# Extracting Progenitor Parameters of Rotating CCSNe via Pattern Recognition and Machine Learning

Laksh Bhasin[1]

Mentor: Alan Weinstein. Co-Mentor: Sarah Gossan.

[1] *LIGO SURF 2014, California Institute of Technology, 1200 E California Blvd., Pasadena, CA 91125, USA*

**Abstract**

Core-collapse supernovae (CCSNe) are among the most energetic events in the universe, releasing up to $10^{53}$ erg = 100 B of gravitational potential energy. Based on theoretical predictions, they are also expected to emit bursts of gravitational waves (GWs) that will be detectable by second-generation laser interferometer GW observatories such as Advanced LIGO (aLIGO), Advanced Virgo, and KAGRA. In a novel pattern-recognition approach, we investigate the inference of progenitor parameters from numerical GW signals produced by state-of-the-art rotating core-collapse simulations. After associating physical processes with characteristic spectrogram features, we develop several machine-learning (ML) algorithms that can accurately (often within ±20% relative error on average) and precisely determine progenitor parameters from optimally-oriented CCSN signals located 5 kpc away from Earth. In particular, our $\pm 2\sigma$ prediction intervals for $\beta_{\mathrm{ic,b}}$ (the ratio of rotational kinetic energy to gravitational energy of the inner core at bounce) are $\sim 0.03$ wide on average at 5 kpc. At a source distance of 10 kpc, we still achieve average relative errors within ±20% for our mean predictions, though our $\pm 2\sigma$ prediction intervals for $\beta_{\mathrm{ic,b}}$ become $\sim 0.05$ wide. In addition to our hand-picked "physical" feature vector (FV) approach, we also investigate FV constructions with principal component analysis (PCA) and the scale-invariant feature transform (SIFT). In the future, our analysis could be implemented in the aLIGO data analysis pipelines to help determine the inner core dynamics of the next galactic CCSN; this information would otherwise be inaccessible via electromagnetic radiation.

## I. Introduction

Towards the end of its hydrostatic-burning phase, a massive star (i.e. $8-10\,M_\odot \lesssim M \lesssim 130\,M_\odot$ at zero-age main sequence (ZAMS)) is composed of several concentric shells that represent its previous burning phases: hydrogen, helium, carbon, neon, oxygen, and silicon. As the silicon shell burns, an iron core starts to develop and increase in mass; when the mass of this core becomes sufficiently large, electron degeneracy pressure can no longer stabilize the core against gravitational forces [1]. This triggers the collapse of the inner core, where material is compressed to supranuclear densities of $\rho \gtrsim \rho_0 \sim 2.7 \times 10^{14}\,\mathrm{g/cm}^3$. Due to the low compressibility of nuclear matter (i.e. the stiffening of the equation of state) and the repulsive nature of the nuclear force, the inner core decelerates and starts to "bounce" back; this creates a hydrodynamic shock wave that propagates outwards until it collides with the supersonically-infalling outer core (see Figure 1, adapted from [1]). However, energy losses (e.g. to disassociation of heavy nuclei in the post-shock region) stall the shock wave, and the wave must somehow be revived to pass through the remaining outer core, produce a supernova (SN), and leave behind a neutron star. If this revival does not take place, black hole formation will occur instead of a core-collapse supernova (CCSN).

As one of the most energetic processes in the universe – releasing $10^{53}$ erg = 100 B of gravitational energy, ∼99% of which is carried away by neutrinos [2] – core-collapse supernovae (CCSNe) are of high astrophysical significance. However, while the above description is in general accurate, the true process that triggers the shock wave's revival is not well understood. The actual CCSN mechanism is believed to be a combination of convection, rotation, neutrino heating, magnetic fields, and accretion shock instabilities.

To get a better understanding of the core dynamics involved in a CCSN, we can study the gravitational waves (GWs) emitted. These waves are emitted from dense nuclear regions that are impenetrable by photons, and carry valuable information about the progenitor's parameters and its inner core's physical processes. Moreover, unlike electromagnetic waves, they remain largely unaffected by intervening material as they travel at the speed of light towards Earth.

Rotating CCSNe are of particular interest to GW astronomers, since the rotation in these progenitors leads to an oblate deformation of the collapsing core. As this deformed core passes through the collapse and bounce phases, it undergoes extreme amounts of acceleration that in turn lead to strong time-varying quadrupole moments. The resulting GW burst signal from pressure-driven core bounce can therefore be fairly strong; for rapidly-rotating cores, it can be detected by second-generation GW detectors out to > 10 kpc [3].

Recent estimates based on historical records of supernovae and the simulated observability at a latitude of 35°N suggest a galactic CCSN rate of $3.2^{+7.3}_{-2.6}$ per century [4]. Given these chances of observability, it is therefore necessary to develop a set of tools that will let us analyze a given GW signature from a CCSN and determine the underlying progenitor parameters. Unfortunately, core collapse events in general cannot be handled with the same template search methods that have been successful with binary inspiral mergers, since their resulting GW strains are dependent on many more variables (e.g. the equation of state (EOS) and neutrino transport schemes, both of which are themselves highly parametrized). Since it is currently not computationally feasible to cover the entire signal parameter space of CCSNe via numerical simulations, alternative parameter estimation and waveform reconstruction techniques have been developed.

Many of these alternative techniques have focused on the GW signal from rotating core collapse, as this tends to have

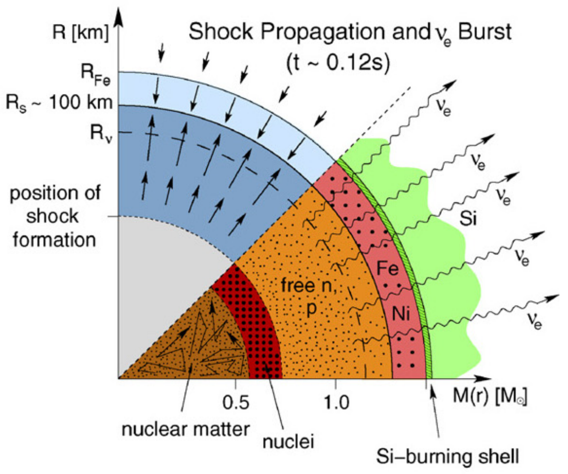


**Figure 1:** *A schematic representation of the post-bounce shock-propagation stage that occurs during a core-collapse supernova. The upper half of this image depicts dynamical conditions, with arrows representing velocity vectors; these indicate a clear shock-wave expansion against gravitational collapse forces. The lower half indicates the star's nuclear composition and nuclear and weak processes; among these, we find neutrino bursts that result from the shock wave losing energy to neutrinos. The horizontal axis gives the enclosed mass in $M_\odot$, and the vertical axis shows the corresponding radii in km (where $R_{\rm Fe}$ is the iron core radius and $R_\nu$ is the neutrinosphere radius). Figure adapted from [1].*

a relatively simple morphology. In particular, the signal consists of a pre-bounce rise in the strain $h$, a large spike at bounce, and a brief ($\lesssim 10$ ms) postbounce ringdown [5] (see, for example, Figure 3). If we purely consider this rotation-induced segment of the GW signal, and ignore the ensuing strains produced from convection, we can also make several simplifications in our parameter estimation problem. For instance, Dimmelmeier *et al.* [6], who considered two finite-temperature nuclear equations of state, found a fairly weak dependence of this GW signal on the EOS. Furthermore, a recent study by Ott *et al.* [3] established a phenomenon known as the "universality of rotating core collapse." Looking at both $12\,M_\odot$ and $40\,M_\odot$ progenitors with the same precollapse rotation rates, Ott *et al.* found fairly minimal differences in the spectral GW energy density $\mathrm{d}E_{\rm GW}/\mathrm{d}f$ in the case of even moderate progenitor rotation. In fact, as Abdikamalov *et al.* [5] recently established, the morphology of the rotating CCSN's GW signal depends most noticeably on the angular momentum of the precollapse core.

This relative simplicity of rotating CCSNe has recently been leveraged by Engels *et al.* [7] for waveform reconstruction. After performing singular value decomposition (SVD) on the Abdikamalov catalog [5] to create a principal component (PC) basis of strain waveforms, Engels *et al.* used least-squares regression to link physical parameters to members of the PC basis. This allowed for an accurate prediction of waveforms given a set of physical progenitor parameters. Furthermore, in the realm of parameter estimation, Abdikamalov *et al.* were able to use matched filtering on noisy waveforms from their own CCSN catalog to infer the total angular momentum of the inner core at bounce; their mean predictions achieved relative errors within ±20%, assuming optimally-oriented progenitors located 10 kpc away from Earth [5]. Edwards *et al.* [8] also managed to achieve accurate results on weak GW CCSN signals by using Bayesian techniques to regress continuous physical parameters on the PC basis from the Abdikamalov training catalog. With a separate set of noisy injection waveforms, they then used their regression model to fit physical parameters on the posterior means of the PC coefficients. By applying this technique to waveforms with low signal-to-noise ratios (SNRs) of 20, they achieved 90% confidence intervals for $\beta_{\rm ic,b}$ (the ratio of rotational kinetic energy to gravitational energy of the inner core at bounce) that were only $\sim 0.06$ wide in the case of an unknown arrival time.

In this paper, we investigate a pattern-recognition/machine-learning (ML) approach to parameter estimation for rotating CCSNe. In particular, we analyze the spectrogram-domain (time-frequency plane) and time-domain waveforms from the Abdikamalov catalog and identify key visual similarities and differences between GWs produced via different progenitors. By developing various image-processing and signal-processing algorithms to automate this feature-identification process, and by combining these features into a feature vector (FV), we are able to simultaneously estimate various progenitor parameters of a given CCSN GW signal via ML techniques. Assuming optimally-oriented progenitors located $D = 5$ kpc from Advanced LIGO (aLIGO), the mean predictions of our best-performing algorithms tend to have relative errors within ±20% for most waveforms in the Abdikamalov catalog. Furthermore, our $\pm 2\sigma$ prediction intervals for $\beta_{\rm ic,b}$ are $\sim 0.03$ wide on average when $D = 5$ kpc; this width gives a reasonable estimate of the spread of our predictions due to different aLIGO noise instantiations. At the fiducial galactic distance of $D = 10$ kpc, our average predictions still remain within ±20% of the true parameter values in terms of relative error, yet the $\pm 2\sigma$ intervals for $\beta_{\rm ic,b}$ become $\sim 0.05$ wide on average.

In addition to $\beta_{\rm ic,b}$, we also look into predicting $J_{\rm ic,b}$ (the total angular momentum of the inner core at bounce), $M_{\rm ic,b}$ (the total mass of the inner core at bounce), and the degree of differential rotation. Our results show that $M_{\rm ic,b}$ can be predicted very accurately (with mean predictions well within ±5% relative error) for progenitors 5 and 10 kpc away, due to the dense sampling of $M_{\rm ic,b}$ space by the Abdikamalov catalog. We also find more accurate differential rotation predictions for higher values of $\beta_{\rm ic,b}$ at $D = 5$ kpc, due to slight differences in waveform morphology that occur in rapidly-rotating models.

The aforementioned results were all achieved with our hand-picked set of seven spectrogram-based and strain-based features (e.g. spectrogram "blob" bandwidths), which we will hereafter refer to as our "physical" feature set. We use the word "physical" to distinguish our choice of features from that of (for instance) principal component analysis (PCA), which involves unphysical principal components. For completeness, we also constructed ML feature vectors using both the PCA of our spectrograms and the scale-invariant feature transform (SIFT, an image-processing algorithm). The SIFT-based "visual bag-of-words" technique is popular for object recognition in real-life images, but it is not sufficiently accurate or precise for our application, even at $D = 1$ kpc. On the other hand, spectrogram-based PCA performs well at $D = 1$ kpc, but its predictions are not as accurate or as precise as those of our physical FV approach at $D = 5$ kpc.

The remainder of this paper is organized as follows. In Section II, we describe our GW data, progenitor parameters, and spectrogram construction procedures. This is followed by a description of our physical FV construction in Section III. In Section IV, we discuss our ML algorithm choice, training/testing procedure, and the details of our random forest implementation. The main parameter prediction results of this paper, based on our physical FV and random forest, are presented in Section V. Lastly, in Section VI, we look into PCA and SIFT as alternative FV construction methods.

## II. GW Data and Preparation

### I. Catalog and Physical Parameters

With regards to our strain data, we use the numerically generated waveforms from Abdikamalov *et al.* [5]. These waveforms were generated in axisymmetric (2D) conformally-flat GR with the CoCoNuT code [9]. For the nuclear physics, Abdikamalov *et al.* used the Lattimer and Swesty (LS) EOS (available for download from `stellarcollapse.org`) with an incompressibility parameter of $K = 220\,\mathrm{MeV}$. For neutrino-related physics, the parametrized deleptonization scheme of [6] and neutrino leakage scheme of [3] were employed. Lastly, due to the universality of rotating core collapse [3], only a single ZAMS mass of $12\,M_\odot$ and solar metallicity (from [10]) is considered.

As for the progenitor variables, the waveforms are parametrized by five different degrees of differential rotation (represented by the letter $A$ in [5]) and various initial central angular velocities $\Omega_\mathrm{c}$. Assuming a cylindrical rotation law, the differential rotation parameter $A$ modifies the angular velocity distribution $\Omega(s)$ via the following equation:

$$\Omega(s) = \Omega_\mathrm{c}\left[1 + \left(\frac{s}{A}\right)^2\right]^{-1}, \tag{1}$$

where $s$ is the cylindrical distance from the center of the progenitor. The differential parameter $A$ was chosen to take on the following five values: $A1$ = 300 km, $A2$ = 417 km, $A3$ = 634 km, $A4$ = 1268 km, and $A5$ = 10000 km. These choices were made so that, in the inner core (i.e. the inner $\sim 1.5M_\odot$) of the progenitor model used, $A1$ corresponds to extreme differential rotation (i.e. $\Omega$ changes rapidly with $s$), whereas $A5$ corresponds to more uniform differential rotation [5]. For the remainder of this report, the Abdikamalov waveforms are referred to by their degree of differential rotation ($A1$, $A2$, etc) and their $\Omega_\mathrm{c}$ value. For instance, "A1O01" refers to a waveform with the $A1$ level of differential rotation and with $\Omega_\mathrm{c} = 1$ rad/s.

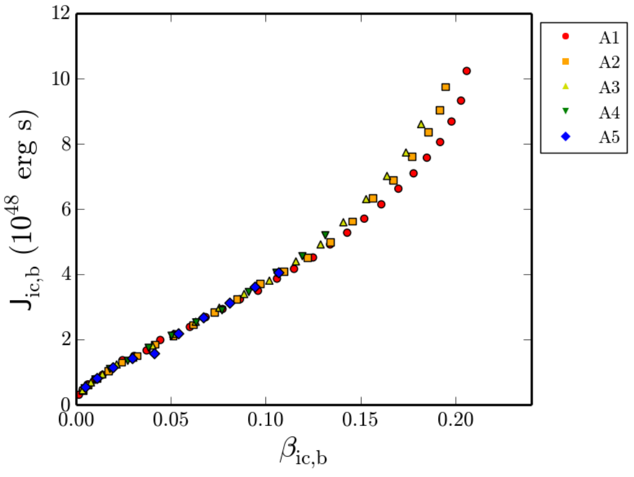
Instead of predicting $\Omega_\mathrm{c}$, we choose instead to infer $\beta_\mathrm{ic,b}$, the ratio of rotational kinetic energy $T$ to gravitational potential energy $|W|$ of the inner core at bounce. As identified in Abdikamalov *et al.* [5], this ratio seems to have a more clear and direct impact on the morphology of a given signal than $A$ and $\Omega_\mathrm{c}$ individually do (due to degeneracies between the impacts of $A$ and $\Omega_\mathrm{c}$). In addition to $\beta_\mathrm{ic,b}$, we predict two other continuous progenitor parameters tabulated in [5]: $J_\mathrm{ic,b}$ (the total angular momentum of the inner core at bounce) and $M_\mathrm{ic,b}$ (the mass of the inner core at bounce). Both of these variables tend to positively correlate with $\beta_\mathrm{ic,b}$ irrespective of the degree of differential rotation A, as shown in Figure 2.

With regards to estimating the differential rotations of our waveforms, we decided to switch to a regression-based approach in place of the discrete classification used in [5] and [8], since the $A$ values essentially represent various numerical distance scales. A regression-based approach would also allow us to simultaneously predict all of our progenitor parameters (the remainder of which are continuous) via a single ML regression algorithm.

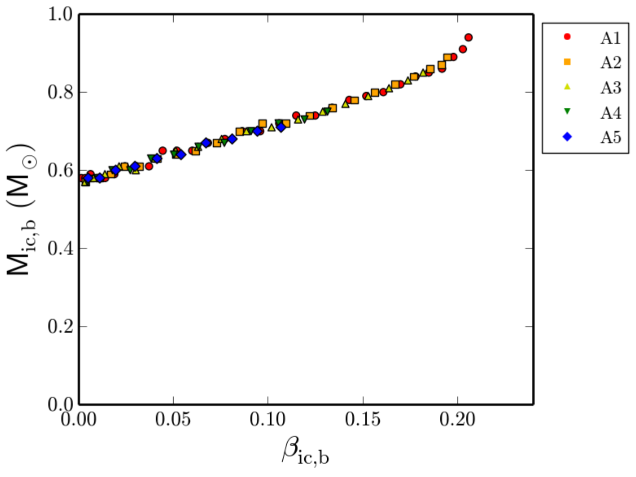
Given the large range of distances covered by the $A$ values, it would not make sense to simply determine $A$ via regression, as interpolating between largely different values would be difficult. Moreover, according to the cylindrical rotation law (Equation (1)), the rotation profiles $\Omega(s)$ do not tend to change as much when $A$ is increased from (say) 9000 km to 10000 km, but they do change quite noticeably from 400 to 1400 km. These facts seem to instead motivate a logarithmic approach. If we take $\log_{10}(A/(1\text{ km}))$ (hereafter referred to as $\log_{10}(A)$ for simplicity), then we get logs of 2.48 ($A1$), 2.62 ($A2$), 2.80 ($A3$), 3.10 ($A4$), and 4.00 ($A5$) for our differential rotations. We can therefore see that $\log_{10}(A1)$ and $\log_{10}(A2)$ are separated by around the same amount as $\log_{10}(A2)$ and $\log_{10}(A3)$ are. If we correspondingly look at the rotation profiles in Figure 1 of [5], we can see that the profiles of $A1$ and $A2$ waveforms are visually separated by around the same amount as the profiles of $A2$ and $A3$ waveforms are from one another (loosely speaking). Thus, when it comes to considering the physical impact of a given differential rotation $A$ on the rotation profile, it makes more sense to look at $\log_{10}(A)$ instead.

### II. Training and Testing Data

A supervised ML algorithm has to be trained on a set of data in order to "learn" the associations between multidimensional feature vectors and their corresponding (known) response variables (in our case, the progenitor parameters of interest). In order to evaluate the generalization of such an algorithm, we must test it on a different set of data and measure the accuracy of its predictions.



**(a)** $J_{\rm ic,b}$ *vs* $\beta_{\rm ic,b}$



**(b)** $M_{\rm ic,b}$ *vs* $\beta_{\rm ic,b}$

**Figure 2:** *$J_{\rm ic,b}$ and $M_{\rm ic,b}$ vs $\beta_{\rm ic,b}$ for the Abdikamalov catalog's waveforms. All three variables correlate positively with one another, irrespective of the degree of differential rotation A.*

We therefore split the Abdikamalov waveforms into training and testing sets. The 92 waveforms that we used for training were the same ones listed in Table II of [5]; these had $\Omega_c$ values separated by 0.5 rad/s for each differential rotation. The 31 waveforms used for testing our procedure came from a subset of the injection set in [5] (partly listed in Table III); we did not consider progenitor masses of 40 $M_\odot$ (signified by "s" in the Abdikamalov catalog), different electron fraction parametrizations $Y_e$ (signified by "m" and "p" in the Abdikamalov catalog), or non-LS equations of state (i.e. we neglected the Shen EOS models). These testing waveforms had values of $\Omega_c$ that differed from those in the training set by at least 0.25 rad/s.

## III. Strain and Spectrogram Data Preparation

To work with our chosen waveforms in the short-time Fourier transform (STFT) spectrogram domain, we resample them evenly at a frequency of 8192 Hz using `NumPy`'s spline linear interpolation method. This frequency was chosen since the spectrograms typically contained negligible power density at frequencies $> 4096$ Hz; thus, by the Nyquist-Shannon Sampling Theorem, we would not risk aliasing or lose much frequency information by sampling our data at 8192 Hz. Finally, to make the waveforms more consistent in their appearance, we aligned their positive peak at bounce (referred to as $h_{1,\rm pos}$ in [5]) to the 1s mark, and padded the signals with zeros until they were all 3s long. This alignment is completely irrelevant in the eyes of our "physical" feature vector and SIFT approaches, but is important for PCA (where consistent positioning of GW signals is important).

One of the most crucial trade-offs in spectrogram analysis is the uncertainty principle, which applies generally to Fourier-transform pairs. In our case, the Gabor limit requires that $\Delta t \Delta f \geq \frac{1}{4\pi}$, where $\Delta t$ is the width of time-domain bins in our spectrogram and $\Delta f$ is the bandwidth of frequency-domain bins. This affects our choice for the NFFT of our spectrograms, i.e. the number of samples used in each short-time Fourier Transform (STFT). Yet another important parameter is the window function for the spectrogram, which is used to minimize the effects of spectral leakages and taper each strain segment at its ends to create a more periodic structure. Since window functions get rid of information at the tapered ends, we make use of sliding windows with an overlap.

After looking at a few "toy model" functions such as sine-Gaussians, we decided to use Kaiser window functions for the purposes of this investigation. These functions have the advantage of an adjustable parameter $\beta$ (not to be confused with the astrophysical $\beta = T/|W|$), which can be used to finely tweak the narrowness of the window; a value of $\beta = 0$ looks similar to a rectangular window, while $\beta = 5$ looks similar to a Hamming window. Decreasing $\beta$ (i.e. widening the window) typically produces spectrogram features that are closer to their true frequencies. However, smaller values of $\beta$ also produce artifacts due to spectral leakage. Taking into account all of these considerations, we tweaked spectrogram generation parameters for our strain data in order to optimally highlight important features in the GW signals. For a waveform sampled at 8192 Hz, we found that an NFFT of 16 worked well at capturing time-dependent variations in our CCSN signals without losing too much frequency resolution (which ended up being 32 Hz per bin). Moreover, a Kaiser window with $\beta = 3$ and an overlap of 66% sufficiently narrowed the frequency spread of features without introducing artifacts.

A typical noiseless waveform and STFT spectrogram (generated with the aforementioned settings) from the Abdikamalov catalog can be seen in Figure 3; we assume that the source is optimally oriented and located $D = 1$ kpc from Earth. At the beginning of the signal, we see the characteristically sharp bounce spike associated with the pressure-dominated bounce; this spike is especially noticeable in this waveform due to

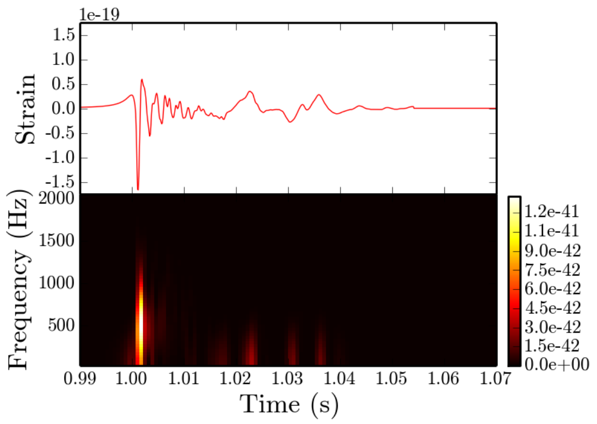


**Figure 3:** *Strain $h_+$ in the time and spectrogram domains for a $12\,M_\odot$ progenitor located 1 kpc away, with differential rotation A1 and $\Omega_\mathrm{c} = 13$ rad/s. The units for the spectrogram's power density (represented by the color bar) are in* strain$^2$ Hz$^{-1}$.

the relatively high $\Omega_\mathrm{c}$ of 13 rad/s. After the spike, we see $\sim 10$ ms of ringdown, which physically corresponds to the protoneutron star (PNS) dissipating its leftover energy. Lastly, around the 1.02 s mark, we start to see growing amplitudes due to prompt convection. Note that we do not see any signs of the violent, sloshing standing accretion shock instability (SASI) since this typically occurs hundreds of milliseconds after bounce [11] – well beyond the Abdikamalov waveforms' simulation length.

In the spectrogram domain (the lower half of Figure 3), all of these features manifest themselves as "blobs" with various durations and frequency bandwidths, and we can clearly distinguish the pressure-driven bounce blob from the convection blobs as it is much "taller" (i.e. has a much larger frequency bandwidth and higher central frequency). These readily-recognizable features are particularly useful for an image-processing approach.

## IV. Noise Generation

In order to create a noisy GW signal, we scale a clean signal to the desired progenitor distance $D$ and then add it to simulated Gaussian Advanced LIGO (aLIGO) detector noise, where the detector is assumed to be in "zero detuning, high-power" (ZDHP) mode [12]. To create this noise, we first generate $N$ points of random Gaussian noise with a mean of 0 and a standard deviation of 1, where $N = f_s t$ is the number of samples desired, $f_s$ is the sampling rate (in Hz), and $t$ is the signal length (in s). We then get the FFT of the random noise and its corresponding frequencies, and then evaluate the ZDHP power spectral density (PSD) at these frequencies via spline-linear interpolation. Since these frequencies can be negative, we take their absolute value before looking at the PSD. We also give the 9 Hz noise value for frequencies $< 9$ Hz as there is no available data for any lower frequencies.

After evaluating the ZDHP PSD at the noise frequencies, we get our coloring spectrum $S_c(f)$. The aLIGO-colored noise in the frequency domain is then given by $\sqrt{S_c(f) \times f_s/2}$ times the FFT of the Gaussian noise, where $\sqrt{f_s/2}$ is a normalizing factor that accounts for our sampling frequency. We then take the IFFT of this product and discard the imaginary part to get our Gaussian aLIGO-colored noise in the time domain.

## III. Physical FV Construction

In the following subsections, we describe how we arrived at seven strain-based and spectrogram-based features that correlated with $\beta_\mathrm{ic,b}$ – and therefore also with $J_\mathrm{ic,b}$ and $M_\mathrm{ic,b}$ due to Figure 2.

### I. Spectrogram-Based Features

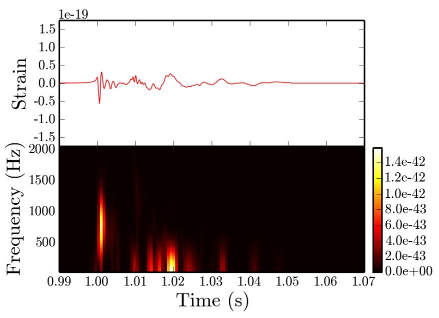
As mentioned in [5], most of the information about the rotational parameters of a CCSN is encoded in its pressure-driven bounce signal, which manifests itself as a large-bandwidth blob in the spectrogram domain. In particular, if we increase $\Omega_\mathrm{c}$ while holding $A$ constant as shown in Figure 4, we find the following trends in our spectrogram:

- The pressure-driven bounce blob becomes more prominent relative to the post-bounce convection blobs.
- The bounce blob's frequency bandwidth decreases.
- The bounce blob's central frequency decreases.
- The bounce blob becomes slightly wider (in the time domain), especially for extreme values of $\Omega_\mathrm{c}$.

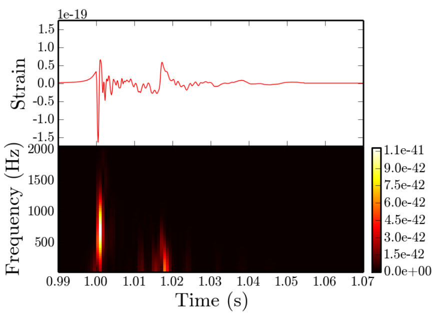
From a signal-processing perspective, the second and third bullet points agree with the fourth: since the bounce takes slightly longer as $\Omega_\mathrm{c}$ and $\beta_\mathrm{ic,b}$ increase, it can be approximated by a lower-frequency sinusoidal wave. Physically, this means that CCSNe with larger angular velocities have a slightly longer bounce, which makes sense given their larger angular momenta. We can also see that more rapidly rotating CCSNe have a more powerful bounce blob in the spectrogram domain; this is just a consequence of their more strongly time-varying quadrupole moments during the collapse and bounce phases.

Based on this visual trend, we note that an image-processing algorithm can key in on the "prominence," bandwidth, central frequency, and duration of the pressure-driven bounce blob in order to infer progenitor parameters. However, no clear correlations could be established between the post-bounce convection blobs' shapes and any rotation-specific parameters; these blobs simply fluctuated too much in bandwidth, time delay, time duration, and power without obeying any clear trend. Furthermore, in practice, the (relatively) low-frequency convection blobs were more susceptible to being wiped out by noise in the spectrogram domain.

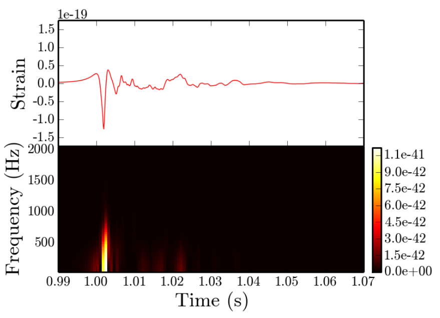
We therefore only have to consider the bounce portion of the GW signal in the spectrogram domain, as the rest of the signal does not clearly encode any physical parameters. While this somewhat limits the efficacy of our image-processing approach, it also simplifies matters greatly since we only need to focus on the bounce blob.



**(a)** $\Omega_c = 5$ *rad/s*



**(b)** $\Omega_c = 10$ *rad/s*



**(c)** $\Omega_c = 15$ *rad/s*

**Figure 4:** *Strain $h_+$ in the time and spectrogram domains for various different values of $\Omega_c$, holding A (the differential rotation parameter) constant at A1. The units for the spectrogram's power density (represented by the color bar) are in* $\text{strain}^2\,\text{Hz}^{-1}$.

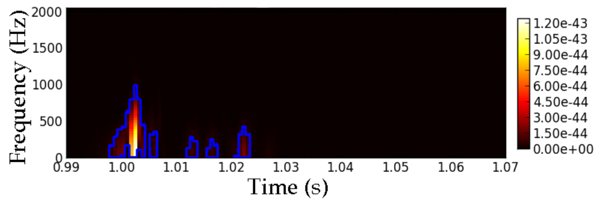


**Figure 5:** *An example of blob detection using our simple binary-image contour-detection approach. This waveform had differential parameter A1 and $\Omega_c = 15.25$ rad/s. The contours (in blue) have been overlaid on our original spectrogram, with axes added for reference.*

In order to actually apply image processing techniques on our waveforms via `OpenCV`, we treat the spectrograms' power density arrays as grayscale images, where rows represent frequency bins up to 2048 Hz (sufficient in our case) separated by 32 Hz, and columns represent time bins up to 3.00 s separated by $\frac{3}{4096}$ s. The grayscale intensities of each image are normalized so that the brightest pixel of each image has a dimensionless intensity of 255.

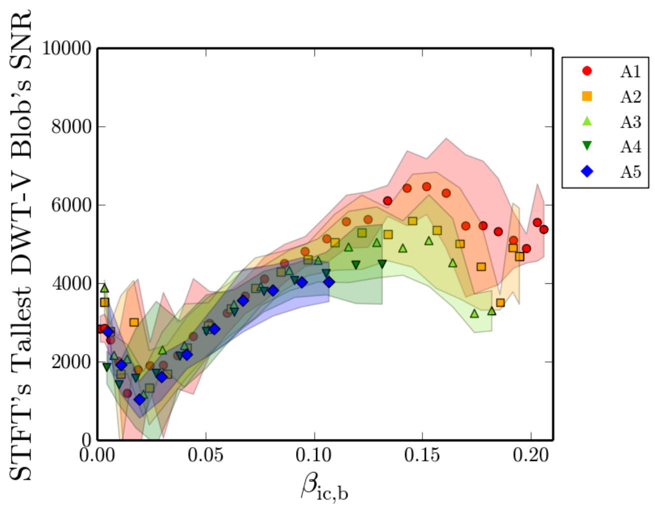
Then, to automate the extraction of the bounce blob, we use a simple binary-image contour-detection algorithm implemented in Python's `OpenCV` library [13]. The first step of this algorithm is to convert our grayscale spectrogram image into a binary image by taking all pixels with intensities greater than a threshold (empirically determined to be 13) and setting their intensities to 255. Pixels below this threshold are set to zero. We then apply `OpenCV`'s `findContours()` function, which uses a border-following approach described by Suzuki *et al.* [14] to find the contours in a binary image. As shown in Figure 5 (in the case of a noiseless signal), this algorithm succeeds at identifying the blobs of interest in our spectrograms.

After finding the bounce blob, it is straightforward to determine its bandwidth (based on its height) and its central frequency (via centroiding). In order to calculate the blob's power content, however, we choose to find its signal-to-noise ratio (SNR); this is both dimensionless (unlike the spectrogram's power) and takes on values with reasonable orders of magnitude. The $\text{SNR}^2$ of a blob can be computed by taking the power data in a spectrogram (which has units of $\text{strain}^2\,\text{Hz}^{-1}$), multiplying it by 4, and dividing it by the aLIGO ZDHP PSD (which has units of $\text{Hz}^{-1}$) at each frequency to get a dimensionless result. This is then summed over all pixels in the blob, and a square root is taken to get the blob's SNR.

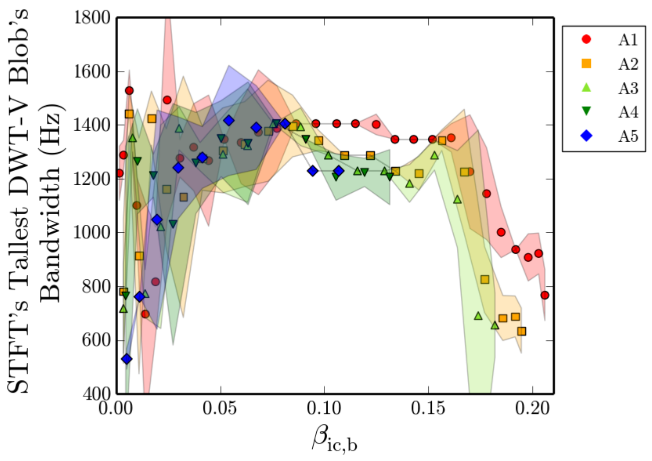
In noisy signals, the procedure of finding the bounce blob becomes more complicated. When we add aLIGO-colored ZDHP Gaussian noise, the noise manifests itself in the form of long (time-wise), relatively high-power, low-frequency bands in the spectrogram domain. These "noise blobs" often can (due to their random appearances) obscure our bounce blob and thwart our contour-detection algorithms.

In order to get around this limitation, we note that our bounce blob constitutes a "vertical detail" in our spectrogram, while the noise was predominately a "horizontal detail" due to its long time duration. Following this line of reasoning, we performed a discrete wavelet transform (DWT) in the vertical direction (hereafter referred to as DWT-V) on each of our input images, using the `PyWavelets` package [15]. Each of these DWT-Vs used the `Symlets 4` wavelets, which are near-symmetric, orthogonal, and have a compact support of four. Since the detail coefficients of a DWT capture edges and fine details, while noise is more of a long-lasting band, this approach ended up being particularly effective at filtering out noise. However, a single DWT also downsizes an input image (by $\approx \frac{1}{2}$ in each dimension) due to its recursive nature, and hence this decreases the time and frequency resolution of our data even more.

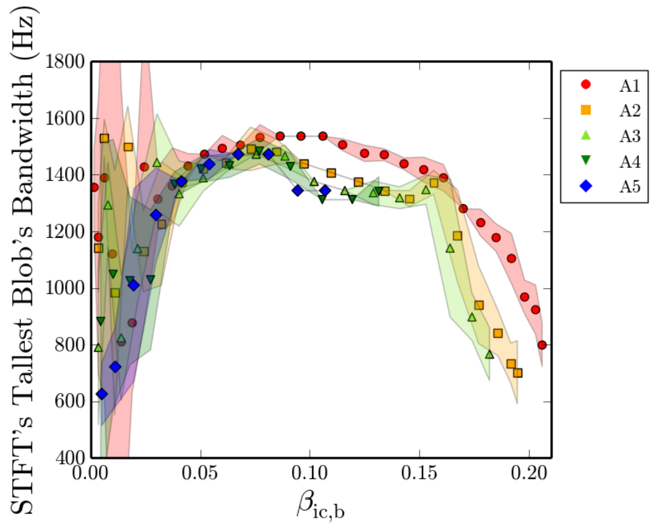
Performing the DWT-V gave us three fairly noise-resistant features: the SNR, bandwidth (in Hz), and central frequency (in Hz) of the STFT spectrogram's tallest DWT-V blob (determined via border-following as before). We also added the STFT spectrogram's tallest regular (i.e. non-DWT) blob's bandwidth (in Hz) to our feature vector, since we found that this was not too negatively affected by noise at most distances. Of course, this bandwidth should correlate with the tallest DWT-V blob's bandwidth, but it tends to be numerically different and has a higher frequency resolution (due to the DWT-V's resolution downscaling).



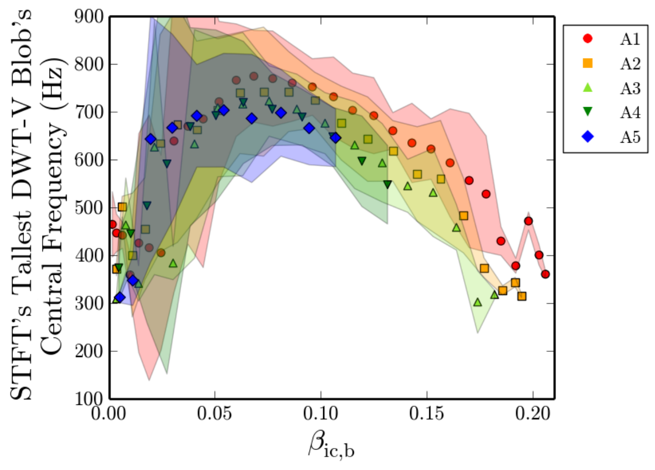
**(a)** *STFT Spectrogram's Tallest DWT-V Blob's SNR*



**(b)** *STFT Spectrogram's Tallest DWT-V Blob's Bandwidth*



**(c)** *STFT Spectrogram's Tallest Regular Blob's Bandwidth*



**(d)** *STFT Spectrogram's Tallest DWT-V Blob's Central Frequency*

**Figure 6:** *Our four spectrogram-based features of interest as functions of $\beta_{\rm ic,b}$ for all Abdikamalov training waveforms, across all five differential rotations. We have chosen $D = 1$ kpc for our source's distance and have included $\pm 3\sigma$ statistical error bars that result from twenty different noise instantiations.*

The relations between our four spectrogram-based features and $\beta_{\mathrm{ic,b}}$ are shown in Figure 6. To include the effects of noise in this figure, we use twenty separate instantiations of aLIGO noise for each waveform in the catalog, where each waveform is scaled to $D = 1$ kpc before adding noise. We then graph the mean feature values for each waveform with points, and use translucent shading for $\pm 3\sigma$ statistical error bars to show the expected spread due to noise. While one could argue that symmetric, $\pm 3\sigma$ intervals are a bit excessive, they do reasonably indicate the kinds of variation we could expect simply due to randomness in the aLIGO measurement procedure.

As Figure 6 demonstrates, our spectrogram-based features seem to depend mostly on $\beta_{\mathrm{ic,b}}$, with significantly less dependence on the differential rotation. We also see the aforementioned visual trends in our figure. For instance, the tallest DWT-V blob's SNR tends to increase with $\beta_{\mathrm{ic,b}}$ before mostly flattening out. In the central frequency graph and the two bandwidth graphs, we see a decreasing trend for $\beta_{\mathrm{ic,b}} \gtrsim 0.05$ as mentioned earlier. On the other hand, for $\beta_{\mathrm{ic,b}} \lesssim 0.05$, the bandwidth and central frequency increase with $\beta_{\mathrm{ic,b}}$ as the blob is starting to fade into view on our spectrogram. Unfortunately, for very low values of $\beta_{\mathrm{ic,b}}$, there is no clear trend in any of these visual features since the bounce blob is too dim to be seen. This is another important drawback of taking an image-processing approach to parameter estimation.

## II. Strain-Based Features

Since the four aforementioned spectrogram-based features are not sufficient for an ML algorithm – especially since only one of them changes monotonically with $\beta_{\mathrm{ic,b}}$ – we also include three additional features that can be determined directly from the strain $h_+(t)$ without image processing.

For example, we can directly use the strain data to get a sense of "how much" of our entire waveform's signal is due to (say) convection, and how much is due to the pressure-driven bounce. This can be quantized by looking at the signal-to-noise ratio (SNR) integrand of our training waveforms over various bands in frequency space, since we expect the pressure-driven bounce signal to occupy higher frequencies than the convection signal. The integrand of interest in this case is given by:

$$\mathrm{SNRSI} = 4\frac{\left|\tilde{h}(f)\right|^2}{S_h(f)}, \qquad (2)$$

where $\tilde{h}(f)$ is the Fourier transform of the strain $h(t)$, $S_h(f)$ is the one-sided aLIGO ZDHP spectral noise density, and SNRSI stands for "SNR Squared's Integrand." In Figure 7, we have plotted this integrand as a function of frequency for the noiseless A1O07 ($\Omega_{\mathrm{c}} = 7$ rad/s), A1O12.5 ($\Omega_{\mathrm{c}} = 12.5$ rad/s), and A1O15.5 ($\Omega_{\mathrm{c}} = 15.5$ rad/s) waveforms. We can clearly see the emergence of the pressure-driven bounce in the 400+ Hz band of Figure 7b. Moreover, as we increase $\Omega_{\mathrm{c}}$, the peak corresponding to the pressure-driven bounce shifts towards lower frequencies. This agrees with what we had seen in the spectrogram domain: the bounce blob's central frequency tends to decrease as we increase $\Omega_{\mathrm{c}}$ (holding $A$ constant).

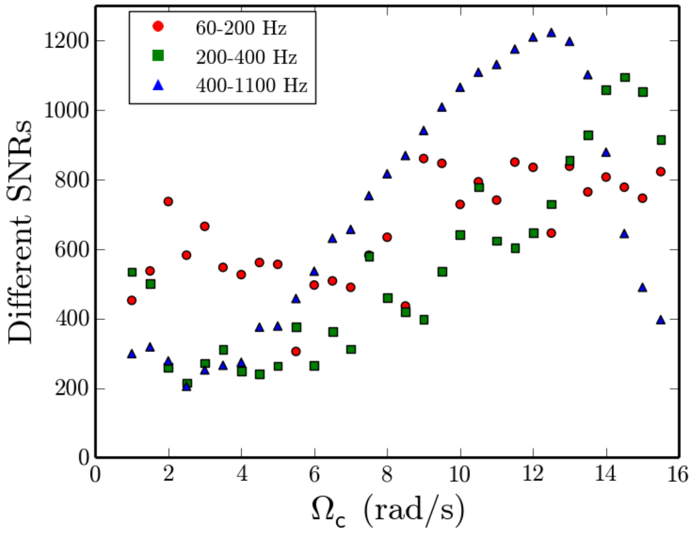
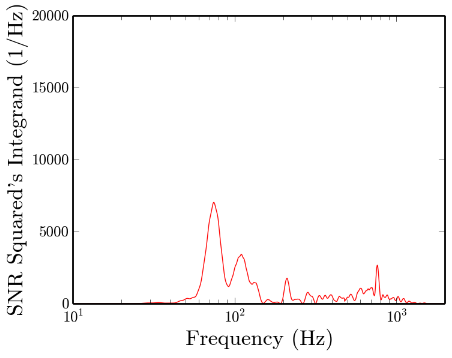


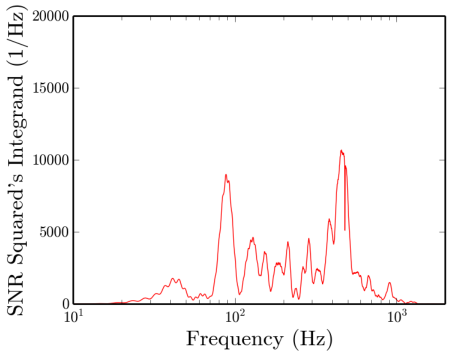
**Figure 8:** *SNR in various frequency bands for noiseless A1 waveforms from the Abdikamalov catalog, as a function of $\Omega_{\mathrm{c}}$. We set $D = 1$ kpc for these waveforms.*

In order to quantify the amount of signal from each contributing source (e.g. pressure-driven bounce vs. convection) in each waveform, we split up our frequency domain into three bands: a 60-200 Hz band for low-frequency neutrino convection content; a 200-400 Hz band for higher-frequency prompt convection; and a 400-1100 Hz band that predominantly captures the pressure-driven bounce (although the bounce sometimes shifted towards frequencies less than 400 Hz). Higher-frequency content was ignored as it was mostly suppressed by the aLIGO noise PSD. We then performed Riemann integration on the SNRSI to calculate the SNR$^2$ in each of the three bands, and took the square root of this result to get a total of three SNR features.

The results of this integration are shown in Figure 8, where we plot each SNR as a function of $\Omega_{\mathrm{c}}$ for noiseless $A1$ waveforms, assuming a 1 kpc distance from the source. The pressure-driven bounce's gradual emergence can be seen in the almost monotonic increase of the 400-1100 Hz SNR from $\Omega_{\mathrm{c}} = 4$ rad/s to $\Omega_{\mathrm{c}} = 12$ rad/s. At higher values of $\Omega_{\mathrm{c}}$, the bounce signal starts to move into the 200-400 Hz band instead. We can also see a general (but less clear) increase in the 200-400 Hz SNR as a function of $\Omega_{\mathrm{c}}$. Physically, this might seem to indicate an increase in prompt convection with $\Omega_{\mathrm{c}}$, but it is more likely that some of the bounce signal is creeping into the 200-400 Hz band.

The 60-200 Hz band, on the other hand, exhibits very unclear behavior as a function of $\Omega_{\mathrm{c}}$. It's possible that this is just the physical truth, i.e. that the neutrino convection signal is simply too stochastic (in both the time and frequency domains) for us to extract any meaningful conclusions about progenitor parameters. On the other hand, it is likely that the Abdikamalov simulations did not run long enough to resolve


**(a)** $\Omega_c = 7\ rad/s$


**(b)** $\Omega_c = 12.5\ rad/s$

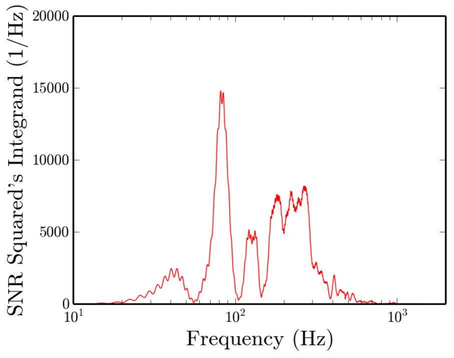

**(c)** $\Omega_c = 15.5\ rad/s$

**Figure 7:** *The SNRSI (defined in (2)) as a function of frequency for three A1 (strongly differentially rotating) waveforms from the Abdikamalov catalog. We set $D = 1$ kpc. Note that we have not added any noise to the underlying waveforms.*

all of the post-bounce convection, and so we simply have incomplete convection data.

Despite these concerns, the 200-400 Hz and 400-1100 Hz SNRs are still useful features to use for categorizing our waveforms. Moreover, the SNR integrals tend to be very robust to noise. While the individual integrands (at specific frequencies) might be less robust to random, Gaussian, aLIGO-colored noise, the effects of noise tend to "cancel out" reasonably well when it comes to measuring the integral as a whole.

As our third and final strain-based feature, we looked into measuring a proxy for the bounce duration. As mentioned in the previous subsection, the bounce blob became slightly wider in the time domain, especially for large values of $\beta_{\rm ic,b}$. However, we cannot easily measure the bounce blob's duration via the STFT spectrogram due to its limited time resolution, and so we instead use time-domain methods. In order to tackle noise, we first apply a 20th-order low-pass Butterworth filter to our signal, with a passband frequency of 1530 Hz and a stopband frequency of 1800 Hz. This gets rid of some of the very high-frequency random noise in our strain data and helps minimize the number of "false peaks" in our strain signal. Next, we apply a 3rd-order high-pass Butterworth filter with a passband frequency of $\sim 82$ Hz and a stopband frequency of $\sim 20.5$ Hz. This removes most low-frequency, large oscillations from our strain signal, and helps ensure that – at points where no CCSN signal was present – the filtered $h(t)$ was close to zero.

After applying these two filters, we determined the most negative minimum in the strain, and assumed it was the negative trough that occurs at bounce time in CCSN signals. We then found the positive maximum right before this trough, and assumed that it was the positive peak in $h(t)$ right before bounce. These points can be seen in Figure 9. As a proxy for the bounce duration, we then take the time difference between these two extrema. In general, this algorithm works well for finding the bounce duration for $\beta_{\rm ic,b} \gtrsim 0.025$; for lower values, it often keys in on unrelated convection extrema since the bounce signal is too weak.

In Figure 10, we plot our three strain-based features of interest as functions of $\beta_{\rm ic,b}$ for all differential rotations. Once

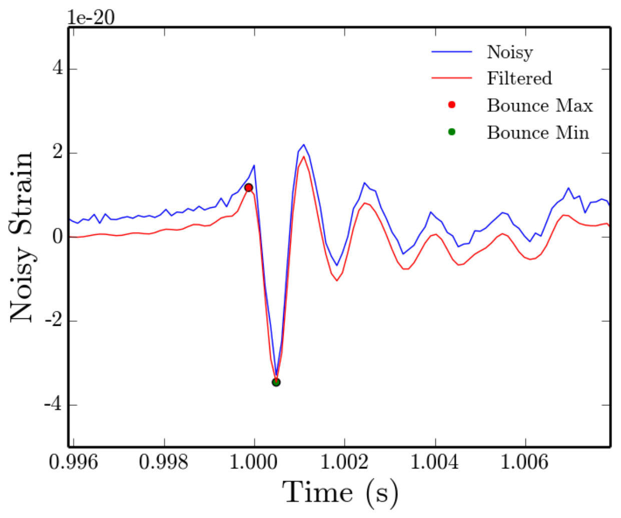
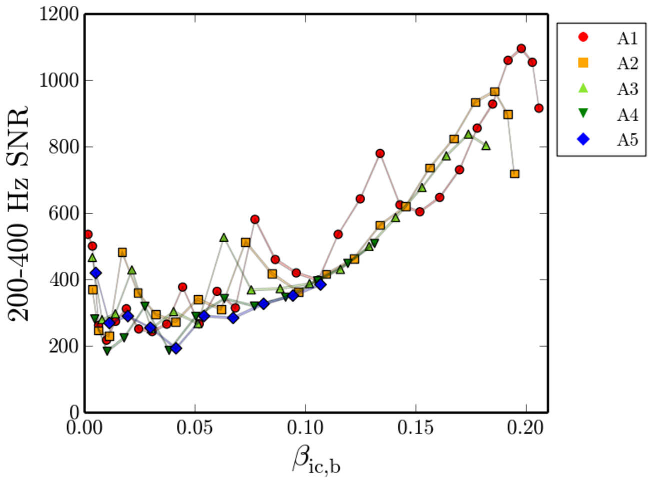


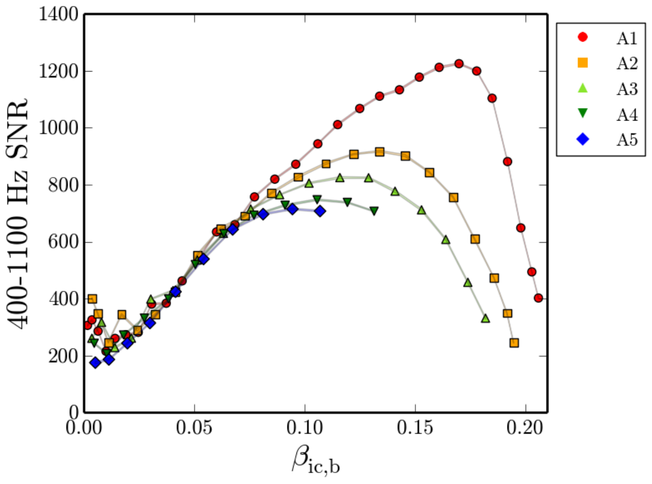
**Figure 9:** *Filtering and extrema-finding on a noisy strain signal ($D = 1$ kpc) for the A1O04 waveform.*

again, we scale our waveforms to $D = 1$ kpc, use twenty instantiations of aLIGO noise per waveform, and include $\pm 3\sigma$ statistical error bars.

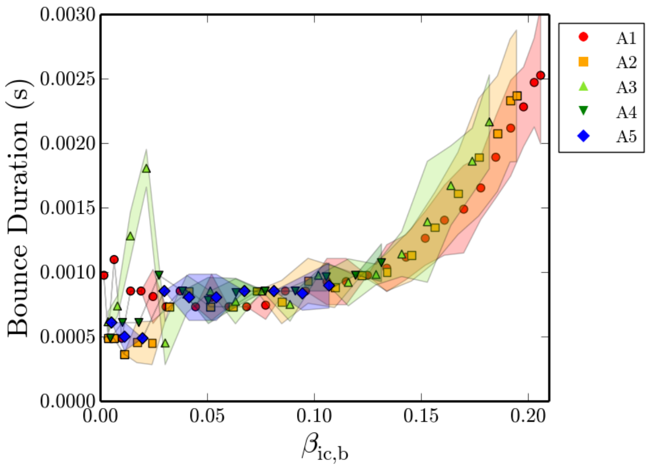
It is worth noting that, in Figure 10b, for high values of $\beta_{\rm ic,b}$ (i.e. $\beta_{\rm ic,b} \geq 0.10$), there are discrepancies in the 400-1100 Hz SNR for different differential rotations (holding $\beta_{\rm ic,b}$ constant). This finding agrees with Figure 9 of [5], where Abdikamalov *et al.* noticed that rapidly spinning models (with $\beta_{\rm ic,b} \sim 0.10$) had significant differences in waveform morphology with different values of $A$. Specifically, while the length of the bounce spike (in the time domain) remained the same for all values of $A$, more strongly differentially rotating models had more extreme strain values during the bounce phase. In our case, this directly corresponds to different values for the 400-1100 Hz SNR (Figure 10b) for $\beta_{\rm ic,b} \geq 0.10$, as well as slight differences in the tallest DWT-V blob's SNR (Figure 6a) for the same range of $\beta_{\rm ic,b}$ values. These deviations are useful in helping us estimate the degree of differential rotation $A$ for rapidly rotating cores ($\beta_{\rm ic,b} \geq 0.10$).



**(a)** *200-400 Hz SNR*



**(b)** *400-1100 Hz SNR*



**(c)** *Bounce Duration*

**Figure 10:** *Our three strain-based features of interest as functions of $\beta_{\rm ic,b}$ for all Abdikamalov waveforms, across all five differential rotations. We have chosen $D = 1$ kpc for our source's distance and have included $\pm 3\sigma$ statistical error bars that result from twenty different noise instantiations.*

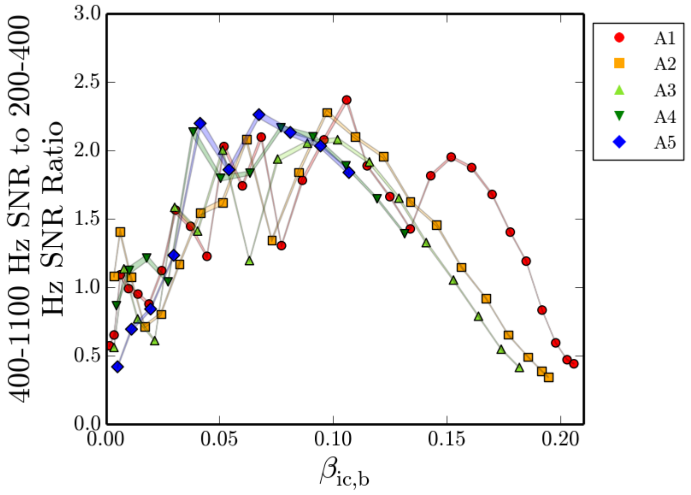


**Figure 11:** *The ratio of the 400-1100 Hz SNR to the 200-400 Hz SNR for waveforms from the Abdikamalov catalog, as a function of $\beta_{\rm ic,b}$. We assume $D = 1$ kpc for our sources, and have graphed $\pm 3\sigma$ statistical error bars that result from twenty different noise instantiations.*

Lastly, we note that our physical feature vector includes the SNRs of various frequency bands, and these SNRs scale inversely with the source's distance $D$. This can be problematic since the extrinsic parameter $D$ then becomes a confounding variable; multiple waveforms with different distances $D$ and different inner core dynamics could have similar SNRs. We can avoid this issue by taking the ratio of our two SNRs in Figure 10, as this would get rid of the scaling factor $D$. Unfortunately, as shown in Figure 11, the ratio of the 400-1100 Hz SNR to the 200-400 Hz SNR has a much more discontinuous appearance as a function of $\beta_{\rm ic,b}$. This appearance cannot be entirely accounted for by the statistical error bars due to separate noise instantiations, and might partly be a result of approximate neutrino transport schemes in the Abdikamalov simulations. Furthermore, if we take this SNR ratio, we end up losing the mostly monotonic nature of the 200-400 Hz SNR (as a function of $\beta_{\rm ic,b}$); this is problematic since monotonic features are preferable to work with when it comes to estimating $\beta_{\rm ic,b}$ via machine learning. Since we do not want to sacrifice the smooth and interpolable 400-1100 Hz SNR feature and the monotonic 200-400 Hz SNR feature, we treat the 200-400 Hz and 400-1100 Hz SNRs as separate features, and assume that $D$ will be approximately known through other astrophysical measurements (which is reasonable for a CCSN).

## IV. ML Algorithm Methods

Due to the nontrivial natures of our features in Figures 6 and 10, it is difficult to analytically propose a multivariate statistical regression (MVSR) model that can map our physical FV to a set of progenitor parameters. This complication makes it infeasible to use Markov Chain Monte Carlo (MCMC) regression methods, as we cannot easily construct an accurate multidimensional posterior distribution to sample from. Moreover, if we are interested in simultaneously determining other progenitor parameters – such as $J_{\rm ic,b}$ (the total angular momentum of the inner core at bounce) or $M_{\rm ic,b}$ (the total mass of the inner core at bounce) – that *do not scale linearly* with $\beta_{\rm ic,b}$ (as shown in Figure 2), a Bayesian approach would require us to have several different MVSR mappings from a given feature vector to each parameter. These complications are avoided by instead using ML methods.

### I. Algorithm Choice

A supervised ML regression algorithm works with a set of training data:

$$T_n = \{(X_i, Y_i)\}_{i=1}^{n}, \qquad (3)$$

where $X_i$ is a multidimensional feature vector, and $Y_i$ is a set of $V$ corresponding response variables (e.g. $\beta_{\rm ic,b}$) associated with that feature vector. This training step varies significantly from algorithm to algorithm. For a simple nearest-neighbors (NN) approach, this can involve the construction of a $k$-d tree that efficiently partitions $M$-dimensional feature space (in our physical FV case, $M = 7$), allowing us to identify nearby feature vectors from the training set. For a boosted decision tree algorithm, this can involve the construction of several boosted, poorly-performing decision trees; here, "boosting" refers to a sequential process where later decision trees focus on classifying training data that earlier decision trees incorrectly classified. Similarly, for random forests (RFs), this is just the construction of a forest of decision trees, where randomization occurs (1) when choosing the feature for each decision split, and (2) when choosing samples (with replacement) on which each tree is trained (a technique called "bootstrap aggregating" or "bagging"). A significantly more in-depth review of local learning, decision trees, and ensemble learning can be found in Chapters 13-15 of [16].

After "fitting" our ML regressor/classifier on training data, the algorithm then makes predictions about the response variable $\hat{Y}$ for a *test* feature vector $X$ that is not in the training set $T_n$. This procedure depends on the chosen algorithm, and so we considered three separate approaches to determine an optimal choice: $k$-nearest-neighbor ($k$-NN), random forests (RFs), and random subspaces of $k$-NN regressors (RS-$k$-NN).

The $k$-NN algorithm tends to be good at capturing local variation in data, as it only considers the $k$ nearest neighbors to $X$ and can employ inverse-distance-based weightings (where dimensions are standardized[1] to account for different units and scales). On the other hand, while RF regression can also be viewed as a weighted neighborhood scheme (as suggested in Section 1 of [17]), non-negligible weights might be given to very inaccurate response values $Y_i$ from our training set.

[1]We standardize feature values by subtracting off each feature's mean (averaged over the entire training set) and dividing by its standard deviation (over the entire training set). In the testing phase, feature values are standardized by the *same* feature means and standard deviations used for standardizing the training set.

This renders the RF approach slightly more susceptible to inaccurate response values.

At the same time, the RF approach has two main benefits over $k$-NN: it can account for correlations between response variables, as well as unbalanced training sets. With regards to correlations in predicting a set of $V$ variables, $k$-NN simply treats the $V$ response variables independently. Specifically, it just finds the closest $k$ feature vectors to $X$ in the training set and weights their response variables by Euclidean distance; thus, factoring out noise, we expect the same prediction results regardless of whether $k$-NN separately or simultaneously determined the $V$ variables. Even if the popular Mahalanobis distance metric is applied, only correlations between feature vector components can be accounted for, not those between response variables. On the other hand, when random forests deal with $V > 1$ response variables, they compute the average reduction in their quality criterion (e.g. mean square error) across all $V$ outputs to arrive at a splitting criterion for each node. If the output values for a given feature vector are correlated, looking at the average reduction in a single forest (instead of building independent forests) lets us take into account these correlations. This is advantageous for us since we know that $M_{\mathrm{ic,b}}$, $J_{\mathrm{ic,b}}$, and $\beta_{\mathrm{ic,b}}$ are correlated, and all three of these are dependent on the rotation profile and hence $\log_{10}(A)$.

Moreover, random forests have a more appropriate method of dealing with the issue of unbalanced classes. This occurs when the training data is (1) split unequally among the classes of interest, and (2) we have reason to believe that this splitting method is not representative of how real data will actually be partitioned. In the case of the Abdikamalov training catalog, we have a relative abundance of strongly differentially rotating models (e.g. $A1$) and a relative dearth of uniformly differentially rotating models (e.g. $A5$). This is problematic since it would make our algorithm relatively poor at inferring $\log_{10}(A)$ from uniformly rotating CCSN signals.

This discrepancy in waveform numbers occurs since, while the training models for each differential rotation are always separated by $\Delta\Omega_{\mathrm{c}} = 0.5$ rad/s, models with low to moderate differential rotation tend to *not* collapse at large values of $\Omega_{\mathrm{c}}$. Physically, this occurs since – with higher $\Omega_{\mathrm{c}}$ – uniformly and mildly differentially rotating models (e.g. $A3$-$A5$) have a larger amount of centrifugal support at the start of a simulation, and hence do not collapse [5]. But this does not necessarily mean that CCSNe from strongly differentially rotating progenitors will occur more frequently in nature, since stellar evolution might not favor such progenitors.

For the purposes of our ML approach, we want to assume a uniform prior and consider all differential rotations equally likely. From the random forest point of view, this is easily accomplished by re-weighting the priors used for each node's splitting criterion. In our case, we re-weight all of the waveforms of a given differential rotation $A$ by $\alpha_{\min}/\alpha$, where $\alpha$ is the number of training waveforms with differential rotation $A$ and $\alpha_{\min}$ is the lowest value of $\alpha$. On the other hand, $k$-NN cannot really tweak any priors; to account for unbalanced classes, one must either undersample the majority class or oversample the minority class [16]. Undersampling is infeasible since there are three times more $A1$ waveforms than $A5$ ones. Oversampling, on the other hand, involves synthesizing new feature vectors via the Synthetic Minority Oversampling Technique (SMOTE) [18], which randomly interpolates between the nearest same-class neighbors to a given waveform. While it is already problematic to deal with unphysical, synthesized data, SMOTE also has the drawback of only oversampling by integer factors; thus, true equality cannot be achieved.

To decide between RF and $k$-NN for our physical FV approach, we empirically evaluate the aforementioned trade-offs with the Python ML library `scikit-learn` [19]. Holding $D$ constant at 5 kpc, we find that $k$-NN[2] is noticeably less precise than balanced RF[3] in terms of the width of its $\pm 2\sigma$ prediction intervals for most progenitor parameters. Qualitative comparisons between the $\beta_{\mathrm{ic,b}}$ predictions of these two methodologies can be found in Appendix A. Due to RF's robustness to noise and aforementioned theoretical advantages, we therefore focus solely on RF-related results in Section V.

Lastly, we briefly consider the RS-$k$-NN approach since – while a single $k$-NN regressor might do reasonably well – NN algorithms tend to perform relatively poorly with very-high-dimensional data due to the so-called "curse of dimensionality" [16]. Thus, we apply the random subspace (RS) ensemble learning approach with 150 4-NN regressors trained on 40 randomly chosen training samples (with $D = 5$ kpc) and three randomly chosen features each. As shown in Figure 22 of Appendix A, however, the RS-4-NN method actually tends to undershoot more often at high $\beta_{\mathrm{ic,b}}$ values compared to the usual 4-NN approach, and is overall slightly less accurate. This seems to suggest that the curse of dimensionality is not a very significant issue for us. To the contrary, we *need* all seven feature vector dimensions to properly distinguish between training waveforms at all $\beta_{\mathrm{ic,b}}$ values, and randomness can often exclude useful monotonic features.

## II. RF Algorithm Implementation

To construct our regression algorithms, we use the `RandomForestRegressor` module from `scikit-learn`. For our forest parameters, we grow 150 decision trees (as RF does not overfit) with bootstrap aggregation ("bagging"), maximal depth (i.e. nodes expanded until all leaves are pure, where purity is measured by the mean-square-error (MSE) quality criterion), and consider all features at each decision split. To account for correlations, we simultaneously predict all four progenitor parameters – $\beta_{\mathrm{ic,b}}$, $J_{\mathrm{ic,b}}$, $M_{\mathrm{ic,b}}$, and $\log_{10}(A)$ – with a single random forest. Lastly, we re-weight our training waveforms' priors to account for unbalanced differential rotation classes, using the aforementioned procedure

[2]With an optimal case of $k = 4$, inverse-distance-based weightings, and 3-nearest-neighbors SMOTE-balancing for $\log_{10}(A)$ regression

[3]Implementation details for RF are outlined in the following subsection.

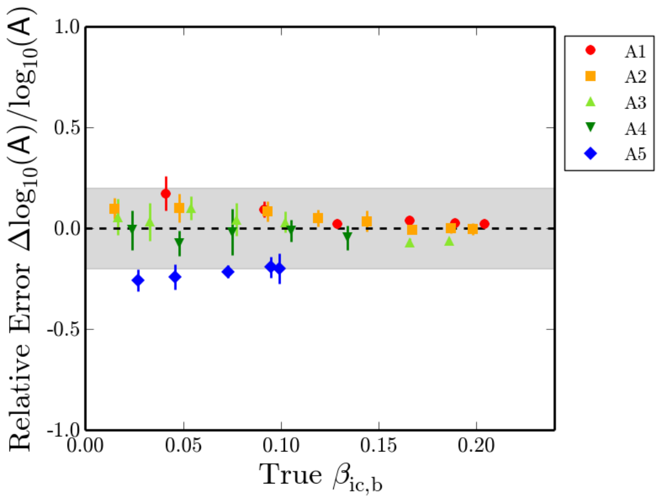


**Figure 12:** *Relative errors in* $\log_{10}(A)$ *prediction with our RF implementation and physical FV. We set* $D = 5$ *kpc and include* $\pm 2\sigma$ *statistical error bars. The gray region represents points within* $\pm 20\%$ *relative error.*

| Quantity | ARE @ 5 kpc (%) | ARE @ 10 kpc (%) |
|---|---|---|
| $\beta_{\rm ic,b}$ | 17.2 (1.2) | 16.1 (0.4) |
| $J_{\rm ic,b}$ | 7.0 (1.1) | 8.3 (0.7) |
| $M_{\rm ic,b}$ | 1.6 (0.1) | 1.7 (0.1) |
| $\log_{10}(A)$ | 10.8 (0.1) | 11.9 (0.1) |

**Table 1:** *Average relative errors (as percentages) of various progenitor parameter predictions with our RF approach, at both* $D = 5$ *kpc and* $D = 10$ *kpc. These results are averaged over three runs at each distance* $D$*. In parentheses, we display the (sample) standard deviation of each quantity over the three runs.*

of assigning a sample weight of $\alpha_{\rm min}/\alpha$ to each waveform's feature-vector/response-variable pair.

As for forming the seven-dimensional feature vectors $X_i$ for our training data, we use the mean values of all seven features for each training waveform, averaged over 20 instantiations of aLIGO ZDHP noise. We choose to use noisy data for our training procedure since this allows us to be realistic about how various features (e.g. bandwidths and central frequencies) are skewed by noise. Our RF regressor is then fitted on these feature vectors and their corresponding progenitor parameters, and a forest of decision trees is constructed.

With regards to our test data, we inject each waveform in the testing set (scaled to the same distance $D$ as the training set data) in 20 separate instantiations of aLIGO ZDHP noise in order to simulate actual GW signal measurements. For each test waveform, we then use the same image-processing and strain-based methods to construct our seven-dimensional feature vectors. Our RF algorithm then predicts progenitor parameters for each feature vector, resulting in twenty predictions for each test waveform. The averages and $\pm 2\sigma$ spreads of these predictions are shown in the following section.

## V. Physical FV ML Results

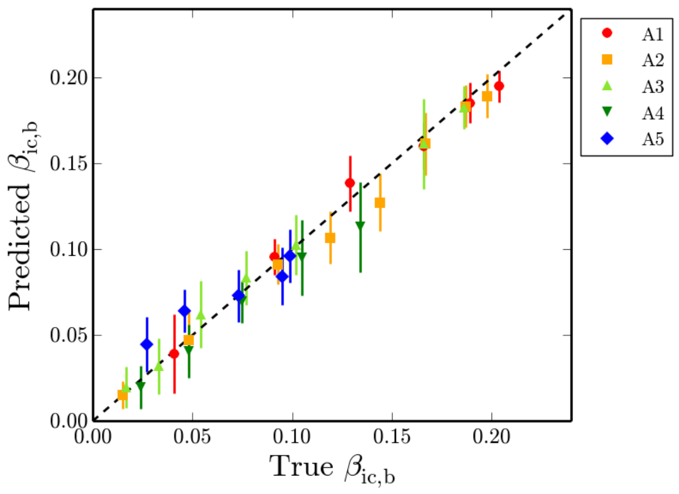
Following our aforementioned procedure with our physical FV and $D = 5$ kpc, we arrived at the results in Figures 12 and 13. Qualitatively speaking, we see that the predicted $\beta_{\rm ic,b}$ values are consistently close to the true $\beta_{\rm ic,b}$ values (i.e. they are not biased), with slightly lower discrepancies at very high values of $\beta_{\rm ic,b}$ ($\beta_{\rm ic,b} \gtrsim 0.17$). This trend makes sense since the bounce blob starts to become more prevalent (relative to noise) for larger $\beta_{\rm ic,b}$, and many of our image-based features focus on describing the parameters of this blob. As for the relative errors $\Delta\beta_{\rm ic,b}/\beta_{\rm ic,b}$ in our predictions, we see that these tend to be within $\pm 20\%$, with more accurate predictions at larger values of $\beta_{\rm ic,b}$.

The mean prediction relative errors from our $J_{\rm ic,b}$ regression tend to remain within $\pm 10\%$, which suggests that our algorithm does a better job at regressing on $J_{\rm ic,b}$ than it does on $\beta_{\rm ic,b}$. Physically, this may result from our chosen features correlating more closely with the total angular momentum of the inner core at bounce than with $\beta = T/|W|$. Furthermore, our $M_{\rm ic,b}$ prediction tends to be accurate within $\pm 5\%$ for most values of the true $M_{\rm ic,b}$. This can probably be attributed to how densely sampled our training data models are in $M_{\rm ic,b}$ space, as shown in Figure 2.

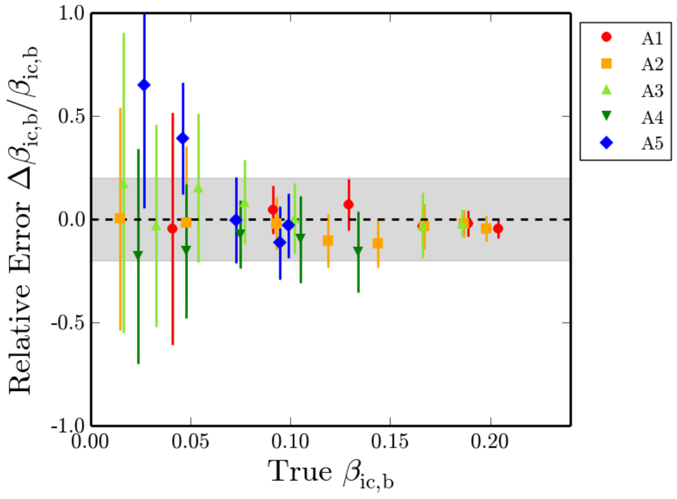
As for our differential rotation predictions, the FV approach performs more accurately with test waveforms with $\beta_{\rm ic,b} \gtrsim 0.10$. If we look at Figure 10b, this makes sense since the 400-1100 Hz SNR is the only feature in our vector that really distinguishes between the differential rotations (holding $\beta_{\rm ic,b}$ constant) – and that too only for $\beta_{\rm ic,b} \gtrsim 0.10$. We also note that, while the $\log_{10}(A)$ predictions seem to have low relative errors, the true errors in $A$ are understated by the logarithmic scale. Thus, even small relative errors can correspond to a misclassification. In order to generally improve the performance of our differential rotation predictions, future simulations will have to span the $\log_{10}(A)$ space more evenly and completely. After all, it is simply infeasible to precisely interpolate between five discrete values with random forests. This is especially an issue with the uniformly-rotating $A5$ waveforms and their extremely high $A$ values.

As a more quantitative measure of overall performance, we take the mean predictions of each quantity (averaged over all instantiations of a waveform) and take the square root of the mean-square relative errors (averaged over all test waveforms) of these mean predictions; for the remainder of this report, this metric is referred to as the "average relative error" or ARE. We average these AREs over three runs of our algorithm at each distance, where each run looks at 20 separate noise instantiations per test waveform. Our results are tabulated in Table 1, where we consider both $D = 5$ kpc and $D = 10$ kpc.

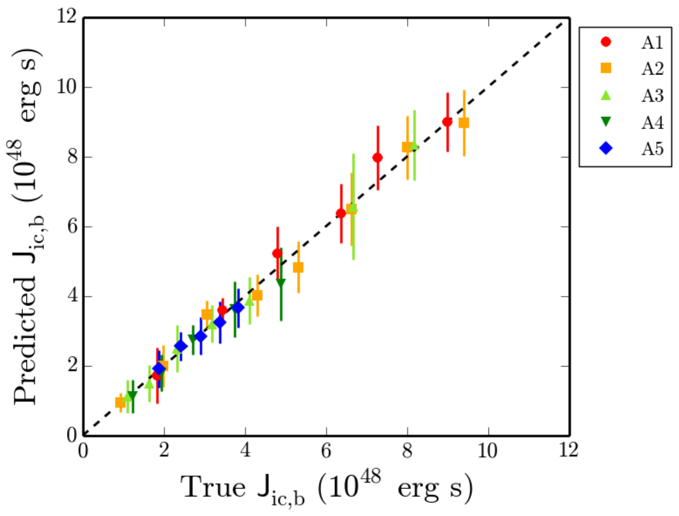
While these quantitative results for $D = 10$ kpc seem promising, it is important to note that the ARE only looks



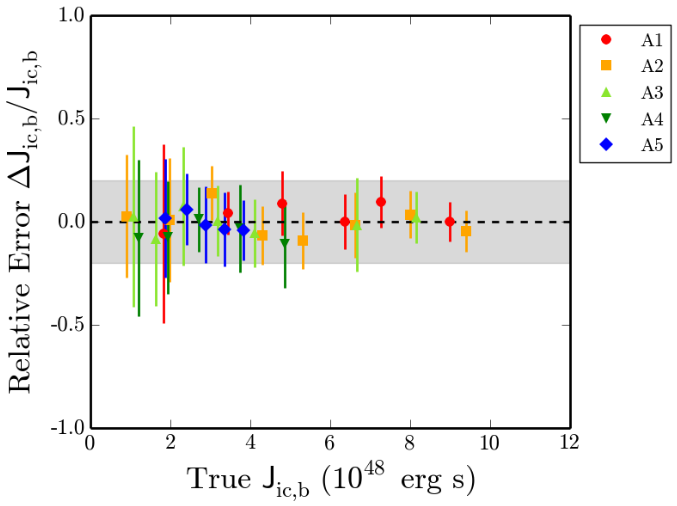
**(a)** $\beta_{\rm ic,b}$ *Predictions at 5 kpc*



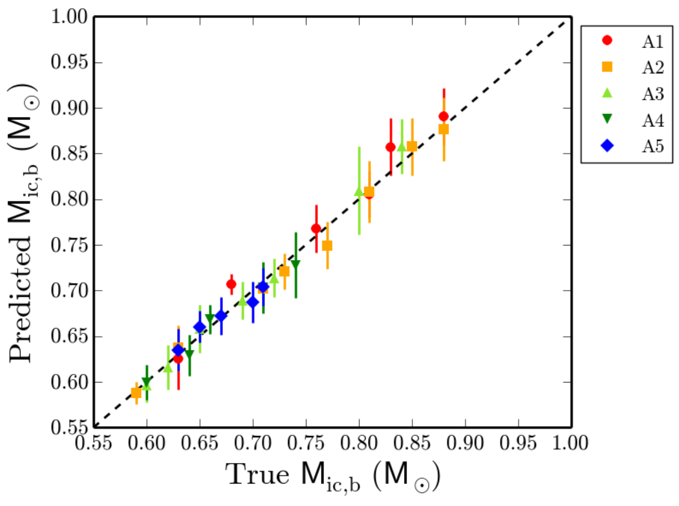
**(b)** $\beta_{\rm ic,b}$ *Relative Errors at 5 kpc*



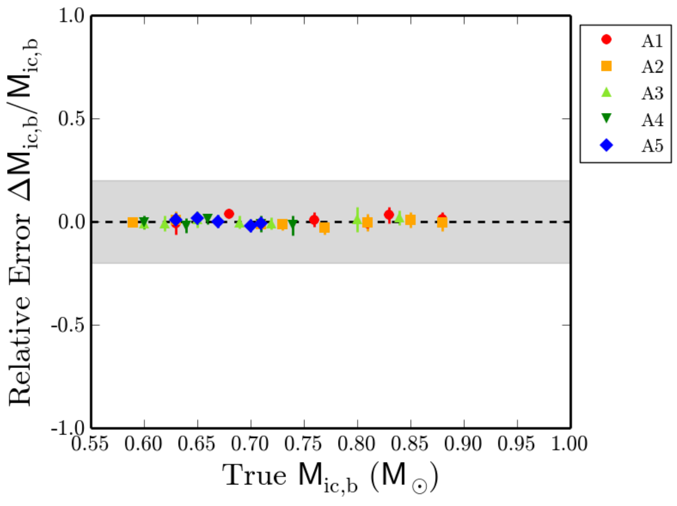
**(c)** $J_{\rm ic,b}$ *Predictions at 5 kpc*



**(d)** $J_{\rm ic,b}$ *Relative Errors at 5 kpc*



**(e)** $M_{\rm ic,b}$ *Predictions at 5 kpc*



**(f)** $M_{\rm ic,b}$ *Relative Errors at 5 kpc*

**Figure 13:** *Predictions and relative errors for continuous progenitor parameters with our RF implementation and physical FV. We have chosen $D = 5$ kpc for our source's distance and include $\pm 2\sigma$ statistical error bars that result from 20 different noise instantiations of the test waveforms. The gray region represents points within $\pm 20\%$ relative error.*

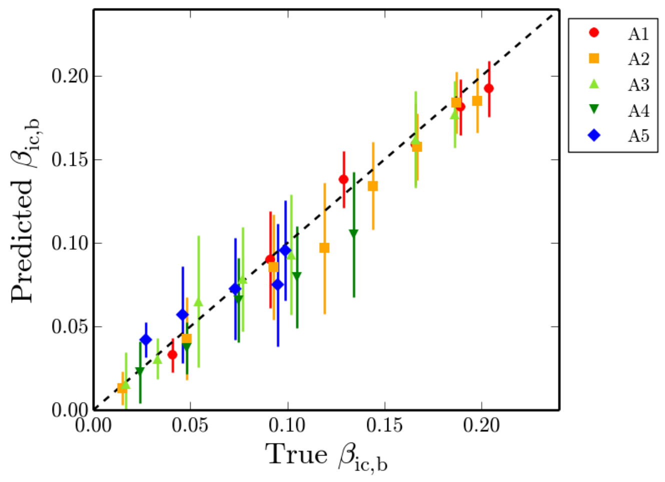


**Figure 14:** *Predicted $\beta_{\rm ic,b}$ values (with $\pm 2\sigma$ statistical error bars) for our RF implementation with physical FVs at $D = 10$ kpc.*

at the relative error of the *mean predicted* value (averaged over 20 noise instantiations). In other words, it does not give us a sense of the spread of our predicted values due to different noise instantiations, and this spread tends to increase with $D$. This increase can be seen by looking at Figure 14, where we set $D = 10$ kpc and predict $\beta_{\rm ic,b}$. At this distance, our predictions retain their levels of accuracy (on average), but are only precise in the case of rapidly-rotating models ($\beta_{\rm ic,b} \gtrsim 0.17$), where the bounce blob is more prominent and where some trends (e.g. changes in the bounce duration) become more extreme. Statistically, the imprecise performance of our algorithm at 10 kpc makes sense since many of the trends in our feature vector data can become obscured by detector noise as our signal gets weaker. Other progenitor parameter predictions show similar decreases in accuracy and increases in statistical error when we increase $D$ to 10 kpc.

To determine how our algorithm's performance varies with $D$, we graphed our predictions' AREs as a function of $D$ for six source distances from 5 to 10 kpc. We also graph our average $\beta_{\rm ic,b}$ prediction imprecision as a function of $D$; this latter quantity is measured as the average full width of our $\pm 2\sigma$ prediction intervals for $\beta_{\rm ic,b}$ over all 31 test waveforms. Our results are shown in Figure 15; for each distance, we have averaged results over three runs of our algorithm.

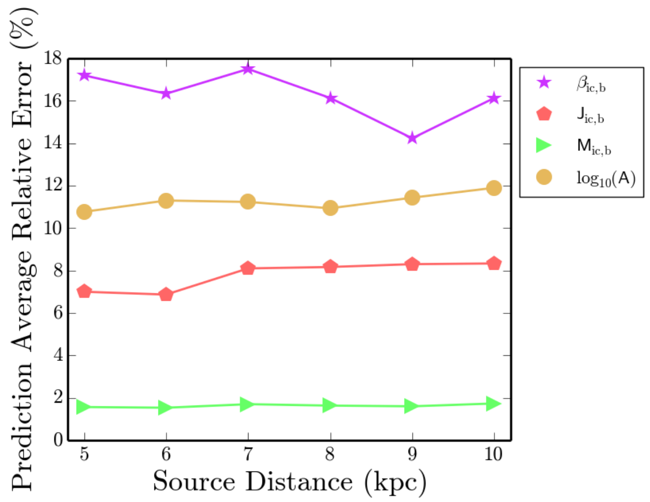
Figure 15a shows that there is no significant trend in the ARE of most of our quantities' predictions as a function of source distance $D$. While this seems counter-intuitive, it is important to remember that our algorithm uses a different set of training data for each distance $D$, in order to account for how feature trends appear different with weaker signals (e.g. the SNRs inversely scale with $D$). Since our chosen physical features still remain mostly interpolable in our training data, it therefore makes sense that our noisy test data would yield accurate results *on average*, even as $D$ increases. However, the spreading effect of noise becomes clear in Figure 15b, as our average $\beta_{\rm ic,b}$ imprecision worsens from $\sim 0.03$ to $\sim 0.05$; similar trends apply to the prediction imprecisions of other progenitor parameters.

| Feature | Importance |
|---|---|
| STFT's Tallest DWT-V Blob's SNR | 0.069 |
| STFT's Tallest DWT-V Blob's Bandwidth | 0.033 |
| STFT's Tallest Regular Blob's Bandwidth | 0.063 |
| STFT's Tallest DWT-V Blob's Central Frequency | 0.030 |
| 200-400 Hz SNR | 0.538 |
| 400-1100 Hz SNR | 0.109 |
| Bounce Duration | 0.157 |

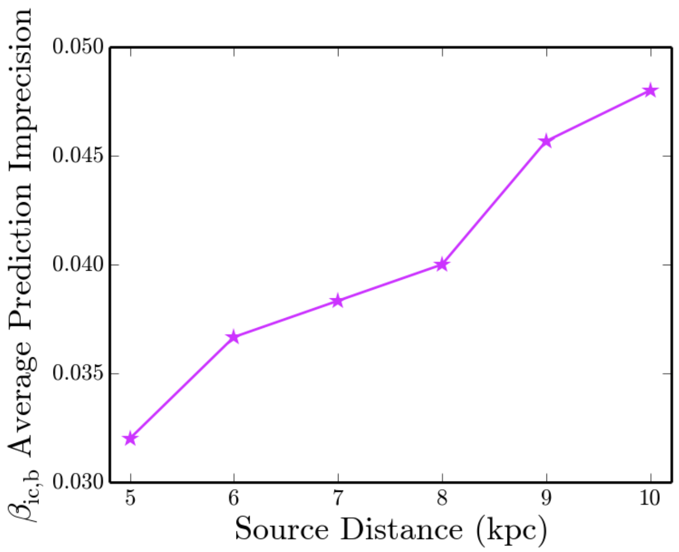
**Table 2:** *Feature importances computed by* `scikit-learn` *for our RF algorithm. These values are based on training data with $D = 5$ kpc, and are only meant to give a qualitative idea of importances; due to randomness, they will fluctuate slightly from forest to forest.*

As a final point of interest, we record in Table 2 the feature importances produced by `scikit-learn` for our RF algorithm with $D = 5$ kpc. A feature's "importance" in this sense is directly related to the decision trees constructed for a set of training data. One can imagine treating the depth (in a tree) of a feature used as a decision node as a measure of the relative importance of that feature when it comes to predicting the response variable; features near the top of a tree tend to contribute to the final prediction result for a much larger fraction of test data. Following this line of reasoning, a feature's importance in `scikit-learn` is defined as the expected fraction of samples whose classifications are aided by that feature [20]. These importance values are normalized so as to sum to unity.

Table 2 suggests that, in determining progenitor parameters via random forests, the 200-400 Hz SNR, 400-1100 Hz SNR, and bounce duration play the most important roles when it comes to decision splits. This can be understood by looking at the relatively non-fluctuating relationships between these features and $\beta_{\rm ic,b}$ in Figure 10. Moreover, since the 200-400 Hz SNR has a mostly monotonic dependence on $\beta_{\rm ic,b}$, it makes sense that its importance would be particularly high; this also explains the relatively high importance of the monotonically increasing bounce duration. Spectrogram-based (i.e. bounce-blob-related) features, on the other hand, tend to have lower feature importances due to their fluctuations and mostly nonmonotonic behaviors.



**(a)** *Average Relative Error*



**(b)** $\beta_{\mathrm{ic,b}}$ *Average Prediction Imprecision*

**Figure 15:** *ARE in all quantity predictions and imprecision of* $\beta_{\mathrm{ic,b}}$ *predictions as a function of source distance, using our RF algorithm with physical FVs. Values in this figure are averaged over three runs at each distance.*

## VI. Alternative FVs and Results

In addition to our physical feature vectors, we also considered two other spectrogram-based FV approaches: principal component analysis (PCA) and the Scale-Invariant Feature Transform (SIFT).

### I. Spectrogram-Based PCA FVs

Consider an $m \times n$ matrix $A$ comprised of $m$ one-dimensional waveforms (the matrix's rows) that are all $n$ samples long. Performing PCA on this data set amounts to constructing a singular value decomposition (SVD) of $A$, which in turn involves factoring $A$ as

$$A = U\Sigma V^T, \tag{4}$$

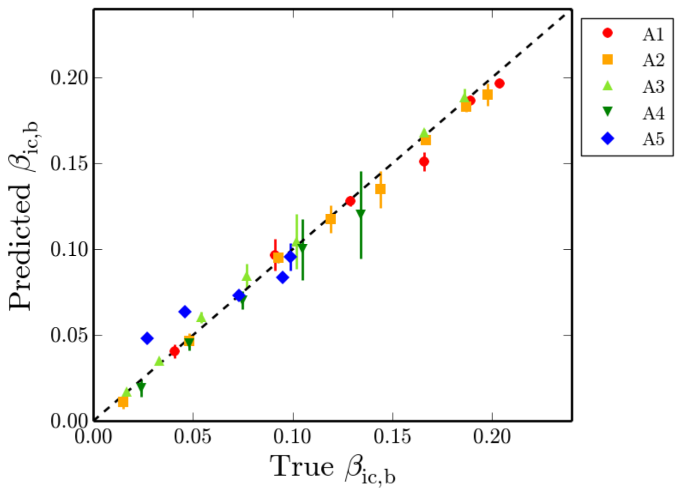
where $U$ is a unitary $m \times m$ matrix whose columns correspond to the eigenvectors of $AA^T$, $V$ is a unitary $n \times n$ matrix whose columns correspond to the eigenvectors of $A^TA$, and $\Sigma$ is an $m \times n$ matrix whose diagonal elements correspond to the square root of the corresponding eigenvalues. The eigenvectors in $V$ form an orthonormal basis that spans the catalog of waveforms used to construct $A$. These eigenvectors are our principal components (PCs); typically, they are ranked by their corresponding eigenvalues, so that the first PC consists of the most common features in the waveforms. The most significant $p \leq \min(m, n)$ PCs are then selected, and can be used to reconstruct an arbitrary one-dimensional waveform $h$ in terms of just $p$ components via projection.

While PCA has been applied in other CCSN analysis approaches for model selection [21] and parameter estimation [8], these approaches have typically focused on one-dimensional strain signals. In this section, we look into employing PCA with spectrograms instead. First, we ensure that our strain signals are *aligned* as mentioned in Section II, so that our resulting PCs are temporally coherent. We then take STFT spectrograms of each *noiseless* waveform in our training set (scaled to a chosen distance $D$), using the same settings mentioned earlier. In order to use more convenient dimensionless units for each STFT spectrogram, we multiply each bin's power content by $4.0/S_h(f)$, where $S_h(f)$ is an analytic approximation to the aLIGO ZDHP PSD as stated in Equation 4.7 of [22]. An analytic PSD is used to create a smooth image unaffected by resonances.

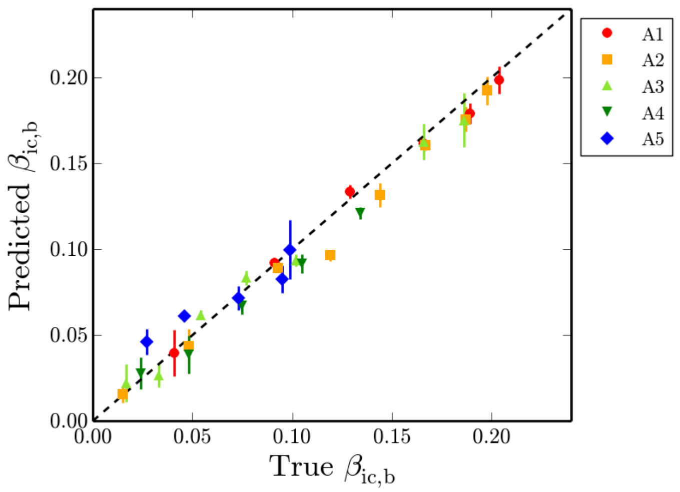
We then flatten each waveform's spectrogram (in row-major order) into a one-dimensional array, which we append onto our matrix $A$. To perform PCA on the noiseless spectrograms in $A$, we use `scikit-learn`'s randomized PCA algorithm [23] with $p$ PCs; this randomized approach has a faster computation time than classic PCA's $\mathcal{O}(n^3)$ cost (where $n$ is the number of points in each flattened spectrogram).

We can then PCA transform noisy training waveforms (with twenty instantiations per waveform), in order to represent them in terms of just $p$ components; this simply involves a projection of each waveform onto the orthonormal basis of our $p$ PCs. If we average these projections over all noise instantiations, this then gives us a $p$-dimensional feature vector for each training waveform, which can be used with an ML algorithm to predict progenitor parameters.

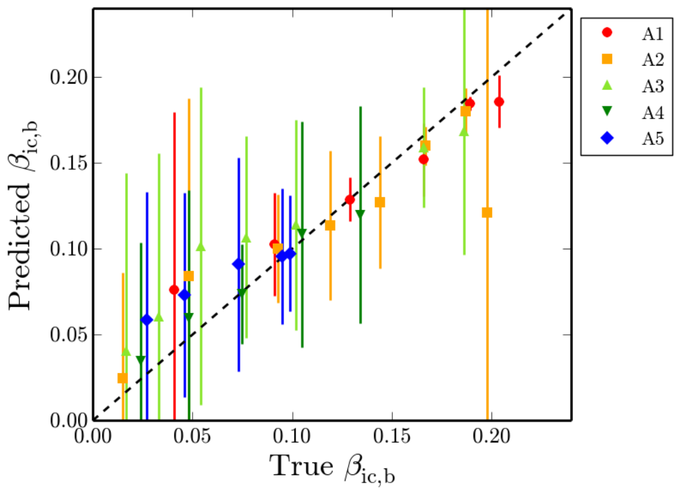
We once again use random forests for our ML procedure, with the same implementation details as in Section IV. In this case, RF is preferable to $k$-NN since the curse of dimensionality (for large $p$) can negatively impact NN algorithms, especially when the $p$ dimensions are of unequal importance as they are with PCs. For our test data, we take noisy STFT spectrograms (20 instantiations per waveform again), scale



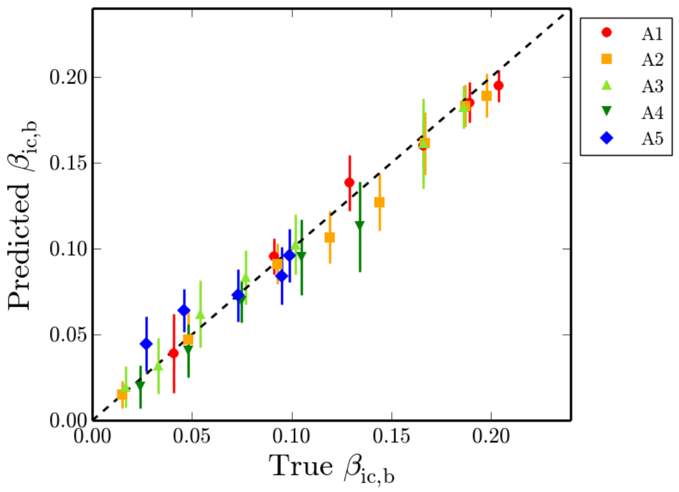
**(a)** *PCA-Based 50-Component FV at 1 kpc*



**(b)** *Physical 7-Dimensional FV at 1 kpc*



**(c)** *PCA-Based 50-Component FV at 5 kpc*



**(d)** *Physical 7-Dimensional FV at 5 kpc*

**Figure 16:** *Predicted $\beta_{\rm ic,b}$ vs true $\beta_{\rm ic,b}$, using both PCA-based FVs and physical FVs. We include results for both $D = 1$ kpc and $D = 5$ kpc.*

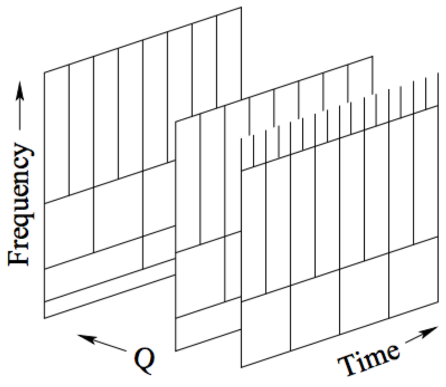


**Figure 17:** *The tiling of time-frequency-Q space in a QT spectrogram. The tiles are spaced linearly in time and logarithmically in frequency and Q. Figure adapted from [25].*

them by $4.0/S_h(f)$, flatten them into one dimension, and PCA transform to get a $p$-dimensional feature vector. Our test waveforms must also be aligned as before, since PCA relies on temporal coherence (in our case, a *known bounce time*).

From a qualitative point of view, we found that more PC components generally led to a better performance in predicting $\beta_{\mathrm{ic,b}}$. In the end, we decided on $p = 50$ since we found that our random forest's feature importances were on the order of $10^{-4}$ when additional PCs were added.

Our results for $\beta_{\mathrm{ic,b}}$ prediction with this ML procedure at $D = 1$ kpc and $D = 5$ kpc are shown in Figure 16, placed side-by-side with the results from our physical FV approach (similar results qualitatively apply for other progenitor parameters). We can see that, at $D = 1$ kpc, the PCA approach actually performs better in terms of both precision and accuracy. However, when we increase $D$ to 5 kpc, PCA becomes less accurate and significantly less precise. This poor performance might result from PCA keying in on noise or other features (e.g. convection blobs) that are not strongly correlated with our chosen progenitor parameters. It might also result from the loss of phase information and suboptimal time-frequency resolutions in spectrograms, which would suggest that PCA is not entirely suited for spectrogram-only use.

## II. SIFT-Based FVs

In the image-processing community, David Lowe's scale-invariant feature transform (SIFT) is often used in conjunction with the "visual bag-of-words" technique to categorize and (later) recognize real-life objects. SIFT uses a difference-of-Gaussians (DoG) pyramid approach at various image scales to identify the high-contrast parts (e.g. edges or corners) of an image. *Keypoints* are identified as scale-space extrema on each of these DoG pyramids. In order to describe these keypoints in an orientation-independent manner, the SIFT algorithm determines the overall orientation of a keypoint based on its pixels' gradient magnitudes and orientations. The algorithm then uses a 128-dimensional *keypoint descriptor* to

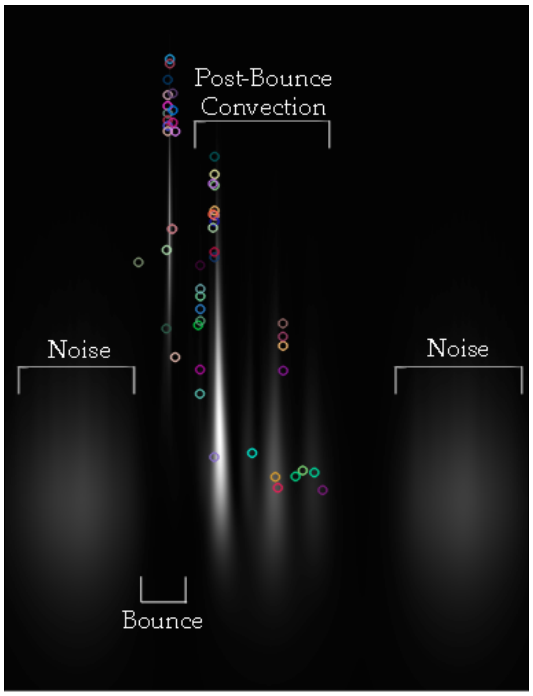
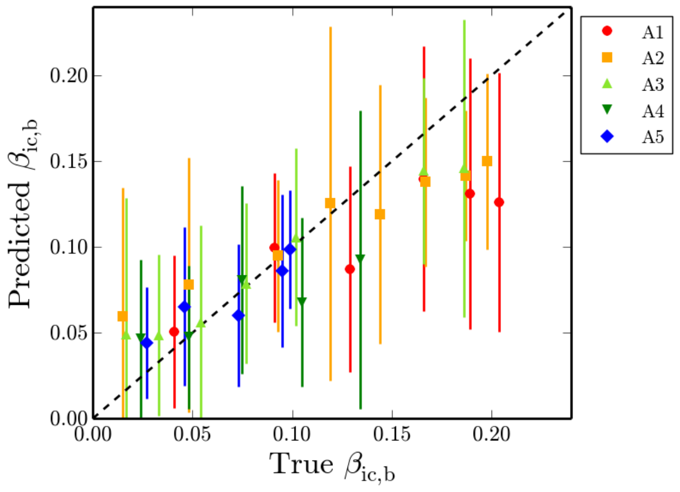


**Figure 18:** *Grayscale QT spectrogram of noisy A1O10 waveform at 1 kpc, with SIFT keypoints overlaid as colored circles. Axes are logarithmically scaled in frequency (vertical axis) and linearly scaled in time (horizontal axis). Note that SIFT keys in on the bounce and convection blobs, but ignores noise blobs.*

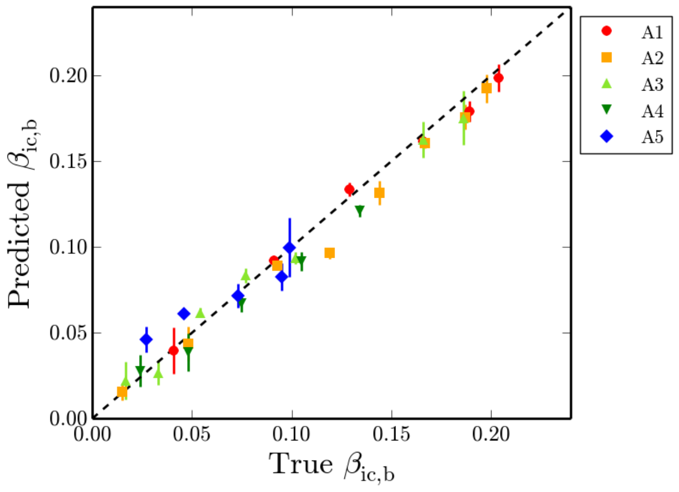
store information about keypoint pixels' gradients relative to this overall orientation. The interested reader should refer to Lowe's original paper [24] for further information on SIFT.

Due to the STFT's poor time resolution, we chose to use Q-Transform (QT) spectrograms instead of STFT spectrograms. QT spectrograms differ from conventional STFT spectrograms in that they use non-orthogonal sine-Gaussian functions with differing quality factors $Q$ as their basis, as opposed to the orthogonal family of trigonometric functions used in the STFT. Unlike the STFT, QTs do not have a rectangular time-frequency tiling. Instead, they have a higher time resolution at high frequencies, and hence a lower frequency resolution at those frequencies (due to the Gabor limit). For this reason, they tend to be logarithmically tiled in frequency space. This tiling can be seen visually in Figure 17.

To bring out convection blobs more (as these tend to be at low frequencies) and achieve a finer time resolution, we use logarithmically-spaced frequencies, time bins that are 1/2048 s long, and $Q = 0.2$ as parameters for `astroML`'s `wavelet_PSD` function [26]. We then convert our QT spectrogram into strain$^2$ units by multiplying its powers by $4.0/S_h(f)$. To avoid resonances, which can create edges in our spectrogram, we use the analytic expression for $S_h(f)$ as before. Our spectrogram is then normalized so that the brightest pixel has an intensity of 255.



**(a)** *SIFT-Based 80-Component FV at 1 kpc*



**(b)** *Physical 7-Dimensional FV at 1 kpc*

**Figure 19:** *Predicted $\beta_{\rm ic,b}$ vs true $\beta_{\rm ic,b}$, using both SIFT-based FVs and physical FVs. We have chosen $D = 1$ kpc for our source's distance and have included $\pm 2\sigma$ statistical error bars that result from 20 different noise instantiations of the test waveforms.*

We then apply `OpenCV`'s SIFT algorithm [27] to detect keypoints. In terms of our SIFT parameters, we use four layers in each octave (i.e. blur levels for each Gaussian pyramid); a contrast threshold of 0.005 to keep low-contrast keypoints; an edge threshold of 80 to keep edge-related keypoints; $\sigma = 0.5$ for our Gaussian kernel's standard deviation at octave 0; and a maximum of 50 keypoints per image. These parameters allow SIFT to highlight keypoints related to the bounce and convection blobs and ignore most noise-related blobs, as shown in Figure 18 for a noisy A1O10 waveform at 1 kpc.

In order to actually turn the keypoints from each training waveform into a feature vector, we need to use the "visual bag-of-words" technique. To do this, we use twenty noise instantiations per training waveform and create an array of all of the SIFT descriptors found in these instantiations. We then group these 128-dimensional SIFT descriptors into 80 clusters using `scikit-learn`'s $k$-means clustering algorithm with $k = 80$ and default parameters. The centers of these clusters form a "visual vocabulary" of 80 words that describe the types of features in our training set; for consistency, we order our cluster centers by their first dimension.

For each noise instantiation in the training set, we create a histogram that is 80 bins long. Then, for each SIFT keypoint found for this instantiation's QT spectrogram, we find the nearest neighbor to it from our visual vocabulary and add 1 to the corresponding cluster's bin. Each histogram is then normalized by dividing by the square root of its summed components. For each training waveform, we then have twenty histograms – one for each noise instantiation. These are averaged and then renormalized to give an 80-dimensional feature vector for each waveform.

In our testing phase, we use the same RF ML algorithm (due to high dimensionality) and create QT spectrograms for each of twenty noise instantiations per test waveform. The SIFT keypoints from each spectrogram are used to construct histograms based on nearest-neighbor binning as before, and these histograms are normalized to serve as feature vectors.

Our results for $\beta_{\rm ic,b}$ prediction at $D = 1$ kpc with SIFT are shown in Figure 19, placed side-by-side with the results from our physical FV approach at the same distance. While the predicted $\beta_{\rm ic,b}$ does generally seem to increase with the true $\beta_{\rm ic,b}$ for $\beta_{\rm ic,b} \lesssim 0.15$, our SIFT-based algorithm is unable to accurately predict $\beta_{\rm ic,b}$ for rapidly-rotating waveforms. This might result from a relative dearth of keypoints in rapidly-rotating models' spectrograms, since the bounce blob tends to be much more prominent than any of the convection blobs.

Compared to our physical feature vector construction, we can also see that the SIFT/"visual bag-of-words" approach is far less accurate and precise at predicting $\beta_{\rm ic,b}$, even at the relatively close distance of $D = 1$ kpc. This poor performance occurs for many reasons. For example, SIFT misses out on the context of keypoints (e.g. relative positions) and is unable to stitch them together in any meaningful way. In addition, SIFT is mostly useful for recognizing starkly different objects (e.g. cars vs bikes) in real-world images, and cannot be expected to perform well with objects that morph as slowly as our bounce blob. Lastly, it's possible that SIFT is thrown off by convection-blob-related keypoints that do not vary continuously with $\beta_{\rm ic,b}$ or other progenitor parameters – and hence cannot be reasonably "interpolated."

## VII. Conclusions

We have presented multiple pattern-recognition/ML approaches for inferring the progenitor parameters of rotating CCSNe from GW strain data. Our best-performing algorithm took advantage of waveform morphology trends in both the spectrogram and time domains, and used a combination of image processing and signal processing techniques to extract features buried in aLIGO noise. By combining these features into a seven-dimensional feature vector and applying a random forest machine-learning algorithm, we were able to simultaneously estimate various progenitor parameters for a given CCSN. Our predictions were accurate *on average* at both $D = 5$ kpc and $D = 10$ kpc, with average relative errors (for the mean prediction) remaining within $\pm 20\%$ for $\beta_{\mathrm{ic,b}}$, $\pm 10\%$ for $J_{\mathrm{ic,b}}$, and $\pm 5\%$ for $M_{\mathrm{ic,b}}$. However, the spread of our predictions increased noticeably with $D$; for instance, the average full width of $\pm 2\sigma$ $\beta_{\mathrm{ic,b}}$ prediction intervals increased from $\sim 0.03$ at 5 kpc to $\sim 0.05$ at 10 kpc.

When it comes to determining differential rotations, our physical FV approach only really performed accurately for $\beta_{\mathrm{ic,b}} \gtrsim 0.10$. This was the point where slight differences in waveform morphology started to emerge between models with the same $\beta_{\mathrm{ic,b}}$ and different differential rotations. Even though our procedure performed more accurately with rapidly-rotating models, it was still difficult to interpolate between the five discrete $\log_{10}(A)$ values presented in the Abdikamalov catalog. Future simulations will have to cover $\log_{10}(A)$ space more evenly and completely if differential rotations are to be numerically predicted from actual signals – especially in the case of near-uniform rotation.

To explore additional image-processing approaches to FV construction, we also looked into both spectrogram-based PCA and SIFT. The SIFT-based "visual bag-of-words" technique was found to be very inaccurate and imprecise at $D = 1$ kpc. On the other hand, spectrogram-based PCA worked very well at $D = 1$ kpc, although its predictions became noticeably less accurate and precise at $D = 5$ kpc – especially in comparison to our physical FV approach. Thus, it might be worthwhile to use PCA at small source distances $D$, especially since – at these distances – the diminished impact of noise makes it easier to determine the signal's arrival time.

With regards to future work, the pattern-recognition and ML algorithms that we've described in this paper can be reused to train and test on new rotating core-collapse models as they become available. By training on newer and presumably more accurate data, our algorithm will become better-equipped at inferring parameters from actual rotating CCSN GW signals buried in aLIGO noise.

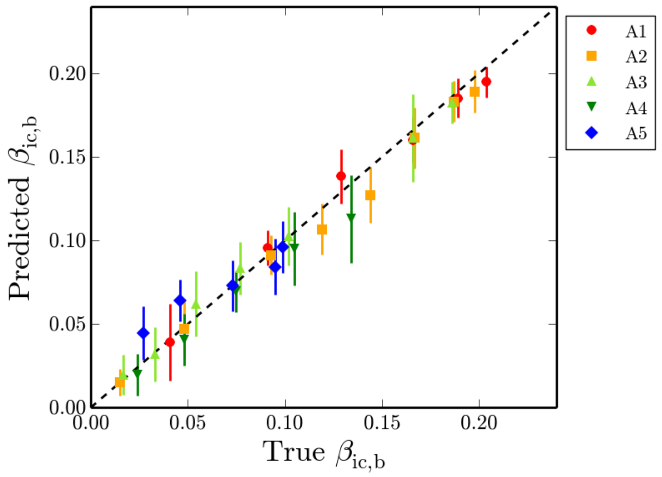
## VIII. Acknowledgments

I would like to thank my mentors Alan Weinstein and Sarah Gossan for their support, discussions, comments, and suggested references over the course of my Summer Undergraduate Research Fellowship (SURF) at Caltech. I would also like to acknowledge the NSF for funding my research.

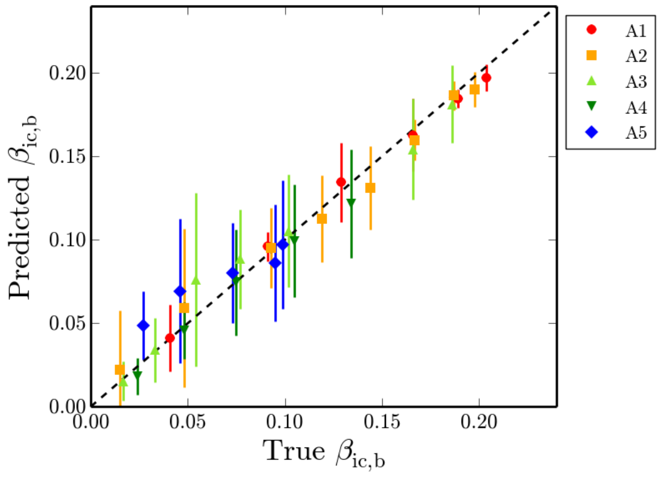
## A. Comparisons of ML Algorithms on Physical FVs

### I. RF vs 4-NN

Qualitatively, the 4-NN algorithm is much less precise than RF in its predictions at both $D = 5$ kpc and $D = 10$ kpc, as shown in Figures 20 (5 kpc) and 21 (10 kpc).

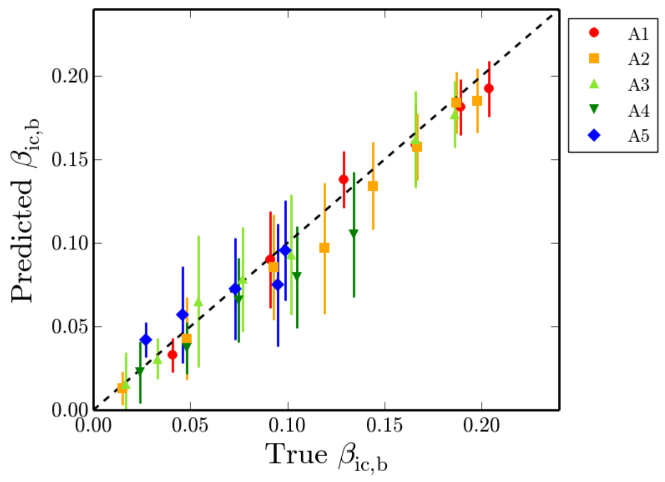


**(a)** *RF $\beta_{\mathrm{ic,b}}$ Predictions at 5 kpc*

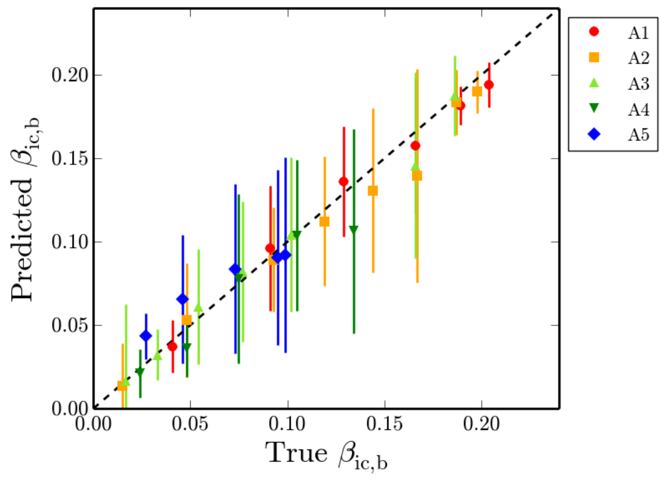


**(b)** 4-*NN $\beta_{\mathrm{ic,b}}$ Predictions at 5 kpc*

**Figure 20:** *Predicted $\beta_{\mathrm{ic,b}}$ vs true $\beta_{\mathrm{ic,b}}$, using both 4-NN and RF regressions. We have chosen $D = 5$ kpc for our source's distance and have included $\pm 2\sigma$ statistical error bars that result from 20 different noise instantiations of the test waveforms.*



**(a)** *RF $\beta_{\mathrm{ic,b}}$ Predictions at 10 kpc*

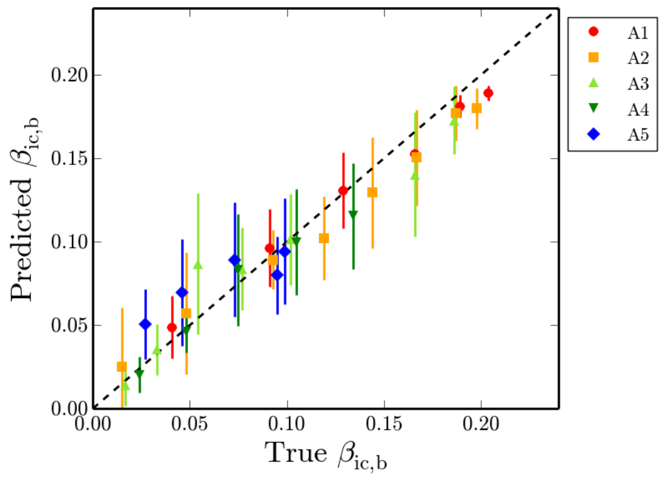


**(b)** *4-NN $\beta_{\mathrm{ic,b}}$ Predictions at 10 kpc*

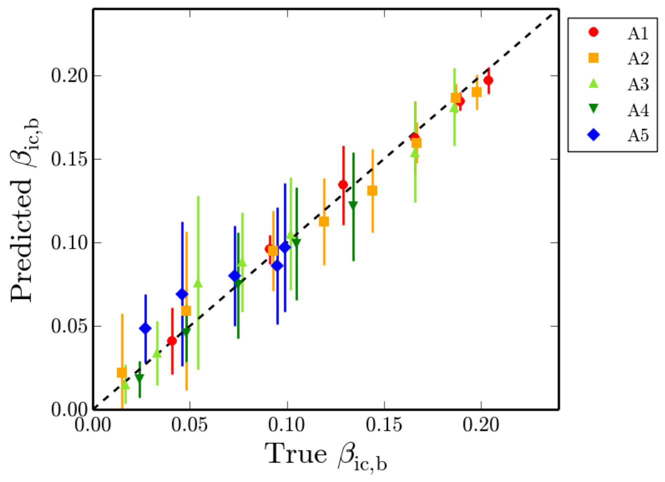
**Figure 21:** *The same comparison as in Figure 20, but with $D = 10$ kpc.*

## II. RS-4-NN vs 4-NN

A qualitative comparison between 4-NN and RS-4-NN is shown in Figure 22, where we predict $\beta_{\mathrm{ic,b}}$ with $D = 5$ kpc. Both algorithms perform similarly, although RS-4-NN tends to be less accurate at predicting $\beta_{\mathrm{ic,b}}$, especially in the case of rapidly rotating models (specifically, for $\beta_{\mathrm{ic,b}} \gtrsim 0.15$).



**(a)** *RS-4-NN $\beta_{\mathrm{ic,b}}$ Predictions*



**(b)** *4-NN $\beta_{\mathrm{ic,b}}$ Predictions*

**Figure 22:** *Predicted $\beta_{\mathrm{ic,b}}$ vs true $\beta_{\mathrm{ic,b}}$, using both RS-4-NN and 4-NN regressions. We have chosen $D = 5$ kpc for our source's distance and have included $\pm 2\sigma$ statistical error bars that result from 20 different noise instantiations of the test waveforms.*

## References

[1] Janka HT, Langanke K, Marek A, Martínez-Pinedo G, and Müller B (2007). "Theory of Core-Collapse Supernovae." *Physics Reports*, vol. 442, pp. 38-74. doi:10.1016/j.physrep.2007.02.002.

[2] Ott CD (2009). "Topical Review: The Gravitational-Wave Signature of Core-Collapse Supernovae." *Classical and Quantum Gravity*, vol. 26. doi:10.1088/0264-9381/26/6/063001.

[3] Ott CD *et al* (2012). "Correlated Gravitational Wave and Neutrino Signals from General-Relativistic Rapidly Rotating Iron Core Collapse." *Physical Review D*, vol. 86. doi:10.1103/PhysRevD.86.024026.

[4] Adams SM, Kochanek CS, Beacom JF, Vagins MR, and Stanek KZ (2013). "Observing the Next Galactic Supernova." *Astrophysical Journal*, vol. 778, pp. 164-178. doi:10.1088/0004-637X/778/2/164.

[5] Abdikamalov E, Gossan S, DeMaio AM, and Ott CD (2013). "Measuring the Angular Momentum Distribution in Core-Collapse Supernova Progenitors with Gravitational Waves." *ArXiv e-prints*. doi:2013arXiv1311.3678A.

[6] Dimmelmeier H, Ott CD, Marek A, and Janka HT (2008). "Gravitational Wave Burst Signal from Core Collapse of Rotating Stars." *Physical Review D*, vol. 78. doi:10.1103/PhysRevD.78.064056.

[7] Engels WJ, Frey R, and Ott CD (2014). "Multivariate Regression Analysis of Gravitational Waves from Rotating Core Collapse." *ArXiv e-prints*. arXiv:1406.1164.

[8] Edwards MC, Meyer R, and Christensen N (2014). "Bayesian Parameter Estimation of Core Collapse Supernovae Using Gravitational Wave Simulations." *ArXiv e-prints*. arXiv:1407.7549.

[9] `CoCoNuT` Code. Accessed June 2014. URL.

[10] Woosley SE and Heger A (2007). "Nucleosynthesis and Remnants in Massive Stars of Solar Metallicity." *Physics Reports*, vol. 442, pp. 269âĂŞ283. doi:10.1016/j.physrep.2007.02.009.

[11] Müller B, Janka HT, and Marek A (2013). "A New Multi-Dimensional General Relativistic Neutrino Hydrodynamics Code of Core-Collapse Supernovae." *The Astrophysical Journal*, vol. 766. doi:10.1088/0004-637X/766/1/43.

[12] Advanced LIGO Anticipated Sensitivity Curves. Accessed June 2014. URL.

[13] `OpenCV` 3.0.0-dev Documentation. Accessed May 2014. URL.

[14] Suzuki S and Abe K (1985). "Topological Structural Analysis of Digitized Binary Images by Border Following." *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 32âĂŞ46. doi:10.1016/0734-189X(85)90016-7.

[15] `PyWavelets` – Discrete Wavelet Transform in Python. Accessed June 2014. URL.

[16] Narsky I and Porter FC (2014). "Statistical Analysis Techniques in Particle Physics: Fits, Density Estimation and Supervised Learning." *Wiley-VCH*, ed. 1. ISBN 978-3-527-41086-6.

[17] Lin Y and Jeon Y (2002). "Random Forests and Adaptive Nearest Neighbors." *Journal of the American Statistical Association*. doi:10.1.1.153.9168.

[18] Chawla NV, Bowyer KW Bowyer, Hall LO, and Kegelmeyer WP (2002). "SMOTE: Synthetic Minority Over-Sampling Technique." *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357. doi:10.1613/jair.953.

[19] Pedregosa F *et al* (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830. URL.

[20] Ensemble Methods – `scikit-learn` 0.15.0 Documentation. Accessed July 2014. URL.

[21] Logue J, Ott CD, Heng IS, Kalmus P, and Scargill JHC (2012). "Inferring Core-Collapse Supernova Physics With Gravitational Waves." *Physical Review D*, vol. 86. doi:10.1103/PhysRevD.86.044023.

[22] Ajith P (2011). "Addressing the Spin Question in Gravitational-Wave Searches: Waveform Templates for Inspiralling Compact Binaries with Nonprecessing Spins." *Physical Review D*, vol. 84. doi:10.1103/PhysRevD.84.084037.

[23] Halko N, Martinsson P, Shkolnisky Y, and Tygert M (2011). "An Algorithm for the Principal Component Analysis of Large Data Sets." *SIAM Journal on Scientific Computing*, vol. 33, pp. 2580-2594. arXiv:1007.5510.

[24] Lowe DG (2004). "Distinctive Image Features from Scale-Invariant Keypoints." *International Journal of Computer Vision*, vol. 60, pp. 91-110. doi:10.1023/B:VISI.0000029664.99615.94.

[25] Chatterji, SK (2004). "The Q Transform Search for Gravitational-Wave Bursts With LIGO." *9th Gravitational Wave Data Analysis Workshop*. LIGO-G040521-00-Z.

[26] `astroML`: Machine Learning and Data Mining for Astronomy. Accessed July 2014. URL.

[27] OpenCV SIFT. Accessed June 2014. URL.