STAMATIS VRETINARIS

DATA ANALYSIS OF GRAVITATIONAL WAVES FROM BINARY NEUTRON STARS AND MACHINE LEARNING APPLICATIONS

M.Sc Computational Physics
Department of Physics, Faculty of Sciences
Aristotle University of Thessaloniki

Supervisor: Nikolaos Stergioulas

June 27, 2019
Stamatis Vretinaris: Data analysis of gravitational waves from binary neutron stars and machine learning applications © June 27, 2019

SUPERVISORS:
Nikolaos Stergioulas
Dedicated to the eternal quest of humankind to acquire knowledge.

To my parents Apostolis and Maria. Thank you for being my parents. Thank you for your sacrifices so I could study and follow my dream.
ABSTRACT

In the present work we studied gravitational waves from neutron star binaries. After merging, the remnant of the collision is a hypermassive metastable neutron star that strongly emits gravitational waves. These gravitational waves are heavily depending on the equation of state that governs neutron stars. We developed a tool for modeling the postmerger phase, which connect postmerger characteristics such as the main frequencies with general characteristics of neutron stars such as the chirp mass and the radius. In the second part of this work we applied machine learning techniques in order to cluster the aforementioned gravitational waves confirming the theoretical classification scheme suggested by Bauswein & Stergioulas, PRD, 91, 124056 (2015)
There is no royal road to science, and only those who do not dread the fatiguing climb of its steep paths have a chance of gaining its luminous summits.
— Karl Marx, Capital, Vol. 1: A Critical Analysis of Capitalist Production

ACKNOWLEDGEMENTS

I would like to thank all of the people that stood by me and helped me throughout the last years.

Many thanks to my friends at the Physics Lab of the Aristotle University of Thessaloniki for providing me with a friendly and productive environment to work and for all their support regarding technical questions about computers and programming.
I want to thank my fellow master students Alkis, Nikos, Michalis and Triandafilos for their friendship, interesting talks and ideas we exchanged.
A big thank you to my best friend Stelios who supported me all these years.
Words cannot put how much I thank Irida for supporting me so many years.

I would like to thank the members of the committee, Prof. Pappas and Prof Kougioumtzis, for carefully reading this thesis and for their useful comments. Further, I would like to thank Prof. Kougioumtzis for his help regarding data analysis and statistics topics that I faced during this thesis.
A very enormous big thank you to my mentor and supervisor Prof. Nikolaos Stergioulias who taught me everything and guided me through my first steps as a researcher starting from my B.Sc years. Without his input I wouldn’t have become what I am now.

Last but not least I want to thank my parents for everything they have been through just to make it possible for me to study in the first place at all. Without your effort none of this would have existed. Thank you for being my parents.
CONTENTS

I THEORETICAL INTRODUCTION 1
  1 INTRODUCTION 3
    1.1 Origin and interpretation of peaks in postmerger GW spectra . . . . . . 3
      1.1.1 Fundamental postmerger frequency $f_{peak}$ . . . . . . . . . . . . 4
      1.1.2 Secondary postmerger frequencies . . . . . . . . . . . . . . . . . . 4
    1.2 Spectral classification . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
    1.3 Empirical Relations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

II EMPIRICAL RELATIONS AND MACHINE LEARNING 9
  2 EMPIRICAL SURFACES FOR GWS IN THE POSTMERGER PHASE 11
    2.1 Data sets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.1.1 Bauswein et al dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.1.2 CORE dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.1.3 Combined dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
      2.1.4 Waveform preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . 13
      2.1.5 Equations of State . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
    2.2 Empirical relations based on the Bauswein dataset . . . . . . . . . . . . . 14
      2.2.1 Empirical relations for frequencies . . . . . . . . . . . . . . . . . . . . 14
      2.2.2 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
      2.2.3 Empirical relations for radii . . . . . . . . . . . . . . . . . . . . . . . . 19
    2.3 Empirical relations based on the CORE dataset . . . . . . . . . . . . . . . . 26
      2.3.1 Empirical relations for frequencies . . . . . . . . . . . . . . . . . . . . 26
      2.3.2 Empirical relations for radii . . . . . . . . . . . . . . . . . . . . . . . . 27
      2.3.3 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
    2.4 Combined surfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
      2.4.1 Combined surfaces for frequencies . . . . . . . . . . . . . . . . . . . . 30
      2.4.2 Combined surfaces for radii . . . . . . . . . . . . . . . . . . . . . . . . 31

3 MACHINE LEARNING 35
  3.1 Cluster Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
    3.1.1 DBSCAN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
    3.1.2 Affinity Propagation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
  3.2 Clustering of Bauswein data . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
    3.2.1 Distance and Similarity . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
    3.2.2 Clustering Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
  3.3 Classification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
Figure 1 Left panel: Illustration of the eigenfunction of the pressure oscillation with frequency $f = f_{\text{peak}}$ in the equatorial plane for a $1.35\text{-}1.35\, M_\odot$ merger with the Shen EoS. The figure is taken from [23]. Right panel: The GW spectrum of a $1.35\text{-}1.35\, M_\odot$ merger with the DD2 EoS. The green dashed line shows the GW spectrum of a simulation of a late-time merger remnant of the same model which was perturbed with a velocity field suitable to excite the fundamental quadrupolar fluid oscillation mode. Figure is taken from [4].

Figure 2 Rest mass density evolution in the equatorial plane for a $1.35\text{-}1.35\, M_\odot$ merger with the DD2 EoS. Black and white dots trace the positions of selected fluid elements of the antipodal bulges, which complete a full orbit within a millisecond. The cross and circle mark the double cores, which rotate at a faster frequency than the antipodal bulges. Figures from [2].

Figure 3 Different types of postmerger dynamics of different merger models. The outcome of a given calculation is shown at $M = M_{\text{tot}}/2$ plotted on the mass-radius relation of the EoS employed in the simulation. Red squares indicate Type I, black crosses stand for Type II and blue circles mark Type III. Figure from [2].

Figure 4 Rescaled dominant postmerger GW frequency $f_{\text{peak}}/M_{\text{tot}}$ as function of the radius $R_{1.6}$ of a non rotating NS with a gravitational mass of 1.6 for different EoSs and different total binary mass (plus signs for $2.4\, M_\odot$, circles for $2.7\, M_\odot$, crosses for $3.0\, M_\odot$) and a mass ratio of unity. Figure taken from [4].

Figure 5 Left panel: Binary mass configuration of Bauswein et al dataset. Right panel: Mass configuration with respect to the chirp mass for the Bauswein et al dataset.

Figure 6 Left panel: Binary mass configuration of the CORE dataset. Right panel: Mass configuration with respect to the chirp mass for the CORE dataset.
Figure 7 Surfaces $f_{\text{peak}}(R_x, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{\text{peak}}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_{\odot}$), while the light blue surface represents the empirical relations of the form of Equation 3. In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and 1.8$M_{\odot}$ was used. The surfaces are shown only in regions where data points are available.

Figure 8 Surfaces $f_{\text{peak}}(R_x, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{2-0}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_{\odot}$), while the light blue surface represents the empirical relations of the form of Equation 3. In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and 1.8$M_{\odot}$ was used. The surfaces are shown only in regions where data points are available.

Figure 9 Surfaces $f_{\text{peak}}(R_x, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{\text{spiral}}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_{\odot}$), while the light blue surface represents the empirical relations of the form of 3. In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and 1.8$M_{\odot}$ was used. The surfaces are shown only in regions where data points are available.

Figure 10 Empirical surfaces for frequencies with $R_{1.6}$ and $R_{1.8}$ using the whole dataset including unequal mass models. Blue surface corresponds to $f_j = f_{\text{peak}}$ [kHz], red surface corresponds to $f_j = f_{\text{spiral}}$ [kHz] and the green surface corresponds to $f_j = f_{2-0}$ [kHz]. The surfaces are shown only in regions where data exists.

Figure 11 In each panel, the percentage of data points that are closer in radius to each of the empirical relations (constructed with the corresponding frequency) is shown. The top row shows results of equal-mass configurations only and the bottom row uses all CFC/SPH data. For all masses, the radius is obtain more accurately when using the empirical relations for $f_{\text{peak}}$ in more than 50% of cases.
Figure 12  Radii empirical surfaces for equal binary mass configurations and $f_{\text{peak}}$ on the left and all mass configurations on the right with $f_{2-0}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data. ................................................................. 22

Figure 13  Radii empirical surfaces for equal binary mass configurations and $f_{\text{peak}}$ on the left and all mass configurations on the right with $f_{\text{peak}}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data. ................................................................. 23

Figure 14  Radii empirical surfaces for equal binary mass configurations and $f_{\text{peak}}$ on the left and all mass configurations on the right with $f_{2-0}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data. ................................................................. 24

Figure 15  Radii empirical surfaces for equal binary mass configurations and $f_{\text{spiral}}$ on the left and all mass configurations on the right with $f_{\text{peak}}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data. ................................................................. 25

Figure 16  Predictions for APR, DD2 and TM1 using $f_{\text{peak}}$ in Equation 12. The triangles correspond to a high mass model with $M_{\text{chirp}}$ larger than 1.3 thus we used the separated equations to predict. The squares indicate low mass models with $M_{\text{chirp}}$ less than 1.3. The errorbars indicate the maximum residual. The predictions were done based on the Bauswein data set. ................................................................. 26

Figure 17  With light blue color we denote the corresponding surface of the Bauswein dataset, and green color corresponds to the frequencies extracted from the CoRe GW catalogue. $f_{\text{peak}}(R_{1.8}, M_{\text{chirp}})$ surface for all mass binary configuration and data points from the CoRe GW catalogue. ................................................. 27

Figure 18  Empirical relations for the equal-mass models on the left and for the whole set of models on the right. Blue surface is Bauswein et al data surfaces, and green points are frequencies extracted from the CORE GW catalogue. ................. 28

Figure 19  Empirical relations for the equal-mass models on the left and for the whole set of models on the right. Blue surface is Bauswein et al data surfaces, and green points are frequencies extracted from the CORE GW catalogue. ................. 29
Figure 20  Empirical relations for the equal-mass models on the left and for the whole set of models on the right. Blue surface is Bauswein et al data surfaces, and green points are frequencies extracted from the CORE GW catalogue.

Figure 21  Combined data sets surfaces for frequencies. Red points correspond to Bauswein data and green points correspond to data extracted from the CoRe GW catalogue. Left figure: surface for $R_{1.8}$ and equal binary mass configurations. Right figure: surface for $R_{1.8}$ and all binary mass configurations.

Figure 22  Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data sets surfaces. The red points correspond to Bauswein et al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.

Figure 23  Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data sets surfaces. The red points correspond to Bauswein et al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.

Figure 24  Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data sets surfaces. The red points correspond to Bauswein et al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.

Figure 25  An illustration how DBSCAN works. Figure taken from [20]

Figure 26  How Affinity Propagation works. Figure taken from [13]

Figure 27  How message passing procedure works in Affinity Propagation. Figure taken from [13]

Figure 28  The calculated distance matrix. The matrix is symmetric thus the information lies only in one half of the matrix, either over or under the diagonal. On the x and y axis the numbers correspond to the time series.

Figure 29  Clustering results for the DBSCAN algorithm. The time series data are displayed at $M = M_{\text{tot}}/2$ and the corresponding radius to that mass. With gray lines we indicate the equations of state.
Figure 30  Top panel: The theoretical classification done by [REF]. Bottom panel: Clustering results for the Affinity Propagation algorithm. The time series data are displayed at $M = M_{tot}/2$ and the corresponding radius to that mass. With gray lines we indicate the equations of state. .......................... 42

Figure 31  Demonstration of the Kernel trick. The data is transformed using a kernel function $\Omega(x)$ and the classification is now performed in the higher dimensional space where the data is clear to divide. Figure taken from [10] .......................... 44

Figure 32  A basic depiction of a neural network with one hidden layer. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another. .......................... 45

Figure 33  Classification results using the Support Vector Machine algorithm. For the first figure we used a polynomial kernel of degree 3. For the second figure we used a polynomial kernel of degree 4 and for the last figure of degree 5. The crosses correspond to Type I, the squares to Type II and the circles to Type III models. .......................... 46

Figure 34  Classification results using a Neural Network model. The crosses correspond to Type I, the squares to Type II and the circles to Type III models. .......................... 46
<table>
<thead>
<tr>
<th>Table</th>
<th>Regression information for $f_{\text{peak}}$ frequency as $f_j$ in Equation 3 and only equal masses in mass configuration.</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{peak}}$ frequency as $f_j$ in Equation 3 and all masses in mass configuration.</td>
<td>92</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{2-0}$ frequency as $f_j$ in Equation 3 and only equal masses in mass configuration.</td>
<td>92</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{2-0}$ frequency as $f_j$ in Equation 3 and all masses in mass configuration.</td>
<td>93</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 3 and only equal masses in mass configuration.</td>
<td>93</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 3 and all masses in mass configuration.</td>
<td>93</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{peak}}$ frequency as $f_j$ in Equation 12 and only equal masses in mass configuration.</td>
<td>96</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{2-0}$ frequency as $f_j$ in Equation 12 and only equal masses in mass configuration.</td>
<td>96</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{2-0}$ frequency as $f_j$ in Equation 12 and all masses in mass configuration.</td>
<td>97</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 12 and only equal masses in mass configuration.</td>
<td>98</td>
</tr>
<tr>
<td>Table</td>
<td>Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 12 and all masses in mass configuration.</td>
<td>98</td>
</tr>
<tr>
<td>Table</td>
<td>Frequency data set we computed using the CoRe catalogue.</td>
<td>99</td>
</tr>
<tr>
<td>Table</td>
<td>TOV information for the equations of state we used in this paper.</td>
<td>100</td>
</tr>
<tr>
<td>Table</td>
<td>Bauswein et al data set.</td>
<td>101</td>
</tr>
<tr>
<td>Table</td>
<td>Bauswein et al data set.</td>
<td>102</td>
</tr>
<tr>
<td>Table</td>
<td>Bauswein et al data set.</td>
<td>103</td>
</tr>
<tr>
<td>Listing</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>Python code for computing the $R_x = f(f_j, M_{chirp})$ relations using the Bauswein data</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>Python code for computing the $f_j = f(R_x, M_{chirp})$ relations using the Bauswein data</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>Python code for computing the $R_x = f(f_j, M_{chirp})$ relations using the CORE data</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>Python code for computing the $f_j = f(R_x, M_{chirp})$ relations using the CORE data</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Python code for computing the $R_x = f(f_j, M_{chirp})$ relations using the combined dataset</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>Python code for computing the $f_j = f(R_x, M_{chirp})$ relations using the combined dataset</td>
<td>83</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>GW</td>
<td>Gravitational Wave</td>
<td></td>
</tr>
<tr>
<td>NS</td>
<td>Neutron Star</td>
<td></td>
</tr>
<tr>
<td>EoS</td>
<td>Equation of State</td>
<td></td>
</tr>
<tr>
<td>FWHM</td>
<td>Full Width at Half Maximum</td>
<td></td>
</tr>
</tbody>
</table>
Part I

THEORETICAL INTRODUCTION
INTRODUCTION

One of the targets for ground based gravitational wave detectors such as Advanced LIGO, Advanced Virgo, Kagra and Ligo India are Binary Neutron Star mergers. This kind of GW observations will provide information about the properties of high-density matter. The outcome of such an event will probably be a differentially rotating, strongly oscillating remnant [9-38 in Bauswein2015]. The very first detection of such a GW was done in August 2017. The Advanced LIGO-Advanced Virgo network observed a compact binary merger with a total mass of $2.74 \, M_{\odot}$ at a distance of about 40 Mpc. Gamma rays were detected after 1.7 seconds. Due to the sensitivity of the instruments we could not observe the post-merger phase of the GW but even with the inspiral part, constrains were placed. From the inspiral part we inferred the tidal deformability which describes a “matter effect” on the GW thus linking it to the EoS of NS matter. The measurement implies that nuclear matter cannot be very stiff and that the radii of NSs with masses of about $1.35 \, M_{\odot}$ cannot be larger than 14 km. Stronger constrains will be placed once we will observe the post-merger phase, it has been shown that detecting the dominant postmerger GW frequency $f_{\text{peak}}$ would strongly constrain the radius and the maximum mass of non rotating NSs [1, 3, 5, 8, 14]. In this chapter we will describe the post merger dynamics of a BNS merger.

1.1 ORIGIN AND INTERPRETATION OF PEAKS IN POSTMERGER GW SPECTRA

There are several distinct peaks in the GW spectrum of the postmerger phase, which are produced by certain physical mechanisms connected to oscillation modes and dynamical features of the postmerger remnant. Understanding these mechanisms is essential for the interpretation of the GW postmerger signals. In addition, GW searches are more sensitive if additional information about the signal to be detected is provided. Here we will present the current status of understanding of the dominant peak $f_{\text{peak}}$ and explain the origin of the two secondary peaks at lower frequencies, since observationally only the frequencies smaller than $f_{\text{peak}}$ are relevant.
1.1.1 Fundamental postmerger frequency $f_{\text{peak}}$

In [23] they applied a Fourier extraction of the eigenfunctions from simulation data for the first time to NS merger remnants. The analysis showed that the fundamental quadrupolar fluid frequency coincides with the dominant postmerger frequency $f_{\text{peak}}$, as shown in Figure 1. The dominant oscillation frequency in the GW spectrum is a generic feature which occurs in all merger simulations, that do not result in a prompt collapse [21].

1.1.2 Secondary postmerger frequencies

Considering the dynamics of the process, it is clear that the fundamental quasi-radial mode of the remnant is likely to be excited at a frequency which we will call $f_0$. Since the geometry of the NSs are nearly spherical this mode will produce only weak GWs at a frequency that the inspiral dominates. But a non-linear coupling between the quasi-radial oscillation and the fundamental quadrupolar mode does emit strong GWs and would explain one of the distinct secondary frequencies which lie at lower frequencies than the $f_{\text{peak}}$. At the lowest non-linear interaction level this coupling results in the appearance of a quasi-linear combination frequency $f_{2\pm 0} = f_{\text{peak}} \pm f_0$. The existence of such a frequency is a natural
consequence of the nonlinear evolution of two different frequencies simultaneously in the same star. These coupling mechanisms were extensively studied in [17].

\( f_{\text{spiral}} \)  In the GW spectra there exists one more secondary peak which lies between the frequencies \( f_{2-0} \) and \( f_{\text{peak}} \). In [2] they provided evidence that this secondary peak is generated by the orbital motion of two bulges, that form right after the merging, at the surface of the merger remnant (see Figure 2). During the merger the stars are strongly tidally deformed and matter of this tidal deformation at the outer edges cannot follow the fast inner rotation of the cores which lie in the inner part of the remnant. The matter at the outer edges forms antipodal bulges, which orbit around the central remnant at a lower orbital frequency. This structure survives for a few milliseconds i.e. after about two revolutions. Because of the spiral-like pattern the deformation described above, the authors in [2] called the frequency that this mechanism radiates GWs as \( f_{\text{spiral}} \).

Figure 2: Rest mass density evolution in the equatorial plane for a 1.35-1.35 \( M_\odot \) merger with the DD2 EoS. Black and white dots trace the positions of selected fluid elements of the antipodal bulges, which complete a full orbit within a millisecond. The cross and circle mark the double cores, which rotate at a faster frequency than the antipodal bulges. Figures from [2]
1.2 Spectral Classification

The first attempt to classify the GW spectra based on the understanding of the physical origin of the secondary peaks in the GW spectra was performed by [2]. Their analysis focuses in a larger number of simulations done with a SPH code by [5, 16], which imploys the CFC approximation for the evolution of the space-time [15, 27]. They identified three different types of postmerger dynamics and GW spectra for remnants which do not result in a prompt collapse, based on the relative strength between the secondary peaks $f_{2-0}$ and $f_{\text{spiral}}$.

![Figure 3: Different types of postmerger dynamics of different merger models. The outcome of a given calculation is shown at $M = M_{\text{tot}}/2$ plotted on the mass-radius relation of the EoS employed in the simulation. Red squares indicate Type I, black crosses stand for Type II and blue circles mark Type III. Figure from [2].](image)

**Type I** When the total mass $M_{\text{tot}}$ is not too far from the threshold mass for prompt collapse, the evolution of the central lapse function is dominated by a very strong quasi-radial oscillation of the remnant. Because of the strongly excited quasi-radial oscillation the secondary peak $f_{2-0}$ is the dominant secondary peak in the GW spectra while $f_{\text{spiral}}$ is much weaker.
**Type II**  For intermediate total binary masses the secondary frequencies $f_{2-0}$ and $f_{\text{spiral}}$ have comparable strength in the GW spectra and are well separated.

**Type III**  When the total binary mass is significantly below the threshold mass for quasi radial collapse the time evolution of the central lapse function is dominated by the $f_{\text{peak}} - f_{\text{spiral}}$ modulation that we described in the previous section as a result of the rotating spiral pattern with the two antipodal bulges. In the evolution of the central lapse function this modulation has typically smaller amplitude than the Type I variations. The dominant secondary peak is thus $f_{\text{spiral}}$ while the $f_{2-0}$ is either weak or hidden inside the background.

For any given EoS there is a continuous transition from one type to the other depending on the total binary mass. Type I and III are limiting cases of the more general Type II. For example, a binary with total binary mass of $M_{\text{tot}} = 2.7 \, M_\odot$ all three types are possible depending on the EoS. Softer EoSs lead to Type I behavior in the postmerger dynamics whereas very stiff EoSs lead to Type III dynamics. This picture of the postmerger dynamics based on this spectral classification is shown in Figure 3

Based on this classification scheme the authors of [2] derived regions where the secondary peaks lie. For total masses $2.4 M_\odot \leq M_{\text{tot}} \leq 3.0 M_\odot$ and mass ratios of $q = 1$ the secondary peaks appear in distinct frequency regions

- $f_{2-0}$: $f_{\text{peak}} - 1.3 \, kHz \leq f_{2-0} \leq f_{\text{peak}} - 0.9 \, kHz$,
- $f_{\text{spiral}}$: $f_{\text{peak}} - 0.9 \, kHz \leq f_{\text{spiral}} \leq f_{\text{peak}} - 0.5 \, kHz$,

thus providing a very useful tool for detecting/interpreting those frequencies in future GW observations. Although, this picture was derived from simulations with mass ratios of $q = 1$. For unequal binary mass configuration the picture should change a bit but the whole scheme stands as we will see in the next chapters.

1.3 EMPIRICAL RELATIONS

Similar to the fundamental quadrupolar frequency $f_{\text{peak}}$ the secondary peaks as well depend on the EoS. It is important to explore this dependence, understand it and properly model it thus providing us, another tool to place further constrains on the EoSs using future GW observations of the postmerger phase. It is important to note that the secondary peaks have usually lower signal to noise ratio in comparison with the main $f_{\text{peak}}$ and have broader FWHM too.

Empirical relations between the dominant postmerger frequency $f_{\text{peak}}$ and EoS properties were first investigated for a fixed binary mass configuration in [1, 5].
Stellar parameters of non rotating NSs are uniquely linked to the EoS through the Tolman-Oppenheimer-Volkoff (TOV) equations. For example, the peak frequency $f_{\text{peak}}$ of $1.35-1.35M_\odot$ mergers shows a clear correlation with the radius $R_{1.35}$ of a non rotating NS with mass $M = 1.35M_\odot$, (see Fig. 4 in [5] and Fig. 12 in [1]).

Similar tight correlations exist for other fiducial masses (see Figs. 9 to 12 in [5]). The tightest relation for a $1.35-1.35M_\odot$ is with the radius $R_{1.6}$. This relation can be written as

$$f_{\text{peak}} = \begin{cases} -0.2823 \cdot R_{1.6} + 6.284, & \text{for } f_{\text{peak}} < 2.8\text{kHz}, \\ -0.4667 \cdot R_{1.6} + 8.713, & \text{for } f_{\text{peak}} > 2.8\text{kHz}. \end{cases}$$

(1)

(the maximum deviation of the data points from a least-square fit is considered as figure of merit to assess the quality and accuracy of the relations). For $R_{1.6}$ the maximum scatter is less than 200 m. For other fixed binary masses, e.g. $1.2-1.2M_\odot$, $1.2-1.5M_\odot$ or $1.5-1.5M_\odot$ mergers, similar scalings between $f_{\text{peak}}$ and NS radii exist, scaled by the total mass [4]. This empirical relations are being shown in Figure 4.

![Figure 4: Rescaled dominant postmerger GW frequency $f_{\text{peak}}/M_{\text{tot}}$ as function of the radius $R_{1.6}$ of a non rotating NS with a gravitational mass of 1.6 for different EoSs and different total binary mass (plus signs for 2.4 $M_\odot$, circles for 2.7 $M_\odot$, crosses for 3.0 $M_\odot$) and a mass ratio of unity. Figure taken from [4]](image)

$$f_{\text{peak}}/M_{\text{tot}} = 0.0157 \cdot R_{1.6}^2 - 0.5495 \cdot R_{1.6} + 5.5030.\quad (2)$$

See [6] for a similar rescaling but with the tidal coupling constant.
Part II

EMPIRICAL RELATIONS AND MACHINE LEARNING
EMPIRICAL SURFACES FOR GWS IN THE POSTMERGER PHASE

In the previous chapter we discussed some of the previous work done on empirical relations for GWs focusing on the postmerger part of the signal. In this chapter we will present our work regarding the extension of such empirical relations.

2.1 DATA SETS

We will have a quick look on the data sets we used in order to apply our analyses. The primary data set we used are GWs produced by [5, 16] with a smoothed-particle hydrodynamics (SPH) code in the general-relativistic spatial conformal flatness approximation (CFC) [15, 27].

2.1.1 Bauswein et al dataset

Our first GW catalogue of BNS mergers is produced with 3D simulation code of [5, 16]. The code employs a SPH method for the hydrodynamics and the CFC approximation for the evolution of spacetime [15, 27]. Gravitational waves are extracted through the quadrupole formalism. The EoSs used are both high temperature and hybrid (cold plus ideal fluid thermal part). There are 49 equal mass models, with masses ranging from $1.2 M_\odot$ to $1.9 M_\odot$ and 41 unequal mass models, with masses ranging from $1.2 M_\odot$ to $2.0 M_\odot$ and mass ratios as low as 0.67. More information about the dataset can be seen in Figure 5.

2.1.2 CORE dataset

The CORE dataset is a large public database of BNS merger waveforms constructed through simulations in full numerical relativity. We selected a subset of models available, for which the initial stars have zero spin and eccentricity lower than 0.02. In cases where the same model is available for multiple resolutions, we selected the highest resolution (denoted as Ro1 in [11]). Also, in cases where multiple waveforms were available for initial setups that differed only slightly in mass (due to a different initial separation distance), we selected the model with the lowest initial GW frequency (at the start of the simulation, before merger) which
corresponds to the largest initial separation distance. It includes equal mass models in the mass range $1.35 \, M_\odot$ to $1.5 \, M_\odot$ and unequal mass models in the mass range $0.94 \, M_\odot$ to $1.94 \, M_\odot$ and mass ratio as low as $0.49$. More information about the dataset can be seen in Figure 6.

2.1.3 Combined dataset

We merged the two datasets into a bigger combined dataset and applied our analyses on the combined dataset. The two datasets can be merged since the compu-
Figure 6: Left panel: Binary mass configuration of the CORE dataset. Right panel: Mass configuration with respect to the chirp mass for the CORE dataset.

2.1.4 Waveform preprocessing

Initially we applied a Tukey window to the time domain data with an alpha factor of $\alpha = 0.1$ and zero padded each time series to 16384 samples in total. As a next step we calculate the Fourier transform of the time series waveform. Finally we construct the effective amplitude $h_{\text{eff}} = \hat{h} \sqrt{f}$, where $\hat{h}$ is the Fourier transform of the time domain GW signal $h$ and $f$ is the frequency. The extraction of the $f_{\text{peak}}$...
frequency is unambiguous, since its the highest peak present in the GW spectra. For the extraction and identification of the secondary peaks $f_{2-0}$ and $f_{\text{spiral}}$ we used the classification scheme introduced in [2] and explain in chapter 1.

2.1.5 Equations of State

In order to relate the postmerger GW frequencies to the radius of individual non-rotating stars, we computed nonrotating models of different masses with the same set of EOSs as for the BNS merger simulations. For EOSs that are defined as piecewise polytropes in [19], we used the pyTOVpp code \(^1\), whereas other EOSs were used in their original tabulated form with the RNS code [22]. Notice that the CFC/SPH simulation code uses the tabulated form for all EOSs. Small discrepancies that arise in the determination of the radius of a nonrotating star between the tabulated and the piecewise polytropic approximation of an EOS are within the maximum deviation of the empirical relations.

2.2 Empirical relations based on the Bauswein dataset

Using a least-squares minimization method we construct two-parameter relations of the form $f_j(R_x, M_{\text{chirp}})$, where $j$ stands for one of the three frequency peaks $f_{\text{peak}}$, $f_{2-0}$ or $f_{\text{spiral}}$ and $x$ stands for the mass of the nonrotating models, in solar masses (e.g. $R_{1.6}$ stands for the radius of a nonrotating model of mass $M = 1.6 \, M_\odot$). Relations are obtained both for the subset of equal mass configurations and for the whole set of models, which includes both equal and unequal mass configurations.

2.2.1 Empirical relations for frequencies

The two-parameter empirical relations of the form $f_j(R_x, M_{\text{chirp}})$ were chosen to be second-order expansions in the two parameters (including a mixed term):

$$f_j/M_{\text{chirp}} = b_0 + b_1 M_{\text{chirp}} + b_2 R_x + b_3 M_{\text{chirp}}^2 + b_4 R_x M_{\text{chirp}} + b_5 R_x^2,$$  \hspace{1cm} (3)

In each case we investigate the maximum residual and the adjusted coefficient of determination $R^2$.

---

\(^1\) The python code pyTOVpp was used, available at https://github.com/niksterg/pyTOVpp
2.2.1.1 Empirical relations for $f_{\text{peak}}$

For the dominant postmerger frequency $f_{\text{peak}}$ and for the subset of equal-mass configurations, the empirical relation with smallest error is obtained for neutron stars of mass $1.6M_\odot$.

\[
\frac{f_{\text{peak}}}{M_{\text{chirp}}} = 13.822 - 0.576M_{\text{chirp}} - 1.375R_{1.6} \\
+0.479M_{\text{chirp}}^2 - 0.073R_{1.6}M_{\text{chirp}} + 0.044R_{1.6}^2.
\]  

(4)

This fit has a maximum residual of 0.196 kHz over the whole parameter space and $R^2 = 0.98$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses, are shown in Table 1 in Appendix B. The maximum residual ranges from 0.196 kHz to 0.257 kHz.

For the whole set of models (including both equal and unequal masses), we display the empirical relations of the form of Eq. (Equation 3), for $R_X = 1.2, 1.4, 1.6$ and $1.8M_\odot$, in. The empirical relation with the smallest residual is obtained for neutron stars of mass $1.8M_\odot$:

\[
\frac{f_{\text{peak}}}{M_{\text{chirp}}} = 10.942 - 0.369M_{\text{chirp}} - 0.987R_{1.8} \\
+1.095M_{\text{chirp}}^2 - 0.201R_{1.8}M_{\text{chirp}} + 0.036R_{1.8}^2,
\]

which has a maximum residual of 0.247 kHz over the whole parameter space and $R^2 = 0.976$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses are shown in Table 2 in Appendix B. The maximum residual ranges from 0.247 kHz to 0.374 kHz.
Figure 7: Surfaces $f_{\text{peak}}(R, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{\text{peak}}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_\odot$), while the light blue surface represents the empirical relations of the form of Equation 3. In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and 1.8$M_\odot$ was used. The surfaces are shown only in regions where data points are available.

2.2.1.2 Empirical relations for $f_{2-0}$

For the secondary postmerger frequency $f_{2-0}$ and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for neutron stars of mass 1.6$M_\odot$:

$$f_{2-0}/M_{\text{chirp}} = 8.943 + 4.059M_{\text{chirp}} - 1.332R_{1.6} - 0.358M_{\text{chirp}}^2 - 0.182R_{1.6}M_{\text{chirp}} + 0.048R_{1.6}^2,$$

with a maximum residual of 0.229 kHz and $R^2 = 0.931$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses are shown in Table 3 in Appendix B. The maximum residual ranges from 0.229 kHz to 0.366 kHz. For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for neutron stars of mass 1.6$M_\odot$:

$$f_{2-0}/M_{\text{chirp}} = 9.586 + 4.09M_{\text{chirp}} - 1.427R_{1.6} + 0.048M_{\text{chirp}}^2 - 0.261R_{1.6}M_{\text{chirp}} + 0.055R_{1.6}^2,$$

with a maximum residual of 0.252 kHz and $R^2 = 0.947$. 
The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses are shown in Table 4 in Appendix B. The maximum residual ranges from 0.252 kHz to 0.383 kHz.

![Graphs showing empirical relations between $f_{2-0}/M_{\text{chirp}}$ and $R_{1.2}$, $R_{1.4}$, $R_{1.6}$, and $R_{1.8}$ for different masses.]

Figure 8: Surfaces $f_{\text{peak}}(R_x, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{2-0}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_\odot$), while the light blue surface represents the empirical relations of the form of Equation 3. In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and 1.8$M_\odot$ was used. The surfaces are shown only in regions where data points are available.

2.2.1.3 Empirical relations for $f_{\text{spiral}}$

For the secondary postmerger frequency $f_{\text{spiral}}$ and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for neutron stars of mass 1.8$M_\odot$:

$$f_{\text{spiral}}/M_{\text{chirp}} = 6.264 + 1.929 M_{\text{chirp}} - 0.645 R_{1.8} + 0.881 M_{\text{chirp}}^2 - 0.311 R_{1.8} M_{\text{chirp}} + 0.03 R_{1.8}^2,$$

with a maximum residual of 0.286 kHz and $R^2 = 0.944$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses are shown in Table 5 in Appendix B. The maximum residual ranges from 0.286 kHz to 0.422 kHz.
For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained again for neutron stars of mass $1.8M_\odot$:

$$f_{\text{spiral}}/M_{\text{chirp}} = 5.846 + 1.75M_{\text{chirp}} - 0.555R_{1.8}$$
$$+ 1.002M_{\text{chirp}}^2 - 0.316R_{1.8}M_{\text{chirp}} + 0.026R_{1.8}^2$$

with a maximum residual of 0.27 kHz and $R^2 = 0.93$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other masses are shown in Table 6 in Appendix B. The maximum residual ranges from 0.27 kHz to 0.438 kHz.

![Figure 9: Surfaces $f_{\text{peak}}(R_x, M_{\text{chirp}})$ using the whole SPH/CFC data set. Red dots show the extracted frequencies $f_{\text{spiral}}$ scaled by the chirp mass $M_{\text{chirp}}$ (in units of kHz/$M_\odot$), while the light blue surface represents the empirical relations of the form of ??.

In the different panels, the radius of nonrotating neutron stars of mass 1.2, 1.4, 1.6 and $1.8M_\odot$ was used. The surfaces are shown only in regions where data points are available.

### 2.2.2 Discussion

As explained in chapter 1 the secondary frequencies lie in distinct regions in the postmerger GW spectrum. In Figure 10 we display the surfaces corresponding to the empirical relations for the three different postmerger frequencies $f_{\text{peak}}$, $f_{\text{spiral}}$ and $f_{2-0}$ for the whole CFC/SPH dataset, as a function of $M_{\text{chirp}}$ and $R_x$ (using $R_{1.6}$ in the left panel and and $R_{1.8}$ in the right panel). The surfaces are shown only in regions where data exists. It is clear that all three frequencies are distinct...
in the whole parameter space. This verifies that the two secondary post-merger frequencies $f_{2-0}$ and $f_{\text{spiral}}$ are distinct, each satisfying a different empirical relation, as proposed in [Bauswein2015]. In contrast, our findings do not agree with the proposed "universal" relation of a single secondary postmerger frequency as proposed in [24, 25].

![Empirical surfaces for frequencies with $R_{1.6}$ and $R_{1.8}$ using the whole dataset including unequal mass models. Blue surface corresponds to $f_j = f_{\text{peak}}$ [kHz], red surface corresponds to $f_j = f_{\text{spiral}}$ [kHz] and the green surface corresponds to $f_j = f_{2-0}$ [kHz]. The surfaces are shown only in regions where data exists.](image)

**Figure 10:**

2.2.3 **Empirical relations for radii**

Following the same procedure as for the frequencies we construct empirical relations for the radii. We tried different mathematical expressions and among them was the lowest complexity level, such as the linear case:

$$R_x = b_0 + b_1 M_{\text{chirp}} + b_2 f_j$$  \hspace{1cm} (10)

But the surfaces generated by the second degree case performed better than the above. The reader can find all the regression information about both linear and quadratic case in the Appendix B.

**Separation of dataset** By looking at Figure 5 and Figure 6 one sees clearly that the two datasets doesn’t cover the same span in terms of the chirp mass $M_{\text{chirp}}$. In order to compare “equal” cases we separated the Bauswein dataset in two. With ‘sep’ in the tables we denote that the dataset has been separated at $M_{\text{chirp}} = 1.3$. So for the $R_{1.2,1.4}(f_j, M_{\text{chirp}})$ surfaces, we used only the data that had $M_{\text{chirp}} < 1.3$, and for $R_{1.8}(f_j, M_{\text{chirp}})$ surfaces, only the data that had $M_{\text{chirp}} > 1.3$. This is natural to assume since the lower mass ($M_{\text{chirp}}$) binaries are not suitable to use in order
to infer information for the heavier postmerger remnants. This means that taking into account the low chirp mass models to infer for the high mass remnants will result in false input and will lead to misinterpretations in our conclusions, and vice versa. Taking all of the above into account the equation we used for regression is:

\[
R_x = b_0 + b_1 M_{\text{chirp}} + b_2 f_j / M_{\text{chirp}} + b_3 M_{\text{chirp}}^2 + b_4 f_j + b_5 (f_j / M_{\text{chirp}})^2,
\]

\[(12)\]

**WHICH EMPIRICAL RELATION IS THE MOST ACCURATE?** This question has to be answered before we present the empirical relations for radii. It is easy to notice that one can construct three empirical relations for a radius, i.e. for \( R_{1.6} \) we can have the following empirical relations:

- Using \( f_{\text{peak}} \) as \( f_j \) in Equation 12 : \( R_{1.6}(f_{\text{peak}}, M_{\text{chirp}}) \)
- Using \( f_{2-0} \) as \( f_j \) in Equation 12 : \( R_{1.6}(f_{2-0}, M_{\text{chirp}}) \)
- Using \( f_{\text{spiral}} \) as \( f_j \) in Equation 12 : \( R_{1.6}(f_{\text{spiral}}, M_{\text{chirp}}) \)

In order to answer this question we studied the differences between the true radius of the EoS for a given mass (e.g. \( R_{1.6} \)), and the prediction given by the empirical surfaces

\[
d = |R_{x}^{\text{true}} - R_{x}^{\text{predicted}}|,
\]

We found that in most cases (above 50%), see Figure 11, the surface that is generated using \( f_{\text{peak}} \) as \( f_j \) in Equation 12 gives the closest predictions to the true value of the radius. Since there are cases where the other surfaces which used the secondary frequencies perform better than the \( f_{\text{peak}} \) it is advised to be careful in which surface to use. We also caution that secondary peaks have typically a larger FWHM and a lower SNR. This will result in a larger measurement error of the frequency and thus further degrade the predictive power of using surfaces of secondary features. Finally, the following surfaces are valid only in the regions where data points exists, which means the region

\[
M_{\text{chirp}} > 1.3 \quad \wedge \quad f_{\text{peak}} / M_{\text{chirp}} > 2.2
\]

\[(13)\]

must be avoided and not used. Taking this into account we will use only the surfaces generated with \( f_{\text{peak}} \) as \( f_j \).
For the \( R_{1.2} \) and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{1.2}^{\text{sep}} = 52.201 - 29.769 M_{\text{chirp}} - 15.398 f_{\text{peak}} / M_{\text{chirp}} + 8.918 M_{\text{chirp}}^2 + 3.333 f_{\text{peak}} + 1.832 \left( f_{\text{peak}} / M_{\text{chirp}} \right)^2,
\]

(14)

with a maximum residual of 0.52 km and \( R^2 = 0.945 \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Table 7, Table 9 and Table 11 in Appendix B. The maximum residual ranges from 0.52 km to 0.803 km.

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{1.2}^{\text{sep}} = 56.906 - 37.252 M_{\text{chirp}} - 15.701 f_{\text{peak}} / M_{\text{chirp}} + 11.756 M_{\text{chirp}}^2 + 3.638 f_{\text{peak}} + 1.83 \left( f_{\text{peak}} / M_{\text{chirp}} \right)^2,
\]

(15)

with a maximum residual of 0.526 km and \( R^2 = 0.951 \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Ta-
ble 8, Table 10 and Table 12 in Appendix B. The maximum residual ranges from 0.526 km to 0.737 km.

\[
R_{\text{sep}}^{1.4} = 51.229 - 30.463M_{\text{chirp}} - 14.143 \frac{f_{\text{peak}}}{M_{\text{chirp}}} + 9.46M_{\text{chirp}}^2 + 3.09f_{\text{peak}} + 1.612 \left( \frac{f_{\text{peak}}}{M_{\text{chirp}}} \right)^2, \tag{16}
\]

with a maximum residual of 0.412 km and \( R^2 = 0.966 \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Table 7, Table 9 and Table 11 in Appendix B. The maximum residual ranges from 0.412 km to 0.731 km.

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{\text{sep}}^{1.4} = 55.809 - 37.642M_{\text{chirp}} - 14.473 \frac{f_{\text{peak}}}{M_{\text{chirp}}} + 12.15M_{\text{chirp}}^2 + 3.41f_{\text{peak}} + 1.609 \left( \frac{f_{\text{peak}}}{M_{\text{chirp}}} \right)^2, \tag{17}
\]

with a maximum residual of 0.493 km and \( R^2 = 0.968 \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Table 8, Table 10 and Table 12 in Appendix B. The maximum residual ranges from 0.493 km to 0.676 km.
2.2 Empirical relations based on the Bauswein dataset

2.2.3.3 Empirical relations for $R_{1.6}$

For the $R_{1.6}$ and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$, but using the secondary postmerger frequency $f_{2-0}$ gave close performance:

$$R_{1.6} = 41.316 - 16.654M_{\text{chirp}} - 12.458f_{\text{peak}}/M_{\text{chirp}} + 3.722M_{\text{chirp}}^2 + 2.936f_{\text{peak}} + 1.269 \left( f_{\text{peak}}/M_{\text{chirp}} \right)^2 ,$$

(18)

$$R_{1.6} = 15.271 + 4.123M_{\text{chirp}} - 6.661f_{2-0}/M_{\text{chirp}} - 1.188M_{\text{chirp}}^2 + 1.23f_{2-0} + 0.783 \left( f_{2-0}/M_{\text{chirp}} \right)^2 ,$$

(19)

with a maximum residual of 0.462 km and $R^2 = 0.97$ for the empirical relation using $f_{\text{peak}}$ and a maximum residual of 0.465 km and $R^2 = 0.942$ for the empirical relation using $f_{2-0}$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other frequencies are shown in Table 7, Table 9 and Table 11 in Appendix B. The maximum residual ranges from 0.462 km to 0.706 km.

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the secondary postmerger frequency $f_{2-0}$, but using the dominant postmerger frequency $f_{\text{peak}}$ gave close performance:

$$R_{1.6} = 43.796 - 19.984M_{\text{chirp}} - 12.921f_{\text{peak}}/M_{\text{chirp}} + 4.674M_{\text{chirp}}^2 + 3.371f_{\text{peak}} + 1.26 \left( f_{\text{peak}}/M_{\text{chirp}} \right)^2 ,$$

(20)

$$R_{1.6} = 17.764 + 2.497M_{\text{chirp}} - 8.797f_{2-0}/M_{\text{chirp}} - 0.639M_{\text{chirp}}^2 + 1.393f_{2-0} + 1.452 \left( f_{2-0}/M_{\text{chirp}} \right)^2 ,$$

(21)

Figure 13: Radii empirical surfaces for equal binary mass configurations and $f_{\text{peak}}$ on the left and all mass configurations on the right with $f_{\text{peak}}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al. data.
with a maximum residual of 0.518 km and \( R^2 = 0.955 \) for the empirical relation using \( f_{2-0} \) and a maximum residual of 0.526 km and \( R^2 = 0.969 \) for the empirical relation using \( f_{\text{peak}} \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Table 8, Table 10 and Table 12 in Appendix B. The maximum residual ranges from 0.518 km to 0.674 km.

The maximum residual ranges from 0.212 km to 0.597 km.

\[
R_{1.8}^{\text{sep}} = 55.934 - 37.162 M_{\text{chirp}} - 17.139 f_{\text{spiral}} / M_{\text{chirp}} \\
+ 7.961 M_{\text{chirp}}^2 + 9.897 f_{\text{spiral}} - 0.382 \left( f_{\text{spiral}} / M_{\text{chirp}} \right)^2, 
\]  

(22)

\[
R_{1.8}^{\text{sep}} = 33.802 - 3.069 M_{\text{chirp}} - 15.522 f_{\text{peak}} / M_{\text{chirp}} \\
- 1.439 M_{\text{chirp}}^2 + 4.112 f_{\text{peak}} + 1.605 \left( f_{\text{peak}} / M_{\text{chirp}} \right)^2, 
\]  

(23)

Figures 14: Radii empirical surfaces for equal binary mass configurations and \( f_{\text{peak}} \) on the left and all mass configurations on the right with \( f_{2-0} \). The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data.

2.2.3.4 Empirical relations for \( R_{1.8} \)

For the \( R_{1.8} \) and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the secondary postmerger frequency \( f_{\text{spiral}} \), but using the dominant postmerger frequency \( f_{\text{peak}} \) gave close performance:

with a maximum residual of 0.212 km and \( R^2 = 0.951 \) for the empirical relation using \( f_{\text{spiral}} \) and a maximum residual of 0.276 km and \( R^2 = 0.951 \) for the empirical relation using \( f_{\text{peak}} \). The coefficients \( b_0 - b_5 \) for the empirical relations constructed for other frequencies are shown in Table 7, Table 9 and Table 11 in Appendix B. The maximum residual ranges from 0.212 km to 0.597 km.
2.2 Empirical Relations Based on the Bauswein Dataset

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.8}^{\text{sep}} = 54.467 - 38.851M_{\text{chirp}} - 13.992f_{\text{peak}}/M_{\text{chirp}} + 9.305M_{\text{chirp}}^2 + 8.453f_{\text{peak}} - 0.614\left(f_{\text{peak}}/M_{\text{chirp}}\right)^2,$$

with a maximum residual of 0.275 km and $R^2 = 0.958$. The coefficients $b_0 - b_5$ for the empirical relations constructed for other frequencies are shown in Table 8, Table 10 and Table 12 in Appendix B. The maximum residual ranges from 0.275 km to 0.569 km.

Figure 15: Radii empirical surfaces for equal binary mass configurations and $f_{\text{spiral}}$ on the left and all mass configurations on the right with $f_{\text{peak}}$. The surfaces are plotted only in the areas where data exists. The red points correspond to Bauswein et al data.

2.2.3.5 Discussion

In general, we will be able to measure the chirp mass and the $f_{\text{peak}}$ with high accuracy [CITATION]. There is a chance that the error bars will be larger if the beaming effect reported in [26] is taken into account. These surfaces can be used to predict the $R_x$ radii of the star. Since there are more than one surface one can construct from $f_{\text{peak}}$, one can derive predictions for $R_{1.2}$ to $R_{1.8}$ and place bounds on the M-R diagram regarding the possible equations of states. An application of this is shown in Figure 16, where the predictions and the error bars are displayed for $R_{1.4}$, $R_{1.6}$ and $R_{1.8}$.
Figure 16: Predictions for APR, DD2 and TM1 using $f_{\text{peak}}$ in Equation 12. The triangles correspond to a high mass model with $M_{\text{chirp}}$ larger than 1.3 thus we used the separated equations to predict. The squares indicate low mass models with $M_{\text{chirp}}$ less than 1.3. The errorbars indicate the maximum residual. The predictions were done based on the Bauswein data set.

2.3 EMPIRICAL RELATIONS BASED ON THE CORE DATASET

We applied the exact same methodology and searched for empirical relations for $f_{\text{peak}}$, both $f_{\text{peak}}(R_x, M_{\text{chirp}})$ and $R_x(f_{\text{peak}}, M_{\text{chirp}})$ empirical relations. In order to apply the same for $f_{\text{spiral}}$ and $f_{2-0}$ we would need the simulation data to extract the quasi-radial frequency from the hydrodynamical simulation, which we don’t have available.

We found that again it is possible to fit a surface with high accuracy to the data and those surfaces coincide with the previous surfaces.

2.3.1 Empirical relations for frequencies

The relevant surfaces to validate are the ones we used the $f_{\text{peak}}$ frequency in the previous section. Due to the low number of data points the linear case of surfaces performed better in contrast to the quadratic Equation 3.

$$f_{\text{peak}}/M_{\text{chirp}} = b_0 + b_1 M_{\text{chirp}} + b_2 R_x,$$

(25)
For the dominant postmerger frequency peak and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for neutron stars of mass $1.8 M_\odot$:

$$f_{\text{peak}}/M_{\text{chirp}} = 8.198 - 1.782 M_{\text{chirp}} - 0.298 R_{1.8},$$

with a maximum residual of $0.147$ kHz and $R^2 = 0.959$. The maximum residual from the Bauswein surfaces is $0.063$ kHz.

For the dominant postmerger frequency peak and using the whole set of models, the empirical relation with the smallest error is obtained for neutron stars of mass $1.8 M_\odot$:

$$f_{\text{peak}}/M_{\text{chirp}} = 5.996 - 0.039 M_{\text{chirp}} - 0.287 R_{1.8},$$

with a maximum residual of $0.194$ kHz and $R^2 = 0.943$. The maximum residual from the Bauswein surfaces is $0.103$ kHz.

Figure 17: With light blue color we denote the corresponding surface of the Bauswein dataset, and green color corresponds to the frequencies extracted from the CoRe GW catalogue. $f_{\text{peak}}(R_{1.8}, M_{\text{chirp}})$ surface for all mass binary configuration and data points from the CoRe GW catalogue.

2.3.2  *Empirical relations for radii*

Since the CoRe catalogue has chirp masses from 1 to 1.2 in order to get a better sense of comparison we will compare the surfaces from subsection 2.2.3 which we used the separated dataset at $M_{\text{chirp}} = 1.3$. If one wants to compare the surfaces with our non-separated dataset with the CoRe catalogue frequencies, one could plot the given surfaces since the numbers are available in the current work.

The equation that performed best in the full numerical relativity dataset was the linear Equation 11, but one must remember that this dataset spans only from $1 < M_{\text{chirp}} < 1.2$ and in our dataset we had chirp masses as high as 1.7. This could be just a linear approximation for this low number of data points in this limited chirp mass regime.
2.3.2.1 Empirical relations for $R_{1.2}$

For the $R_{1.2}$ and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.2} = 15.881 + 2.864 M_{\text{chirp}} - 2.73 f_{\text{peak}} / M_{\text{chirp}}$$  \hfill (28)

with a maximum residual of $0.5$ km and $R^2 = 0.886$. For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.2} = 17.845 + 1.122 M_{\text{chirp}} - 2.646 f_{\text{peak}} / M_{\text{chirp}}$$  \hfill (29)

with a maximum residual of $0.798$ km and $R^2 = 0.882$.

Figure 18: Empirical relations for the equal-mass models on the left and for the whole set of models on the right. Blue surface is Bauswein et al data surfaces, and green points are frequencies extracted from the CORE GW catalogue.

2.3.2.2 Empirical relations for $R_{1.4}$

For the $R_{1.4}$ and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.4} = 16.924 + 2.198 M_{\text{chirp}} - 2.839 f_{\text{peak}} / M_{\text{chirp}},$$  \hfill (30)

with a maximum residual of $0.408$ km and $R^2 = 0.926$. For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.4} = 18.269 + 1.069 M_{\text{chirp}} - 2.808 f_{\text{peak}} / M_{\text{chirp}},$$  \hfill (31)
with a maximum residual of 0.703 km and $R^2 = 0.917$.

\[ R_{1.6} = 19.532 + 0.216 M_{\text{chirp}} - 2.985 f_{\text{peak}} / M_{\text{chirp}}, \]  

(32)

with a maximum residual of 0.393 km and $R^2 = 0.955$. For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

\[ R_{1.6} = 18.27 + 1.064 M_{\text{chirp}} - 2.806 f_{\text{peak}} / M_{\text{chirp}}, \]  

(33)

with a maximum residual of 0.703 km and $R^2 = 0.914$.

Figure 20: Empirical relations for the equal-mass models on the left and for the whole set of models on the right. Blue surface is Bauswein et al data surfaces, and green points are frequencies extracted from the CORE GW catalogue.
2.3.3 Discussion

As we see from Figure 17 to Figure 20 our empirical relations of the Bauswein data set are in good agreement with the frequencies extracted from the full numerical relativity catalogue.

2.4 Combined surfaces

As a last step in our work we present the surfaces after combining the Bauswein et al data set and the frequencies we extracted from the CoRe GW catalogue.

2.4.1 Combined surfaces for frequencies

The corresponding surfaces can be seen in Figure 21.

2.4.1.1 Empirical relations for $f_{\text{peak}}$

For the dominant postmerger frequency peak and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for neutron stars of mass $1.8M_\odot$.

$$f_{\text{peak}}/M_{\text{chirp}} = 11.476 + 0.025M_{\text{chirp}} - 1.102R_{1.8}$$
$$+1.181M_{\text{chirp}}^2 - 0.242R_{1.8}M_{\text{chirp}} + 0.042R_{1.8}^2,\quad (34)$$

with a maximum residual of 0.14 kHz and $R^2 = 0.975$. Using the whole set of models, the empirical relation with the smallest error is obtained for neutron stars of mass $1.8M_\odot$.

$$f_{\text{peak}}/M_{\text{chirp}} = 9.044 + 0.713M_{\text{chirp}} - 0.804R_{1.8}$$
$$+1.017M_{\text{chirp}}^2 - 0.259R_{1.8}M_{\text{chirp}} + 0.031R_{1.8}^2,\quad (35)$$

with a maximum residual of 0.197 kHz and $R^2 = 0.966$. 
Figure 21: Combined data sets surfaces for frequencies. Red points correspond to Bauwsein data and green points correspond to data extracted from the CoRe GW catalogue. Left figure: surface for $R_{1.8}$ and equal binary mass configurations. Right figure: surface for $R_{1.8}$ and all binary mass configurations.

2.4.2 Combined surfaces for radii

2.4.2.1 Empirical relations for $R_{1.2}$

For the $R_{1.2}$ and using the subset of *equal-mass* configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.2} = 37.177 - 14.073M_{\text{chirp}} - 10.474f_{\text{peak}}/M_{\text{chirp}}$$

$$+ 3.038M_{\text{chirp}}^2 + 2.603f_{\text{peak}} + 0.979 \left( f_{\text{peak}}/M_{\text{chirp}} \right)^2,$$

(36)

with a maximum residual of 0.862 km and $R^2 = 0.9$.

For the *whole set* of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.2}^{\text{sep}} = 30.595 - 8.388M_{\text{chirp}} - 7.804f_{\text{peak}}/M_{\text{chirp}}$$

$$+ 1.768M_{\text{chirp}}^2 + 1.491f_{\text{peak}} + 0.697 \left( f_{\text{peak}}/M_{\text{chirp}} \right)^2,$$

(37)

with a maximum residual of 0.872 km and $R^2 = 0.901$. 
Figure 22: Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data sets surfaces. The red points correspond to Bauswein et. al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.

2.4.2.2 Empirical relations for \( R_{1.4} \)

For the \( R_{1.4} \) and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{1.4} = 37.476 - 14.921 M_{\text{chirp}} - 10.202 \frac{f_{\text{peak}}}{M_{\text{chirp}}} \\
+ 3.374 M_{\text{chirp}}^2 + 2.613 f_{\text{peak}} + 0.902 \left( \frac{f_{\text{peak}}}{M_{\text{chirp}}} \right)^2,
\]

with a maximum residual of 0.752 km and \( R^2 = 0.935 \).

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{1.4} = 32.418 - 10.616 M_{\text{chirp}} - 8.151 \frac{f_{\text{peak}}}{M_{\text{chirp}}} \\
+ 2.364 M_{\text{chirp}}^2 + 1.849 f_{\text{peak}} + 0.66 \left( \frac{f_{\text{peak}}}{M_{\text{chirp}}} \right)^2,
\]

with a maximum residual of 0.764 km and \( R^2 = 0.93 \).

2.4.2.3 Empirical relations for \( R_{1.6} \)

For the \( R_{1.6} \) and using the subset of equal-mass configurations, the empirical relation with the smallest error is obtained for the dominant postmerger frequency \( f_{\text{peak}} \):

\[
R_{1.6} = 39.258 - 16.672 M_{\text{chirp}} - 10.784 \frac{f_{\text{peak}}}{M_{\text{chirp}}} \\
+ 3.952 M_{\text{chirp}}^2 + 2.75 f_{\text{peak}} + 0.971 \left( \frac{f_{\text{peak}}}{M_{\text{chirp}}} \right)^2,
\]
with a maximum residual of 0.605 km and $R^2 = 0.962$.

For the whole set of models (including both equal and unequal masses), the empirical relation with the smallest error is obtained for the dominant postmerger frequency $f_{\text{peak}}$:

$$R_{1.6} = 35.442 - 13.46 M_{\text{chirp}} - 9.262 \frac{f_{\text{peak}}}{M_{\text{chirp}}} + 3.118 M_{\text{chirp}}^2 + 2.307 f_{\text{peak}} + 0.758 \left(\frac{f_{\text{peak}}}{M_{\text{chirp}}}\right)^2,$$  \hspace{1cm} (41)

with a maximum residual of 0.654 km and $R^2 = 0.954$. 

---

**Figure 23**: Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data set surfaces. The red points correspond to Bauswein et. al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.

**Figure 24**: Empirical relations for radii using the equal-mass models on the left and the whole set of models of the combined data set on the right. Blue surface is the combined data sets surfaces. The red points correspond to Bauswein et. al. data and green points correspond to frequencies extracted from the CoRe GW catalogue.
Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions just by relying on patterns and inference instead. Machine learning algorithms build a mathematical model based on data provided, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task [Ref Wiki].

### 3.1 Cluster Analysis

Cluster analysis or clustering is the task of grouping a set of objects in a way that the objects of the same group, so called clusters, are more similar to each other than to the objects belonging to the other groups. It is a standard task of exploratory data mining and a common technique of statistical data analysis [Ref wiki].

Cluster analysis can be achieved by various algorithms that differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances between cluster members. The appropriate clustering algorithm, including parameters such as the distance function to use or the number of expected clusters, depend on the data set on wants to cluster.

Clustering belongs to the unsupervised machine learning algorithms that allow us to discover hidden structures in data where we do not know the right answer beforehand. The goal of clustering is to find a natural way to group the data according to some rule, so that items in the same cluster are more similar to each other than to those from different clusters.

#### 3.1.1 DBSCAN

Density-based Spatial Clustering of Applications with Noise (DBSCAN) is a non parametric algorithm that relies on estimating the density of clusters, proposed by Martin Ester, Hans-Peter Kriegel, JÃ¼rg Sander and Xiaowei Xu in 1996 [12]. If the algorithm is provided with a set of points in some arbitrary space, it groups together points that are close, points with many nearby neighbors, and classify as outliers points that are in low-density regions whose nearest neighbors are too far
DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

Figure 25: An illustration how DBSCAN works. Figure taken from [20]

The model introduced by DBSCAN uses a simple minimum density level estimation, based on a threshold for the number of neighbors, let’s call this parameter MinimumPoints, within the radius $\epsilon$ (with an arbitrary distance measure). Objects with more than MinimumPoints neighbors within this radius (including the query point) are considered to be a core point. The intuition of DBSCAN is to find those areas, which satisfy this minimum density, and which are separated by areas of lower density. Instead, all points within the $\epsilon$ radius of a core point are considered to be part of the same cluster as the core point. If any of these neighbors is again a core point, their neighborhoods are transitively included (density reachable). Non-core points in this set are called border points, and all points within the same set are density connected. Points which are not reachable from any core point are considered noise and do not belong to any cluster.

For example in Figure 25 the variable MinimumPoints is set to 4. Point A and the other red points are core points, because the area surrounding these points in an $\epsilon$ radius and contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.
3.1.1.1 The Algorithm

The DBSCAN algorithm can be abstracted into the following steps:

- Find the points in the $\epsilon$ (eps) neighborhood of every point, and identify the core points with more than $MinimumPoints$ neighbors.

- Find the connected components of core points on the neighbor graph, ignoring all non-core points.

- Assign each non-core point to a nearby cluster if the cluster is an $\epsilon$ (eps) neighbor, otherwise assign it to noise.

A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory. The original DBSCAN algorithm does not require this by performing these steps for one point at a time.

Advantages

- DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means.

- DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.

- DBSCAN has a notion of noise, and is robust to outliers.

- DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.

3.1.2 Affinity Propagation

Affinity Propagation is a clustering algorithm based on the concept of “message passing” between data points, proposed by Frey, Brendan J. and Dueck, Delbert in 2007 [13]. Unlike clustering algorithms such as k-means or k-medoids, affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm. This is ideal for our case since we want to detect how many classes our data set has. Similar to k-medoids, affinity propagation finds "exemplars", members of the input set that are representative of clusters.
A common approach is to use data to learn a set of centers such that the sum of squared errors between data points and their nearest centers is small. When the centers are selected from actual data points, they are called “exemplars”.

Affinity propagation is a method that simultaneously considers all data points as potential exemplars. By viewing each data point as a node in a network, we devised a method that recursively transmits real-valued messages along edges of the network until a good set of exemplars and corresponding clusters emerges. As described later, messages are updated on the basis of simple formulas that search for minima of an appropriately chosen energy function. At any point in time, the magnitude of each message reflects the current affinity that one data point has for choosing another data point as its exemplar. Figure 26 illustrates how clusters gradually emerge during the message-passing procedure.

Figure 26: How Affinity Propagation works. Figure taken from [13].

3.1.2.1 The algorithm

Affinity propagation takes as input a collection of real-valued similarities between data points, where the similarity \( s(i,k) \) indicates how well the data point with index \( k \) is suited to be the exemplar for data point \( i \).

There are two kinds of message exchanged between data points, and each takes into account a different kind of competition. Messages can be combined at any stage to decide which points are exemplars and, for every other point, which exemplar it belongs to.

The “responsibility” \( r(i,k) \) sent from data point \( i \) to candidate exemplar point \( k \), reflects the accumulated evidence for how well-suited point \( k \) is to serve
as the exemplar for point $i$, taking into account other potential exemplars for point $i$.

The “availability” $a(i,k)$ sent from candidate exemplar point $k$ to point $i$, reflects the accumulated evidence for how appropriate it would be for point $i$ to choose point $k$ as its exemplar, taking into account the support from other points that point $k$ should be an exemplar.

Figure 27: How message passing procedure works in Affinity Propagation. Figure taken from [13].

### 3.2 Clustering of Bauswein Data

We talked about the classification scheme introduced in [2] in chapter 1. In this chapter we will present our work on classifying the Bauswein data. We used Machine Learning (ML) techniques to cluster and classify the dataset.

#### 3.2.1 Distance and Similarity

As explained in the theoretical introduction about clustering the first step in order to cluster our data is to compute the similarities between all the data. In most datasets using a metric and computing distances is a straight forward task. In our occasion we have to deal with time series and a plain distance between a time series cannot be defined. As a measure of distance we used the scalar

\[ D = 1 - \mathcal{M}, \]

where $\mathcal{M}$ is the match between two time series calculated as

\[ \mathcal{M} = \max_{t_0, \phi_0} \frac{s|h}{\sqrt{(s|s)(h|h)}}, \]
with \((\cdot,\cdot)\) we denote the scalar product
\[
(\bar{s}|h) = 4\text{Re} \int_{f_{low}}^{f_{high}} \tilde{s}(f) \tilde{h}^*(f) \frac{S_n(f)}{S_n(f)} df,
\]
with \(\tilde{s}\) we denote the Fourier transform of \(s\) and with asterisk the conjugate. As \(S_n(f)\) we used the advanced LIGO PSD noise curve which is publicly available. We calculated the distances between all of the GW spectra from \(f_{low} = 1\) kHz to \(f_{high} = 4\) kHz, where the post merger information lies with prospects of detection in the near future, resulting in a matrix \(n \times n\), where \(n\) is the total number of waveforms of our dataset, which is called the distance matrix.

Figure 28: The calculated distance matrix. The matrix is symmetric thus the information lies only in one half of the matrix, either over or under the diagonal. On the x and y axis the numbers correspond to the time series.

In some algorithms we had to use the similarity matrix which is just
\[
\mathcal{S} = 1 - D \Rightarrow \mathcal{S} = \mathcal{M},
\]
which means that the Match \(\mathcal{M}\) itself is a measure of similarity. We must note that the scalar product of Equation 44 is a cross-correlator and the technique of using the cross-correlation of two time series as a measure of distance between them, is popular in the data analysis community.

3.2.2 Clustering Results

We used Spectral Clustering, DBSCAN and Affinity Propagation to cluster the Bauswein data set. Next we will present the results of each method.
3.2.2.1 DBSCAN

In order to implement the DBSCAN algorithm we used the public python3 library Scikit-Learn [18]. We used an $\epsilon = 0.05$ which means that the time series have to be 95% similar to be considered in the same cluster and a minimum of 5 points per cluster.

![Figure 29: Clustering results for the DBSCAN algorithm. The time series data are displayed at $M = M_{\text{tot}}/2$ and the corresponding radius to that mass. With gray lines we indicate the equations of state.](image)

Our results with the DBSCAN algorithm, see Figure 29, agree to the general morphology of the post merger dynamics described in [2]. The noise points could be classified correctly with the help of an additional step of classifying them with an classification algorithm. We must note again that the method does not need any input regarding the number of clusters to search and it detected itself 3 distinct clusters in the Bauswein dataset as predicted by [2].

3.2.2.2 Affinity Propagation

For the implementation of the Affinity Propagation technique we used the public python3 library Scikit-Learn [18]. We used a damping factor of 0.82, which is the extent to which the current value is maintained relative to incoming values (weighted as $1 - \text{damping}$). This in order to avoid numerical oscillations when updating the messages passed. After many trials we observed that a preference value of 0.34 gave the best performance.
Our results using Affinity Propagation are in excellent agreement with the theoretical classification of [2]. We see that the algorithm detects three distinct clusters that correspond to the Type I, II and III of [2]. The morphology of the clusters is the same as in the theoretical picture as well. We must note that we did not put any information about the number of clusters to be detected. Moreover, this result extends the theoretical picture since the theoretical classification was done for equal binary mass configuration models. We performed the clustering on the whole dataset including mass ratios as low as 0.67 and up to 1, proving that this classification holds for unequal binary mass configurations as well. In Figure 30 we see the similarity between the theoretical classification on the left panel and our results on the right panel.
3.3 Classification

Now the problem we have to face is categorizing new data according to the clusters we formed in the previous section with Cluster analysis. The difference between cluster analysis and classification is that the later is a supervised machine learning method whereas the first is an unsupervised. With cluster analysis one can find hidden structures in the dataset and with classification one can classify new data according to the given categories. We found the labels of each class in the previous section using cluster analysis and now we want to find a way of classifying new data, thus we will use some classification technique. Next we will review the two methods we used.

3.3.1 Support Vector Machines

Support Vector Machines or known as SVM is a machine learning technique used for classification [9]. SVM primary performs linear classification but with the use of the kernel trick, which we will see later, SVM can perform non-linear classification [7].

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

3.3.1.1 Kernel Tricks

In its simplest form, the kernel trick means transforming data into another dimension that has a clear dividing margin between classes of data. For example, in Figure 31 the data is transformed with a Gaussian Kernel. We see that with the application of the transformation the data can now be well separated. In our case we used a Polynomial Kernel that is defined as:

\[ K(x, y) = (x^Ty + c)^d \]  

where \( d \) is the degree of the Polynomial Kernel, and \( x, y \) are vectors in the input space.
44 machine learning

Figure 31: Demonstration of the Kernel trick. The data is transformed using a kernel function $\Omega(x)$ and the classification is now performed in the higher dimensional space where the data is clear to divide. Figure taken from [10]

3.3.2 Neural Network models

Neural networks are part of a subsection of machine learning called deep learning. Artificial neural networks are inspired by the biological neural networks that constitute brains. The neural network is not itself a method or an algorithm but rather a framework for many different machine learning algorithms. Such algorithms "learn" to perform tasks by considering examples without being programmed to apply any specific rule. In Figure 32 we can see a basic setup of a neural network with one hidden layer.
3.4 CLASSIFICATION RESULTS

We used the labels we extracted by performing cluster analysis for the Bauswein dataset, to classify new data. Next we will present our results using Support Vector Machines and a Neural Network model.

3.4.1 SVM results

In order to implement the SVM algorithm we used the public python3 library Scikit-Learn [18]. The configuration of the algorithm was a Polynomial kernel with degrees $d = 3, 4$ and 5 and $\gamma$ set to "auto". All the other parameters are set to the default values of the library.

We see in Figure 33 that the algorithm succeeds in classifying the regions according to the labels we provided. The degree of the polynomial kernel does not alter the result much.

3.4.2 Neural Network results

We trained a Neural Network to perform a simple classification. We implemented this procedure with the public python3 library Scikit-Learn [18]. Since our data set is not large we used the classic LBFGS method in order to minimize the loss function.

Figure 32: A basic depiction of a neural network with one hidden layer. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.
Figure 33: Classification results using the Support Vector Machine algorithm. For the first figure we used a polynomial kernel of degree 3. For the second figure we used a polynomial kernel of degree 4 and for the last figure of degree 5. The crosses correspond to Type I, the squares to Type II and the circles to Type III models.

Limited memory BFGS (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm using a limited amount of computer memory. It is a popular algorithm for parameter estimation in machine learning. The algorithm’s target problem is to minimize $f(x)$ over unconstrained values of the real-vector $x$ where $f$ is a differentiable scalar function.

Figure 34: Classification results using a Neural Network model. The crosses correspond to Type I, the squares to Type II and the circles to Type III models.

This result is in close agreement with the SVM case with polynomial kernel of degree 3, see Figure 33 a. These two cases allow a Type II waveform to exist in the low radius regime and hence they are more realistic.
APPENDIX A

A.1 PYTHON CODES

Listing 1: Python code for computing the $R_x = f(f_j, M_{chirp})$ relations using the Bauswein data

```python
#!/usr/bin/python3

#############################################################
## Author : Stamatis Vretinaris ##
## This program fits a surface to datapoints of neutron star characteristics and gives a prediction plus error and Rsquared, adjR squared ##
#############################################################

import numpy as np
import scipy.linalg
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.stats import norm
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist
import scipy.stats
import argparse, sys
import statsmodels
from statsmodels.stats.stattools import durbin_watson
import matplotlib2tikz as tikz
import re
from matplotlib import cm
import os
```
import matplotlib.pyplot as plt
import scipy

plt.rcParams.update({'axes.labelsize': 25})

from mpl_toolkits.mplot3d import Axes3D
np.set_printoptions(suppress=True)

def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0'):
        return False
    else:
        raise argparse.ArgumentTypeError('Boolean value expected. ')

parser=argparse.ArgumentParser()
parser.add_argument('−m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('−m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('−r1', type=float, help='Radius of of 1st Neutron Star')
parser.add_argument('−r2', type=float, help='Radius of of 2nd Neutron Star')
parser.add_argument('−ro', type=float, help='R_x where x is 1.2,1.4,1.6,1.8')
parser.add_argument('−m', type=str, default="1.6",help='This x in ro above')
parser.add_argument('−d', type=int, help='Degree of the polynomy to fit')
parser.add_argument('−p', type=str2bool,default=False,help='To plot or not to plot?!This is the question')
parser.add_argument('−e', type=str2bool,default=False,help='Calculate everything only for equal mass')
parser.add_argument('−f', type=str, default="peak",help='Which frequency to predict')
parser.add_argument('−−report', type=str2bool, default=False,help='Whether to report results or not')
parser.add_argument('−−print', type=str, default='""',help='What results to print for other programs to read,seperate with "," ')
parser.add_argument('−−mr', type=str2bool,default=False,help='Use M/R for axis when −e option is enabled')
parser.add_argument('−fm', type=str2bool,default=False,help='Use freq/Mchirp for z axis')
parser.add_argument('−ex', type=str,default="None",help='Which eos to exclude,format: EOS_1212')
parser.add_argument('−fo', type=float,default=3000,help='What fpeak value for yo ')
A.1 python codes

parser.add_argument('--sep', type=str2bool, default=False, help='If you want to separate the data set with respect to the chirp Mass')
args = parser.parse_args()
toprint = args.print
if ',' in toprint:
toprint = toprint.split(', ,')
else:
toprint = [str(toprint)]

def fitfunc(coeff, x, y, degree, powers):
if powers == False:
poly = PolynomialFeatures(degree)
data = np.stack((x, y), axis=-1)
tmp = poly.fit_transform(data)
return np.dot(tmp, coeff)
else:
b = np.ones(len(x), 1)
for i in range(1, degree + 1):
b = np.hstack((b, x.reshape(len(x), 1)**(i)))
b = np.hstack((b, y.reshape(len(y), 1)**(i)))
return np.dot(b, coeff)

def fitfunc2(p, x, y, degree, powers):
if powers == False:
poly = PolynomialFeatures(degree)
data = np.stack((x, y), axis=-1)
tmp = poly.fit_transform(data)
return np.dot(tmp, np.array(p))
else:
b = np.ones(len(x), 1)
for i in range(1, degree + 1):
b = np.hstack((b, x.reshape(len(x), 1)**(i)))
b = np.hstack((b, y.reshape(len(y), 1)**(i)))
return np.dot(b, np.array(p))

def func(a, x, y, degree, z, powers):
tmp = fitfunc(a, x, y, degree, powers) - z
return tmp.flatten()

def residual(p, x, y, degree, z, powers):
tmp = fitfunc2(p, x, y, degree, powers) - z
return tmp.flatten()
```python
def chirpIt(r1, r2):
    return (r1 * r2)**(3/5) / (r1 + r2)**(1/5)

degree = args.d
only_powers = False
plots = args.p

data = pd.read_table("summary2.csv", header=0, sep=" ",")
data[['fpeak', 'fspiral', 'f2-0', 'Mchirp', 'm1', 'm2', 'r1', 'r2', 'R_1.2', 'R_1.4', 'R_1.6', 'R_1.8']] = data[['fpeak', 'fspiral', 'f2-0', 'Mchirp', 'm1', 'm2', 'r1', 'r2', 'R_1.2', 'R_1.4', 'R_1.6', 'R_1.8']].apply(pd.to_numeric, errors="coerce")

equal_masses = args.e
wrongf = data[data.fpeak < 1800]
if len(wrongf) != 0:
    print("EoS with fpeak<1500 and excluded:")
    print(wrongf[['eos', 'fpeak', 'Mchirp']])
    print("======================================")
ofreq = wrongf['fpeak'].values
for i in ofreq:
    data = data[data.fpeak != i]
rm = args.rm

data = data.dropna(subset=['R_%s' %rm])
#data = data.drop(data[(data['eos']=="ls220") & (data['m1'] == 1.5) & (data['m2'] == 1.5)].index)
data = data.dropna(subset=['f%s' %args.f])

if args.sep == 1:
    if args.rm == "1.2":
        data = data[(data['Mchirp'] < 1.313)]
    if args.rm == "1.4":
        data = data[(data['Mchirp'] < 1.313)]
    if args.rm == "1.8":
        data = data[(data['Mchirp'] > 1.313)]

if args.ex != "None":
    name2ex = args.ex
eos2ex = name2ex.split("_")[0]
masses2ex = name2ex.split("_")[1]
split = re.split("(1|2)", masses2ex)
if "1212" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
    m22ex = float(split[5]+"."+split[7])
```

---

50 Appendix A

```python
def chirpIt(r1, r2):
    return (r1 * r2)**(3/5) / (r1 + r2)**(1/5)

degree = args.d
only_powers = False
plots = args.p

data = pd.read_table("summary2.csv", header=0, sep=" ",")
data[['fpeak', 'fspiral', 'f2-0', 'Mchirp', 'm1', 'm2', 'r1', 'r2', 'R_1.2', 'R_1.4', 'R_1.6', 'R_1.8']] = data[['fpeak', 'fspiral', 'f2-0', 'Mchirp', 'm1', 'm2', 'r1', 'r2', 'R_1.2', 'R_1.4', 'R_1.6', 'R_1.8']].apply(pd.to_numeric, errors="coerce")

equal_masses = args.e
wrongf = data[data.fpeak < 1800]
if len(wrongf) != 0:
    print("EoS with fpeak<1500 and excluded:")
    print(wrongf[['eos', 'fpeak', 'Mchirp']])
    print("======================================")
ofreq = wrongf['fpeak'].values
for i in ofreq:
    data = data[data.fpeak != i]
rm = args.rm

data = data.dropna(subset=['R_%s' %rm])
#data = data.drop(data[(data['eos']=="ls220") & (data['m1'] == 1.5) & (data['m2'] == 1.5)].index)
data = data.dropna(subset=['f%s' %args.f])

if args.sep == 1:
    if args.rm == "1.2":
        data = data[(data['Mchirp'] < 1.313)]
    if args.rm == "1.4":
        data = data[(data['Mchirp'] < 1.313)]
    if args.rm == "1.8":
        data = data[(data['Mchirp'] > 1.313)]

if args.ex != "None":
    name2ex = args.ex
eos2ex = name2ex.split("_")[0]
masses2ex = name2ex.split("_")[1]
split = re.split("(1|2)", masses2ex)
if "1212" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
    m22ex = float(split[5]+"."+split[7])
```
elif "121" in masses2ex:
    m12ex = float(split[1] + "." + split[3])
    m22ex = float(split[5] + "." + split[6])
else:
    m12ex = float(split[1] + "." + split[2])
    m22ex = float(split[3] + "." + split[4])

    f0 = data["f% s" % args.f][(data["eos"] == eos2ex) 
    & (data["m1"] == m12ex) 
    & (data["m2"] == m22ex)].values
    Mchirp0 = data["Mchirp"][(data["eos"] == eos2ex) 
    & (data["m1"] == m12ex) 
    & (data["m2"] == m22ex)].values

    if len(f0) == 0:
        exit()
    else:
        f0 = args.f0
        m1 = args.m1
        m2 = args.m2
        Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)

    freq = args.f
    if equal_masses == True:
        data = data[(data["m1"] == data["m2")]
        x0 = Mchirp0
        z = data["R_ %s" % rm].values
        x = data["Mchirp"].values

        if args.fm == True:
            y = data["f% s" % args.f].values / data["Mchirp"].values
            y0 = f0*10**(-3) / Mchirp0
            y = y*10**(-3)
        else:
            y = data["f% s" % args.f].values
            y0 = f0*10**(-3)
            y = y*10**(-3)

    k = 20000
    a = -k
    b = k

    if only_powers == False:
        dof = int(comb(2+degree, degree))
        init_guess = (b-a) * np.random.random(int(comb(2+degree, degree))) + a
    else:
        dof = int(2*degree+1)
        init_guess = (b-a) * np.ones(4) + a

    p = lmfit.Parameters()
for i in range(int(dof)):
    p.add('b %s' %i, value = init_guess[i], min=a, max=b)

# create Minimizer
mini = lmfit.Minimizer(residual, params=p, fcn_args=(x,y,degree,z,only_powers)
    , nan_policy='omit')
# first solve with Nelder-Mead
out1 = mini.minimize(method='Nelder')
# then solve with Levenberg-Marquardt using the
# Nelder-Mead solution as a starting point
out2 = mini.minimize(method='leastsq', params=out1.params)
if args.report == True:
    lmfit.report_fit(out2, min_correl=0.9)
fitParams = np.array(out2.params)
toprintparams = np.round(fitParams,3)
np.savetxt("./fitparams.txt", toprintparams, delimiter=" ",
residuals1 = z - fitfunc2(fitParams,x,y,degree,only_powers)
durbin = statsmodels.stats.stattools.durbin_watson(residuals1)
n = len(x)
sigma2 = np.dot(residuals1.T,residuals1) / (n-dof-1)
sigma = np.sqrt(sigma2)
SSres1 = np.sum(residuals1**2)
SStot = np.sum((z-z.mean())**2)
R21 = 1 - SSres1/SStot
if only_powers == False:
    p = comb(2+degree,degree) -1
else:
    p = 2*degree
adjR21 = 1-(1-R21) * ((n-1)/(n-p-1))
poly = PolynomialFeatures(degree)
data = np.stack((x,y), axis=-1)
observation = np.array([x0,y0]).reshape(1,-1)
tmp = poly.fit_transform(data)
tmp2 = poly.fit_transform(observation)
X = tmp * fitParams
x0 = tmp2 * fitParams
sey0 = sigma * np.sqrt(1 + np.dot(x0 , np.dot( inv(np.dot(X.T,X)) ,x0.T) ))
alpha = 1-(90/100)
tvalue = tdist.interval(1-alpha/2,n-dof-1,loc=0,scale=1)
confidence_interval_low = tvalue[0] * sey0
confidence_interval_up = tvalue[1] * sey0

if args.report == True:
    print('M1 ="%s", M2 ="%s"
print("R1 ="%s", R2 ="%s"
print("Fitting for f%" %freq)
A.1 python codes

```python
print("R_%s =" %rm,r0)
if only_powers == True:
    if args.fm == True:
        print(np.sum(x0),"+", confidence_interval_up)
    else:
        print(np.sum(x0),"+", confidence_interval_up)
else:
    if args.fm == True:
        print(np.sum(x0),"+", confidence_interval_up[0][0])
    else:
        print(np.sum(x0),"+", confidence_interval_up[0][0])
print("R^2 =",R21 )
print("adjR^2 =",adjR21)
if " " not in toprint:
    if only_powers == True:
        results={"prediction":np.sum(x0),"error": confidence_interval_up,"R^2 ": R21,"adjR^2": adjR21,"mean_residual": np.mean(np.abs(residuals1)),"max_residual":np.max(np.abs(residuals1)),"sigma residual”: np.std(np.abs(residuals1))}
    else:
        results={"fitparams": np.round(fitParams,decimals=3),"mean_residual ": np.round(np.mean(np.abs(residuals1)),3),"prediction":np.round (np.sum(x0),5),"error": np.round(confidence_interval_up[0][0],5) ,"R^2": np.round(R21,3),"adjR^2": np.round(adjR21,3),"max_residual ":round(np.max(np.abs(residuals1)),3),"sigma_residual":np.round( np.std(np.abs(residuals1)),3)}
    for i in toprint:
        print(i," : ",results[i] )
if plots == True:
    zend1 = fitfunc(fitParams,x,y,degree,only_powers)
    data = np.c_[x,y]
    mn = np.min(data, axis=0)
    mx = np.max(data, axis=0)
    zmin = z.min()
    zmax = z.max()
    x_grid = np.linspace(mn[0],mx[0],50)
    y_grid = np.linspace(mn[1],mx[1],50)
    fig = plt.figure(1,figsize=(11,7))
    plt.clf()
    ax = fig.gca(projection='3d')
    for i,j,k,h in zip(x,y,z,zend1):
        ax.plot([i,i],[j,j],[k,h],color = 'black')
    a = fitParams
```
if args.f == "peak":
    x_grid = np.linspace(mn[0], mx[0], 50)
    x_grid2 = np.linspace(mn[0], 1.313, 50)

    y_grid = np.linspace(mn[1], 2.2, 50)
    y_grid2 = np.linspace(2.2, mx[1], 50)

elif args.f == "2−0":
    x_grid = np.linspace(mn[0], mx[0], 50)
    x_grid2 = np.linspace(mn[0], 1.313, 50)

    y_grid = np.linspace(mn[1], 1.6, 50)
    y_grid2 = np.linspace(1.6, mx[1], 50)

elif args.f == "spiral":
    x_grid = np.linspace(mn[0], mx[0], 50)
    x_grid2 = np.linspace(mn[0], 1.313, 50)

    y_grid = np.linspace(mn[1], 1.7, 50)
    y_grid2 = np.linspace(1.7, mx[1], 50)

    x_grid, y_grid = np.meshgrid(x_grid, y_grid)
    x_grid2, y_grid2 = np.meshgrid(x_grid2, y_grid2)

    x_grid * y_grid + a[5] * y_grid**2


    ax.plot_surface(x_grid, y_grid, z_grid, linewidth=0, alpha=0.3, color='skyblue')
    if rm != "1.8":
        ax.plot_surface(x_grid2, y_grid2, z_grid2, linewidth=0, alpha=0.3, color='skyblue')
    ax.scatter(x, y, z, color='red')

if args.mr == True:
    ax.set_xlabel('\$M_{\text{chirp}}/R\$', size=22)
else:
    ax.set_xlabel('\$M_{\text{chirp}} \, [\text{M}_\odot]$', size=22)
ax.zaxis.set_rotate_label(False)

ax.set_zlabel('\$R_{\text{%s}} \, [\text{km}]$\%s' % (rm, size=22))
ax.zaxis.label.set_rotation(0)

if args.fm == True:
    ax.set_ylabel('\$f_{\text{%s}}/M_{\text{chirp}}\, [\text{kHz}/M_\odot]$\%s' % (args.f, size=22))
else:
    ax.set_ylabel('\$f_{\text{%s}}\, [\text{kHz}]$\%s' % (args.f, size=22))
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 30
ax.zaxis.labelpad = 40
ax.zaxis._axinfo[‘juggled’] = (1,2,0)
ax.view_init(elev=37, azim=-11)
#tikz.save("./figures/surface.tex")
if equal_masses == True:
    e = 1
else:
    e = 0
rm = int(float(rm)*10)
plt.savefig(" ./figures/surface_R%s_f%s_e%s.pdf" %(rm,args.f,e),format=’pdf’,dpi=1200)
os.system("pdfcrop figures/surface_R%s_f%s_e%s.pdf figures/surface_R%s_f%s_e%s.pdf" %(rm,args.f,e,rm,args.f,e))
os.system("cp figures/surface_R%s_f%s_e%s.pdf /home/stamatis/Research/Papers/2019/universal_relations/figures" %(rm,args.f,e))

Listing 2: Python code for computing the \( f_j = f(R_x, M_{\text{chirp}}) \) relations using the Bauswein data

#!/usr/bin/python3

import numpy as np
import scipy.linalg
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from astropy.table import Table, Column
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.stats import norm
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist
import scipy.stats
import argparse, sys
np.set_printoptions(suppress=True)
import os
import re
plt.rcParams.update({
    'axes.labelsize': 25})
def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0'):
        return False
else:
    raise argparse.ArgumentTypeError('Boolean value expected. ')
parser=argparse.ArgumentParser()
parser.add_argument('--m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('--m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('--r1', type=float, help='Radius of of 1st Neutron Star')
parser.add_argument('--r2', type=float, help='Radius of of 2nd Neutron Star')
parser.add_argument('--ro', type=float, help='R_x where x is 1.2,1.4,1.6,1.8')
parser.add_argument('--m+', type=str, default="1.6",help='This x in ro above')
parser.add_argument('--d', type=int, help='Degree of the polynomy to fit')
parser.add_argument('--p', type=str2bool,default=False,help='To plot or not to plot?!This is the question')
parser.add_argument('--e', type=str2bool,default=False,help='Calculate everything only for equal mass')
parser.add_argument('--f', type=str, default="peak",help='Which frequency to predict')
parser.add_argument('--report', type=str2bool, default=False,help='Whether to report results or not')
parser.add_argument('--print', type=str, default="",help='What results to print for other programs to read,seperate with ","')
parser.add_argument('--mr', type=str2bool,default=False,help='Use M/R for axis when -e option is enabled')
parser.add_argument('--fm', type=str2bool,default=False,help='Use freq/Mchirp for z axis')
parser.add_argument('--ex', type=str,default="None",help='Which eos to exclude,format: EOS_1212')
args=parser.parse_args()
toprint = args.print
if
    toprint = toprint.split(",")
else:
    toprint = ["%s" %toprint]
if args.ex == "None":
m1 = args.m1
m2 = args.m2
Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)
r0 = args.r0
r1 = args.r1
r2 = args.r2
Rchirp0 = (r1 * r2)**(3/5) / (r1 + r2)**(1/5)

def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
        data = np.stack((x, y), axis=-1)
        tmp = poly.fit_transform(data)
        return np.dot(tmp,np.array(p))
    else:
        b = np.ones((len(x),1))
        for i in range(1,degree+1):
            b = np.hstack((b,x.reshape(len(x),1)**(i)))
            b = np.hstack((b,y.reshape(len(y),1)**(i)))
        return np.dot(b,np.array(p))

def residual(p,x,y,degree,z,powers):
    tmp = fitfunc(p,x,y,degree,powers) - z
    return tmp.flatten()

def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
        data = np.stack((x, y), axis=-1)
        tmp = poly.fit_transform(data)
        return np.dot(tmp,np.array(p))
    else:
        b = np.ones((len(x),1))
        for i in range(1,degree+1):
            b = np.hstack((b,x.reshape(len(x),1)**(i)))
            b = np.hstack((b,y.reshape(len(y),1)**(i)))
        return np.dot(b,np.array(p))

def residual(p,x,y,degree,z,powers):
    tmp = fitfunc(p,x,y,degree,powers) - z
    return tmp.flatten()

def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
        data = np.stack((x, y), axis=-1)
        tmp = poly.fit_transform(data)
        return np.dot(tmp,np.array(p))
    else:
        b = np.ones((len(x),1))
        for i in range(1,degree+1):
            b = np.hstack((b,x.reshape(len(x),1)**(i)))
            b = np.hstack((b,y.reshape(len(y),1)**(i)))
        return np.dot(b,np.array(p))

def residual(p,x,y,degree,z,powers):
    tmp = fitfunc(p,x,y,degree,powers) - z
    return tmp.flatten()
masses2ex = name2ex.split("_")[1]
split = re.split("(1|2)",masses2ex)
if "1212" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
    m22ex = float(split[5]+"."+split[7])
elif "121" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
    m22ex = float(split[5]+"."+split[6])
else:
    m12ex = float(split[1]+"."+split[2])
    m22ex = float(split[3]+"."+split[4])

m1 = m12ex
m2 = m22ex
r0 = data["R_%s" %args.rm][(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].values
Mchirp0 = data["Mchirp"][(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].values

if len(r0)==0:
    print("error in excluding")
    exit()

if freq == "peak":
data = data.dropna(subset=["fpeak"])
if args.fm == True:
z = (data["fpeak"].values / data["Mchirp"]).values/1e3
else:
z = data["fpeak"].values/1e3
elif freq == "spiral":
data = data.dropna(subset=["fspiral"])  
if args.fm == True:
z = (data["fspiral"].values / data["Mchirp"]).values/1e3
else:
z = data["fspiral"].values/1e3
elif freq == "2−0":
data = data.dropna(subset=["f2−0"])  
if args.fm == True:
z = (data["f2−0"].values / data["Mchirp"]).values/1e3
else:
z = data["f2−0"].values/1e3
else:
    print("FrequencyError: Non valid choice for frequency.")
    print("Valid Options:")
    print("1. "peak" 
    "spiral"

else:
    print("2−0")
    print("Valid Options:")
    print("1. "peak" 
    "spiral"

    "2−0")
)
exit()

wrongf = data[data.fpeak < 1800]

if len(wrongf) != 0:
    print("EoS with fpeak<1500 and excluded: ")
    print(wrongf[['eos', 'fpeak', 'Mchirp']])
    ofreq = wrongf['fpeak'].values
    for i in ofreq:
        data = data[data.fpeak != i]

if equal_masses == True and args.mr == True:
    data = data.dropna(subset=['r1'])
    x = data['Mchirp'].values / data['r1'].values
    y = data['R_%s' %rm].values
    x0 = Mchirp0 / r1
    y0 = r0
else:
    x0 = Mchirp0
    y0 = r0

x = data['Mchirp'].values
y = data['R_%s' %rm].values

k = 20000
a = -k
b = k

if only_powers == False:
    dof = int(comb(2+degree,degree))
    init_guess = (b-a) * np.random.random(int(comb(2+degree,degree))) + a
else:
    dof = int(2*degree+1)
    init_guess = (b-a) * np.ones(int(2*degree+1)) + a
p = lmfit.Parameters()
for i in range(int(dof)):
    p.add('b_%s' %i, value = init_guess[i], min=a, max=b)

# create Minimizer
mini = lmfit.Minimizer(residual, params=p, fcn_args=(x, y, degree, z, only_powers)
, nan_policy='omit')
# first solve with Nelder-Mead
out1 = mini.minimize(method='Nelder')
# then solve with Levenberg-Marquardt using the
# Nelder-Mead solution as a starting point
out2 = mini.minimize(method='leastsq', params=out1.params)
fitParams = np.array(out2.params)
toprintparams = np.round(fitParams,3)
np.savetxt("./fitparams.txt", toprintparams, delimiter=",")

if args.fm == True:
    residuals1 = out2.residual*x
else:
    residuals1 = out2.residual

n = len(x)
sigma2 = np.dot(residuals1.T, residuals1) / (n - dof - 1)
sigma = np.sqrt(sigma2)
observation = np.array([x0, y0]).reshape(1, -1)

if only_powers == True:
    tmp = np.ones((len(x), 1))

for i in range(1, degree + 1):
    tmp = np.hstack((tmp, x.reshape(len(x), 1)**(i)))
    tmp = np.hstack((tmp, y.reshape(len(y), 1)**(i)))
    tmp2 = np.hstack((tmp2, x0**(i)))
    tmp2 = np.hstack((tmp2, y0**(i)))

else:
    poly = PolynomialFeatures(degree)
data2 = np.stack((x, y), axis=-1)
tmp = poly.fit_transform(data2)
tmp2 = poly.fit_transform(observation)

X = tmp * fitParams
x0 = tmp2 * fitParams
sey0 = sigma * np.sqrt(1 + np.dot(x0, np.dot(inv(np.dot(X.T, X)), x0.T)))
alpha = 0.05
tvalue = tdist.interval(1 - alpha / 2, n - dof - 1, loc=0, scale=1)
confidence_interval_low = tvalue[0] * sey0
confidence_interval_up = tvalue[1] * sey0
SSres1 = np.sum(residuals1**2)
SStot = np.sum((z - z.mean())**2)
R21 = 1 - SSres1 / SStot

if only_powers == False:
    p = comb(2 + degree, degree) - 1
else:
    p = 2 * degree
adjR21 = 1 - (1 - R21) * ((n - 1) / (n - p - 1))
prediction = np.sum(x0) * Mchirp0
max_resid = np.round(np.max(np.abs(residuals1)), 3)

if args.report == True:
    print("M1 =", m1, " M2 =", m2)
    print("R1 =", r1, " R2 =", r2)
    print("Fitting for f% s" % freq)
    print("R_%s =" % rm, r0)
    if only_powers == True:
        if args.fm == True:
            print(np.sum(x0) * Mchirp0, "+-", confidence_interval_up)
else:
    print(np.sum(x0),"\n", confidence_interval_up)

else:
    if args.fm == True:
        print(np.sum(x0)*Mchirp0,"\n", confidence_interval_up[0][0])
    else:
        print(np.sum(x0),"\n", confidence_interval_up[0][0])
print("R^2 =",R21)

print("adjR^2 =",adjR21)

if "" not in toprint:
    if only_powers == True:
        results={"prediction":np.sum(x0),"error": confidence_interval_up,"R^2": R21,"adjR^2": adjR21}
    else:
        results={"prediction":prediction,"error": confidence_interval_up[0][0],"R^2": np.round(R21,3),"adjR^2": adjR21,"fitparams": np.round(fitParams,3),"mean_residual":np.round(np.mean(np.abs(residuals1)),3),"max_residual": max_resid,"sigma_residual":np.round(np.std(np.abs(residuals1)),3)}

for i in toprint:
    print(i," : ",results[i])

if plots == True:
    zend1 = fitfunc(fitParams,x,y,degree,only_powers)

data = np.c_[x,y]
mn = np.min(data, axis=0)
mx = np.max(data, axis=0)
zmin = z.min()
zmax = z.max()

x_grid = np.linspace(mn[0],mx[0],50)
y_grid = np.linspace(mn[1],mx[1],50)
fig = plt.figure(1,figsize=(11,7))
plt.clf()
ax = fig.gca(projection='3d')
for i,j,k,h in zip(x,y,z,zend1):
    ax.plot([i,i],[j,j],[k,h],color = 'black')
a = fitParams
x_grid = np.linspace(mn[0],1.318,50)
x_grid2 = np.linspace(1.318,mx[0],50)
y_grid = np.linspace(mn[1],mx[1],50)
y_grid2 = np.linspace(13,mx[1],50)
x_grid, y_grid = np.meshgrid(x_grid, y_grid)
x_grid2, y_grid2 = np.meshgrid(x_grid2,y_grid2)
x_grid * y_grid + a[5] * y_grid**2
\[ z_{\text{grid}2} = a[0] + a[1] \times x_{\text{grid2}} + a[2] \times y_{\text{grid2}} + a[3] \times x_{\text{grid2}}^2 + a[4] \times x_{\text{grid2}} \times y_{\text{grid2}} + a[5] \times y_{\text{grid2}}^2 \]

```python
ax.plot_surface(x.grid, y.grid, z.grid, linewidth=0, alpha=0.3, color='skyblue')
ax.plot_surface(x.grid2, y.grid2, z.grid2, linewidth=0, alpha=0.3, color='skyblue')
ax.zaxis.set_major_locator(LinearLocator(5))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.scatter(x, y, z, color='red', depthshade=False)
ax.zaxis.set_rotate_label(False)
if args.mr == True:
    ax.set_xlabel('\$M_{\text{chirp}}/R\$')
else:
    ax.set_xlabel('\$M_{\text{chirp}} \[\text{M}_\odot\]\$')
ax.set_ylabel('\$R_{\%s} \[\text{km}\]\$' %rm)
if args.fm == True:
    ax.set_zlabel('\$f_{\%s}/M_{\text{chirp}}\n\[\text{kHz}/\text{M}_\odot\]\$' %args.f)
else:
    ax.set_zlabel('\$f_{\%s} \[\text{kHz}\]\$' %args.f)
ax.zaxis.label.set_rotation(0)
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20
ax.zaxis.labelpad = 45
ax.zaxis._axinfo['juggled'] = (1, 2, 0)
ax.view_init(elev=37, azim=-11)
if args.e == True:
    e = 1
else:
    e = 0
rm = int(float(rm)*10)
plt.savefig('./figures/surface_fRM_f%sf_R%se%se.pdf' % (args.f, rm, e), format='pdf', dpi=200)
os.system('pdfcrop figures/surface_fRM_f%sf_R%se%se.pdf figures/surface_fRM_f%sf_R%se%se.pdf' % (args.f, rm, e, args.f, rm, e))
os.system('cp figures/surface_fRM_f%sf_R%se%se.pdf /home/stamatis/Research/Papers/2019/universal_relations/figures' % (args.f, rm, e))
```

Listing 3: Python code for computing the \( R_x = f(f_j, M_{\text{chirp}}) \) relations using the CORE data

```bash
#!/usr/bin/python3
```

# Appendix A
# Author: Stamatis Vretinaris

This program fits a surface to datapoints of neutron star characteristics and gives a prediction plus error and Rsquared, adjRsquared.

```
import numpy as np
import scipy.linalg
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from astropy.table import Table, Column
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.stats import norm
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist
import scipy.stats
import argparse, sys
import statsmodels
from statsmodels.stats.stattools import durbin_watson
import matplotlib2tikz as tikz
import re
from matplotlib import cm
import os
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
np.set_printoptions(suppress=True)
def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0'):
        return False
    else:
        raise argparse.ArgumentTypeError('Boolean value expected. ')

parser=argparse.ArgumentParser()
parser.add_argument('--m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('--m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('--r1', type=float, help='Radius of of 1st Neutron Star')
parser.add_argument('--r2', type=float, help='Radius of of 2nd Neutron Star')
parser.add_argument('--ro', type=float, help='R_x where x is 1.2,1.4,1.6,1.8')
```
parser.add_argument('--mn', type=str, default="1.6", help='This x in r0 above')
parser.add_argument('--d', type=int, help='Degree of the polynomy to fit')
parser.add_argument('--p', type=str2bool, default=False, help='To plot or not to plot? This is the question')
parser.add_argument('--e', type=str2bool, default=False, help='Calculate everything only for equal mass')
parser.add_argument('--f', type=str, default="peak", help='Which frequency to predict')
parser.add_argument('--report', type=str2bool, default=False, help='Whether to report results or not')
parser.add_argument('--print', type=str, default='', help='What results to print for other programs to read, separate with ","')
parser.add_argument('--mr', type=str2bool, default=False, help='Use M/R for axis when --e option is enabled')
parser.add_argument('--fm', type=str2bool, default=False, help='Use freq/Mchirp for z axis')
parser.add_argument('--ex', type=str, default="None", help='Which eos to exclude, format: EOS_1212')
parser.add_argument('--fo', type=float, default=3000, help='What fpeak value for yo')

args = parser.parse_args()

toprint = args.print
if "," in toprint:
    toprint = toprint.split(",")
else:
    toprint = ['%s' %toprint]

if args.ex == "None":
    r0 = args.r0
    r1 = args.r1
    r2 = args.r2
    Rchrip0 = (r1 * r2)**(3/5) / (r1 + r2)**(1/5)

    def fitfunc(p,x,y,degree,powers):
        if powers == False:
            poly = PolynomialFeatures(degree)
            data = np.stack((x, y), axis=-1)
            tmp = poly.fit_transform(data)
            return np.dot(tmp, np.array(p))
        else:
            b = np.ones((len(x),1))
            for i in range(1,degree+1):
                b = np.hstack((b, x.reshape(len(x),1)**(i)))
            return np.array(b)
b = np.hstack((b,y.reshape(len(y),1)**(i)))
return np.dot(b,np.array(p))

def residual(p,x,y,degree,z,powers):
    tmp = fitfunc(p,x,y,degree,powers) - z
    return tmp.flatten()
degree = args.d
only_powers = False
plots = args.p
data = pd.read_table("completeCore2.csv",header=0,sep=",")
data["fpeak","Mchirp","m1","m2","f_gw","R1.2","R1.4","R1.6","R1.8"] = data["fpeak","Mchirp","m1","m2","f_gw","R1.2","R1.4","R1.6","R1.8"].apply(pd.to_numeric,errors="coerce")
equal_masses = args.e
wrongf = data[data.fpeak < 1800]
if len(wrongf) != 0:
    print("EoS with fpeak<1800 and excluded:")
    print(wrongf["eos","fpeak","Mchirp"])
    print("====================================")
ofreq = wrongf["fpeak"].values
for i in ofreq:
    data = data[data.fpeak != i]
data = data[data["Mchirp"]<1.3]
rm =args.rm
data = data.dropna(subset="R %s" %rm)
#data = data.drop(data[(data["eos"]=="ls220") & (data["m1"] == 1.5) & (data["m2"] == 1.5)].index)
data = data.dropna(subset="f% s" %args.f)
if args.ex != "None":
    name2ex = args.ex
eos2ex = name2ex.split("_")[0]
masses2ex = name2ex.split("_")[1]
split = re.split("(1|2)"),masses2ex
if "1212" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
m22ex = float(split[5]+"."+split[7])
elif "121" in masses2ex:
    m12ex = float(split[1]+"."+split[3])
m22ex = float(split[5]+"."+split[6])
else:
    m12ex = float(split[1]+"."+split[2])
m22ex = float(split[3]+"."+split[4])
f0 = data["f% s" %args.f][(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].values
Mchirp0 = data['Mchirp'][(data['eos'] == eos2ex) & (data['m1'] == m12ex) & (data['m2'] == m22ex)].values

data = data.drop(data[(data['eos'] == eos2ex) & (data['m1'] == m12ex) & (data['m2'] == m22ex)].index)

if len(f0)==0:
    exit()
else:
    f0 = args.f0
    m1 = args.m1
    m2 = args.m2

Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)

freq = args.f

if equal_masses == True:
    data = data[(data['m1'] == data['m2'])]

x0 = Mchirp0

z = data['R % s' %rm].values
x = data['Mchirp'].values

if args.fm == True:
    y = data[' f% s' %args.f].values / data['Mchirp'].values
    y0 = f0*10**(y3) / Mchirp0
else:
    y = data[' f% s' %args.f].values
    y0 = f0*10**(y3)

k = 20000
a = -k
b = k

if only_powers == False:
    dof = int(comb(2+degree,degree))
    init_guess = (b-a) * np.random.random(int(comb(2+degree,degree))) + a
else:
    dof = int(2*degree+1)
    init_guess = (b-a) * np.ones(4) + a

p = lmfit.Parameters()

for i in range(int(dof)):
    p.add('b % s' %i, value = init_guess[i], min=a, max=b)

    mini = lmfit.Minimizer(residual,params=p,fcn_args=(x,y,degree,z,only_powers), nan_policy='omit')
    out1 = mini.minimize(method='Nelder-Mead')
    # then solve with Levenberg-Marquardt using the
    # Nelder-Mead solution as a starting point
    out2 = mini.minimize(method='leastsq', params=out1.params)
if args.report == True:
    lmfit.report_fit(out2, min_correl=0.9)
    fitParams = np.array(out2.params)
    toprintparams = np.round(fitParams,3)
    np.savetxt("./fitparams.txt",toprintparams,delimiter=",")
residuals1 = z - fitfunc2(fitParams,x,y,degree,only_powers)

n = len(x)
sigma2 = np.dot(residuals1.T,residuals1) / (n-dof-1)
sigma = np.sqrt(sigma2)
SSres1 = np.sum(residuals1**2)
SStot = np.sum((z-z.mean())**2)
R21 = 1 - SSres1/SStot
if only_powers == False:
    p = comb(2+degree,degree) -1
else:
    p = 2*degree
adjR21 = 1-(1-R21) * ((n-1)/(n-p-1))

poly = PolynomialFeatures(degree)
data = np.stack((x,y), axis=-1)
observation = np.array([x0,y0]).reshape(1,-1)
tmp = poly.fit_transform(data)
tmp2 = poly.fit_transform(observation)
X = tmp * fitParams
x0 = tmp2 * fitParams
sey0 = sigma * np.sqrt(1+ np.dot(x0 , np.dot( inv(np.dot(X.T,X)) ,x0.T) ) )

alpha = 0.05
tvalue = tdist.interval(1-alpha/2,n-dof-1,loc=0,scale=1)
confidence_interval_low = tvalue[0] * sey0
confidence_interval_up = tvalue[1] * sey0
if args.report == True:
    print("M1 =",m1," M2 =",m2)
    print("R1 =",r1," R2 =",r2)
    print("Fitting for f%" %freq)
    print("R_ %s =" %rm,r0)
    if only_powers == True:
        if args.fm == True:
            print(np.sum(x0),"+-", confidence_interval_up)
        else:
            print(np.sum(x0),"+-", confidence_interval_up)
    else:
        if args.fm == True:
            print(np.sum(x0),"+-", confidence_interval_up[0][0])
        else:
            print(np.sum(x0),"+-", confidence_interval_up[0][0])
print("R^2 =",R21 )
print("adjR^2 =",adjR21)
if "" not in toprint:
    if only_powers == True:
        results={"prediction":np.sum(x0),"error": confidence_interval_up,"R^2": R21,"adjR^2": adjR21,"mean_residual": np.mean(np.abs(residuals1)),"max_residual":np.max(np.abs(residuals1)),"sigma residual":np.std(np.abs(residuals1))}
else:
    results={"fitparams": np.round(fitParams,decimals=3),"mean_residual ": np.round(np.mean(np.abs(residuals1)),3),"prediction":np.round(np.sum(x0),5),"R^2": np.round(R21,3),"adjR^2": np.round(adjR21,3),"max_residual ":round(np.max(np.abs(residuals1)),3),"sigma_residual":np.round(np.std(np.abs(residuals1)),3)}
for i in toprint:
    print(i," : ",results[i] )
if plots == True:
    fig = plt.figure(1,figsize=(11,7))
    plt.clf()
    ax = fig.gca(projection='3d')
    os.system("rm fitparams . txt ")
    if args.e == True:
        os.system('./surface_fit.py -m1 1.7 -m2 1.7 -r1 14.171627 -r2 14.171627 -ro 14.259572279568 -d 2 -f peak -m %6 -e 1 -fm 1 - sep 1 ' %rm)
    else:
        os.system('./surface_fit.py -m1 1.7 -m2 1.7 -r1 14.171627 -r2 14.171627 -ro 14.259572279568 -d 2 -f peak -m %6 -e 0 -fm 1 - sep 1 ' %rm)
    a = np.loadtxt("fitparams .txt",delimiter="\n")
data = np.c_[x,y]
mn = np.min(data, axis=0)
mx = np.max(data, axis=0)
zmin = z.min()
zmax = z.max()
x_grid = np.linspace(mn[0],mx[0],50)
y_grid = np.linspace(mn[1],mx[1],50)
x_grid, y_grid = np.meshgrid(x_grid, y_grid)
ax.plot_surface(x_grid, y_grid , z_grid_andi ,cstride=8, alpha = 0.3, linewidth=0,color="skyblue")
data = np.c_[x,y]
mn = np.min(data, axis=0)
mx = np.max(data, axis=0)
zmin = z.min()
zmax = z.max()
x_grid = np.linspace(mn[0], mx[0], 50)
y_grid = np.linspace(mn[1], mx[1], 50)
zend1 = fitfunc2(a, x, y, 2, only_powers)
AndiCoreResidual = z - zend1
max_ACR = np.max(AndiCoreResidual)
print("max residual (andi-core): ", max_ACR)
for i, j, k, h in zip(x, y, z, zend1):
    ax.plot([i, i], [j, j], [k, h], color = 'black ')
x_grid, y_grid = np.meshgrid(x_grid, y_grid)
if degree == 1:
y_grid
elif degree == 2:
y_grid + fitParams[5] * y_grid**2
ax.scatter(x, y, z, color="green")
if args.mr == True:
    ax.set_xlabel('$M_{\mathrm{ chirp}}/R$ $[\mathrm{M _ \odot}/\mathrm{k m }]$ ',size=22)
else:
    ax.set_xlabel('$M_{\mathrm{ chirp}}$ $[\mathrm{M _ \odot}]$ ',size=22)
ax.zaxis.set_rotate_label(False)
ax.set_zlabel("$R_{\mathrm {%s}}$ $[\mathrm{k m}]$ " %rm,size=22)
ax.zaxis.label.set_rotation(0)
if args.fm == True:
    ax.set_ylabel("$f_{\mathrm {%s}}/M _{\mathrm{ chirp}}$ $[\mathrm{kHz}/\mathrm{M _ \odot}]$" %args.f,size=22)
else:
    ax.set_ylabel("$f_{\mathrm {%s}}$ $[\mathrm{kHz}]$" %args.f,size=22)
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 30
ax.zaxis.labelpad = 45
ax.xaxis._axinfo['juggled '] = (1,2,0)
ax.view_init(elev=37, azim=-11)
#tikz.save("./figures/surface.tex")
rm = int(float(rm)*10)
if args.e == True:
    e = 1
else:
    e = 0
Listing 4: Python code for computing the $f_j = f(R_x, M_{\text{chirp}})$ relations using the CORE data

```python
#!usr/bin/python3

import numpy as np
import scipy.linalg
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from astropy.table import Table, Column
from matplotlib.ticker import LinearLocator, FormatStrFormatter

from scipy.stats import norm
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist

import scipy.stats
import argparse, sys
np.set_printoptions(suppress=True)
import os
plt.rcParams.update({'axes.labelsize': 25})

def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1', 'True'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0', 'False'):
```
return False

else:
    raise argparse.ArgumentTypeError('Boolean value expected. ')

parser=argparse.ArgumentParser()
parser.add_argument('--m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('--m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('--r1', type=float, help='Radius of 1st Neutron Star')
parser.add_argument('--r2', type=float, help='Radius of 2nd Neutron Star')
parser.add_argument('--r0', type=float, help='R_x where x is 1.2, 1.4, 1.6, 1.8')
parser.add_argument('--m', type=str, default='1.6', help='This x in r0 above')
parser.add_argument('--d', type=int, help='Degree of the polynomial to fit')
parser.add_argument('--p', type=str2bool,default=False,help='To plot or not to plot?! This is the question')
parser.add_argument('--e', type=str2bool,default=False,help='Calculate everything only for equal mass')
parser.add_argument('--f', type=str, default='peak',help='Which frequency to predict')
parser.add_argument('--report', type=str2bool, default=False,help='Whether to report results or not')
parser.add_argument('--print', type=str, default='',help='What results to print for other programs to read, separate with ",")
parser.add_argument('--mr', type=str2bool,default=False,help='Use M/R for axis when --e option is enabled')
parser.add_argument('--fm', type=str2bool,default=False,help='Use freq/Mchirp for z axis')

args=parser.parse_args()
toprint = args.print
if ',' in toprint:
    toprint = toprint.split(',')
else:
    toprint = ['%' + toprint]
m1 = args.m1
m2 = args.m2
Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)
r0 = args.r0
r1 = args.r1
r2 = args.r2
Rchrip0 = (r1 * r2)**(3/5) / (r1 + r2)**(1/5)
def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
data = np.stack((x, y), axis=-1)
tmp = poly.fit_transform(data)
```python
return np.dot(tmp, np.array(p))
else:
b = np.ones((len(x), 1))
for i in range(1, degree + 1):
b = np.hstack((b, x.reshape(len(x), 1)**(i)))
return np.dot(b, np.array(p))

def residual(p, x, y, degree, z, powers):
tmp = fitfunc(p, x, y, degree, powers) - z
return tmp.flatten()

def fitfunc(p, x, y, degree, powers):
tmp = np.dot(x, np.array(p))**degree
for i in range(1, degree + 1):
tmp = np.hstack((tmp, x.reshape(len(x), 1)**i))
return np.dot(tmp, np.array(p))

def residual(p, x, y, degree, z, powers):
tmp = fitfunc(p, x, y, degree, powers) - z
return tmp.flatten()

def fitfunc(p, x, y, degree, powers):
tmp = np.dot(x, np.array(p))**degree
for i in range(1, degree + 1):
tmp = np.hstack((tmp, x.reshape(len(x), 1)**i))
return np.dot(tmp, np.array(p))

def residual(p, x, y, degree, z, powers):
tmp = fitfunc(p, x, y, degree, powers) - z
return tmp.flatten()
```

```python
def fitfunc(p, x, y, degree, powers):
tmp = np.dot(x, np.array(p))**degree
for i in range(1, degree + 1):
tmp = np.hstack((tmp, x.reshape(len(x), 1)**i))
return np.dot(tmp, np.array(p))
```
x0 = Mchirp0
y0 = r0
x = data['Mchirp'].values
y = data['Rms % rms'].values
k = 20000
a = -k
b = k
if only_powers == False:
    dof = int(comb(2+degree,degree))
    init_guess = (b-a) * np.random.random(int(comb(2+degree,degree))) + a
else:
    dof = int(2*degree+1)
    init_guess = (b-a) * np.ones(int(2*degree+1)) + a
p = lmfit.Parameters()
for i in range(int(dof)):
    p.add('b %s ' %i, value = init_guess[i], min=a, max=b)
# create Minimizer
mini = lmfit.Minimizer(residual, params=p, fcn_args=(x,y,degree,z,only_powers), nan_policy='omit')
# first solve with Nelder-Mead
out1 = mini.minimize(method='Nelder')
# then solve with Levenberg-Marquardt using the
# Nelder-Mead solution as a starting point
out2 = mini.minimize(method='leastsq', params=out1.params)
fitParams = np.array(out2.params)
toprintparams = np.round(fitParams,3)
np.savetxt('./fitparams.txt', toprintparams, delimiter=',')
residuals1 = z - fitfunc2(fitParams,x,y,degree,only_powers)
residuals1 = out2.residual * x0
n = len(x)
sigma2 = np.dot(residuals1.T,residuals1) / (n-dof-1)
sigma = np.sqrt(sigma2)
observation = np.array([x0,y0]).reshape(1,-1)
if only_powers == True:
    tmp = np.ones((len(x),1))
    tmp2 = np.ones(1)
    for i in range(1,degree+1):
        tmp = np.hstack((tmp,x.reshape(len(x),1)**(i)))
        tmp2 = np.hstack((tmp2,x0**(i)))
else:
    poly = PolynomialFeatures(degree)
data = np.stack((x,y), axis=-1)
tmp = poly.fit_transform(data)
tmp2 = poly.fit_transform(observation)
X = tmp * fitParams
x0 = tmp2 * fitParams
sey0 = sigma * np.sqrt(1 + np.dot(x0, np.dot(inv(np.dot(X.T, X)), x0.T)))

alpha = 0.05
tvalue = tdist.interval(1-alpha/2, n-dof-1, loc=0, scale=1)
confidence_interval_low = tvalue[0] * sey0
confidence_interval_up = tvalue[1] * sey0
SSres1 = np.sum(residuals1**2)
SStot = np.sum((z-z.mean())**2)
R21 = 1 - SSres1/SStot
n = len(x)
if only_powers == False:
p = comb(2+degree, degree) - 1
else:
p = 2*degree
adjR21 = 1-(1-R21) * ((n-1)/(n-p-1))
if args.report == True:
    print("M1 =", m1, " M2 =", m2)
    print("R1 =", r1, " R2 =", r2)
    print("Fitting for f%" %freq)
    print("R^2 =" %rm, r0)
if only_powers == True:
    if args.fm == True:
        print(np.sum(x0)*Mchirp0 ,"+-", confidence_interval_up)
    else:
        print(np.sum(x0),"+-", confidence_interval_up)
else:
    if args.fm == True:
        print(np.sum(x0)*Mchirp0,"+-", confidence_interval_up[0][0])
    else:
        print(np.sum(x0),"+-", confidence_interval_up[0][0])
print("R^2 =",R21)
print("adjR^2 =",adjR21)
if "" not in toprint:
    if only_powers == True:
        results={"prediction":np.sum(x0)*Mchirp0,"error":
            confidence_interval_up,"R2": R21,"adjR2": adjR21}
    else:
        results={"prediction":np.sum(x0)*Mchirp0,"error":
            confidence_interval_up[0][0],"R2": np.round(R21,3),"adjR2":
            adjR21,"fitparams": np.round(fitParams,3),"mean_residual":np.
            round(np.mean(np.abs(residuals1)),3),"max_residual":np.
            round(np.max(np.abs(residuals1)),3),"sigma_residual":np.
            round(np.std(np.abs(residuals1)),3)}
for i in toprint:
    print(i,": ", results[i])

if plots == True:
    fig = plt.figure(1, figsize=(11, 7))
    plt.clf()

    ax = fig.gca(projection='3d')
    data = np.c_[x, y]
    mn = np.min(data, axis=0)
    mx = np.max(data, axis=0)
    zmin = z.min()
    zmax = z.max()

    x_grid = np.linspace(mn[0], mx[0], 50)
    y_grid = np.linspace(mn[1], mx[1], 50)
    x_grid, y_grid = np.meshgrid(x_grid, y_grid)

    os.popen("./surface_fit_fj.py -m1.7 -m2.17 -r1 14.171627 -r2 14.171627 -ro 14.259572279568 -d 2 -fm 1 -p 0 -mn %s -e %s -f peak" % (rm, args.e)).read().strip("\n")

    a = np.loadtxt("fitparams.txt")

    ax.plot_surface(x_grid, y_grid, z_gridandi, cstride=8, alpha=0.3, linewidth=0, color="skyblue")
    zend1 = fitfunc2(a, x, y, 2, only_powers)
    AndiCoreResidual = z - zend1

    max_ACR = np.max(AndiCoreResidual)

    for i, j, k, h in zip(x, y, z, zend1):
        ax.plot([i, i], [j, j], [k, h], color='black')

    if args.d == 1:
    elif args.d == 2:

    ax.zaxis.set_major_locator(LinearLocator(5))
    ax.zaxis.set_major_formatter(FormatStrFormatter('%0.1f'))
    ax.scatter(x, y, z, color="green", depthshade=False)
    ax.set_zlabel(False)

    if args.mr == True:
        ax.set_xlabel('$M_{\text{chirp}}/R$')
    else:
        ax.set_xlabel('$M_{\text{chirp}}/[\text{M}_\odot]$')

    ax.set_ylabel('$R_{\text{%s}}/[\text{k m}]$' % rm)
    ax.set_xlabel('$R_{\text{%s}}/[\text{M}_\odot]$' % rm)

    if args.fm == True:
Listing 5: Python code for computing the $R_x = f(f_j, M_{\text{chirp}})$ relations using the combined dataset

```python
#!/usr/bin/python3

# Author: Stamatis Vretinaris
# This program fits a surface to datapoints of neutron star characteristics and gives a prediction plus error and Rsquared, adjRsquared

import numpy as np
import scipy.linalg
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from astropy.table import Table, Column
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.stats import norm
```
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist
import scipy.stats
import argparse, sys
import statsmodels
from statsmodels.stats.stattools import durbin_watson
import matplotlib2tikz as tikz
import re
from matplotlib import cm
import os
import matplotlib.pyplot as plt
plt.rcParams.update({'axes.labelsize': 25})
from mpl_toolkits.mplot3d import Axes3D
np.set_printoptions(suppress=True)

def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0'):
        return False
    else:
        raise argparse.ArgumentTypeError('Boolean value expected.')

parser=argparse.ArgumentParser()
parser.add_argument('--m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('--r1', type=float, help='Radius of of 1st Neutron Star')
parser.add_argument('--r2', type=float, help='Radius of of 2nd Neutron Star')
parser.add_argument('--ro', type=float, help='R_x where x is 1.2, 1.4, 1.6, 1.8')
parser.add_argument('--m', type=str, default="1.6", help='This x in ro above')
parser.add_argument('--d', type=int, help='Degree of the polyonym to fit')
parser.add_argument('--p', type=str2bool, default=False, help='To plot or not to plot?! This is the question')
parser.add_argument('--e', type=str2bool, default=False, help='Calculate everything only for equal mass')
parser.add_argument('--f', type=str, default="peak", help='Which frequency to predict')
parser.add_argument('--report', type=str2bool, default=False, help='Whether to report results or not')
parser.add_argument('--print', type=str, default="", help='What results to print for other programs to read, seperate with "," ')
parser.add_argument(‘−mr’, type=str2bool,default=False,help=’Use M/R for axis when −e option is enabled’)
parser.add_argument(‘−fm’, type=str2bool,default=False,help=’Use freq/Mchirp for z axis’)
parser.add_argument(‘−ex’, type=str,default=”None”,help=’Which eos to exclude, format: EOS_1212’)
parser.add_argument(‘−fo’, type=float,default=3000,help=’What fpeak value for yo’)
parser.add_argument(‘−sep’, type=str2bool,default=False,help=’If you want to seperate the data set with respect to the chirp Mass’)
args=parser.parse_args()
toprint = args.print
if ” , ” in toprint:
toprint = toprint.split(”,")
else:
toprint = [”%s ” %toprint]
def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
        data = np.stack((x, y), axis=-1)
        tmp = poly.fit_transform(data)
        return np.dot(tmp,np.array(p))
    else:
        b = np.ones((len(x),1))
        for i in range(1,degree+1):
            b = np.hstack((b,x.reshape(len(x),1)**(i)))
            b = np.hstack((b,y.reshape(len(y),1)**(i)))
        return np.dot(b,np.array(p))

def residual(p,x,y,degree,z,powers):
    tmp = fitfunc(p,x,y,degree,powers) - z
    return tmp.flatten()

degree = args.d
only_powers = False
plots = args.p
data1 = pd.read_table(”summary2.csv”,header=0,sep=”,")
data1 = data1.rename(columns={”R_1.2” : ”R1.2”,”R_1.4” : ”R1.4”,”R_1.6” : ”R1.6”,”R_1.8” : ”R1.8”})
data1[[”fpeak”,”fspiral”,”f2−0”,”Mchirp”,”m”,”m2”,”r1”,”r2”,”R1.2”, ”R1.4”,”R1.6”,”R1.8”]] = data1[[”fpeak”,”fspiral”,”f2−0”,”Mchirp”,”m”,”m2”,”r1”,”r2”,”R1.2”, ”R1.4”,”R1.6”,”R1.8”]].apply(pd.to_numeric,errors=”coerce ”)
data2 = pd.read_table(”completeCore2.csv”,header=0,sep=”,")
data2[["fpeak", "Mchirp", "m1", "m2", "f_gw", "R1.2", "R1.4", "R1.6", "R1.8"]]
= 
data2[["fpeak", "Mchirp", "m1", "m2", "f_gw", "R1.2", "R1.4", "R1.6", "R1.8"]].

apply(pd.to_numeric, errors="coerce")
data1 = data1[["eos", "fpeak", "Mchirp", "m1", "m2", "R1.2", "R1.4", "R1.6", "R1.8"]]
data = data1.append(data2)

equal_masses = args.e

if len(wrongf) != 0:
    if args.rm == 1.2:
        data = data[(data["Mchirp"] < 1.313)]
    if args.rm == 1.4:
        data = data[(data["Mchirp"] < 1.313)]
    if args.rm == 1.8:
        data = data[(data["Mchirp"] > 1.313)]

if args.ex != "None":
    name2ex = args.ex
    eos2ex = name2ex.split("_")[0]
    masses2ex = name2ex.split("_")[1]

split = re.split("([1-2])", masses2ex)
if "1212" in masses2ex:
    m12ex = float(split[1]) + "." + split[3]
    m22ex = float(split[5]) + "." + split[7]
elif "121" in masses2ex:
    m12ex = float(split[1]) + "." + split[3]
    m22ex = float(split[5]) + "." + split[6]
else:
    m12ex = float(split[1]) + "." + split[2]
    m22ex = float(split[3]) + "." + split[4]

f0 = data["f%6" % args.f][(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].values
Mchirp0 = data["Mchirp"][(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].values
data = data.drop(data[(data["eos"] == eos2ex) & (data["m1"] == m12ex) & (data["m2"] == m22ex)].index)

if len(f0)==0:
    exit()
else:
    f0 = args.f0
    m1 = args.m1
    m2 = args.m2

Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)

freq = args.f
if equal_masses == True:
data = data[(data["m1"] == data["m2"])]
x0 = Mchirp0

z = data["R % s" %rm].values
z1 = data1["R % s" %rm].values
z2 = data2["R % s" %rm].values
x = data["Mchirp"].values
x1 = data1["Mchirp"].values
x2 = data2["Mchirp"].values

if args.fm == True:
y = data["f% s" %args.f].values / data["Mchirp"].values
y0 = f0*10**(-3) / Mchirp0
y = y*10**(-3)
y1 = (data1["f% s" %args.f].values / data1["Mchirp"].values)*10**(-3)
y2 = (data2["f% s" %args.f].values / data2["Mchirp"].values)*10**(-3)
else:
y = data["f% s" %args.f].values
y0 = f0*10**(-3)
y = y*10**(-3)
y1 = (data1["f% s" %args.f].values)*10**(-3)
y2 = (data2["f% s" %args.f].values)*10**(-3)

k = 20000
a = -k
b = k

if only_powers == False:
dof = int(comb(2+degree,degree))
init_guess = (b-a) * np.random.random(int(comb(2+degree,degree))) + a
else:
dof = int(2*degree+1)
init_guess = (b-a) * np.ones(4 ) + a
p = lmfit.Parameters()
for i in range(int(dof)):
p.add("f% s" %i, value = init_guess[i], min=a, max=b)
# create Minimizer
mini = lmfit.Minimizer(residual, params=p, fcn_args=(x, y, degree, z, only_powers)
  , nan_policy='omit')

# first solve with Nelder-Mead
out1 = mini.minimize(method='Nelder')

# then solve with Levenberg-Marquardt using the Nelder-Mead solution as a starting point
out2 = mini.minimize(method='leastsq', params=out1.params)

if args.report == True:
    lmfit.report_fit(out2, min_correl=0.9)

fitParams = np.array(out2.params)

toprintparams = np.round(fitParams, 3)
np.savetxt("./fitparams.txt", toprintparams, delimiter=" ", newline=" 
")
residuals1 = z - fitfunc2(fitParams, x, y, degree, only_powers)
durbin = statsmodels.stats.stattools.durbin_watson(residuals1)

n = len(x)
sigma2 = np.dot(residuals1.T, residuals1) / (n - dof - 1)
sigma = np.sqrt(sigma2)
SSres1 = np.sum(residuals1 ** 2)
SStot = np.sum(((z - z.mean()) ** 2)
R21 = 1 - SSres1/SStot

if only_powers == False:
p = comb(2 + degree, degree) - 1
else:
p = 2 * degree
adjR21 = 1-(1-R21) * ((n-1)/(n-p-1))

poly = PolynomialFeatures(degree)
data = np.stack((x, y), axis=-1)
observation = np.array([x0, y0]).reshape(1, -1)
tmp = poly.fit_transform(data)
tmp2 = poly.fit_transform(observation)

X = tmp * fitParams
x0 = tmp2 * fitParams
sey0 = sigma * np.sqrt(1 + np.dot(x0, np.dot(inv(np.dot(X.T, X)), x0.T)))
alpha = 0.05
tvalue = tdist.interval(1-alpha/2, n-dof-1, loc=0, scale=1)

confidence_interval_low = tvalue[0] * sey0
confidence_interval_up = tvalue[1] * sey0

if args.report == True:
    print("M1 =", m1, " M2 =", m2)
    print("R1 =", r1, " R2 =", r2)
print("Fitting for f%" %freq)
print("R%" %rm, r0)
if only_powers == True:
    if args.fm == True:
print(np.sum(x0), "+-", confidence_interval_up)

else:
    print(np.sum(x0), "+-", confidence_interval_up)
else:
    if args.fm == True:
        print(np.sum(x0), "+-", confidence_interval_up[0][0])
    else:
        print(np.sum(x0), "+-", confidence_interval_up[0][0])

print("R^2 =", R21)
print("adjR^2 =", adjR21)
if "" not in toprint:
    if only_powers == True:
        results = {"prediction": np.sum(x0), "error": confidence_interval_up, "R2": R21, "adjR2": adjR21, "mean_residual": np.mean(np.abs(residuals1)), "max_residual": np.max(np.abs(residuals1)), "sigma residual": np.std(np.abs(residuals1))}
    else:
        results = {"fitparams": np.round(fitParams, decimals=3), "mean_residual": np.round(np.mean(np.abs(residuals1)), 3), "prediction": np.round(np.sum(x0), 5), "error": np.round(confidence_interval_up[0][0], 5), "R2": np.round(R21, 3), "adjR2": np.round(adjR21, 3), "max_residual": np.round(np.max(np.abs(residuals1)), 3), "sigma_residual": np.round(np.std(np.abs(residuals1)), 3)}
    for i in toprint:
        print(i, ":", results[i])

if plots == True:
    zend1 = fitfunc(fitParams, x, y, degree, only_powers)
    data = np.c_[x, y]
    mn = np.min(data, axis=0)
    mx = np.max(data, axis=0)
    zmin = z.min()
    zmax = z.max()
    x_grid = np.linspace(mn[0], mx[0], 50)
    y_grid = np.linspace(mn[1], y_floor, 50)
    fig = plt.figure(1, figsize=(11, 7))
    plt.clf()
    ax = fig.gca(projection='3d')
    for i, j, k, h in zip(x_grid, y_grid, z_grid, zend1):
        ax.plot([i, i], [j, j], [k, h], color = 'black')
    a = fitParams
    x_grid = np.linspace(mn[0], mx[0], 50)
    x_grid2 = np.linspace(mn[0], 1.313, 50)
    y_grid = np.linspace(mn[1], 1.25, 50)
    y_grid2 = np.linspace(2.2, mx[1], 50)
    x_grid, y_grid = np.meshgrid(x_grid, y_grid)
x_grid2, y_grid2 = np.meshgrid(x_grid2, y_grid2)


ax.plot_surface(x_grid, y_grid, z_grid, linewidth=0, alpha=0.3, color='indigo')

if rm != "1.8":  
    ax.plot_surface(x_grid2, y_grid2, z_grid2, linewidth=0, alpha=0.3, color='indigo')

# 1 -> Andi data
# 2 -> Core data

if args.mr == True:
    ax.set_xlabel('$M_{\mathrm{chirp}}/R$', size=22)
else:
    ax.set_xlabel('$M_{\mathrm{chirp}} \[
\mathrm{M _ \odot} \]$', size=22)

ax.zaxis.set_rotate_label(False)

ax.set_zlabel('$R_{\mathrm{%s}} \[
\mathrm{km} \]$',%rm,size=22)

if args.fm == True:
    ax.set_ylabel('$f_{\mathrm{%s}}/M_{\mathrm{chirp}} \[
\mathrm{kHz}/\mathrm{M _ \odot} \]$', args.f,size=22)
else:
    ax.set_ylabel('$f_{\mathrm{%s}}$' %args.f,size=22)

ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20
ax.zaxis.labelpad = 40

ax.zaxis._axinfo[\'juggled\'] = (1,2,0)

ax.view_init(elev=37, azim=-11)

#tikz.save("./figures/surface.tex")

if equal_masses == True:
    e = 1
else:
    e = 0

rm = int(float(rm)*10)

plt.savefig("./figures/surfaceCombined_R%s_fpeak_e%s.pdf"%(rm,e),format="pdf",dpi=1200)

os.system("pdfcrop figures/surfaceCombined_R%s_fpeak_e%s.pdf figures/surfaceCombined_R%s_fpeak_e%s.pdf"%(rm,e,rm,e))

os.system("cp figures/surfaceCombined_R%s_fpeak_e%s.pdf /home/stamatis/Research/Papers/2019/universal_relations/figures"%(rm,e))
Listing 6: Python code for computing the $f_j = f(R_x, M_{chirp})$ relations using the combined dataset

```python
#!/usr/bin/python3

# Author: Stamatis Vretinaris
# This program fits a surface to datapoints of neutron star characteristics and gives a prediction plus error and Rsquared, adjRsquared

import numpy as np
import scipy.linalg
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from scipy.optimize import least_squares
from scipy.misc import comb
from astropy.table import Table, Column
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from scipy.stats import norm
import matplotlib.mlab as mlab
import lmfit
from numpy.linalg import inv
from scipy.stats import t as tdist
import argparse, sys
np.set_printoptions(suppress=True)
import os
plt.rcParams.update({'axes.labelsize': 25})
def str2bool(v):
    if v.lower() in ('yes', 'true', 't', 'y', '1', 'True'):
        return True
    elif v.lower() in ('no', 'false', 'f', 'n', '0', 'False'):
        return False
    else:
        raise argparse.ArgumentTypeError('Boolean value expected."
parser=argparse.ArgumentParser()
parser.add_argument('−m1', type=float, help='Mass of 1st Neutron Star')
parser.add_argument('−m2', type=float, help='Mass of 2nd Neutron Star')
parser.add_argument('−r1', type=float, help='Radius of of 1st Neutron Star')"
A.1 python codes

```python
parser.add_argument('--r2', type=float, help='Radius of of 2nd Neutron Star')
parser.add_argument('--ro', type=float, help='R_x where x is 1.2, 1.4, 1.6, 1.8')
parser.add_argument('--rn', type=str, default="1.6", help='This x in ro above')
parser.add_argument('--d', type=int, help='Degree of the polynomial to fit')
parser.add_argument('--p', type=str2bool, default=False, help='To plot or not to plot?!! This is the question')
parser.add_argument('--e', type=str2bool, default=False, help='Calculate everything only for equal mass')
parser.add_argument('--f', type=str, default="peak", help='Which frequency to predict')
parser.add_argument('--report', type=str2bool, default=False, help='Whether to report results or not')
parser.add_argument('--print', type=str, default='', help='What results to print for other programs to read, separate with "\n"')
parser.add_argument('--mr', type=str2bool, default=False, help='Use M/R for axis when --e option is enabled')
parser.add_argument('--fm', type=str2bool, default=False, help='Use freq/Mchirp for z axis')
args=parser.parse_args()
toprint = args.print
if "," in toprint:
toprint = toprint.split(",")
else:
toprint = ['%s' %toprint]
m1 = args.m1
m2 = args.m2
Mchirp0 = (m1 * m2)**(3/5) / (m1 + m2)**(1/5)
r0 = args.r0
r1 = args.r1
r2 = args.r2
Rchirp0 = (r1 * r2)**(3/5) / (r1 + r2)**(1/5)
def fitfunc(p,x,y,degree,powers):
    if powers == False:
        poly = PolynomialFeatures(degree)
data = np.stack((x, y), axis=-1)
tmp = poly.fit_transform(data)
return np.dot(tmp,np.array(p))
else:
    b = np.ones((len(x),1))
    for i in range(1,degree+1):
        b = np.hstack((b,x.reshape(len(x),1)**(i)))
        b = np.hstack((b,y.reshape(len(y),1)**(i)))
    return np.dot(b,np.array(p))
```

def residual(p, x, y, degree, z, powers):
    tmp = fitfunc(p, x, y, degree, powers) - z
    return tmp.flatten()

degree = args.d
only_powers = False
plots = args.p

data2 = pd.read_table("completeCore.csv", header=0, sep=" ","")
data2["fpeak", "Mchirp", "m1", "m2", "f_gw", "R1.2", "R1.4", "R1.6", "R1.8"] = data2["fpeak", "Mchirp", "m1", "m2", "f_gw", "R1.2", "R1.4", "R1.6", "R1.8"].apply(pd.to_numeric, errors="coerce")
data1 = pd.read_table("summary.csv", header=0, sep=" ",")
data1 = data1.rename(columns={"R1.2": "R1.2", "R1.4": "R1.4", "R1.6": "R1.6", "R1.8": "R1.8")
data1["fpeak", "fspiral", "f2-0", "Mchirp", "m1", "m2", "r1", "r2", "R1.2", "R1.4", "R1.6", "R1.8"] = data1["fpeak", "fspiral", "f2-0", "Mchirp", "m1", "m2", "r1", "r2", "R1.2", "R1.4", "R1.6", "R1.8"]].apply(pd.to_numeric, errors="coerce")
data1 = data1["eos", "fpeak", "Mchirp", "m1", "m2", "R1.2", "R1.4", "R1.6", "R1.8"]
data2 = data2["eos", "fpeak", "Mchirp", "m1", "m2", "R1.2", "R1.4", "R1.6", "R1.8"]
data = pd.concat([data1, data2])
equal_masses = args.e
if args.e == True:
    data = data[(data["m1"] == data["m2")])
rm = args.rm
data = data.dropna(subset=["R%s" % rm])
# data = data.drop(data[(data["eos"] == "ls220") & (data["m1"] == 1.5) & (data["m2"] == 1.5)].index)
wrongf = data[data.fpeak < 1800]

if len(wrongf) != 0:
    print("EoS with fpeak<1800 and excluded:")
    print(wrongf["eos", "fpeak", "Mchirp"])
    print("====================================")
    ofreq = wrongf["fpeak"].values
    for i in ofreq:
        data = data[data.fpeak != i]
    freq = args.f
if freq == "peak":
    data = data.dropna(subset=["fpeak")]
if args.fm == True:
    z = (data["fpeak"].values / data["Mchirp").values]*10**(-3)
z1 = (data1["fpeak").values / data1["Mchirp").values]*10**(-3)
z2 = (data2["fpeak").values / data2["Mchirp").values]*10**(-3)
else:
z = data['fpeak'].values * 10**(-3)
z1 = data1['fpeak'].values * 10**(-3)
z2 = data2['fpeak'].values * 10**(-3)

else:
    print("FrequencyError: Non valid choice for frequency."")
    print("Valid Options:")
    print("1. \"peak\"")
    exit()

x0 = Mchirp0
y0 = r0

x = data['Mchirp'].values
x1 = data1['Mchirp'].values
x2 = data2['Mchirp'].values
y = data['R %s' %rm].values
y1 = data1['R %s' %rm].values
y2 = data2['R %s' %rm].values
k = 2000
a = -k
b = k

if only_powers == False:
    dof = int(comb(2+degree,degree))
    init_guess = (b-a) * np.random.random(int(comb(2+degree,degree))) + a
else:
    dof = int(2*degree+1)
    init_guess = (b-a) * np.ones(int(2*degree+1)) + a

p = lmfit.Parameters()
for i in range(int(dof)):
    p.add('b %s ' %i, value = init_guess[i], min=a, max=b)

# create Minimizer
mini = lmfit.Minimizer(residual, params=p, fcn_args=(x, y, degree, z, only_powers), nan_policy='omit')

# first solve with Nelder-Mead
out1 = mini.minimize(method='Nelder')
# then solve with Levenberg-Marquardt using the
# Nelder-Mead solution as a starting point
out2 = mini.minimize(method='leastsq', params=out1.params)

fitParams = np.array(out2.params)
toprintparams = np.round(fitParams,3)
np.savetxt("./fitparams . txt ", toprintparams, delimiter=" ",
residuals1 = z - fitfunc2(fitParams,x,y,degree,only_powers)
residuals1 = out2.residual+x0

n = len(x)
sigma2 = np.dot(residuals1.T,residuals1) / (n-dof-1)
sigma = np.sqrt(sigma2)
observation = np.array([[x0,y0]]).reshape(1,-1)
if only_powers == True:
    tmp = np.ones((len(x),1))
    tmp2 = np.ones(1)
    for i in range(1,degree+1):
        tmp = np.hstack((tmp,x.reshape(len(x),1)**(i)))
        tmp = np.hstack((tmp,y.reshape(len(y),1)**(i)))
    tmp2 = np.hstack((tmp2,x0**(i)))
    tmp2 = np.hstack((tmp2,y0**(i)))
else:
    poly = PolynomialFeatures(degree)
    data = np.stack((x,y), axis=-1)
    tmp = poly.fit_transform(data)
    tmp2 = poly.fit_transform(observation)
    X = tmp * fitParams
    x0 = tmp2 * fitParams
    sey0 = sigma * np.sqrt(1 + np.dot(x0 , np.dot( inv(np.dot(X.T,X)) ,x0.T) ))
alpha = 0.05
    tvalue = tdist.interval(1-alpha/2,n-dof-1,loc=0,scale=1)
    confidence_interval_low = tvalue[0] * sey0
    confidence_interval_up = tvalue[1] * sey0
    SSres1 = np.sum(residuals1**2)
    SStot = np.sum((z-z.mean())**2)
    R21 = 1 - SSres1/SStot
    n = len(x)
    if only_powers == False:
        p = comb(2+degree,degree) -1
    else:
        p = 2*degree
        adjR21 = 1-(1-R21) * ((n-1)/(n-p-1))
    if args.report == True:
        print("M_1 =",m1," M_2 =",m2)
        print("R_1 =",r1," R_2 =",r2)
        print("Fitting for f% s" %freq)
        print("R_ %s =" %rm,r0)
    if only_powers == True:
        if args.fm == True:
            print(np.sum(x0)*Mchirp0 ,"+-", confidence_interval_up)
        else:
            print(np.sum(x0),"+-", confidence_interval_up)
    else:
        if args.fm == True:
            print(np.sum(x0)*Mchirp0,"+-", confidence_interval_up[0][0])
        else:
            print(np.sum(x0),"+-", confidence_interval_up[0][0])
        print("R^2 =",R21)
```python
print("adjR^2 =",adjR21)

if '' not in toprint:
    if only_powers == True:
        results={"prediction":np.sum(x0),"error": confidence_interval_up,"R^2": R21,"adjR^2": adjR21}
    else:
        results={"prediction":np.sum(x0)*Mchirp0,"error": confidence_interval_up[0][0],"R^2": np.round(R21,3),"adjR^2": adjR21,"fitparams": np.round(fitParams,3),"mean_residual":np.round(np.mean(abs(residuals1)),3),"max_residual":np.round(np.max(abs(residuals1)),3),"sigma_residual":np.round(np.std(abs(residuals1)),3)}

for i in toprint:
    print(i," : ",results[i])

if plots == True:
    zend1 = fitfunc(fitParams,x,y,degree,only_powers)
    min_chirp = np.min(x)
    max_chirp = np.max(x)
    min_r = np.min(y)
    max_r = np.max(y)
    zmin = z.min()
    zmax = z.max()

    fig = plt.figure(1,figsize=(11,7))
    plt.clf()
    ax = fig.gca(projection='3d')
    for i,j,k,h in zip(x,y,z,zend1):
        ax.plot([i,i],[j,j],[k,h],color = 'black ')
    a = fitParams
    x_grid = np.linspace(min_chirp,1.318,50)
    x_grid2 = np.linspace(1.318,max_chirp,50)
    y_grid = np.linspace(min_r,max_r,50)
    y_grid2 = np.linspace(13,max_r,50)
    x_grid, y_grid = np.meshgrid(x_grid, y_grid)
    x_grid2, y_grid2 = np.meshgrid(x_grid2,y_grid2)
    ax.plot_surface(x_grid, y_grid , z_grid,linewidth=0,alpha=0.3,color='indigo ')
    ax.plot_surface(x_grid2, y_grid2 , z_grid2,linewidth=0,alpha=0.3,color='indigo ')
    ax.zaxis.set_major_locator(LinearLocator(5))
    ax.zaxis.set_major_formatter(FormatStrFormatter(’%.01f’))
    # 1 -> Andi data
```
# 2 -> Core data

```python
ax.scatter(x1,y1,z1,s=70,marker="o" ,color="red",depthshade=False)
ax.scatter(x2,y2,z2,s=70,marker="o" ,color="green",depthshade=False)
ax.zaxis.set_rotate_label(False)
if args.mr == True:
    ax.set_xlabel("$M_{\text{chirp}}/R$")
else:
    ax.set_xlabel("$M_{\text{chirp}} [\text{M}_\odot]$")
ax.set_ylabel("$R_{\%s} [\text{km}]$" %rm)
if args.fm == True:
    ax.set_zlabel("$f_{\%s}/M_{\text{chirp}} [\text{kHz}/\text{M}_\odot]$" %args.f)
else:
    ax.set_zlabel("$f_{\%s}$" %args.f)
ax.zaxis.label.set_rotation(0)
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20
ax.zaxis.labelpad = 50
ax.zaxis._axinfo['juggled'] = (1,2,0)
ax.view_init(elev=37, azim=-11)
rm = int(float(rm) *10)
if args.e == True:
    e = 1
else:
    e = 0
plt.savefig("./figures/surfaceCombined_fRM_fpeak_R%s_e%s.pdf" %(rm,e),
            format="pdf",dpi=200)
os.system("pdfcrop figures/surfaceCombined_fRM_fpeak_R%s_e%s.pdf
deprecatedfigures/surfaceCombined_fRM_fpeak_R%s_e%s.pdf" %(rm,e,rm,e))
os.system("cp figures/surfaceCombined_fRM_fpeak_R%s_e%s.pdf /home/
stamatis/Research/Papers/2019/universal_relations/figures" %(rm,e))
```
APPENDIX B

B.1 REGRESSION INFORMATION FOR EMPIRICAL RELATIONS FOR FREQUENCIES
Table 1: Regression information for $f_{\text{peak}}$ frequency as $f_3$ in Equation 3 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{peak}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>18.203</td>
<td>-1.505</td>
<td>-1.944</td>
<td>-0.16</td>
<td>0.107</td>
<td>0.057</td>
<td>0.893</td>
<td>0.257</td>
<td>0.092</td>
<td>0.06</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>16.013</td>
<td>-1.092</td>
<td>-1.649</td>
<td>0.104</td>
<td>0.031</td>
<td>0.049</td>
<td>0.93</td>
<td>0.227</td>
<td>0.073</td>
<td>0.052</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>13.822</td>
<td>-0.576</td>
<td>-1.375</td>
<td>0.479</td>
<td>-0.073</td>
<td>0.044</td>
<td>0.956</td>
<td>0.196</td>
<td>0.056</td>
<td>0.044</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>12.168</td>
<td>-0.049</td>
<td>-1.205</td>
<td>0.954</td>
<td>-0.197</td>
<td>0.044</td>
<td>0.953</td>
<td>0.215</td>
<td>0.055</td>
<td>0.048</td>
</tr>
</tbody>
</table>

Table 2: Regression information for $f_{\text{peak}}$ frequency as $f_1$ in Equation 3 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{peak}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>16.91</td>
<td>-1.896</td>
<td>-1.708</td>
<td>0.2</td>
<td>0.068</td>
<td>0.05</td>
<td>0.88</td>
<td>0.374</td>
<td>0.096</td>
<td>0.07</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>14.819</td>
<td>-1.474</td>
<td>-1.43</td>
<td>0.414</td>
<td>-0.0</td>
<td>0.043</td>
<td>0.916</td>
<td>0.337</td>
<td>0.078</td>
<td>0.062</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>12.696</td>
<td>-0.935</td>
<td>-1.17</td>
<td>0.713</td>
<td>-0.092</td>
<td>0.037</td>
<td>0.943</td>
<td>0.298</td>
<td>0.062</td>
<td>0.053</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>10.942</td>
<td>-0.369</td>
<td>-0.987</td>
<td>1.095</td>
<td>-0.201</td>
<td>0.036</td>
<td>0.948</td>
<td>0.247</td>
<td>0.06</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 3: Regression information for $f_{2-0}$ frequency as $f_1$ in Equation 3 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{2-0}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>12.607</td>
<td>3.074</td>
<td>-1.791</td>
<td>-0.928</td>
<td>-0.012</td>
<td>0.058</td>
<td>0.718</td>
<td>0.366</td>
<td>0.114</td>
<td>0.077</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>10.859</td>
<td>3.586</td>
<td>-1.571</td>
<td>-0.706</td>
<td>-0.087</td>
<td>0.053</td>
<td>0.79</td>
<td>0.306</td>
<td>0.096</td>
<td>0.069</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>8.943</td>
<td>4.059</td>
<td>-1.332</td>
<td>-0.358</td>
<td>-0.182</td>
<td>0.048</td>
<td>0.849</td>
<td>0.229</td>
<td>0.08</td>
<td>0.061</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>7.797</td>
<td>4.773</td>
<td>-1.256</td>
<td>0.285</td>
<td>-0.357</td>
<td>0.055</td>
<td>0.86</td>
<td>0.269</td>
<td>0.075</td>
<td>0.061</td>
</tr>
</tbody>
</table>
Table 4: Regression information for $f_{2-0}$ frequency as $f_j$ in Equation 3 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{2-0}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>13.237</td>
<td>3.278</td>
<td>-1.894</td>
<td>-0.504</td>
<td>-0.107</td>
<td>0.066</td>
<td>0.785</td>
<td>0.383</td>
<td>0.105</td>
<td>0.081</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>11.549</td>
<td>3.76</td>
<td>-1.683</td>
<td>-0.26</td>
<td>-0.184</td>
<td>0.061</td>
<td>0.841</td>
<td>0.324</td>
<td>0.088</td>
<td>0.072</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>9.586</td>
<td>4.09</td>
<td>-1.427</td>
<td>0.048</td>
<td>-0.261</td>
<td>0.055</td>
<td>0.885</td>
<td>0.252</td>
<td>0.075</td>
<td>0.061</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>8.007</td>
<td>4.356</td>
<td>-1.241</td>
<td>0.558</td>
<td>-0.375</td>
<td>0.054</td>
<td>0.896</td>
<td>0.258</td>
<td>0.072</td>
<td>0.057</td>
</tr>
</tbody>
</table>

Table 5: Regression information for $f_{spiral}$ frequency as $f_j$ in Equation 3 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{spiral}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>10.565</td>
<td>1.013</td>
<td>-1.185</td>
<td>-0.184</td>
<td>-0.052</td>
<td>0.038</td>
<td>0.788</td>
<td>0.422</td>
<td>0.097</td>
<td>0.081</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>8.687</td>
<td>1.398</td>
<td>-0.934</td>
<td>0.096</td>
<td>-0.13</td>
<td>0.032</td>
<td>0.835</td>
<td>0.37</td>
<td>0.083</td>
<td>0.075</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>7.019</td>
<td>1.756</td>
<td>-0.721</td>
<td>0.468</td>
<td>-0.222</td>
<td>0.028</td>
<td>0.872</td>
<td>0.306</td>
<td>0.074</td>
<td>0.065</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>6.264</td>
<td>1.929</td>
<td>-0.645</td>
<td>0.881</td>
<td>-0.311</td>
<td>0.03</td>
<td>0.877</td>
<td>0.286</td>
<td>0.075</td>
<td>0.061</td>
</tr>
</tbody>
</table>

Table 6: Regression information for $f_{spiral}$ frequency as $f_j$ in Equation 3 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{spiral}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1.2}$</td>
<td>8.942</td>
<td>0.926</td>
<td>-0.926</td>
<td>-0.069</td>
<td>-0.061</td>
<td>0.028</td>
<td>0.773</td>
<td>0.438</td>
<td>0.109</td>
<td>0.079</td>
</tr>
<tr>
<td>$R_{1.4}$</td>
<td>7.356</td>
<td>1.321</td>
<td>-0.719</td>
<td>0.218</td>
<td>-0.141</td>
<td>0.024</td>
<td>0.814</td>
<td>0.383</td>
<td>0.1</td>
<td>0.07</td>
</tr>
<tr>
<td>$R_{1.6}$</td>
<td>6.107</td>
<td>1.666</td>
<td>-0.567</td>
<td>0.596</td>
<td>-0.234</td>
<td>0.022</td>
<td>0.845</td>
<td>0.316</td>
<td>0.092</td>
<td>0.063</td>
</tr>
<tr>
<td>$R_{1.8}$</td>
<td>5.846</td>
<td>1.75</td>
<td>-0.555</td>
<td>1.002</td>
<td>-0.316</td>
<td>0.026</td>
<td>0.846</td>
<td>0.27</td>
<td>0.089</td>
<td>0.066</td>
</tr>
</tbody>
</table>
B.2 regression information for empirical relations for radii

B.2 REGRESSION INFORMATION FOR EMPIRICAL RELATIONS FOR RADII
Table 7: Regression information for $f_{\text{peak}}$ frequency as $f_j$ in Equation 12 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{peak}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1,2}$</td>
<td>41.63</td>
<td>-15.017</td>
<td>-13.692</td>
<td>2.903</td>
<td>3.089</td>
<td>1.525</td>
<td>0.919</td>
<td>0.69</td>
<td>0.232</td>
<td>0.151</td>
</tr>
<tr>
<td>$R_{1,2}^{\text{sep}}$</td>
<td>52.201</td>
<td>-29.769</td>
<td>-15.398</td>
<td>8.918</td>
<td>3.333</td>
<td>1.832</td>
<td>0.945</td>
<td>0.52</td>
<td>0.191</td>
<td>0.131</td>
</tr>
<tr>
<td>$R_{1,4}$</td>
<td>40.891</td>
<td>-15.539</td>
<td>-12.71</td>
<td>3.235</td>
<td>2.967</td>
<td>1.333</td>
<td>0.949</td>
<td>0.578</td>
<td>0.181</td>
<td>0.129</td>
</tr>
<tr>
<td>$R_{1,4}^{\text{sep}}$</td>
<td>51.229</td>
<td>-30.463</td>
<td>-14.143</td>
<td>9.46</td>
<td>3.09</td>
<td>1.612</td>
<td>0.966</td>
<td>0.412</td>
<td>0.147</td>
<td>0.108</td>
</tr>
<tr>
<td>$R_{1,6}$</td>
<td>41.316</td>
<td>-16.654</td>
<td>-12.458</td>
<td>3.722</td>
<td>2.936</td>
<td>1.269</td>
<td>0.969</td>
<td>0.462</td>
<td>0.139</td>
<td>0.108</td>
</tr>
<tr>
<td>$R_{1,8}$</td>
<td>43.685</td>
<td>-18.889</td>
<td>-13.352</td>
<td>4.505</td>
<td>3.075</td>
<td>1.397</td>
<td>0.969</td>
<td>0.654</td>
<td>0.136</td>
<td>0.128</td>
</tr>
<tr>
<td>$R_{1,8}^{\text{sep}}$</td>
<td>33.802</td>
<td>-3.069</td>
<td>-15.522</td>
<td>-1.439</td>
<td>4.112</td>
<td>1.605</td>
<td>0.951</td>
<td>0.276</td>
<td>0.107</td>
<td>0.067</td>
</tr>
</tbody>
</table>

Table 8: Regression information for $f_{\text{peak}}$ frequency as $f_j$ in Equation 12 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{peak}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1,2}$</td>
<td>44.776</td>
<td>-19.351</td>
<td>-14.253</td>
<td>4.19</td>
<td>3.615</td>
<td>1.52</td>
<td>0.929</td>
<td>0.731</td>
<td>0.225</td>
<td>0.156</td>
</tr>
<tr>
<td>$R_{1,2}^{\text{sep}}$</td>
<td>56.906</td>
<td>-37.252</td>
<td>-15.701</td>
<td>11.756</td>
<td>3.638</td>
<td>1.83</td>
<td>0.951</td>
<td>0.526</td>
<td>0.19</td>
<td>0.134</td>
</tr>
<tr>
<td>$R_{1,4}$</td>
<td>43.865</td>
<td>-19.499</td>
<td>-13.296</td>
<td>4.392</td>
<td>3.463</td>
<td>1.338</td>
<td>0.952</td>
<td>0.621</td>
<td>0.182</td>
<td>0.135</td>
</tr>
<tr>
<td>$R_{1,4}^{\text{sep}}$</td>
<td>55.809</td>
<td>-37.642</td>
<td>-14.473</td>
<td>12.15</td>
<td>3.41</td>
<td>1.609</td>
<td>0.968</td>
<td>0.493</td>
<td>0.154</td>
<td>0.109</td>
</tr>
<tr>
<td>$R_{1,6}$</td>
<td>43.796</td>
<td>-19.684</td>
<td>-12.921</td>
<td>4.674</td>
<td>3.371</td>
<td>1.26</td>
<td>0.969</td>
<td>0.526</td>
<td>0.144</td>
<td>0.117</td>
</tr>
<tr>
<td>$R_{1,8}$</td>
<td>45.02</td>
<td>-21.087</td>
<td>-13.382</td>
<td>5.111</td>
<td>3.385</td>
<td>1.323</td>
<td>0.972</td>
<td>0.647</td>
<td>0.141</td>
<td>0.124</td>
</tr>
<tr>
<td>$R_{1,8}^{\text{sep}}$</td>
<td>28.796</td>
<td>-7.668</td>
<td>-6.631</td>
<td>0.516</td>
<td>3.478</td>
<td>-0.492</td>
<td>0.958</td>
<td>0.275</td>
<td>0.11</td>
<td>0.067</td>
</tr>
</tbody>
</table>
Table 9: Regression information for $f_{2-0}$ frequency as $f_1$ in Equation 12 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{2-0}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{12}$</td>
<td>13.015</td>
<td>7.143</td>
<td>-5.856</td>
<td>-1.883</td>
<td>0.059</td>
<td>1.049</td>
<td>0.873</td>
<td>0.679</td>
<td>0.272</td>
<td>0.183</td>
</tr>
<tr>
<td>$R_{12}^{sep}$</td>
<td>29.638</td>
<td>-19.343</td>
<td>-7.525</td>
<td>9.087</td>
<td>0.521</td>
<td>1.454</td>
<td>0.871</td>
<td>0.595</td>
<td>0.259</td>
<td>0.157</td>
</tr>
<tr>
<td>$R_{14}$</td>
<td>14.129</td>
<td>5.692</td>
<td>-6.233</td>
<td>-1.602</td>
<td>0.71</td>
<td>0.881</td>
<td>0.912</td>
<td>0.573</td>
<td>0.227</td>
<td>0.155</td>
</tr>
<tr>
<td>$R_{14}^{sep}$</td>
<td>28.249</td>
<td>-17.137</td>
<td>-7.399</td>
<td>7.988</td>
<td>0.9</td>
<td>1.22</td>
<td>0.912</td>
<td>0.506</td>
<td>0.205</td>
<td>0.147</td>
</tr>
<tr>
<td>$R_{16}$</td>
<td>15.271</td>
<td>4.123</td>
<td>-6.661</td>
<td>-1.188</td>
<td>1.23</td>
<td>0.783</td>
<td>0.942</td>
<td>0.465</td>
<td>0.186</td>
<td>0.13</td>
</tr>
<tr>
<td>$R_{18}$</td>
<td>16.527</td>
<td>2.287</td>
<td>-7.213</td>
<td>-0.562</td>
<td>1.618</td>
<td>0.778</td>
<td>0.951</td>
<td>0.626</td>
<td>0.165</td>
<td>0.146</td>
</tr>
<tr>
<td>$R_{18}^{sep}$</td>
<td>34.725</td>
<td>-15.096</td>
<td>-15.795</td>
<td>4.743</td>
<td>2.745</td>
<td>3.623</td>
<td>0.779</td>
<td>0.597</td>
<td>0.176</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Table 10: Regression information for $f_{2-0}$ frequency as $f_1$ in Equation 12 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{2-0}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{12}$</td>
<td>17.48</td>
<td>4.185</td>
<td>-9.647</td>
<td>-1.144</td>
<td>0.856</td>
<td>2.014</td>
<td>0.909</td>
<td>0.742</td>
<td>0.245</td>
<td>0.178</td>
</tr>
<tr>
<td>$R_{12}^{sep}$</td>
<td>31.374</td>
<td>-19.386</td>
<td>-9.852</td>
<td>9.845</td>
<td>-0.763</td>
<td>2.753</td>
<td>0.918</td>
<td>0.646</td>
<td>0.218</td>
<td>0.167</td>
</tr>
<tr>
<td>$R_{14}$</td>
<td>17.73</td>
<td>3.279</td>
<td>-9.268</td>
<td>-0.923</td>
<td>1.214</td>
<td>1.713</td>
<td>0.935</td>
<td>0.636</td>
<td>0.207</td>
<td>0.154</td>
</tr>
<tr>
<td>$R_{14}^{sep}$</td>
<td>30.105</td>
<td>-17.84</td>
<td>-9.364</td>
<td>8.874</td>
<td>-0.137</td>
<td>2.305</td>
<td>0.942</td>
<td>0.564</td>
<td>0.18</td>
<td>0.144</td>
</tr>
<tr>
<td>$R_{16}$</td>
<td>17.764</td>
<td>2.497</td>
<td>-8.797</td>
<td>-0.639</td>
<td>1.393</td>
<td>1.452</td>
<td>0.955</td>
<td>0.518</td>
<td>0.174</td>
<td>0.13</td>
</tr>
<tr>
<td>$R_{18}$</td>
<td>17.512</td>
<td>1.842</td>
<td>-8.211</td>
<td>-0.254</td>
<td>1.364</td>
<td>1.229</td>
<td>0.96</td>
<td>0.623</td>
<td>0.161</td>
<td>0.138</td>
</tr>
<tr>
<td>$R_{18}^{sep}$</td>
<td>0.747</td>
<td>24.015</td>
<td>-7.446</td>
<td>-6.192</td>
<td>-2.413</td>
<td>3.136</td>
<td>0.83</td>
<td>0.569</td>
<td>0.184</td>
<td>0.17</td>
</tr>
</tbody>
</table>
Table 11: Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 12 and only equal masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{spiral}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1,2}$</td>
<td>40.105</td>
<td>-17.154</td>
<td>-16.204</td>
<td>3.38</td>
<td>5.343</td>
<td>1.826</td>
<td>0.895</td>
<td>0.713</td>
<td>0.27</td>
<td>0.174</td>
</tr>
<tr>
<td>$R_{1,2}^{\text{sep}}$</td>
<td>41.603</td>
<td>-14.538</td>
<td>-19.426</td>
<td>0.955</td>
<td>6.832</td>
<td>2.257</td>
<td>0.905</td>
<td>0.803</td>
<td>0.245</td>
<td>0.181</td>
</tr>
<tr>
<td>$R_{1,4}$</td>
<td>39.694</td>
<td>-17.542</td>
<td>-15.372</td>
<td>3.61</td>
<td>5.245</td>
<td>1.601</td>
<td>0.923</td>
<td>0.637</td>
<td>0.229</td>
<td>0.158</td>
</tr>
<tr>
<td>$R_{1,4}^{\text{sep}}$</td>
<td>40.407</td>
<td>-14.521</td>
<td>-17.99</td>
<td>1.256</td>
<td>6.435</td>
<td>1.958</td>
<td>0.926</td>
<td>0.731</td>
<td>0.212</td>
<td>0.167</td>
</tr>
<tr>
<td>$R_{1,8}$</td>
<td>40.081</td>
<td>-18.359</td>
<td>-15.205</td>
<td>3.98</td>
<td>5.187</td>
<td>1.544</td>
<td>0.941</td>
<td>0.706</td>
<td>0.197</td>
<td>0.152</td>
</tr>
<tr>
<td>$R_{1,8}^{\text{sep}}$</td>
<td>41.976</td>
<td>-20.13</td>
<td>-16.179</td>
<td>4.636</td>
<td>5.27</td>
<td>1.75</td>
<td>0.939</td>
<td>0.914</td>
<td>0.205</td>
<td>0.168</td>
</tr>
<tr>
<td>$R_{1,8}^{\text{sep}}$</td>
<td>55.934</td>
<td>-37.162</td>
<td>-17.139</td>
<td>7.961</td>
<td>9.897</td>
<td>-0.382</td>
<td>0.951</td>
<td>0.212</td>
<td>0.117</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 12: Regression information for $f_{\text{spiral}}$ frequency as $f_j$ in Equation 12 and all masses in mass configuration.

<table>
<thead>
<tr>
<th>$f_{\text{spiral}}$</th>
<th>$b_0$</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$R^2$</th>
<th>max res</th>
<th>mean res</th>
<th>$\sigma$ res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{1,2}$</td>
<td>30.89</td>
<td>-10.914</td>
<td>-10.213</td>
<td>2.289</td>
<td>3.368</td>
<td>0.842</td>
<td>0.891</td>
<td>0.734</td>
<td>0.279</td>
<td>0.178</td>
</tr>
<tr>
<td>$R_{1,2}^{\text{sep}}$</td>
<td>38.805</td>
<td>-20.695</td>
<td>-12.612</td>
<td>5.561</td>
<td>4.366</td>
<td>1.212</td>
<td>0.898</td>
<td>0.737</td>
<td>0.275</td>
<td>0.18</td>
</tr>
<tr>
<td>$R_{1,4}$</td>
<td>30.64</td>
<td>-11.693</td>
<td>-9.289</td>
<td>2.64</td>
<td>3.318</td>
<td>0.575</td>
<td>0.913</td>
<td>0.672</td>
<td>0.252</td>
<td>0.157</td>
</tr>
<tr>
<td>$R_{1,4}^{\text{sep}}$</td>
<td>37.684</td>
<td>-20.594</td>
<td>-11.299</td>
<td>5.687</td>
<td>4.143</td>
<td>0.89</td>
<td>0.914</td>
<td>0.676</td>
<td>0.25</td>
<td>0.169</td>
</tr>
<tr>
<td>$R_{1,6}$</td>
<td>30.762</td>
<td>-12.647</td>
<td>-8.704</td>
<td>3.081</td>
<td>3.225</td>
<td>0.414</td>
<td>0.926</td>
<td>0.674</td>
<td>0.236</td>
<td>0.147</td>
</tr>
<tr>
<td>$R_{1,8}$</td>
<td>31.591</td>
<td>-14.039</td>
<td>-8.679</td>
<td>3.701</td>
<td>3.117</td>
<td>0.404</td>
<td>0.922</td>
<td>0.906</td>
<td>0.243</td>
<td>0.177</td>
</tr>
<tr>
<td>$R_{1,8}^{\text{sep}}$</td>
<td>54.468</td>
<td>-38.851</td>
<td>-13.993</td>
<td>9.305</td>
<td>8.453</td>
<td>-0.614</td>
<td>0.921</td>
<td>0.34</td>
<td>0.138</td>
<td>0.082</td>
</tr>
</tbody>
</table>
### B.2.1 Information table of Bauswein et al. data set

<table>
<thead>
<tr>
<th>EoS</th>
<th>$f_{\text{peak}}$ [kHz]</th>
<th>$f_{\text{gw}}$ [Hz]</th>
<th>$m_1$ [$M_\odot$]</th>
<th>$m_2$ [$M_\odot$]</th>
<th>$M_{\text{chirp}}$ [$M_\odot$]</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALF2</td>
<td>2.38</td>
<td>491.40</td>
<td>1.10</td>
<td>1.40</td>
<td>1.08</td>
<td>0.79</td>
</tr>
<tr>
<td>ALF2</td>
<td>2.75</td>
<td>454.79</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>ALF2</td>
<td>2.75</td>
<td>419.69</td>
<td>1.22</td>
<td>1.53</td>
<td>1.19</td>
<td>0.80</td>
</tr>
<tr>
<td>ALF2</td>
<td>2.75</td>
<td>422.82</td>
<td>1.38</td>
<td>1.38</td>
<td>1.20</td>
<td>1.00</td>
</tr>
<tr>
<td>ENG</td>
<td>3.00</td>
<td>454.57</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>H4</td>
<td>2.54</td>
<td>410.94</td>
<td>1.00</td>
<td>1.75</td>
<td>1.14</td>
<td>0.57</td>
</tr>
<tr>
<td>H4</td>
<td>2.56</td>
<td>410.86</td>
<td>1.10</td>
<td>1.65</td>
<td>1.17</td>
<td>0.67</td>
</tr>
<tr>
<td>H4</td>
<td>2.41</td>
<td>454.55</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>H4</td>
<td>2.47</td>
<td>409.87</td>
<td>1.22</td>
<td>1.53</td>
<td>1.19</td>
<td>0.80</td>
</tr>
<tr>
<td>H4</td>
<td>2.53</td>
<td>408.95</td>
<td>1.38</td>
<td>1.38</td>
<td>1.20</td>
<td>1.00</td>
</tr>
<tr>
<td>H4</td>
<td>2.86</td>
<td>424.99</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>MPA1</td>
<td>2.81</td>
<td>454.55</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>MS1b</td>
<td>1.88</td>
<td>389.50</td>
<td>1.00</td>
<td>1.50</td>
<td>1.06</td>
<td>0.67</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.08</td>
<td>490.42</td>
<td>1.10</td>
<td>1.40</td>
<td>1.08</td>
<td>0.79</td>
</tr>
<tr>
<td>MS1b</td>
<td>1.96</td>
<td>406.44</td>
<td>1.00</td>
<td>1.75</td>
<td>1.14</td>
<td>0.57</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.15</td>
<td>402.75</td>
<td>0.94</td>
<td>1.94</td>
<td>1.16</td>
<td>0.49</td>
</tr>
<tr>
<td>MS1b</td>
<td>1.97</td>
<td>406.58</td>
<td>1.10</td>
<td>1.65</td>
<td>1.17</td>
<td>0.67</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.09</td>
<td>418.48</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.01</td>
<td>406.59</td>
<td>1.22</td>
<td>1.53</td>
<td>1.19</td>
<td>0.80</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.09</td>
<td>407.98</td>
<td>1.38</td>
<td>1.38</td>
<td>1.20</td>
<td>1.00</td>
</tr>
<tr>
<td>MS1b</td>
<td>2.22</td>
<td>419.74</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>SLy</td>
<td>2.94</td>
<td>407.76</td>
<td>1.00</td>
<td>1.50</td>
<td>1.06</td>
<td>0.67</td>
</tr>
<tr>
<td>SLy</td>
<td>2.94</td>
<td>392.39</td>
<td>1.11</td>
<td>1.36</td>
<td>1.07</td>
<td>0.82</td>
</tr>
<tr>
<td>SLy</td>
<td>2.79</td>
<td>491.39</td>
<td>1.10</td>
<td>1.40</td>
<td>1.08</td>
<td>0.79</td>
</tr>
<tr>
<td>SLy</td>
<td>3.06</td>
<td>471.74</td>
<td>1.20</td>
<td>1.40</td>
<td>1.13</td>
<td>0.86</td>
</tr>
<tr>
<td>SLy</td>
<td>3.40</td>
<td>453.15</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>SLy</td>
<td>3.46</td>
<td>426.13</td>
<td>1.22</td>
<td>1.53</td>
<td>1.19</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Table 13: Frequency data set we computed using the CoRe catalogue.
Table 14: TOV information for the equations of state we used in this paper.

<table>
<thead>
<tr>
<th>EoS</th>
<th>$R_{1.2}$</th>
<th>$R_{1.4}$</th>
<th>$R_{1.6}$</th>
<th>$R_{1.8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR</td>
<td>11.37</td>
<td>11.34</td>
<td>11.27</td>
<td>11.14</td>
</tr>
<tr>
<td>BHBLP</td>
<td>13.08</td>
<td>13.14</td>
<td>13.12</td>
<td>12.95</td>
</tr>
<tr>
<td>DD2</td>
<td>13.08</td>
<td>13.15</td>
<td>13.18</td>
<td>13.16</td>
</tr>
<tr>
<td>LS220</td>
<td>12.65</td>
<td>12.56</td>
<td>12.36</td>
<td>12.00</td>
</tr>
<tr>
<td>LS375</td>
<td>13.43</td>
<td>13.57</td>
<td>13.67</td>
<td>13.72</td>
</tr>
<tr>
<td>NL3</td>
<td>14.64</td>
<td>14.69</td>
<td>14.73</td>
<td>14.75</td>
</tr>
<tr>
<td>SFHO</td>
<td>11.89</td>
<td>11.81</td>
<td>11.68</td>
<td>11.43</td>
</tr>
<tr>
<td>SFHX</td>
<td>11.87</td>
<td>11.92</td>
<td>11.90</td>
<td>11.79</td>
</tr>
<tr>
<td>TM1</td>
<td>14.44</td>
<td>14.38</td>
<td>14.26</td>
<td>14.06</td>
</tr>
<tr>
<td>ALF2</td>
<td>12.64</td>
<td>12.70</td>
<td>12.64</td>
<td>12.40</td>
</tr>
<tr>
<td>ENG</td>
<td>11.97</td>
<td>11.97</td>
<td>11.92</td>
<td>11.79</td>
</tr>
<tr>
<td>H4</td>
<td>14.04</td>
<td>13.95</td>
<td>13.74</td>
<td>13.32</td>
</tr>
<tr>
<td>MPA1</td>
<td>12.37</td>
<td>12.43</td>
<td>12.47</td>
<td>12.46</td>
</tr>
<tr>
<td>MS1b</td>
<td>14.43</td>
<td>14.53</td>
<td>14.60</td>
<td>14.65</td>
</tr>
<tr>
<td>SLy</td>
<td>11.80</td>
<td>11.71</td>
<td>11.55</td>
<td>11.27</td>
</tr>
</tbody>
</table>
### Table 15: Bauswein et al data set.

<table>
<thead>
<tr>
<th>EoS</th>
<th>$f_{\text{peak}}$ [kHz]</th>
<th>$f_{2-0}$ [kHz]</th>
<th>$f_{\text{spiral}}$ [kHz]</th>
<th>$m_1$ [$M_\odot$]</th>
<th>$m_2$ [$M_\odot$]</th>
<th>$M_{\text{chirp}}$ [$M_\odot$]</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>apr</td>
<td>3.46</td>
<td>2.33</td>
<td>2.68</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>bhblp</td>
<td>2.95</td>
<td>-</td>
<td>2.05</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>bhblp</td>
<td>2.65</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>bhblp</td>
<td>2.61</td>
<td>1.55</td>
<td>1.87</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>bhblp</td>
<td>2.43</td>
<td>-</td>
<td>1.81</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.67</td>
<td>1.54</td>
<td>1.94</td>
<td>1.35</td>
<td>1.50</td>
<td>1.24</td>
<td>0.90</td>
</tr>
<tr>
<td>dd2</td>
<td>3.08</td>
<td>2.18</td>
<td>-</td>
<td>1.65</td>
<td>1.65</td>
<td>1.44</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.62</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.80</td>
<td>1.27</td>
<td>0.67</td>
</tr>
<tr>
<td>dd2</td>
<td>2.66</td>
<td>1.56</td>
<td>2.06</td>
<td>1.40</td>
<td>1.40</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.77</td>
<td>1.72</td>
<td>2.13</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.80</td>
<td>1.72</td>
<td>-</td>
<td>1.35</td>
<td>1.80</td>
<td>1.35</td>
<td>0.75</td>
</tr>
<tr>
<td>dd2</td>
<td>2.50</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.60</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>dd2</td>
<td>2.96</td>
<td>1.97</td>
<td>2.32</td>
<td>1.60</td>
<td>1.60</td>
<td>1.39</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.55</td>
<td>1.41</td>
<td>1.86</td>
<td>1.30</td>
<td>1.40</td>
<td>1.17</td>
<td>0.93</td>
</tr>
<tr>
<td>dd2</td>
<td>2.54</td>
<td>1.29</td>
<td>1.95</td>
<td>1.30</td>
<td>1.30</td>
<td>1.13</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.61</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>dd2</td>
<td>2.94</td>
<td>1.98</td>
<td>-</td>
<td>1.50</td>
<td>1.80</td>
<td>1.43</td>
<td>0.83</td>
</tr>
<tr>
<td>dd2</td>
<td>2.49</td>
<td>1.26</td>
<td>1.76</td>
<td>1.20</td>
<td>1.35</td>
<td>1.11</td>
<td>0.89</td>
</tr>
<tr>
<td>dd2</td>
<td>2.41</td>
<td>1.20</td>
<td>1.80</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.60</td>
<td>1.47</td>
<td>1.98</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>dd2</td>
<td>2.90</td>
<td>1.95</td>
<td>-</td>
<td>1.50</td>
<td>1.70</td>
<td>1.39</td>
<td>0.88</td>
</tr>
<tr>
<td>ls220</td>
<td>3.06</td>
<td>2.07</td>
<td>2.54</td>
<td>1.40</td>
<td>1.40</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>ls220</td>
<td>2.59</td>
<td>1.42</td>
<td>1.77</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>ls220</td>
<td>3.43</td>
<td>2.32</td>
<td>2.59</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>ls220</td>
<td>2.85</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>ls220</td>
<td>2.87</td>
<td>1.72</td>
<td>2.18</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>ls375</td>
<td>2.39</td>
<td>1.33</td>
<td>1.76</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>ls375</td>
<td>3.07</td>
<td>2.11</td>
<td>2.42</td>
<td>1.80</td>
<td>1.80</td>
<td>1.57</td>
<td>1.00</td>
</tr>
<tr>
<td>ls375</td>
<td>2.41</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>ls375</td>
<td>2.23</td>
<td>-</td>
<td>1.65</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>ls375</td>
<td>2.56</td>
<td>1.50</td>
<td>1.83</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>ls375</td>
<td>2.69</td>
<td>1.66</td>
<td>2.05</td>
<td>1.60</td>
<td>1.60</td>
<td>1.39</td>
<td>1.00</td>
</tr>
<tr>
<td>EoS</td>
<td>$f_{\text{peak}}$ [kHz]</td>
<td>$f_{2-0}$ [kHz]</td>
<td>$f_{\text{spiral}}$ [kHz]</td>
<td>$m_1$ [$M_\odot$]</td>
<td>$m_2$ [$M_\odot$]</td>
<td>$M_{\text{chirp}}$ [$M_\odot$]</td>
<td>q</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------------</td>
<td>---</td>
</tr>
<tr>
<td>nl3</td>
<td>2.58</td>
<td>-</td>
<td>1.81</td>
<td>1.80</td>
<td>1.80</td>
<td>1.57</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.34</td>
<td>1.38</td>
<td>-</td>
<td>1.35</td>
<td>2.00</td>
<td>1.42</td>
<td>0.68</td>
</tr>
<tr>
<td>nl3</td>
<td>2.08</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>nl3</td>
<td>2.20</td>
<td>-</td>
<td>1.60</td>
<td>1.35</td>
<td>1.50</td>
<td>1.24</td>
<td>0.90</td>
</tr>
<tr>
<td>nl3</td>
<td>2.19</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.80</td>
<td>1.27</td>
<td>0.67</td>
</tr>
<tr>
<td>nl3</td>
<td>2.15</td>
<td>-</td>
<td>1.60</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.27</td>
<td>1.33</td>
<td>-</td>
<td>1.35</td>
<td>1.80</td>
<td>1.35</td>
<td>0.75</td>
</tr>
<tr>
<td>nl3</td>
<td>2.07</td>
<td>-</td>
<td>1.30</td>
<td>1.20</td>
<td>1.35</td>
<td>1.11</td>
<td>0.89</td>
</tr>
<tr>
<td>nl3</td>
<td>2.69</td>
<td>-</td>
<td>1.94</td>
<td>1.80</td>
<td>2.00</td>
<td>1.65</td>
<td>0.90</td>
</tr>
<tr>
<td>nl3</td>
<td>2.36</td>
<td>1.45</td>
<td>-</td>
<td>1.50</td>
<td>1.80</td>
<td>1.43</td>
<td>0.83</td>
</tr>
<tr>
<td>nl3</td>
<td>2.79</td>
<td>1.83</td>
<td>2.10</td>
<td>1.90</td>
<td>1.90</td>
<td>1.65</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.33</td>
<td>-</td>
<td>1.60</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.13</td>
<td>-</td>
<td>1.55</td>
<td>1.30</td>
<td>1.40</td>
<td>1.17</td>
<td>0.93</td>
</tr>
<tr>
<td>nl3</td>
<td>2.37</td>
<td>1.39</td>
<td>1.79</td>
<td>1.60</td>
<td>1.60</td>
<td>1.39</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.04</td>
<td>-</td>
<td>1.47</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>nl3</td>
<td>2.07</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.60</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>sfho</td>
<td>3.07</td>
<td>1.85</td>
<td>-</td>
<td>1.20</td>
<td>1.35</td>
<td>1.11</td>
<td>0.89</td>
</tr>
<tr>
<td>sfho</td>
<td>3.30</td>
<td>2.15</td>
<td>-</td>
<td>1.20</td>
<td>1.60</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>sfho</td>
<td>3.31</td>
<td>2.34</td>
<td>-</td>
<td>1.30</td>
<td>1.50</td>
<td>1.22</td>
<td>0.87</td>
</tr>
<tr>
<td>sfho</td>
<td>3.24</td>
<td>2.21</td>
<td>-</td>
<td>1.30</td>
<td>1.40</td>
<td>1.17</td>
<td>0.93</td>
</tr>
<tr>
<td>sfho</td>
<td>3.28</td>
<td>2.13</td>
<td>2.48</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>sfho</td>
<td>3.39</td>
<td>-</td>
<td>2.51</td>
<td>1.40</td>
<td>1.40</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>sfho</td>
<td>2.99</td>
<td>1.70</td>
<td>2.21</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>sfho</td>
<td>3.13</td>
<td>1.96</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.41</td>
<td>2.31</td>
<td>2.56</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.31</td>
<td>2.23</td>
<td>2.56</td>
<td>1.45</td>
<td>1.45</td>
<td>1.26</td>
<td>1.00</td>
</tr>
<tr>
<td>sfhx</td>
<td>2.91</td>
<td>1.81</td>
<td>2.06</td>
<td>1.20</td>
<td>1.35</td>
<td>1.11</td>
<td>0.89</td>
</tr>
<tr>
<td>sfhx</td>
<td>2.81</td>
<td>1.55</td>
<td>2.01</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>sfhx</td>
<td>2.85</td>
<td>1.86</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.06</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.60</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.20</td>
<td>2.11</td>
<td>-</td>
<td>1.35</td>
<td>1.50</td>
<td>1.24</td>
<td>0.90</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.16</td>
<td>2.02</td>
<td>2.43</td>
<td>1.40</td>
<td>1.40</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>sfhx</td>
<td>3.08</td>
<td>1.86</td>
<td>2.34</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>shen</td>
<td>2.19</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>sly4</td>
<td>3.33</td>
<td>2.18</td>
<td>2.55</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
</tbody>
</table>
### Table 17: Bauswein et al data set.

<table>
<thead>
<tr>
<th>EoS</th>
<th>$f_{\text{peak}}$</th>
<th>$f_{2-0}$</th>
<th>$f_{\text{spiral}}$</th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$M_{\text{chirp}}$</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>tm1</td>
<td>2.23</td>
<td>1.16</td>
<td>1.43</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>tm1</td>
<td>2.90</td>
<td>1.61</td>
<td>2.11</td>
<td>1.70</td>
<td>1.70</td>
<td>1.48</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.44</td>
<td>1.32</td>
<td>1.64</td>
<td>1.20</td>
<td>1.80</td>
<td>1.27</td>
<td>0.67</td>
</tr>
<tr>
<td>tm1</td>
<td>2.61</td>
<td>1.67</td>
<td>2.01</td>
<td>1.60</td>
<td>1.60</td>
<td>1.39</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.56</td>
<td>1.30</td>
<td>-</td>
<td>1.35</td>
<td>1.80</td>
<td>1.35</td>
<td>0.75</td>
</tr>
<tr>
<td>tm1</td>
<td>2.73</td>
<td>1.83</td>
<td>2.10</td>
<td>1.65</td>
<td>1.65</td>
<td>1.44</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.25</td>
<td>-</td>
<td>1.65</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.12</td>
<td>-</td>
<td>1.52</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>0.93</td>
</tr>
<tr>
<td>tm1</td>
<td>2.52</td>
<td>1.49</td>
<td>1.79</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.34</td>
<td>1.28</td>
<td>1.57</td>
<td>1.35</td>
<td>1.50</td>
<td>1.24</td>
<td>0.90</td>
</tr>
<tr>
<td>tm1</td>
<td>2.26</td>
<td>1.18</td>
<td>-</td>
<td>1.20</td>
<td>1.60</td>
<td>1.20</td>
<td>0.75</td>
</tr>
<tr>
<td>tm1</td>
<td>2.17</td>
<td>-</td>
<td>1.46</td>
<td>1.20</td>
<td>1.35</td>
<td>1.11</td>
<td>0.89</td>
</tr>
<tr>
<td>tm1</td>
<td>2.74</td>
<td>1.71</td>
<td>1.87</td>
<td>1.50</td>
<td>1.80</td>
<td>1.43</td>
<td>0.83</td>
</tr>
<tr>
<td>tm1</td>
<td>2.33</td>
<td>1.20</td>
<td>1.73</td>
<td>1.40</td>
<td>1.40</td>
<td>1.22</td>
<td>1.00</td>
</tr>
<tr>
<td>tm1</td>
<td>2.20</td>
<td>-</td>
<td>1.64</td>
<td>1.30</td>
<td>1.30</td>
<td>1.13</td>
<td>1.00</td>
</tr>
<tr>
<td>tma</td>
<td>2.15</td>
<td>1.15</td>
<td>1.64</td>
<td>1.20</td>
<td>1.20</td>
<td>1.04</td>
<td>1.00</td>
</tr>
<tr>
<td>tma</td>
<td>2.97</td>
<td>1.63</td>
<td>2.18</td>
<td>1.60</td>
<td>1.60</td>
<td>1.39</td>
<td>1.00</td>
</tr>
<tr>
<td>tma</td>
<td>2.75</td>
<td>1.84</td>
<td>2.13</td>
<td>1.55</td>
<td>1.55</td>
<td>1.35</td>
<td>1.00</td>
</tr>
<tr>
<td>tma</td>
<td>2.33</td>
<td>1.26</td>
<td>-</td>
<td>1.20</td>
<td>1.50</td>
<td>1.17</td>
<td>0.80</td>
</tr>
<tr>
<td>tma</td>
<td>2.38</td>
<td>1.21</td>
<td>1.76</td>
<td>1.35</td>
<td>1.35</td>
<td>1.18</td>
<td>1.00</td>
</tr>
<tr>
<td>tma</td>
<td>2.73</td>
<td>1.82</td>
<td>2.21</td>
<td>1.50</td>
<td>1.50</td>
<td>1.31</td>
<td>1.00</td>
</tr>
</tbody>
</table>


DECLARATION

Put your declaration here.

*Thessaloniki, June 27, 2019*

_______________________
Stamatis Vretinaris