Documentation SPINspiral: Parameter estimation on gravitational-wave signals from spinning binary inspirals

Marc van der Sluys[1], Vivien Raymond[2], Ilya Mandel[3]
[1]Radboud University Nijmegen, [2]Northwestern University, [3]MIT
http://www.astro.ru.nl/∼sluys/index.php?title=SPINspiral

October 25, 2011

Abstract

SPINspiral was developed at Northwestern University to analyse gravitational-wave signals from stellar-mass binary inspirals, as can be detected by ground-based interferometers like LIGO and Virgo. The code performs parameter estimation on such signals, using Markov-chain Monte-Carlo (MCMC) techniques. This analysis includes the spins of the binary components. SPINspiral was adapted from an MCMC code for non-spinning binary inspirals written by Christian Röver [?].

Contents

1 Getting started

1.1 Quick start

1. You will need to install the LIGO Analysis Library (LAL) [?] if you want to use the LAL waveforms.
2. Check out a working copy from the `SVN` and cd to the directory `trunk`, or go to the directory `trunk` and update your working copy (`svn up`).
3. Check whether a proper `Makefile` is in your trunk. There's an example file in `doc/Makefiles/`, make sure it is configured for your compiler and type `make clean SPINspiral` for the first compilation and `make SPINspiral` for subsequent ones. On Fugu, use `make clean && condor_compile make -j6 SPINspiral`. The executable file `SPINspiral` should be produced.
4. Make sure you have the six configuration files `SPINspiral.input*` in your trunk. There are example files in `doc/input_*/`; you'll need four files from `doc/input_all/` and two from one or two of the other directories, depending on which waveforms to use for injection and parameter estimation. If unsure, use the two from `doc/input_apostolatos/`. Set `nIter` in `SPINspiral.input.mcmc` to 10 or 100 for a first test run.
5. Type `./SPINspiral` and SPINspiral should run with the parameters specified in the input files.

1.2 SVN

SPINspiral is currently maintained at Northwestern in a version-control system called `Subversion` or `SVN`. The repository can be found at `https://ciera.northwestern.edu/svn/sluys/mcmc_code/`. A concise `SVN` HOWTO with links for more info can be found at `http://www.astro.ru.nl/~sluys/index.php?page=Public/svn-howto`. The first time you want to retrieve the source code from the server, you need to check it out with: `svn checkout --username <user> https://ciera.northwestern.edu/svn/sluys/mcmc_code/`, which will prompt for the password that comes with your user name `<user>`. It will create a directory `mcmc_code/` in your current directory, and the default working directory is `mcmc_code/trunk/`.

2 Source code

2.1 Source files

SPINspiral currently has ten source files (`mcmc_*.c`) and is written in plain C. An example Makefile that should compile your code is included in the `SVN`, in the directory `doc/`.

SPINspiral is split into several routines and functions that give it a modular character. The routines are grouped into six source files:

SPINspiral_main.c Main routine. The idea is to keep this routine as small as possible and use it to call other routines.

SPINspiral_parameters.c Contains routines that deal with reading input files and choosing and setting (starting or injection) parameters.

SPINspiral_data.c Contains routines for data and noise reading and handling.

SPINspiral_templates.c Contains routines to generate templates or obtain them from LAL.

SPINspiral_signal.c Contains routines to compute things like overlaps and likelihoods.

SPINspiral_mcmc.c Contains the MCMC core.

SPINspiral_lal.c Contains interfaces to LAL routines.

SPINspiral_nolal.c Contains dummy interfaces to LAL routines, used when not linking against LAL.

SPINspiral_routines.c Contains more general supporting functions.

SPINspiral_3rdparty.c Contains third-party routines.

3 Command-line options

Most of SPINSPIRAL's options ($\sim 250 - 300$) are specified in input files (see Section 4). However, a small — but possibly growing — number of options can be passed to the program using command-line options:

`-i <main input file name>` override the name of the main input file (Section 4.1)

`--injXMLfilename <file name>` specify an XML file to read injection data from

`--injXMLnr <0-...>` specify the injection number in the injection XML file

`--mChirp` specify a trigger value for the chirp mass ($\mathcal{M}$) in $M_\odot$

`--eta` specify a trigger value for the mass ratio (η)

`--tc` specify a trigger value for the GPS time of coalescence (s)

`--dist` specify a trigger value for the distance (Mpc)

`--nIter` specify the desired number of MCMC iterations

`--nSkip` specify the number of step of which the output should be skipped between two saved iterations

`--network` specify the network configuration, *e.g.* H1=[1], H1L1=[1,2], H1L1V1=[1,2,3], V1H1=[3,1]

`--downsample` specify the downsample factor for the data

`--beforetc` specify the number of seconds of data before t_c that should be analysed

`--aftertc` specify the number of seconds of data after t_c that should be analysed

`--Flow` specify the low frequency cut-off (Hz)

`--Fhigh` specify the high frequency cut-off (Hz)

`--nPSDsegment` specify number of data segments to estimate the PSD

`--lPSDsegment` specify length of each data segment to estimate the PSD

`--outputPath` specify the directory where the output will be stored. Default is running directory.

`--cache` specify the cache files to run. Overrides most of the information form SPINspiral.input.data.

`--channel` specify the list of channels to run, e.g. [H1:LSC-STRAIN,L1:LSC-STRAIN].

`--PSDstart` specify the GPS time for the start of the PSD. The default value is the begining of the cache file.

4 Input files

SPINspiral has six different ASCII input files. The main input file is assumed to be called `SPINspiral.input`, but can be specified differently using the command-line option `-i <main input file name>`. The names of the other five input files are specified in the main input file. Example input files can be found in the `doc/input_*` directories in the svn trunk. The directory `doc/input_all` contains the four waveform-independent files, the other `doc/input_*` directories contain the two files that specify the injection and MCMC templates.

4.1 SPINspiral.input

The file `SPINspiral.input` is the main input file and specifies the main mode of operation and the names of the other input files.

4.1.1 Operation and output

doSNR Calculate the SNR: 0-no, 1-yes. Default: 1.

doMCMC Do MCMC: 0-no, 1-yes. Default: 1.

doMatch Calculate matches: 0-no, 1-yes. Default: 0.

intscrout Print initialisation output to screen: 0-no, 1-yes. Default: 0.

writeSignalWrite signal, noise, PSDs to file: 0-no, 1-yes. Default: 0.

printMuch Print long stretches of output (1) or not (0). Default: 0.

4.1.2 Secondary input files

mcmcFilename Name of the MCMC input file. Default: SPINspiral.input.mcmc.

dataFilename Name of the data/noise input file. Default: SPINspiral.input.data.

injectionFilename Name of the software injection input file. Default: SPINspiral.input.injection.

parameterFilename Name of the MCMC parameter input file. Default: SPINspiral.input.parameters.

systemFilename Name of the file with system-dependent parameters. Default: SPINspiral.input.system.

4.2 SPINspiral.input.mcmc

The MCMC input file

4.2.1 Basic settings

nIter Total number of iterations to be computed, *e.g.* 10^7.

thinOutput Number of iterations to be skipped between stored steps (100 for 1d).

thinScreenOutput Number of iterations between screen outputs in the MCMC (1000 for 1d).

MCMCseed Random number seed to start the MCMC: 0-let system clock determine seed, > 0: use the specified seed. Default: 0.

adaptiveMCMC Use adaptation: 0-no, 1-yes. Default: 1.

acceptRateTarget Target acceptance rate for MCMC (0.0-1.0). We used 0.25 for a long time.

minlogL Minimum value for the log Likelihood to accept a jump. We used 0 for a long time, this number shouldn't be positive! Try -1.e3.

blockFrac Fraction of uncorrelated updates that is updated as a block of all parameters (≤ 0.0: none, ≥ 1.0: all). Default: 0.1.

4.2.2 Correlated update proposals

correlatedUpdates Do correlated update proposals: 0-no, 1-yes but update the matrix only once, 2-yes and update the matrix every `ncorr` iterations. Default: 2.

corrFrac Fraction of update proposals that is correlated (0.0-1.0, 0.7 seems OK). corrupd must be 2. Should this replace corrupd? Default: 0.7.

nCorr Number of iterations for which the covariance matrix is calculated. Default: $10^3 - 10^4$.

matAccFr Fraction of elements on the diagonal that must 'improve' in order to accept a new covariance matrix. ??? 0.6-0.8 for unimodal, 0.0-0.2 for multimodal??? Default: 0.5.

prMatrixInfo Print information to screen on proposed matrix updates: 0-none, 1-some (default), 2-add the old and new matrices. Default: 1.

4.2.3 Annealing

annealTemp0 Starting temperature of the chain, *e.g.* 100.0. Set 1.0 for no temperature effect. Default: 1.0.

annealNburn Number of iterations for the burn-in phase (1e4) at this number, the temperature drops to 1.0. Default: 10^5.

annealNburn0 Number of iterations during which temp=temp0 (e.g. 0.1*annealNburn, should be lower than 0.9*annealNburn). Default: 10^5.

4.2.4 Parallel tempering

parallelTempering Use parallel tempering: 0-no, 1-auto, fixed T ladder, 2-auto, sinusoid T ladder, 3-manual, fixed T ladder, 4-manual, sinusoid T ladder. For a manual ladder, see near the bottom of the file. Default: 2.

nTemps Number of steps in the temperature ladder for parallel tempering, typically 5-10. Default: 5.

maxTemp Maximum temperature in automatic parallel-tempering ladder (equidistant in $\log(T)$), typically 20-100. Default: 40.

saveHotChains Save hot ($T > 1$) parallel-tempering chains: 0-no (just the $T = 1$ chain), $>$ 0-yes; for every saved $T = 1$ point, save every savehotchains-th hot point. Default: 100.

prParTempInfo Print information to screen on the temperature chains: 0-none, 1-some ladder info (default), 2-add chain-swap matrix. Default: 2.

4.2.5 Manual temperature ladder for parallel tempering (tempLadder[])

At least `nTemps` increasing temperature values, starting with 1.0, *e.g.* `1.00 2.00 4.00 8.00 16.00`

4.3 SPINspiral.input.data

The data/noise input file

4.3.1 General

datasetName Name of the data set used (for printing purposes), up to 80 characters, *e.g.* NINJA data set

4.3.2 Detector network

networksize Set the number of detectors that make up the network; read in networksize block of IFO data below (Currently 1–3)

selectifos Select the IFOs to use 1: H1, 2: L1, 3: V, *e.g.* `1 2 3`

4.3.3 Data handling

downsamplefactor Downsample the sampling frequency of the detector (16-20kHz for the detectors, 4kHz for NINJA) by this factor. Default (for detectors): 4.0. 10+1.4Mo needs $\sim 16\times$ a¡0.1, 8x: $a \leq 0.8$, 4x: $a > 0.8$. **Notice the difference of a factor of ~ 4 in the original sampling rate between detector data and *e.g.* the NINJA data files!**

databeforetc The stretch of data in seconds before presumed coalescence that is read in as part of the data segment, *e.g.* 6.0.

dataaftertc The stretch of data in seconds after presumed coalescence that is read in as part of the data segment, *e.g.* 1.0.

lowfrequencycut Templates and overlap integration start at this frequency, *e.g.* 40.0.

highfrequencycut Overlap integration ends at this frequency, *e.g.* 400.0.

tukeywin Parameter for Tukey-window used in `dataFT()` (non-flat fraction of window); Use 0.15 for Virgo data. Default: 0.15.

4.3.4 Noise PSD estimation

PSDsegmentNumber Number of data segments used for the PSD estimation. Default: 32, quick test runs: 8 or 4.

PSDsegmentLength Length of each data segment used for PSD estimation. Default: 8.0 $\rightarrow$ 32x8.0s = 256s, quick test runs: 4.0 or 2.0 (8x2.0 = 16s; 4x1.0s = 4s).

4.3.5 IFO i

Parameters for the location, orientation and data file for each detector used (see also Sect. 7):

name Detector name, *e.g.* Hanford.

lati Latitude (degrees), *e.g.* 46.45.

longi Longitude (degrees), *e.g.* 119.41.

rightarm Orientation of the 'right' arm (degrees), *e.g.* 36.80.

leftarm Orientation of the 'left' arm (degrees), *e.g.* 126.80.

ch1name Name of the data channel in the Frame file, *e.g.* `H1:STRAIN` (see Sect. 7).

ch1filepath Subdirectory of the path in `SPINspiral.input.system` where the data sits. Use "." for no subdirectory. Default: ".".

ch1fileprefix Prefix of the Frame data file name, *e.g.* `H-H1_NINJA_NOISE` (see Sect. 7).

ch1filesuffix Suffix of the Frame file name, *e.g.* `-1024.gwf` (see Sect. 7).

ch1filesize 'Size' (in seconds) of the data in each data Frame file, *e.g.* 1024.

ch1fileoffset If the Frame file name ends in: -839366009-128.gwf (where 128 is the length of the data stream), `fileoffset` = `mod(839366009,128)`, *e.g.* 743 (see Sect. 7).

ch1doubleprecision Data in data Frame file is double precision (1) or not (0). Default: 0.

add2channels Keep 0, unless you want to read a signal from file. Default: 0.

noiseGPSstart GPS time to start reading data to generate a noise PSD, *e.g.* 894377200.

noisechannel Name of the data channel in the Frame file, *e.g.* `H1:STRAIN`.

noisefilepath Directory of the path in `SPINspiral.input.system` where the noise files sit. Use "." for no subdirectory. Default: ".".

noisefileprefix Prefix of the Frame data file name, *e.g.* `H-H1_NINJA_NOISE`.

noisefilesuffix Suffix of the Frame data file name, *e.g.* `-1024.gwf`.

noisefilesize 'Size' (in seconds) of the data in each noise Frame file, *e.g.* 1024.

noisefileoffset Modulo between the start time and length of a Frame file. If the Frame file name ends in: `-839366009-128.gwf`, `fileoffset` = `mod(839366009,128)`, *e.g.* 743.

noisedoubleprecision Data in data Frame file is double precision (1) or nor (0). Default: 0.

4.4 SPINspiral.input.injection

The software injection input file

4.4.1 General

injectSignal Inject a signal into the data (1) or not (0).

injectionWaveform Waveform version used for the software injection: 1 for 1.5PN 12-parameter Apostolatos, 2 for 3.5PN 12-parameter LAL, 3 for 3.5PN 15-parameter LAL.

injectionPNorder Post-Newtonian order at which the injection signal should be generated, *e.g.* 1.5, 2.0, 3.5.

injectionSNR If > 0: scale the distance such that the injection network SNR becomes injectionSNR.

injRanSeed Random number seed for random injection parameters. Don't change between serial chains of the same run! Default: 12345.

4.4.2 Table: Parameters

Following the "General" section is a table with nine columns containing the following data:

Number Currently just to guide the eye in this file.

ID a unique number for each parameter. See Section 6 for a list of current IDs.

InjectionValue injection value for the parameter, can be overwritten depending on `ranInjPar`.

RanInjPar randomise the injection value:

- **0** no; inject InjectionValue.
- **1** yes; inject random value from a Gaussian distribution with centre InjectionValue and width Sigma (min BoundLow, max BoundUp).
- **2** yes; inject random value from range determined by BoundLow-BoundUp (make sure your MCMC prior matches this!).

Sigma width of the Gaussian distribution to draw from for `RanInjPar=1`, *e.g.* 0.1.

BoundType type of boundaries to use for the ranges of the injection parameters:

- **1** general range, BoundLow-BoundUp
- **2** general range, InjectionValue + BoundLow - InjectionValue + BoundUp; BoundLow must be ≤ 0, BoundUp must be ≥ 0.
- **3** general range, InjectionValue * BoundLow - InjectionValue * BoundUp; BoundLow must be ≤ 1, BoundUp must be ≥ 1.

BoundLow, Up : used to determine upper or lower bound for BoundType = 1,2.

Description of the parameter, to increase readability of the file.

4.5 SPINspiral.input.parameters

4.5.1 General

mcmcWaveform Waveform version used as MCMC template: 1 for 1.5PN 12-parameter Apostolatos, 2 for 3.5PN 12-parameter LAL, 3 for 3.5PN 15-parameter LAL, 4 for non-spinning LAL.

mcmcPNorder Post-Newtonian order at which the MCMC template should be used, *e.g.* 1.5, 2.0, 3.5.

priorSet Set of priors to use, currently: 1.

offsetMCMC Start the MCMC with offset initial parameters: 0-no: use injection parameters, overrules Start below; 1-yes: use Start below. Default: 1.

offsetX Start the MCMC from a Gaussian distribution with a width of (offsetX times Sigma), if Start==2,4 below. Default: 10.0.

4.5.2 Parameters

Number Currently just to guide the eye in this file.

ID a unique number for a parameter. See Section 6 for a list of current IDs.

BestValue best value for the parameter, *e.g.* from the trigger; use to start from or close to.

Fix fix an MCMC parameter (i.e., keep it constant throughout the MCMC run):

0 no.

1 yes; fix to the starting value determined by `Start`

Start where to start the Markov chains:

1 start at best value.

2 start near best value (Gaussian distribution with width sigma).

3 start at injection value.

4 start near injection value (Gaussian distribution with width sigma).

5 start randomly from range determined by BoundLow-BoundUp (see prior type).

Sigma width of the Gaussian distribution to start from for start=2,4; also used for diagonal of first correlation matrix.

4.5.3 Priors

Type Type of prior range:

11 general range, BoundLow-BoundUp.

12 general range, best value+BoundLow - best value+BoundUp; BoundLow must be ≤ 0, BoundUp must be ≥ 0.

13 general range, best value*BoundLow - best value*BoundUp; BoundLow must be ≤ 1, BoundUp must be ≥ 1.

14 general range, injection value+BoundLow - injection value+BoundUp; BoundLow must be ≤ 0, BoundUp must be ≥ 0.

15 general range, injection value*BoundLow - injection value*BoundUp; BoundLow must be ≤ 1, BoundUp must be ≥ 1.

21 periodic boundaries $0 - 2\pi$; BoundLow, BoundUp will be ignored.

22 periodic boundaries $0 - \pi$; BoundLow, BoundUp will be ignored.

BoundLow/Up used to determine upper or lower bound, depending on prior type. Ignored for periodic boundaries.

Description symbol/name of the parameter, to increase readability of the file.

4.6 SPINspiral.input.system

A file with system-dependent parameters (well, currently one).

datadir Data directory (actual data files may be in a subdirectory of this, see `SPINspiral.input.data`), *e.g.* `/home/user/MCMC/data`

5 Output files

5.1 SPINspiral.output.123456.12

6 Parameter catalogue

Goal: assign a unique identifier to each possible variable that has been used in SPINspiral so far. The identifier is stored in the variable `parID[]` and `injID[]` in the `runPar` and `mcmcvariables` structs. The reverse identification is stored in `parRevID[]` and `injRevID[]`. Thus, if the first parameter is t_c, then `parID[0]`=11 and `parRevID[11]`=0. Whether a parameter is used (1) or not (0) for the software injection or MCMC is stored in `injParUse[]` and `mcmcParUse[]`. The arrays `parAbrev[]` and `parAbrv[]` contain short and ultrashort parameter names respectively. The variable `parDef[i]` signifies whether the parameter with ID i is defined (1) or not (0).

6.1 Time

Number	Variable	Description	Unit	Range
11	t_c	GPS time of coalescence	s	$[0,\infty[$
12	$t_{40\,\mathrm{Hz}}$	GPS time at 40 Hz	s	$[0,\infty[$

6.2 Distance

Number	Variable	Description	Unit	Range
21	d_L^3	Luminosity distance	Mpc^3	$]0,\infty[$
22	${}^\mathrm{e}\log(d_\mathrm{L}/\mathrm{Mpc})$	Luminosity distance	—	$]-\infty,\infty[$

6.3 Sky position

Number	Variable	Description	Unit	Range
31	R.A. (α)	Right ascension	rad	$[0,2\pi[$
32	sin(Dec) ($\sin(\delta)$)	Declination	—	$[-1,1]$

6.4 Phase

Number	Variable	Description	Unit	Range
41	$\varphi_\mathrm{orb,c}$	Orbital phase at coalescence	rad	$[0,2\pi[$

6.5 Orientation

Number	Variable	Description	Unit	Range
51	$\cos(\iota)$	Inclination	—	$[-1,1]$
52	ψ	Polarisation angle	rad	$[0,\pi[$
53	$\sin(\theta_{J_0})$	'Declination of J_0'	—	$[-1,1]$
54	φ_{J_0}	'R.A. of J_0'	rad	$[0,2\pi[$

6.6 Mass

Number	Variable	Description	Unit	Range
61	$\mathcal{M}$	Chirp mass	$M_\odot$	$[0,\infty[$
62	η	Symmetric mass ratio	—	$[0,0.25]$
63	M_1	Mass 1	$M_\odot$	$[0,\infty[$
64	M_2	Mass 2	$M_\odot$	$[0,\infty[$
65	$\mathcal{M}^{1/6}$	(Chirp mass/$M_\odot$)$^{1/6}$		$[0,\infty[$
66	M_tot	Total mass	$M_\odot$	$[0,\infty[$
67	q	Mass ratio (M_1/M_2)	—	$[0,\infty[$

6.7 Spin 1

Number	Variable	Description	Unit	Range
71	$a_\mathrm{spin,1}$	Spin magnitude 1	—	$[0,1]$
72	$\cos(\theta_\mathrm{spin,1})$	Spin tilt 1	—	$[-1,1]$
73	$\varphi_\mathrm{spin,1}$ (α_c)	Spin phase 1	rad	$[0,2\pi[$
75	$S_\mathrm{1,x}$	Spin 1 magnitude, x-component	—	$[0,1]$
76	$S_\mathrm{1,y}$	Spin 1 magnitude, y-component	—	$[0,1]$
77	$S_\mathrm{1,z}$	Spin 1 magnitude, z-component	—	$[0,1]$

6.8 Spin 2

Number	Variable	Description	Unit	Range
81	$a_{\mathrm{spin},2}$	Spin magnitude 2	—	$[0, 1]$
82	$\cos(\theta_{\mathrm{spin},2})$	Spin tilt 2	—	$[-1, 1]$
83	$\varphi_{\mathrm{spin},2}$	Spin phase 2	rad	$[0, 2\pi[$
85	$S_{2,\mathrm{x}}$	Spin 2 magnitude, x-component	—	$[0, 1]$
86	$S_{2,\mathrm{y}}$	Spin 2 magnitude, y-component	—	$[0, 1]$
87	$S_{2,\mathrm{z}}$	Spin 2 magnitude, z-component	—	$[0, 1]$

6.9 Merger, ringdown

Number	Variable	Description	Unit	Range
91				$[,]$

7 Reading Frame files

Frame files are designed to store detector data and are also used to store synthetic noise. They can be read using `libframe` [?], see *e.g.* `/export/apps/lsc/libframe/` on Fugu.

SPINspiral can read Frame files. The parameters in `SPINspiral.input.data` (Sect. 4.3.5) determine how this is done.

The name of a Frame file typically looks like `H-H1_SOME_TEXT-123456789-123.gwf` (using a file with detector data from H1 as an example). The first part (`H-H1_SOME_TEXT-`) goes into the parameter `ch1fileprefix`, while the last part (`-123.gwf`) is stored in `ch1filesuffix`.

The example number `123456789` indicates the starting GPS time (in seconds) of the data in the file, while `123` is the length of the data strech in the file, again in seconds (and again an example). In order for the code to know at which GPS time Frame files start and stop, we need to feed the modulus of the two numbers (`mod(123456789,123) = 90` here) to the parameter `ch1fileoffset`. At the moment, we cannot read subsequent files of different length (using `FrCopy`, which can copy multiple files into one, is a quick-and-dirty solution. The ability to read cache files would be a better one).

One of the more obscure parameters is the channel name `ch1name`. It can be found using the program `FrDump`, which comes with `libframe`:

```
FrDump -i H-H1_SOME_TEXT-123456789-123.gwf |grep ProcData
```

The channel name is probably going to look like `H1:STRAIN` (for the case of the H1 detector).

References