Publications of the Astronomical Society of the Pacific, 131:024503 (16pp), 2019 February https://doi.org/10.1088/1538-3873/aaef0b

PyCBC Inference: A Python-based Parameter Estimation Toolkit for Compact Binary Coalescence Signals

C. M. Biwer[1,2], Collin D. Capano[3], Soumi De[2], Miriam Cabero[3], Duncan A. Brown[2], Alexander H. Nitz[3], and V. Raymond[4,5]
[1] Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[2] Department of Physics, Syracuse University, Syracuse, NY 13244, USA
[3] Albert-Einstein-Institut, Max-Planck-Institut für Gravitationsphysik, D-30167 Hannover, Germany
[4] Albert-Einstein-Institut, Max-Planck-Institut für Gravitationsphysik, D-14476 Potsdam, Germany
[5] School of Physics and Astronomy, Cardiff University, Cardiff, CF243AA, Wales, UK
Received 2018 August 13; accepted 2018 November 6; published 2019 January 11

Abstract

We introduce new modules in the open-source PyCBC gravitational-wave astronomy toolkit that implement Bayesian inference for compact-object binary mergers. We review the Bayesian inference methods implemented and describe the structure of the modules. We demonstrate that the PyCBC Inference modules produce unbiased estimates of the parameters of a simulated population of binary black hole mergers. We show that the parameters' posterior distributions obtained using our new code agree well with the published estimates for binary black holes in the first Advanced LIGO–Virgo observing run.

Key words: gravitational waves – methods: data analysis – methods: statistical

Online material: color figures

1. Introduction

The observations of six binary black hole mergers (Abbott et al. 2016b, 2017a, 2017b, 2017c) and the binary neutron star merger GW170817 (Abbott et al. 2017a) by Advanced LIGO (Aasi et al. 2015) and Virgo (Acernese et al. 2015) have established the field of gravitational-wave astronomy. Understanding the origin, evolution, and physics of gravitational-wave sources requires accurately measuring the properties of detected events. In practice, this is performed using Bayesian inference (Bayes & Price 1763; Jaynes 2003). Bayesian inference allows us to determine the signal model that is best supported by observations and to obtain posterior probability densities for a model's parameters, hence inferring the properties of the source. In this paper, we present PyCBC Inference, a set of Python modules that implement Bayesian inference in the PyCBC open-source toolkit for gravitational-wave astronomy (Nitz et al. 2018). PyCBC Inference has been used to perform Bayesian inference for several astrophysical problems, including testing the black hole area increase law (Cabero et al. 2018); combining multimessenger observations of GW170817 to constrain the viewing angle of the binary (Finstad et al. 2018); and measuring the tidal deformabilities and radii of the neutron stars of GW170817 (De et al. 2018).

We provide a comprehensive description of the methods and code implemented in PyCBC Inference. We then demonstrate that PyCBC Inference can produce unbiased estimates of the parameters of a simulated population of binary black holes. We show that PyCBC Inference can recover posterior probability distributions that are in good agreement with the published measurements of the binary black holes detected in the first LIGO–Virgo observing run (Abbott et al. 2016b). This paper is organized as follows. Section 2 gives an overview of the Bayesian inference methods used in gravitational-wave astronomy for compact-object binary mergers. We provide an overview of the waveform models used; the likelihood function for a known signal in stationary, Gaussian noise; the sampling methods used to estimate the posterior probability densities and the evidence; the selection of independent samples; and the estimation of parameter values from posterior probabilities. Section 3 describes the design of the PyCBC Inference software and how the methods described in Section 2 are implemented in the code. Section 4 uses a simulated population of binary black holes and the black hole mergers detected in the first LIGO–Virgo observing run to demonstrate the use of PyCBC Inference. We provide the posterior probability densities for the events GW150914, GW151226, and LVT151012, and the command lines and configurations to reproduce these results as supplemental materials: https://github.com/gwastro/pycbc-inference-paper (De et al. 2018). Finally, we summarize the status of the code and possible future developments in Section 5.

2. Bayesian Inference for Binary Mergers

In gravitational-wave astronomy, Bayesian methods are used to infer the properties of detected astrophysical sources

(Finn & Chernoff 1993; Cutler & Flanagan 1994; Nicholson & Vecchio 1998; Christensen & Meyer 2001). Given the observed data $\vec{d}(t)$—here this is data from a gravitational-wave detector network in which a search has identified a signal (Allen et al. 2012; Usman et al. 2016; Nitz et al. 2017)—Bayes' theorem (Bayes & Price 1763; Jaynes 2003) states that for a hypothesis H,

$$p(\vec{\vartheta}|\vec{d}(t), H) = \frac{p(\vec{d}(t)|\vec{\vartheta}, H)p(\vec{\vartheta}|H)}{p(\vec{d}(t)|H)}, \tag{1}$$

where we define $p(A|B)$ as the conditional probability of event A given event B. In our case, the hypothesis H is the model of the gravitational-wave signal, and $\vec{\vartheta}$ are the parameters of this model. Together, these describe the properties of the astrophysical source of the gravitational waves. The posterior probability density, $p(\vec{\vartheta}|\vec{d}(t), H)$, in Equation (1) is the conditional probability that the signal has parameters $\vec{\vartheta}$ given the observation $\vec{d}(t)$ and waveform model H. The posterior probability density is proportional to the prior probability density, $p(\vec{\vartheta}|H)$, which describes our knowledge about the parameters before considering the observed datum $\vec{d}(t)$, and the likelihood, $p(\vec{d}(t)|\vec{\vartheta}, H)$, which is the probability of obtaining the observation $\vec{d}(t)$ given the waveform model H with parameters $\vec{\vartheta}$.

Often, we are only interested in a subset of the parameters $\vec{\vartheta}$. To obtain a probability distribution on one or a few parameters, we marginalize the posterior probability by integrating $p(\vec{d}(t)|\vec{\vartheta}, H)p(\vec{\vartheta}|H)$ over the unwanted parameters. Marginalizing over all parameters yields the evidence, $p(\vec{d}(t)|H)$, which is the denominator in Equation (1). The evidence serves as a normalization constant of the posterior probability for the given model H. If we have two competing models H_A and H_B, the evidence can be used to determine which model is favored by the data via the Bayes factor (Kass & Raftery 1995; Gelman & Meng 1998; Skilling 2006),

$$\mathcal{B} = \frac{p(\vec{d}(t)|H_A)}{p(\vec{d}(t)|H_B)}. \tag{2}$$

If $\mathcal{B}$ is greater than 1, then model H_A is favored over H_B, with the magnitude of $\mathcal{B}$ indicating the degree of belief.

PyCBC Inference can compute Bayes factors and produce marginalized posterior probability densities given the data from a network of gravitational-wave observatories with N detectors $\vec{d}(t) = \{d_i(t);\ 1 < i < N\}$, and a model H that describes the astrophysical source. In the remainder of this section, we review the methods used to compute these quantities.

2.1. Waveform Models

The gravitational waves radiated in a binary merger's source frame are described by the component masses, $m_{1,2}$, the three-dimensional spin vectors, $\vec{s}_{1,2}$, of the compact objects (Thorne 1987), and the binary's eccentricity, e (Peters 1964). A parameter ϕ describes the phase of the binary at a fiducial reference time, although this is not usually of physical interest. For binaries containing neutron stars, additional parameters $\Lambda_{1,2}$ describe the star's tidal deformabilities (Flanagan & Hinderer 2008; Hinderer 2008), which depend on the equation of state of the neutron stars. The waveform observed by the Earth-based detector network depends on six additional parameters: the signal's time of arrival, t_c, the binary's luminosity distance, d_L, and four Euler angles that describe the transformation from the binary's frame to the detector network frame (Wahlquist 1987). These angles are typically written as the binary's right ascension, α, declination, δ, a polarization angle, Ψ, and the inclination angle, ι (the angle between the binary's angular momentum axis and the line of sight). The convention adopted in PyCBC for ι denotes $\iota = 0$ as a "face-on" binary (the line of sight parallel to binary angular momentum), $\iota = \frac{\pi}{2}$ as an "edge-on" binary (the line of sight perpendicular to binary angular momentum), and $\iota = \pi$ as a "face-off" binary (the line of sight anti-parallel to binary angular momentum).

Binary mergers present a challenging problem for Bayesian inference, as the dimensionality of the signal parameter space is large. This is further complicated by correlations between the signal's parameters. For example, at leading order, the gravitational waveform depends on the chirp mass, $\mathcal{M}$ (Peters & Mathews 1963). The mass ratio enters the waveform at higher orders, and is more difficult to measure. This results in an amplitude-dependent degeneracy between the component masses (Christensen & Meyer 2001). Similarly, the binary's mass ratio can be degenerate with its spin (Hannam et al. 2013), although this degeneracy can be broken if the binary is precessing. Much of the effort of parameter estimation in gravitational-wave astronomy has focused on developing computationally feasible ways to explore this signal space, and on extracting physically interesting parameters (or combinations of parameters) from the large, degenerate parameter space (see, e.g., Veitch et al. 2015 and references therein). However, in many problems of interest, we are not concerned with the full parameter space described above. For example, field binaries (ie. binaries formed in isolation and not influenced by surrounding stars) are expected to have negligible eccentricity when they are observed by Advanced LIGO and Virgo (Peters & Mathews 1963), so eccentricity is neglected in the waveform models. Simplifying assumptions can be made about the compact object's spins (e.g., the spins are aligned with the binary's orbital angular momentum) reducing the dimensionality of the waveform parameters space.

Given a set of parameters, $\vec{\vartheta}$, one can obtain a model of the gravitational-wave signal from a binary merger using a variety of different methods, including post-Newtonian theory (see, e.g., Blanchet (2006) and references therein), analytic models calibrated against numerical simulations (Buonanno & Damour 1999, 2000; Damour et al. 2000; Damour 2001; Ajith et al. 2007, 2011; Santamaria et al. 2010), perturbation theory (Teukolsky 1972; Berti et al. 2009), and full numerical solution of the Einstein equations (see, e.g., Cardoso et al. 2015 and references therein). Obtaining posterior probabilities and evidences can require calculating $\mathcal{O}(10^9)$ template waveforms, which restricts us to models that are computationally efficient to calculate. The cost of full numerical simulations makes them prohibitively expensive at present. Even some analytic models are too costly to be used, and surrogate models have been developed that capture the features of these waveforms at reduced computational cost (Pürrer 2016; Lackey et al. 2017).

The specific choice of the waveform model H for an analysis depends on the physics that we wish to explore, computational cost limitations, and the level of accuracy desired in the model. A variety of waveform models are available for use in PyCBC Inference, either directly implemented in PyCBC or via calls to the LIGO Algorithm Library (LAL; Mercer et al. 2017). We refer to the PyCBC and LAL documentation, and references therein, for detailed descriptions of these models. In this paper, we demonstrate the use of PyCBC Inference using the IMRPhenomPv2 (Schmidt et al. 2015; Hannam et al. 2014) waveform model for binary black hole mergers. This model captures the inspiral-merger-ringdown physics of spinning, precessing binaries and parameterizing spin effects using a spin magnitude, a_j, an azimuthal angle, θ_j^a, and a polar angle, θ_j^p, for each of the two compact objects. Examples of using PyCBC Inference with different waveform models include the analysis of De et al. (2018), which used the TaylorF2 post-Newtonian waveform model with tidal corrections, and Cabero et al. (2018) that used a ringdown-only waveform that models the quasi-normal modes of the remnant black hole.

2.2. Likelihood Function

The data observed by the gravitational-wave detector network enters Bayes' theorem through the likelihood, $p(\vec{d}(t)|\vec{\vartheta}, H)$, in Equation (1). Currently, PyCBC Inference assumes that each detector produces stationary Gaussian noise, $n_i(t)$, that is uncorrelated between the detectors in the network. The observed data is then $d_i(t) = n_i(t) + s_i(t)$, where $s_i(t)$ is the gravitational waveform observed in the i-th detector. For detectors that are not identical and co-located (as in the case of the Advanced LIGO–Virgo network), each detector observes a slightly different waveform due to their different antennae patterns, which are functions of the sky position (right ascension and declination) and polarization (Wahlquist 1987).

Under these assumptions, the appropriate form of $p(\vec{d}(t)|\vec{\vartheta}, H)$ is the well-known likelihood for a signal of known morphology in Gaussian noise (see, e.g., Wainstein & Zubakov 1962 for its derivation), which is given by

$$\begin{aligned} p(\vec{d}(t)|\vec{\vartheta}, H) &= \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\langle\tilde{n}_i(f)|\tilde{n}_i(f)\rangle\right] \\ &= \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\langle\tilde{d}_i(f) - \tilde{s}_i(f, \vec{\vartheta})|\tilde{d}_i(f) - \tilde{s}_i(f, \vec{\vartheta})\rangle\right], \end{aligned} \tag{3}$$

where N is the number of detectors in the network. The inner product, $\langle\tilde{a}|\tilde{b}\rangle$, is

$$\langle\tilde{a}_i(f)|\tilde{b}_i(f)\rangle = 4\Re\int_0^\infty \frac{\tilde{a}_i(f)\tilde{b}_i(f)}{S_n^{(i)}(f)}\mathrm{d}f, \tag{4}$$

where $S_n^{(i)}(f)$ is the power spectral density of the of the i-th detector's noise. Here, $\tilde{d}_i(f)$ and $\tilde{n}_i(f)$ are the frequency domain representations of the data and noise, obtained by a Fourier transformation of $d_i(t)$ and $n_i(t)$, respectively. The model waveform, $\tilde{s}_i(f, \vec{\vartheta})$, may be computed directly in the frequency domain or in the time domain, and then Fourier transformed to the frequency domain. There are several operations (e.g., Fourier transforms, noise power spectral density estimation, and inner products) that are common between the calculation of Equation (3) and the computation of the matched-filter signal-to-noise ratio (S/N) in PyCBC (Allen et al. 2012; Usman et al. 2016; Nitz et al. 2017). PyCBC Inference uses these existing functions where appropriate.

In general, gravitational-wave signals consist of a superposition of harmonic modes. However, in many cases, it is sufficient to model only the most dominant mode, as the subdominant harmonics are too weak to be measured. In this case, the signal observed in all detectors has the same simple dependence on the fiducial phase ϕ,

$$\tilde{s}_i(f, \vec{\vartheta}, \phi) = \tilde{s}_i^0(f, \vec{\vartheta}, 0)e^{i\phi}. \tag{5}$$

The posterior probability, $p(\vec{\vartheta}|\vec{d}(t), H)$, can be analytically marginalized over ϕ for such models (Wainstein & Zubakov 1962). Assuming a uniform prior on $\phi \in [0, 2\pi)$, the marginalized posterior is

$$\begin{aligned} \log p(\vec{\vartheta}|\vec{d}(t), H) \propto \log p(\vec{\vartheta}|H) + I_0\left(\left|\sum_i O(\tilde{s}_i^0, \tilde{d}_i)\right|\right) \\ - \frac{1}{2}\sum_i[\langle\tilde{s}_i^0, \tilde{s}_i^0\rangle - \langle\tilde{d}_i, \tilde{d}_i\rangle], \end{aligned} \tag{6}$$

where

$$\tilde{s}_i^0 \equiv \tilde{s}_i(f, \vec{\vartheta}, \phi = 0),$$

$$O(\tilde{s}_i^0, \tilde{d}_i) \equiv 4\int_0^\infty \frac{\tilde{s}_i^*(f; \vartheta, 0)\tilde{d}_i(f)}{S_n^{(i)}(f)}\mathrm{d}f,$$

and I_0 is the modified Bessel function of the first kind.

We have found that analytically marginalizing over ϕ in this manner reduces the computational cost of the analysis by a factor of 2–3. The IMRPhenomPv2 model that we use here is a simplified model of precession that allows for this analytic marginalization (Schmidt et al. 2015; Hannam et al. 2014). As the fiducial phase is generally a nuisance parameter, we use this form of the likelihood function in Sections 4.1 and 4.2.

2.3. Sampling Methods

Stochastic sampling techniques, in particular the Markov chain Monte Carlo (MCMC) methods (Metropolis et al. 1953; Geman & Geman 1984; Gilks et al. 1996; Gelman et al. 1995), have been used to numerically sample the posterior probability density function of astrophysical parameters for binary-merger signals (Christensen & Meyer 2001; Christensen et al. 2004; Rover et al. 2006, 2007; Abbott et al. 2016b, 2017a, 2017b, 2017c, 2017d). Ensemble MCMC algorithms use multiple Markov chains to sample the parameter space. A simple choice to initialize the k-th Markov chain in the ensemble is to draw a set of parameters, $\vec{\vartheta}_1^{(k)}$, from the prior probability density function. The Markov chains move around the parameter space according to the following set of rules. At iteration l, the k-th Markov chain has the set of parameters, $\vec{\vartheta}_l^{(k)}$. The sampling algorithm chooses a new proposed set of parameters, $\vec{\vartheta}_{l'}^{(k)}$, with probability $Q(\vec{\vartheta}_l^{(k)}, \vec{\vartheta}_{l'}^{(k)})$. When a new set of parameters is proposed, the sampler computes an acceptance probability, γ, which determines if the Markov chain should move to the proposed parameter set $\vec{\vartheta}_{l'}^{(k)}$, such that $\vec{\vartheta}_{l+1}^{(k)} = \vec{\vartheta}_{l'}^{(k)}$. If $\vec{\vartheta}_{l'}^{(i)}$ is rejected, then $\vec{\vartheta}_{l+1}^{(k)} = \vec{\vartheta}_l^{(k)}$. After a sufficient number of iterations, the ensemble converges to a distribution that is proportional to a sampling of the posterior probability density function. The true astrophysical parameters, $\vec{\vartheta}$, can then be estimated from histograms of the position of the Markov chains in the parameter space. Different ensemble sampling algorithms make particular choices for the proposal probability $Q(\vec{\vartheta}_l^{(k)}, \vec{\vartheta}_{l'}^{(k)})$ and acceptance probability, γ.

The open-source community has several well-developed software packages that implement algorithms for sampling the posterior probability density function. PyCBC Inference leverages these developments, and we have designed a flexible framework that allows the user to choose from multiple ensemble sampling algorithms. Currently, PyCBC Inference supports the open-source ensemble sampler `emcee` (Foreman-Mackey et al. 2013, 2018), its parallel-tempered version `emcee_pt` (Foreman-Mackey et al. 2018; Vousden et al. 2016), and the `kombine` (Farr & Farr 2015; Farr et al. 2018) sampler. All three are ensemble MCMC samplers. The sampling algorithm advances the positions of the Markov chains based on their previous positions and provides PyCBC Inference these positions as they get updated.

The `emcee_pt` sampler is a parallel-tempered sampler, which advances multiple ensembles based on the tempering or the "temperatures" used to explore the posterior probability density function. The posterior probability density function for a particular temperature, T, is modified, such that

$$p_T(\vec{\vartheta}|\vec{d}(t), H) = \frac{p(\vec{d}(t)|\vec{\vartheta}, H)^{\frac{1}{T}} p(\vec{\vartheta}|H)}{p(\vec{d}(t)|H)}. \quad (7)$$

The `emcee_pt` sampler uses several temperatures in parallel, and the position of Markov chains are swapped between temperatures using an acceptance criteria described in Vousden et al. (2016). Mixing of Markov chains from the different temperatures makes parallel-tempered samplers suitable for sampling posterior probability density functions with widely separated modes in the parameter space (Vousden et al. 2016). The `emcee` sampler performs the sampling using one temperature, where $T = 1$.

The `kombine` sampler on the other hand uses clustered kernel-density estimates to construct its proposal distribution, and proposals are accepted using the Metropolis–Hastings condition (Hastings 1970). The `kombine` sampler has been included in PyCBC Inference due to its efficient sampling which significantly lowers the computational cost of an analysis relative to the `emcee_pt` sampler. However, in Section 4.1, we found that the nominal configuration of the `kombine` sampler produced biased estimates of parameters for binary black holes.

2.4. Selection of Independent Samples

The output returned by the sampling algorithms discussed in Section 2.3 are Markov chains. Successive states of these chains are not independent, as Markov processes depend on the previous state (Christensen et al. 2004). The autocorrelation length, τ_K, of a Markov chain is a measure of the number of iterations required to produce independent samples of the posterior probability density function (Madras & Sokal 1988). The autocorrelation length of the k-th Markov chain, $X_l^{(k)} = \{\vec{\vartheta}_g^{(k)}; 1 < g < l\}$, of length, l, obtained from the sampling algorithm is defined as

$$\tau_K = 1 + 2\sum_{i=1}^{K} \hat{R}_i, \quad (8)$$

where K is the first iteration along the Markov chain the condition, $m\tau_K \leqslant K$ is true, m being a parameter which in PyCBC Inference is set to 5 (Madras & Sokal 1988). The autocorrelation function, $\hat{R}_i$, is defined as

$$\hat{R}_i = \frac{1}{l\sigma^2}\sum_{t=1}^{l-i}(X_t - \mu)(X_{t+i} - \mu), \quad (9)$$

where X_t are the samples of $X_l^{(k)}$ between the 0-th and the t-th iteration, X_{t+i} are the samples of $X_l^{(k)}$ between the 0-th and the

$(t+1)$-th iterations. Here, μ and σ^2 are the mean and variance of X_t, respectively.

The initial positions of the Markov chains influence their subsequent positions. The length of the Markov chains before they are considered to have lost any memory of the initial positions is called the "burn-in" period. It is a common practice in MCMC analyses to discard samples from the burn-in period to prevent any bias introduced by the initial positions of the Markov chains on the estimates of the parameters from the MCMC. PyCBC Inference has several methods to determine when the Markov chains are past the burn-in period. Here, we describe two methods, `max_posterior` and `n_acl`, which we have found to work well with the `kombine` and `emcee_pt` samplers used in Sections 4.1 and 4.2.

The `max_posterior` algorithm is an implementation of the burn-in test used for the MCMC sampler in Veitch et al. (2015). In this method, the k-th Markov chain is considered to be past the burn-in period at the first iteration, l, for which

$$\log \mathcal{L}_l^{(k)}(\vec{\vartheta}) \geqslant \max_{k,l} \log \mathcal{L}(\vec{\vartheta}) - \frac{N_p}{2}, \tag{10}$$

where $\mathcal{L}(\vec{\vartheta})$ is the prior-weighted likelihood,

$$\mathcal{L}(\vec{\vartheta}) = p(\vec{d}(t)|\vec{\vartheta}, H)p(\vec{\vartheta}|H), \tag{11}$$

and N_p is the number of dimensions in the parameter space. The maximization, $\max_{k,l} \log \mathcal{L}(\vec{\vartheta})$, is carried out over all Markov chains and iterations. The ensemble is considered to be past the burn-in period at the first iteration, where all Markov chains pass this test. We have found this test works well with the `kombine` sampler if the network S/N of the signal is $\gtrsim 5$.

We have found that the `max_posterior` test returns an iteration in the MCMC, while all Markov chains are still converging when used with the `emcee_pt` sampler. This underestimates the burn-in period because there is still an influence from the initial points on the samples of the posterior probability density function; however, adding at least one autocorrelation length to the burn-in period computed with the `max_posterior` test reduces this effect. Therefore, we use the `n_acl` test with the `emcee_pt` sampler. This test posits that the sampler is past the burn-in period if the length of the chains exceed 10 times the autocorrelation length. The autocorrelation length is calculated using samples from the second half of the Markov chains. If the test is satisfied, the sampler is considered to be past the burn-in period at the midway point of the Markov chains.

Correlations between the neighboring samples after the burn-in period are removed by "thinning" or drawing samples from the Markov chains with an interval of the autocorrelation length (Christensen et al. 2004). This is done so that the samples used to estimate the posterior probability density function are independent. Therefore, the number of independent samples of the posterior probability density function is equal to the number of Markov chains used in the ensemble times the number of iterations after the burn-in period divided by the autocorrelation length. PyCBC Inference will run until it has obtained the desired number of independent samples after the burn-in period.

2.5. Credible Intervals

After discarding samples from the burn-in period and thinning the remaining samples of the Markov chains, the product is the set of independent samples as described in Section 2.4. Typically we summarize the measurement of a given parameter using a credible interval. The $x\%$ credible interval is an interval where the true parameter value lies with a probability of $x\%$. PyCBC Inference provides the capability to calculate credible intervals based on percentile values. In the percentile method, the $x\%$ credible interval of a parameter value is written as A_{-B}^{+C}, where A is typically the 50th percentile (median) of the marginalized histograms. The values $A - B$ and $A + C$ represent the lower and upper boundaries of the $x\%$ credible interval, and they are computed as the $(50 - x/2)$-th and $(50 + x/2)$-th percentiles, respectively.

An alternative method of calculating a credible interval estimate is the Highest Posterior Density (HPD) method. An x % HPD interval is the shortest interval that contains $x\%$ of the probability. The percentile method explained above imposes a non-zero lower boundary to the interval being measured. This can be perceived as a limitation in cases where the weight of histogram at the $\sim$0-th percentile is not significantly different from the weight at the lower boundary of the credible interval. Intervals constructed using the HPD method may be preferred in such cases. Previous studies have noted that HPD intervals may be useful when the posterior distribution is not symmetric (Chen et al. 2000). PyCBC Inference uses HPD to construct confidence contours for two-dimensional marginal distributions, but HPD is not used in the construction of one-dimensional credible intervals for a single parameter. This functionality will be included in a future release of PyCBC Inference.

3. The PyCBC Inference Toolkit

In this section we describe the implementation of PyCBC Inference within the broader PyCBC toolkit. PyCBC provides both modules for developing code and executables for performing specific tasks with these modules. The code is available on the public GitHub repository at https://github.com/gwastro/pycbc, with executables located in the directory `bin/inference` and the modules in the directory `pycbc/inference`. PyCBC Inference provides an executable called `pycbc_inference` that is the main engine for performing Bayesian inference with PyCBC. A call graph of `pycbc_inference` is shown in Figure 1. In this section, we review the structure of the main engine and the Python objects used to build `pycbc_inference`.

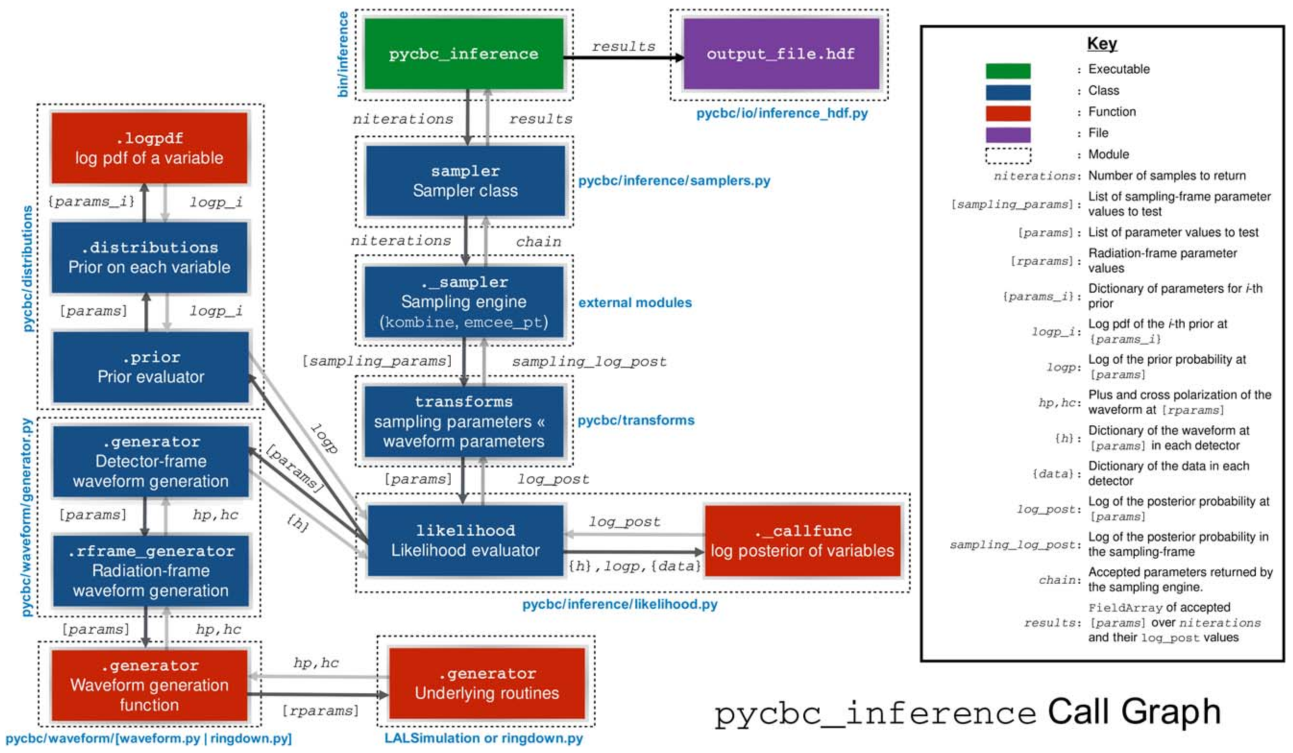

bin/inference
pycbc_inference
results
output_file.hdf
pycbc/io/inference_hdf.py
niterations
results
sampler
Sampler class
pycbc/inference/samplers.py
niterations
chain
._sampler
Sampling engine
(kombine, emcee_pt)
external modules
[sampling_params]
sampling_log_post
transforms
sampling parameters « waveform parameters
pycbc/transforms
[params]
log_post
likelihood
Likelihood evaluator
log_post
._callfunc
log posterior of variables
{h}, logp, {data}
pycbc/inference/likelihood.py
pycbc/distributions
.logpdf
log pdf of a variable
{params_i}
logp_i
.distributions
Prior on each variable
[params]
logp_i
.prior
Prior evaluator
logp
[params]
{h}
pycbc/waveform/generator.py
.generator
Detector-frame waveform generation
[params]
hp,hc
.rframe_generator
Radiation-frame waveform generation
[params]
hp,hc
.generator
Waveform generation function
hp,hc
[rparams]
.generator
Underlying routines
pycbc/waveform/[waveform.py | ringdown.py]
LALSimulation or ringdown.py
Key
🟩 : Executable
🟦 : Class
🟥 : Function
🟪 : File
: Module
niterations: Number of samples to return
[sampling_params]: List of sampling-frame parameter values to test
[params]: List of parameter values to test
[rparams]: Radiation-frame parameter values
{params_i}: Dictionary of parameters for i-th prior
logp_i: Log pdf of the i-th prior at {params_i}
logp: Log of the prior probability at [params]
hp,hc: Plus and cross polarization of the waveform at [rparams]
{h}: Dictionary of the waveform at [params] in each detector
{data}: Dictionary of the data in each detector
log_post: Log of the posterior probability at [params]
sampling_log_post: Log of the posterior probability in the sampling-frame
chain: Accepted parameters returned by the sampling engine.
results: FieldArray of accepted [params] over niterations and their log_post values
pycbc_inference Call Graph

Figure 1. Executable `pycbc_inference` samples the posterior probability density function. For an iteration in an ensemble MCMC algorithm, the `Sampler` object uses the `LikelihoodEvaluator` object to compute the natural logarithm of the posterior probability, and returns it to `pycbc_inference`. The `LikelihoodEvaluator` object uses the `Generator` object to generate the waveform and `Distribution` objects to evaluate the prior probability density function. Samples are periodically written to the output file.

(A color version of this figure is available in the online journal.)

3.1. `pycbc_inference` *Executable*

The methods presented in Sections 2.1, 2.2, 2.3, 2.4, and 2.5 are used to build the executable `pycbc_inference`. For faster performances, `pycbc_inference` can be run on high-throughput computing frameworks such as HTCondor (Tannenbaum et al. 2001; Thain et al. 2005) and the processes for running the samplers can be parallelized over multiple compute nodes using MPI (Dalcin et al. 2011; Dalcín et al. 2008, 2005). The execution of the likelihood computation and the PSD estimation are done using either single-threaded or parallel FFT engines, such as FFTW (Frigo & Johnson 2005) or the Intel Math Kernel Library (MKL). For maximum flexibility in heterogeneous computing environments, the processing scheme to be used is specified at runtime as a command line option to `pycbc_inference`.

The input to `pycbc_inference` is a configuration file which contains up to seven types of sections. The `variable_args` section specifies the parameters that are to be varied in the MCMC. There is a `prior` section for each parameter in the `variable_args` section that contains arguments to initialize the prior probability density function for that parameter. There is a `static_args` section specifying any parameter for waveform generation along with its assigned value that should be fixed in the ensemble MCMC. Optionally, the configuration file may also include a `constraint` section(s) containing any conditions that constrain the prior probability density functions of the parameters. For efficient convergence of a Markov chain, it may be desirable to sample the prior probability density function in a different coordinate system than that defined by the parameters in the `variable_args` section or the parameters inputted to the waveform generation functions. Therefore, the configuration file may contain a `sampling_parameters` and `sampling_transforms` section(s) that specifies the transformations between parameters in the `variable_args` sections and the parameters evaluated in the prior probability density function. Finally, the waveform generation functions recognize only a specific set of input parameters. The `waveform_transforms` section(s) may be provided which maps parameters in the `variable_args` section to parameters understood by the waveform generation functions. More details on application of constraints and execution of coordinate transformations are provided in Sections 3.3 and 3.4, respectively.

The location of the configuration file, gravitational-wave detector data files, data conditioning settings, and settings for the ensemble MCMC are supplied on the command line interface to `pycbc_inference`. The results from running `pycbc_inference` are stored in a HDF (Collette et al. 2018) file whose location is provided on the command line to `pycbc_inference` as well. The main results of interest are stored under the HDF groups `''samples''` and `''likelihood_stats''`. The `''samples''` group contains the history of the Markov chains as separate data sets for each of the variable parameters. The `''likelihood_stats''` group contains a data set of the natural logarithm of the Jacobian, which is needed to transform from the variable parameters to sampling parameters, a data set containing the natural logarithm of the prior probabilities, and a data set containing the natural logarithm of the likelihood ratio Λ where

$$\Lambda = \log \frac{p(\vec{d}(t)|\vec{\vartheta}, H)}{p(\vec{d}(t)|\vec{n})} = -\frac{1}{2}\sum_{i=1}^{N}[\langle \tilde{s}_i(f, \vec{\vartheta})|\tilde{s}_i(f, \vec{\vartheta})\rangle - 2\langle \tilde{d}_i(f)|\tilde{s}_i(f, \vec{\vartheta})\rangle]. \quad (12)$$

The natural logarithm of the noise likelihood,

$$\log p(\vec{d}(t)|\vec{n}) = -\frac{1}{2}\sum_{i=1}^{N}\langle \tilde{d}_i(f)|\tilde{d}_i(f)\rangle, \quad (13)$$

is stored as an attribute in the output file. The sum of the natural logarithms of Λ and $p(\vec{d}(t)|\vec{n})$ is used to compute the likelihood in Equation (3). The values for Λ and $\log p(\vec{d}(t)|\vec{n})$ are stored instead of the likelihood in Equation (3), because Λ is the quantity maximized in gravitational-wave searches and it is closely related the matched-filter S/N (i.e., the $\langle \tilde{d}_i(f)|\tilde{s}_i(f, \vec{\vartheta})\rangle$ components in Equation (12)) (Allen et al. 2012; Usman et al. 2016). Each of the data sets under the `''samples''` group and the `''likelihood_stats''` group has shape `nwalkers` × `niterations` if the sampling algorithm used in the analysis did not include parallel tempering, and has shape `ntemps` × `nwalkers` × `niterations` for parallel-tempered samplers. Here, `nwalkers` is the number of Markov chains, `niterations` is the number of iterations, and `ntemps` is the number of temperatures.

`pycbc_inference` terminates once it reaches the desired number of independent samples after the burn-in period using the methods in Section 2.4. The number of independent samples is set on the command line by the analyst. It is computationally expensive to obtain the desired number of independent samples using ensemble MCMC methods, and the `pycbc_inference` processes may terminate early due to problems on distributed-computing networks. Therefore, `pycbc_inference` has checkpointing implemented which allows users to resume an analysis from the last set of positions of Markov chains written to the output file. The samples from the Markov chains should be written at regular intervals so that `pycbc_inference` can resume the ensemble MCMC from the position of the Markov chains near the "state" the process was terminated. The state is determined from a random number generator and the position of the Markov chains. The frequency with which `pycbc_inference` writes the samples from Markov chains and the state of the random number generator to the output file and a backup file is specified by the user on the command line. A backup file is written by `pycbc_inference` because the output file from `pycbc_inference` may be corrupted. For example, if the process is aborted while writing to the output file, then the output file may be corrupted. In that case, samples and the state of the random number generator are loaded from the backup file, and the backup file is copied to the output file. This ensures that the `pycbc_inference` process can always be resumed.

For analyses that use the `emcee_pt` sampler, the likelihood can be used to compute the natural logarithm of the evidence using the `emcee_pt` sampler's `thermodynamic_integration_log_evidence` function (Foreman-Mackey et al. 2018). Then, the evidences from two analyses can be used to compute the Bayes factor, $\mathcal{B}$, for the comparison of two waveform models.

We provide example configuration files and run scripts for the analysis of the binary black hole mergers detected in the Advanced LIGO's first observing run in https://github.com/gwastro/pycbc-inference-paper. These examples can be used with the open-source data sets provided by the LIGO Open Science Center (Vallisneri et al. 2015). The results of these analyses are presented in Section 4.2.

3.2. `Sampler` Objects

The PyCBC Inference modules provide a set of `Sampler` objects which execute the Bayesian sampling methods. These objects provide classes and functions for using open-source samplers, such as `emcee` (Foreman-Mackey et al. 2018) `emcee_pt`, (Foreman-Mackey et al. 2018), or `kombine` (Farr et al. 2018). This acts as an interface between PyCBC Inference and the external sampler package. The executable `pycbc_inference` initializes, executes, and saves the output from the `Sampler` objects. A particular `Sampler` object is chosen on the command line of `pycbc_inference` with the `--sampler` option. The `Sampler` object provides the external sampler package the positions of the Markov chains in the parameter space, the natural logarithm of the posterior probabilities at the current iteration, the current state determined from the random number generator, and the number of iterations that the sampler is requested to run starting from the current iteration. After running for the given number of iterations, the sampler returns the updated positions of the

Markov chains, the natural logarithm of the posterior probabilities, and the new state.

3.3. `Transform` Objects

The `Transform` objects in PyCBC Inference are used to perform transformations between different coordinate systems. Currently, the `Transform` objects are used in two cases: sampling transforms and waveform transforms.

Sampling transforms are used for transforming parameters that are varied in the ensemble MCMC to a different coordinate system before evaluating the prior probability density function. As there exists degeneracies between several parameters in a waveform model, it is useful to parameterize the waveform using a preferred set of parameters which could minimize the correlations. This leads to more efficient sampling, and therefore it leads to faster convergence of the Markov chains. One example of a sampling transformation is the transformation between the component masses m_1 and m_2 to chirp mass, $\mathcal{M}$, and mass ratio, q. The convention adopted for q in PyCBC Inference is $q = m_1/m_2$, where m_1 and m_2 are the component masses with $m_1 > m_2$. The chirp mass, $\mathcal{M}$, is the most accurately measured parameter in a waveform model because it is in the leading order term of the post-Newtonian expression of the waveform model. In contrast, the degeneracies of the mass ratio with spin introduces uncertainties in measurements of the component masses. Therefore, sampling in $\mathcal{M}$ and q proves to be more efficient than m_1 and m_2 (Rover et al. 2006; Veitch et al. 2015; Farr et al. 2016). In the GW150914, LVT151012, and GW151226 configuration files in https://github.com/gwastro/pycbc-inference-paper (De et al. 2018), we demonstrate how to allow the `Sampler` object to provide priors in the (m_1, m_2) coordinates, and specify sampling transformations in the $(\mathcal{M}, q)$ coordinates.

Waveform transforms are used to transform variable parameters in the ensemble MCMC that may not be recognized by the waveform model functions to parameters that are recognized. For example, the total mass, $M_{\rm tot} = m_1 + m_2$, and the mass ratio q may be used as variable parameters in the MCMC; however, these parameters will not be accepted by the waveform model functions. Waveform transforms will be used in this case to transform $M_{\rm tot}$, and q to the component masses m_1 and m_2, which are accepted by the waveform model functions.

3.4. `LikelihoodEvaluator` Object

The `LikelihoodEvaluator` object computes the natural logarithm of the prior-weighted likelihood given by the numerator of Equation (1). Because the evidence is constant for a given waveform model, the prior-weighted likelihood is proportional to the posterior probability density function, and it can be used in sampling algorithms to compute the acceptance probability γ instead of the full posterior probability density function. The prior-weighted likelihood is computed for each new set of parameters as the `Sampler` objects advance the Markov chains through the parameter space.

3.5. `Distribution` Objects

The `LikelihoodEvaluator` object must compute the prior probability density function, $p(\vec{\vartheta}|H)$. There exists several `Distribution` objects that provide functions for evaluating the prior probability density function to use for each parameter, and for drawing random samples from these distributions. Currently, PyCBC Inference provides the following `Distributions`:

1. `Arbitrary` : reads a set of samples stored in a HDF format file and uses Gaussian kernel-density estimation (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html) to construct the distribution;
2. `CosAngle` : a cosine distribution;
3. `SinAngle` : a sine distribution;
4. `Gaussian` : a multivariate Gaussian distribution;
5. `Uniform` : a multidimensional uniform distribution;
6. `UniformAngle` : a uniform distribution between 0 and 2π;
7. `UniformLog` : a multidimensional distribution that is uniform in its logarithm;
8. `UniformPowerLaw` : a multidimensional distribution that is uniform in a power law;
9. `UniformSky` : a two-dimensional isotropic distribution; and
10. `UniformSolidAngle` : a two-dimensional distribution that is uniform in solid angle.

Multiple `Distribution` objects are needed to define the prior probability density function for all parameters. The `JointDistribution` object combines the individual prior probability density functions, providing a single interface for the `LikelihoodEvaluator` to evaluate the prior probability density function for all parameters. As the sampling algorithm advances the positions of the Markov chains, the `JointDistribution` computes the product of the prior probability density functions for the proposed new set of points in the parameter space. The `JointDistribution` can apply additional constraints on the prior probability density functions of parameters and it renormalizes the prior probability density function accordingly. If multiple constraints are provided, then the union of all constraints are applied. We demonstrate how to apply a cut on $\mathcal{M}$ and q obtained from the m_1 and m_2 prior probability density functions in the GW151226 configuration file in https://github.com/gwastro/pycbc-inference-paper (De et al. 2018).

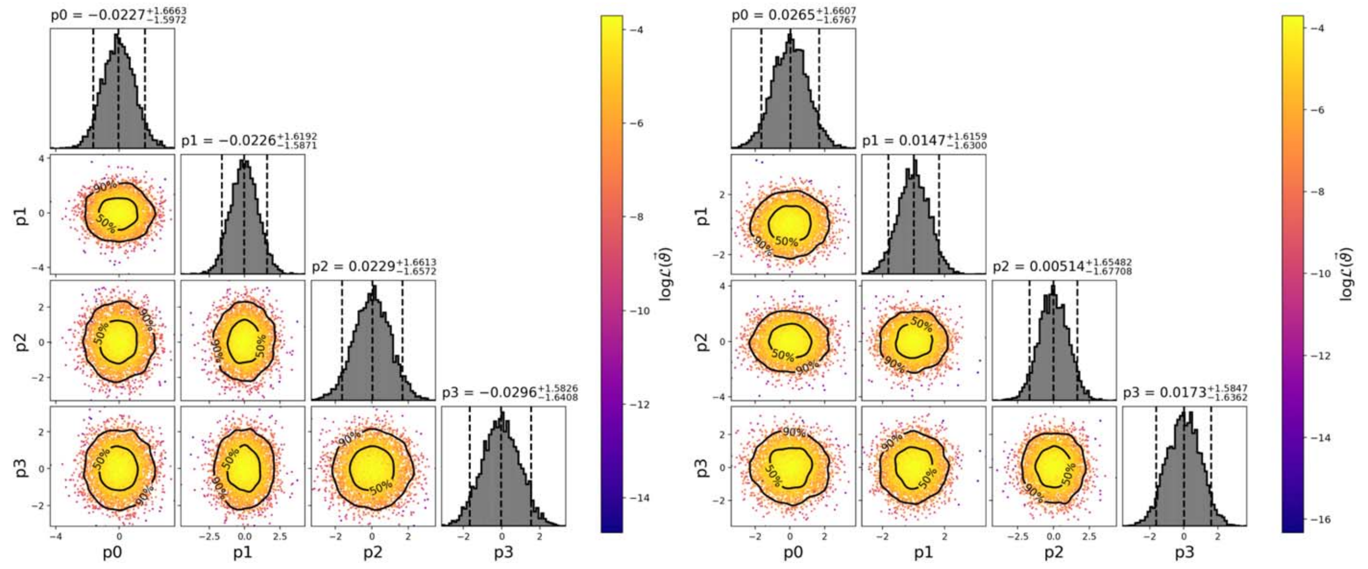


Figure 2. Samples of the posterior probability density function for a four-dimensional Gaussian distribution. Typically, these results are shown as a scatter plot matrix of independent samples. Here, the points in the scatter plot matrix are colored by the natural logarithm of the prior-weighted likelihood, $\log \mathcal{L}(\vec{\vartheta})$. At the top of each column is the marginalized one-dimensional histogram for a particular model parameter. In this case, each parameter, p_i, is the mean of a Gaussian in the range (0, 1). The median and 90% credible interval are superimposed on the marginalized histograms. Left: results obtained from the `emcee_pt` sampler. Right: results obtained from the `kombine` sampler.

(A color version of this figure is available in the online journal.)

3.6. `Generator` Objects

As part of the likelihood calculation described in Section 2.2, a waveform, $\tilde{s}_i(f, \vec{\vartheta})$, is generated from a waveform model, H, and set of parameters, $\vec{\vartheta}$. PyCBC Inference provides `Generator` objects that allow waveforms $\tilde{s}_i(f, \vec{\vartheta})$ to be generated for waveform models described in Section 2.1 using PyCBC's interface to LAL (Mercer et al. 2017). There are also `Generator` objects provided for generating ringdown waveforms as used in Cabero et al. (2018). Given the waveform model provided in the configuration file, `pycbc_inference` will automatically select the associated `Generator` object.

4. Validation of the Toolkit

PyCBC Inference includes tools for visualizing the results of parameter estimation, and several analytic functions that can be used to test the generation of known posterior probabilities. Two common ways to visualize results are a scatter plot matrix of the independent samples of the Markov chains, and marginalized one-dimensional histograms showing the bounds of each parameter's credible interval. Analytic likelihood functions available to validate the code include the multivariate normal, Rosenbrock, eggbox, and volcano functions. An example showing the visualization of results from a multivariate Gaussian test is shown in Figure 2. This figure was generated using the executable `pycbc_inference_plot_posterior`, which makes extensive use of tools from the open-source packages Matplotlib (Hunter 2007) and SciPy (Jones et al. 2001).

We can also validate the performance of PyCBC Inference by (i) determining if the inferred parameters of a population of simulated signals agrees with the known parameters of that population and (ii) comparing PyCBC Inference parameter measurements of astrophysical signals to the published LIGO–Virgo estimates that used a different inference code. In this section, we first check that the credible intervals match the probability of finding the simulated signal parameters in that interval; that is, that x% of signals should have parameter values in the x% credible interval. We then compare the recovered parameters of the binary black hole mergers GW150914, GW151226, and LVT151012 to those published in Abbott et al. (2016b). The validation tests and analyses presented in this section have been performed with the PyCBC v1.12.3 release.

4.1. Simulated Signals

To test the performance of PyCBC Inference, we generate 100 realizations of stationary Gaussian noise colored by power-spectral densities representative of the sensitivity of Advanced LIGO detectors at the time of the detection of GW150914 (Vallisneri et al. 2015). To each realization of noise we add a simulated signal whose parameters are drawn from the same prior probability density function used in the analysis of GW150914 (Abbott et al. 2016d), with an additional cut placed on distance to avoid having too many injections with low matched-filter S/N (Allen et al. 2012). The resulting injections have matched-filter S/Ns between 5 and 160, with the majority between ∼10 and ∼40. We then perform a parameter estimation analysis on each signal to obtain credible intervals on all parameters.

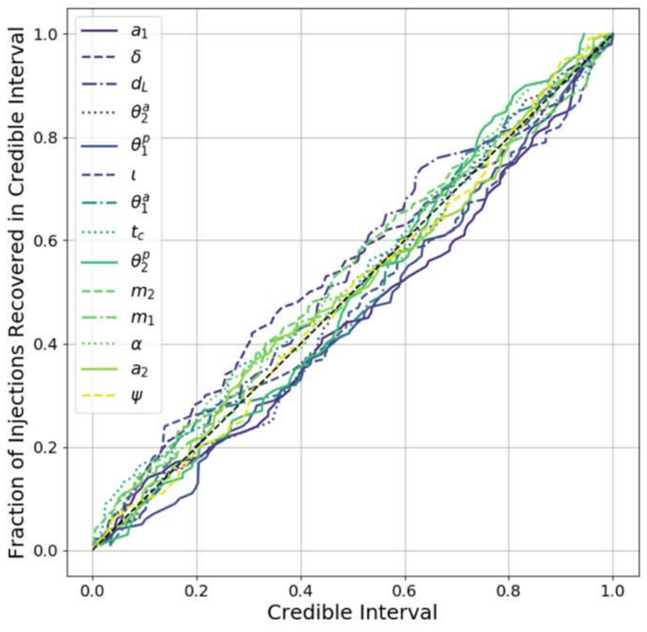
We perform this test using both the `emcee_pt` and `kombine` samplers. For the `emcee_pt` sampler, we use 200 Markov chains and 20 temperatures. We run the sampler until we obtain at least 2000 independent samples after the burn-in period as determined using the `n_acl` burn-in test. For the `kombine` sampler, we use 5000 Markov chains and the `max_posterior` burn-in test. As a result, we need only to run the `kombine` sampler until the burn-in test is satisfied, at which point we immediately have 5000 independent samples of the posterior probability density function.

Both simulated signals and the waveforms in the likelihood computation are generated using IMRPhenomPv2 (Schmidt et al. 2015; Hannam et al. 2014). This waveform model has 15 parameters. To reduce computational cost, we analytically marginalize over the fiducial phase, ϕ, by using Equation (6) for the posterior probability, thereby reducing the number of sampled parameters to 14. For each parameter, we count the number of times the simulated parameter falls within the measured credible interval.

Figure 3 summarizes the result of this test using the `emcee_pt` and `kombine` samplers. For each of the parameters we plot the fraction of signals whose true simulated values fall within specific values of credible intervals as a function of the corresponding credible interval values (this is referred to as a percentile-percentile plot). We expect the former to equal the latter for all parameters, though some fluctuation is expected due to noise. We see that all parameters follow a 1-to-1 relation, though the results from the `kombine` sampler have greater variance then the `emcee_pt` sampler.

To quantify the deviations seen in Figure 3, we perform a Kolmogorov–Smirnov (KS) test on each parameter to see whether the percentile-percentile curves match the expected 1-to-1 relation. If the samplers and code are performing as expected, then these p-values should in turn follow a uniform distribution. We therefore perform another KS test on the collection of p-values, obtaining a two-tailed p-value of 0.7 for the `emcee_pt` sampler and 0.01 for the `kombine` sampler. In other words, if the `emcee_pt` sampler provides an unbiased estimate of the parameters, then there is a ∼70% chance that we would obtain a collection of percentile-percentile curves more extreme than seen in Figure 3. For the `kombine` sampler, the probability of obtaining a more extreme collection of curves than that seen in Figure 3 is only ∼1%.

A low p-value indicates that the two distributions in the KS test differ. Therefore, based on these results, we have confidence that PyCBC Inference provides unbiased estimates of binary black hole parameters when used with the `emcee_pt` sampler and the above settings. The `kombine` sampler does not appear to provide unbiased parameter estimates when used to sample the full parameter space of precessing binary black holes with the settings we have used.



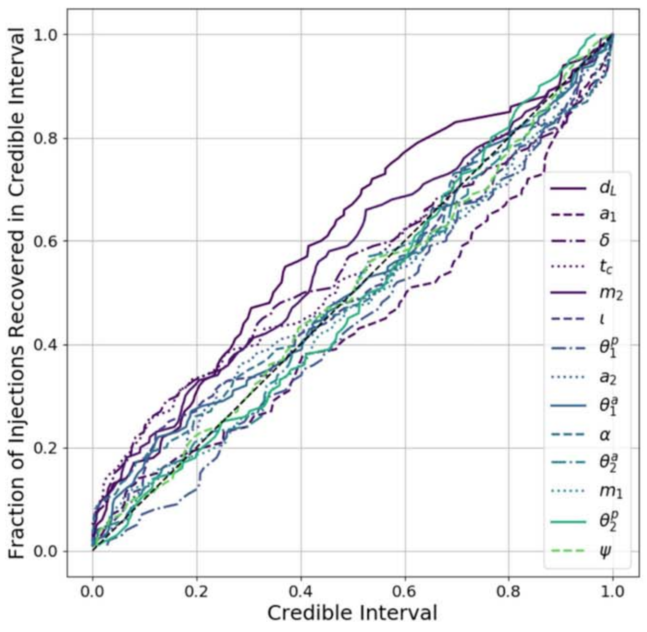


Figure 3. Fraction of simulated signals with parameter values within a credible interval as a function of credible interval. Plotted are all 14 parameters varied in the MCMC analyses. The diagonal line indicates the ideal 1-to-1 relation that is expected if the samplers provide unbiased estimates of the parameters. We perform a Kolmogorov–Smirnov (KS) test on each parameter to obtain two-tailed p-value indicating the consistency between the curves and the diagonal line. Top: results using the `emcee_pt` sampler. Bottom: results using the `kombine` sampler.
(A color version of this figure is available in the online journal.)

Table 1
Results from PyCBC Inference Analysis of GW150914, GW151226, and LVT151012. Quoted are the Median and 90% Credible Interval Values for the Parameters of Interest. Interpretations of these Results are Summarized in Section 4.2

Parameter	GW150914	GW151226	LVT151012
$\mathcal{M}^{\rm det}$	$31.03^{+1.555}_{-1.527}\,M_\odot$	$9.71^{+0.059}_{-0.058}\,M_\odot$	$18.09^{+1.04}_{-0.782}\,M_\odot$
$m_1^{\rm det}$	$38.9^{+5.5}_{-3.3}\,M_\odot$	$15.0^{+8.2}_{-3.3}\,M_\odot$	$27.3^{+16.7}_{-5.9}\,M_\odot$
$m_2^{\rm det}$	$32.9^{+3.2}_{-4.9}\,M_\odot$	$8.5^{+2.3}_{-2.5}\,M_\odot$	$16.3^{+4.3}_{-5.8}\,M_\odot$
$\mathcal{M}^{\rm src}$	$28.18^{+1.59}_{-1.38}\,M_\odot$	$8.86^{+0.302}_{-0.246}\,M_\odot$	$14.99^{+1.302}_{-0.964}\,M_\odot$
$m_1^{\rm src}$	$35.4^{+5.2}_{-3.1}\,M_\odot$	$13.7^{+7.4}_{-3.1}\,M_\odot$	$22.6^{+14.3}_{-5.1}\,M_\odot$
$m_2^{\rm src}$	$29.9^{+3.0}_{-4.4}\,M_\odot$	$7.7^{+2.1}_{-2.3}\,M_\odot$	$13.5^{+3.8}_{-4.8}\,M_\odot$
q	$1.18^{+0.39}_{-0.16}$	$1.76^{+2.15}_{-0.69}$	$1.68^{+2.51}_{-0.63}$
$\chi_{\rm eff}$	$-0.0324^{+0.11}_{-0.12}$	$0.1986^{+0.17}_{-0.07}$	$0.0046^{+0.25}_{-0.16}$
a_1	$0.31^{+0.57}_{-0.28}$	$0.53^{+0.40}_{-0.46}$	$0.33^{+0.54}_{-0.3}$
a_2	$0.31^{+0.57}_{-0.29}$	$0.52^{+0.41}_{-0.45}$	$0.36^{+0.53}_{-0.32}$
d_L	493^{+125}_{-196} Mpc	461^{+157}_{-181} Mpc	1062^{+454}_{-458} Mpc

4.2. Astrophysical Events

In this section, we present PyCBC Inference measurements of properties of the binary black hole sources of the two gravitational-wave signals GW150914 and GW151226, and the third gravitational-wave signal LVT151012 consistent with the properties of a binary black hole source from Advanced LIGO's first observing run (Abbott et al. 2016a, 2016b). We perform the parameter estimation analysis on the Advanced LIGO data available for these events at the LIGO Open Science Center (Vallisneri et al. 2015). We use the `emcee_pt` sampler for these analyses. For computing the likelihood, we analyze the gravitational-wave data set, $\vec{d}\,(t)$, from the Hanford and Livingston detectors. $\vec{d}\,(t)$ in our analyses are taken from GPS time intervals 1126259452 to 1126259468 for GW150914, 1135136340 to 1135136356 for GW151226, and 1128678874 to 1128678906 for LVT151012. Detection of gravitational waves from the search pipeline (Nitz et al. 2018; Usman et al. 2016; Dal Canton et al. 2014; Nitz et al. 2017; Abbott et al. 2016c) gives initial estimates of the mass, hence estimates of the length of the signal. From results of the search, LVT151012 was a longer signal with more cycles than the other two events, and LVT151012 had characteristics that were in agreement with a lower mass source than GW150914 and GW151226. Therefore, more data is required for the analysis of LVT151012. The PSD used in the likelihood is constructed using the median PSD estimation method described in Allen et al. (2012) with 8 s Hann-windowed segments (overlapped by 4 s) taken from GPS times 1126258950 to 1126259974 for GW150914, 1135136238 to 1135137262 for GW151226, and 1128678388 to 1128679412 for LVT151012. The PSD estimate is truncated to 4 s in the time domain using the method described in Allen et al. (2012). The data set is sampled at 2048 Hz, and the likelihood is evaluated between a low-frequency cutoff of 20 Hz and 1024 Hz.

The waveforms $\tilde{s}_i(f, \vec{\vartheta})$ used in the likelihood are generated using the IMRPhenomPv2 (Schmidt et al. 2015; Hannam et al. 2014) model implemented in the LIGO Algorithm Library (LAL; Mercer et al. 2017). The parameters measured in the MCMC for these three events are $\vec{\vartheta} = \{\alpha, \delta, \psi, m_1, m_2, d_L, \iota, t_c, a_1, a_2, \theta_1^a, \theta_2^a, \theta_1^p, \theta_2^p\}$, and we analytically marginalize over the fiducial phase, ϕ. These parameters form the complete set of parameters to construct a waveform from a binary black hole merger, and are the same parameters that were inferred from the parameter estimation analyses in Abbott et al. (2016b). As faster convergence of m_1 and m_2 can be obtained with mass parameterizations of the waveform in $\mathcal{M}$ and q we perform the coordinate transformation from (m_1, m_2) to $(\mathcal{M}, q)$ before evaluating the priors.

We assume uniform prior distributions for the binary component masses $m_{1,2} \in [10, 80]\,M_\odot$ for GW150914, $m_{1,2} \in [5, 80]\,M_\odot$ for LVT151012, and $m_{1,2}$ corresponding to chirp mass $\mathcal{M} \in [9.5, 10.5]\,M_\odot$ and mass ratio $q \in [1, 18]$ for GW151226. We use uniform priors on the spin magnitudes $a_{1,2} \in [0.0, 0.99]$. We use a uniform solid angle prior for the spin angles, where $\theta_{1,2}^a$ is a uniform distribution $\theta_{1,2}^a \in [0, 2\pi)$, and $\theta_{1,2}^p$ is a sine-angle distribution. For the luminosity distance, we use a uniform in volume prior with $d_L \in [10, 1000]$ Mpc for GW150914, $d_L \in [10, 1500]$ Mpc for GW151226, and $d_L \in [10, 2500]$ Mpc for LVT151012. We use uniform priors for the arrival time $t_c \in [t_s - 0.2\,s, t_s + 0.2\,s]$, where t_s is the trigger time for the particular event obtained from the gravitational-wave search (Abbott et al. 2016b, 2016c). For the sky location parameters, we use a uniform distribution prior for $\alpha \in [0, 2\pi)$ and a cosine-angle distribution prior for δ. The priors described above are the same as those used in Abbott et al. (2016b).

The parameter estimation analysis produces distributions that are a sampling of the posterior probability density function for the variable parameters from the ensemble MCMC. We map these distributions obtained directly from the analysis to obtain estimates of other parameters of interest such as the chirp mass, $\mathcal{M}$, mass ratio, q, effective spin, $\chi_{\rm eff}$, and the precession spin, χ_p (Schmidt et al. 2015) parameters. We use d_L to relate the detector-frame masses obtained from the MCMC to the source-frame masses using the standard Λ-CDM cosmology (Schutz 1986; Finn & Chernoff 1993).

Recorded in Table 1, is a summary of the median and 90% credible interval values calculated for GW150914, GW151226, and LVT151012 analyses. Results for $m_1^{\rm src} - m_2^{\rm src}$, $q - \chi_{\rm eff}$, and $d_L - \iota$ are shown in Figures 4, 5, and 6 for GW150914, GW151226, and LVT151012, respectively. The two-dimensional plots in these figures show the 50% and 90% credible

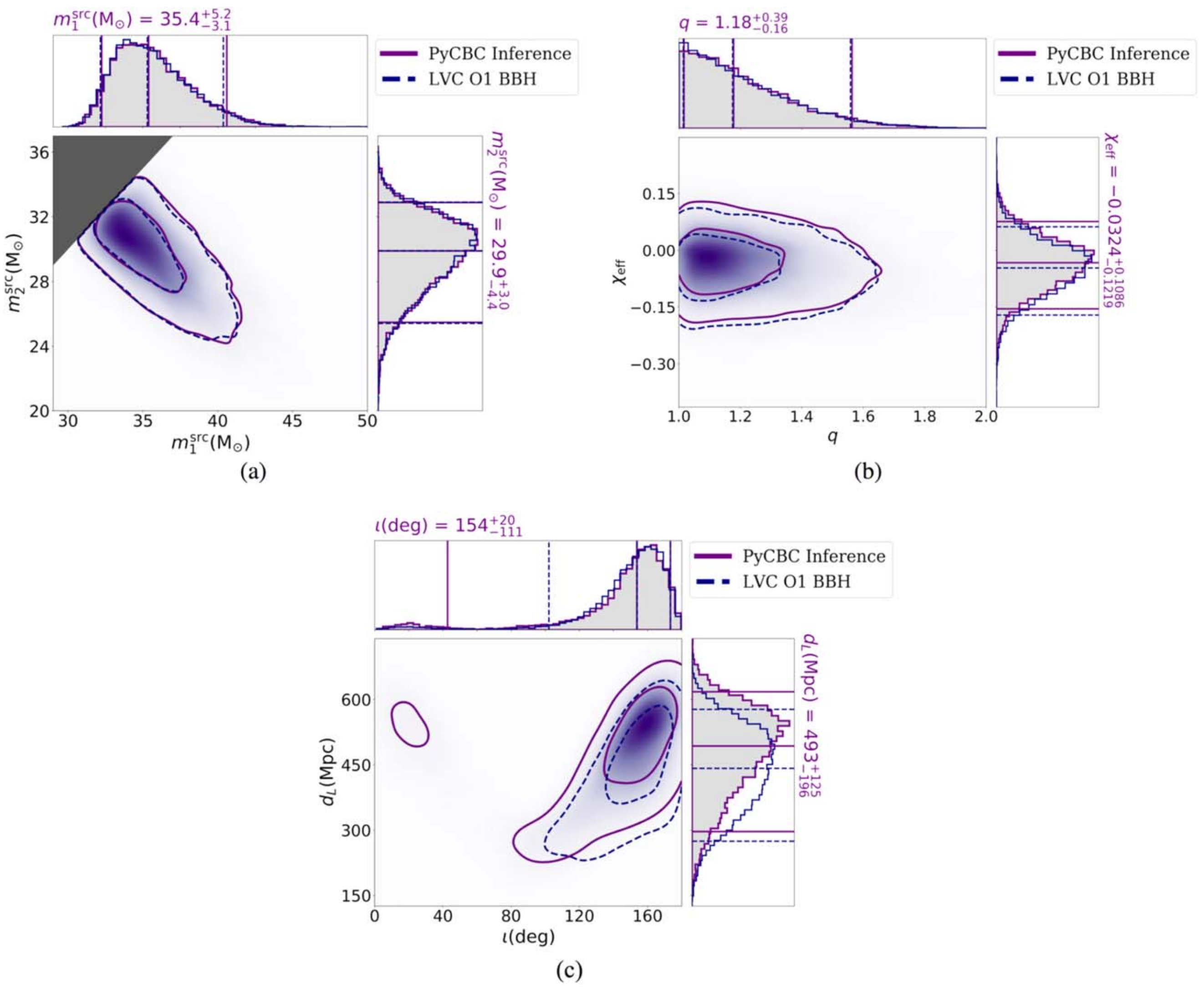
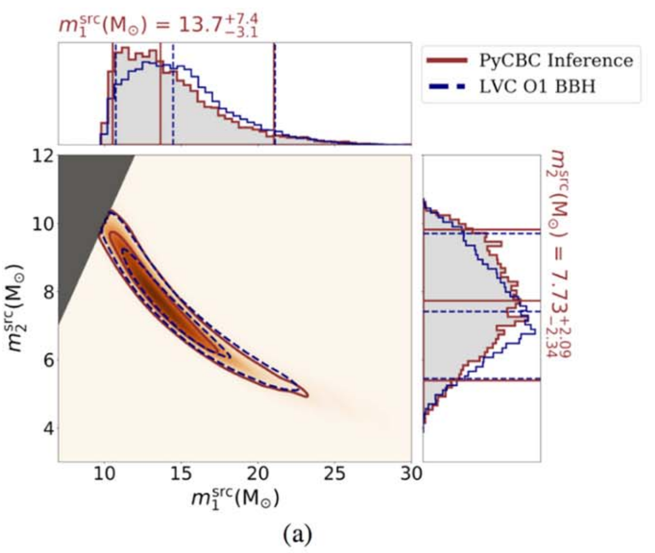
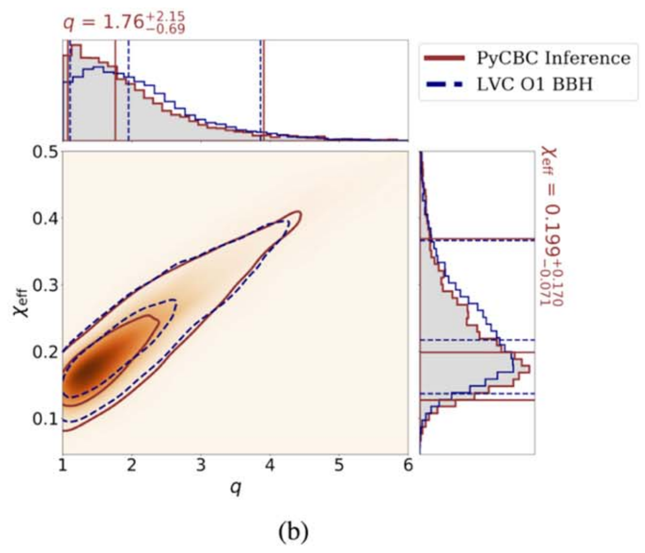


Figure 4. Posterior probability densities for the main parameters of interest from the PyCBC Inference analysis of GW150914. The parameters plotted are (a) $m_1^{\rm src} - m_2^{\rm src}$, (b: $q - \chi_{\rm eff}$, and (c) $d_L - \iota$. The bottom-left panel in each of (a), (b), and (c) show two-dimensional probability densities with 50% and 90% credible contour regions from the PyCBC Inference posteriors. The top-left and the bottom-right panels in each figure show one-dimensional posterior distributions for the individual parameters with solid lines at the 5%, 50%, and 95% percentiles. For comparison, we also show 50% and 90% credible regions, one-dimensional posterior probabilities with dashed lines at 5%, 50%, and 95% percentiles using the posterior samples obtained from the LIGO Open Science Center (Vallisneri et al. 2015) for the GW150914 analysis reported in Abbott et al. (2016b), using the IMRPhenomPv2 model. The measurements show that masses for GW150914 are much better constrained as compared with the other parameters presented. Though there is support for the system being both face-on and face-off, there seems to be slightly more preference for a face-off system. The posteriors suggest a preference for lower spins. Our measurements are in agreement with the results presented in Abbott et al. (2016b) within the statistical errors of measurement of the parameters.

(A color version of this figure is available in the online journal.)

regions, and the one-dimensional marginal distributions show the median and 90% credible intervals. Overlaid are the one-dimensional marginal distributions, median, and 90% credible intervals, as well as the 50% and 90% credible regions using the samples obtained from the LIGO Open Science Center (Vallisneri et al. 2015) for the analyses of the three events reported in Abbott et al. (2016b) using the IMRPhenomPv2 model. The results show that GW150914 has the highest mass components among the three events. GW150914 has more support for equal mass ratios, whereas the GW151226 and LVT151012 posteriors support more asymmetric mass ratios. Overall, there is preference for smaller spins, with GW151226 having the highest spins among the three events. While the inclination and luminosity distances are not very well constrained, with generally a support for both face-on ($\iota = 0$, line of sight parallel to binary angular momentum) and face-off





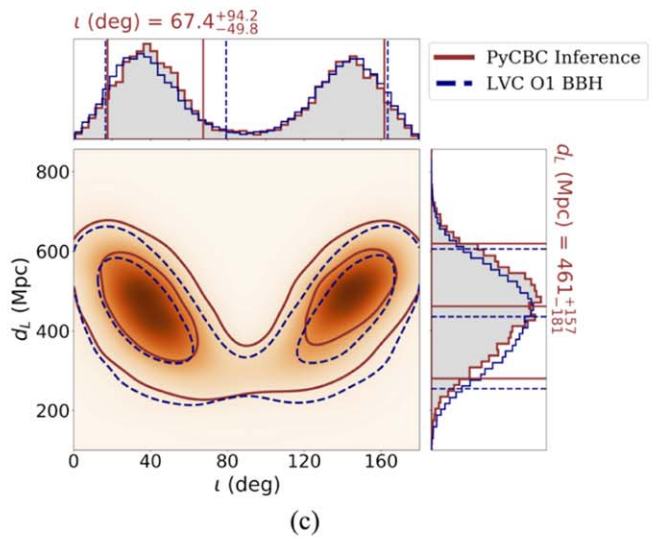


Figure 5. Posterior probability densities for the main parameters of interest from the PyCBC Inference analysis of GW151226. The parameters plotted are (a) $m_1^{src} - m_2^{src}$, (b) $q - \chi_{eff}$, and (c) $d_L - \iota$. The bottom-left panel in each of (a), (b), and (c) show two-dimensional probability densities with 50% and 90% credible contour regions from the PyCBC Inference posteriors. The top-left and the bottom-right panels in each figure show one-dimensional posterior distributions for the individual parameters with solid lines at the 5%, 50%, and 95% percentiles. For comparison, we also show 50% and 90% credible regions, one-dimensional posterior probabilities with dashed lines at 5%, 50%, and 95% percentiles using the posterior samples obtained from the LIGO Open Science Center (Vallisneri et al. 2015) for the GW151226 analysis reported in Abbott et al. (2016b) using the IMRPhenomPv2 model. The measurements show that GW151226 is the lowest mass and fastest spinning binary among the three O1 events presented in this work. The posteriors support asymmetric mass ratios. Inclination ι and distance d_L are not well constrained, and there is support for the system being both face-on and face-off. Our measurements are in agreement with the results presented in Abbott et al. (2016b) within the statistical errors of measurement of the parameters.

(A color version of this figure is available in the online journal.)

($\iota = \pi$, line of sight anti-parallel to binary angular momentum) systems for all the three events, GW150914 seems to have more preference for face-off systems. We also computed χ_p for each of the three events and found no significant measurements of precession. Overall, our results are in agreement with those presented in Abbott et al. (2016b) within the statistical errors of measurement.

5. Conclusions

In this paper we have described PyCBC Inference, a Python-based toolkit with a simplified interface for parameter estimation studies of compact-object binary mergers. We have used this toolkit to estimate the parameters of the gravitational-wave events GW150914, GW151226, and LVT151012; our results are consistent with previously published values. In these

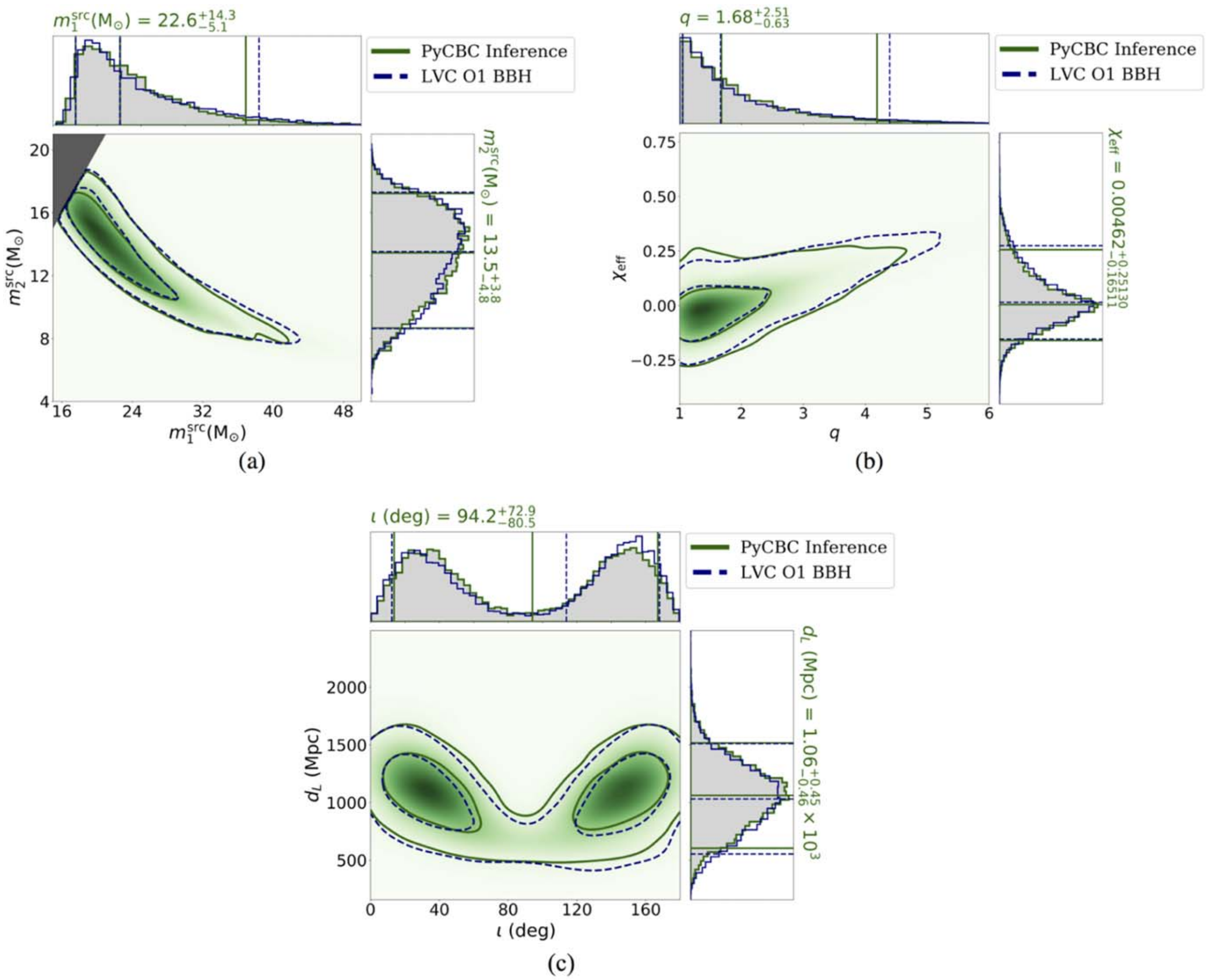


Figure 6. Posterior probability densities for the main parameters of interest from the PyCBC Inference analysis of LVT151012. The parameters plotted are (a) $m_1^{src} - m_2^{src}$, (b) $q - \chi_{eff}$, and (c) $d_L - \iota$. The bottom-left panel in each of (a), (b), and (c) show two-dimensional probability densities with 50% and 90% credible contour regions from the PyCBC Inference posteriors. The top-left and the bottom-right panels in each figure show one-dimensional posterior distributions for the individual parameters with solid lines at the 5%, 50%, and 95% percentiles. For comparison, we also show 50% and 90% credible regions, one-dimensional posterior probabilities with dashed lines at 5%, 50%, and 95% percentiles using the posterior samples obtained from the LIGO Open Science Center (Vallisneri et al. 2015) for the LVT151012 analysis reported in Abbott et al. (2016b) using the IMRPhenomPv2 model. The measurements again show that the spins, inclination, and distance are not very well constrained and there is support for the system being both face-on and face-off. Our measurements are in agreement with the results presented in Abbott et al. (2016b) within the statistical errors of measurement of the parameters.

(A color version of this figure is available in the online journal.)

analyses, we do not marginalize over calibration uncertainty of the measured strain in our results, which was included in prior work (for example Abbott et al. 2016b, 2016d, 2016e). We will implement this in PyCBC Inference in the future. We have made the samples of the posterior probability density function from the PyCBC Inference analysis of all three events available in https://github.com/gwastro/pycbc-inference-paper (De et al. 2018) along with the instructions and configuration files needed to replicate these results. The source code and documentation for PyCBC Inference is available as part of the PyCBC software package at http://pycbc.org.

PyCBC Inference has already been used to produce several astrophysical results: (i) a test of the black hole area increase law (Cabero et al. 2018); (ii) measuring the viewing angle of GW170817 with electromagnetic and gravitational-wave signals (Finstad et al. 2018); and (iii) measuring the tidal deformabilities and radii of neutron stars from the observation of GW170817 (De et al. 2018). The results presented in this paper and in the studies above demonstrate the capability of PyCBC Inference to perform gravitational-wave parameter estimation analyses. Future developments under consideration are implementation of models to marginalize over calibration errors, generic algorithms to perform model selection, HPD to compute credible intervals, methods for faster computation of the likelihood, and more independent samplers for cross-comparison and validation of results for parameter estimation and model selection.

The authors thank Will Farr and Ben Farr for valuable insights into the intricacies of ensemble MCMCs. We also thank Ian Harry, Christopher Berry, and Daniel Wysocki for helpful comments on the manuscript. This work was supported by NSF awards PHY-1404395 (DAB, CMB), PHY-1707954 (DAB, SD), and PHY-1607169 (SD). S.D. was also supported by the Inaugural Kathy '73 and Stan '72 Walters Endowed Fund for Science Research graduate fellowship at Syracuse University. Computations were supported by Syracuse University and NSF award OAC-1541396. We also acknowledge the Max Planck Gesellschaft for support and the Atlas cluster computing team at AEI Hannover. D.A.B. thanks the École de Physique des Houches for hospitality during the completion of this manuscript. The authors thank the LIGO Scientific Collaboration for access to the data and acknowledge the support of the United States National Science Foundation (NSF) for the construction and operation of the LIGO Laboratory and Advanced LIGO, as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, and the Max-Planck-Society (MPS) for support of the construction of Advanced LIGO. Additional support for Advanced LIGO was provided by the Australian Research Council. This research has made use of data obtained from the Gravitational Wave Open Science Center: https://www.gw-openscience.org.

References

Aasi, J., et al. (LIGO Scientific) 2015, CQGra, 32, 074001
Abbott, B., et al. (Virgo, LIGO Scientific) 2017a, PhRvL, 119, 161101
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2016a, PhRvL, 116, 061102
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2016b, PhRvX, 6, 041015
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2016c, PhRvD, 93, 122003
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2016d, PhRvL, 116, 241102
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2017b, ApJ, 851, L35
Abbott, B. P., et al. (Virgo, LIGO Scientific) 2017c, PhRvL, 119, 141101
Abbott, B. P., et al. (VIRGO, LIGO Scientific) 2017d, PhRvL, 118, 22110
Abbott, T. D., et al. (Virgo, LIGO Scientific) 2016e, PhRvX, 6, 041014
Acernese, F., et al. (VIRGO) 2015, CQGra, 32, 024001
Ajith, P., Babak, S., Chen, Y., et al. 2007, CQGra, 24, S689
Ajith, P., Hannam, M., Husa, S., et al. 2011, PhRvL, 106, 241101
Allen, B., Anderson, W. G., Brady, P. R., Brown, D. A., & Creighton, J. D. E. 2012, PhRvD, 85, 122006
Bayes, M., & Price, M. 1763, Philosophical Transactions (1683–1775), http://doi.org/10.1098/rstl.1763.0053
Berti, E., Cardoso, V., & Starinets, A. O. 2009, CQGra, 26, 163001
Blanchet, L. 2006, LRR, 9, 4
Buonanno, A., & Damour, T. 1999, PhRvD, 59, 084006
Buonanno, A., & Damour, T. 2000, PhRvD, 62, 064015
Cabero, M., Capano, C. D., Fischer-Birnholtz, O., et al. 2018, PhRvD, 97, 124069
Cardoso, V., Gualtieri, L., Herdeiro, C., & Sperhake, U. 2015, LRR, 18, 1
Chen, M. H., Shao, Q. M., & Ibrahim, J. G. 2000, Computing Bayesian Credible and HPD Intervals (New York: Springer)
Christensen, N., Dupuis, R. J., Woan, G., & Meyer, R. 2004, PhRvD, 70, 022001
Christensen, N., Libson, A., & Meyer, R. 2004, CQGra, 21, 317
Christensen, N., & Meyer, R. 2001, PhRvD, 64, 022001
Collette, A., Tocknell, J., Caswell, T. A., et al. 2018, h5py/h5py 2.8.0, Zenodo: doi:10.5281/zenodo.1246321
Cutler, C., & Flanagan, E. E. 1994, PhRvD, 49, 2658
Dal Canton, T., Nitz, A. H., Lundgren, A. P., et al. 2014, PhRvD, 90, 082004
Dalcín, L., Paz, R., & Storti, M. 2005, J. Parallel Distrib. Comput., 65, 1108
Dalcin, L. D., Paz, R. R., Kler, P. A., & Cosimo, A. 2011, AdWR, 34, 1124
Dalcín, Lisandro., Paz, R., Storti, M., & D'Elia, J. 2008, J. Parallel Distrib. Comput., 68, 655
Damour, T. 2001, PhRvD, 64, 124013
Damour, T., Jaranowski, P., & Schaefer, G. 2000, PhRvD, 62, 084011
De., S. 2018, gwastro/pycbc-inference-paper: v2.0 data releasefor the PyCBC Inference paper, https://doi.org/10.5281/zenodo.2009049
De, S., Finstad, D., Lattimer, J. M., et al. 2018, PhRvL, 121, 091102
Farr, B., Berry, C. P. L., Farr, W. M., et al. 2016, ApJ, 825, 116
Farr, B., & Farr, W. M. 2015, In prep
Farr, B., & Farr, W. M. 2018, https://github.com/bfarr/kombine
Finn, L. S., & Chernoff, D. F. 1993, PhRvD, 47, 2198
Finstad, D., De, S., Brown, D. A., Berger, E., & Biwer, C. M. 2018, ApJL, 860, L2
Flanagan, E. E., & Hinderer, T. 2008, PhRvD, 77, 021502
Foreman-Mackey, D., Farr, W. M, Hogg, D. W., et al. 2018, dfm/emcee: emcee v3.0rc1, Zenodo, doi:10.5281/zenodo.1297477
Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306
Frigo, M., & Johnson, S. G. 2005, in IEEE Program Generation, Optimization, and Platform Adaptation (Piscataway, NJ: IEEE), 216
Gelman, A., & Meng, X. 1998, StaSc, 13, 163, http://www.jstor.org/stable/2676756
Gelman, A., Robert, C., Chopin, N., & Rousseau, J. 1995, Bayesian data analysis
Geman, S., & Geman, D. 1984, in IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6 (Piscataway, NJ: IEEE), 721
Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. 1996, Markov Chain Monte Carlo in Practice (London: Chapman and Hall)
Hannam, M., Brown, D. A., Fairhurst, S., Fryer, C. L., & Harry, I. W. 2013, ApJ, 766, L14
Hannam, M., Schmidt, P., Bohé, A., et al. 2014, PhRvL, 113, 151101
Hastings, W. K. 1970, Biometrika, 57, 97
Hinderer, T. 2008, ApJ, 677, 1216
Hunter, J. D. 2007, CSE, 9, 90
Jaynes, E. T. 2003, Probability Theory: The Logic of Science (Cambridge: Cambridge Univ. Press)
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open source scientific tools for Python, http://www.scipy.org/
Kass, R. E., & Raftery, A. E. 1995, J. Am. Stat. Assoc., 90, 773
Lackey, B. D., Bernuzzi, S., Galley, C. R., Meidam, J., & Van Den Broeck, C. 2017, PhRvD, 95, 104036
Madras, N., & Sokal, A. D. 1988, JSP, 50, 109
Mercer, R. A., et al. (Virgo, LIGO Scientific) 2017, LIGO Algorithm Library, https://git.ligo.org/lscsoft/lalsuite
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. 1953, J. Chem. Phys., 21, 1087
Nicholson, D., & Vecchio, A. 1998, PhRvD, 57, 4588
Nitz, A., Harry, I., Brown, D. A., et al. 2018, PyCBC v1.12.3, zenodo, doi:10.5281/zenodo.1410598
Nitz, A. H., Dent, T., Dal Canton, T., Fairhurst, S., & Brown, D. A. 2017, ApJ, 849, 118
Peters, P. C. 1964, PhRv, 136, B1224

Peters, P. C., & Mathews, J. 1963, PhRv, 131, 435
Pürrer, M. 2016, PhRvD, 93, 064041
Rover, C., Meyer, R., & Christensen, N. 2006, CQGra, 23, 4895
Rover, C., Meyer, R., & Christensen, N. 2007, PhRvD, 75, 062004
Santamaria, L., Ohme, F., Ajith, P., et al. 2010, PhRvD, 82, 064016
Schmidt, P., Ohme, F., & Hannam, M. 2015, PhRvD, 91, 024043
Schutz, B. F. 1986, Natur, 323, 310
Skilling, J. 2006, BayAn, 1, 833
Tannenbaum, T., Wright, D., Miller, K., & Livny, M. 2001, in Condor—a Distributed Job Scheduler, Beowulf Cluster Computing with Linux, ed. T. Sterling (Cambridge, MA: MIT Press)
Teukolsky, S. A. 1972, PhRvL, 29, 1114
Thain, D., Tannenbaum, T., & Livny, M. 2005, Concurrency and Computation: Practice and Experience, 17, 323
Thorne, K. S. 1987, in Three Hundred Years of Gravitation, ed. S. W. Hawking & W. Israel (Cambridge: Cambridge Univ. Press), 330
Usman, S. A., Nitz, A. H., Harry, I. W., et al. 2016, CQGra, 33, 215004
Vallisneri, M., Kanner, J., Williams, R., Weinstein, A., & Stephens, B. 2015, JPCS, 610, 012021
Veitch, J., Raymond, V., Farr, B., et al. 2015, PhRvD, 91, 042003
Vousden, W. D., Farr, W. M., & Mandel, I. 2016, MNRAS, 455, 1919
Wahlquist, H. 1987, GReGr, 19, 1101
Wainstein, L. A., & Zubakov, V. D. 1962, Extraction of Signals from Noise (Englewood Cliffs, NJ: Prentice-Hall)