

Parameter estimation of spinning binary inspirals using Markov-chain Monte Carlo

Marc van der Sluys¹, Vivien Raymond^{1,2}, Ilya Mandel¹, Christian Röver^{3,4}, Nelson Christensen⁵, Vicky Kalogera¹, Renate Meyer³, Alberto Vecchio^{1,6}

¹ Physics & Astronomy, Northwestern University, Evanston IL, USA

² Université Louis Pasteur - Strasbourg I, Strasbourg, France

³ Statistics, University of Auckland, New Zealand

⁴ Max-Planck-Institut für Gravitationsphysik, Hannover, Germany

⁵ Physics & Astronomy, Carleton College, Northfield MN, USA

⁶ Physics & Astronomy, U. of Birmingham, Edgbaston, Birmingham, UK

E-mail: sluys@northwestern.edu

Abstract. We present a Markov-chain Monte-Carlo (MCMC) technique to study the source parameters of gravitational-wave signals from the inspirals of stellar-mass compact binaries detected with ground-based gravitational-wave detectors such as LIGO and Virgo, for the case where spin is present in the more massive compact object in the binary. We discuss aspects of the MCMC algorithm that allow us to sample the parameter space in an efficient way. We show sample runs that illustrate the possibilities of our MCMC code and the difficulties that we encounter.

PACS numbers: 02.50.-r, 02.70.Uu, 04.30.Tv, 04.80.Nn, 95.85.Sz

Submitted to: *Classical and Quantum Gravity*

1. Introduction

Inspirals of stellar-mass compact binaries induced by gravitational radiation are among the most promising gravitational-wave sources for ground-based laser interferometers, such as LIGO [1; 2] and Virgo [3]. If such a binary contains a black hole, it is believed to be spinning moderately [4]. A spinning black hole causes the binary orbit to precess, introducing phase and amplitude modulations in the gravitational-wave signal. This should be taken into account in the analysis of the signal. The accuracy with which the binary parameters can be estimated is of significant astrophysical interest.

We developed a code which implements a Markov-chain Monte-Carlo (MCMC) technique [5] to compute the posterior probability-density functions (PDFs) of the source parameters. This code is a modification of an earlier parameter-estimation code for analysis on binaries without spin [6; 7]. In addition to including post-Newtonian gravitational waveforms with a single spinning object [8], we have also implemented a number of improvements designed to make the parameter-space exploration more efficient, such as parallel tempering.

This paper is organised as follows. In section 2, we describe the implementation of the MCMC algorithm in the code and related details, such as data handling, waveform choice,

update proposals, and parallel tempering. Section 3 contains a discussion of some sample runs obtained with our MCMC code. In section 4 we summarise our conclusions, comment on existing issues, and describe planned future improvements.

2. Implementation of MCMC

The code we use to estimate the parameters of a binary inspiral with a spinning member is based on an earlier code that was used by some of us for the case where no spin is present [7]. In this section, we describe some of the features that were taken from the earlier non-spinning MCMC code, as well as some features that were introduced in the present code for use on inspirals with a single spinning object.

2.1. Data handling

For this study, we inject simulated waveforms with parameters of our choice into a stretch of simulated Gaussian, stationary noise at the designed sensitivity level for the detectors [9]. The resulting data is windowed, Fourier transformed, and subsequently examined by the MCMC analysis. The details of the data handling, Fourier transformation, windowing and PSD estimation can be found in [7].

A stretch of 256 seconds of simulated noise data is used to estimate the power-spectral density (PSD) of the noise $S_n(f)$. The signal-to-noise ratio (SNR) of an injected model signal with parameters $\vec{\lambda}$ as detected by a single detector i is computed as follows:

$$\rho_i(\vec{\lambda}) = \sqrt{4 \sum_{f=f_{\text{low}}}^{f_{\text{high}}} \frac{|\tilde{m}(\vec{\lambda}, f)|^2}{S_n(f)} \Delta f}, \quad (1)$$

where $\tilde{m}(\vec{\lambda}, f)$ is the frequency-domain model waveform, $S_n(f)$ is the noise PSD, the sum is computed over the frequency bins between f_{low} and f_{high} and Δf is the width of each frequency bin. The total SNR for a network of N detectors is then given by:

$$\rho_{\text{tot}}(\vec{\lambda}) = \sqrt{\sum_{i=1}^N (\rho_i(\vec{\lambda}))^2}. \quad (2)$$

2.2. Waveform

In order to test our models, we use a simplified waveform that takes into account post-Newtonian (PN) expansions up to the 1.5 PN order in phase and is restricted to the Newtonian order in amplitude. The waveform includes the simple-precession prescription [8]. This choice of waveform template allows us to investigate the first-order effects of spin (spin-orbit coupling), as long as either only one binary member has spin, or the mass ratio is roughly equal to unity. In comparison to higher-PN or double-spin waveforms, this waveform can be computed analytically and has lower dimensionality, so that the computational cost per iteration is lower and the number of iterations needed for convergence is smaller. In this paper, we focus on a fiducial binary consisting of a $10 M_\odot$ spinning black hole and a $1.4 M_\odot$ non-spinning neutron star. Such masses are astrophysically likely [4], and the mass ratio allows us to ignore the typical spin of a neutron star, so that the simple-precession prescription is valid.

The waveform for an inspiral with one spinning object is described by twelve parameters. The parameters are: the chirp mass M_c , symmetric mass ratio η , spin magnitude $a_{\text{spin}} \equiv$

S/M^2 , the constant angle between spin and orbital angular momentum θ_{SL} , the luminosity distance d_L and sky position R.A., Dec., the time, orbital phase and precession phase at coalescence t_c , ϕ_c , α_c , and two angles that define the direction of the total angular-momentum \vec{J}_0 of the binary: θ_{J_0} and ϕ_{J_0} .

Each waveform template is computed in the time domain, and then windowed and Fourier transformed. The calculation of the likelihood, which measures how well a model waveform matches the data, is carried out in the frequency domain.

In this initial study, we focus on waveforms for our fiducial binary ($M_c \approx 3.0M_\odot$, $\eta \approx 0.11$) at a distance of $d_L = 13.0$ Mpc. We analyse the effect of spin in an initial parameter study [10]. The sky position and orientation of the binary are fixed in this study, while the SNR ranges between 12 and 22 per detector, depending on the spin.

2.3. Computation of the likelihood

We follow a Bayesian approach to infer the posterior probability-density functions (PDFs) of the twelve parameters that describe our waveform. The PDF of a parameter vector $\vec{\lambda}$ given an observed data set d follows from Bayes' theorem:

$$p(\vec{\lambda}|d) = \frac{p(\vec{\lambda})p(d|\vec{\lambda})}{p(d)} \propto p(\vec{\lambda})L(d|\vec{\lambda}), \quad (3)$$

where $p(\vec{\lambda})$ is the *prior* distribution of the parameters, and $L(d|\vec{\lambda})$ is the *likelihood* function. We calculate the likelihood for a model waveform $\tilde{m}(\vec{\lambda}, f)$ with parameters $\vec{\lambda}$ and data set $\tilde{d}(f)$ as measured by a detector i in the usual way:

$$L_i(d|\vec{\lambda}) \propto \exp \left(-2 \int_0^\infty \frac{|\tilde{d}(f) - \tilde{m}(\vec{\lambda}, f)|^2}{S_n(f)} df \right). \quad (4)$$

The tildes indicate that both d and m are expressed in the frequency domain. Since we will be considering the *ratio* of likelihoods, we do not need to take into account the normalisation factor, and it is sufficient to compute the proportionality in Eq. 4.

Assuming that the noise of different interferometers is independent, the expression of the PDF given data from a coherent network of N interferometers generalises to:

$$p(\vec{\lambda}|d) \propto p(\vec{\lambda}) \prod_{i=1}^N L_i(d|\vec{\lambda}). \quad (5)$$

2.4. Prior distribution

We use a prior distribution that is uniform in $\log(d_L)$, $\cos(\theta_{\text{SL}})$, $\sin(\text{Dec})$, $\sin(\theta_{J_0})$, (the sine is used for parameters defined in the domain $[-\frac{\pi}{2}, \frac{\pi}{2}]$, the cosine for $\theta_{\text{SL}} \in [0, \pi]$) and in the original scales of the remaining parameters. The allowed ranges for these parameters are between 1 and $6M_\odot$ for M_c , between 0 and 0.25 for η , in the range $t_c \pm 50$ ms, below 100Mpc for d_L , between 0 and 1 for a_{spin} , between -1 and 1 for the angles of which we use the sine or cosine as MCMC parameter, and between 0 and 2π for all other angles. We use these broad, flat priors to keep our study general. However, when additional information becomes available (for instance the time and sky location of a gamma-ray burst), we can restrict our priors.

2.5. Proposals

The Markov chain is created as follows. If in the current iteration i , the chain has the location in parameter space (set of waveform parameters, or state) $\vec{\lambda}_i$, we propose a random jump $\Delta\vec{\lambda}_i$ to the new location $\vec{\lambda}_{i+1} = \vec{\lambda}_i + \Delta\vec{\lambda}_i$. Since the jump proposal is random, the next state of the chain should depend *only* on the current state, thus giving the chain its Markovian property.

We then compute the likelihood for the new state as given by Eq. 5 and determine whether to accept it by comparing the acceptance probability (the left-hand side in Eq. 6) to a random number r drawn from a uniform distribution between 0 and 1:

$$\frac{p(\vec{\lambda}_{i+1}) L(d|\vec{\lambda}_{i+1})}{p(\vec{\lambda}_i) L(d|\vec{\lambda}_i)} > r. \quad (6)$$

The jump to state $\vec{\lambda}_{i+1}$ is accepted if Eq. 6 is fulfilled. Otherwise the jump is rejected, the chain keeps the old parameter set $\vec{\lambda}_{i+1} = \vec{\lambda}_i$ and a new iteration is started by drawing a new random jump proposal $\Delta\vec{\lambda}_{i+1}$ to a different state $\vec{\lambda}_{i+2}$. Equation 6 shows that a new state is always accepted when it improves the product of the prior and the likelihood, and that a larger decrease in this product means a smaller probability of acceptance.

We use an *adaptive* scheme for the proposed jump size [11]. The size of the jump proposal for the parameter λ^j (the j -th element of the vector $\vec{\lambda}$) is drawn from a Gaussian distribution with width σ_{jump}^j . Thus, these widths form a vector $\vec{\sigma}_{\text{jump}}$ with the same number of elements as $\vec{\lambda}$. The adaptation of the jump size consists of increasing σ_{jump}^j when a jump proposal in the parameter λ^j is accepted and decreasing it when a proposal is rejected. In a typical run, the increase of σ_{jump}^j is a factor of ~ 8 and the decrease a factor of ~ 2 , which results in the target acceptance ratio of about 25%.

2.5.1. Uncorrelated proposals The default method for choosing a jump proposal is to draw the jump size independently in the different dimensions of the parameter space. This implies that adaptation is done per parameter as well. We make these updates in two categories. The first category contains per-parameter updates, where the likelihood is calculated after proposing a jump in one parameter only, thus deciding whether to accept the jump for each parameter separately. The second category involves proposing a jump in all parameters at once before calculating the likelihood only once. This is typically done in 10% of the uncorrelated update proposals. For both categories of uncorrelated update proposals the same vector $\vec{\sigma}_{\text{jump}}$ is used.

2.5.2. Correlated proposals There can exist strong correlations between parameters, in which case uncorrelated updates can be very inefficient. We implemented a method to calculate the correlations between the parameters of a block of n_{corr} iterations (typically $n_{\text{corr}} \approx 10^4$). We then draw the subsequent n_{corr} jump proposals from a multivariate normal distribution that is given by the Cholesky decomposition of this covariance matrix.

We recompute the covariance matrix and its Cholesky decomposition at the end of each block of n_{corr} iterations, and decide whether to use the new matrix or not by checking how the diagonal elements of the matrix have changed. We find that if we accept each proposed matrix update (for which the covariance matrix is positive definite), our proposed jump sizes may become very small. This problem does not arise if we accept the new matrix only when $\sim 50\%$ of the diagonal elements have *decreased* in value.

The correlated update proposals are always block updates of all twelve parameters at once, hence there is a separate $\sigma_{\text{jump, corr}}$ for these updates. In a typical MCMC run, 70–90% of the update proposals are done in a correlated way, and 10–30% in an uncorrelated way.

2.6. Parallel tempering

The problem that arises when using MCMC for parameter estimation, and especially to find the (unknown) modes of the PDFs, is that the chains should typically be broad enough to sample the whole allowed parameter ranges, while also being able to probe the region of maximum likelihood in a detailed way. These two demands are almost mutually exclusive, but the technique known as *parallel tempering* offers a solution.

Parallel tempering consists of several parallel chains that each have a different ‘temperature’. In addition to the default Markov chain with $T = 1$, parallel chains of higher temperature are computed. Hotter chains are more likely to accept a jump that decreases the likelihood, by adjusting Eq. 6 to accept a jump when

$$\frac{p(\vec{\lambda}_{i+1})}{p(\vec{\lambda}_i)} \left(\frac{L(d|\vec{\lambda}_{i+1})}{L(d|\vec{\lambda}_i)} \right)^{\frac{1}{T}} > r, \quad (7)$$

where $T \geq 1$ is the temperature of the chain. (Eq. 7 can be viewed as the definition of “temperature”.) The property of more frequently accepting jumps that lower the likelihood allows a hot chain to move around in parameter space more widely than a cooler chain, thus allowing it to discover different modes. Hence, a combination of hot and cool chains can probe both wide parameter ranges and the narrow region(s) of maximum likelihood. In order to do so, the chains must be able to exchange information. This is done by swapping the parameter sets between two parallel chains with $T_m < T_n$ whenever:

$$\left(\frac{L_n}{L_m} \right)^{\frac{1}{T_m} - \frac{1}{T_n}} > r. \quad (8)$$

Since the likelihood that is needed to determine whether to swap the parameter sets was already calculated, this decision comes almost for free, and we make it for every pair of chains at every iteration. Output of parallel chains with different temperatures for a sample run is shown in Fig. 2 in Sect. 3.3.

2.6.1. Setting up a temperature ladder The temperature ladder is determined by setting the lower temperature to $T = 1$. This is the only chain that is saved and used to create the PDFs. One also has to choose a maximum temperature T_{\max} , which is typically the lowest temperature that allows the chain to scatter over the whole allowed parameter ranges quickly. In our test runs, we find that we need to increase the value of T_{\max} when injecting a signal with a higher SNR. The last quantity to choose is the number of parallel chains N_{ch} in the temperature ladder. This will be a compromise between high computation speed (low N_{ch}) and high swap efficiency for the chains by having small differences between adjacent temperatures (high N_{ch}). The temperatures are then chosen equidistantly in $\log(T)$. Our typical setup is $N_{\text{ch}} \approx 7$ and $T_{\max} \approx 30 - 50$ for SNRs between 10 and 20.

2.6.2. Sinusoidal temperatures The obvious drawback of parallel tempering is that one has to calculate a handful of chains, instead of just one. In order to reduce the number of chains in the temperature ladder, we have tested our simulations with *sinusoidal* temperatures, for all chains with $T > 1$. In order to do this, we set up our temperature ladder as before, but now sinusoidally oscillate the temperature of each chain $m \neq 1$ with an amplitude ΔT_m . We find that the swapping is efficient when we choose $\Delta T_m = T_m - T_{m-1}$ for each chain $m > 1$, so that the minimum temperature of each chain is equal to the mean temperature of the next cooler chain. Furthermore, we make sure that adjacent chains are in antiphase, so that there is

an optimal overlap at the extrema. In this setup, we can use $N_{\text{ch}} \approx 4 - 5$ and $T_{\text{max}} \approx 15 - 30$ for SNRs between 10 and 20, thus reducing the computational cost of the MCMC runs with parallel tempering. We suggest that the period of the temperature variation should not be too close to n_{corr} (see Sect. 2.5.2) and that a too short period may endanger the Markovian nature of the chain. Hence we chose a period that is $\sim 5 \times n_{\text{corr}}$.

3. MCMC simulations

In order to accurately sample the parameter space in our MCMC runs, we typically need to accumulate a few million iterations. A run with two detectors takes about a week to compute 2×10^6 iterations on a single 2.8 GHz CPU. The run time scales linearly with the number of detectors.

In an MCMC run, we simulate a data set by injecting a signal into detector noise, as described in Sect. 2.1. This data can be either Gaussian, stationary noise that is simulated at the designed sensitivity of the detectors, or real LIGO or Virgo data. The test simulations described in this study were all done using synthetic noise.

For our simulations, we inject the signal with a given set of parameters into the noise, appropriately projecting it onto one or more of the following gravitational-wave detectors: the 4-km LIGO detectors at Hanford, WA (H1), Livingston, LA (L1) in the USA, or the 3-km Virgo detector near Pisa, Italy. This way, we can do coherent parameter estimation with such a network of detectors, as described in Sect. 2.3.

Test simulations that we carry out at the moment focus on a fiducial binary that consists of a $10 M_{\odot}$ spinning black hole and a $1.4 M_{\odot}$ non-spinning neutron star at a distance of typically 15-20 Mpc. The purpose of these test simulations is twofold. Firstly, we want to establish to which accuracy the source parameters can be measured. Secondly, our code must be able to find these source parameters when started from arbitrary values. These are two different goals that require slightly different simulations to test in an efficient way.

3.1. Accuracy of parameter estimation

For a fixed network SNR (see Eq. 2), the accuracy with which the parameters can be determined depends mainly on three factors: the number of detectors in the network, the sky position and orientation of the binary (these quantities determine how the network SNR is distributed over the different interferometers) and the magnitude and the direction of the spin of the black hole. We are therefore carrying out a systematic study in which we vary these parameters in order to map their influence on the accuracy of the parameter estimation. Because of the large number of parameters that is varied, and the timescale of 1-2 weeks that is needed for the chains to accumulate a sufficient number of iterations, this is a lengthy process. To speed up these simulations, we usually start the Markov chains from the true source parameter values which were used for the software injection. We must be careful, however: in addition to parameter-estimation uncertainties caused by noise, which are quickly measured by seeding the chains with the true parameters, there can also be uncertainties due to degeneracies or near-degeneracies in the parameter space, which may not become apparent in short runs starting from the true parameter values. We typically start five serial chains for the same analysis. Each of these Markov chains uses parallel tempering, as described in Sect. 2.6, and hence consists of several *parallel* chains each with a different temperature. Although these serial chains start from the same (true) values, they use a different seed for the random-number generator and therefore produce different and independent Markov chains.

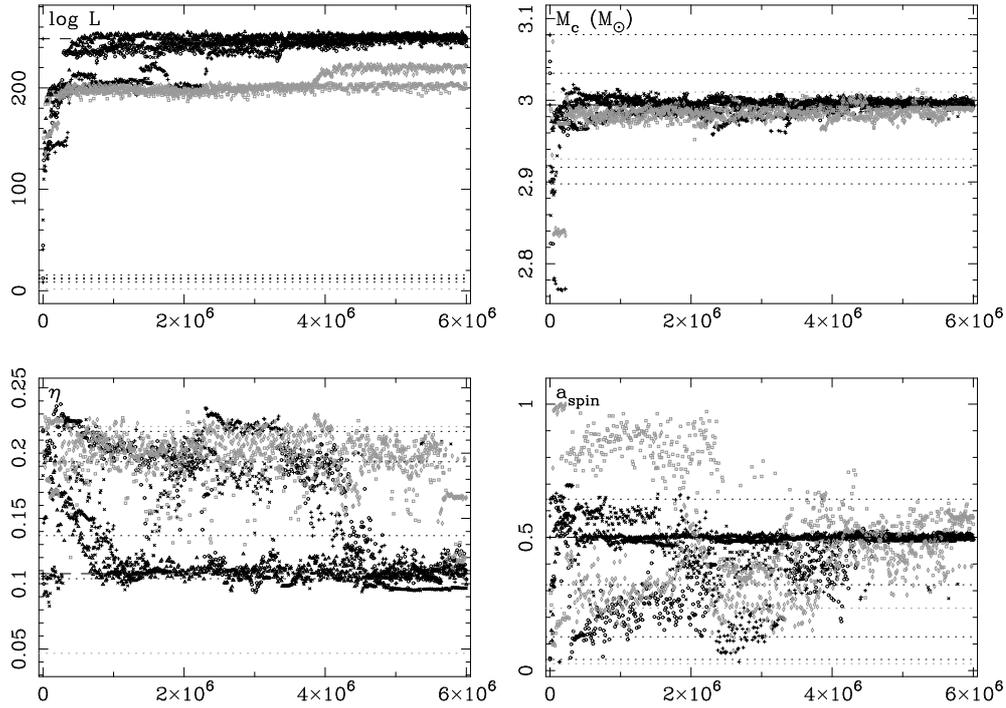


Figure 1. Likelihood and Markov chains for the parameter estimation on a simulated injection. The MCMC run consists of six serial chains plotted in two shades of grey. All of the four black chains find the parameters of the injected signal after $\sim 4.5 \times 10^6$ iterations, while the two grey chains have found other, local maxima. The figure shows the difference between the logarithm of the likelihood $\log L$ for a waveform with the current parameter values and $\log L$ for the null waveform (upper-left panel), and the chain projections for the chirp mass M_c (upper-right), symmetric mass ratio η (lower-left) and spin magnitude a_{spin} (lower-right). All horizontal axes show the iteration number in the chains. The six dotted lines in each panel indicate the starting value of each chain of the corresponding colour. The dashed black lines are the parameter values of the injected signal, and the corresponding value for the likelihood. One out of $\sim 10^4$ calculated iterations is plotted.

3.2. Finding the true parameters

The second purpose of the simulations with our MCMC code is to find the true source parameters in the case where these are unknown. In order to test the ability of our MCMC code to do so, we carry out a semi-blind analysis, in which the chains are started from offset (*i.e.*, non-true) parameter values. For the chirp mass and the time of coalescence, these starting values are drawn from a Gaussian distribution that is centred on the value of the injected parameter, with a standard deviation of about $0.1 M_\odot$ and 30 ms respectively. The other ten parameters are drawn randomly from the allowed ranges (see *e.g.* Fig. 1). By selecting the starting values for the chains this way, we model the information that will be available after a detection trigger is analysed by the LIGO-Virgo data-analysis pipeline.

When starting chains from offset values, we typically use up to ten serial chains. The reason for using a larger number of chains is that chains may get stuck at a local maximum in likelihood. In the case of our semi-blind analysis, it is easy to recognise such chains, since we can see whether the chains have found the likelihood of the injected signal. In the case of

a real analysis, we need to be sure that the chains have found the highest likelihood present. One way to do this is by starting multiple serial chains from different positions in parameter space and requiring that they find the same highest value for the likelihood. Figure 1 shows a run where six chains were started from offset values. After about 4.5×10^6 iterations, the four black chains have all found the same likelihood and parameter values, whereas the two grey chains are exploring different parts of parameter space. The latter two chains are clearly at a lower value of the likelihood and have therefore found *local* maxima in parameter space. If the run were continued, the last two chains should eventually find the parameter values of the injected signal.

3.3. Example of parallel tempering

Figure 2 shows how the parallel chains with different temperatures cooperate in the case of parallel tempering. The figure shows the likelihood and Markov chains for four different parallel chains. The chains with $T > 1$ have sinusoidal temperatures in the ranges 1.0–4.0, 2.5–10.1 and 6.3–25.5. The coolest chain (black plusses) has the highest likelihood and the smallest spread in the parameters, whereas the hottest chain (black squares) has the lowest likelihood and largest spread. The spread in likelihood in especially the second coolest chain (dark-grey circles) can be attributed in part to the sinusoidal variation in temperature; whenever the temperature drops, the chain may climb a nearby ‘hill’ in likelihood, find a high value and communicate the location of that hill to the cooler chain. The two ‘sudden’ jumps for the coolest chain, around iterations 5.2×10^5 and 1.52×10^6 , may be explained that way. In the first jump, the chain moves to the correct value for the chirp mass, in the second jump, the ‘true’ values for the mass ratio and spin magnitude are found. In most MCMC runs, we start 5–10 of these sets of parallel chains, but only save the output for the $T = 1$ chains.

4. Conclusions and future work

We have developed a Markov-chain Monte-Carlo (MCMC) algorithm that we use to estimate the twelve physical parameters of the gravitational waves emitted during a spinning compact-binary inspiral that can be detected with ground-based gravitational-wave observatories like LIGO and Virgo. In Section 2, we discuss many of the implemented features that are needed to run this code efficiently. In Section 3 we show examples of MCMC simulations carried out with our code.

We are constantly working on improving the efficiency with which the Markov chains explore the parameter space. In particular, a more efficient sampler speeds up the search for the true modes of the PDFs when the Markov chains are started from offset (*i.e.* non-true) initial parameter values, as they would be in the case of a real detection.

When the structure of the likelihood function in the parameter space is very rich, and the SNR is high, there are many sharply peaked local maxima of the likelihood. In this case, the MCMC algorithm is likely to get stuck on some of these local maxima, as random jumps are unlikely to lead from one local maximum to another. Coherent changes in parameters based on an improved analytical understanding of the waveform may allow us to efficiently traverse the peaks of this complicated parameter space. We are currently working on gaining a sufficient understanding of the harmonic structure of the waveform, which would allow us to implement such coherent jumps.

A part of our effort is directed at understanding degeneracies that exist between parameters, especially the sky position and orientation of the binary. For example, we find cases where the PDF for the sky position is more-or-less uniform over the whole sky, cases

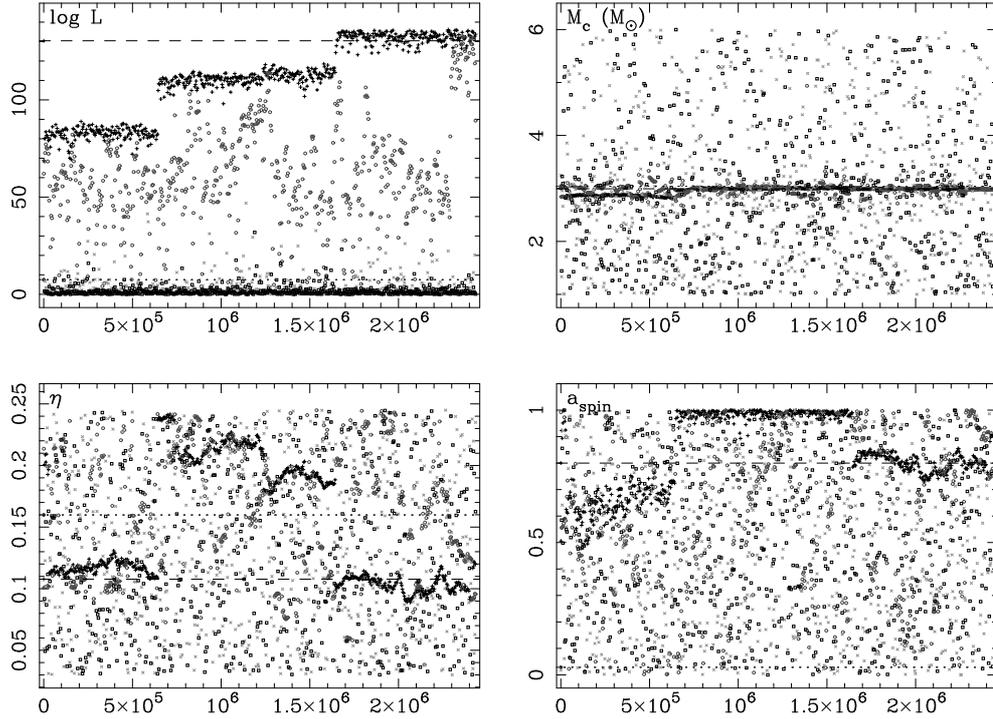


Figure 2. Likelihood and Markov chains for chains of different temperature in a sample run with parallel tempering. The panels contain the same information as those in Fig. 1, and the dashed and dotted lines have the same meaning as in that figure. The chains with $T > 1$ have sinusoidal temperatures and chains with increasing temperature are displayed by black pluses ($T = 1.0$), dark-grey circles ($T_{\text{mean}} \approx 2.5$), light-grey crosses ($T_{\text{mean}} \approx 6.3$) and black squares ($T_{\text{mean}} \approx 16.0$). One out of ~ 4000 calculated iterations is plotted.

where there are multiple regions in the sky where the binary could be, and cases where one unique sky position is resolved. We need to understand better how these degeneracies depend on the number of detectors in the network and on the exact configuration of that network with respect to the binary position and orientation, and on the spin of the black hole.

Another part of our work focuses on the implementation of a more realistic, higher-order post-Newtonian (PN) waveform, that includes the spin of both binary components [12]. The inclusion of the second spin will allow us to investigate inspirals where both spins are equally important, such as for double-neutron-star and double-black-hole binaries. The implementation of a higher-order PN waveform is expected to increase the accuracy of parameter estimation and to reduce the bias that inevitably arises when using approximate waveforms.

At the moment, we are testing our MCMC code regularly on software injections into real interferometer data (*e.g.* LIGO S5 playground data) and we plan to explore the possibility of doing follow-up on candidate events that come out of the LIGO detection pipeline [13; 14]. We have detailed plans to include this MCMC code as a final stage in the LIGO pipeline, in order to provide a post-processing tool that can be used after a detection.

Acknowledgements

This work is partially supported by a Packard Foundation Fellowship, a NASA BEFS grant (NNG06GH87G), and a NSF Gravitational Physics grant (PHY-0653321) to VK; NSF Gravitational Physics grant PHY-0553422 to NC; Royal Society of New Zealand Marsden Fund grant UOA-204 to RM and CR; UK Science and Technology Facilities Council grant to AV. Computations were performed on the Fugu computer cluster funded by NSF MRI grant PHY-0619274 to VK.

References

- [1] Abramovici A, Althouse W E and Drever R W P *et al* 1992 *Science* **256** 325–333
- [2] Waldman, S J (for the LIGO Science Collaboration) 2006 *Class. Quantum Grav.* **23** S653–S660
- [3] Acernese F *et al* 2007 *Class. Quant. Grav.* **24** S381–S388
- [4] Belczynski K, Taam R E, Rantsiou E and van der Sluys M 2008 *ApJ* **682** 474–486
- [5] Gilks W R, Richardson S and Spiegelhalter D J 1996 *Markov chain Monte Carlo in practice* (London/Boca Raton, Fl.: Chapman & Hall/CRC)
- [6] Röver C, Meyer R and Christensen N 2007 *Phys. Rev. D* **75** 062004
- [7] Röver C 2007 Ph.D. thesis The University of Auckland URL <http://hdl.handle.net/2292/2356>
- [8] Apostolatos T A, Cutler C, Sussman G J and Thorne K S 1994 *Phys. Rev. D* **49** 6274–6297
- [9] Sigg D 2004 *Class. Quantum Grav.* **21** S409–S415
- [10] Van der Sluys M V *et al* 2008 *ApJL*, submitted (*Preprint* <http://arxiv.org/abs/0710.1897>)
- [11] Atchadé Y F and Rosenthal J S 2005 *Bernoulli* **11** 815–828
- [12] Buonanno A, Chen Y and Vallisneri M 2003 *Phys. Rev. D* **67** 104025
- [13] The LIGO Scientific Collaboration: Abbott B 2008 *Phys. Rev. D* **77** 062002
- [14] The LIGO Scientific Collaboration: Abbott B 2008 *Phys. Rev. D* **78** 042002