Extracting Astrophysical Parameters from Gravitational-Wave Observations

Karla Guardado
MIT Class of 2015
Mentors Alan Weinstein & Vivien Raymond
LIGO Caltech SURF
November 1, 2013

I. ABSTRACT

LIGO (Laser Interferometer Gravitational-Wave Observatory) is engaged in the search for gravitational waves. An expected class of sources are compact binary systems that lose energy through gravitational radiation during coalescence, just as predicted by Einstein's theory of general relativity. In this project, we wished to identity these gravitational-wave signatures. I worked on data analysis for LIGO at the California Institute of Technology (Caltech), particularly on parameter estimation. We used a number of post-Newtonian based approximations to describe the coalescences; these are methods of approximating solutions to Einstein's field equations—the equations of general relativity. For this project, we focused on a simulated black hole-black hole binary. We compared several approximants and extracted all necessary parameters. Determination of these parameters provided insight into the underlying physics, as well as how important choice of approximant on parameter recovery for spinning signals was within known limitations of each approximant. Using inference software developed within the LIGO-VIRGO collaboration, we also sought to evaluate its performances in this regard.

II. INTRODUCTION

Compact binary systems lose energy through the emission of gravitational waves during coalescence, just as predicted by Einstein's theory of general relativity. In this project, we wished to identity these gravitational-wave signatures. Compact binary coalescences are desirable targets from which to extract astrophysical parameters because LIGO can expect relatively high detection rates from these. There exist three types: neutron star - neutron star, black hole - neutron star, and black hole – black hole. The first of these has a comparatively low mass and can be tidally disturbed as well as emit light. However, there exists no astrophysical evidence for the last of these, mostly because they are harder to detect as they have none of the aforementioned properties. Parameter estimation allows us to assemble better models of massive star evolution and population. Determination of these parameters provides insight into the underlying physics of compact binary systems. There exist a total of nine parameters we look for in non-spinning binaries, eleven for binaries with aligned spins, and fifteen for binaries with misaligned spins. These parameters can also be split into two types: intrinsic and extrinsic. Intrinsic parameters govern the shape (i.e., phase and amplitude evolution) of the waveform. Extrinsic parameters govern the overall observed amplitude of the waveform. Examples of parameters include masses of the components, spins and angular momenta, and locations in the sky. Mass measurements could, in general, help us identify the maximum mass of neutron stars, minimum

mass of stellar black holes, neutron star equation of state, and thus presence or absence of the "mass-gap" [1] between the heaviest neutron stars and the lightest black holes seen in simulations of the core collapse of massive stars. Spin measurements could also, in general, provide insight into binary formation scenarios and supernovae processes. And the locations could tell us about the host galaxies and the environments in which these form. [2] These results, of course, will not be available until our detectors are in place—an estimated two years in the future. So for this project, we worked with simulated signals so that we could test and measure the extent of our recovery abilities in preparation for 2015.

We used a number of approximation models to describe the coalescences. Of interest for this project were the approximants known as SpinTaylorT4 and SpinTaylorT2 [3], which we sought to compare and observe differences in injections and recoveries. One can compare parameter estimation approximants, measuring biases from one to the other and characterize the posterior for different classes of signals. This includes different SNRs (signal to noise ratio), masses, locations, etc. [4] The *Optimal Network SNR* is defined as:

$$SNR = \sqrt{\sum_{det} \int_{fLow}^{fHigh} \frac{|s_{\mathrm{det}}(f, \overrightarrow{\theta})|^2}{S_{det}(f)} df} \quad (1)$$

with each detector given as det so that s_{det} is the signal in that detector and $S_{det}(f)$ is the noise power spectral density (PSD) function for that detector. This way, we can know beforehand what kind of information can be gained by what kind of signals, how, and at what costs.

As no exact solution of Einstein's field equations is known that describes gravitational waves, we employ these post-Newtonian approximations. While known to work for slow speeds relative to the speed of light and weak gravitational fields, they have also proven remarkably effective in describing certain strong-field, fast motion systems, i.e. the inspiral phase of binary black hole coalescence. Thus, it is important for us to understand how and what kind of results we get from using them for when and if we detect gravitational waves, or risk inaccurate or imprecise physics.

I worked with inference software developed within the LIGO-VIRGO collaboration and worked to evaluate its performance in terms of parameter estimation. The code is a C implementation of several parameter estimation algorithms, driven by the Markov-chain-Monte-Carlo (MCMC) methods [5].

The inference software is an application of Bayesian inference. An interpretation of statistics that expresses probabilities in terms of "degrees of belief." Bayesian inference uses Bayes' theorem [6] to calculate the posterior probability density of a parameter set $\overrightarrow{\theta}$, given dataset $\{d\}$ under a model H,

$$p(\vec{\theta}|\{d\}, H) = \frac{p(\vec{\theta}|H)p(\{d\}|\vec{\theta}, H)}{P(\{d\}|H)} \quad (2)$$

where $p(\vec{\theta}|H)$ is the *prior* probability density function—describing knowledge about the parameters within a model H before the data is analyzed, and $p(\{d\}|\vec{\theta}, H)$ is the *likelihood* function—denoting the probability under the model H of obtaining the dataset $\{d\}$ for a given parameter set $\vec{\theta}$.

[7] The MCMC methods are a class of algorithms for sampling from high dimensional parameter spaces based on constructing a Markov chain that sets the desired distribution as its equilibrium distribution. [8] A Markov chain is a conditionally independent collection of random variables. The Monte Carlo methods rely on repeated sampling to obtain PDFs. It is a random process, in which the next state depends only on the current state (this "memory loss" is known as the Markov property). When confined to parameter

space, the sample points are drawn from the desired and known distribution of the parameters. This results in a powerful stochastic sampling method, in which the quality of the sample improves as a function of the number of steps [9]. Used here, the MCMC methods estimate multidimensional probability distributions for our Bayesian posterior and the parameters.

III. METHODS

For this project, we focused on a simulated black hole-black hole binary. We compared several approximants and extracted all necessary parameters to determine how well each approximant recovered them and under what circumstances. We judged the results based on accuracy and precision. To be more succinct, with accuracy, I am referring to the degree of 'closeness' to the actual values. As with precision, I am referring to the error of a given measurement—the less of which is the more precise. [10]

After familiarizing myself with the project, I proceeded to install a Scientific Linux partition on my PC that would be beneficial to future development tasks, as well as a plethora of LIGO-specific software. After acquiring access to our data clusters, I learned about Linux computing and LIGO-specific computing on our clusters.

I ran and post-processed a number of jobs on the clusters. In particular, there were three that I ran and processed so that I could edit a tutorial that demonstrates how to run them. Created by my mentor, Vivien Raymond, I had been asked to tailor it for the benefit of the more unfamiliar user.

I worked with some python code that when run generates injection files (simulated data). I made several of these injections and ran quick jobs on each of them, recovering each with a SpinTaylorT4 and SpinTaylorT2 approximant. These were processed to extract all parameters. On a few occasions, run outputs were found to be lacking, in which case greater processing power was allotted as needed.

IV. RESULTS

Here is an example for a black hole - black hole binary including spins. One member of the binary was five solar masses and the other ten solar masses. This is enough mass for the detectors to be sensitive, but not enough as to involve the merger or ring-down phases of coalescence. This signal, labeled event0, was run four times to include the two combinations of approximants for both injections and recoveries.

Figure 1 is a two-dimensional plot of the masses with "zero noise." Zero-noise is an approximation where the noise level of the detector is set to zero, while keeping the expected PSD from equation (1). This approximation has the effect of producing the same results as if we ran over many different

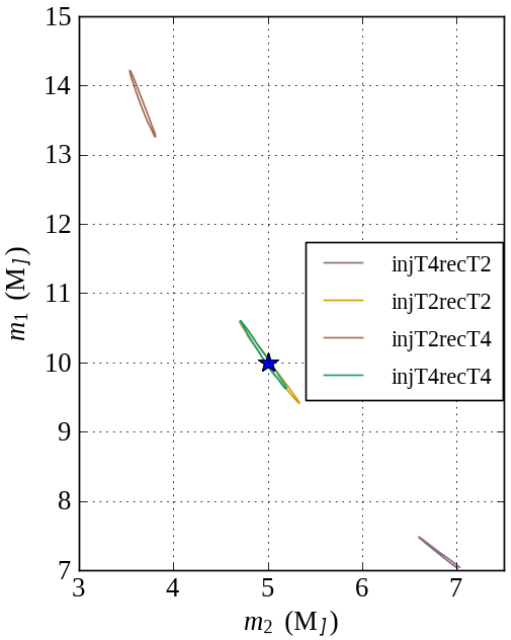

Figure 1 A two dimensional plot of the masses for all four combinations of injections and recoveries. The label "injT4recT2" stands for the SpinTaylorT4 approximant used to inject the simulated signal and the SpinTaylorT2 approximant used as the recovery.

realizations of the noise and averaged the Probability Density Functions. It has also been rescaled for an SNR of about twenty.

In general, all other parameters were presented in the same way, as Figures 2 and 3 can illustrate. For each run, all other parameters were fixed to their randomly generated injection values.

Please reference Appendix I for the tutorial I wrote and for an additional brief tutorial on how to run jobs on a pipeline. The tutorial explains how to run both hardware and injection events with examples.

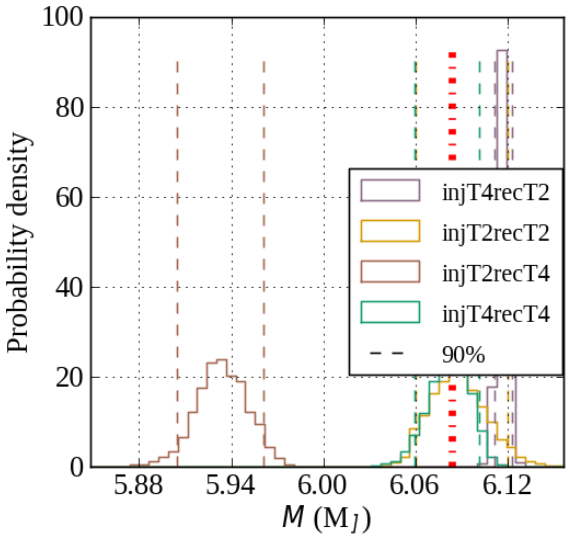

Figure 2 The probability density distribution for the symmetric mass ratio parameter, $\eta = (m_1 + m_2)/(m_1 m_2)$.

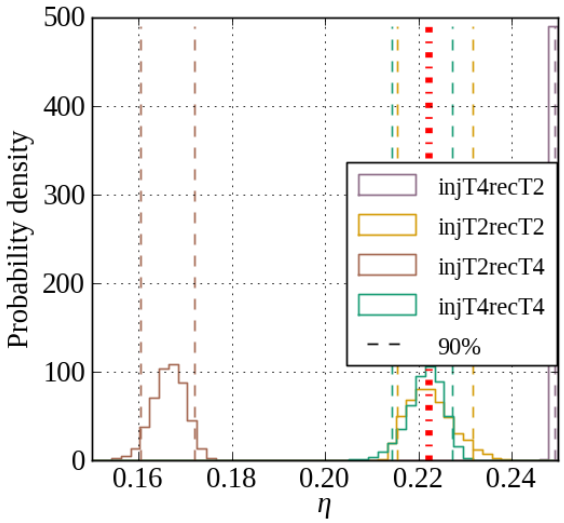

Figure 3 The probability density distribution for the chirp mass parameter, $M = (m_1 m_2)^{3/5}/(m_1 m_2)^{1/5}$.

V. CONCLUSIONS

As Figures 1, 2, and 3 all illustrate in this particular example, when we injected and recovered with the same approximation, we recovered the correct parameter range. However, when we injected and recovered with different combinations of approximants, we did not recover the correct parameter range. This provides some considerations for LIGO in the future.

As a follow up, we might consider providing a more complete simulation, identifying regions of parameter space where the difference in results is significant, compare more and different approximants, and include more parameters and events.

VI. ACKNOWLEDGEMENTS

I would like to thank Caltech and LIGO SURF for giving me the opportunity to work on such a fascinating project at a wonderful institution. I would also like to thank MURF (Minority Undergraduate Research Fellowship) for extending a warm welcome to me into their family. I would like to extend my gratitude to the National Science Foundation and the NSHP (National Society for Hispanic Physicists) for awarding me the Victor Blanco Fellowship, making this opportunity possible. And of course, a big thank-you to deserving my mentors Alan Weinstein, Vivien Raymond, and Kent Blackburn for their continual support.

VII. REFERENCES

[1] Belczynski, K. et al. 17 October 2011. Missing Black Holes Unveil the Supernova Explosion Mechanism. Astronomical Observatory, Warsaw University, Poland. http://astrobites.org/2011/10/17/mind-the-black-hole-mass-gap/

[2] Aasi, J. et al. 5 April 2013. Parameter Estimation for Compact Binary

Coalescence Signals with the First Generation Gravitational-Wave Detector Network. http://arxiv.org/pdf/1304.1775v1.pdf

[3] Blanchet, Luc. 1 June 2006. Gravitational Radiation from Post-Newtonian Sources and Inspiraling Compact Binaries. http://relativity.livingreviews.org/Articles/lrr-2006-4/download/lrr-2006-4Color.pdf

[4] Raymond, V. et al. 2 April 2009. Degeneracies in Sky Localization Determination from a Spinning Coalescing Binary through Gravitational Wave Observations: A Markov-Chain Monte-Carlo Analysis for Two Detectors. http://arxiv.org/pdf/0812.4302.pdf

[5] "Markov chain Monte Carlo." *Wikipedia: the Free Encyclopedia*. Wikimedia Foundation, Inc. 29 July 2013. http://en.wikipedia.org/wiki/Markov_chain

[6] "Markov chain." *Wikipedia: the Free Encyclopedia*. Wikimedia Foundation, Inc. 22 April 2013. http://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo

[7] Data Analysis Software Working Group. LSC Algorithm Library Suite. 1999. https://www.lsc-group.phys.uwm.edu/daswg/projects/lalsuite.html

[8] "Bayes' Theorem." *Wikipedia: the Free Encyclopedia*. Wikimedia Foundation, Inc. 12 May 2013. http://en.wikipedia.org/wiki/Bayes'_theorem#Bayes.27_rule

[9] "Accuracy and Precision." *Wikipedia: the Free Encyclopedia*. Wikimedia Foundation, Inc. 6 May 2013. http://en.wikipedia.org/wiki/Accuracy_and_precision

[10] Schutz, B.F. 9 November 1999. Gravitational wave astronomy. http://arxiv.org/pdf/grqc/9911034v1.pdf

APPENDIX I

This is the tutorial I wrote. “Big-Dog” was a blind hardware injection and S6PE-event0 was a blind software injection.

Running

Example: Investigating The Trigger, The Event, The Big/Black/Blind Dog, at GPS 968654557.95

- First you need a cache file per detector. Execute in the directory you wish to run:

```
ligo_data_find -o H -t H1_LDAS_C02_L2 -s 968654500 -e 968654850 -l | grep localhost/archive > 968654557-H1.cache
ligo_data_find -o L -t L1_LDAS_C02_L2 -s 968654500 -e 968654850 -l | grep localhost/archive > 968654557-L1.cache
ligo_data_find -o V -t HrecOnline -s 968654500 -e 968654850 -l | grep localhost/archive > 968654557-V1.cache
```

This will create the 3 cache files 968654557-[H,L,V]1.cache in your working directory. Ending at GPS time 9968654850 will leave enough room for the PSD estimation (256 seconds by default). (We are not using the time before The Trigger because of the glitch in L1).

Alternatively, if you have already downloaded the data, you can run the following:

```
ls H*.gwf | lalapps_path2cache > 968654557-H1.cache
ls L*.gwf | lalapps_path2cache > 968654557-L1.cache
ls V*.gwf | lalapps_path2cache > 968654557-V1.cache
```

Otherwise, you can run the code with the argument

```
-np 16 lalinference_mcmc --IFO [H1,L1,V1] --cache [968654557-H1.cache,968654557-L1.cache,968654557-V1.cache] --PSDstart
968654559.950 --PSDlength 256 --srate 2048 --seglen 8 --trigtime 968654557.95 --channel [H1:LDAS-STRAIN,L1:LDAS-
STRAIN,V1:h_16384Hz] --Niter 100000000 --Neff 1000 --approx SpinTaylorT4twoPointFivePN --ampOrder newtonian --crazyInjectionHLSign
--trigSNR 12.5
```

out of your working directory. Instructions on how to submit a condor job can be found here: https://www.lsc-group.phys.uwm.edu/ligovirgo/cbcnote/JointS5/BayesianFollowUpLALInference_MCMC/karla/condor.

Above, a SpinTaylorT4 approximation was used, but our fastest approximant is the TaylorF2:

```
-np 8 lalinference_mcmc --IFO [H1,L1,V1] --cache [968654557-H1.cache,968654557-L1.cache,968654557-V1.cache] --PSDstart
968654559.950 --PSDlength 256 --srate 2048 --seglen 8 --trigtime 968654557.95 --channel [H1:LDAS-STRAIN,L1:LDAS-
STRAIN,V1:h_16384Hz] --Niter 100000000 --Neff 1000 --approx TaylorF2threePointFivePN
```

Thia example is by default non-spinning, but you can add aligned spins by appending

```
--aligned-spin
```

to the argument.

To run on simulated data

Example argument:

```
-np 8 lalinference_mcmc --IFO [H1,L1,V1] --cache [LALLIGO,LALLIGO,LALVirgo] --PSDstart 90000000 --PSDlength 256 --srate 2048 --
seglen 8 --trigtime 900000000 --Niter 100000000 --Neff 1000 --approx TaylorF2threePointFivePN --inj "insert injection xml file
name here" --event "event number" --dataseed "integer value"
```

The `--cache` option accepts `LALLIGO`, `LALVirgo` and `LALAdLIGO` as arguments. No `--channel` is needed.

Post-processing

After having installed and sourced pylal and glue from master, you can run:

```
cbcBayesPostProc.py -d PTMCMC.output."run ID".00 --deltaLogL=4.5 --downsample=10000 --lalinfmcmc --outpath=post --skyres=0.5 --
dievidence
```

in your working directory to post-process and get output. Specify as many `PTMCMC.output."run ID".00` as needed, and `--deltaLogL="number of parameters'/2`, i.e. `--deltaLogL=4.5` for non-spinning runs and `--deltaLogL=7.5` for fully spinning runs.

How to submit a condor job

Condor is what the LIGO Data Grid uses to run and process computational jobs. In order to submit a condor job, you will need to run the command `condor_submit "filename.sub"` where "filename.sub" is a condor submit file. Here is an example of one:

```
####################
# condor_vanilla_mpi.sub: MPI in the condor vanilla universe
#
# LIMITATIONS: this script will run a multi-threaded job on a single node only.
# 8 parallel tempering chains will be run on 8 cores. To change that number,
# you need to change it in 3 locations:
#  arguments = -np <number of chains> /data/ligo/<lalinference_mcmc location>
#  request_cpus = <number of chains>
#  request_memory = <number of chains>*2048
#
# You can automatically submit the same job n times by changing "queue 1" to "queue n"
#
# Questions/comments: vivien.raymond@ligo.org
#
####################
universe   = vanilla
getenv     = true
executable = /usr/lib64/openmpi/bin/mpirun
arguments  = -np 16 lalinference_mcmc --IFO [H1,L1,V1] --cache [968654557-H1.cache,968654557-L1.cache,968654557-V1.cache] --
PSDstart 968654559.950 --PSDlength 256 --srate 2048 --seglen 8 --trigtime 968654557.95 --channel [H1:LDAS-STRAIN,L1:LDAS-
STRAIN,V1:h_16384Hz] --Niter 100000000 --Neff 1000 --approx SpinTaylorT4twoPointFivePN --ampOrder newtonian --crazyInjectionHLSign
--trigSNR 12.5-approx SpinTaylorT4threePointFivePN
output     = lalinference-BigDog-$(Process).out
error      = lalinference-BigDog-$(Process).err
log        = /usr1/"user.name"/lalinference-BigDog-$(Process).log #change as appropriate
request_cpus = 8
request_memory = 8*2048

+Online_CBC_PE=True
Requirements = (TARGET.Online_CBC_PE =?= True)

queue 3
```

To check on your jobs once submitted, you can run `condor_q "user.name"`.

Alternatively, how to run jobs on a pipeline:

```
-First, you need an initialization file (a ".ini" file) in your working directory. For example,
/home/vivien/MBTAOnline/pipeline_test/lalinference_pipe_example.ini; I can run this exact command to include my mentor's .ini file
in my working directory.
-Next, you need to "source" (access) a pipeline. For example, I can run source /home/vivien/lsc/pipeline-beta/etc/lscsoftrc to
source my mentor's pipeline in my working directory.
-Then run the following command, specifying of course gid and .ini file path, as here:
lalinference_pipe --run-path ./ --gid G21187 --daglog-path/usr1/karla.guardado/../lalinference_pipe_example.ini
```

You can see output here: https://gracedb.ligo.org/events/.