The Astrophysical Journal, 952:25 (23pp), 2023 July 20 https://doi.org/10.3847/1538-4357/acd775

pygwb: A Python-based Library for Gravitational-wave Background Searches

Arianna I. Renzini[1,2], Alba Romero-Rodríguez[3], Colm Talbot[4], Max Lalleman[5], Shivaraj Kandhasamy[6], Kevin Turbang[3,5], Sylvia Biscoveanu[4,7], Katarina Martinovic[8], Patrick Meyers[9], Leo Tsukada[10,11], Kamiel Janssens[5,12], Derek Davis[1,2], Andrew Matas[13], Philip Charlton[14], Guo-Chin Liu[15], Irina Dvorkin[16,17], Sharan Banagiri[18], Sukanta Bose[6], Thomas Callister[19], Federico De Lillo[20], Luca D'Onofrio[21,22], Fabio Garufi[21,22], Gregg Harry[23], Jessica Lawrence[24], Vuk Mandic[25], Adrian Macquet[26], Ioannis Michaloliakos[27], Sanjit Mitra[6], Kiet Pham[25], Rosa Poggiani[28,29], Tania Regimbau[30], Joseph D. Romano[24], Nick van Remortel[5], and Haowen Zhong[25]
[1] LIGO Laboratory, California Institute of Technology, Pasadena, CA 91125, USA; arenzini@caltech.edu
[2] Department of Physics, California Institute of Technology, Pasadena, CA 91125, USA
[3] Theoretische Natuurkunde, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
[4] Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[5] Universiteit Antwerpen, Prinsstraat 13, B-2000 Antwerpen, Belgium
[6] Inter-University Centre for Astronomy and Astrophysics, Pune 411007, India
[7] LIGO Laboratory, Massachusetts Institute of Technology, 185 Albany St, Cambridge, MA 02139, USA
[8] Theoretical Particle Physics and Cosmology Group, Physics Department, King's College London, University of London, Strand, London, WC2R 2LS, UK
[9] Theoretical Astrophysics Group, California Institute of Technology, Pasadena, CA 91125, USA
[10] Department of Physics, The Pennsylvania State University, University Park, PA 16802, USA
[11] Institute for Gravitation and the Cosmos, The Pennsylvania State University, University Park, PA 16802, USA
[12] Université Côte d'Azur, Observatoire Côte d'Azur, ARTEMIS, 06304 Nice, France
[13] Max Planck Institute for Gravitational Physics (Albert Einstein Institute), D-14476 Potsdam, Germany
[14] OzGrav, Charles Sturt University, Wagga Wagga, NSW 2678, Australia
[15] Department of Physics, Tamkang University, Danshui Dist., New Taipei City 25137, Taiwan
[16] Institut d'Astrophysique de Paris, Sorbonne Université & CNRS, UMR 7095, 98 bis bd Arago, F-75014 Paris, France
[17] Université Paris Cité, CNRS, Astroparticule et Cosmologie, F-75013 Paris, France
[18] Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA), Northwestern University, 1800 Sherman Ave, Evanston, IL 60201, USA
[19] Kavli Institute for Cosmological Physics, The University of Chicago, 5640 S. Ellis Ave., Chicago, IL 60615, USA
[20] Centre for Cosmology, Particle Physics and Phenomenology (CP3), Université catholique de Louvain, Louvain-la-Neuve, B-1348, Belgium
[21] Università di Napoli "Federico II," Dipartimento di Fisica "Ettore Pancini,"Compl. Univ. di Monte S. Angelo, Via Cinthia 21, I-80126, Napoli, Italy
[22] INFN, Sezione di Napoli, Compl. Univ. di Monte S. Angelo, Edificio G, Via Cinthia, I-80126, Napoli, Italy
[23] Physics Department, American University, Washington, DC 20016, USA
[24] Department of Physics, Texas Tech University, Lubbock, TX 79409, USA
[25] School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455, USA
[26] Institut de Física d'Altes Energies (IFAE), The Barcelona Institute of Science and Technology, Campus UAB, E-08193 Bellaterra (Barcelona) Spain
[27] Department of Physics, University of Florida, Gainesville, FL 32611, USA
[28] Università di Pisa, I-56127 Pisa, Italy
[29] INFN, Sezione di Pisa, I-56127 Pisa, Italy
[30] LAPP, CNRS, 9 Chemin de Bellevue, F-74941 Annecy-le-Vieux, France
Received 2023 March 27; revised 2023 May 14; accepted 2023 May 18; published 2023 July 14

Abstract

The collection of gravitational waves (GWs) that are either too weak or too numerous to be individually resolved is commonly referred to as the gravitational-wave background (GWB). A confident detection and model-driven characterization of such a signal will provide invaluable information about the evolution of the universe and the population of GW sources within it. We present a new, user-friendly, Python-based package for GW data analysis to search for an isotropic GWB in ground-based interferometer data. We employ cross-correlation spectra of GW detector pairs to construct an optimal estimator of the Gaussian and isotropic GWB, and Bayesian parameter estimation to constrain GWB models. The modularity and clarity of the code allow for both a shallow learning curve and flexibility in adjusting the analysis to one's own needs. We describe the individual modules that make up `pygwb`, following the traditional steps of stochastic analyses carried out within the LIGO, Virgo, and KAGRA Collaboration. We then describe the built-in pipeline that combines the different modules and validate it with both mock data and real GW data from the O3 Advanced LIGO and Virgo observing run. We successfully recover all mock data injections and reproduce published results.

Unified Astronomy Thesaurus concepts: Gravitational wave astronomy (675); Gravitational waves (678); Gravitational wave detectors (676)

1. Introduction

Since the first direct gravitational-wave (GW) detection (Abbott et al. 2016), the field of GW astrophysics has exploded, now encompassing a wide range of instrumental and observational campaigns across the globe. These detection efforts monitor a vast range of frequencies, from the nanohertz to the kilohertz, and are sensitive to a multitude of GW sources emitting therein. While the GW sources in each band may present extremely different characteristics, a potential candidate for *all* GW measurements is a gravitational-wave background (GWB), given by the collection of all GWs too faint to be individually resolved or by the incoherent overlap of a large

number of signals in the same band (Regimbau 2011; Christensen 2019; Renzini et al. 2022). This sort of signal has been targeted in several different data sets (Abbott et al. 2004, 2007, 2009, 2017, 2019, 2021) using search methods that estimate the GW strain signal power, modeling the signal as stochastic, frequently resorting to cross-correlation of multiple independent observations (Allen & Romano 1999). These searches are often referred to as stochastic searches by the GW detection community, and these backgrounds are often referred to as stochastic GWBs (SGWBs), even though, in practice, not all target background signals are fully described by stochastic variables,[31] and this definition may imply an approximation. So far, no confident detection of a GWB has been claimed.

With this paper, we present `pygwb` (Renzini et al. 2023), a new Python-based package tailored to searches for isotropic GWBs with current ground-based interferometers, namely the Laser Interferometer Gravitational-wave Observatory (LIGO; Aasi et al. 2015a), the Virgo observatory (Acernese et al. 2014), and the KAGRA detector (Akutsu et al. 2020), and with the potential to be expanded and adapted to several other detection efforts. The core analysis tools, described in detail in what follows, are heavily inspired by the LIGO, Virgo, and KAGRA Collaboration (LVK) stochastic analysis code, `stochastic.m`. The latter consists of a set of `MATLAB` scripts easily parallelizable on a high-throughput computing cluster, and has been used in LVK data analysis for the past data acquisition runs (Abbott et al. 2007, 2009, 2017, 2019, 2021). These include the three observing runs: O1 (2015 September–2016 January), O2 (2016 November–2017 August), and O3 (2019 April–2020 March), performed with Advanced LIGO Hanford and Livingston and Advanced Virgo Acernese et al. (2014) for parts of O2 and O3. Data from Virgo have been included in stochastic analyses as of the latest observing run. The analysis consists in the calculation of an optimal statistic (Allen & Romano 1999) from the data of multiple interferometers, which is directly related to the amplitude of the GWB signal.

A notable change throughout the years of stochastic GW analyses has been the constant shift toward Bayesian parameter estimation (Mandic et al. 2012; Abbott et al. 2021). To date, there is no preferred stochastic parameter estimation software, and different groups have employed private scripts. To extend the scope of the stochastic search beyond the optimal statistic, we include a parameter estimation module in `pygwb`, based on the `Bilby` package (Ashton et al. 2019), which allows the user to test both predefined and user-defined models and obtain posterior distributions on the parameters of interest.

The steady inflow of ever-improving GW data open for analysis (LIGO Scientific Collaboration et al. 2021) has been a catalyst for open-source GW data analysis codebase development. By adopting the Python language and focusing on user-friendliness, flexibility, and portability, we intend to introduce stochastic searches to the wider GW community. Detecting a GWB with ground-based interferometers will be a community effort, and we expect search pipelines to evolve along the way. The format and structure of `pygwb` facilitates this evolution, and conversely the package is suitable for beginners approaching GWB data analysis for the first time.

This paper is structured as follows. In Section 2, concepts related to the characterization and detection methods of a GWB are reviewed. A detailed overview of the individual modules that make up the `pygwb` package follows in Section 3, outlining the steps of LVK stochastic analyses. Several manager objects that store relevant data and handle the analysis internally are described in Section 4. The built-in `pygwb` pipeline, which combines individual modules and performs the search for an isotropic GWB, is presented in Section 5. To test the capabilities of the pipeline, mock data sets with a variety of simulated signals are analyzed in Section 6.1. To conclude, results from the analysis of the third LVK collaboration observing run, O3, are presented and compared with collaboration results in Section 6.2.

2. The Isotropic Stochastic Analysis

An SGWB is characterized by its spectral emission, which is the target of stochastic GW searches. The spectrum is typically parameterized by the GW fractional energy density spectrum $\Omega_{\rm GW}(f)$, such that

$$\Omega_{\rm GW}(f) = \frac{1}{\rho_{\rm c}}\frac{d\rho_{\rm GW}(f)}{d\ln f}, \tag{1}$$

where $d\rho_{\rm GW}$ is the energy density of GWs in the frequency band f to $f+df$ and $\rho_{\rm c}$ is the critical energy density in the universe. When integrated over $d\log f$, $\Omega_{\rm GW}(f)$ gives the total dimensionless GW energy density. The $\Omega_{\rm GW}(f)$ spectrum is thus directly related to the intensity of GWs. Specifically, from Equation (1), it follows that (Allen & Romano 1999)

$$\Omega_{\rm GW}(f) = \frac{4\pi^2 f^3}{\rho_{\rm c} G} S_h(f), \tag{2}$$

where the strain spectral density $S_h(f)$ is defined as the polarization-averaged second moment of the stochastic GW strain field, decomposed into its polarization components h_+ and $h_\times$,

$$\begin{aligned}&\langle h_+(f,\hat{\boldsymbol{n}})h_+^*(f',\hat{\boldsymbol{n}}')\rangle + \langle h_\times(f,\hat{\boldsymbol{n}})h_\times^*(f',\hat{\boldsymbol{n}}')\rangle \\ &= \delta^{(2)}(\boldsymbol{n},\boldsymbol{n}')\delta(f-f')S_h(f,\hat{\boldsymbol{n}}),\end{aligned} \tag{3}$$

assuming statistical homogeneity. The unit vectors $\hat{\boldsymbol{n}}$, $\hat{\boldsymbol{n}}'$ span the 2-sphere, while $f \in \mathbb{R}$. In the plane wave formalism, h_+ and $h_\times$ in Equation (3) are the Fourier coefficients of the time-domain strain fields. If these are stochastically distributed, these give rise to an SGWB that we describe solely through the statistical moments of the distribution. In particular, a Gaussian SGWB is fully described by its second moments, hence the spectral density in Equation (3) is the primary target of a search that assumes the signal to be both stochastic and Gaussian. More details on these quantities can be found, for example, in Romano & Cornish (2017).

Laser interferometers such as LIGO and Virgo are sensitive to the strain field in the time domain coming from all directions, $h(t)$. These detectors measure the GW strain filtered through a linear response function F (see the definition in Romano & Cornish 2017), plus a detector noise component n, which we may write in shorthand as

$$d(t) = F(t)\star h(t) + n(t), \tag{4}$$

[31] To avoid confusion, in this paper we will use the term SGWB to refer to signals that are indeed defined as stochastic fields.

where "⋆" indicates a convolution operation. Given that the SGWB signal is weak and hard to distinguish from instrumental noise, cross-correlating two independent, time-coincident data streams with uncorrelated noise is an effective way to construct an estimator for $\Omega_{\rm GW}(f)$. We assume our target stochastic GW signal is stationary, Gaussian, and isotropic. We further assume the detector noise is Gaussian and uncorrelated between detectors, which is a fair assumption in the case of ground-based interferometers at current detector sensitivity[32] (after specific mitigation; Abbott et al. 2021; Janssens et al. 2023), and that the noise amplitude is much larger than the signal amplitude. Under these assumptions,[33] it has been shown (Aasi et al. 2015b; Romano & Cornish 2017) that the cross-correlation-based minimum variance unbiased estimator (MVUE) of $\Omega_{\rm GW}$ at a frequency bin f and the corresponding variance is given by

$$\hat{\Omega}_{{\rm GW},f} = \frac{\Re[C_{IJ,f}]}{\gamma_{IJ}(f) S_0(f)} \tag{5}$$

and

$$\sigma^2_{{\rm GW},f} = \frac{1}{2T\Delta f} \frac{P_{I,f} P_{J,f}}{\gamma^2_{IJ}(f) S^2_0(f)}, \tag{6}$$

where $C_{IJ,f}$ is the one-sided cross-spectral density (CSD) and $P_{I,f}$ is the one-sided (auto-)power spectral density (PSD) of the strain data d_t from two detectors (I,J), as defined below in Section 3.2.[34] Note that throughout this work we will denote continuous functions of the frequency with the notation (f), whereas discrete functions of the frequency will be denoted with a subscript $_f$. Typically, in this paper, discrete functions of frequency are estimators for continuous functions, and in equations such as Equations (5) and (6), which mix discrete and continuous functions, our notation implies that continuous functions are evaluated at the discrete set of frequencies for which we know the values of the discrete functions. In the above, T is the duration of the data used to produce the above spectral densities, and $\gamma_{IJ}(f)$ is the cross-correlated GW response, or overlap reduction function (ORF), which is the polarization- and sky-averaged cross-correlation of the individual detector responses, F_I. The ORF normalized for a pair of perpendicular-arm interferometers is given by Allen & Romano (1999):

$$\gamma_{IJ}(f) = \frac{5}{8\pi} \sum_A \int_{S^2} d\hat{\boldsymbol{n}} F^A_I(f, \hat{\boldsymbol{n}}) F^A_J(f, \hat{\boldsymbol{n}}) e^{-i2\pi f \hat{\boldsymbol{n}} \cdot (\boldsymbol{x}_I - \boldsymbol{x}_J)}, \tag{7}$$

where $\hat{\boldsymbol{n}}$ is the unit vector on the sky, in an arbitrary basis,[35] $\boldsymbol{x}_I - \boldsymbol{x}_J$ is the difference between the position vectors of the two detectors I and J, respectively, and A spans the polarization basis. The ORF quantifies the reduction in sensitivity of the cross-correlation stochastic search due to the detectors not being coaligned and colocated and having different nontrivial responses. The function S_0 is defined as (Romano & Cornish 2017; Renzini et al. 2022)

$$S_0(f) = \frac{3H_0^2}{10\pi^2} \frac{1}{f^3} \tag{8}$$

and converts a GW strain power spectrum into a fractional energy density. The derivation of S_0 is shown in Allen & Romano (1999), and note that it includes the normalization factor of the ORF, $5/8\pi$, which ensures $\gamma_{IJ}(f) \equiv 1$ for coaligned, colocated detectors.

There are two important considerations to make regarding the estimator in Equations (5) and (6). First, the implementation of a discrete Fourier transform (DFT) over a finite time T in the estimator of the continuous nonperiodic quantity $\Omega_{\rm GW}(f)$ may create spectral artifacts, as seen in Whelan (2004) and Press et al. (2007). We outline how this is handled in Section 3.2. Second, as the estimator is initially derived as a minimal variance estimator in the time domain (Allen & Romano 1999), the narrowband frequency estimator in Equation (5) is actually obtained from a broadband one, as will be clarified in Section 3.3. In the rest of this paper, we refer to $\hat{\Omega}_{{\rm GW},f}$ as the optimal estimator of the signal spectrum $\Omega_{\rm GW}(f)$. The optimality of the estimator can either be justified by the proof that this is an MVUE or equivalently by showing that it maximizes a reasonable likelihood for the data. When performing parameter estimation, as outlined in Section 3.6, we in fact employ a Gaussian likelihood that is maximized by $\hat{\Omega}_{{\rm GW},f}$.

In stochastic analyses with current interferometers, we take advantage of long observing times to improve detection statistics. In practice, the data are segmented into smaller chunks and analyzed individually, before they are optimally combined to produce an estimate. This is convenient, due to potential nonstationarities in the detector noise over both short timescales, such as the length of an individual data segment, and long timescales, such as the total observation time, as well as reducing computational costs. Assuming each time segment is independent, we perform a weighted average over all segments to calculate $\hat{\Omega}_{{\rm GW},f}$ for long observations. This average can be thought of as an approximation to the ensemble averages in Equation (3). Hence the more independent observations that are averaged over, the better the measurement. The averaging procedure is described in full in Section 3.3.

The narrowband statistic of Equations (5) and (6) assumes each frequency bin is independent. The information from each bin can be combined under the assumption of a known GW spectral density distribution. In GWB analyses, it is most common to assume a power-law (PL) spectral shape for $\Omega_{\rm GW}$:

$$\Omega_{\rm GW}(f) = \Omega_{\rm ref} \left(\frac{f}{f_{\rm ref}} \right)^{\alpha}, \tag{9}$$

[32] In future detectors, correlated noise will become a significant problem, and quite a few methods for mitigating it have been proposed, including Wiener filtering and Bayesian parameter estimation (Thrane et al. 2013, 2014; Coughlin et al. 2016, 2018; Himemoto & Taruya 2017; Himemoto & Taruya 2019; Meyers et al. 2020; Janssens et al. 2021, 2023; Himemoto et al. 2023).

[33] Failure of stationarity or Gaussianity implies the estimator is suboptimal, yet still valid (Drasco & Flanagan 2003; Lawrence et al. 2023); failure of isotropy would also induce a bias, and the target signal would be ill-defined (Tsukada et al. 2023.

[34] Note that, in previous works, the notation C_{IJ} was used to define the cross-correlation statistic itself Abbott et al. (2021). This is not the case in this paper.

[35] The ORF in `pygwb` is calculated in geocentric coordinates.

where α is the spectral index of the signal, $f_{\rm ref}$ is a reference frequency, and $\Omega_{\rm ref}$ is defined as $\Omega_{\rm ref} \equiv \Omega_{\rm GW}(f_{\rm ref})$. Under this assumption, the rescaling

$$H_{{\rm ref},\alpha}(f) = \left(\frac{f}{f_{\rm ref}}\right)^{\alpha} \tag{10}$$

can be used to reweight the estimate of the spectrum $\hat{\Omega}_{{\rm GW},f}$, obtained for $\alpha = 0$, to optimize the statistic for a specific spectral index α at a chosen reference frequency $f_{\rm ref}$, reducing the search to the estimation of a single number, $\Omega_{\rm ref}$. This procedure is referred to as *reweighting* and is clarified in Section 3.3. Alternatively, it is also possible to keep α as a free parameter in the analysis and estimate both $\Omega_{\rm ref}$ and α from the data. This is described in Section 3.6.

3. Individual Modules

What follows is a detailed step-by-step presentation of the stochastic analysis pipeline. We follow the natural structure of the code for clarity, as we introduce each module individually. To start, we present the `preprocessing` module, which preconditions the time-domain strain from GW detectors for spectral analysis. In `spectral`, we explain the power spectrum and cross-spectrum calculations, which produce the $P_{I,f}$ and $C_{IJ,f}$ spectra in Equations (5) and (6). We then describe `postprocessing`, which includes the averaging procedures employed over large data sets to obtain an optimal estimate of the signal amplitude, starting from the quantities in Equations (5) and (6), and knowledge of the expected spectral shape. In `delta-sigma cut` and `notch`, we present modules that focus on data quality checks and the implementation of relevant time-domain and frequency-domain data cuts. We then describe the built-in parameter estimation module `pe`, based on `Bilby` (Ashton et al. 2019), a Python-based Bayesian inference library widely used in GW data analysis. Finally, we present the `simulator` module, which includes different mock data injection techniques for GWB study and detection validation.

A schematic of the `pygwb` package is presented in Figure 1. This includes the manager objects `Interferometer`, `Baseline`, and `Network`, presented in Section 4.

3.1. `Preprocessing`

Preprocessing is the first step of stochastic GW data analysis, in which data are read, downsampled, and high-pass-filtered. The pipeline can use public data available from the Gravitational-wave Open Science Center (GWOSC; LIGO Scientific Collaboration et al. 2021), private data (data stored on the LVK servers restricted to members of the collaboration), or local data. Data are read using existing `gwpy` (Macleod et al. 2021) `TimeSeries` methods. We denote the raw data measured at detector I over the time period T by $s_I(t_k)$ in what follows, where t_k are discrete times given by $t_k \equiv k\delta t$. The values of k are positive integers between 0 and $T/\delta t - 1$ and δt is the sampling period, which in the LIGO, Virgo, and KAGRA interferometers is $1/(16384\ {\rm Hz})$. The raw strain data from the two interferometers, $s_I(t_k)$ and $s_J(t_k)$, are downsampled to a user-defined sampling frequency $f_{\rm samp}$, using a user-defined resampling window (a *Hamming* window, by default). The downsampling is performed to reduce the memory and computational requirements of the analysis. This is achieved using an existing `gwpy TimeSeries` filtering method for strain data. Note that selecting an $f_{\rm samp}$ implies fixing a Nyquist frequency of $f_{\rm Nyquist} = f_{\rm samp}/2$ for the analysis. The Nyquist frequency is the highest frequency included in the Fourier expansion at a given sampling rate. Hence, frequencies above it cannot be probed. To avoid this becoming a limitation, $f_{\rm samp}$ should be chosen to be high enough to contain the full spectrum of the signal of interest, within the reasonable sensitivity of the detector.

The low-frequency content of ground-based interferometer data (in particular, below 10 Hz) is dominated by seismic and control noise (Buikema et al. 2020). For this reason, frequencies below a given (user-defined) cutoff frequency are high-pass-filtered, i.e., excluded from the analysis. In previous isotropic GWB searches (Abbott et al. 2009, 2017, 2019, 2021), the input data are high-pass-filtered using a 16th-order Butterworth filter with a specific knee frequency. A 16th-order Butterworth filter is built by first computing its transfer function (in zero-pole-gain form) using the `scipy` library and then filtering the data with the relevant `gwpy TimeSeries` method. The design of the high-pass filter is fixed in the module, only allowing the user to specify the knee frequency. The default value of the knee frequency is 11 Hz, which was chosen to avoid spectral leakage from the noise power spectrum below 20 Hz (Abbott et al. 2021). See Figure 2 for an example of data before and after preprocessing.

At this point, the data may also be screened for large bursts of power in the detector data with a high signal-to-noise ratio (SNR) or *glitches*, due to instrumental or environmental disturbances, which are known to bias estimates of stochastic analyses (Usman et al. 2016; Pankow et al. 2018; Davis et al. 2021; Acernese et al. 2022a; Davis & Walker 2022). Historically, segments with loud glitches were flagged and excluded from analysis by nonstationarity cuts (see Section 3.4). In O3, a series of exceptionally loud glitches appeared in the data, which led to large fractions of data being removed by previously employed nonstationarity cuts (Abbott et al. 2021). Hence, an alternative technique called *gating* was employed to address these loud glitches and drastically reduce the amount of data removed (Matas et al. 2021). Gating is performed internally by `pygwb` by multiplying the data by an inverse *Planck-taper* window (McKechan et al. 2010). Time periods around samples in the whitened data that have an absolute value above a chosen threshold are marked for gating independently for each interferometer. The width of the gate must be sufficiently large to remove the entirety of the relevant glitch. The required width may change, based on the data quality of the specific data in the analysis, and hence must be empirically determined. The tapering length of the window must also be sufficiently long to minimize the addition of artifacts by the gating; 0.25 s is found to be sufficient (Davis & Walker 2022). This technique is generically beneficial for the analysis of data that are non-Gaussian, such as real GW detector data. The gating implemented in `pygwb` is highly customizable to the specific needs of the analysis; default gating parameters are shown below in Table 1. For more details on gating and parameter choices, see Davis & Walker (2022).

Finally, the module also allows the performance of a time-shifted analysis in which one of the two time series is shifted in time by an integer number of seconds before the cross-correlation is performed. This technique is employed as a detector noise characterization tool, since it removes the potential correlation due to a broadband GWB, while

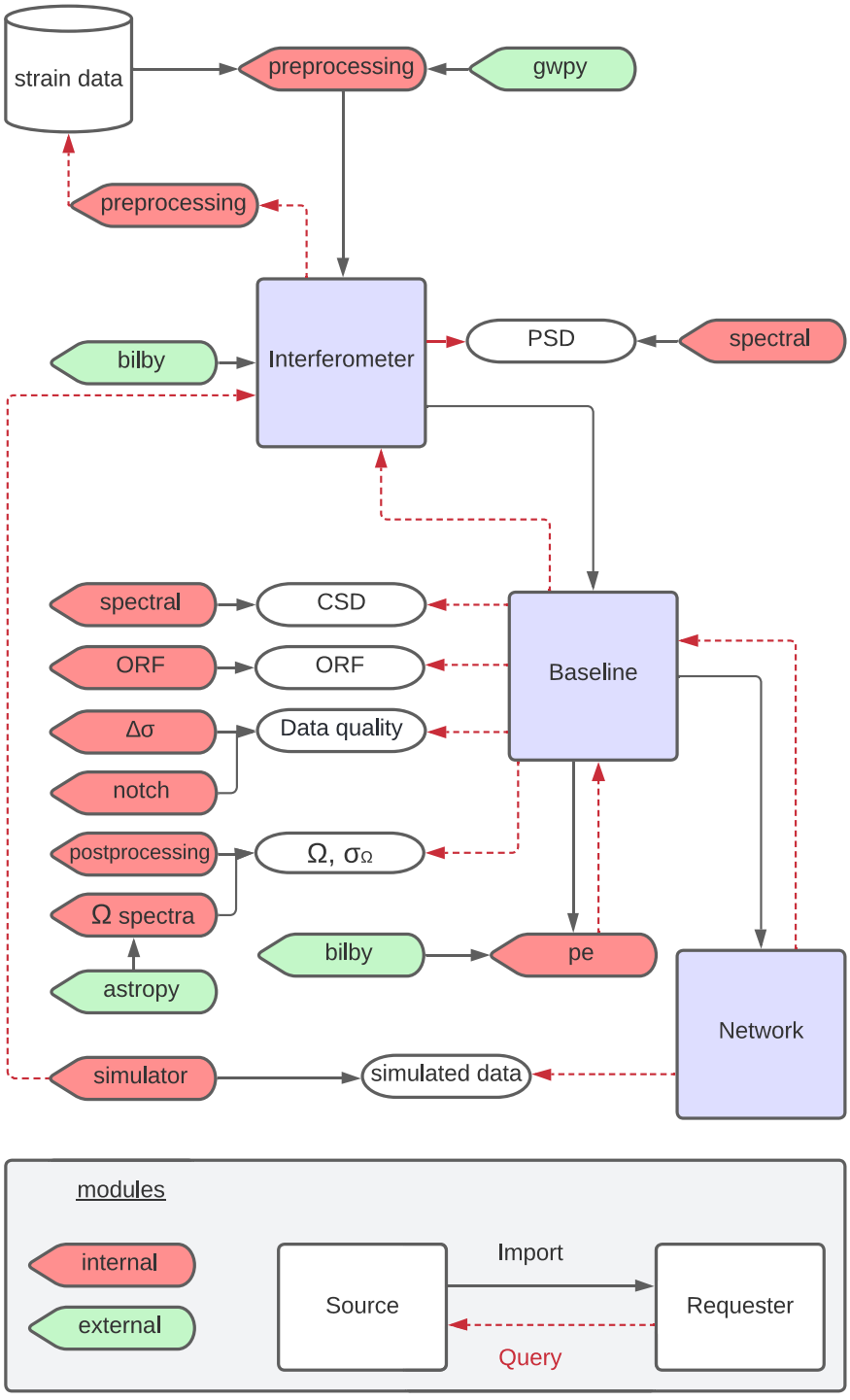

Figure 1. Schematic overview of the `pygwb` analysis flow. In the blue squares, we show the manager objects of the code that handle the analysis internally. These manager objects query (red arrows) different modules for specific objects, calculations, or quantities (rounded bubbles), imported (gray arrows) by either internal (i.e., within `pygwb`) or external modules (i.e., outside of `pygwb`). Internal modules are indicated in red, while external modules are indicated in green.

preserving instrumental correlations with coherence times greater than the applied time shift, like nearly sinusoidal spectral artifacts from, e.g., electronics (Covas et al. 2018).[36]

[36] It is worth noting that this time shift will probably not help identify correlated broadband stochastic noise, such as correlated magnetic noise from Schumann resonances, as this is largely caused by lightning strikes and the correlation between detectors is due to seeing the same stochastic signal in both detectors. This is in contrast to chance coherence between coincident periodic artifacts (*lines*) at multiple sites that one can find by implementing time shifts.

The time shift is a user-defined parameter that should always be greater than the light travel time between detectors (i.e., 10 ms for the LIGO Hanford and Livingston detectors) and smaller than the segment duration. Typically, a time shift of 1 s is used.

3.2. `Spectral`

The role of the `spectral` module is to compute, for each time segment of duration T, the discrete frequency-domain

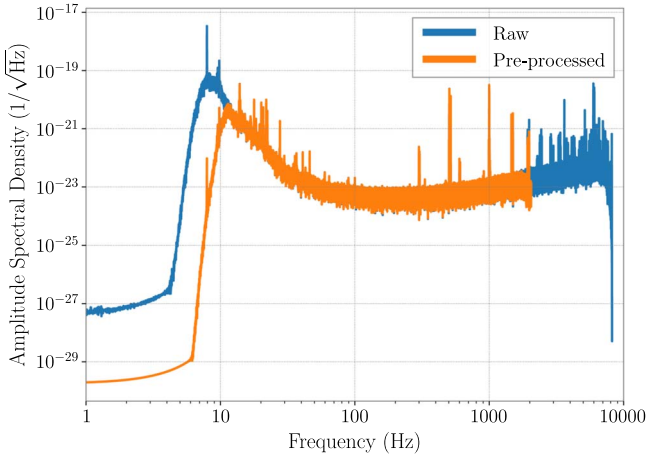

Figure 2. Comparison between the amplitude spectral density of a raw (blue solid line) and preprocessed (orange solid line) 192 s segment of LIGO Livingston O3 data. Preprocessing consists of downsampling the data to 4096 Hz and then removing the low-frequency content below 10 Hz.

quantities $C_{IJ,f}$, $P_{I,f}$, and $P_{J,f}$ used in Equations (5) and (6). The one-sided cross- and auto-PSDs C_{IJ} and P_I, respectively, of a single segment are defined as

$$C_{IJ,f} = \frac{2}{T}\tilde{s}^*_{I,f}\tilde{s}_{J,f}, \quad P_{I,f} = \frac{2}{T}|\tilde{s}_{I,f}|^2, \tag{11}$$

where $\tilde{s}_f$ are DFTs of $s(t_k)$, defined by

$$\tilde{s}_f \equiv \sum_{t_k=0}^{T-\delta t} s(t_k)\, e^{-i2\pi m t_k/T}, \tag{12}$$

where $f = m\delta f$, with m a natural number between 0 and $1/(2\,\delta t\,\delta f)$, and δf is the desired frequency resolution, chosen such that $1/(2\delta t\delta f)$ is an integer.

The segmented data are windowed before calculating the Fourier transforms to avoid spectral leakage due to discontinuities at the ends of the segments. The user may define their own choice of window, which defaults to the *Hann* window if none is selected. The `spectral` module uses methods from `scipy.signal` to calculate spectrograms $\tilde{s}^t_f$ of the given data, which are then used to calculate the list of $C^t_{IJ,f}$, $P^t_{I,f}$, and $P^t_{J,f}$ quantities, corresponding to different time segments labeled by t in the data set. By default, these are calculated with a 50% time overlap, to account for the impact of the windowing. However, the user may redefine the overlap between consecutive segments to be used throughout the analysis to better suit any choice of window.

Different averaging procedures are employed to reduce the fluctuations in the spectra estimates and compress the data. The procedures we employ are selected to minimize sensitivity loss. In the estimates of $P^t_{I,f}$ and $P^t_{J,f}$, we employ Welch's estimation method of PSDs (Welch 1967), which is known to produce minimum variance estimates of the PSD, implemented as follows. Each segment is divided into subsegments of duration $1/\delta f$, which are DFTed individually. The autocorrelated power $|\tilde{s}_{I,f}|^2$ is then averaged over the subsegments to obtain estimates of $P^t_{I,f}$ and $P^t_{J,f}$ for time t. This procedure returns spectra at the desired frequency resolution δf, which is typically much larger than the original resolution $1/T$ Hz.

As the power varies slowly with frequency,[37] we can average over neighboring frequencies using a process known as coarse-graining (Talbot et al. 2021). This is the default procedure employed in the CSD estimation. The resulting spectra are returned at the desired frequency resolution δf. Note that the data are zero-padded before calculating Fourier transforms for $C^t_{IJ,f}$, to avoid wraparound problems arising from finite data (Abbott et al. 2004; Whelan 2004; Press et al. 2007), and hence coarse-graining is required to achieve the desired frequency resolution. Zero-padding simply entails appending a vector of zeros equal to the length of the segment before taking the Fourier transform.

To further reduce fluctuations in the PSD estimates, the $P^t_{I,f}$ quantities are averaged over neighboring segments to obtain the final estimate $\bar{P}^t_{I,f}$ of the PSD at a given time t. This is appropriate, as the noise in GW detectors is (most often) approximately stationary over periods of a few minutes. We often refer to the initial (unbarred) quantities as "naive" and the final (barred) quantities as "average" estimates in the rest of this paper, to avoid confusion. By default, only nearest neighbors are used for the calculation, such that the PSD at time t is an average of the naive PSDs calculated for times $t - T$ and $t + T$. The user may define any even number D of segments to be used to perform this average, which are taken before and after the reference time t, such that the PSD is averaged over naive PSDs at times $t - DT/2$ and $t + DT/2$.

Figure 3 shows the cross- and auto-PSDs of 192 s of data from the Hanford and Livingston detectors during O3, while Figure 4 shows a 2 hr spectrogram of Hanford data during O3, produced with the `spectral` module.

3.3. `Postprocessing`

Once a set of data, comprised of an uninterrupted stretch of time-series data, has been preprocessed and average PSD and CSD estimates have been calculated for each segment of data within the set, one can combine those separate time segments to construct a final, time-averaged estimate of the GWB amplitude.

Due to the aggressive windowing choice we typically make, and the subsequent overlapping of time segments, we must be careful in combining time segments together. The overlapping and windowing cause overlapping time segments to be correlated with one another. Within each processed set, individual time segments must be combined while accounting for this covariance. A detailed calculation and discussion of this covariance can be found in Lazzarini & Romano (2004), while effective approximations to that full calculation can also be used (see, e.g., Section IIIB of Ain et al. 2015).

To start, we construct the estimate of the GWB in a single segment t. As detailed in Section 2, the GWB search is often framed in terms of constructing a point estimate for $\Omega_{\rm GW}(f_{\rm ref})$, the energy density of the GWB at the specific frequency $f_{\rm ref}$, assuming a PL for the GWB with spectral index α. We refer to the estimator of this quantity as $\hat{\Omega}^{\alpha}_{\rm ref}$, in general, and for a single time segment of data, it can be constructed using a weighted average over the individual frequency bin estimators $\hat{\Omega}_f$ and σ_f,

[37] The power varies slowly with frequency except in very few bins, where narrowband spectral artifacts or *lines* are present, as discussed in Section 3.5.

Table 1
Default Parameters for the `pygwb_pipe` Script as Well as the `Parameters` Data Class

Parameter	Default Value	Description
	Script Arguments	
`output_path`	`""`	Output data path
`calc_pt_est`	`True`	If `True`, calculate point estimates
`apply_dsc`	`True`	If `True`, apply $\Delta\sigma$ cut
`pickle_out`	`True`	If `True`, pickle postprocessed baseline
`wipe_ifo`	`True`	If `True`, set interferometer strain data to 0
	Data Specifics	
`interferometer_list`	`["H1", "L1"]`	List of (2) interferometers
`t0`	0	Analysis start time
`tf`	100	Analysis end time
`data_type`	`public`	Data accessibility
`channel`	`GWOSC-16KHZ_R1_STRAIN`	Data channel name
	Preprocessing	
`tag`	C00	Descriptive data tag
`new_sample_rate`	4096 Hz	Downsampled sample rate
`input_sample_rate`	16,384 Hz	Input sample rate
`cutoff_frequency`	11 Hz	Lower frequency cutoff
`segment_duration`	192 s	Individual segment duration
`number_cropped_seconds`	2 s	Preprocessing cropped seconds
`window_downsampling`	hamming	Downsampling window
`ftype`	fir	Downsampling filter
`time_shift`	0 s	Time shift duration
	Gating	
`gate_data`	`False`	If `True`, self-gate data
`gate_tzero`	1 s	0 time half-width duration
`gate_tpad`	0.5 s	Gating window tapering
`gate_threshold`	50	Gating threshold
`cluster_window`	0.5	Gating cluster window
`gate_whiten`	`True`	If `True`, whiten data before gating
	Spectral Density Estimation	
`frequency_resolution`	1/32 Hz	Output frequency resolution
`overlap_factor`	0.5	Consecutive segment fractional overlap
`N_average_segments_welch_psd`	2	Average PSD segment number
`zeropad_csd`	`True`	If true zeropad the CSD
	FFT Window Specifics	
`window_fft_dict`	`hann`	FFT window parameter dictionary
	Postprocessing	
`polarization`	`tensor`	ORF polarization basis
`alpha`	0	Spectral index α
`fref`	25 Hz	Reference frequency $f_{\rm ref}$
`flow`	20 Hz	Lowest frequency included
`fhigh`	1726 Hz	Highest frequency included
	Data Quality Specifics	
`notch_list_path`	`""`	Notch list file path
`calibration_epsilon`	0	Calibration coefficient
`alphas_delta_sigma_cut`	$[-5, 0, 3]$	List of $\Delta\sigma$ cut spectral indices
`delta_sigma_cut`	0.2	$\Delta\sigma$ cut cutoff value
`return_naive_and_averaged_sigmas`	`False`	If `True`, return both σ and $\bar{\sigma}$ used in $\Delta\sigma$ calculation
	Output Specifics	
`save_data_type`	`npz`	Output data type
	Local Data Locations	
`local_data_path_dict`	{}	Dictionary of local data paths

Note. Most of these choices reflect defaults chosen in the past when analyzing LIGO and Virgo data. Notably, the default start and end times for the analysis are not meaningful and represent placeholders for the user-defined times. A default initialization file is included in the package with meaningful start and end times present in the O3 open data set.

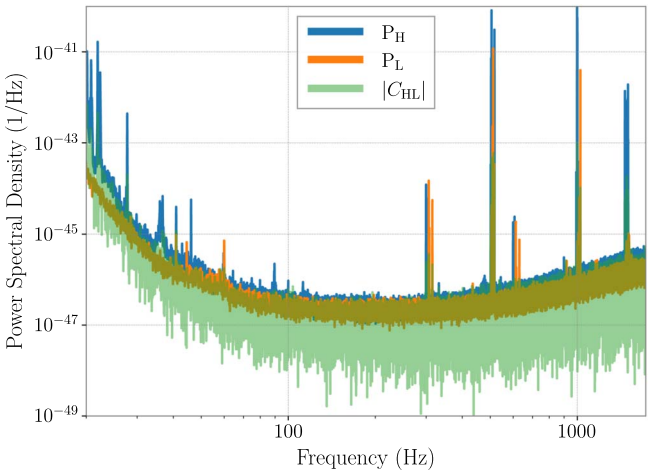

Figure 3. An example of cross- and auto-PSDs of the Hanford and Livingston detector data during O3.

described in Equations (5) and (6), calculated per segment t, as

$$\hat{\Omega}^{\alpha}_{\mathrm{ref},t} = \frac{\sum_f \hat{\Omega}_{t,f} H_{\mathrm{ref},\alpha}(f)\bar{\sigma}^{-2}_{t,f}}{\sum_f H^2_{\mathrm{ref},\alpha}(f)\bar{\sigma}^{-2}_{t,f}}, \tag{13}$$

$$\sigma^{\alpha}_{\mathrm{ref},t} = \left[\sum_f H^2_{\mathrm{ref},\alpha}(f)\bar{\sigma}^{-2}_{t,f}\right]^{-\frac{1}{2}}, \tag{14}$$

where the rescaling $H_{\mathrm{ref},\alpha}(f)$ is defined in Equation (10). The average variance spectrum per segment, $\bar{\sigma}^2_{t,f}$, is calculated using the average PSDs described in Section 3.2. These broadband quantities can be calculated for each time segment t, and then this set of estimators at each time can be combined to account for the overlap between time segments discussed above. We first lay out how to perform this combination assuming we have calculated the quantities above for each individual time segment. Then, we discuss how to alternatively average the estimators in each frequency bin over time independently, before combining them into an integrated quantity at the end. The latter calculation is normalized such that it gives the same result as the former. To avoid heavy notation, we drop the bars that indicate average quantities in the rest of this section—all variances used for the following calculations are average variances, as defined above.

To construct an estimator for the GWB using a set of measurements in short, overlapping time segments, we first combine the segments that are nonoverlapping. If the overlap between segments is 50% or less, then this amounts to separately performing inverse-noise-weighted averaging over the even- and odd-indexed segments:

$$\sigma^2_{\mathrm{odd}} = \frac{1}{\sum_{t\in\mathrm{odd}} \sigma_t^{-2}}, \tag{15}$$

$$\Omega_{\mathrm{odd}} = \frac{\sum_{t\in\mathrm{odd}} \Omega_t \sigma_t^{-2}}{\sum_{t\in\mathrm{odd}} \sigma_t^{-2}}, \tag{16}$$

where the quantities $\Omega_t \equiv \hat{\Omega}^{\alpha}_{\mathrm{ref},t}$ and $\sigma_t \equiv \sigma^{\alpha}_{\mathrm{ref},t}$ for each time segment t. Analogous expressions are calculated for Ω_{even} and σ_{even}. Subscripts refer to even/odd time segments, and we drop here the subscripts $_{\mathrm{GW,ref}}$, and $_{\alpha}$ used to construct the integrated quantities, to lighten the notation. We refer to the final, frequency- and time-averaged estimate as $\hat{\Omega}_{\mathrm{ref}}$ for now.

Next, we calculate the cross-covariance between point estimates in the odd and even segment combinations (Lazzarini & Romano 2004),

$$\sigma^2_{oe} = \sigma^2_{eo} \equiv \langle \Omega_{\mathrm{odd}}\Omega_{\mathrm{even}}\rangle - \langle\Omega_{\mathrm{odd}}\rangle\langle\Omega_{\mathrm{even}}\rangle \tag{17}$$

$$= \frac{1}{2}\frac{\bar{w}^4_{\mathrm{ovl}}}{\bar{w}^4}\left[\sigma^2_{\mathrm{odd}} + \sigma^2_{\mathrm{even}} - \frac{1}{2}\sigma^2_{\mathrm{odd}}\sigma^2_{\mathrm{even}}(\sigma_1^{-2} + \sigma^{-2}_{2M-1})\right], \tag{18}$$

where M is the number of *independent* segments, and so $2M-1$ is the total number of overlapping segments, with the *window factors* $\bar{w}^4_{\mathrm{ovl}}$ and $\bar{w}^4$, as defined in Appendix A. For the sake of compactness, we rewrite this as

$$\sigma^2_{oe} = \frac{k}{2}\sigma^2_{\mathrm{odd}}\sigma^2_{\mathrm{even}}\sigma^{-2}_{IJ}, \tag{19}$$

$$\sigma^2_{IJ} = \left[\sigma^{-2}_{\mathrm{odd}} + \sigma^{-2}_{\mathrm{even}} - \frac{1}{2}(\sigma_1^{-2} + \sigma^{-2}_{2M-1})\right]^{-1}, \tag{20}$$

where $k = \bar{w}^4_{\mathrm{ovl}}/\bar{w}^4$.

The covariance matrix between even/odd segment sets is then defined as

$$\boldsymbol{C} = \begin{pmatrix} \sigma^2_{\mathrm{odd}} & \sigma^2_{\mathrm{oe}} \\ \sigma^2_{\mathrm{oe}} & \sigma^2_{\mathrm{even}} \end{pmatrix}, \tag{21}$$

which we use to construct the optimal combination of segments to obtain the point estimate $\hat{\Omega}_{\mathrm{ref}}$ and its variance σ^2_{ref}. These are given by

$$\hat{\Omega}_{\mathrm{ref}} = \frac{\sum_{i=1}^2 \lambda_i \Omega_i}{\sum_{j=1}^2 \lambda_j}, \tag{22}$$

$$\sigma^2_{\mathrm{ref}} = b^2_{\mathrm{avg}}\left(\sum_{k=1}^2 \lambda_k\right)^{-2} \sum_{i=1}^2\sum_{j=1}^2 \lambda_i C_{ij} \lambda_j, \tag{23}$$

with

$$\lambda_i = \sum_{j=1}^2 (\boldsymbol{C}^{-1})_{ij}, \tag{24}$$

where the i, j indices label odd/even quantities. The bias factor b_{avg}, which arises due to the harsh windowing of the data, has been included in Equation (23). The derivation of the bias factor is described in Appendix A. If combining over nonoverlapping segments, then $\sigma^2_{\mathrm{oe}} = 0$, and this method reduces to the typical inverse-noise-weighted average that one would expect.

The above expressions are for a broadband estimator, but in practice the `postprocessing` module combines over time segments before combining over frequency bins. We refer to the estimated narrowband quantities as $\hat{\Omega}_{\mathrm{ref},f}$ and $\sigma_{\mathrm{ref},f}$. This notation indicates that, once a PL spectral model is applied, the estimate in a frequency bin represents an estimate of the GWB at the reference frequency of the PL, assuming the chosen spectral shape.

We normalize $\hat{\Omega}_{\mathrm{ref},f}$ and $\sigma_{\mathrm{ref},f}$, such that, when performing a weighted average over frequency bins *after* combining overlapping time segments, we get the same result as Equations (22)

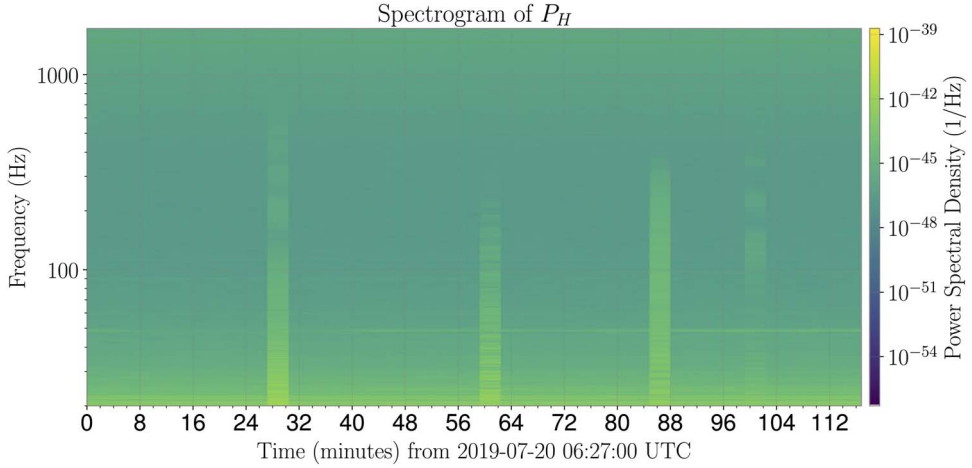

Figure 4. An example spectrogram showing 2 hr of LIGO Hanford data during O3. The visible vertical columns correspond to noisy segments, which are usually removed from the analysis (see Section 3.4).

and (23) (which assume construction of a broadband estimator *before* combining overlapping time segments). This results in the following expression for $\sigma^{-2}_{\mathrm{ref},f}$,

$$\sigma^{-2}_{\mathrm{ref},f} = b^{-2}_{\mathrm{avg}} \frac{[\sigma^{-2}_{\mathrm{odd},f} + \sigma^{-2}_{\mathrm{even},f} - k\sigma^{-2}_{IJ,f}]}{1 - \frac{k^2}{4}\sigma^2_{\mathrm{odd}}\sigma^2_{\mathrm{even}}\sigma^{-4}_{IJ}}, \tag{25}$$

and a corresponding expression for $\hat{\Omega}_{\mathrm{ref},f}$,

$$\hat{\Omega}_{\mathrm{ref},f} = \frac{\Omega_{\mathrm{odd},f}\sigma^{-2}_{\mathrm{odd},f}\left(1 - \frac{k}{2}\sigma^2_{\mathrm{odd}}\sigma^{-2}_{IJ}\right) + \Omega_{\mathrm{even},f}\sigma^{-2}_{\mathrm{even},f}\left(1 - \frac{k}{2}\sigma^2_{\mathrm{even}}\sigma^{-2}_{IJ}\right)}{\sigma^{-2}_{\mathrm{odd},f} + \sigma^{-2}_{\mathrm{even},f} - k\sigma^{-2}_{IJ,f}}. \tag{26}$$

The even and odd estimators for each frequency bin are defined as in Equations (15) and (16), except applied to individual bin-by-bin estimators calculated at each time segment. As discussed above, these expressions have been normalized, such that

$$\hat{\Omega}_{\mathrm{ref}} = \frac{\sum_f \hat{\Omega}_{\mathrm{ref},f}\sigma^{-2}_{\mathrm{ref},f}}{\sum_f \sigma^{-2}_{\mathrm{ref},f}}, \tag{27}$$

$$\sigma^2_{\mathrm{ref}} = \left[\sum_f \sigma^{-2}_{\mathrm{ref},f}\right]^{-1}. \tag{28}$$

The `postprocessing` module implements the above expressions to estimate $\hat{\Omega}_{\mathrm{ref},f}$ and $\sigma_{\mathrm{ref},f}$ at a fixed α, and returns them in form of an `OmegaSpectrum` object, which subclasses the classic `gwpy.FrequencySeries` and adds two key attributes: the spectral index α and the reference frequency f_{ref} at which the spectrum is calculated. By default, `pygwb` assumes a PL spectral index $\alpha = 0$ and a reference frequency $f_{\mathrm{ref}} = 25$ Hz when constructing the above estimators. To explicitly include the α dependence in our results, we refer to the final postprocessed spectra as $\hat{\Omega}^{\alpha}_{\mathrm{ref},f}$ and $\sigma^{\alpha}_{\mathrm{ref},f}$.

One of the advantages of averaging over time before averaging over frequency is that one can reweight $\hat{\Omega}^{\alpha}_{\mathrm{ref},f}$ and $\sigma^{\alpha}_{\mathrm{ref},f}$ to be estimators for different choices of α, without needing to average over all time segments again for a new choice of α. The `OmegaSpectrum` object has a built-in method to perform a reweighting to change either the f_{ref} or α used to calculate Ω_{ref}, employing the relation

$$\Omega^{\mathrm{ref}_1,\alpha_1}_{\mathrm{GW}}(f) = \Omega^{\mathrm{ref}_2,\alpha_2}_{\mathrm{GW}}(f)\frac{H_{\mathrm{ref}_1,\alpha_1}(f)}{H_{\mathrm{ref}_1,\alpha_2}(f)}, \tag{29}$$

derived using Equation (9), which implies the following relation between amplitudes at different reference frequencies:

$$\Omega_{\mathrm{ref}_1} = \Omega_{\mathrm{ref}_2}\frac{H_{\mathrm{ref}_2,\alpha}(f)}{H_{\mathrm{ref}_1,\alpha}(f)}. \tag{30}$$

This allows the quick calculation of time- and frequency-averaged estimates of the GWB amplitude associated with a specific PL model.

The default Hubble constant H_0, required in the scaling $S_0(f)$ in Equation (8), is chosen to be $H_0 = 67.7$ km/(Mpc$\cdots$), drawn from the Planck 2018 observations (Aghanim et al. 2020) and imported directly from the `astropy` package. This is an attribute of the `OmegaSpectrum` and may be reset by the user.

3.4. `Delta-sigma Cut`

In general, the noise level in ground-based detectors changes slowly on timescales of tens of minutes to hours. The variance σ^2_{GW} (see Equation (6)) associated to each segment is an indicator of that level of noise, which typically changes at roughly the percent level from one data segment to the next. However, there are occasional very loud disturbances to the detectors, such as glitches, which violate the Gaussianity of the noise. Autogating procedures are in place, as explained in Section 3.1, to remove loud glitches from the data; however, the procedure does not remove all nonstationarities. To avoid biases due to these noise events, an automated technique to exclude them from the analysis has been developed (Abbott et al. 2007). To this end, the `pygwb` package includes the `delta-sigma cut` module, which flags specific segments to be cut from the analyzed set. Note that inverse noise weighting, as explained in Section 3.3, also reduces the effect of non-Gaussian noise artifacts.

The "$\Delta\sigma$ cut" calculation consists of comparing the σ_{GW} of a segment t, σ_t, to that of its nearest neighbors and flagging it for removal in case their values differ by more than a chosen threshold. Conceptually, the calculation is based on the simple

inequality

$$\frac{|\sigma_i - \sigma_{i+1}| + |\sigma_i - \sigma_{i-1}|}{2\sigma_i} > \text{threshold}, \tag{31}$$

where i is a segment index. However, in practice, we perform an analogous, more sophisticated calculation, which compares the naive and average segment variances $\sigma_{t,\alpha}$ and $\bar{\sigma}_{t,\alpha}$. These are derived from the unweighted naive and average segment variances computed with Equation (6) using naive and average PSDs per segment (see Section 3.2 for details), respectively, which are then reweighted by the index α, as shown in Equation (14). The final expression employed in the calculation is

$$\frac{|\bar{\sigma}_{t,\alpha} b_{\text{avg}} - \sigma_{t,\alpha} b_{\text{nav}}|}{\bar{\sigma}_{t,\alpha} b_{\text{avg}}} > \text{threshold}, \tag{32}$$

which also takes into account the bias factors that arise due to the different impacts of windowing on naive and average quantities (see Appendix A for details). Past analyses have used a threshold of 0.2, as this has been shown to yield a Gaussian distribution for the remaining (uncut) segment variances (Abbott et al. 2009). For more details on this choice, see Meyers (2018).

The $\Delta\sigma$ cut calculation is performed assuming different spectral indices α, as each PL is sensitive to a different frequency band (see Figure 5). The union of all the segments flagged for each α is taken, leading to a full list of segments to discard from the analysis. The default choice of α values in the `delta-sigma cut` module is $\alpha = \{-5, 0, 3\}$, as this adequately covers most of the frequency band of LVK searches, from 20–1726 Hz (Abbott et al. 2021), at current sensitivity. These may be easily modified by the user. This would be especially recommended if the search were carried out over a different set of frequencies, or for data from detectors with a spectral sensitivity different than that for Advanced LIGO, Advanced Virgo, or KAGRA. Often, the value of $\alpha = 5$ is also considered, and was employed in the most recent LVK isotropic search (Abbott et al. 2021). The analysis performed at a spectral index $\alpha = -5$ is mostly sensitive to nonstationary effects in the ∼15–50 Hz range, while in the case of $\alpha = 0$ the analysis is sensitive to effects between ∼40–80 Hz, for $\alpha = 3$ from ∼90–500 Hz, and finally $\alpha = 5$ is most sensitive to fluctuations at the higher frequencies, above ∼500 Hz. These higher frequencies are not always included in this sort of analysis, due to reduced sensitivity in this range, hence $\alpha = 5$ is not a default value used for the cut.

As the $\Delta\sigma$ cut only compares neighboring segments, long stretches of loud noise-contaminated data can pass the test and be included in the analysis. We are currently working to improve this by monitoring and flagging longer stretches of nonstationary noise and prolonged loud noise conditions.

3.5. `Notch`

Ground-based laser interferometers present many narrow-frequency noise artifacts that are typically persistent in time and are generally referred to as noise lines. Some examples are calibration lines and mechanical resonances (Davis et al. 2021; Acernese et al. 2022b; van Remortel et al. 2023). The `notch` module provides the framework to properly deal with these noise lines in the case of the search for an isotropic GWB. The solution is to "notch out" these noise lines, i.e., set the values of the spectra at the affected frequency bins to zero. Note that the `notch` module is not built to identify these lines, as this is typically done by detector characterization experts working closely with instrumentalists running the detectors. Rather, the final product of the `notch` module is a frequency mask that may be applied to the relevant spectra in the analysis.

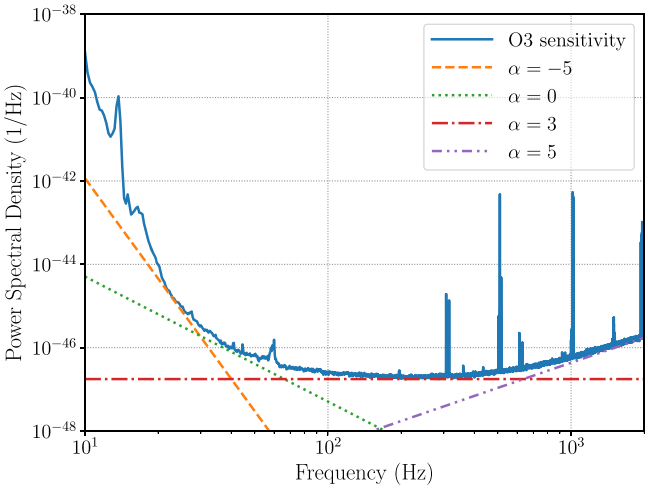

Figure 5. In this plot, PL spectra with different spectral indices are compared to the O3 sensitivity curve of LIGO Livingston. Each PL is sensitive to a different frequency band. This makes it necessary to repeat the $\Delta\sigma$ cut assuming different α, since this allows checking for noise fluctuations in the whole range of frequencies analyzed. The O3 sensitivity curve for LIGO Livingston was retrieved from O'Reilly et al. (2020).

The key object of the `notch` module is the `StochNotchList`, which is a list of `StochNotch` objects. A `StochNotch` object represents a physical noise line that has been identified and needs to be removed from the data analysis. The object has a minimum and maximum frequency, indicating the contaminated frequency region. Furthermore, it also comes with a descriptive string that allows the user to keep track of the reason why the line was notched. All the different `StochNotch` objects for a certain analysis are then stored in the `StochNotchList`, which contains the entire list of lines to be notched from the analysis.

The notch mask used to apply a set of notches within the analysis is constructed conservatively, such that any frequency that has overlapping frequency content with the noise lines defined in the `StochNotchList` will be removed when applying the notch mask. To maintain generality, we discuss here a generic estimated spectrum $\hat{\Omega}_f$, where its value at frequency f estimates $\Omega_{\text{GW}}(f)$ in the frequency range $[f - \delta f/2, f + \delta f/2]$, where δf is the chosen frequency resolution, as defined in Section 3.2. If a noise line has any overlap with the interval $[f - \delta f/2, f + \delta f/2]$, the f frequency bin is excluded. This implies that a hypothetical delta peak noise line at $f + \delta f/2$ leads to notching both f as well as $f + \delta f$.

We present the creation of a notch mask with an example in Figure 6, which illustrates how our conservative notching strategy excludes frequency bins based on different scenarios of noise lines.

The current code is set up to apply the same notches to an entire stretch of data, which can be considered "time-independent" notching. To allow for time-dependent notching, we could either use the current `notch` module and split the analysis in different segments, each having their own notch list.

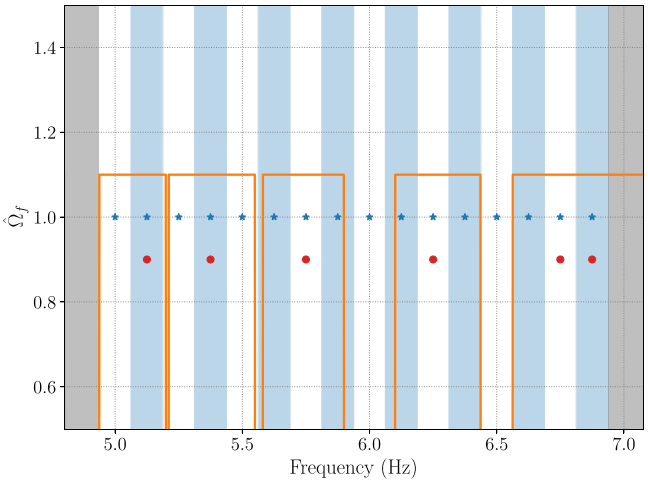

Figure 6. Example of how the notching of noise lines (orange lines) applied to the discrete measurements of the spectrum $\hat{\Omega}_{\mathrm{GW},f}$ (blue stars) leads to a final set of measurements (red dots). The vertical shaded regions indicate the bins, where even bins are white and odd bins are light blue. The orange lines trace out the noise lines, such that a noise line is present where the orange curve is zero. The analyzed data span [5.0, 6.875] Hz, in the unshaded region. In this example, there are five noise lines, from left to right: a noise line ending at the lowest-frequency bin, a noise line entirely contained in one frequency bin, a noise line spread across two frequency bins, a noise line spread across multiple frequency bins, and a noise line from bin edge to bin edge. After our notching procedure, the data are reduced to the bins marked by the red dots. For visual convenience, we have changed the amplitude in these remaining frequency bins by a factor 0.9.

Alternatively, one could extend the current module with an additional parameter that keeps track of which times have to be notched. Since typically the majority of the notched lines in the search for an isotropic GWB with data from the LIGO and Virgo detectors are present during the entire data set, the possible gain of implementing time-dependent notching is expected to be limited.

3.6. `Pe`

Starting from an estimate of the GWB spectrum $\hat{\Omega}_{\mathrm{GW},f}$, with variance $\sigma^2_{\mathrm{GW},f}$, it is possible to place stringent constraints on the GWB amplitude using a hybrid frequentist/Bayesian approach. We consider the general case where we have a set of GWB measurements $\hat{\Omega}^{IJ}_{\mathrm{GW},f}$ from different detector pairs, or *baselines*, *IJ*. We define a Gaussian likelihood for *B* pairs of detectors:

$$p(\hat{\Omega}^{IJ}_{\mathrm{GW},f}|\boldsymbol{\Theta}) \propto \exp\left[-\frac{1}{2}\sum_{IJ}^{B}\sum_{f}\left(\frac{\hat{\Omega}^{IJ}_{\mathrm{GW},f} - \Omega_{\mathrm{M}}(f|\boldsymbol{\Theta})}{\sigma^{IJ}_{\mathrm{GW},f}}\right)^2\right], \tag{33}$$

where $\Omega_{\mathrm{M}}(f|\boldsymbol{\Theta})$ is the GWB model and $\boldsymbol{\Theta}$ are its parameters. Bayes' theorem is used to obtain posterior distributions on the model parameters:

$$p(\boldsymbol{\Theta}|\hat{\Omega}^{IJ}_{\mathrm{GW},f}) \propto p(\hat{\Omega}^{IJ}_{\mathrm{GW},f}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}), \tag{34}$$

where the priors $p(\boldsymbol{\Theta})$ are employed. In practice, when performing parameter estimation on a large data set, we take the postprocessed, *unweighted* (i.e., $\alpha = 0$) estimate $\hat{\Omega}^{0,IJ}_{\mathrm{ref},f}$ to be the measured GWB spectrum in each frequency bin, and plug it into Equation (33). Note that it is necessary for the input spectra used in parameter estimation to be unweighted, as any other value would constitute a model choice and bias results.

Within the `pygwb` package, we include the `pe` module to perform parameter estimation as an integral part of the analysis, which naturally follows the computation of the optimal estimate of the GWB. This is a notable improvement compared to previous LVK analyses, where data products and parameter estimation were handled independently by packages in different programming languages. Furthermore, the `pe` module is a simple and user-friendly toolkit for any model builder to constrain their physical models with GW data.

The `pe` module is built on class inheritance, with `GWBModel` as the parent class. The methods of the parent class are functions shared between different GWB models, e.g., the likelihood formulation in Equation (33), as well as the noise likelihood, given by Equation (33) with $\Omega_{\mathrm{M}}(f|\boldsymbol{\Theta}) \equiv 0$. It is possible to include calibration uncertainty by modifying the `calibration_epsilon` parameter, which defaults to 0. For details on the marginalization over calibration uncertainty, see Appendix B and Whelan et al. (2014). The GW polarization used for analysis is user-defined, and defaults to standard general relativity (GR) polarization (i.e., tensor). More details on the possible polarization choices can be found in Section 4.2. In our implementation of `pe`, we rely on the `Bilby` package (Ashton et al. 2019) to perform parameter space exploration, and employ the sampler `dynesty` by default (Speagle 2020). The user has flexibility in choosing the sampler as well as the sampler settings.

Child classes in the `pe` module inherit attributes and methods from the `GWBModel` class. Each child class represents a single GWB model, and combined they form a catalog of available GWB models that may be probed with GW data. The inheritance structure of the module makes it straightforward to expand the catalog, allowing users of the `pygwb` package to add their own $\Omega_{\mathrm{M}}(f|\boldsymbol{\Theta})$ models. The flexibility of the `pe` module allows the user to combine several GWB models defined within the module. A particularly useful application of this is the modeling of a GWB in the presence of correlated magnetic noise, as discussed in Meyers et al. (2020), or the simultaneous estimation of astrophysical and cosmological GWBs (Martinovic et al. 2021). The `pygwb` documentation (Renzini et al. 2023) contains information on the existing models in the catalog, with a description of the GWB models and their parameters.

3.7. `Simulator`

To both design optimized stochastic analyses and understand our sensitivity to different categories of signals, it is essential to be able to readily simulate realistic interferometer data. To this end, the `simulator` module is primarily designed to generate data that correspond to an isotropic SGWB with a given PSD.

The GWB data in a network of interferometers satisfy a specific correlation matrix, which includes the set of ORFs of the entire detector network to account for the spatial separation and relative orientation of the detectors. Given a generic signal PSD, S_h, the correlation matrix $\boldsymbol{C}(f)$ is given by

$$C_{IJ}(f) = \delta_{IJ}P_I(f) + \gamma_{IJ}S_h(f). \tag{35}$$

Here $\gamma_{IJ}(f)$ is the normalized ORF of the baseline *IJ*, as shown in Equation (7), hence $\gamma_{II}(f) \equiv 1$, and P_I is the noise PSD of interferometer *I*. We have introduced a boldface notation,

which indicates matrices and vectors that span the detector space. The fact that the cross-correlation between detectors for $I \neq J$ only depends on the signal PSD assumes the noise is uncorrelated across all detectors.

The simulation of data correlated according to $\boldsymbol{C}(f)$ proceeds as follows. First, a vector of white, uncorrelated frequency-domain data is generated, $\boldsymbol{v}_f$, with a certain frequency resolution Δf. Then, the data are linearly transformed into the correlated $\boldsymbol{C}$ space by

$$\boldsymbol{x}_f^T = \boldsymbol{v}_f^T \sqrt{\boldsymbol{\Lambda}}_f \boldsymbol{E}_f^T, \tag{36}$$

where $\boldsymbol{\Lambda}$ and $\boldsymbol{E}$ are the eigenvalue and eigenvector matrices of $\boldsymbol{C}$, respectively, calculated in each frequency bin. This transformation results in data $\boldsymbol{x}_f$ that present the correct correlation, and have been colored with the injected noise and signal power spectra, where appropriate. Finally, the frequency-domain data vector is inverse DFTed to obtain a data vector in the time domain.

Data generation in the frequency domain, followed by the IFT to the time domain, can introduce edge effects in the simulated data segments. These may be avoided by *splicing* multiple data segments (Rabiner & Gold 1975). The splicing procedure combines neighboring data segments by windowing and overlapping them, and thus requires more data segments than the actual desired number of segments.

Concretely, we consider the example where $N_{\rm seg} = 1$ and detail the splicing procedure below. As the number of desired segments is 1, $2N_{\rm seg} + 1 = 3$ data segments are simulated. Assuming these are simulated following the procedure outlined above, we denote these three time-domain data segments by $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, and $\boldsymbol{x}_2$. A sine window, defined as

$$w_j = \sin\left(\frac{\pi j}{N}\right), \tag{37}$$

for $0 \leqslant j < N$, where N is the number of samples per segment, is used to window the data, which are then combined as

$$y_{0j} = w_j x_{0j}, \quad \boldsymbol{z}_0 = (\boldsymbol{y}_0[N/2\colon N], \boldsymbol{0}), \tag{38}$$

$$y_{1j} = w_j x_{1j}, \quad \boldsymbol{z}_1 = \boldsymbol{y}_1, \tag{39}$$

$$y_{2j} = w_j x_{2j}, \quad \boldsymbol{z}_2 = (\boldsymbol{0}, \boldsymbol{y}_2[0\colon N/2]), \tag{40}$$

and finally we obtain a single segment of time-domain data $\boldsymbol{z}$:

$$\boldsymbol{z} = \boldsymbol{z}_0 + \boldsymbol{z}_1 + \boldsymbol{z}_2. \tag{41}$$

In the above expressions, $\boldsymbol{0}$ represents an array of zeros with length $N/2$, used to pad the segments, whereas the bracket notation stands for `python` array slicing.

The splicing procedure can introduce a bias in the simulated power spectrum, due to the spectral properties of the window that is applied. However, the `simulator` module was tested for several values of the spectral index of a PL signal PSD, ranging between -3 and 3, yielding a correct injection for these spectral indices. The user should nevertheless exercise caution when using the `simulator` module and be aware of the possible introduction of a bias outside the range of tested spectral indices due to splicing.

We show an example of simulated data in Figure 7. The injected signal and noise PSDs are plotted together with the calculated PSD of a simulated data segment. A thorough testing of the `simulator` module is performed in Section 6.1.1.

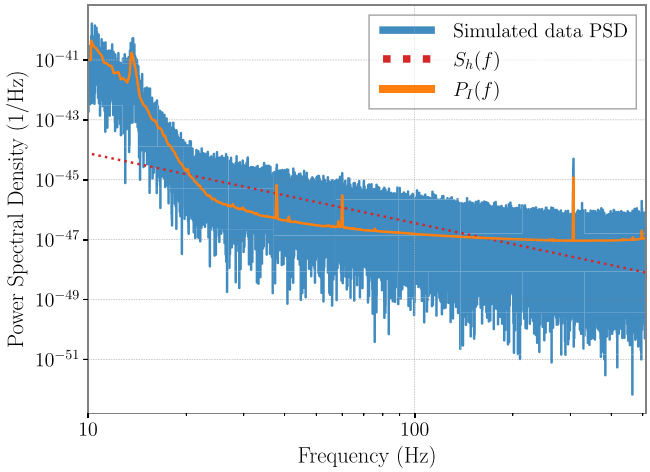

Figure 7. Example of injection using the `simulator` module. The noise PSD (orange) and the injected signal (red) are clearly discernible as parts of the PSD of the simulated data (blue).

4. Manager Objects

`pygwb` counts three manager objects the user can interface with: `Interferometer`, `Baseline`, and `Network`, which are defined in the `detector`, `baseline`, and `network` modules, respectively. Each object is in charge of storing and saving relevant data, and handles data analysis internally. The manager objects are designed such that the user need never call a method from a module directly, but rather will invoke the manager, which queries the relevant module to perform the calculation. For details on how to use these objects, see the complete set of tutorials in the `pygwb` online documentation (Renzini et al. 2023).

4.1. `Detector`

The `detector` module is designed to organize the data products related to a GW detector and provides functions to create and process its internal data. It is formally defined as a subclass of the `Bilby Interferometer` class (Ashton et al. 2019). In what follows, we describe the additional features we have developed, and refer the reader to the `Bilby` documentation for the built-in properties inherited from the parent class.

By default, the `Interferometer` object is initialized by taking geometrical information of a GW detector, such as `latitude`, `longitude`, and `elevation`, as arguments.

```
from pygwb import detector
my_detector = detector.Interferometer
(□args, **kwargs)
```

While the above method allows the user to customize the detector's specification, one can initialize the object based on existing GW detectors, as done in `bilby`'s `Interferometer` class, by calling the `get_empty_interferometer` method.

Once initialized, this object provides several ways to read in and process time-series data, all of which internally call the `preprocessing` module, using information such as a channel name to query the data or pointing to a `numpy` array or a `gwpy` object directly. Additionally, the `Interferometer` object includes processing methods relying on the

spectral module described in Section 3.2 to calculate naive and averaged spectrograms from the stored time-series data.

A pair of Interferometer objects can be used to initialize a Baseline object, as described below. These are then imported as attributes of the Baseline object and store data products specific to each detector.

4.2. Baseline

The Baseline module is by design the core of the pygwb stochastic analysis. Its main role is to manage the cross-correlation between Interferometer data products, combine these into a single cross-spectrum, which represents the point estimate of the analysis, and calculate the associated error, as introduced in Section 2.

The standard initialization of a Baseline object simply requires a pair of Interferometer objects.

```
from pygwb import baseline
H1H2_baseline = baseline.Baseline("H1-H2,"
H1, H2)
```

Here, H1 and H2 are Interferometer objects. It is also possible to load a previously stored Baseline object in pickle format by calling the relevant class method.

The data loaded into the Interferometer objects are automatically imported into the Baseline object upon initialization. The Baseline object relies on the spectral module to calculate cross-correlations between the data streams, following the methodology shown in Section 3.2. Similarly, it relies on the postprocessing module to obtain the point estimate $\hat{\Omega}^{\alpha}_{\rm ref}$ and its variance $\sigma^{\alpha}_{\rm ref}$, as described in Equations (22)–(23). The user may choose to calculate point estimate and sigma spectra or point values; in the latter case, the spectra are automatically stored to facilitate subsequent analyses.

Calculating $\hat{\Omega}^{\alpha}_{\rm ref}$, as well as performing parameter estimation on the GWB spectrum, requires the two-detector ORF, γ_{IJ}, shown in Equation (7). The ORF is calculated at Baseline object initialization, then stored as an attribute. By default, we assume GR, which presents two independent degrees of freedom for the strain field, typically $A = \{+, \times\}$ in the transverse-traceless gauge. For a precise derivation of this function and detector response definitions, see, for example, Romano & Cornish (2017).

The Baseline object is also equipped to probe circularly polarized backgrounds (Seto & Taruya 2007) and non-GR polarizations in the GWB, such as scalar and vector backgrounds (Callister et al. 2017). This requires selecting a different choice of A, according to the chosen polarization type, which can be declared when calculating $\hat{\Omega}^{\alpha}_{\rm ref}$ or the ORF directly. Details on the expressions for non-GR γ_{IJ} functions may be found in the appendix of Callister et al. (2017).

4.3. Network

The network module is designed to handle two different tasks. Its primary purpose is to combine results from different Baseline objects. Similar to the Baseline object, the Network object imports Baseline objects as attributes that may be invoked through the Network. In addition to this functionality, it can also be used to simulate cross-correlated data across a network of detectors. Both signal-only and signal and noise data can be simulated using a Network object. The network module handles all data generation by querying the simulator module.

The Network object can be initialized in two ways. By default, it is initialized through a list of Interferometer objects.

```
from pygwb import network
HLV_network=network.Network('HLV',    [H1,
L1, V1])
```

It is also possible to initialize a Network using a list of Baseline objects, to streamline the combination of results from different baselines that already contain final data products.

```
HLV_network=network.Network.from_base-
lines('HLV',  [HL_baseline,  HV_baseline,
LV_baseline])
```

The combined point estimate and sigma spectra are stored as attributes of the Network. These are combined by performing an inverse-noise-weighted average over the individual baseline final spectra, assuming the data are uncorrelated between baselines, i.e., assuming each baseline provides independent information. This is a valid approximation when working in the large noise limit. Further details can be found in Allen & Romano (1999).

The Network is also designed to produce appropriately correlated simulated data for a network of interferometers. The Network can either simulate data from scratch, or add simulated data to pre-existing data, if the interferometers used to initialize the object contain strain data. The latter is simply done, as strain adds coherently in the time domain. This functionality relies on the simulator module, which performs the data simulation, as discussed in Section 3.7.

5. Analysis Pipeline

We present an overview of the package analysis scripts, which combine the various modules into a GWB analysis pipeline. The pipeline has several default values that may be changed according to the user's requirements. However, we note that thanks to the flexibility of the pygwb package, one can also easily construct an ad hoc pipeline.

5.1. pygwb_pipe

The core script of our analysis suite, pygwb_pipe, is designed to carry out the bulk of the stochastic analysis. It combines the pygwb modules in order to go from the unprocessed data to the optimally averaged $\hat{\Omega}^{\alpha}_{{\rm ref},f}$ and $\sigma^{\alpha}_{{\rm ref},f}$ spectra for a single baseline. To read in the analysis parameters, pygwb_pipe interfaces with the parameters module, specifically designed to handle the analysis parameters, either passed through an initialization file (param_file) or declared in the command line. The module includes the Parameters data class, which stores the chosen parameters. The pipeline may be run from the command line as follows:

```
pygwb_pipe --param_file path_to_param_file
```

All param_file parameters may be alternatively passed from the command line directly. If a mixture of parameter file and command line parameters are passed, the latter will override their corresponding values stored in the parameter file. Additionally, a set of pipeline-specific parameters may be passed from the command line for ease of use, such as whether

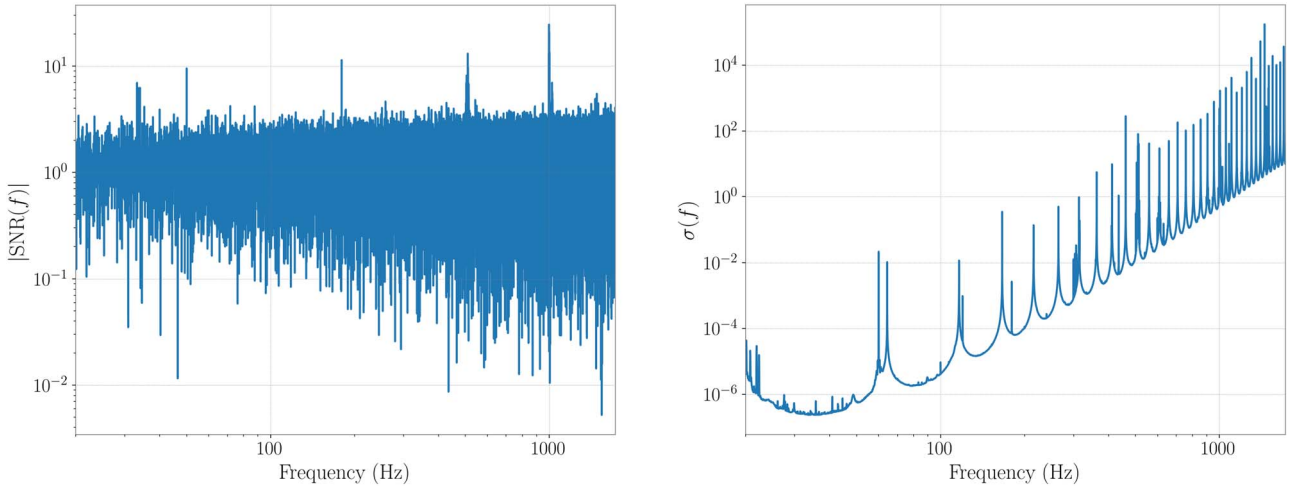

Figure 8. Left: absolute value of the SNR spectrum as a function of frequencies. Right: sigma spectrum as a function of frequency.

to apply data quality cuts. A full list of parameters and their descriptions may be found in Table 1.

After reading in the parameters, two `Interferometer` objects are created accordingly, and data are loaded in and preprocessed using the `preprocessing` module. Depending on the value of the `gate_data` parameter in the initialization file, the gating outlined in Section 3.1 also takes place at this stage. Subsequently, a baseline object is created using the pair of interferometer objects. Recall that the `baseline` module plays a central role in the pipeline and handles the computation of the various quantities of interest, including the (average) PSDs and CSDs of the baseline, relying on the `spectral` module. This is described in more detail in Sections 3.2 and 4.2.

The delta-sigma cut is then performed, and optimally averaged spectra and an overall point estimate are calculated with the relevant `Baseline` methods. The delta-sigma cut is applied by default, but may also be calculated and applied at a later stage. Finally, the spectra, the overall point estimate, and the pickled baselines (if requested) are saved as the output. By default, the output is in `numpy` binary file format, `npz`.

In realistic scenarios, we analyze year-long data sets, so running `pygwb_pipe` in series is suboptimal. However, a long data set can be split into smaller jobs and parallelized on a cluster. The output of each job is then combined into a single set of result spectra $\hat{\Omega}^{\alpha}_{\mathrm{ref},f}$ and $\sigma_{\mathrm{ref},f}$, using the `pygwb_combine` script. The latter simply takes a weighted average over all jobs, assuming each job is an independent measurement of the signal. At this stage, it is possible to implement final postprocessing choices, such as reweighting the spectra to a desired α and f_{ref}, as well as changing the default Hubble constant H_0 at which results are reported.

Details on running the pipeline and combination scripts may be found in our online documentation (Renzini et al. 2023).

5.2. `statistical_checks`

With the `statistical_checks` module, we provide a tool to perform initial statistical analyses of a `pygwb` run result set and to visualize them in preformatted plots. We identify five broad categories of checks.

The first set calculates the running point estimates for $\hat{\Omega}^{\alpha}_{\mathrm{ref}}$ and $\sigma^{\alpha}_{\mathrm{ref}}$ quantities as a function of time, as more data segments are added to the analysis. The values of α and f_{ref} are those used in the analysis and may not be changed at this point. The running averages are cumulative weighted averages of time-ordered segments, and do not take segment-by-segment correlation into account. In the case of a detection, these converge to a biased point estimate and σ, as proper postprocessing is not applied (see Section 3.3). However, the visualization of running quantities is extremely useful to identify trends in the data, and ultimately will flag a possible detection. The module also provides a linear trend analysis, fitting the evolution of the parameters described above as a function of time.

The second set focuses on the SNR spectrum as a function of frequency, defined as

$$\mathrm{S}/\mathrm{N}_f = \frac{\hat{\Omega}^{\alpha}_{\mathrm{ref},f}}{\sigma^{\alpha}_{\mathrm{ref},f}}. \tag{42}$$

The absolute value, real, and imaginary parts of the SNR are calculated, as well as the cumulative SNR. An example of these plots, using the first subset of O3 data further described in Section 6.2, is given in Figure 8. These plots are a faithful representation of the "noisiness" of each frequency bin and how much each bin contributes to the analysis.

The third set of checks produces the IFT of the point estimate spectrum, which should peak around zero seconds in the case of a detection. Time-shifting the data in two detectors by more than the coherence time between the two detectors breaks the coherence between the two data streams, removing any evidence of a GWB signal. Note that the coherence time is determined by the bandwidth of our signal, which is of order 100 Hz, resulting in a coherence time of 10 ms. Hence, a GWB signal will only peak around zero time lag between the outputs of the detectors.

The fourth set studies the effect of the $\Delta\sigma$ data quality cut described in Section 3.4 on the analysis run. To this end, we display several quantities before and after the cut is applied to the data, including the segment values of $\hat{\Omega}^{\alpha}_{\mathrm{ref},i}$, $\sigma^{\alpha}_{\mathrm{ref},i}$, and $\Delta\sigma^{\alpha}_{\mathrm{ref},i}$, and the deviations in SNR,

$$\Delta\mathrm{S}/\mathrm{N}_i = \frac{\hat{\Omega}^{\alpha}_{\mathrm{ref},i} - \langle\hat{\Omega}^{\alpha}_{\mathrm{ref}}\rangle}{\sigma^{\alpha}_{\mathrm{ref},i}}, \tag{43}$$

as a function of time. Here, angle brackets indicate an arithmetic mean over all segments i. We also plot a histogram

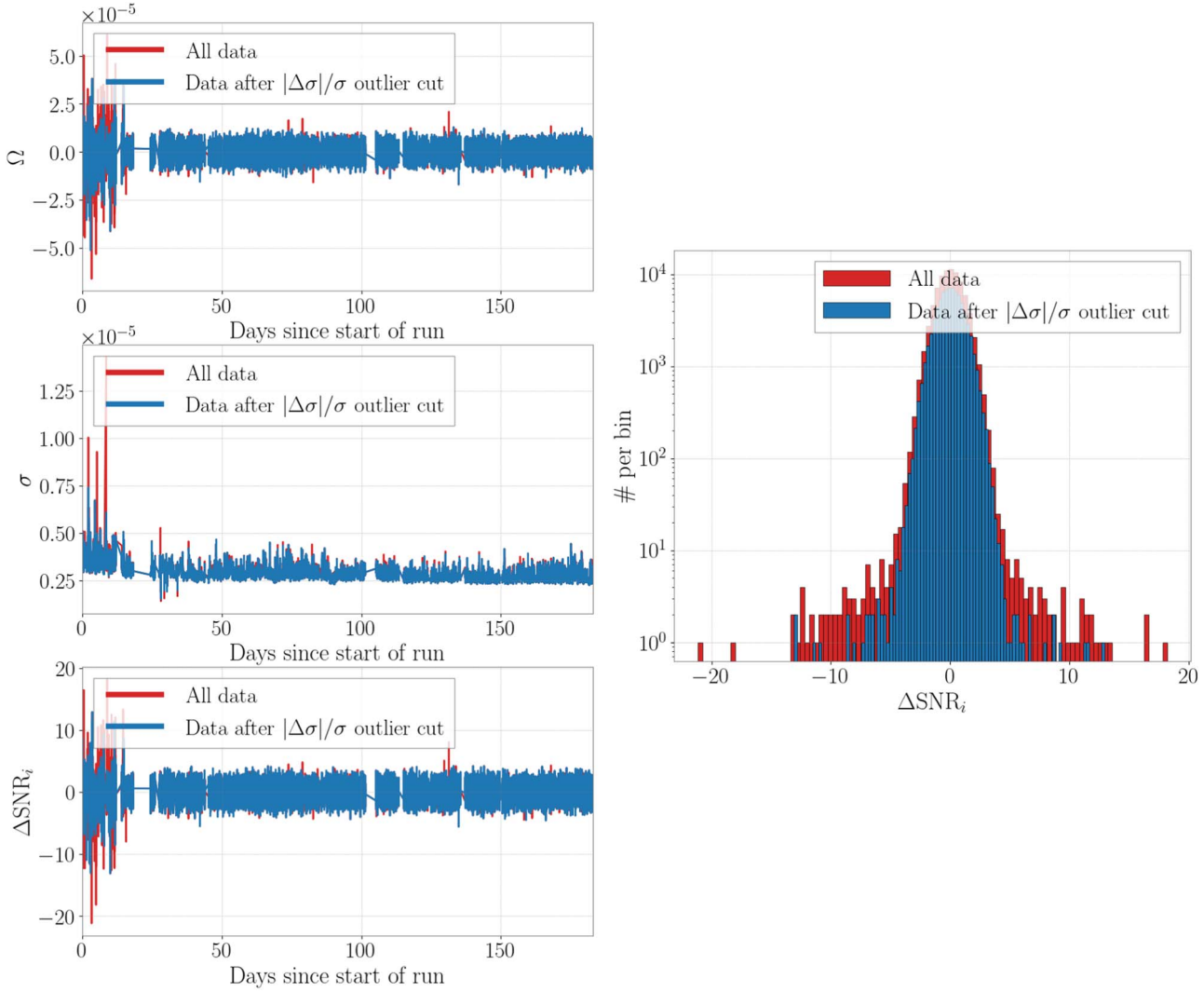

Figure 9. Left: point estimate, sigma, and deviations in $\Delta\mathrm{SNR}_i$ as a function of time before the delta-sigma cut (red) and after the cut (blue). Right: distribution of the deviations in $\Delta\mathrm{SNR}_i$ as a function of time before the delta-sigma cut (red) and after the cut (blue).

of the values of ΔSNR before and after the cut. This distribution should be centered around 0, with a smooth narrower distribution after the application of the $\Delta\sigma$ cut. We additionally plot the ΔSNR as a function of individual $\sigma^{\alpha}_{\mathrm{ref},i}$. Finally, we plot the distribution of the ratios $\sigma^2_{\mathrm{ref},i}/\langle\sigma^2_{\mathrm{ref},i}\rangle$, which should peak around 1. Some representative plots are shown as an example in Figure 9.

The last set of checks concerns a Kolmogorov–Smirnov (KS) test that is used to verify that the $\Delta\mathrm{SNR}_i$ are consistent with a Gaussian distribution. The KS test implementation of this module returns the KS test statistic, which is the maximal deviation from the Gaussian cumulative distribution function, as well as the p-value. These values can be used to make statements about the Gaussianity of the data (Dodge 2008). In addition, the cumulative distribution function is plotted for the data as well as for a Gaussian distribution.

6. Testing

To comprehensively test the `pygwb` analysis suite, we employ an efficient workflow to analyze data sets of increasing complexity. The data sets considered in this paper are as follows:

1. *Continuous SGWB.* A loud stationary and continuous stochastic signal generated with the `simulator` module, injected in Advanced LIGO Hanford and LIGO Livingston, assuming design A+ sensitivity (Barsotti et al. 2018).
2. *Realistic compact binary coalescence (CBC) GWB.* A realistic background of merging binary black holes (BBHs) and binary neutron stars (BNSs), injected in Advanced LIGO Hanford and LIGO Livingston, assuming design A+ sensitivity.
3. *O3 data set.* The full Advanced LIGO Hanford and LIGO Livingston data set from the third LVK observing run (Abbott et al. 2021).

The continuous SGWB (data set 1) is an idealized observing scenario, as our stochastic model matches the target signal perfectly by design, and as the signal is stationary and continuous, our approach is optimal (Drasco & Flanagan 2003; Lawrence et al. 2023). The CBC background (data set 2) is a realistic scenario where the target signal is generated according to astrophysical models, informed by GW detections. In this case, the signal is non-Gaussian, and we expect our approach to be unbiased (Regimbau et al. 2012, 2014; Meacher et al. 2015)

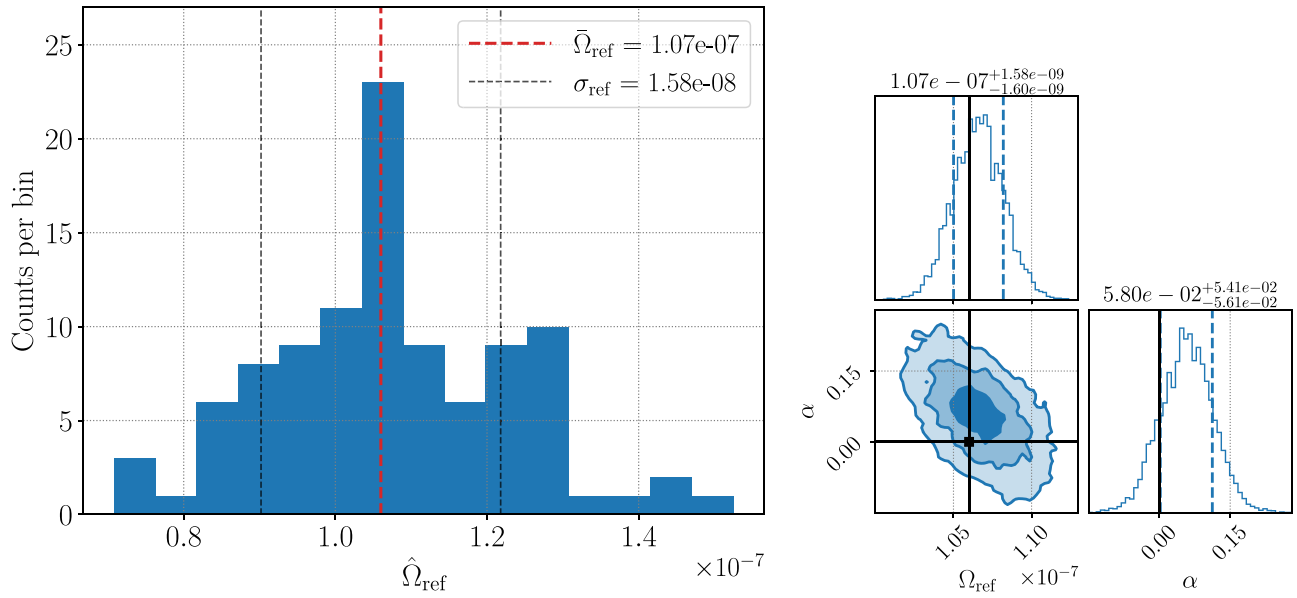

Figure 10. Left: distribution of the recovered point estimate for each day in the data set. The injected value is denoted by the red line. $\bar{\Omega}_{\rm ref}$ and $\sigma_{\rm ref}$ denote the mean and the standard deviation of the 100 point estimates. Right: parameter estimation performed on the 100 days, obtained assuming a log-uniform prior from 10^{-11}–10^{-6} for $\Omega_{\rm ref}$ and a Gaussian prior with mean 0 and standard deviation 1.5 for α. The injected values are denoted by the black lines, while the contours represent the 1σ, 2σ, and 3σ contours. The vertical dashed lines represent the 1σ confidence interval.

but suboptimal (Drasco & Flanagan 2003; Lawrence et al. 2023), due to the intermittent nature of the signal, which is not taken into account in the search method. For more details on the time-domain characteristics of these two types of signals and the detection challenges these present, see, for example, (Regimbau 2022). Finally, the O3 Advanced LIGO data set (data set 3) presents all the complexity of analyzing real GW detector data, including nonstationary noise, a large data volume, and expensive computational requirements.

We handle large data sets by splitting the data into smaller `pygwb_pipe` jobs, assuming each job is independent; these are then combined using the `pygwb_combine` script (see Section 5 for details). We then employ a parameter estimation script, `pygwb_pe`, based on the `pe` module described in Section 3.6, to perform parameter estimation on specific models. For more details on how to run this sort of analysis, we refer users to the online documentation for the most up-to-date workflow instructions (Renzini et al. 2023). In the following, we present the different data sets and summarize our analysis results.

6.1. Mock Data

6.1.1. Stationary and Continuous

We employ the `Network` (Section 4.3) to generate a stationary and continuous SGWB signal with a fixed PSD, $S_h(f)$. This allows us to simultaneously test the module and the whole analysis pipeline. The injected SGWB is scale-invariant, i.e., $\Omega_{\rm GW}(f)$ is constant over frequencies

$$\Omega_{\rm inj}(f) = 1.06 \times 10^{-7}. \tag{44}$$

This is converted to $S_h(f)$ using the relation in Equation (2). The noise $P_n(f)$ is taken to be Gaussian, colored using the the Advanced LIGO noise PSD (Aasi et al. 2015a). 100 days of consecutive data are simulated at a sampling rate of 1024 Hz.

Each of the 100 days is analyzed separately, and we recover a distribution of $\hat{\Omega}^0_{25}$ point estimates, shown in Figure 10 (left), using $\alpha = 0$ and $f_{\rm ref} = 25$ Hz in the pipeline. Analyzing 100 days separately allows us to construct a distribution of recovered point estimates, which is useful to assess the ability of the `simulator` module to inject a stochastic stationary signal. We then perform parameter estimation on the combined 100 days, presented in Figure 10 (right). We assume a log-uniform prior from $10^{-11} - 10^{-6}$ for $\Omega_{\rm ref}$ and a Gaussian prior with mean 0 and standard deviation 1.5 for α. This shows a recovery within 1σ for $\Omega_{\rm ref} = 1.06 \times 10^{-7}$ and within 2σ for the spectral index $\alpha_{\rm inj} = 0$.

The tests above illustrate that the `simulator` module is able to successfully inject a stochastic stationary signal and that the `pygwb` pipeline is able to recover this injection.

6.1.2. Gravitational-wave Background from a Coalescing Compact Binary Population

The inspiral and merger of two compact objects emit a characteristic GW signal. We generate data sets containing a GWB signal resulting from the superposition of GW signals from a set of CBC populations including BBHs and BNSs. To simulate the signals, we employ the code used in the past for the Einstein Telescope (ET) mock data and science challenges (MDSCs; Regimbau et al. 2012, 2014; Meacher et al. 2016) and for the Advanced LIGO and Advanced Virgo MDSC (Meacher et al. 2015). The Monte Carlo algorithm that we use for the generation of a compact binary population up to redshift $z = 10$ is extensively described in Regimbau et al. (2012) and Regimbau et al. (2015). We summarize below the main steps of the simulations.

To generate a CBC population, we assume a merger rate per unit redshift (Belczynski & Kalogera 2001; Bulik et al. 2004; Belczynski et al. 2006; Berger et al. 2007):

$$\frac{dR(z)}{dz} = \frac{dV_c}{dz} r_c(z), \tag{45}$$

where dV_c/dz is the comoving volume element and r_c the coalescence rate as a function of redshift (Regimbau 2011).

Table 2
Parameters and Results of Each Data Set

DATA SET	τ (s)	a	$(\hat{\Omega}_{25}^{2/3} \pm \hat{\sigma}_{25}^{2/3}) \times 10^9$	SNR	$\mathcal{B}^{\rm GW}_{\rm noise}$
PLPP	60	1.5	2.09 ± 0.39	5.4	11.1
PL	54.7	1.7	1.94 ± 0.39	5.0	9.2

Note. The first row refers to the PLPP data set, while the second row refers to the PL one. Column (2): the average time between two successive binary mergers, τ. Column (3): the waveform amplification factor, a. Column (4): the recovered point estimate with 1σ uncertainty on the quantity $\hat{\Omega}^{\alpha}_{\rm ref}$ ($f_{\rm ref} = 25$ Hz, $\alpha = 2/3$). Column (5): the corresponding SNR. Column (6): the log-Bayes factor $\mathcal{B}^{\rm GW}_{\rm noise}$.

The element of comoving volume assumes a Lambda cold dark matter (ΛCDM) cosmology from Planck 2018 (Aghanim et al. 2020; Hubble parameter $H_0 = 67.7\,{\rm km\,s^{-1}\,Mpc^{-1}}$, $\Omega_m = 0.31$, and $\Omega_\Lambda = 1 - \Omega_m$). We assume a coalescence rate normalized to a local rate $r_c(0) = 1\,{\rm Mpc^{-3}\,Myr^{-1}}$ for BNS coalescences and $r_c(0) = 3\,{\rm Mpc^{-3}\,Myr^{-1}}$ for BBH coalescences, assuming the star formation rate from Hopkins & Beacom (2006) and a minimum delay time between binary formation and merger of 20 Myr for BNSs and 50 Myr for BBHs; see Dominik et al. (2012) and Neijssel et al. (2019) for more details. These choices give rise to a data set composed of 87% of BNSs and 13% of BBHs.

The time intervals τ between consecutive CBC events in our population are obtained by sampling an exponential distribution $P(\tau) = \exp(-\tau/\bar{\tau})$, where $\bar{\tau}$ is the average time between consecutive events. This is consistent with the assumption that the coalescence times t_c of the events behave as a Poisson process (Regimbau et al. 2015). The coalescence redshift is drawn from the normalized coalescence rate $p(z) = \bar{\tau} dR/dz(z)$ within $z \in [0,\ 10]$. The sky position $\hat{n}$ of each source is generated isotropically on the sky. The GW polarization angle ψ, the phase angle ϕ_0 at the coalescence time, and the cosine of the inclination angle of the orbital plane to the line of sight ι are all drawn from uniform distributions. The mass function of the components in the BBHs is chosen to be a PL plus peak (PLPP) from the preferred case presented in the LVK collaboration CBC population inference paper (Abbott et al. 2023) or a simple PL (Abbott et al. 2021c), while the BNS masses are drawn from a uniform distribution between 1 and 3 $M_\odot$. The BBH mass functions are used to label the two data sets presented below. Spins are neglected in both cases.

For each source, the signal waveform is generated in the time domain. For BNSs, we use the `TaylorT4` time-domain waveform (Buonanno et al. 2003). For BBH signals, we use the `EOBNRv2` (Buonanno et al. 2009) time-domain waveform from numerical relativity. These are then injected into the LIGO Hanford and Livingston detectors, with the addition of colored Gaussian noise generated from the LIGO A+ design (Barsotti et al. 2018; O'Reilly et al. 2020) expected sensitivity curve, to produce the final data sets.

Following the above prescriptions, we generate two 6 months data sets with sampling frequency 1024 Hz, labeled PLPP and PL, formed by two different CBC populations. The two populations differ by the mass distributions of the BBHs and the average times between consecutive events, as seen in Table 2. The latter are chosen such that the GWB amplitudes of the two data sets match, for ease of comparison. Furthermore,

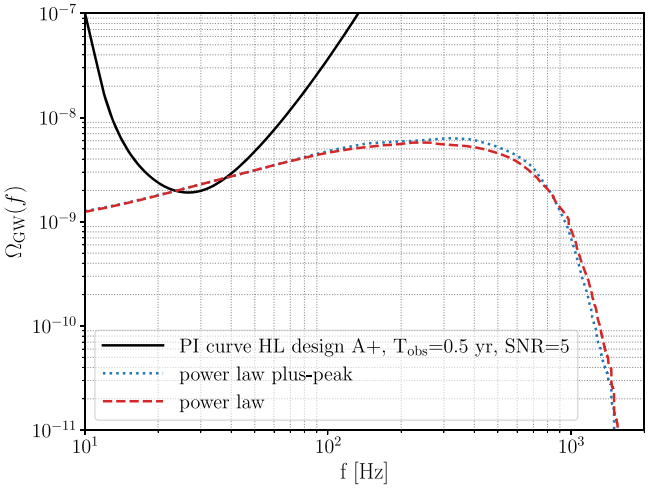

Figure 11. $\Omega_{\rm GW}(f)$ for the data set corresponding to the PLPP (blue) and the PL (red) models for the mass function. The black line is the PI curve for an observation time of 6 months and an expected SNR = 5, assuming the Hanford–Livingston baseline with Advanced LIGO plus design sensitivity. The simulated signals intersect the PI curve, hence they are expected to be detected with an SNR of at least 5.

to obtain an SNR large enough to confidently detect the injected GWB, a small amplification of the signal is required. To this end, the amplitudes of the CBC waveforms are multiplied by 1.5 and 1.7 for the PLPP and the PL data sets, respectively, resulting in an injected value of $\Omega_{\rm ref} = 2.05 \times 10^{-9}$.

The $\Omega_{\rm GW}(f)$ spectrum relative to each data set is obtained by summing the contributions from individual coalescences (Meacher et al. 2015), and is illustrated in Figure 11. As may be observed, in the case of CBC signals, $\Omega_{\rm GW}(f)$ increases as $f^{2/3}$ from the inspiral phase (and then as $f^{5/3}$ from the BBH merger phase), before reaching a peak and steeply decreasing (Marassi et al. 2011). This motivates fixing the spectral index parameter to $\alpha = 2/3$ in our searches. Figure 11 also shows the PL integrated sensitivity (PI) curve (Thrane & Romano 2013) for the Hanford–Livingston baseline, assuming the design A+ sensitivity for the two detectors (Barsotti et al. 2018), an observation time $T_{\rm obs} = 6$ months, and a desired sensitivity of SNR = 5. Given that the PI curve is almost tangent to $\Omega_{\rm ref}$ of the two data sets, we expect to observe the GWB signals with SNR $\sim$5.

We analyze the two data sets in the frequency band 20–500 Hz, using a frequency resolution of 1/32 Hz and a segment duration of 192 s. We choose $\alpha = 2/3$, $f_{\rm ref} = 25$ Hz, and $H0 = 67.7$ km s^{-1} Mpc^{-1} for this analysis. The results of the analysis are summarized in Table 2. We recover the PLPP injection within 1σ, and observe it with SNR= 5.4, while recovering the PL injection within 1σ, with SNR = 5.0. We attribute the differences in the recoveries to the specific data and noise realizations within the data sets.

We then proceed with estimating the parameters α and $\Omega^{\alpha}_{\rm ref}$, modeling $\Omega_{\rm GW}(f)$ as a simple PL in frequency, as given by Equation (9). We assume a log-uniform prior over $\Omega^0_{\rm ref}$, in the range $[\Omega^0_{\rm min},\ \Omega^0_{\rm max}] = [10^{-11},\ 10^{-8}]$, and a Gaussian prior on α with mean $2/3$ and standard deviation $(\log_{10}\Omega^0_{\rm min} - \log_{10}\Omega^0_{\rm min})/2 = 1.5$. Note that the priors in $\Omega^{\alpha}_{\rm ref}$ are defined for $\alpha = 0$. The choice of the prior over α can be understood as follows. The log-uniform prior over $\Omega^0_{\rm ref}$ induces some implicit

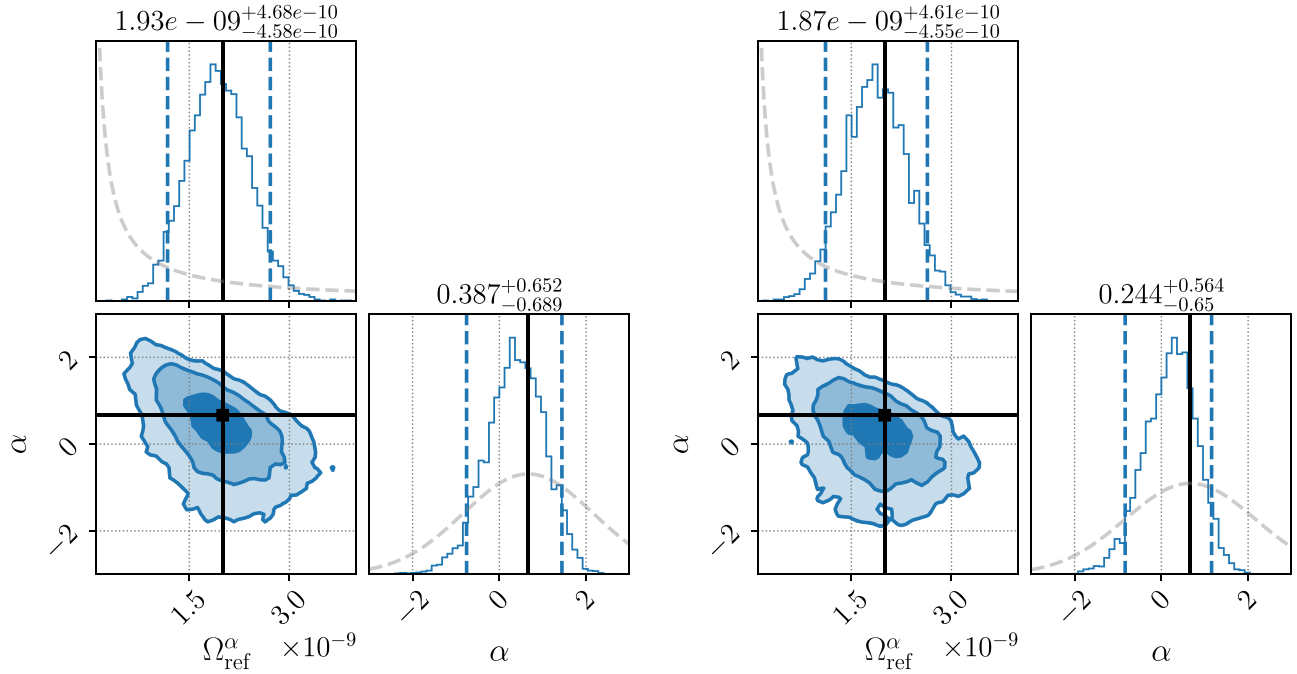

Figure 12. Parameter estimation results. Left: corner plot obtained from running the parameter estimation over the PLPP data set. Right: corner plot obtained from running the parameter estimation over the PL data set. Each plot shows the posteriors on $\Omega_{\rm ref}$ and α obtained assuming a log-uniform prior on $\Omega^0_{\rm ref}$ from 10^{-11}–10^{-8} and a Gaussian prior on α with mean 2/3 and standard deviation of 1.5, respectively, denoted by the gray dashed lines. The injected values are represented by the black lines, indicating recoveries of both the amplitude of the signal and α within 1σ. The vertical blue dashed lines represent the 2σ confidence intervals.

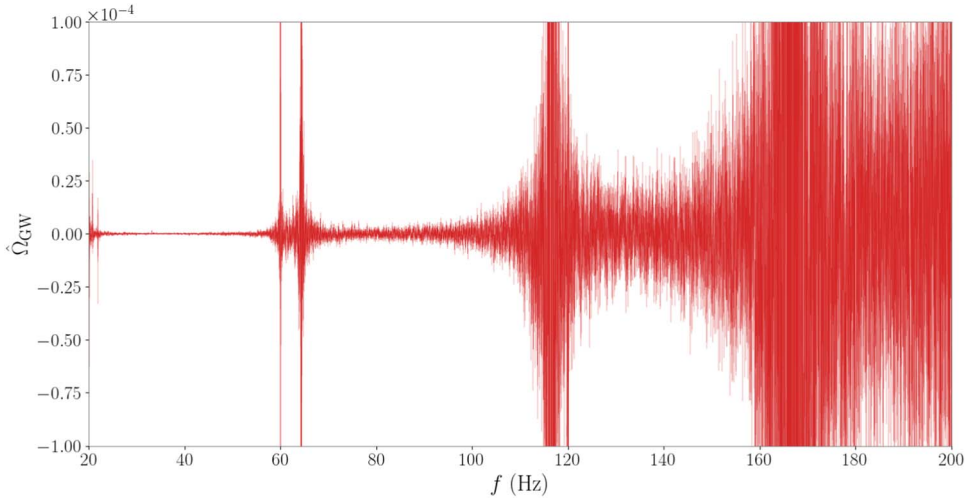

Figure 13. Estimated cross-correlation spectrum $\hat{\Omega}^0_{25} \pm \hat{\sigma}^0_{25}$ from O3 data. By eye, it is possible to spot several narrowband artifacts (lines) that are subsequently excluded from our analysis.

prior over α that can be shown to be a triangular prior centered on $\alpha = 0$ and nonzero for $|\alpha| \leqslant (\log_{10}\Omega^0_{\rm max} - \log_{10}\Omega^0_{\rm min})$. To avoid a vanishing prior outside this range, we choose a Gaussian prior for α with standard deviation comparable with the triangular prior, centered on $\alpha = 2/3$ to better match the injected GWB.

Parameter estimation corner plots are shown in Figure 12. For both data sets, $\Omega_{\rm ref}$ and α are recovered within 1σ. The log-Bayes factors $\mathcal{B}^{\rm GW}_{\rm noise}$ are 11.1 and 9.2 for the PLPP and PL data sets, respectively, indicating strong evidence (Kass & Raftery 1995) for the presence of signal over noise only.

6.2. O3

In this section, we present results from the application of the `pygwb` analysis suite to the full LIGO Hanford and LIGO Livingston O3 data set (LIGO Scientific Collaboration et al. 2021). We set upper limits on the signal from an SGWB and confirm these are consistent with previously published collaboration results (Abbott et al. 2021).

The O3 data run collected between 2019 April 1 and 2020 March 27, divided into two subsets, with an interruption between 2019 October 1 and November 1, with a total coincident livetime of 205.4 days between LIGO Hanford and

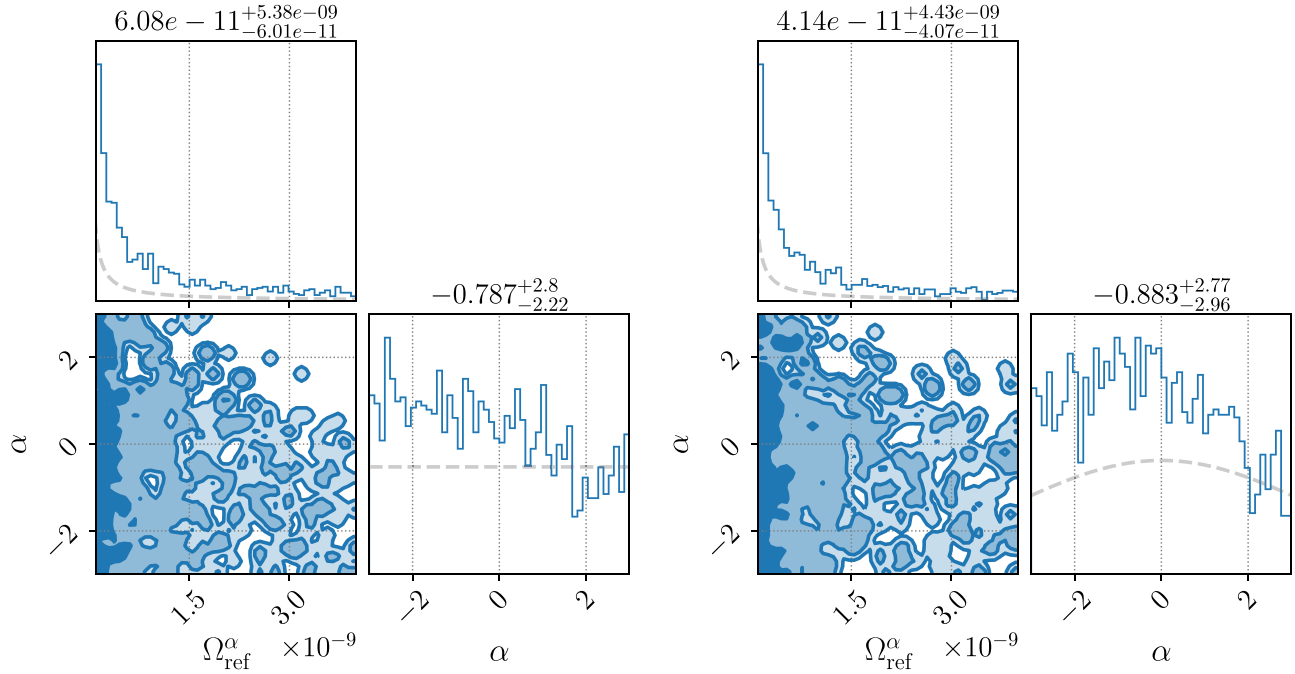

Figure 14. Parameter estimation results with `pygwb_pe` on LVK O3 data, using a log-uniform prior on Ω_{25} and a uniform prior (left) or a Gaussian prior on α (right), as described in the text. The priors are denoted by the gray dashed lines.

Table 3
Summary of `pygwb` Search Results on O3 Data Set

α	$\hat{\Omega}^{\alpha}_{25} \times 10^9$	$\hat{\Omega}_{\rm LVK} \times 10^9$	Prior	$\Omega_{\rm pe}$ (95% UL)	$\alpha_{\rm pe}$
0	-3.4 ± 8.1	-2.1 ± 8.2	UNIFORM	5.44×10^9	$-0.8^{+2.8}_{-2.2}$
2/3	-4.5 ± 6.1	-3.4 ± 6.1	GAUSSIAN	4.06×10^9	$-0.5^{+2.8}_{-2.8}$
3	-1.5 ± 0.9	-1.3 ± 0.9			

Note. On the left, three columns summarizing the point estimates from the weighted optimal statistic, at different spectral indices α. On the right, three columns summarizing the Bayesian upper limits (ULs), with a log-uniform prior on Ω^0_{25} and either a uniform or Gaussian prior on α. These results are consistent with no detection of the amplitude of the background ($\Omega^{\alpha}_{\rm ref}$ is consistent with 0), nor its spectral shape (α remains unconstrained).

LIGO Livingston. These are reduced to 196 days after *category 1 vetoes*[38] (Abbott et al. 2018; Acernese et al. 2022a) and external nonstationarity cuts are applied (for details, see Abbott et al. 2021). The `pygwb` analysis is implemented with the workflow described above. The O3 data, natively sampled at 16,384 Hz, are downsampled to 4096 Hz and high-pass-filtered at 11 Hz.

The time-averaged O3 LIGO Hanford–Livingston cross-correlation spectrum is presented in Figure 13. Our $\Delta\sigma$ threshold excludes 7.8% of the analyzed time (see Section 3.4 for implementation details). This result matches the previous stochastic nonstationarity cut published in Abbott et al. (2021) within 1%, with the previous cut excluding an extra 0.06% of the time. We believe this small variation to be due to a different window bias factor being used in the two analyses (the bias factor calculation used here is outlined in Appendix A).

[38] "Category 1" vetoes flag data that are unsuitable for analysis, such as incorrectly calibrated data, data collected during atypical operations of the instruments, and data with severe data quality issues.

We calculate broadband integrated estimates between 20–1726 Hz of $\Omega_{\rm GW}(f_{\rm ref})$ for different PL spectral models, applying the released O3 notch list (LIGO Scientific Collaboration et al. 2021) to exclude known problematic frequencies (Covas et al. 2018). A summary of the values for the point estimates and the uncertainties for these are presented in Table 3. The uncertainties $\sigma^{\alpha}_{\rm ref}$ agree within 1% with previously published LVK results, presented in Abbott et al. (2021). The point estimates for $\Omega^{\alpha}_{\rm ref}$ fluctuate notably more than the uncertainties. We believe this to be due to small differences in the analyses, to which the point estimates are more sensitive, such as the individual start and end times of each pipeline job and the differences in the nonstationarity cuts described above.

Finally, we perform parameter estimation to constrain $\Omega_{\rm GW}(f_{\rm ref} = 25{\rm Hz}) \equiv \Omega_{25}$ and the spectral index α with O3 data. We employ a log-uniform prior on Ω_{25} spanning $[10^{-13}, 10^{-6}]$, and present results for two different priors on α: a uniform prior between $[-4, 4]$ and a Gaussian prior centered around 0 with norm 3.5 (the latter matches the choice in Abbott et al. 2021). To account for calibration uncertainty, we marginalize over the uncertainty parameter λ, as described in Appendix B, with a combined calibration error for Hanford and Livingston of 1.48%, as in Abbott et al. (2021). Parameter estimation confirms Ω_{25} is consistent with 0 and α remains unconstrained, as may be seen in Figure 14. These results agree with the previous parameter estimation carried out in Abbott et al. (2021).

Our results are quoted at the value of the Hubble parameter $H_0 = 67.9\ {\rm km\ s^{-1}\ Mpc^{-1}}$, in line with published results. This is not the built-in value of H_0, defined in Section 3.3; however, rescaling is straightforward, as it is an overall multiplication factor, which may be changed when postprocessing the run with the `pygwb_combine` script or manually using the built-in functions of the `OmegaSpectrum` object, as explained in Sections 3.3 and 5.

We would like to note that this entire analysis was carried out on a large computing cluster and completed in less than 5 hr of human time. This is an example of the computational efficiency of our package.

7. Conclusions

We present a new Python-based package tailored to GWB searches with current ground-based interferometers. We opt for a modular code, where each module performs specific tasks of the GWB data analysis. The modularity of the code results in large flexibility and offers the possibility of customizing the pipeline according to one's own needs. With the use of the Python language, and the user-friendliness and flexibility of the code, we aim to bring GWB searches to the wider GW community, as the detection of a GWB with ground-based interferometers draws potentially closer. With increasing amounts of GW data, `pygwb` also answers the need for an open-source GWB data analysis tool.

In this paper, we show the application of the `pygwb` package to mock data sets, illustrating how the various modules can be assembled to form a search pipeline, and showing what a GWB detection could look like with our analysis approach. To conclude, we run the `pygwb` pipeline on real GW data from the third observing run (O3) of the LVK collaboration, and recover results in agreement with published results. Both analyses serve as a validation of the software.

The `pygwb` package is designed to evolve along the way and address new analysis needs as they arise. This is facilitated by the structure and the format of `pygwb`, and the management of the online Git repository. The `pygwb` team invites input from the broader community, under the form of Git issues and pull requests. New contributors to the code are always welcome. Official updates and releases of the code will be handled and reviewed internally by the software and review teams, which are due to evolve.

We are aware that other analysis methodologies exist, which accommodate different features of specific GWBs, such as potential anisotropy (Ain et al. 2018) and the intermittency of the BBH background (Smith & Thrane 2018; Lawrence et al. 2023). We look forward to interfacing with these methods and, where useful and appropriate, improving the current codebase to support and encompass more analysis schemes.

Finally, we are particularly excited at the prospect of broadening the scope of the package to include support for next-generation detectors such as ET (Maggiore et al. 2020) and Cosmic Explorer (Reitze et al. 2019). While the science cases and design properties of these detectors are still under development, there is evident interest in targeting GWBs with these detectors within the community (Sathyaprakash et al. 2011; Regimbau et al. 2012, 2014), and a notable increase in sensitivity compared to present-day interferometers is expected.

Acknowledgments

The author list of this paper includes, in order: all `pygwb` code authors, in order of successful GitLab merge requests, at the time of writing; code reviewers; and testers, in alphabetical order.

We would like to thank the LVK stochastic group for its continued support. Special thanks to G. Cella and J. Suresh for valuable comments on the manuscript.

A.I.R. is supported by the NSF award 1912594. A.R.R. is supported in part by the Strategic Research Program "High-Energy Physics" of the Research Council of the Vrije Universiteit Brussel and by the iBOF "Unlocking the Dark Universe with Gravitational Wave Observations: from Quantum Optics to Quantum Gravity" of the Vlaamse Interuniversitaire Raad and by the FWO IRI grant I002123N "Essential Technologies for the Einstein Telescope." K.T. is supported by FWO-Vlaanderen through grant No. 1179522N. P.M.M. is supported by the NANOGrav Physics Frontiers Center, National Science Foundation (NSF), award number 2020265. L.T. is supported by the National Science Foundation through OAC-2103662 and PHY-2011865. K.J. is supported by FWO-Vlaanderen via grant number 11C5720N. F.D.L. is supported by an FRIA Grant of the Belgian Fund for Research, F.R.S.-FNRS. J.L. was supported by NSF Award PHY-2207270. D.D. is supported by the NSF as a part of the LIGO Laboratory. A. M. is supported by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 754510. J.D.R. was supported in part by NSF Award PHY-2207270 and startup funds provided by Texas Tech University. V.M. was supported in part by the NSF award PHY-2110238.

This material is based upon work supported by NSF's LIGO Laboratory, which is a major facility fully funded by the National Science Foundation. LIGO was constructed by the California Institute of Technology and Massachusetts Institute of Technology with funding from the National Science Foundation, and operates under cooperative agreement PHY-1764464. Advanced LIGO was built under award PHY-0823459. The authors also gratefully acknowledge the support of the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO 600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. The authors gratefully acknowledge the Italian Istituto Nazionale di Fisica Nucleare (INFN), the French Centre National de la Recherche Scientifique (CNRS), and the Netherlands Organization for Scientific Research (NWO) for the construction and operation of the Virgo detector and the creation and support of the EGO consortium. The authors also gratefully acknowledge research support from these agencies as well as from the Council of Scientific and Industrial Research of India, the Department of Science and Technology, India, the Science & Engineering Research Board (SERB), India, the Ministry of Human Resource Development, India, the Spanish Agencia Estatal de Investigación (AEI), the Spanish Ministerio de Ciencia e Innovación and Ministerio de Universidades, the Conselleria de Fons Europeus, Universitat i Cultura and the Direcció General de Política Universitaria i Recerca del Govern de les Illes Balears, the Conselleria d'Innovación, Universitats, Ciéncia i Societat Digital de la Generalitat Valenciana and the CERCA Programme Generalitat de Catalunya, Spain, the National Science Centre of Poland and the European Union—European Regional Development Fund; Foundation for Polish Science (FNP), the Swiss National Science Foundation (SNSF), the Russian Foundation for Basic Research, the Russian Science Foundation, the European Commission, the European Social Funds (ESF), the European Regional Development Funds (ERDF), the Royal Society, the Scottish Funding Council, the Scottish Universities Physics Alliance, the Hungarian Scientific Research Fund (OTKA), the French Lyon Institute of Origins (LIO), the Belgian Fonds de la Recherche Scientifique

(FRS-FNRS), Actions de Recherche Concert'es (ARC) and Fonds Wetenschappelijk Onderzoek—Vlaanderen (FWO), Belgium, the Paris Île-de-France Region, the National Research, Development and Innovation Office Hungary (NKFIH), the National Research Foundation of Korea, the Natural Science and Engineering Research Council Canada, Canadian Foundation for Innovation (CFI), the Brazilian Ministry of Science, Technology, and Innovations, the International Center for Theoretical Physics South American Institute for Fundamental Research (ICTP-SAIFR), the Research Grants Council of Hong Kong, the National Natural Science Foundation of China (NSFC), the Leverhulme Trust, the Research Corporation, the Ministry of Science and Technology (MOST), Taiwan, the United States Department of Energy, and the Kavli Foundation. The authors gratefully acknowledge the support of the NSF, STFC, INFN, and CNRS for the provision of computational resources The authors are grateful for computational resources provided by the LIGO Laboratory and supported by NSF Grants PHY-0757058 and PHY-0823459. This work carries LIGO document number P2300048.

Appendix A
Window Functions and Bias Factors

The window factors, $\bar{w}^4_{\rm ovl}$ and $\bar{w}^4$, used in Section 3.3 are defined as in Equations (34) and (24) in Lazzarini & Romano (2004). They are used to correct for the effect windowing has on our estimate of the variances. Actually, these corrections should include contributions from the autocorrelation function (PSD) of the individual detectors or their cross-correlation (see, e.g., Equations (21) and (32) of the same note). However, if the frequency response of the window is sufficiently strongly peaked around zero, then we can treat the transformed windows as delta functions (Whelan 2004) and our expressions for these quantities reduce to

$$\bar{w}^4 = \frac{1}{N}\sum_{i=1}^{N} w_i^4, \tag{A1}$$

where w_i represents the ith sample of the Hann window we use. Likewise, we need to account for the covariance between point estimates calculated in adjacent time segments. The point estimates are each quadratic in the data, windowed, and use 50% overlapping segments of data, and so we must account for the overlapping of the windows applied to the two segments:

$$\bar{w}^4_{\rm ovl} = \frac{1}{N/2}\sum_{i=N/2+1}^{N} w_i^2 w_{i-N/2}^2, \tag{A2}$$

where we see this now as the cross-correlation of the pieces of the two windows that overlap for the two segments.

When calculating the variance of our point estimate, we must estimate the quantity $(P_{1,f}P_{2,f})^{-1}$, which is the expression that appears in the Gaussian likelihood used to construct our optimal estimators (Matas & Romano 2021) and is therefore the relevant quantity when considering the variance of the point estimates. We briefly summarize how to properly estimate this quantity based on the discussion in Appendix B of Matas & Romano (2021), noting that they do not consider the effect of windowing, which we also discuss below.

For a segment of length T, we calculate estimators for the PSDs, $\hat{P}_{I,f}$, where I =1, 2 labels the detector, using Welch's method (Welch 1967). We break our time segment T into 50% overlapping chunks, calculate the PSD in each chunk, and average those estimates together. If we want a PSD with frequency resolution Δf, then we have K overlapping segments where $K = 2T\Delta f - 1$. We can assume our (noisy) estimators for the individual PSDs are unbiased and can be written as the true PSD plus some small deviation, $\hat{P}_{1,f} = P_{1,f} + \delta P_{1,f}$. We now look at the quantity of interest in calculating our variance:

$$\frac{1}{\hat{P}_{1,f}\hat{P}_{2,f}} = \frac{1}{[P_{1,f} + \delta P_{1,f}][P_{2,f} + \delta P_{2,f}]}. \tag{A3}$$

We can expand the denominator, take the expectation value of both sides, and use the fact that $\langle \delta P_{I,f}\rangle = 0$ and $\langle \delta P^2_{I,f}\rangle = {\rm var}P_{I,f}$, where $I = 1, 2$ labels the detector. This gives us

$$\left\langle \frac{1}{\hat{P}_{1,f}\hat{P}_{2,f}} \right\rangle \approx \frac{1}{P_{1,f}P_{2,f}}\left(1 + \frac{{\rm var}P_{1,f}}{P^2_{1,f}} + \frac{{\rm var}P_{2,f}}{P^2_{2,f}} + \cdots\right) \tag{A4}$$

$$= \frac{1}{P_{1,f}P_{2,f}}\left(1 + \frac{2\kappa}{K}\right). \tag{A5}$$

This expression can be compared to Equation (B8) in Matas & Romano (2021), noting that we have an extra term in the variance of our PSDs, κ. This term reduces the "effective" number of averages we perform due to our windowing, where we apply a Hann window with amplitude $\{w_i\}$ at each sample i, as well as the overlapping of our chunks of data. The correction factor is given by Welch (1967):

$$\kappa = \left[1 + 2\left(\frac{\sum_{i=N/2+1}^{N} w_i w_{i-N/2}}{\sum_{i=1}^{N} w_i^2}\right)^2 \frac{K-1}{K}\right]. \tag{A6}$$

In practice, we ignore the term $(K-1)/K$, as it leads to extra corrections that are $\mathcal{O}(K^{-2})$ that are quite small.

We can now define a bias correction factor based on the windowing we choose and the number of averages used in constructing $\hat{P}_{I,f}$. Defining $N_{\rm eff} = \kappa^{-1}K$, we have

$$\hat{\sigma}^{-2}(f) = \left(1 + \frac{2}{N_{\rm eff}}\right)\sigma^{-2}(f), \tag{A7}$$

where we have used simplified notation again, where the hat indicates our estimator for Equation (6) and the unhatted σ indicates the true value.

Taking the square root of both sides and inverting it gives us

$$\sigma = b(N_{\rm eff})\hat{\sigma}, \tag{A8}$$

where the bias factor, $b(N_{\rm eff})$, is given by

$$b(N_{\rm eff}) = \frac{N_{\rm eff}}{N_{\rm eff} - 1}, \tag{A9}$$

assuming $N_{\rm eff}$ is large. In Section 3.4, two different bias factors are discussed. In one case, the "naive" σ is estimated using one segment of length T, which results in fewer effective averages, and a larger bias correction than our typical estimate of σ,

which uses two adjacent segments of length T and therefore twice as many averages.

Appendix B
Marginalizing Over Calibration Uncertainty

Given measurements $\{\hat{\Omega}_i\}$ with uncertainties σ_i^2, as shown in Section 3.6, the following likelihood function can be used to perform parameter estimation on the GWB:

$$p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}) = \mathcal{N}\exp\left[-\frac{1}{2}\sum_f \frac{(\hat{\Omega}_f - \Omega_{\rm M}(f|\boldsymbol{\Theta}))^2}{\sigma_f^2}\right]. \quad \text{(B1)}$$

Here, the $\{\hat{\Omega}_f\}$ are a set of estimators for the GW energy density at discrete frequencies f, $\Omega_{\rm M}(f|\boldsymbol{\Theta})$ is a model for the energy density with parameters $\boldsymbol{\Theta}$, and $\mathcal{N}$ is a normalization constant. We will consider only a single baseline and neglect the sum over the detector pairs IJ appearing in Equation (33); if multiple detector pairs exist, the derivation below can be replicated for each pair.

Equation (B1) assumes that our estimators $\{\hat{\Omega}_f\}$ are direct, unbiased measurements of the underlying energy density spectrum. In general, however, the imperfect amplitude and phase calibration of GW detectors will break this assumption. We can account for the calibration uncertainty by amending our likelihood to introduce a new parameter λ:

$$p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}, \lambda) = \mathcal{N}\exp\left[-\frac{1}{2}\sum_f \frac{(\hat{\Omega}_f - \lambda\Omega_{\rm M}(f|\boldsymbol{\Theta}))^2}{\sigma_f^2}\right]. \quad \text{(B2)}$$

The parameter λ is an unknown multiplicative factor that encapsulates the potential calibration inaccuracy. In the case of perfect amplitude calibration ($\lambda = 1$), then $\{\hat{\Omega}_f\}$ are direct measurements of the underlying (unknown) energy spectrum. But if our calibration is imperfect ($\lambda \neq 1$), then $\{\hat{\Omega}_f\}$ are instead measurements of some multiple $\lambda\Omega(f)$ of the GWB spectrum. Although we do not know λ, it is possible to estimate the *uncertainty* on the instrumental calibration. We will therefore model λ itself as an unknown variable drawn from a normal distribution centered at 1 (corresponding to perfect calibration), but with a variance ϵ^2:

$$p(\lambda) \propto \exp\left[-\frac{1}{2\epsilon^2}(\lambda - 1)^2\right], \quad \text{(B3)}$$

where ϵ is a known amplitude calibration uncertainty. Additionally, we impose the constraint that λ be positive: we expect errors in the amplitude of strain measurements, but not their *sign*. In this case, the probability distribution for λ becomes

$$p(\lambda) = \sqrt{\frac{2}{\pi}}\frac{1}{\epsilon\left[1 + \mathrm{Erf}(\frac{1}{\sqrt{2\epsilon^2}})\right]}\exp\left[-\frac{1}{2\epsilon^2}(\lambda - 1)^2\right], \quad \text{(B4)}$$

normalized to unity on the interval $\lambda \in (0, \infty)$. Equation (B4) is our prior on λ.

We can now use Equation (B4) to marginalize our likelihood (Equation (B2)) over the unknown calibration factor λ. The marginalized likelihood is given by

$$\begin{aligned} p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}) &= \int p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}, \lambda)\, p(\lambda) d\lambda \\ &= \mathcal{N}\sqrt{\frac{2}{\pi}}\frac{1}{\epsilon\left[1 + \mathrm{Erf}(\frac{1}{\sqrt{2\epsilon^2}})\right]}\int_0^\infty \exp\left[-\frac{1}{2}\sum_f \frac{(\hat{\Omega}_f - \lambda\Omega_{\rm M}(f|\boldsymbol{\Theta}))^2}{\sigma_f^2} - \frac{1}{2}\frac{(\lambda - 1)^2}{\epsilon^2}\right] d\lambda. \end{aligned} \quad \text{(B5)}$$

If we define

$$A(\boldsymbol{\Theta}) = \frac{1}{\epsilon^2} + \sum_f \frac{\Omega_{\rm M}(f|\boldsymbol{\Theta})^2}{\sigma_f^2}, \quad \text{(B6)}$$

$$B(\boldsymbol{\Theta}) = \frac{1}{\epsilon^2} + \sum_f \frac{\hat{\Omega}_f\Omega_{\rm M}(f|\boldsymbol{\Theta})}{\sigma_f^2}, \quad \text{(B7)}$$

and

$$C(\boldsymbol{\Theta}) = \frac{1}{\epsilon^2} + \sum_f \frac{\hat{\Omega}_f^2}{\sigma_f^2}, \quad \text{(B8)}$$

the marginal likelihood can be more concisely expressed as

$$\begin{aligned} p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}) = \mathcal{N}\sqrt{\frac{2}{\pi}}&\frac{1}{\epsilon\left[1 + \mathrm{Erf}\left(\frac{1}{\sqrt{2\epsilon^2}}\right)\right]} \\ &\times \int_0^\infty \exp\left[-\frac{1}{2}(A(\boldsymbol{\Theta})\lambda^2 - 2B(\boldsymbol{\Theta})\lambda + C(\boldsymbol{\Theta}))\right] d\lambda; \end{aligned} \quad \text{(B9)}$$

this expression can be analytically integrated to obtain

$$\begin{aligned} p(\{\hat{\Omega}_f\}|\boldsymbol{\Theta}) = \mathcal{N}&\frac{1}{\epsilon\sqrt{A(\boldsymbol{\Theta})}}\left[\frac{1 + \mathrm{Erf}(\frac{B(\boldsymbol{\Theta})}{\sqrt{2A(\boldsymbol{\Theta})}})}{1 + \mathrm{Erf}(\frac{1}{\sqrt{2\epsilon^2}})}\right] \\ &\times \exp\left[-\frac{1}{2}\left(C(\boldsymbol{\Theta}) - \frac{B(\boldsymbol{\Theta})^2}{A(\boldsymbol{\Theta})}\right)\right]. \end{aligned} \quad \text{(B10)}$$

Marginalization of the calibration uncertainty is built into the `pygwb_pe` module, and this calculation is automatically triggered when passing a calibration error $\epsilon \neq 0$. Additional information on the treatment of calibration uncertainties can be found in Whelan et al. (2014).

ORCID iDs

Arianna I. Renzini https://orcid.org/0000-0002-4589-3987
Alba Romero-Rodríguez https://orcid.org/0000-0003-2275-4164
Colm Talbot https://orcid.org/0000-0003-2053-5582
Max Lalleman https://orcid.org/0000-0002-2254-010X
Shivaraj Kandhasamy https://orcid.org/0000-0002-4825-6764
Kevin Turbang https://orcid.org/0000-0002-9296-8603
Sylvia Biscoveanu https://orcid.org/0000-0001-7616-7366
Katarina Martinovic https://orcid.org/0000-0001-5299-7744
Patrick Meyers https://orcid.org/0000-0002-2689-0190
Kamiel Janssens https://orcid.org/0000-0001-8760-4429
Derek Davis https://orcid.org/0000-0001-5620-6751
Philip Charlton https://orcid.org/0000-0002-4263-2706
Irina Dvorkin https://orcid.org/0000-0002-2353-9194
Sharan Banagiri https://orcid.org/0000-0001-7852-7484
Thomas Callister https://orcid.org/0000-0001-9892-177X
Tania Regimbau https://orcid.org/0000-0002-0631-1198
Joseph D. Romano https://orcid.org/0000-0003-4915-3246

References

Aasi, J., Abbott, B. P., Abbott, R., et al. 2015a, CQGra, 32, 074001
Aasi, J., Abadie, J., Abbott, B. P., et al. 2015b, PhRvD, 91, 022003
Abbott, B. P., Abbott, R., Adhikari, R. , et al. 2007, ApJ, 659, 918
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2017, PhRvL, 118, 121101
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2018, CQGra, 35, 065010
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016, PhRvL, 116, 061102
Abbott, B. P., Abbott, R. , Abbott, T. D., et al. 2019, PhRvD, 100, 061101
Abbott, B., Abbott, R., Adhikari, R., et al. 2004, PhRvD, 69, 122004
Abbott, B. P., Abbott, R., Acernese, F., et al. 2009, Natur, 460, 990
Abbott, R., Abbott, T. D. , Abraham, S., et al. 2021, PhRvD, 104, 022004
Abbott, R., Abbott, T. D., Acernese, F., et al. 2023, PhRvX, 13, 011048
Abbott, R., et al. 2021c, ApJL, 913, L7
Acernese, F., Agathos, M., Agatsuma, K., et al. 2014, CQGra, 32, 024001
Acernese, F., Agathos, M., Ain, A., et al. 2022a, arXiv:2210.15634
Acernese, F., Agathos, M., Ain, A,, et al. 2022b, arXiv:2210.15633
Aghanim, N., Akrami, Y., Ashdown, M., et al. 2020, A&A, 641, A6
Ain, A., Dalvi, P., & Mitra, S. 2015, PhRvD, 92, 022003
Ain, A., Suresh, J., & Mitra, S. 2018, PhRvD, 98, 024001
Akutsu, T., Ando, M., Arai, K., et al. 2020, PTEP, 2021, 2021
Allen, B., & Romano, J. D. 1999, PhRvD, 59, 102001
Ashton, G., Hübner, M., Lasky, P. D., et al. 2019, ApJS, 241, 27
Barsotti, L., McCuller, L., Evans, M., & Fritschel, P. 2018, The A+ design curveLIGO-T1800042-v5, Laser Interferometer Gravitational Wave Observatory
Belczynski, K., & Kalogera, V. 2001, ApJL, 550, L183
Belczynski, K., Perna, R., Bulik, T., et al. 2006, ApJ, 648, 1110
Berger, E., Fox, D. B., Price, P. A., et al. 2007, ApJ, 664, 1000
Buikema, A., Cahillane, C., Mansell, G. L., et al. 2020, PhRvD, 102, 062003
Bulik, T., Belczynski, K., & Rudak, B. 2004, A&A, 415, 407
Buonanno, A., Chen, Y.-b., & Vallisneri, M. 2003, PhRvD, 67, 104025
Buonanno, A., Iyer, B., Ochsner, E., Pan, Y., & Sathyaprakash, B. S. 2009, PhRvD, 80, 084043
Callister, T., Biscoveanu, A. S., Christensen, N., et al. 2017, PhRvX, 7, 041058
Christensen, N. 2019, RPPh, 82, 016903
Coughlin, M. W., Christensen, N. L., De Rosa, R., et al. 2016, CQGra, 33, 224003
Coughlin, M. W., Cirone, A., Meyers, P., et al. 2018, PhRvD, 97, 102007
Covas, P. B., Effler, A., Goetz, E., et al. 2018, PhRvD, 97, 082002
Davis, D., & Walker, M. 2022, Galax, 10, 12
Davis, D., Areeda, J. S. , Berger, B. K., et al. 2021, CQGra, 38, 135014
Dodge, Y. 2008, Kolmogorov–Smirnov Test (New York: Springer), 283
Dominik, M., Belczynski, K., Fryer, C., et al. 2012, ApJ, 759, 52
Drasco, S., & Flanagan, E. E. 2003, PhRvD, 67, 082003
Himemoto, Y., Nishizawa, A., & Taruya, A. 2023, PhRvD, 107, 064055
Himemoto, Y., & Taruya, A. 2017, PhRvD, 96, 022004
Himemoto, Y., & Taruya, A. 2019, PhRvD, 100, 082001
Hopkins, A. M., & Beacom, J. F. 2006, ApJ, 651, 142
Janssens, K., Ball, M., Schofield, R. M. S., et al. 2023, PhRvD, 107, 022004
Janssens, K., Martinovic, K., Christensen, N., Meyers, P. M., & Sakellariadou, M. 2021, PhRvD, 104, 122006
Kass, R. E., & Raftery, A. E. 1995, JASA, 90, 773
Lawrence, J., Turbang, K., Matas, A., et al. 2023, PhRvD, 107, 103026
Lazzarini, A., & Romano, J. D. 2004, Use of Overlapping Windows in the Stochastic Background SearchT040089-x0, Laser Interferometer Gravitational Wave Observatory
LIGO Scientific Collaboration, Virgo Collaboration and KAGRA Collaboration 2021, Data for Upper Limits on the Isotropic Gravitational-Wave Background from Advanced LIGO's and Advanced Virgo's Third Observing RunG2001287-v5, Laser Interferometer Gravitational Wave Observatory
Macleod, D. M., Areeda, J. S., Coughlin, S. B., Massinger, T. J., & Urban, A. L. 2021, SoftX, 13, 100657
Maggiore, M., Van Den Broeck, C., Bartolo, N., et al. 2020, JCAP, 03, 050
Mandic, V., Thrane, E., Giampanis, S., & Regimbau, T. 2012, PhRvL, 109, 171102
Marassi, S., Schneider, R., Corvino, G., Ferrari, V., & Portegies Zwart, S. 2011, PhRvD, 84, 124037
Martinovic, K., Meyers, P. M., Sakellariadou, M., & Christensen, N. 2021, PhRvD, 103, 043023
Matas, A., Dvorkin, I., Romero, A., & Regimbau, T. 2021, Applying gating to stochastic searches in O3P2000546-v2, Laser Interferometer Gravitational Wave Observatory
Matas, A., & Romano, J. D. 2021, PhRvD, 103, 062003
McKechan, D. J. A., Robinson, C., & Sathyaprakash, B. S. 2010, CQGra, 27, 084020
Meacher, D., Cannon, K., Hanna, C., Regimbau, T., & Sathyaprakash, B. S. 2016, PhRvD, 93, 024018
Meacher, D., Coughlin, M., Morris, S., et al. 2015, PhRvD, 92, 063002
Meyers, P. M. 2018, PhD thesis, Univ. Minnesota, Twin Cities
Meyers, P. M., Martinovic, K., Christensen, N., & Sakellariadou, M. 2020, PhRvD, 102, 102005
Neijssel, C. J., Vigna-Gómez, A., Stevenson, S., et al. 2019, MNRAS, 490, 3740
O'Reilly, B., Branchesi, M., Haino, S., et al. 2020, GWTC-3: Compact Binary Coalescences Observed by LIGO and VIRGO during the Second Part of the Third Observing RunT2000012-v1, Laser Interferometer Gravitational Wave Observatory
O'Reilly, B., Branchesi, M., Haino, S., et al. 2020, Noise curves used for Simulations in the update of the Observing Scenarios Paper T2000012-v2, Laser Interferometer Gravitational Wave Observatory
Pankow, C., Chatziioannou, K., Chase, E. A., et al. 2018, PhRvD, 98, 084016
Press, W., Teukolsky, S., Vetterling, W., & Flannery, B. 2007, Numerical Recipes: The Art of Scientific Computing (3rd ed.; Cambridge: Cambridge Univ. Press)
Rabiner, L., & Gold, B. 1975, Theory and Application of Digital Signal Processing (Englewood Cliffs, NJ: Prentice-Hall)
Regimbau, T. 2011, RAA, 11, 369
Regimbau, T. 2022, Symm, 14, 270
Regimbau, T., Dent, T., Del Pozzo, W., et al. 2012, PhRvD, 86, 122001
Regimbau, T., Meacher, D., & Coughlin, M. 2014, PhRvD, 89, 084046
Regimbau, T., Siellez, K., Meacher, D., Gendre, B., & Boër, M. 2015, ApJ, 799, 69
Reitze, D., Adhikari, R., Ballmer, S., et al. 2019, BAAS, 51, 035
Renzini, A. I., Goncharov, B., Jenkins, A. C., & Meyers, P. M. 2022, Galax, 10, 34
Renzini, A. I., et al. 2023, pygwb v1.3.0, Zenodo, doi: 10.5281/zenodo.8062104
Romano, J. D., & Cornish, N. J. 2017, LRR, 20, 2
Sathyaprakash, B., Abernathy, M., Acernese, F., et al. 2011, arXiv:1108.1423
Seto, N., & Taruya, A. 2007, PhRvL, 99, 121101
Smith, R., & Thrane, E. 2018, PhRvX, 8, 021019
Speagle, J. S. 2020, MNRAS, 493, 3132
Talbot, C., Thrane, E., Biscoveanu, S., & Smith, R. 2021, PhRvD, 3, 043049
Thrane, E., Christensen, N., & Schofield, R. M. S. 2013, PhRvD, 87, 123009
Thrane, E., Christensen, N., Schofield, R. M. S., & Effler, A. 2014, PhRvD, 90, 023013
Thrane, E., & Romano, J. D. 2013, PhRvD, 88, 124032
Tsukada, L., Jaraba, S., Agarwal, D., & Floden, E. 2023, PhRvD, 107, 023024
Usman, S. A., et al. 2016, CQGra, 33, 215004
van Remortel, N., Janssens, K., & Turbang, K. 2023, PrPNP, 128, 104003
Welch, P. 1967, IEEE Trans. on Audio and Electroacoustics, 15, 70
Whelan, J. T. 2004, LIGO DocumentT040125, Laser Interferometer Gravitational Wave Observatory
Whelan, J. T., Robinson, E. L., Romano, J. D., & Thrane, E. H. 2014, JPhCS, 484, 012027