THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF PHYSICS

TREEBANK: Differential Geometric Methods for Fast Template Bank Generation in Searches for Gravitational Waves

JONATHAN WANG
SPRING 2017

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Physics
with honors in Physics

Reviewed and approved* by the following:

Chad Hanna
Assistant Professor of Physics
Thesis Supervisor

Richard Robinett
Professor of Physics
Associate Head for Undergraduate and Graduate Students
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

Gravitational waves are propagating ripples in spacetime originating from non-spherically symmetric accelerating systems. A fundamental prediction of Einstein's theory of general relativity, they are the subject of the most sensitive scientific search in history due to fact the that their effect on Earth is minuscule, with detectable waves squashing and stretching spacetime on the order of 1×10^{-21} strain. On September 14, 2015, the advanced LIGO detectors made the first gravitational wave detection ever, observing the coalescence of two low-spin black holes of approximately $60M_{\odot}$ combined mass. On December 26, 2016, just a few months later, a second gravitational wave was observed from yet another black hole binary, this time of $22M_{\odot}$ total mass. The frequency of these events suggested that astrophysically significant sources of gravitational waves are even more prevalent than predictions estimated, indicating an extremely promising future for LIGO and gravitational wave astronomy. With a possibly bountiful universe of gravitational waves to observe, it is in the interests of the LIGO Scientific Collaboration (LSC) to expand the parameter space across which they can detect gravitational waves. The matched filtering process applied to the detection of compact binary coalescences (CBCs) has proven to be effective so far, but is limited to searches across the mass and z-spin parameters of binaries. This is in large part due to the computational costs and large amounts of time currently required to generate template banks for use in matched filtering. For the rest of this thesis I summarize the motivation, algorithm, and initial results of a new template bank generator for use in matched filtering searches for gravitational waves originating from CBCs. This method, dubbed "treebank", seeks to cut down on the computational cost and time required by the current template bank generator by orders of magnitude through clever applications of differential geometry and foundational ideas in computer science. Treebank utilizes a binary tree decomposition approach to split the bank into distinct hyper-rectangles of approximately constant metric until the expected template density of each of these rectangles is sufficient to cover a user defined minimum match. The placement of templates in these hyper-rectangles can be handled using a geometric approach, a stochastic approach, or by splitting the bank down until the expected template density per hyper-rectangle is only a single template.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I have many people to thank without whom I would have never made it to writing this acknowledgements section. First and foremost are my mom, dad, and sister, Vicki. Without the support of my family I would have never made it through college, much less a thesis. Second I would like to thank the Tri-Omega undergraduate physics group for so many hours of mutual moral support through the darkest hours of our physics education. Third the Schreyer Honors College, the Eberly College of Science, and Dr Robinett for their mentorship and financial support through the Schreyer Honors scholarship and the John and Elizabeth Holmes Teas scholarship. Finally, my most heartfelt thanks go out to the PSU LIGO Group, Stephen Privitera, and especially Chad Hanna, who have revealed to me an unbelievably interesting and exciting world of research through our inquiries into gravitational wave detection.

Chapter 1

Introduction

1.1 Gravitational Waves and the LIGO Project

In 1916 Einstein predicted the phenomenon of gravitational wave emission by accelerating massive objects as part of his general theory of relativity as an explanation of the gravitational force [1]. Gravitational waves are disturbances in the curvature of spacetime which propagate at the speed of light, creating a squashing/stretching effect in a plane perpendicular to the direction of propagation.

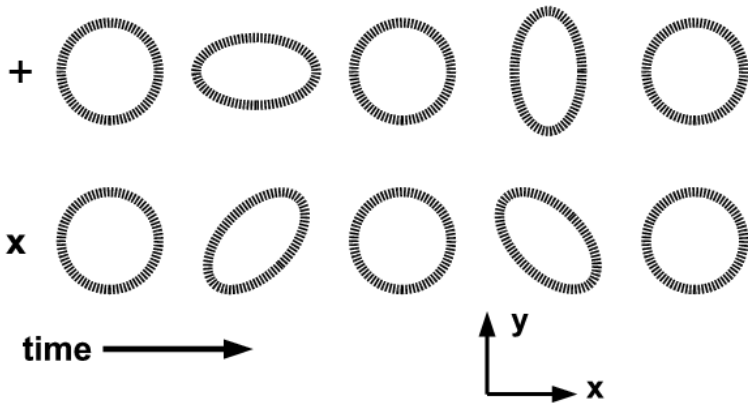

Figure 1.1: Visualization of the effects of a gravitational wave passing through a plane over time. There are two polarizations of gravitational waves called the plus and the cross polarizations. They are defined such that they are orthogonal to each other. In this image the plus polarization is depicted in the top row and the cross polarization on the bottom [2].

The first claim that gravitational waves had been detector came from Dr Joseph Weber of the University of Maryland in 1968 [3]. Weber's experiment utilized "bar" detectors, two aluminum cylinders 2 meters long and 1 meter in diameter spaced approximately 2 km apart. The bar detectors were suspended from steel wires and isolated from seismic and electromagnetic disturbances. A gravitational wave passing through the bars would cause it to resonate at about 1660 Hz, causing the bars to distort slightly. The distortion would be converted into an electric signal by piezoelectric sensors, which could be read of by Weber. Weber claimed to have detected gravitational waves emanating from a source near the center of the Milky Way. Alas, attempts to replicate his results were all unsuccessful and his claims were dismissed by the scientific community. However, his efforts were not entirely wasted, as they proved to be a starting point which attracted other physicists and scientists to tackle to immense challenge of detecting gravitational waves.

Indirect evidence for the existence of gravitational radiation was first put forth by Russell Hulse and Joseph Taylor of Princeton University in 1981 based on their observations in the change of orbit of the pulsar binary system PSR 1993+16. The change in orbit agreed with the amount of energy general relativity predicted such a system would lose to gravitational radiation to within 0.3 percent [4]. The discovery was significant enough that Hulse and Taylor were awarded the 1993 Nobel Prize in Physics. However, the first direct observation of gravitational waves was still a number of years off.

Nowadays the search for gravitational waves is being carried out by the LIGO detectors and the LIGO Laboratory, with research and data analysis being conducted by the LIGO Scientific Collaboration. The LIGO detectors are, in essence, a pair of Michelson interferometers. A Michelson interferometer fundamentally works by splitting a laser into two beams which travel down orthogonal paths. The beams are reflected back towards the splitter and are recombined and examined. As gravitational waves propagate through the detector the length of the detector arms are changed, such that the differences in path length traveled by the split laser can be observed by examining the phase difference of the recombined laser [5].

The LIGO observation project was planned to take place in two phases: an initial pair of interferometers which would carry out scientific runs with no absolute expectations to detect gravitational waves. After these initial runs, the observatories would undergo upgrade to become advanced LIGO [6], which, in the most sensitive frequencies of the detectors, would be 10 times more sensitive than initial LIGO]. From 2002-2010 initial LIGO ran six scientific runs, known as S1-S6, with no detections being made during that time [7] [8]. Construction of advanced LIGO was completed during the summer of 2015, during which it began its first observing run, known as O1. Since then, O1 has been completed, with advanced LIGO currently in the midst of its second observing run, O2.

The NSF commissioned LIGO project seeks to detect gravitational gravitational waves with interferometer sites at Hanford, WA and Livingston, LA. In addition, a third advanced LIGO site is planned for construction in India based upon recommendations that such a project will improve our ability to localize the sources of gravitational waves in the sky [9]. Currently on its second observing run, LIGO seeks to detect and analyze gravitational waves from a host of interesting sources, from the rotation of deformed neutron stars to the cores of supernovae [10]. By far the largest barrier to consistent gravitational wave detection is the strength of the waves produced. The strength of a gravitational wave can be quantized by the fraction by which it deforms the spacetime it passes through, or strain. The most propitious sources of detectable gravitational waves are energetic rotating systems with large spherical asymmetry. The direct observation of gravitational waves comes from detecting their strain, that is the fractional change in spacetime caused by the wave. The strain of a gravitational wave originating from a rotating, spherically asymmetric system can be estimated by [11]

$$h \sim \frac{1}{c^2}\frac{4G(E_{kin}^{ns}/c^2)}{r} \tag{1.1}$$

where E_{kin}^{ns} is the kinetic energy of the non-spherically symmetric components of the system and r is the distance the wave has traveled. This strain is the path difference the laser of the interferometers travel. Calculations based on Equation 1.1 set the target sensitivity of the LIGO detectors at $h \sim 10^{-21}$ to 10^{-22} [11]. Considering that the diameter of a proton is on the order 10^{-15} m, gravitational wave detection is a daunting task.

Considering the incredible sensitivity and human effort required to observe gravitational waves, it is natural to ask why this work is significant and worth pursuing. Firstly, the chance to study gravitational waves directly grants us the ability to probe general relativity even farther. Direct observation allows us to compare reality against theory and either confirm or correct the natural

laws of gravitation physicists have derived.

Additionally, and perhaps more importantly, gravitational radiation provides us with a source of cosmic information completely separate from electromagnetic waves. EM waves are prone to scattering and absorption and present information us on scales of 10^7 Hz and orders of magnitude higher. Meanwhile gravitational waves propagate through spacetime with very little disruption and represent frequency spectra from 10^4 Hz and downwards [11]. Because of these fundamental differences, these two sources of cosmic information grant us astoundingly different lenses through which we can view the universe, with the common analogy being that the EM and gravitational wave spectra work together to paint a full picture of the universe in the same way that audio and video work together to bring together a cohesive television experience.

1.2 Compact Binary Coalescence as a Source of Gravitational Waves

The most promising gravitational wave sources are compact binary coalescences (CBCs) [12]. Of the two detections announced thus far by the LSC, both have been CBC events of inspiralling binary black holes [13], [14]. CBC events are the last few minutes of the overall process by which massive co-orbiting systems, such as binary black holes or binary neutron stars, bleed energy in the form of gravitational waves as they inspiral towards each other and merge into a single body system [15], [16]. As the two bodies are drawn closer together, their orbital frequency begins to dramatically increase in a runaway process. The last few moments of a CBC are characterized by a "chirp" as the frequency of orbit sweeps from ~ 10 Hz to ~ 1000 Hz [12]. The waveforms of the gravitational radiation emitted by the binary system during this process can be represented mathematically as a linear combination of the plus and cross polarizations depicted in Figure 1.1. The precise quadrupole waveforms of the polarizations are given by the equation [17]

$$h_+ \equiv 2\frac{\mathcal{M}}{d_L}(1 + cos^2 i)(\pi \mathcal{M} f)^{2/3} \cos(\Phi + \Psi) \tag{1.2}$$

$$h_\times \equiv 4\frac{\mathcal{M}}{d_L} cosi(\pi \mathcal{M} f)^{2/3} \sin(\Phi + \Psi) \tag{1.3}$$

where

$$\cos(i) \equiv n_I \cdot e_x^S \tag{1.4}$$

$$\mathcal{M} \equiv \text{Chirp mass} = (1 + z)\mu^{3/5} M^{2/5} \tag{1.5}$$

$$\Phi \equiv -2\Big(\frac{T - t}{5\mathcal{M}}\Big)^{5/8} \tag{1.6}$$

$$f \equiv \frac{1}{2\pi}\frac{\partial \Phi}{\partial t}. \tag{1.7}$$

d_L is the luminosity distance to the binary and i is the inclination angle of the binary plane of orbit with respect to the line of sight with the interferometer. An example of a waveform is included below.

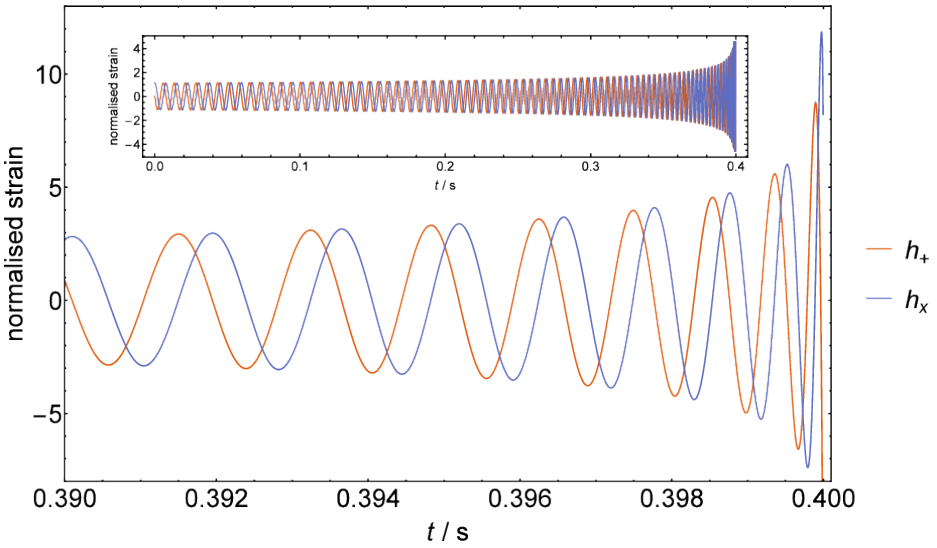

Figure 1.2: A visualization of the normalized plus and cross polarizations of a gravitational wave of $\mathcal{M} = 2.2M_{\odot}$ during the final 0.01 s of inspiral. The smaller graphic shows the final 0.4 s before coalescence. The "chirp" is especially prominent here. [18]

The well-defined waveforms of CBCs make them ideal candidates to be the most prominent sources of gravitational waves during the early days of gravitational wave astronomy [19]. In the case of the known sinusoidal functions such as those of CBC waveforms, it is well documented that matched filtering is the most optimal method of detection [20]. The basic tenets of matched filtering and its application to gravitational wave detection will be outlined in the next section.

1.3 Matched Filtering and Template Banks

Speaking generally, the matched filtering approach to signal processing involves the calculation of a "match" between collected data and a series of theoretically calculated templates. If the match between the signal and a template exceeds a user-defined threshold, the signal is registered as a successful detection. In the case of gravitational waves, this refers to the comparison of data coming from the LIGO detectors and the CBC waveforms calculated using general relativity.

At this point it is necessary to outline the mathematical motivations for the match between signal and theory. This begins with the definition of an inner product between two abstract functions $h(f)$, which are functions of frequency

$$\langle h_1|h_2\rangle = 2\int_{f_{low}}^{\infty} \frac{\tilde{h_1}(f)\tilde{h_2^*}(f) + \tilde{h_1^*}(f)\tilde{h_2}(f)}{S_n(f)} df \tag{1.8}$$

where $S_n(f)$ is the non-constant detector noise spectrum which varies as a function of frequency. With this definition of inner product, we can establish an equation for the match $M(\lambda, \Delta\lambda)$ between

two gravitational wave templates $u(f;\mu_k,\lambda_k)$ and $u(f;\mu_k+\Delta\mu_k,\lambda_k+\Delta\lambda_k)$, where $k=1,2,3...$, and μ and λ are extrinsic and intrinsic parameters of the templates respectively. Intrinsic parameters refer to the "dynamical" parameters which dictate the shape of the waveforms that are of interest to this project. Extrinsic parameters refer to "kinematical" parameters which affect the offset of waveforms in the parameter space and primarily affect the amplitude of the waveform. Examples of intrinsic parameters are mass and spin of the binary. Examples of intrinsic parameters are the time of coalescence and waveform phase at coalescence. Fast Fourier transforms can be used to quickly compute inner products between the two templates which quickly explore the full range of extrinsic parameters μ for a fixed λ_k. We are interested primarily in the intrinsic variables because, as was previously stated, most extrinsic parameters only affect waveform amplitude and are thus of little consequence to the inner product calculation. The few extrinsic parameters which do affect the inner product and thus, as we will soon see, the match between two signals are simply maximized over. We define match as the value of the inner product between the two values maximized across the extrinsic parameters.

$$M(\lambda,\Delta\lambda)\equiv\max_{\mu,\Delta\mu}\langle u(\mu,\lambda)|u(\mu+\Delta\mu,\lambda+\Delta\lambda)\rangle \tag{1.9}$$

where M is normalized to have a maximum value of 1 when $\Delta\lambda=0$. With the match between two signals now properly defined, we have the rudimentary tools needed to begin data searches for gravitational waves [21].

Of course, a logical next step would be to determine a proper set of theoretical templates against which we could compare our data and compute matches, which we call a template bank. Since the intrinsic parameters are not known *a priori*, data must be compared against a wide range of templates with dynamic parameters across as many intrinsic variables as possible. Unfortunately the construction of a template bank is not as simple as calculating as many templates as possible across the parameter space. While such "overcoverage" would guarantee that a signal processed through the template bank would be detected, the computational costs associated with the match calculation against every template would be overwhelming. Conversely, if a template bank is too sparse it could fail to make a detection, an arguably worse outcome than increased computational costs.

It thus becomes clear that there is an ideal bank size which optimizes the computational efficiency of the search while maintaining sufficient coverage throughout the entire parameter space. In order to quantify this ideal template density, we must first understand our parameter space a little bit more. We begin by expanding M around $\Delta\lambda=0$. We use the resulting power series to obtain

$$M(\lambda,\Delta\lambda)\approx 1+\frac{1}{2}\left(\frac{\partial^2 M}{\partial\Delta\lambda^i\partial\Delta\lambda^j}\right)_{\Delta\lambda^k=0}\Delta\lambda^i\Delta\lambda^j \tag{1.10}$$

where the second term resembles a metric which we define as

$$g_{ij}=-\frac{1}{2}\left(\frac{\partial^2 M}{\partial\Delta\lambda^i\partial\Delta\lambda^j}\right)_{\Delta\lambda^k=0} \tag{1.11}$$

With this metric we come across the useful revelation that the mismatch $MM = 1-M$ between two templates is simply the proper distance between them squared. We can consider the intrinsic parameter space of the template bank to be an N-dimensional lattice with a unit cell of side length dl. If the templates are closely spaced such that $dl \ll 1$, we come to the following conclusion

$$MM = 1 - g_{ij}\Delta\lambda^i\Delta\lambda^j = 1 - N\left(\frac{dl}{2}\right)^2. \tag{1.12}$$

The mismatch MM is determined according to the judgement of the experimentalist, who chooses it based off of their desired quality and thoroughness of the search. The ideal number of templates present in the bank is determined to be

$$\mathcal{N} = \frac{\int d^N\lambda\sqrt{det||g_{ij}||}}{\left(2\sqrt{1-MM/N}\right)^N}. \tag{1.13}$$

In this idealized bank, templates reside at each corner of the unit cells.

While elegant, the Owen geometric approach has two major flaws which, at the moment, hinder our ability to use it outright as a template placement method in current template banks. First, it relies on the ability to take the derivative of the match analytically. Second, it assumes that we are working with a coordinate system that assumes a globally constant metric. Even today we do not possess the ability to rectify either of these issues, seemingly rendering the Owen approach unfeasible. Thus the development of an alternative template placement method was necessary.

1.4 sbank

The tool currently used by the LIGO Scientific Collaboration to generate template banks for CBC waveforms is sbank [22], where the 's' stands for stochastic. As its name suggests, sbank fundamentally works through a random tiling method, proposing templates throughout the mass and spin parameter space and rejecting proposals which are "too close", a template tiling method first proposed by Harry, Allen, and Sathyaprakash [23]. Of course, there are more nuances to the process than simply throwing around templates with reckless abandon. The appropriate density of templates across regions of the parameter space will vary depending on the metric "local" to those regions. From this issue follows the idea that tile placement is best done in a coordinate system over which the metric is slowly varying. In the case of sbank, this was determined to be dimensionless chirp time coordinates $\theta_0, \theta_3, \theta_{3S}$ [22] where the transformations into this coordinate system are

$$\mathcal{M} = \frac{1}{16\pi f_0}\left(\frac{125}{2\theta_0^3}\right)^{1/5} \tag{1.14}$$

$$\eta = \left(\frac{16\pi^5}{25}\frac{\theta_0^2}{\theta_3^5}\right)^{1/3} \tag{1.15}$$

$$\chi = \frac{48\pi\theta_{3S}}{113\theta_3} \tag{1.16}$$

where $\mathcal{M}$ is chirp mass defined in Equation 1.5, η is the symmetric mass ratio, and χ is the reduced spin parameter. The advantage of this coordinate system is that waveforms in the dimensionless chirp time space have leading order terms which are almost linear, resulting in a more constant metric across the new parameter space. It should be noted that sbank simply passively takes advantage of this property of dimensionless chirp time space and does not use the metric itself in the template placement process.

"Closeness" of templates in the sbank's stochastic placement method is determined by computing the match between a proposed template and the templates which have been accepted by the bank up to that point. If the maximum value of these matches is below a minimum match threshold defined by the user, then the proposal is accepted. This method is incredibly robust as it ensures that the template bank is proofed against "holes" where potential signals make it through the matched filtering process undetected. However, for large parameter spaces the stochastic tiling process can also become incredibly computationally expensive, as each accepted template will require in a worst case scenario $\mathcal{O}(N^2K)$ match calculations, where N is the number of accepted templates and K is a number based off the minimum number of calculations deemed acceptable to determine that the bank space has been sufficiently probed. For large values of N and K, the last few templates in the bank can be quite computationally demanding.

While it can not totally circumvent this scaling issue, sbank does implement some tricks to reduce the number of match calculations it has to perform, which are namely as follows

- Each proposed template is only checked against "neighboring" accepted tiles. The conditions for determining whether or not templates are neighbors is based upon the ratio of the θ_0 values. Below a certain θ_0 fraction the templates can be assumed to be far away that their match will be low.
- Instead of blindly comparing the proposal against the entire accepted bank, it is logical to cut off the process and immediately reject a template if it exceeds the minimum match threshold. In order to facilitate this condition, templates are compared starting with those of the greatest θ_0 values and working towards the farther templates.
- Proposals are made uniformly in the dimensionless chirp time parameter space in order to avoid the possibility of overchecking regions of the parameter space which are already densely populated, an issue which could arise from pure stochastic sampling.
- If the metric of a region is known, it can be used to approximate the match much more simply than the relatively expensive inner product calculations.

With these computational techniques the number of required match operations per proposal is reduced to $\mathcal{O}(NMK)$, where $M < N$.

The sbank method is a robust, dependable algorithm which has produced high quality template banks for the LSC. However, it is far from perfect. While sbank does make use of some clever strategies to make significant cuts to its computational requirements, it can not escape the computational costs which runaway as the size of the bank grows bigger. While the stochastic

technique can be used in theory to create banks of any size and dimensionality, these computational costs limit the usefulness of sbank in the dimensionality of the banks it can produce. While a 9-dimensional bank has been created using sbank, its production was incredibly slow [24]. In addition, there is a certain lack of elegance to stochastic placement, which, at its core, is just a brute force algorithm. In order to more effectively and efficiently search for gravitational waves originating from CBCs across a wider spectrum of intrinsic parameters, we require a new method which will make the creation of large, high-dimensioned template banks computationally feasible. We present a new algorithm, dubbed 'treebank', which designs template banks using a geometric approach and will improve upon sbank's bank creation efficiency by several orders of magnitude.

Chapter 2

treebank: Background and Methods

2.1 Basic Tenets and Algorithm

The treebank algorithm is a comprehensive method to compute template banks comparable to those produced by sbank, but with phenomenally reduced computational requirements and runtimes. It fundamentally works by dividing the parameter space into a set of hypercubes via a binary tree decomposition, where the splitting condition is is determined by the metric as calculated at the center of each hypercube. This is, in a sense, a return to the older days of template placement under the geometric schemes envisioned by Ben Owen which revolved around the computation and use of the metric to tile template banks. Our algorithm is laid out in detail here

1. Calculate the metric at the center of a hypercube. Note that the term hypercube is used here to generically refer to a partition of the bank parameter space. This term can refer to any dimensionality, including those of 3 dimensions and lower.

2. Use the metric to decide whether or not to split the hypercube. If it is determined that it should split, bifurcate the hypercube along its longest bisector, forming two "new" hypercubes. Otherwise, leave it alone.

3. Reiterate this process throughout the parameter space and each hypercube until all hypercubes no longer require splitting.

4. Place templates according to one of three possible tiling methods: geometric placement, stochastic, or single template placements.

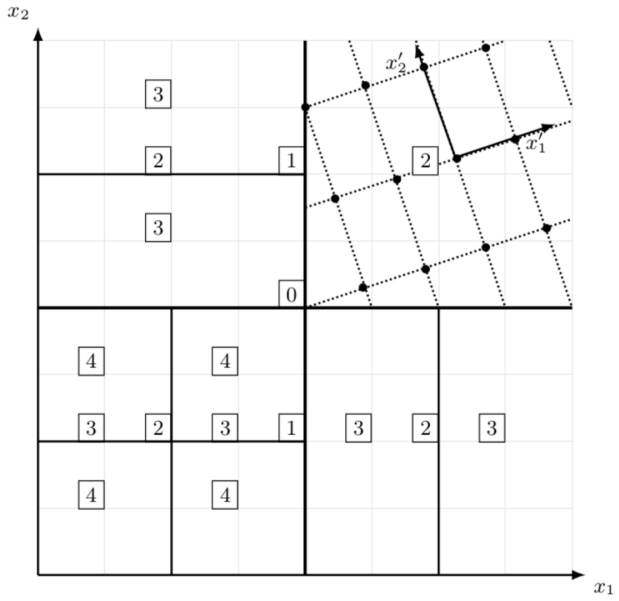

Figure 2.1: Visualization of the splitting process for a generic $\{x_1, x_2\}$ coordinate system. The centers of hypercubes are numbered, where each number represents a "layer" of splitting.

The upper right hypercube in Figure 2.1 represents a transformed $\{x_1', x_2'\}$ coordinate system. This transformation is used in the geometric tiling scheme and will be expounded upon in a later

section.

This is the overall method which drives treebank. Its computational advantage over sbank comes from the division of the parameter space into hypercube over which the metric is roughly constant, which allows for relatively simple template placement throughout each hypercube. The metric, which is evaluated as a numerical derivative of the mismatch, is computationally cheap compared to explicit match inner product calculations which drive the bulk of sbank's computation. The metric calculations used by treebank are as follows

$$g_{ii} = \frac{\delta_{ii}^2}{(\Delta x_i)^2} \tag{2.1}$$

$$g_{ij} = g_{ji} = \frac{\delta_{ij}^2 - g_{ii}(\Delta x_i)^2 - g_{jj}(\Delta x_j)^2}{2\Delta x_i \Delta x_j} \tag{2.2}$$

where δ^2 is the mismatch and the Δx are intrinsic parameters.

2.2 The Splitting Process

A hypercube splits if it meets one of the following criteria

1. It exceeds a target template density based upon Equation 1.13
2. The difference between the metric calculated at opposite boundaries of the hypercube exceeds a target threshold

Both of these criteria are related the mismatch deemed acceptable by the user. The binary tree decomposition method works well as it preserves a number of useful features of the parameter space. We can rest assured that the entire parameter space is accounted for without having to manually check for match overlap throughout regions where the metric has varied. Additionally, this method automatically splits the parameter space into regions such that the template density is roughly appropriate across all regions of the template bank, which acts as a built in check against overcoverage.

The motivation behind the splitting process is to divide the parameter space into regions over which the metric varies slowly enough that it can be approximated as constant. Over regions with a relatively "flat" metric we can employ more elegant tiling methods which don't rely on the brute-force match calculations used by sbank. There are currently three template placement methods which demonstrate promise for treebank: geometric placement, stochastic placement, and single template placement. While only one of these will be implemented into the final version of treebank, all three are described in this thesis in order to grant a clearer picture of the intricacies of this project, as well as to enforce the thoroughness of the explorations that went into determining the most optimal bank creation methods. I begin with the geometric method.

2.3 Geometric Tile Placement

Of the proposed template placement methods, the geometric approach is perhaps the most natural method to arrive at. The fundamental principle is simple: we discern a transformation matrix such that the coordinate system of the hypercube we wish to tile has a transformed metric that is the identity matrix. Mathematically this is described as

$$g' = \mathbf{M}g = \mathbf{I} \tag{2.3}$$

where $\mathbf{M}$ is a to-be-determined transformation. The mismatch δ^2 should be invariant across all different coordinate systems, so that the following must be true

$$\begin{aligned} \delta^2 &= \overrightarrow{\Delta x}^T g \overrightarrow{\Delta x} \\ &= \overrightarrow{\Delta x'}^T g' \overrightarrow{\Delta x'} \end{aligned} \tag{2.4}$$

where $\overrightarrow{\Delta x'}$ represents parameter vector in the transformed space where, again, g' is the identity matrix. We can take advantage of the fact that the metric is symmetric and positive definite in order to use a Cholesky decomposition

$$g = \mathbf{M}^T\mathbf{M} \tag{2.5}$$

to determine a unique solution for $\mathbf{M}$.

Recalling from Equation 1.12 that the ideal unit cell length dl of the lattice of the intrinsic parameter space is determined solely by δ^2 and the dimensionality of the bank, we realize that dl is also invariant across transformations into different coordinate systems. Armed with the knowledge and the fact that the transformed metric is the identity matrix, it becomes clear that template placement within a transformed coordinate system is simply a task of laying down tiles along the eigenvectors of the transformed space in spacings of dl. In order to ensure that the original hypercube is sufficiently tiled, a "bounding box" is drawn around the hypercube in the transformed coordinate system, such that the entire volume of the hypercube is encapsulated within the bounding box.

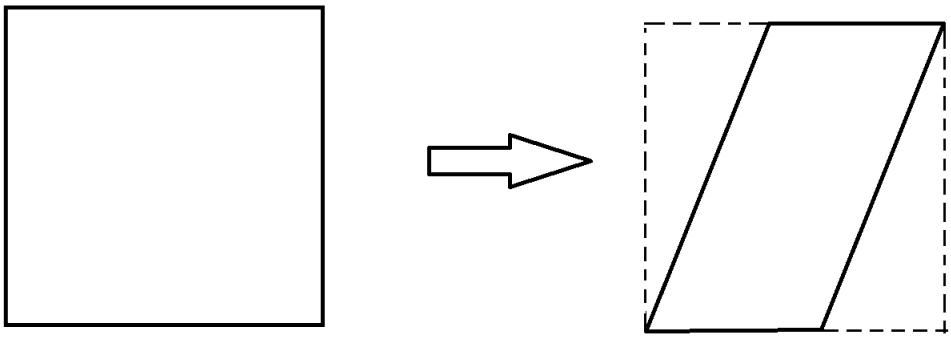

Figure 2.2: 2D visualization of the transformation of a hypercube from generic x coordinates to x' coordinates. The dotted line around the transformed hypercube represents the bounding box. Templates are placed evenly throughout the entire bounding box and removed after transforming back the original coordinates if they do not fall within the confines of the hypercube.

Once this process is complete, it is a simple matter of transforming the templates back to the original coordinates of the parameter space and removing any which do not reside within the hypercube. Repeating this process for every hypercube created by the binary tree decomposition leaves us with an elegant template bank tailored to the desired δ^2 mismatch.

While simple and refined in theory, the geometric approach did encounter issues when actually implemented into the treebank code. There is no guarantee that the hypercube will not undergo drastic change in the transformed space. If the transformed hypercube were to be heavily elongated in any particular direction, then the bounding box drawn around it would have to be very large as well. The result is a massive bounding box which contains a hypercube whose volume is much smaller than that of the bounding box. When the bounding box is tiled, the number of templates being processed is far, far greater than what will ultimately end up within the hypercube. During tests of the geometric tiling approach it was determined that many if not most hypercubes suffer from this issue.

While the added computational burden of these extraneous templates is not appreciable in 2D banks, additional parameter dimensions cause the computational requirements of the program to increase by several orders of magnitude, threatening to quickly break treebank's ability to scale into higher dimensions. Several solutions to this issue were proposed, however all methods were found to generalize back to the bounding box approach. As such, until a suitable technique to deal with the tiling of the hypercube in the transformed space is found, the geometric approach is impractical to use, despite its theoretical elegance.

2.4 Stochastic Tile Placement

The stochastic placement method is a hybrid approach which utilizes the random tiling techniques associated with sbank in combination with a locally constant metric across each hyper rectangle.

From Equation 1.13 we can calculate the required template density $\mathcal{N}$ for a certain level of mismatch, assuming that we have a metric which describes the parameter space. Of course, the binary splitting of the bank provides us with exactly that for each hypercube. Using the target $\mathcal{N}$ as a limit on the number of templates to be placed within a given hypercube, we propose tiles at random coordinates in the target region. If the proposed tile has at least a separation of dl from all other tiles in the hypercube it is accepted and tiled.

In exploring the stochastic placement method, we discovered that care must be given to the tiling along the boundaries between hypercubes. While templates within each individual cube are guaranteed to be spaced appropriately, it is possible the templates placed at the boundaries of each hypercube are closer than a separation of dl. The failure of the initial stochastic algorithm to address this issue with the discrete regions of the template space results in a severe bank overcoverage issue.

We devised a solution to this boundary issue by allowing each hypercube to keep track of its neighboring cube, where a neighbor is defined as a hypercube which shares a boundary or a vertex with the cube we are examining.

Because of the nature of the binary tree decomposition, in which all hypercubes interlock with no overlap to compose exactly the entire parameter space, we are able to set the condition that a two hypercubes are neighbors if the vertices of one cube overlap with the boundaries of another at any point.

However, without additional conditions to follow, we would still have check every hypercube against every other to determine whether or not they were neighbors, which, for larger banks, adds an undesirable computational cost. After some investigations into the properties of binary tree decompositions of geometric spaces, the following rules concerning neighbors

- Define a parent as a hypercube and its children as the two nodes it becomes after splitting. Each of those children nodes is a sibling to the other. Each child automatically picks up its sibling as a neighbor. In addition, a child can ”inherit” none, some, or all of its parent’s neighbors, but no more.

While these rules may not seem immediately very useful, they have powerful implications when it comes to reducing the complexity of determining neighbors. Instead of having to check a hypercube against every other node in the parameter space, only the neighbors of its parents must be checked. Obviously in the case of the initial, parentless node there is no need to check for neighbors.

After establishing the neighbors of a hypercube, proposed templates can be checked against templates in neighbors, thus ensuring that tiles are not placed close to templates across hypercube boundaries. Since the neighbors are determined during the splitting process, before the tiling, there is no worry that any neighbors can be unaccounted for. Investigations determined that the implementation of the neighbors tracking did indeed help greatly in suppressing the overoverage issue of the stochastic placement method.

A second technique used to address overcoverage in the stochastic method involved splitting and tiling in a new coordinate system, much like sbank and its use of dimensionless chirp time coordinates. Working in directly with basic mass coordinates forces us to work in a parameter space where the metric is highly variable, leaving the highly metric dependent methods of treebank vulnerable to miscalculation and errors. Some transformed coordinate systems feature more slowly changing metrics, which would allow us to make better approximations of local regions where the metric is roughly constant. For the purposes of improving the stochastic placement method, we explored 2D template bank creation using coordinates of mass ratio and total mass as opposed to the masses of the elements of the binary system. The definition of mass ratio Q and total mass M_T are simple and are as follows

$$M_T = m_1 + m_2 \tag{2.6}$$

$$Q = \frac{m_1}{m_2} \tag{2.7}$$

where m_1 and m_2 are the masses of the binary components. m_1 is defined to be less than m_2. We essentially transform the coordinates of the parameter space before beginning any actual calculations, and split and tile in the $\{M_T, Q\}$ space, and the transforming everything back to the original coordinates.

Although robust due to its simplicity, the stochastic method still has its fair share of issues, namely that, even with neighbors tracking and a change in coordinates implemented, it bears a much greater computational cost than the geometric method. This is to be expected of course, as it abandons the elegant, simple algorithm of geometric placement in favor of a tactic which ultimately boils down to an almost brute force approach. In a sense, the stochastic method is at its core the same as sbank, just relying on a different set of tricks to increase the computational efficiency of the template bank generation. However, since the stochastic method relies on approximations to the metric in order to determine template density, it fails to tile the parameter space as properly as sbank, which computes each of its tile precisely using an exact match calculation. As a result, running treebank using stochastic placement has been found to consistently overpopulate template banks compared to sbank.

2.5 Single Template Tile Placement

The single template method is the most recently devised of the three tiling processes. It attempts to circumvent the issues that arise with the template placement of the other methods by

altering the splitting conditions of the banks so that each node of the binary tree decomposition houses only a single template, which is placed at the center of the hypercube.

This is accomplished by setting the condition for splitting to occur if $\mathcal{N} > 1$. The single template method derives its power from the fact that it reduces the complexity of the bank generation process even further, requiring only the metric calculation and splitting process to be correct, as tiling a single template per hypercube is trivial.

Unfortunately for the sake of this thesis, at the time of writing the single template method was derived and implemented only in the past two weeks, so testing concerning its robustness has not been as thorough as the other tiling methods. Fortunately, though, it does seem to require as rigorous examination as the other methods to begin with, as it has vastly outperformed the other methods almost immediately from the outset in all areas of interest, including runtime, bank size, scaling to higher dimensions, and performance in bank simulation injection tests.

Due to its great success, the single template tiling method will, barring any unforeseen issues with it, be implemented in the final build of treebank. That being said, it still possesses some minor issues that have to be worked out. Namely there seem to be very specific cases where the metric approximation detailed in Equations 2.1 and 2.2 is inaccurate. As a consequence, there are some small holes in the single template bank where simulated gravitational waveform injections fail to be detected. However, these issues are minor and are expected to be fixed in the very near future.

As a whole, single template tiling shows an incredibly amount of promise. It is slightly unusual as it forgoes the use of "proper" techniques such as utilizing dl to ensure that regions are sufficiently tiled. However, this is perhaps the most interesting aspect of the template placement method, as it reveal to us that CBC template bank theory developed and relied upon in the past is indeed suboptimal technique which is due for an update.

Chapter 3

treebank: Performance, Analysis, and Results

3.1 Quantifying Performance

Before delving into the successes and weaknesses of treebank, it is necessary to outline criteria by which we can explicitly evaluate treebank against current technology, e.g. sbank. We establish three major metrics to this end

1. Runtime - The amount of time it takes to create a template bank is our best means by which to determine the computational stress treebank places upon a computer. Of particular interest is the scaling of treebank as it generates template banks of higher intrinsic parameter dimensionality.

2. Bank size/Overcoverage - As was previously stated, template banks can not be either too small or too large, lest they have holes or take too long to run a matched filtering search through respectively.

3. Banksim Performance - Bank simulations (banksims) are a tool used to judge the quality of a template bank. It simulates gravitational waveform signals which are fed through the template bank. If the bank fails to recover the waveform injection, then it is noted that there is a hole at the coordinates of the lost injection.

Naturally, sbank is used as the standard against which these criteria are compared. If treebank is ultimately unable to perform as well as current tools, then there is no reason to switch to a subpar bank generator.

In addition to measuring treebank's performance, we also take some time to explore the behavior of the metric calculation across the parameter space, as understanding the accuracy of the metric is integral to all processes of treebank.

3.2 Runtime Analysis

Runtime is a fairly straightforward criterion to determine, as it involves simply timing a process from beginning to completion. There are two major measurements for runtime: real time and CPU time.

Real time is the amount of time that passes "in real life" from the moment a command is run to when it is fully executed. CPU time is a normalized measurement of how long it would take a program to run if it is given one hundred percent of a CPU's computing power. Both have merit when being considered as measurements of resources required by the program.

CPU time can be either greater than or less than the real time. It will be less than the real time if the process spends a large amount of time idle or waiting. With multicore computers it can become greater as time spent running calculations by each individual core are all added together. For instance, a program on a 4-core machine which runs for one minute but splits its computational load so that each core is fully preoccupied for the duration of the minute will have a runtime of

one real minute, but a CPU time of four CPU minutes.

Since CPU time is a more direct measurement of the computational cost of a program, it is the efficiency measurement that we will be focusing on in this analysis. We quantify the CPU runtime of the creation of a template bank across several different parameter boundaries. The largest set of parameters is based off of the CBC matched filtering search conducted during the O1 run [25].

Mass Boundaries	**Spin Boundaries**	**sbank CPU Time**	**treebank CPU Time**
$(2.0, 3.0)$	$(0.0, 0.0)$	14 minutes	4 seconds
$(2.0, 12.0)$	$(0.0, 0.0)$	105 minutes	25 seconds
$(7.0, 12.0)$	$(0.0, 0.0)$	22 minutes	2 seconds
$(7.0, 12.0)$	$(0, 0.95)$	38 minutes	4 seconds
$(2.0, 99.0)$	$(0, 0.98)$	100 hours	1 hours

Table 3.1: Runtimes required across five runs for both sbank and treebank. All masses are in solar masses and all times are in CPU time.

It is clear that treebank achieves its goal of improved computational efficiency with flying colors. Direct comparison between sbank and treebank reveals that the binary decomposition method is consistently in the regime of 1×10^2 times faster than the stochastic placement method.

In addition to experimental data, we can also perform a rough analysis of the time complexity of the number of match calculations treebank requires per waveform template in a bank.

We first calculate the number of metric evaluations as

$$\text{number of metric evaluations} = \sum_{n=1}^{n} 2^n = 2^{n+1} - 1 \tag{3.1}$$

where n is the number of splits the bank has undergone. Each metric evaluation requires $N(N+1)/2$ match calculations where N is the dimensionality of the bank. Thus the total number of match calculations required in a bank is

$$\text{number of match evaluations} = \frac{(2^{n+1}-1)N(N+1)}{2} = \mathcal{O}(2^n N^2). \tag{3.2}$$

Generically speaking without taking a specific tiling method into account, each hypercube contains roughly N_V^* templates, based on the accuracy of the metric approximation of the hypercube. The template bank will contain approximately $2^n N_V^*$, thus the number of match calculations required per template is

$$\frac{\text{number of match evaluations}}{\text{number of templates}} = \mathcal{O}(N^2/N_V^*) = \mathcal{O}(1) \tag{3.3}$$

where $N_V^* \sim N^2$. In the case of the single template placement method, where $N_V^* = 1$, the number of match calculations required is instead $\mathcal{O}(N^2)$. Since N will be a relatively small constant, both

time complexity estimations are very promising in terms of their implications on the scaling of the number of match calculations required based on the size of the bank.

As a whole, these results are incredibly promising. The massive reduction in runtime is clear evidence that the fundamental methodology behind treebank and its differential geometry approach using the metric to bank splitting and tiling is solid and worth pursuing.

3.3 Bank Size Analysis

As was previously stated, there is a sweet spot for the number of templates populating a bank. We compute an expected number of templates based on the ratio of the volume of an N-dimensional hypercube against the volume of a single template, or in other words the number of non-overlapping templates which will fit within a hypercube

$$\text{expected number of templates} = \frac{V_{hypercube}}{V_{template}} \tag{3.4}$$

where

$$V_{template} = \frac{(\pi\delta^2)^{N/2}}{\Gamma(N/2+1)}. \tag{3.5}$$

$\Gamma(N/2+1)$ is the gamma factorial function.

Naturally, we also use sbank's template densities as a benchmark against which to compare treebank. The results of bank sizes across five template banks are compared here between sbank and treebank

Mass Boundaries	**Spin Boundaries**	**Expected**	**sbank**	**treebank**
$(2.0, 3.0)$	$(0.0, 0.0)$	193	640	394
$(2.0, 12.0)$	$(0.0, 0.0)$	2403	4598	3909
$(7.0, 12.0)$	$(0.0, 0.0)$	16	102	39
$(7.0, 12.0)$	$(0, 0.95)$	6	974	39
$(2.0, 99.0)$	$(0, 0.98)$	~ 290000	~ 300000	~ 700000

Table 3.2: Template bank sizes across five parameter boundaries.

Here our numbers are more perplexing. In comparison to sbank, treebank, for the most part, tends to be undercovering. Compared to the expected number of templates, however, treebank is always placing too many templates. Of course whether this is ultimately of any detriment depends on the time taken to run a banksim and the results of the banksim, which will be explored in the next section.

That being said, the disparity between treebank and both the expected number of templates and sbank is indicative of some underlying issue. The most likely candidates for this lie in either the splitting conditions or the metric calculation. To this end fine tuning of the splitting conditions in conjunction with reviews of both the code and the literature upon which the mathematics is based will be necessary to optimize our bank sizes.

3.4 Banksim Analysis

The bank simulations used to test template banks pass a large number of injection templates through each bank, usually ranging from 1000 to 10000 templates based on the thoroughness with which the bank is being checked. For the sake of succinctness, only a single banksim analysis will be included in this section. The appendix may be referenced for plots detailing the performance of banksims across other parameter boundaries.

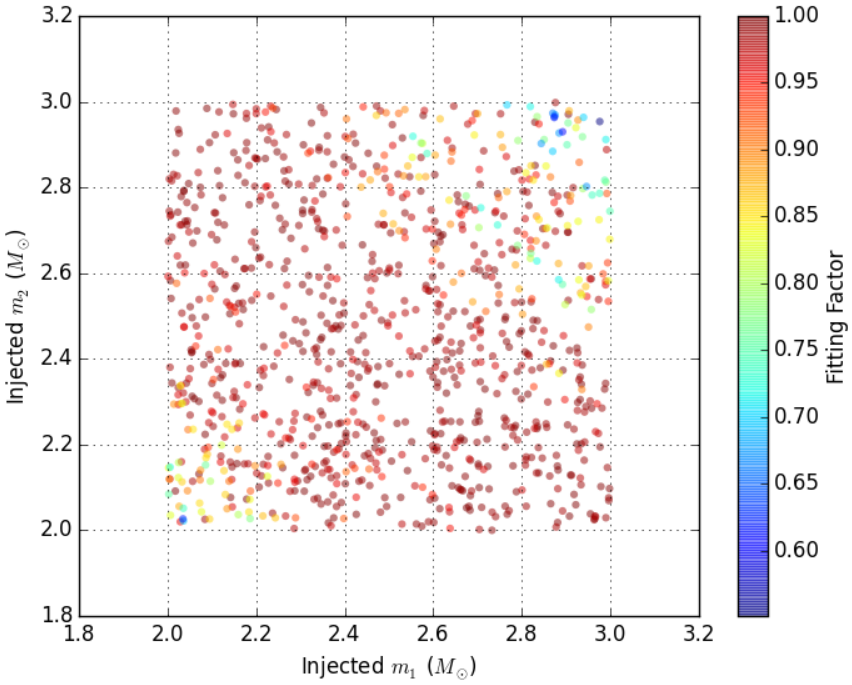

Figure 3.1: Banksim of a non-spinning $(2, 3)M_\odot$ bank generated by treebank. The bank was generated with the goal of having a 0.97 minimum match, or 0.03 mismatch. The axis are the mass coordinates of the injections. The color map displays the success with which the injection was detected in terms of the maximal match determined between an injection and the template bank.

Our banksims primarily reveal that treebank currently struggles with undercoverage. In particular the corners of the bank space tend to display the most prominent holes. Considering what we've seen from the bank size comparisons in the previous section, these results are not entirely surprising. The stochastic placement method and sbank are by no means optimal, but their great

strength is their robustness. Since treebank's bank sizes fall so short of sbank's, the undercoverage problem is logical. An improvement in banksim performance should follow once the bank size issues are fixed.

3.5 Metric Analysis

As has hopefully been made clear at this point, the reliable approximation of the metric forms the foundation upon which treebank lies. It is therefore worth taking a bit of time to gain an idea of the accuracy of the metric and how to changes throughout the parameter space. We employ a random sampling method to check the metric calculation of a series of single hypercubes. The metric which defines the hypercube is computed at the center of the cube. The fractional difference between the explicit match and the metric match is then calculated around the immediate vicinity of the hypercube center. Heatmaps of the process is depicted in the figures below

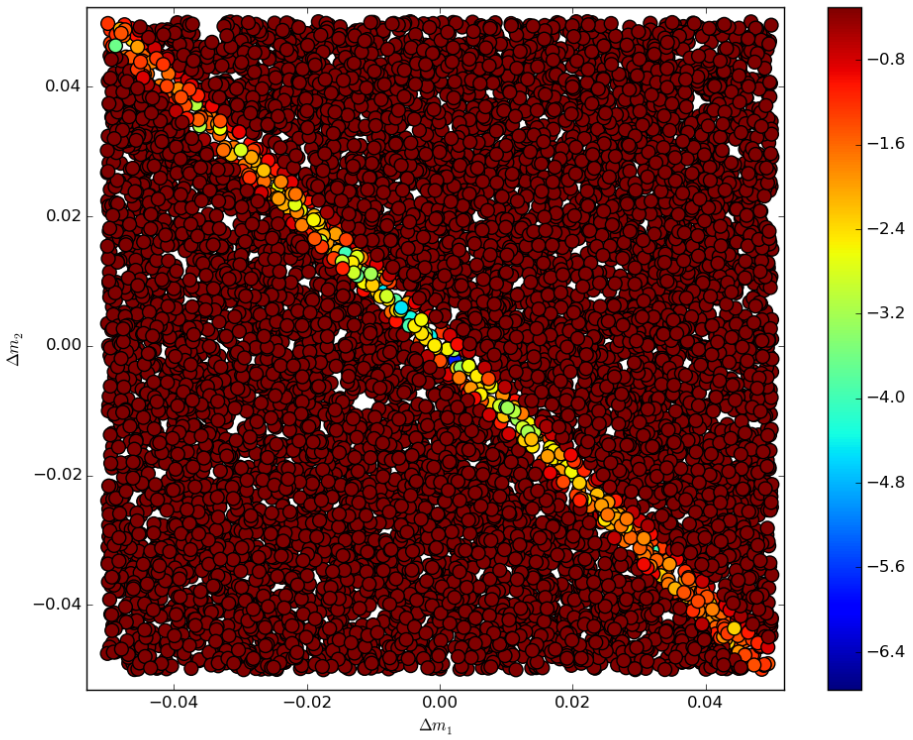

Figure 3.2: Heatmap of the difference between the exact match calculation and the metric match approximation for a hypercube of mass boundaries $(1, 2)M_\odot$ and spin boundaries $(-0.985, 0.985)$. The axis are small deviations in the mass parameter vector. Note that the metric itself is being calculated at the center of the hypercube at $(0.5, 0.5)M_\odot$ and spin $(0, 0)$. The color scale describes the fractional difference between the exact match and metric match according to the natural log of the difference.

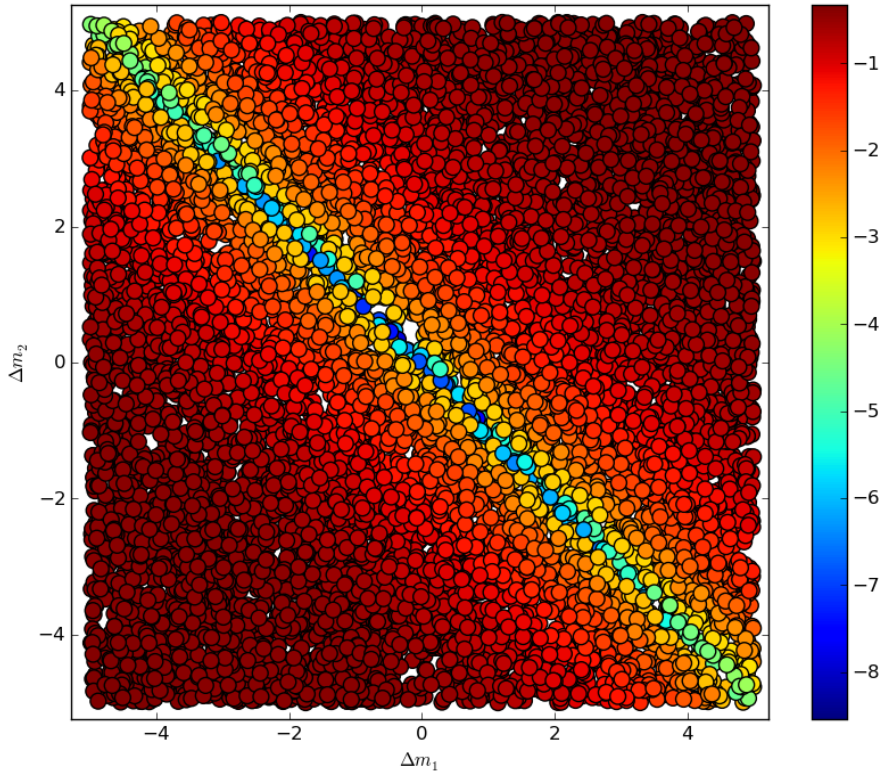

Figure 3.3: Heatmap of the difference between the exact match calculation and the metric match approximation for a hypercube of mass boundaries $(2, 99)M_\odot$ and spin boundaries $(-0.985, 0.985)$ for both binary elements. The metric is being evaluated at the center of the hypercube at $(50.5, 50.5)M_\odot$ and spin $(0, 0)$. The match comparisons are sampled throughout the mass space.

Additional heatmaps can be found in the Appendix. For now we turn our attention to two important differences between Figure 3.1 and Figure 3.2: the scale and coverage of the metric.

The change in scale is dramatic. The order of magnitude of the smallest difference in metric of the $1 - 2M_\odot$ bank is larger than that of the $2 - 99M_\odot$ bank by about 1.5 orders of magnitude. Looking at both graphs objectively, we can see that far from their "effective diagonal", both parameter spaces drop off such that the fractional difference is off by more than a 10-percent fractional change, which is less than ideal. However, the effective diagonal of the higher parameter space is much thicker, indicating a greater effectiveness and an altogether "flatter" space. While the lower parameter regions do address this by tiling more densely, their tighter effective diagonals inherently make them more prone to gaps in the coverage of the metric. Together, these observations indicate an issue with the metric calculation in lower mass regions.

These results are not entirely surprising, as we expect the metric to behave with more volatility in the lower mass regions than the higher mass ones. The intuition here is that in lower mass regions deviations in the parameter space are proportionally large variations compared to the same

deviations in the larger mass space. After all, the difference between a $2.1M_{\odot}$ binary compared to a $2.0M_{\odot}$ is markedly more disparate than a $99.1M_{\odot}$ and a $99.0M_{\odot}$ binary.

Chapter 4

treebank: Conclusions and Outlook

4.1 Conclusion

The treebank template generation algorithm is a promising work in matched filtering signal processing of gravitational waves originating from compact binary coalescences. It employs clever uses of differential geometry and computer science to speed up template bank creation compared to the current standard, sbank.

The fundamental process revolves around the numerical calculation of the metric of a section of an N-dimensional bank parameter space and splitting the space into a series of N-dimensional hypercubes based on properties of that metric through a binary tree decomposition. Once the space has been partitioned, individual hypercubes are tiled with waveform templates. For the foreseeable future, treebank will utilize a tiling scheme where hypercubes are split until only a single template placed at their centers is necessary to provide adequate coverage for the hypercube volume. Other current possibilities include a geometric tiling approach more in line with the original template spacing works of Ben Owen, or a stochastic approach similar to sbank.

The treebank method shows incredible promise as a tool which will expand the ability of the LSC to efficiently detect CBCs across a wide range of astrophysical parameters, with runtime improvements on the order of 1×10^2 times reductions in computational costs, measured in terms of CPU time.

While fast, treebank is still a work in progress and will require some fine tuning before implementation within the analytical tools used by the LSC. Most notable are its issues with bank size and performance in bank sims, although the solutions to these problems go hand in hand.

4.2 Looking Forward

Understanding and improving the metric calculation is key to fixing the remaining issues with treebank, with the goal being to consistently pass banksims with little fuss.

Once treebank works satisfactorily for all test cases, we can proceed to generating banks of higher dimensionality, hopefully proving that the method scales well up to 8 or 9 dimensions.

An interesting idea to keep in mind would be a return to a tiling method more faithful the Owen geometric approach, which has the distinction of being the theoretically ”correct” way to tile the template space.

Bibliography

[1] Albert Einstein. Approximative integration of the field equations of gravitation.

[2] P. Kalmus. Effect of a passing gravitational wave on test particles, 2009. [Online; accessed March 28, 2017].

[3] J. Weber. Gravitational-wave-detector events. *Phys. Rev. Lett.*, 20:1307–1308, Jun 1968.

[4] J. M. Weisberg, J. H. Taylor, and L. A. Fowler. Gravitational waves from an orbiting pulsar. *Scientific American*, 245:74–82, October 1981.

[5] The LIGO Scientific Collaboration and B. Abbott. Detector description and performance for the first coincidence observations between ligo and geo. 2003.

[6] R.E. Vogt et al. The construction, operation, and supporting research and development of a laser interferometer gravitational-wave observatory. *LIGO Document Control Center*, 1989.

[7] The LIGO Scientific Collaboration. Advanced ligo. 2014.

[8] the LIGO Scientific Collaboration et al. Search for gravitational waves from low mass compact binary coalescence in ligo's sixth science run and virgo's science runs 2 and 3. 2011.

[9] Stephen Fairhurst. Improved source localization with ligo india. 2012.

[10] A. Abramovici et al. Ligo: The laser interferometer gravitational-wave observatory.

[11] K. Thorne. Gravitational waves.

[12] C. Cutler et al. The last three minutes: Issues in gravitational wave measurements of coalescing compact binaries. *Physical Review Letters*, 70, May 1993.

[13] The LIGO Scientific Collaboration and the Virgo Collaboration. Observation of gravitational waves from a binary black hole merger. 2016.

[14] The LIGO Scientific Collaboration and the Virgo Collaboration. Gw151226: Observation of gravitational waves from a 22-solar-mass binary black hole coalescence. 2016.

[15] The LIGO Scientific Collaboration and B. Abbott. Analysis of ligo data for gravitational waves from binary neutron stars. 2003.

[16] LIGO Scientific Collaboration and B. Abbott et. al. Search for gravitational waves from binary black hole inspirals in ligo data. 2005.

[17] L. S. Finn et al. Observing binary inspiral in gravitational radiation: One interferometer. *Phys. Rev. D.*, Jan 1993.

[18] R. Cole. Likelihood smoothing using gravitational wave surrogate models.

[19] B.S. Sathyaprakash B.J. Owen. Matched filtering of gravitational wave from inspiraling compact binaries: Computational cost and template placement. *Phys Rev D*, Aug 1998.

[20] C.W. Helstrom. *Statistical Theory of Signal Detection*. Permagon, London, 1968.

[21] Benjamin J. Owen. Search templates for gravitational waves from inspiraling binaries: Choice of template spacing. 1995.

[22] P. Ajith, N. Fotopoulos, S. Privitera, A. Neunzert, N. Mazumder, and A. J. Weinstein. An effectual template bank for the detection of gravitational waves from inspiralling compact binaries with generic spins. 2012.

[23] Ian Harry, Bruce Allen, and B. S. Sathyaprakash. A stochastic template placement algorithm for gravitational wave data analysis. 2009.

[24] Ian Harry, Stephen Privitera, Alejandro Boh, and Alessandra Buonanno. Searching for gravitational waves from compact binaries with precessing spins. 2016.

[25] The LIGO Scientific Collaboration, the Virgo Collaboration, et al. Gw150914: First results from the search for binary black hole coalescence with advanced ligo. 2016.

Appendices

.1 Appendix A: Banksims

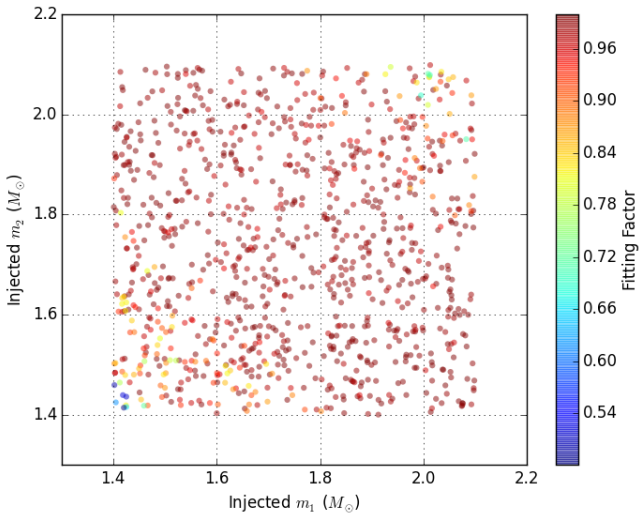

Figure 1: Banksim of a non-spinning bank. Boundaries at $(1.4, 2.2)M_\odot$. Minimum match criteria of 0.97. This was a small parameter test case.

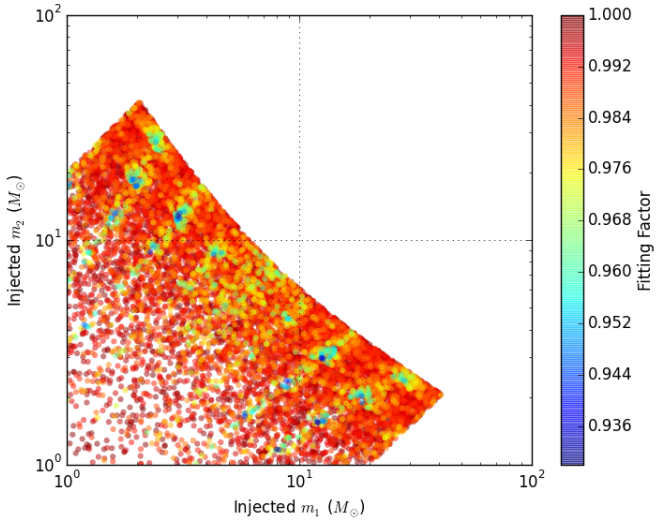

Figure 2: Banksim of a non-spinning bank. Boundaries at $(2.0, 99.0)M_\odot$. Minimum match criteria of 0.97. Note that this template bank had the restriction that the maximum mass ratio was set to 40.0, hence the slanted right edge.

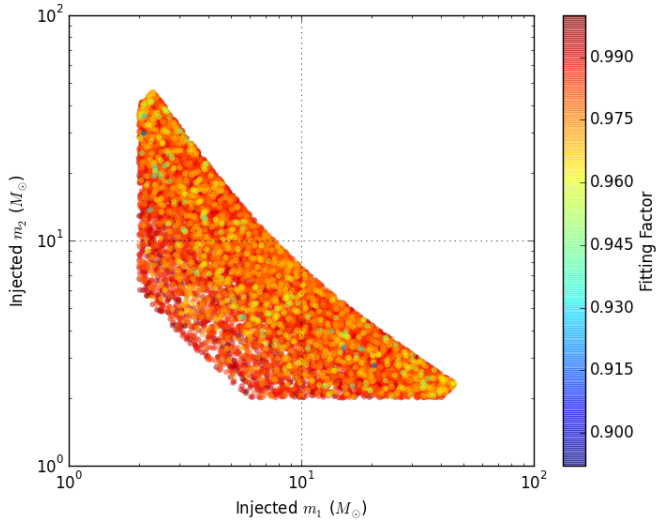

Figure 3: Mass banksim of a spinning bank. Boundaries at $(2.0, 99.0)M_\odot$ and $(-0.985, 0.985)$ spin. Minimum match criteria at 0.97. Note this template bank again has restrictions on the minimum and maximum mass ratios.

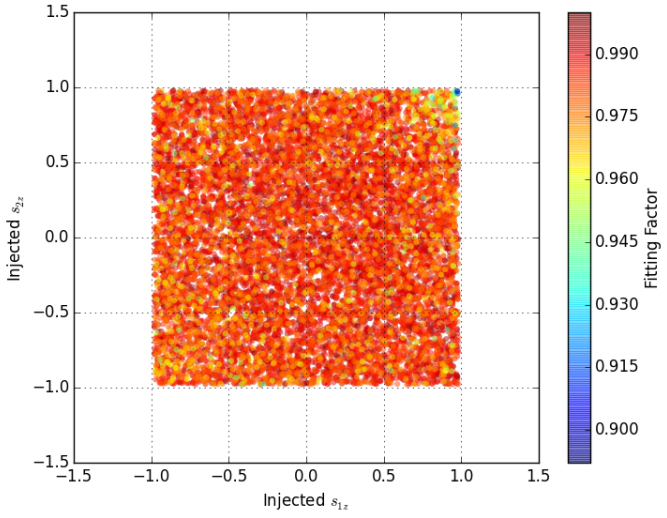

Figure 4: Spin banksim of a spinning bank. Boundaries at $(2.0, 99.0)M_\odot$ and $(-0.985, 0.985)$ spin. Minimum match criteria at 0.97.

.2 Appendix B: Metric Tests

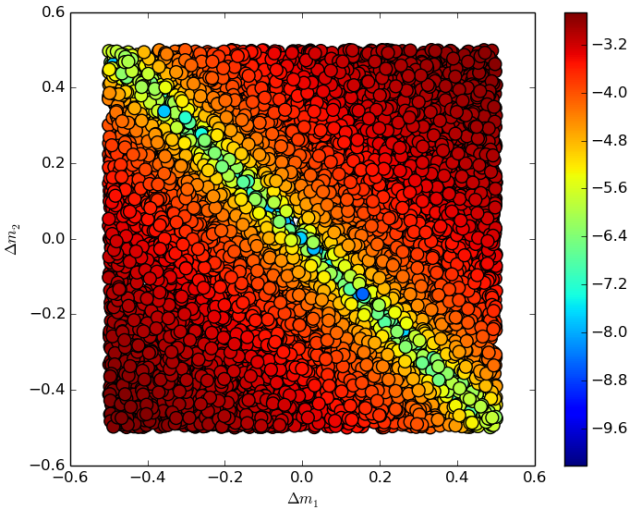

Figure 5: A heatmap of the fractional difference between the metric match approximation and the exact mass calculation around a hypercube with (mass1, mass2, spin1, spin2) coordinates of $(25, 25, 0, 0)$. Perturbations around the center take place along the mass parameter vectors.

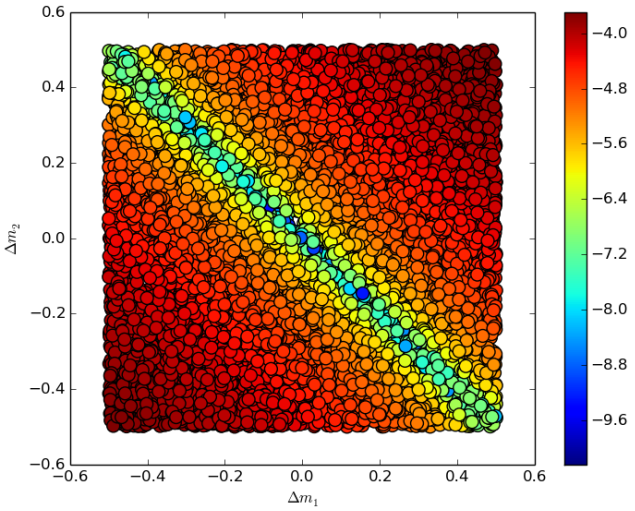

Figure 6: A heatmap of the fractional difference between the metric match approximation and the exact mass calculation around a hypercube with (mass1, mass2, spin1, spin2) coordinates of $(49, 49, 0, 0)$. Perturbations around the center take place along the mass parameter vectors.

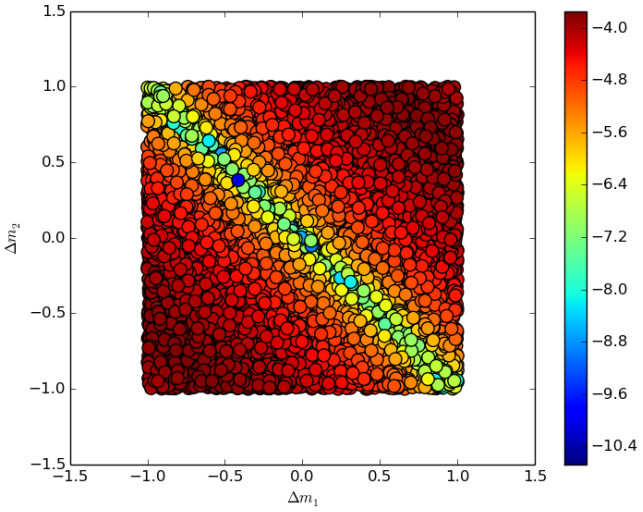

Figure 7: A heatmap of the fractional difference between the metric match approximation and the exact mass calculation around a hypercube with (mass1, mass2, spin1, spin2) coordinates of $(75, 75, 0, 0)$. Perturbations around the center take place along the mass parameter vectors.

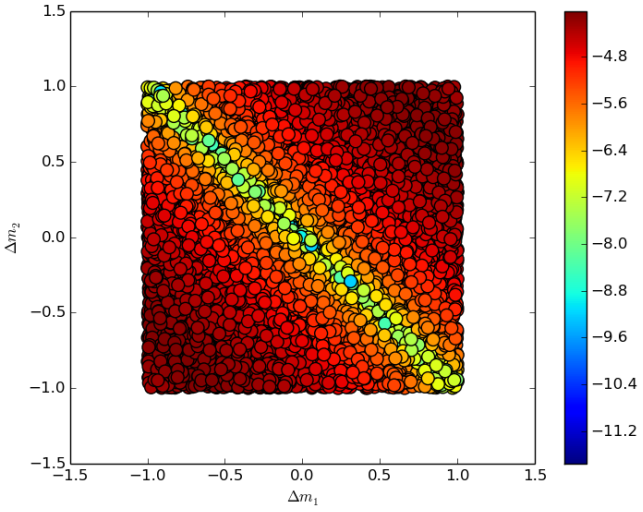

Figure 8: A heatmap of the fractional difference between the metric match approximation and the exact mass calculation around a hypercube with (mass1, mass2, spin1, spin2) coordinates of $(100, 100, 0, 0)$. Perturbations around the center take place along the mass parameter vectors.

Academic Vita

Jonathan Wang

Contact Information

Email: jzw5477@psu.edu

Phone Number: 412.849.9717

Address:

The Pennsylvania State University

Institute for Gravitation and the Cosmos

334 Whitmore Laboratory, E Pollock Road

University Park, Pennsylvania 16801

About

I am a senior at the Pennsylvania State University majoring in physics and minoring in mathematics. I am primarily interested in computational physics and the development of tools and software which open the path to better analytics and simulations of physical phenomenon. I am particularly intrigued by the field of gravitational wave astronomy.

Education

The Pennsylvania State University

B.S., Physics (General Option)

Minor in Mathematics

My coursework is comprised of standards of a physics education including classical mechanics, E&M, quantum mechanics, thermodynamics, solid state physics and subatomics, as well as calculus, statistics, and coding.

Technical Skills

In addition to proficiencies in problem solving and critical thinking garnered from the successful completion of the majority of a undergraduate education in physics I possess knowledge in the following skills:

- Coding in Python, C++, and Java
- Utilizing the bash shell language
- Familiarity with version control using GIT
- Data and statistics analysis
- Basic experimental skills

Honors and Scholarships

2013 - present, Schreyer Honors Scholar

2013 – 2016, Dean's list for all intermediary semesters

2016, John and Elizabeth Holmes Teas Scholarship

Community Service and Research Experience

Undergraduate Research Assistant – The PSU LIGO Group – The Pennsylvania State University

August 2015 to present

The Penn State LIGO group develops techniques which aid in the real-time detection of gravitational waves by the LIGO detectors in Hanford, California and Livingston, Louisiana. It is part of the LIGO Compact Binary Coalescence (CBC) group which focuses on detecting inspiraling binary black hole and neutron star systems.

My work with Dr Chad Hanna is focused on the development of a faster, more computationally efficient method of generating template banks for use in matched filtering data analysis techniques by the CBC group. The methods we are developing produce template banks approximately six times faster than the tool currently used by the CBC group. It is currently our hope to publish a paper detailing this improved template bank generation algorithm within the near future.

Summer Outreach Program Staff – Science-U Outreach/The PSU LIGO Group – The Pennsylvania State University

July 2016

The PSU LIGO Group in conjunction with Science-U, Penn State's primary scientific youth outreach program, hosted high school students for approximately a week for a summer session. The topics presented concerned the fundamentals of gravitational waves, matched filter analysis, and the basics of using bash commands.

As a part of the PSU LIGO Group and the staff member closest in age to the kids, I assisted the participants with their activities, supervised them, and gave a presentation on my work with template banks along with an interactive demonstration on the advantages of parallel computing.

Personal Interests

Outside of computational physics, my primary interest is in music. I have been the assistant director of Blue in the FACE acapella for approximately two years as well as a member of the Penn State Glee Club. I am also an active member of the Penn State Fencing Club.

References

References are available upon request