

Follow-up MCMC manual

Christian Röver

May 26, 2009

Contents

1	Introduction	2
2	Usage	2
2.1	What the code does	2
2.2	Command line options: general	3
2.3	Command line options: details	3
2.4	Examples	6
3	C code internals	7
3.1	Initialisation	7
3.1.1	The <code>DataFramework</code> structure	7
3.1.2	The <code>McmcFramework</code> structure	7
3.1.3	The <code>interferometer</code> structure	7
3.1.4	The <code>init()</code> function	7
3.2	MCMC	7
3.2.1	The <code>loglikelihood()</code> function	7
3.2.2	The <code>logprior()</code> function	8
3.2.3	The <code>priordraw()</code> function	8
3.2.4	The <code>importancesample()</code> function	8
3.2.5	The <code>metropolishastings()</code> function	8
3.3	Currently provided templates etc.	8
3.3.1	“ <code>InspiralNoSpin</code> ” templates	8
3.3.2	“ <code>BurstSineGaussian</code> ” templates	10
3.4	How to add templates etc.	10

1 Introduction

The follow-up MCMC code is mainly intended for parameter estimation at the end of a detection pipeline, but it is also able to simulate data (generate noise and inject signals), e.g. for parameter estimation studies. It is supposed to be easily usable via its command-line options, and also easily extensible by its modular code structure. By now there are several waveform templates available, non-spinning inspiral waveforms as well as sine-Gaussian burst waveforms. The code should then be extensible to other waveform families with identical or different parametrisations. The code can then run MCMCs for parameter estimation in a Bayesian framework, using data read from files, or using simulated noise and injected signals.

2 Usage

2.1 What the code does

Data is either read from files, or noise may be simulated based on specific noise curve settings and a signal may be injected. The noise's power spectral density is estimated from additional data that does not overlap with the considered data set, and which is supposed to consist of noise only. Data is processed completely in the Fourier-domain, i.e., the analysis is based on Fourier-transformed data, the (estimated) noise power spectrum, and Fourier-domain signal templates. In order to reduce leakage effects, the data, any time-domain templates, and the data used for spectrum estimation are windowed. In order to ensure that leakage affects the spectrum estimate and template in the same way as the actual data, the same sample size and window function is used for data, spectrum estimation and FFTing of time-domain signal templates. The power spectral density is estimated basically using Welch's method (Welch, 1967), but without overlap of segments. Data (usually sampled at 16 384 or 20 000 Hz) is by default low-pass filtered and downsampled by a factor of 4 (Crochiere, 1979).

The data is eventually analysed assuming the noise to be Gaussian with a known spectral density; the resulting likelihood function is given in Finn (1992). The a priori information needs to be specified in terms of the joint prior distribution for the parameters of the signal in question. The parameters in general include the location/orientation parameters (declination, right ascension and polarisation, (Röver, 2007, ch. 4)). Other signal-specific parameters might be for example chirp mass, mass ratio, phase, coalescence time, inclination angle, etc. in case of binary inspiral signals.

The aim is to compute integrals (moments, quantiles, marginal densities,...) of the joint posterior probability distribution of the signal parameters, i.e. to extract information about parameters conditional on the data at hand. The posterior distribution is proportional to the product of prior density and likelihood function. Integrals are computed using Monte Carlo integration. The code implements a Markov chain Monte Carlo (MCMC) algorithm, the Metropolis-Hastings sampler, and it returns a text file of simulated draws from the posterior distribution in order to be used for Monte Carlo integration (Röver, 2007, ch. 3).

2.2 Command line options: general

Most command-line options are set by passing a command of the form “`--option value`” to the program, where “`--option`” is the name of the command line argument, and “`value`” is its value, which may be a number or a character string. Some options require vectors as arguments, which are set by a call like “`--option [name1=value1,name2=value2,name3=value3]`”. Note that the option parsing function cannot handle extra white space within the square brackets. In some cases, the names of the vector elements do not need to be given; one example is the `--randomseed` option, where a specification of (e.g.) `--randomseed [12345,67890]` (without any name specification) is sufficient.

In the following, the different command line options are explained in more detail, a summarizing table and subsequent examples hopefully help clarifying more. Running the followup code without any options (or the `--help` option) specified will also display a brief summary of available options.

2.3 Command line options: details

The `--template` option allows to specify the signal template to be used for likelihood computations within the MCMC. Currently the following options are available: “20SP” (2.0 PN stationary-phase inspiral template), “25SP” (2.5 PN stationary-phase inspiral template), “2025” (2.0 PN amplitude / 2.5 PN phase inspiral template), “2535” (2.5 PN amplitude / 3.5 PN phase inspiral template), “LALTaylorT2PN00”, “LALTaylorT2PN10”, “LALTaylorT2PN15”, “LALTaylorT2PN20”, “LALTaylorT3PN00”, “LALTaylorT3PN10”, “LALTaylorT3PN15”, “LALTaylorT3PN20” (LAL ‘TaylorT2’ and ‘TaylorT3’ templates from Newtonian to 2.0 PN order), and “SineGaussian” (Sine-Gaussian burst template). More details about the different templates are given in section 3.3 below.

The `--outfilename` option is used for specifying the file paths and names to be used for storing MCMC output and other information. The argument is of the format “`/home/user/example`” (without a file name extension), and then two files, “`.../example.log`” and “`.../example.csv`”, are going to be generated. The former (`.log`) file is used for logging general information about the run, like command line arguments, date and time, etc. The latter (`.csv`) file then contains the actual MCMC output. Both files are in comma-separated-values (CSV) format, the very first line being a header line (Shafranovich, 2005). Currently every 100th MCMC iteration is logged. The `--iterations` option is used to specify the number of MCMC iterations to be simulated, and the `--randomseed` option needs to be specified in order to initialise the random number generator. One or two arguments (integers, passed in an un-named vector) may be specified for the `--randomseed` option. If one is specified, a repeated run with the same arguments will yield exactly the same output. Specifying two arguments only makes sense when simulating data (instead of reading from files). If two are specified, then the first one will be used for simulation of data and PSD estimation, while the second one will be used for initialisation of the importance sampling and MCMC stage. This way one can do several runs on identical simulated data (and spectrum estimate), but with independent MCMCs. In order to achieve that, the first one is specified the same, while second one is specified differently for each individual run.

The time frame of the data to be analysed is specified through the `--tcenter`,

`--tbefore` and `--tafter` options. `--tcenter` provides the pivotal point of the resulting time window; in the case of an inspiral signal, this is usually a prior guess of the coalescence time (in GPS seconds). `--tbefore` and `--tafter` denote how many seconds of data are to be read before and after that point in time. The data is Tukey-windowed (Röver, 2007) before Fourier-transformation, and it is ensured that the time span `[tcenter - tbefore, tcenter + tafter]` falls within the *flat* region of the Tukey-window, and so the resulting amount of data actually being read is greater than just `(tbefore+tafter)` seconds. The Tukey-window’s parameter (denoting the fraction in which it is flat; $0 < \alpha < 1$) is set via the `--tukey` option; the default value is $\alpha = 0.1$. For little/short data (e.g. for analysing burst signals), larger values for α *might* be appropriate. Due to the way the data range eventually processed is selected (so that it falls within the Tukey window’s flat region), it should be $\alpha \ll 1$, so that there actually *is* a flat region.

Data may be read from *Frame format*, which is handled internally via the *Frame library*. The file names for data and noise (PSD estimation) are supplied via a *cache file*. A cache file here looks like:

```
...
L L1_RDS_C03_L2 847546596 128
file://localhost/data/L-L1_RDS_C03_L2-847546596-128.gwf
L L1_RDS_C03_L2 847546724 128
file://localhost/data/L-L1_RDS_C03_L2-847546724-128.gwf
L L1_RDS_C03_L2 847546852 128
file://localhost/data/L-L1_RDS_C03_L2-847546852-128.gwf
...
```

i.e., it contains in particular GPS start time and duration, and the path and file name for each GWF file. One can specify a single cache file (e.g. `--cachefile /home/data/H1.cache`) or several cache files (`--cachefile [/home/data/H1.cache,/home/data/L1.cache,/home/data/V1.cache]`). Interferometer locations, file sizes etc. will be gathered from the cache files. File channels need to be specified correspondingly for each data set, e.g. `--filechannel [H1:LSC-STRAIN,H2:LSC-STRAIN,L1:LSC-STRAIN]`. Noise spectrum estimation is also done based on frame file listed in the cache file(s).

If one wants to have data simulated instead of read from files, one needs to specify the interferometer location(s) via the `--network` argument, for example `--network H` or `--network [H,H,L,V]`. Possible interferometer sites are (by now) Hanford, Livingston, Pisa and Hannover (H, L, V and G, respectively).

For both fake data generation as well as the actual MCMC, the noise’s power spectral density (PSD) needs to be known. For simulated noise, the PSD is specified in terms of a character string, the options currently available are `“initialLigo”`, `“advancedLigo”`, `“Virgo”` and `“Geo”`. These functions are basically the same as those provided in LAL (`LALLIGOIPsd()`, `LALAdvLIGOPsd()` and `LALVIRGOPsd()`) (Creighton et al., 2007; Damour et al., 2001), with the difference that here these are bounded above at 2×10^{20} the value at their respective ‘sweet spots’ (so they are defined at low frequencies as well). This only makes a difference at rather low frequencies, kicking in at 21.9 Hz for initial LIGO, 0.300 Hz for advanced LIGO, and 1.42 Hz for Virgo.

If data is read from files, the noise spectrum also needs to be estimated from data, which is preferably close in time and does not overlap with the data actually analysed. For this purpose, a number of data stretches of the same

size as the actual data are read, windowed, their spectra are computed and eventually averaged. The data range to be used for this purpose is defined by `--psdestimatestart` and `--psdestimateend`, and `--psdestimaten` gives the (maximum) number of segments to be used. Data are read and spectra averaged until either the end of the range or the maximum number are reached. For simulated data, one can also plug in the “known” power spectrum for use with the following MCMC by setting the `--fixspecdens` flag, but the default (and highly recommended) option is to also estimate the spectrum from a number of random noise samples. The only figure to specify in this case is the number of noise samples to be averaged over, `--psdestimaten`.

The `--freqlower` and `--frequpper` options are used to specify the frequency range to consider for likelihood computations. By default, this is set to the interval from 40 Hz to $\frac{7}{8}$ of the Nyquist frequency.

The `--fixed` option is used to specify the parameters that are supposed to be fixed at certain values. For example, one may wish to fix the sky location parameters to certain values, which would require a specification like `--fixed [declination=0.51,rightascension=0.73]`. The elements of the passed vector need to be named correctly.

In order to “inject” a signal into the (real or simulated) data, you need to specify the `--inject` option. It needs to be followed by the complete (!) set of parameters for the signal template in question. The elements of the passed vector need to be named properly (see e.g. examples below, or the template details in section 3.3). The signal template is by default taken to be the same as the one used for parameter estimation (which was specified through the `--template` option), but a different one may be specified using the `--injecttemplate` option.

The starting parameter values of the MCMC may be specified through the `--start` option. If not (or only partly) specified, a random draw from the prior distribution is used as a starting value(s) for unspecified parameters. Alternatively, one can use importance resampling for generating a set of starting values (see e.g. Röver (2007)). The number of initial samples needs to be specified via the `--importanceresample` option. With no other options given, importance resampling is based on drawing a (large) sample of parameter values from the prior distribution, out of which the starting value is resampled. The `--guess` option may be used to provide rough estimates of some parameters, e.g. trigger values from a detection pipeline. This option by now is only implemented for “*InspiralNoSpin*” templates. For more details see section 3.3.1 below.

The tunable parameters of the prior distribution may be set using the `--priorparameters` argument, an un-named vector. For more details about parameters of different priors see section 3.3 below.

The command line options `--tbefore`, `--tafter`, `--filechannel`, `--specdens`, `--psdestimatestart`, `--psdestimateend`, `--psdestimaten`, `--freqlower` and `--frequpper` may be specified either with a single argument, implying that the same setting applies for the whole network, or with a number of arguments matching the number of data sets used. One might for example wish to use settings like `--freqlower [40,40,30] --frequpper 1500`.

The `--fixed`, `--start` and `--inject` parameter vectors may be provided in terms of geographical (`[... ,latitude=1.23,longitude=2.34,...]`) or celestial coordinates (`[... ,declination=1.23,rightascension=2.34,...]`). The MCMC output will always be logged in terms of celestial coordinates, though.

Table 1: Brief description of command-line arguments and their effects. For more details see the main text.

command	default	options/effect/purpose
<code>--help</code>		display help message
<code>--template</code>		specify signal template
<code>--outfilename</code>		output (text) file name
<code>--iterations</code>	1 000 000	number of MCMC iterations
<code>--randomseed</code>		2 or 4 integers for initialisation
<code>--tcenter</code>		‘center’ of data to be analysed
<code>--tbefore</code>	30	margin before...
<code>--tafter</code>	1	...and after above ‘center’ (s)
<code>--tukey</code>	0.1	parameter for Tukey window
<code>--cachefile</code>		cache file(s) for data to be analysed
<code>--network</code>		interferometer sites
<code>--filechannel</code>		channel(s) for gwf files
<code>--specdens</code>	initialLigo	spectral density(-ies) to be used
<code>--fixspecdens</code>		fix PSD (instead of estimating)
<code>--psdestimatestart</code>		starting time(s) for PSD estimation
<code>--psdestimateend</code>		end time(s) for PSD estimation
<code>--psdestimaten</code>	100	number(s) of segments to average over
<code>--freqlower</code>	40	lowest frequency considered (Hz)
<code>--frequpper</code>	$\frac{7}{8} \times f_{\text{Nyquist}}$	greatest frequency considered (Hz)
<code>--fixed</code>		vector of fixed parameters
<code>--start</code>		vector of starting values
<code>--guess</code>		vector of parameter guesses
<code>--inject</code>		vector of injection parameters
<code>--injecttemplate</code>	<code>--template</code>	template to use for injection
<code>--importanceresample</code>		number of starting value draws
<code>--priorparameters</code>		vector of prior parameters

2.4 Examples

Simulate data, inject a signal and recover it:

```
./followupMcmc --network [H,L,V] --randomseed [123,456] --tcenter 100 --outfilename /home/user/data/example01 --template 25SP --specdens [initialLigo,initialLigo,Virgo] --freqlower [40,40,30] --tbefore [20.0,20.0,30.0] --inject [chirpmass=2.0,massratio=0.24,inclination=1.0,time=100.0,logdistance=3.2,latitude=0.5,longitude=-1.0,polarisation=1.0,phase=3.0] --guess [chirpmass=2.0,massratio=0.24,time=100,distance=25.0] --importanceresample 10000
```

Run MCMC on data read from files:

```
./followupMcmc --cachefile [/home/user/data/H1.cache,/home/user/data/H2.cache,/home/user/data/L1.cache] --template 25SP --tcenter 873739911.131 --tbefore 20.0 --filechannel [H1:LSC-STRAIN,H2:LSC-STRAIN,L1:LSC-STRAIN] --psdestimatestart 873737200 --psdestimateend 873739500 --importanceresample 100000 --randomseed 123 --outfilename /home/user/data/example02 --iterations 10000000 --priorparameters [0.75,7,873739911.081,873739911.181,40,80] --guess [2.0,0.13,873739911.131,40]
```

3 C code internals

3.1 Initialisation

3.1.1 The DataFramework structure

This structure contains all the information related to the data, like paths, file-names, sampling rate, power spectral density, the data itself in time- and fourier-domain, etc.; also includes the “`fftw_plan`” structure for transforming data or corresponding templates.

3.1.2 The McmcFramework structure

This structure contains all information relevant for the MCMC algorithm to be run; in particular: parametrisation / templates to be used, names of parameters, values of possibly fixed parameters, starting values, details of proposal distributions etc. etc.

The `‘.template’` contains the exact template to be used within the MCMC, while the `‘.parameterset’` slot indicates the ‘family’ the template belongs to; possible values are for example ‘20SP’ (2.0PN stationary phase), or ‘2535’ (2.5PN amplitude, 3.5PN phase) for the template, both of which belong to the ‘family’ of templates with ‘`InspiralNoSpin`’ parameters.

The `‘.startvalue’` slot in particular contains the starting parameter vector for the MCMC algorithm. Length and names of this vector are frequently checked, and so it is regarded as a “reference”, also e.g. for matching proposal parameters with proposal covariance matrix entries.

3.1.3 The interferometer structure

Contains the relevant information about interferometers (location, orientation, name). A list of all available interferometers is initialised (by the `ifoInit()` function) and kept, and each `DataFramework` structure contains a pointer to its corresponding interferometer in order to compute local parameters (altitude, azimuth,...) from global parameters (latitude, longitude,...). Interferometers’ parameters are hard-coded in the `ifoInit()` function; this is also where additional interferometers would need to be added.

3.1.4 The init() function

This function does the initialisation of `DataFramework` and `McmcFramework` structures, some of which is controlled via the provided command line options. Includes command-line option parsing, default settings, settings of noise power spectrum, `readData()`, filtering, downsampling, `simulateData()`, FT/windowing setup, prior setup, proposal covariance initialisation, signal injection, etc.

3.2 MCMC

3.2.1 The loglikelihood() function

Calls the `signaltemplate()` “wrapper” function, which in turn calls the appropriate (frequency-domain) signal template function based on the setting given in the `‘.template’` slot of the `McmcFramework` structure. Takes (noise) power

spectral density from the `DataFramework`’s `.powspec` slot. Likelihood is computed as in (Finn, 1992). Note also the closely related `signaltonoiseratio()` function.

3.2.2 The `logprior()` function

(Non-normalised) log prior density function. Wrapper function depending on `McmcFramework.template` slot.

3.2.3 The `priordraw()` function

Generates samples from the prior distribution. See also `init()` and `importanceresample()`.

3.2.4 The `importanceresample()` function

Generates starting values for the MCMC using *importance resampling*, and is called within the `init()` function. Importance resampling is supposed to generate a draw that approximately follows a given distribution, which in this case is the posterior distribution. It proceeds by generating a (large) sample from a distribution similar to the desired one, and out of that draws (*resamples*) another, smaller, sample with probabilities that are based on *importance ratios*. For efficiency, implementation is done as outlined in (Röver, 2007).

By default the ‘approximative’ distribution is taken to be the prior distribution. If some rough estimates of some parameters are provided through the ‘`--guess`’ command line option (e.g. from trigger values), then the approximative distribution is narrowed down to a neighbourhood of these provided parameter guesses.

3.2.5 The `metropolishastings()` function

By now actually implements a Metropolis sampler (not a Metropolis-*Hastings* sampler) although some of the infrastructure is already in place. Proposal distributions by now are hard-coded, depending on the `McmcFramework.template` slot. Uses the `logprior()`, `loglikelihood()` and `propose()` wrapper functions, and logs resulting MCMC samples to a text file using the `logtofile()` function.

3.3 Currently provided templates etc.

3.3.1 “InspirationalNoSpin” templates

General: 9 parameters: azimuth $\in [0, 2\pi]$, altitude $\in [0, \pi]$, polarisation $\in [0, \pi]$, time $\in [?, ?]$, phase $\in [0, 2\pi]$, logdistance $\in [-\infty, \infty]$, chirp mass $\in [?, ?]$, mass ratio $\in [0, 0.25]$, inclination $\in [0, \pi]$.

Note that hardware injections (by now) usually are done assuming *overhead sky location* and *optimal orientation*. This means that altitude=0, azimuth=0, polarisation=0 and inclination=0. You can use the “`--fixed`” option to run the MCMC on the 5 remaining parameters only (by specifying `--fixed [altitude=0,azimuth=0,polarisation=0,inclination=0]`).

Prior: as in Röver (2007, page 78 sqq.). Detection probability generally seems to decrease from ‘certain’ to ‘impossible’ for SNRs roughly between 9.0–9.5 and 5.0–8.0, see Umstätter and Tinto (2008). Conservative choice: set $\text{SNR}_{90\%} := 8.0$ and $\text{SNR}_{10\%} := 4.0$. For initial LIGO noise, a 2+2Ms inspiral yields SNRs 8.0 and 4.0 at 40Mpc and 80Mpc respectively. Note also the related *inspiral range* discussion in Finn and Chernoff (1993, Sec. V.B).

The six parameters that may be set via the `--priorparameters` command-line argument are (in this order!):

1. lower mass bound (default: $1.0 M_{\odot}$)
2. upper mass bound (default: $15.0 M_{\odot}$)
3. lower coalescence time bound (default: `tcenter`−0.050 s)
4. upper coalescence time bound (default: `tcenter`+0.050 s)
5. distance at which an *optimally oriented* 2+2 M_{\odot} inspiral has 90% detection probability (default: 40.0 Mpc)
6. distance at which an *optimally oriented* 2+2 M_{\odot} inspiral has 10% detection probability (default: 80.0 Mpc)

Using the “--guess” option: The parameters that may be supplied are (in this order):

1. chirp mass,
2. mass ratio,
3. coalescence time, and
4. luminosity distance.

Example: `--guess [2.003,0.247,100.001,15]`. With this option specified, the initial parameter draws in the importance resampling stage are drawn from near the provided values. “*Near provided values*” here means:

- log-Normal centered at chirp mass value, standard deviation 0.01 ($\approx 1\%$)
- Normal centered at mass ratio value, standard deviation 0.05
- Normal centered at trigger time, standard deviation 0.005s (5ms)
- log-Normal centered at luminosity distance, standard deviation 0.333 ($\approx 33\%$)

2.0PN & 2.5PN stationary phase: (Tanaka and Tagoshi, 2000).

2.0PN amplitude / 2.5PN phase: (Blanchet, 2001). Termination of waveform as soon as frequency starts decreasing.

2.5PN amplitude / 3.5PN phase: (Blanchet et al., 2002, 2004; Arun et al., 2004). Phase evolution termination as above.

Restricted 2.0 PN: (Arnaud et al., 2007, Sec. 4.4). Without tapering (yet?).

LAL templates: By now the only properly working templates are the `TaylorT2` and `TaylorT3` templates up to 2PN order (Creighton et al., 2007).

3.3.2 “BurstSineGaussian” templates

General: 8 parameters: azimuth $\in [0, 2\pi]$, altitude $\in [0, \pi]$, polarisation $\in [0, \pi]$, time $\in [?, ?]$, phase $\in [0, 2\pi]$, logamplitude $\in [-\infty, \infty]$, logsigma $\in [-\infty, \infty]$, frequency $\in [0, \infty]$

Prior: Uniform for azimuth, $\cos(\text{altitude})$, polarisation, time, phase and frequency. (For now) Exponential distribution for amplitude and frequency, which is the *maximum entropy* distribution for a given prior expectation value.

The six parameters that may be set via the `--priorparameters` command-line argument are:

1. lower frequency (f) bound (default: 1 Hz)
2. upper frequency (f) bound (default: 1500 Hz)
3. lower time (μ) bound (default: `tcenter`−0.050 s)
4. upper time (μ) bound (default: `tcenter`+0.050 s)
5. expected amplitude (default: 1.0×10^{-20} Hz)
6. expected sigma (default: 0.050 s)

Sine-Gaussian burst: (time-domain) signal waveform:

$$s(t) = a \exp\left(\frac{(t-\mu)^2}{2\sigma^2}\right) \sin(2\pi f(t - \mu) + \phi)$$

where a is the amplitude, μ is the time parameter, σ is the width parameter, f is the frequency and ϕ is the phase. Note also the key figure $Q := 2\pi f\sigma$ relating peak width σ to frequency f (Weinstein, 2003).

3.4 How to add templates etc.

Specify which “family” (`enum signal`) the new template belongs to; add if necessary. Specify new name (`enum template`) of new template. If family is new, change the `init()` and `vectorSetup()` functions to do initialisation for the new family... number of parameters, parameter names, default settings etc. Also change the “wrapper” functions `logprior()`, `priordraw()` and `propose()`. If template family is already present, you only need to change the `signaltemplate()` function and provide a corresponding new alternative (frequency-domain) template generating function.

References

Arnaud, K. A. et al. (2007, October). An overview of the second round of the Mock LISA Data Challenges. *Classical and Quantum Gravity* 24(19), S551–S564.

- Arun, K. G., L. Blanchet, B. R. Iyer, and M. S. S. Qusailah (2004, August). The 2.5 PN gravitational wave polarizations from inspiralling compact binaries in circular orbits. *Classical and Quantum Gravity* 21(15), 3771–3801. Note the erratum Arun et al. (2005).
- Arun, K. G., L. Blanchet, B. R. Iyer, and M. S. S. Qusailah (2005, July). Corrigendum: The 2.5PN gravitational wave polarizations from inspiralling compact binaries in circular orbits. *Classical and Quantum Gravity* 22(14), 3115–3117. (See also Arun et al. (2004)).
- Blanchet, L. (2001). Post-Newtonian computation of binary inspiral waveforms. In I. Ciufolini, V. Gorini, U. Moschella, and P. Fré (Eds.), *Gravitational waves: Proceedings of the Como school on gravitational waves in astrophysics*. Bristol: Institute of Physics Publishing. See also Arxiv preprint gr-qc/0104084.
- Blanchet, L., T. Damour, G. Esposito-Farèse, and B. R. Iyer (2004, August). Gravitational radiation from inspiralling compact binaries completed at the third post-Newtonian order. *Physical Review Letters* 93(9), 091101.
- Blanchet, L., G. Faye, B. R. Iyer, and B. Joguet (2002, March). Gravitational-wave inspiral of compact binary systems to 7/2 post-Newtonian order. *Physical Review D* 65(6), 061501. Note the erratum Blanchet et al. (2005).
- Blanchet, L., G. Faye, B. R. Iyer, and B. Joguet (2005, June). Erratum: Gravitational-wave inspiral of compact binary systems to 7/2 post-Newtonian order. *Physical Review D* 71(12), 129902. (See also Blanchet et al. (2002)).
- Creighton, J. et al. (2007, February). *LAL software documentation*. URL <http://www.lsc-group.phys.uwm.edu/daswg/projects/lal.html>.
- Crochiere, R. E. (1979). A general program to perform sampling rate conversion of data by rational ratios. In A. C. Schell et al. (Eds.), *Programs for digital signal processing*, Chapter 8.2. New York: IEEE Press.
- Damour, T., B. R. Iyer, and B. S. Sathyaprakash (2001, January). Comparison of search templates for gravitational waves from binary inspiral. *Physical Review D* 63(4), 044023.
- Finn, L. S. (1992, December). Detection, measurement, and gravitational radiation. *Physical Review D* 46(12), 5236–5249.
- Finn, L. S. and D. F. Chernoff (1993, March). Observing binary inspiral in gravitational radiation: One interferometer. *Physical Review D* 47(6), 2198–2219.
- Röver, C. (2007). *Bayesian inference on astrophysical binary inspirals based on gravitational-wave measurements*. Ph. D. thesis, The University of Auckland. URL <http://hdl.handle.net/2292/2356>.
- Shafranovich, Y. (2005, October). Common format and MIME type for comma-separated values (CSV) files. URL <http://tools.ietf.org/html/rfc4180>.
- Tanaka, T. and H. Tagoshi (2000, October). Use of new coordinates for the template space in a hierarchical search for gravitational waves from inspiraling binaries. *Physical Review D* 62(8), 082001.

- Umstätter, R. and M. Tinto (2008, April). Bayesian comparison of post-Newtonian approximations of gravitational wave chirp signals. *Physical Review D* 77(8), 082002.
- Weinstein, A. J. (2003). LIGO burst simulations. URL <http://www.ligo.caltech.edu/~ajw/bursts/burstsim.html>.
- Welch, P. D. (1967, June). The use of Fast Fourier Transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics* AU-15(2), 70–73.