==1 || wincount==1)
          {//get sigma of det1/2 data
            xsigma=TMath::RMS(xbins,nx);
            ysigma=TMath::RMS(xbins,ny);
            getsigma=0;
          }
        for(int xc=0;xc<xbins;xc++)
          {//loop through data
            SEC=xc/xfs;
            time=tS.totalS()+SEC;//-timeoff;
            if (time>=SEGS[i] && time<=SEGE[i])
              {
                x=nx[xc];
                y=ny[xc];
                Corr.apply(x,y);//apply RTCC
                cormax=Corr.getMax();//get max value
                if(cormax>=thresh && correc==0 && ((time-TRE)>=trigskip))
                  {//initiate trigger
                    correc=1;
                    Corr.setRec(correc);
                    tscount=0;
                    reccount=0;
                  }
                if(correc==1)
                  {
                    reccount++;
                    if(reccount==1)
                      {
                        Tcount++;
                        cormaxmem=0;
                        TRS=time-timeoff;
                      }
                    if(cormax>cormaxmem)
                      {
                        cormaxmem=cormax;
                        cormaxlagmem=Corr.getLag();
                        cormaxmemtime=time;
                      }
                    //power ratio
                    xpowdelta=oldxpow/Corr.getXpower();
                    ypowdelta=oldypow/Corr.getYpower();
                    oldxpow=Corr.getXpower();
                    oldypow=Corr.getYpower();
                    if(xpowdelta>0.999 && ypowdelta>0.999)
                      {tscount++;}
                    else{tscount=0;}
                  }
                if((correc==1 &&
                   (cormax<thresh) &&
                    reccount>maxlag && (tscount>3*maxlag)) || reccount==recl)
                  {//end trigger
                    TRE=time-timeoff;
                    double flow=0;
                    double fhigh=3000;
                    double fstep=1;
                    double xfreqest,yfreqest,xamp,yamp,xyamp;
                    double xsamp,ysamp,xcamp,ycamp,frq,t;
                    double xampmax=0;
                    double yampmax=0;
                    double xyampmax=0;
                    double xyfreqest;
                    int Nf=(int)((fhigh-flow+1)/fstep);
                    xREC=Corr.getXRec();//return det1 data record
                    yREC=Corr.getYRec();//return det1 data record
                    double xx,yy,xmm,ymm,phi;
                    double pcut=0.8;
                    double xprob,yprob;
                    double xexp,yexp;
                    int xpos,ypos;
                    if(cormaxlagmem==0){xpos=0; ypos=0;}
                    if(cormaxlagmem<0){xpos=0; ypos=-cormaxlagmem;}
                    if(cormaxlagmem>0){xpos=cormaxlagmem; ypos=0;}
                    double xmean=TMath::Mean(reccount,xREC);
                    double ymean=TMath::Mean(reccount,yREC);
                    double xssigma=TMath::RMS(reccount,xREC);//signal sigma
                    double yssigma=TMath::RMS(reccount,yREC);
                    double normcor=0;
                    double xmax=TMath::MaxElement(reccount,xREC);
                    double xmin=TMath::MinElement(reccount,xREC);
                    double ymax=TMath::MaxElement(reccount,yREC);
                    double ymin=TMath::MinElement(reccount,yREC);
                    if(fabs(xmin)>xmax){xmax=fabs(xmin);}
                    if(fabs(ymin)>ymax){ymax=fabs(ymin);}
                    double* xpdata=new double[reccount];
                    double* ypdata=new double[reccount];
                    double xSNR=(xssigma*xssigma-xsigma*xsigma)/
                      (xsigma*xsigma);
                    double ySNR=(yssigma*yssigma-ysigma*ysigma)/
                      (ysigma*ysigma);
                    double xpsigma=xsigma;//prob sigma
                    double ypsigma=ysigma;
                    if(xSNR<1){xpsigma=xssigma;}//invert sigma if SNR<1
                    if(ySNR<1){ypsigma=yssigma;}
                    double xmaxexp=-(xmax-xmean)*(xmax-xmean)/
                      (2*xpsigma*xpsigma);
                    double ymaxexp=-(ymax-ymean)*(ymax-ymean)/
                      (2*ypsigma*ypsigma);
                    double xmaxprob=(1-TMath::Exp(xmaxexp));
                    double ymaxprob=(1-TMath::Exp(ymaxexp));
                    // norm corln & prob
                    int xmaxpos,ymaxpos;
                    for(int h=0;h<reccount;h++)
                      {//operate on lag aligned data
                        if(h+xpos<reccount && h+ypos<reccount)
                          {
                            xx=xREC[h+xpos];
                            yy=yREC[h+ypos];
                            xmm=xx-xmean;
                            ymm=yy-ymean;
                            normcor+=xmm*ymm;
                            if(fabs(xx)==xmax)
                              {xmax=fabs(xx); xmaxpos=h+xpos;}
                            if(fabs(yy)==ymax)
                              {ymax=fabs(yy); ymaxpos=h+ypos;}
                            //Gaussian filter
                            xexp=-xmm*xmm/(2*xpsigma*xpsigma);
                            yexp=-ymm*ymm/(2*ypsigma*ypsigma);
                            xprob=(1-TMath::Exp(xexp))/xmaxprob;
                            yprob=(1-TMath::Exp(yexp))/ymaxprob;
                            if(xprob<pcut){xprob=0;}
                            if(yprob<pcut){yprob=0;}
                            xpdata[h]=xprob*xx;
                            ypdata[h]=yprob*yy;
                          }
                      }
                    normcor*=1/(xssigma*yssigma*double(reccount));
                    normcor=fabs(normcor);//normalised CC value
                    //freq estimate
                    for(int j=0;j<Nf;j++)
                      {
                        frq=flow+j*fstep;
                        xsamp=0; ysamp=0; xcamp=0; ycamp=0;
                        for(int h=0;h<reccount-1;h++)
                          {
                            if(h+xpos<reccount && h+ypos<reccount)
                              {
                                t=(1/16384.0)*h;
                                phi=2*pi*frq*t;
                                if(j==0)
                                  {//spike filter
                                    if(xpdata[h-1]==0 && xpdata[h]!=0
                                       && xpdata[h+1]==0)
                                      {xpdata[h]=0;}
                                    if(xpdata[h-1]==0 && xpdata[h]!=0 &&
                                       xpdata[h+1]!=0 &&  xpdata[h+2]==0)
                                      {xpdata[h]=0; xpdata[h+1]=0;}
                                    if(xpdata[h-1]==0 && xpdata[h]!=0 &&
                                        xpdata[h+1]!=0 &&  xpdata[h+2]!=0 &&
                                       xpdata[h+3]==0 && xpdata[h+4]==0)
                                      {xpdata[h]=0; xpdata[h+1]=0;
                                        xpdata[h+2]=0;}
                                    if(ypdata[h-1]==0 && ypdata[h]!=0
                                       && ypdata[h+1]==0)
                                      {ypdata[h]=0;}
                                    if(ypdata[h-1]==0 && ypdata[h]!=0 &&
                                       ypdata[h+1]!=0 &&  ypdata[h+2]==0)
                                      {ypdata[h]=0; ypdata[h+1]=0;}
                                    if(ypdata[h-1]==0 && ypdata[h]!=0 &&
                                        ypdata[h+1]!=0 &&  ypdata[h+2]!=0 &&
                                       ypdata[h+3]==0 && ypdata[h+4]==0)
                                      {ypdata[h]=0; ypdata[h+1]=0;
                                        ypdata[h+2]=0;}
                                  }
                                if(xpdata[h]!=0)
                                  {
                                    xsamp+=xpdata[h]*TMath::Sin(phi);
                                    xcamp+=xpdata[h]*TMath::Cos(phi);
                                  }
                                if(ypdata[h]!=0)
                                  {
                                    ysamp+=ypdata[h]*TMath::Sin(phi);
                                    ycamp+=ypdata[h]*TMath::Cos(phi);
                                  }
                              }
```

```cpp
                           }
                         xamp=xsamp*xsamp+xcamp*xcamp;
                         yamp=ysamp*ysamp+ycamp*ycamp;
                         xyamp=xamp*yamp;
                         if(xamp>xampmax){xampmax=xamp; xfreqest=frq;}
                         if(yamp>yampmax){yampmax=yamp; yfreqest=frq;}
                         if(xyamp>xyampmax){xyampmax=xyamp; xyfreqest=frq;}
                       }
                     //
                     double xmaxtime=TRS+(-(2*maxlag+1)+xmaxpos)/16384.0;
                     double ymaxtime=TRS+(-(2*maxlag+1)+ymaxpos)/16384.0;
                     // output to trigger file
                     Trigfile<<Tcount<<"\t"
                             <<std::fixed
                             <<std::setprecision(9)
                             <<TRS<<"\t"//trigger start
                             <<TRE<<"\t"//trigger end
                             <<xmaxtime<<"\t"//det1 max
                             <<ymaxtime<<"\t"//det 2 max
                             <<std::fixed
                             <<cormaxlagmem<<"\t"//lag
                             <<normcor<<"\t"//normalised CC
                             <<xfreqest<<"\t"//det1 freq
                             <<yfreqest<<"\t"//det2 freq
                             <<xyfreqest<<"\t"//freq prod
                             <<std::scientific
                             <<xsigma<<"\t"//det1 sigma
                             <<xssigma<<"\t"//det1 trig sigma
                             <<ysigma<<"\t"//det2 sigma
                             <<yssigma<<"\t"//det2 trig sigma
                             <<xSNR<<"\t"//det1 trig SNR
                             <<ySNR<<"\t"//det2 trig SNR
                             <<std::endl;
                     std::cout<<Tcount<<"\t"
                              <<std::fixed
                              <<std::setprecision(9)
                              <<TRS<<"\t"
                              <<TRE<<"\t"
                              <<xmaxtime<<"\t"
                              <<ymaxtime<<"\t"
                              <<std::fixed
                              <<cormaxlagmem<<"\t"
                              <<normcor<<"\t"
                              <<std::fixed
                              <<xfreqest<<"\t"
                              <<yfreqest<<"\t"
                              <<xSNR<<"\t"
                              <<ySNR<<"\t"
                              <<std::endl;
                     std::cout<<xmaxpos<<"\t"<<ymaxpos<<std::endl;
                     correc=0;
                     Corr.setRec(correc);
                     reccount=0;
                     cormaxmem=0;
                     cormaxlagmem=0;
                   }
               }
           }
       }
   }
 input.close();
 Trigfile.close();
 return 0;
}
```

Bibliography

[1] C. Grupen, Astroparticle Physics, Springer (2006), ISBN -10 3-540-25312-2.

[2] I. Newton, Sir Isaac Newton's Mathematical Principles of Natural Philosophy and his System of the World, English translation from Latin by A. Motte (1729), Revised Translation by Florian Cajori, University of California Press (1934), ISBN 0-520-00928-2.

[3] A. Einstein, Nherungsweise Integration der Feldgleichungen der Gravitation. Sitzungsberichte der Kniglich Preussischen Akademie der Wissenschaften Berlin (1916), 688696.

[4] R. V. Pound and J. L. Snider, Effect of Gravity on Gamma Radiation, Phy. Rev., Vol 140, Number 3B, 8 November 1965.

[5] R. D. Blandford and R. Narayan, Cosmological Applications of Gravitational Lensing, Annual review of astronomy and astrophysics. **30**:311-358 (1992).

[6] I. I. Shapiro, Phys. Rev. Lett. 13, 789 (1964).

[7] J. M. Weisberg, J. H. Taylor and L. A. Fowler, Gravitational Waves from an Orbiting Pulsar, Scientific American **245**, 74-82 (1981).

[8] J. M. Weisberg and J. H. Taylor, Relativistic Binary Pulsar B1913+16: Thirty Years of Observations and Analysis, arXiv:astro-ph/0407149v1 (2004).

[9] A. Einstein, ber Gravitationswellen. Sitzungsberichte der Kniglich Preussischen Akademie der Wissenschaften Berlin (1918), 154167.

[10] K. S. Thorne, Gravitational Waves, Particle and Nuclear Astrophysics and Cosmology in the Next Millennium (Eds E. Kolb and R. D. Peccei) p 160, arXiv:gr-qc/9506086 v1 30 Jun 95.

[11] C. W. Misner K. S. Thorne & J. A. Wheeler, Gravitation, W. H. Freeman and Company (1973), ISBN 0-7167-0344-0.

[12] B. F. Schutz, A First Course in General Relativity, Cambridge University Press (2009), ISBN-13 978-0-521-88705-2.

[13] L. Ju et al. Detection of gravitational waves, Rep. Prog. Phys.**63** (2000) 1317-1427.

[14] J. D. E. Creighton and W. G. Anderson, Gravitational-Wave Physics and Astronomy, Wiley-VCH, Weinheim (2011).

[15] T. W. Baumgarte and S. L. Shapiro, Binary Black Hole Mergers, Physics Today Vol 64 10 (2011) p 32.

[16] D. Kennefick, Travelling at the Speed of Thought: Einstein and the Quest for Gravitational Waves:Princeton University Press (2007), chap 5 Do Gravitational Waves Exist?

[17] C. M. DeWitt and D. Rickles (editors), The Role of Gravitation in Physics, Report from the 1957 Chapel Hill Conference Edition Open Access (2011) ISBN 978-3-86931-963-6.

[18] J. Weber, Evidence for Discovery of Gravitational Radiation, Phys.Rev.Lett 22, 1320 16 June 1969.

[19] H. Collins, Gravitys Shadow: The Search for Gravitational Waves, The University of Chicago Press, Chicago, ISBN: 0-226-11377-9.

[20] J. P. Zendri et al. Status report of the gravitational wave detector AURIGA, in Gravitational Waves and Experimental gravitation (2003), proceeding of the XXXVIIIth Moriond Workshop, Dumarchez and J. Tran Thanh Van, The Gioi Publishers, Vietnam, 37-42.

[21] M. E. Gerstenstein and V. I. Pustovoit, On the Detection of Low Frequency Gravitational Waves, (1963) Soviet Physics-JETP 16:43335 (Orig. pub. in Russian 1962.)

[22] G. E. Moss, R. L. Miller and R. L. Forward, Photon-Noise-Limited Laser Transducer for Gravitational Antenna, Applied Optics (1971) **10**, 24952498.

[23] R. Weiss, Electromagnetically Coupled Broadband Gravitational Antenna, Quarterly Progress Report, Research Laboratory of Electronics (1972), MIT **105**.

[24] R. W. Drever et al., Proc. Int. Mtg on Experimental Gravitation, Pavia 1977 (1979) 365.

[25] W. Winkler, Proc. Int. Mtg on Experimental Gravitation, Pavia 1977 (1979) 351.

[26] P. S. Linsay, P. R. Saulson and R. Weiss, A Study of a Long Baseline Gravitational Wave Antenna System (MIT, Cambridge, Mass., 1983), unpublished.

[27] R. W. P. Drever, F. J. Raab, K. S. Thorne, R. Vogt, and R. Weiss, A Laser Interferometer Gravitational-wave Observatory (LIGO) Technical Report, 1989,https://dcc.ligo.org/LIGOM890001/public/main.

[28] Committee on Setting Priorities for NSF-Sponsored Large Research Facility Projects et al., SETTING PRIORITIES for LARGE RESEARCH FACILITY PROJECTS supported by the NATIONAL SCIENCE FOUNDATION, THE NATIONAL ACADEMIES PRESS (2004) pp. 109 -117 ISBN 0-309-09084-9.

[29] T. Accadia et al, Status and perspectives of the Virgo gravitational wave detector, Journal of Physics: Conference Series, Volume 203, Number 1 (2010).

[30] B. Willke et al., The GEO 600 gravitational wave detector, Class. Quantum Grav. 19 (2002) 13771387.

[31] G. Heinzel on behalf of the TAMA team, Status and first results of TAMA300, Class. Quantum Grav. **18** (2001) 41134119.

[32] R. T. Cahill and K. Kitto, Michelson-Morley Experiments Revisited and the Cosmic Background Radiation Preferred Frame, arXiv:physics/0205065v1 23 May 2002.

[33] A. Freise, Interferometer Techniques for Gravitational-Wave Detection, Living Reviews in Relativity, **13** (2010), 1.

[34] M. Maggiore, Gravitational Waves Volume 1:Theory and Experiments, Oxford University Press (2011), ISBN 978-0-19-857074-5.

[35] J. Dietrich, Realizing LIGO, Engineering and Science, 61 (2). pp. 8-17. ISSN 0013-7812 1998.

[36] I. S. Gradshteyn and I. M. Ryzhik, Tables of Integrals, Series, and Products, Academic Press, San Diego; London, 1994, 5th Ed, equation 8.511 4.

[37] R. W. P. Drever et al., Laser Phase and Frequency Stabilization Using an Optical Resonator', Appl. Phys. B 31 (1983) 97 – 105.

[38] A. C. Searle et al. Spectral Line Removal in the LIGO Data Analysis System (LDAS), arXiv:gr-qc/0308024v1, Class.Quant.Grav. 20 S721-S730 2003.

[39] E. Thrane, N. Christensen and R. M. S. Schofield, Correlated magnetic noise in global networks of gravitational-wave interferometers:observations and implications (2013) arXiv:1303.2613 [astro-ph.IM].

[40] E. J. Daw et al. Long term study of the seismic environment at LIGO, arXiv:gr-qc/0403046v1 11 Mar 2004.

[41] D. Sigg, Gravitational Waves, LIGO-P980007-00-D (1998).

[42] J. R. Smith for the LIGO Scientific Collaboration, The Path to the Enhanced and Advanced LIGO Gravitational-Wave Detectors, arXiv:0902.0381v2 [gr-qc].

[43] J. Aasi et al. The LIGO Scientific Collaboration, Characterization of the LIGO detectors during their sixth science run, 18 Nov 2014, arXiv:1410.7764v2 [gr-qc].

[44] B. P. Abbott et al., LIGO: the Laser Interferometer Gravitational-Wave Observatory, The LIGO Scientific Collaboration, Rep. Prog. Phys. **72** (2009) 076901 (25pp).

[45] The LIGO Scientific Collaboration, Advanced LIGO, arXiv:1411.4547v1 [gr-qc] (2014).

[46] F. Acernese et al. Advanced Virgo: a 2nd generation interferometric gravitational wave detector, arXiv:1408.3978v3 [gr-qc] (2014).

[47] K. L. Dooley et al. GEO 600 and the GEO-HF upgrade program: successes and challenges, arXiv:1510.00317v1 [physics.ins-det] (2015).

[48] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery Numerical Recipes: the art of scientific computing, Cambridge University Press (2007) ISBN 978-0-521-88068-8.

[49] K. F. Riley M. P. Hobson and S. J. Bence, Mathematical Methods for Physics and Engineering (3^{rd} Edition, 2006), Cambridge University Press, ISBN-13 978-0-521-86153-3.

[50] C. E. Shannon, Communication in the Presence of Noise, Proc. Institute of Radio Engineers (1949). **37** (1): 1021.

[51] D. W. Jordan and P. Smith, Mathematical Techniques (4^{th} Edition, 2008), Oxford University Press, ISBN 978-0-19-928201-2.

[52] S. V. Vaseghi, Advanced Digital Signal Processing and Noise Reduction (2008) ISBN 978-0-470-75406-1.

[53] S. Chatterji et al., Multiresolution techniques for the detection of gravitational-wave bursts, arXiv:gr-qc/0412119v1, Class.Quant.Grav. 21 S1809-S1818 2004.

[54] R. Balasubramanian et al., Results from the first burst hardware injections performed on GEO 600, Class. Quantum Grav. **22** (2005) 30153028.

[55] G. Vajente, 12th ILIAS WG1 meeting, Geneva, March 29th -30th 2007. www.ego-gw.it/ilias-gw/WP1docs/vajente_290307.ppt.

[56] The LIGO Scientific Collaboration, Tuning matched filter searches for compact binary coalescence, LIGO-T070109-01-Z 2007/09/15.

[57] B. Allen, A chi-squared time-frequency discriminator for gravitational wave detection, Phys.Rev.D71:062001 (2005), arXiv:gr-qc/0405045v2.

[58] T. Pradier et al., An efficient filter for detecting gravitational wave bursts in interferometric detectors, Phys. Rev. D 63, 042002 (2001), arXiv:gr-qc/0010037v1.

[59] M. Was et al. ,On the background estimation by time slides in a network of gravitational wave detectors, Class. Quantum Grav., **27**:015005, 2010.

[60] P. C. Gregory, Bayesian Logical Data Analysis for the Physical Sciences: a comparative approach with Mathematica support (2010) Cambridge University Press ISBN 978-0-521-15012-5.

[61] M. K. Simon, Probability Distributions Involving Gaussian Random Variables, Springer 2002 ISBN: 978-1-4020-7058-7.

[62] K. Blackburn et al. The LIGO Scientific Collaboration, Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors (IGWD), LIGO-T970130-v1 (2009).

[63] L. Cadonati, Data Quality Vetoes in LIGO S5 Searches for Gravitational Wave Transients, LIGO-G060628-00-Z.

[64] M. Vallisneri et al. The LIGO Open Science Center (2015) arXiv:1410.4839v2 [gr-qc].

[65] A. L. Stuver and L. S. Finn, GravEn: Software for the simulation of gravitational wave detector network response, Class.Quant.Grav. 23 (2006) S799-S807.

[66] A. Weinstein et al. Hardware Burst Injections in Pre-S2→S2 (LSC Meeting March 17th-20th), Livingston, LA, LIGO-G030081.

[67] K. Riles, Notes on Relating Sine-Gaussian Waveforms to Astrophysical Source Strengths, (2004) LIGO-T040055-00-Z.

[68] http://labcit.ligo.caltech.edu/~jzweizig/dmt/DMTProject/
Last accessed 2nd Oct 2015.

[69] R. Brun and F. Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. Meth. in Phys. Res. A 389 (1997) 81-86. See also root.cern.ch/.

[70] S. Chatterji, Data Conditioning for Gravitational Wave Burst Analysis, LIGO-G030439-00-Z.

[71] E. Daw, Complex-Tap IIR Filters for the Dynamic Characterisation of Sinusoids, LIGO P1500210-v1.

[72] H. Goldstein, C. Poole and J. Safko, Classical Mechanics 3$^{\rm rd}$ Edition ISBN-13:978-0201657029.

[73] A. Zehtabian and H. Hassanpour, A Non-destructive Approach for Noise Reduction in Time Domain, World Applied Sciences Journal 6 (1): 53-63, (2009).

[74] E. D. Black and R. N. Gutenkunst, An Introduction to Signal Extraction in Interferometric Gravitational-Wave Detectors, Am. J. Phys., Vol. 71, No. 4, April 2003.

[75] S. A. Hughes, Listening to the Universe with Gravitational-Wave Astronomy arXiv:astro-ph/0210481v3 Annals of Physics, Volume 303, Issue 1, p. 142-178 2003.

[76] B. Schutz, Gravity from the ground up, Cambridge University Press (2003), ISBN-13 978-0-521-45506-0.

[77] W. East, and B. Lantz, Notch Filter Design, August 29, 2005, LIGO-T050162-00-R.

[78] Abadie et al. (The LIGO Scientific Collaboration, The Virgo Collaboration), All-sky search for gravitational-wave bursts in the second joint LIGO-Virgo run, J. Phys. Rev. D 85, 122007 (20 June 2012), arXiv:1202.2788v3.

[79] S. Klimenko et al., Coherent method for detection of gravitational wave bursts,Class. Quantum Grav. 25, 114029 (2008), arXiv:0802.3232v2.