UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Astrophysics and Cosmology

Final Dissertation

A machine learning approach to parameter inference in gravitational-wave signal analysis

Thesis supervisor:

Prof. Giacomo Ciani

Candidate:

Matteo Scialpi

Thesis co-supervisors:

Prof. Edoardo Milotti

Prof. Agata Trovato

Academic Year 2022/2023

Abstract

Gravitational Wave (GW) physics is now in its golden age thanks to modern interferometers. The fourth observing run is now ongoing with two of the four second-generation detectors, collecting GW signals coming from Compact Binary Coalescences (CBCs). These systems are formed by black holes and/or neutron stars which lose energy and angular momentum in favour of GW emission, spiraling toward each other until they merge. The characteristic waveform has a chirping behaviour, with a frequency increasing with time. These GW signals are gold mines of physical information on the emitting system.

The data analysis of these signals has two main aspects: detection and parameter estimation. For what concerns detection, two approaches are used right now: matched filtering, which compares numerical waveform with raw interferometers' output to highlight the signal, and the study of bursts, which highlights the coherence of arbitrary signals in different detectors. Both these techniques need to be fast enough to allow for electromagnetic follow-up with a relatively short delay. The offline parameter inference process is based on Bayesian techniques and is rather lengthy (individual processing Markov Chain Monte Carlo runs can take a month or more).

My thesis has the goal of introducing a fast parameter estimation for unmodeled (burst) methods which produce only phenomenological, de-noised waveforms with, at best, a rough estimate of only a few parameters. The implementation of an approach for fast parameter inference in this unmodeled analysis, taking as input the reconstructed waveform, could be extremely useful for multimessenger observations.

In this context, Keith et al. (2021a) proposed to use Physics Informed Neural Networks (PINNs; Raissi et al. (2019)) in GW data analysis. These PINNs are a machine learning approach which includes physical prior information in the algorithm itself. Taking a clean chirping waveform as input, the algorithm of Keith et al. (2021a) demonstrated a successful application of this concept and was able to reconstruct the compact object's orbits before coalescence with great detail, starting only from a parameterized Post-Newtonian model. The PINN environment could become a key tool to infer parameters from GW signals with a simple physical ansatz.

As part of my thesis work, I reviewed in detail GW physics and the PINN environment and I updated the algorithm described in Keith et al. (2021a). Their ground-breaking work introduces PINNs for the first time in the analysis of GW signals, however it does so without considering some important details. In particular, I noted that the algorithm of Keith et al. (2021a) spans a very constrained parameter space. In this thesis I introduce some of the missing details and I recode the algorithm from scratch. My implementation includes the learning of the phenomenological differential equation that describes the frequency evolution over time of the chirping GW, within a different, but more physical, parameter space. As a test, starting from a waveform as training data, and from the Newtonian approximation of the GW chirp, I infer the chirp mass, the GW phase and the frequency exponent in the differential equation. The resulting algorithm is robust and uses realistic physical conditions. This is a necessary first step to realize parameter inference with PINNs on real gravitational wave data.

Contents

Introduction

On September 14th, 2015 the first Gravitational Wave (GW) signal was detected by ground-based interferometer detectors (Abbott et al., 2016b). This detection is an important milestone in modern experimental and theoretical physics. GWs were predicted by the theory of General Relativity (GR) already in 1916 (Einstein, 1916) as waves propagating in space-time. After some initial skepticism the search for GWs continued for decades and spanned one generation of resonant bar gravitational antennas and two generations of interferometric detectors. From a physical point of view, their existence is a great achievement of GR. After the first experimental observation, they have become a new powerful tool to understand the Universe. Indeed, GWs are a true goldmine of information on the emitting systems.

Gravitational-wave signals are produced by changes in mass-energy distribution at the quadrupole (or higher multipole) level. The most likely events that can produce GWs are the coalescences of compact objects like Black Holes (BHs) or Neutron Stars (NSs) which are extremely dense objects with a mass similar or larger then the solar mass (1 $M_\odot \approx 1.989 \cdot 10^{30}$ kg), enclosed within a radius of the order of tens of km. These fascinating objects are studied by scientists because they are perfect environments to study physics at its extremes and as probes of stellar or cosmic evolution. In particular, they can also be found in a binary system, orbiting each other. These orbital configurations are strong GWs emitters and the outgoing flux of GWs causes a loss of energy from the system, leading to an inspiral phase which continues until a plunge phase when the gravitational potential becomes too strong, leading to the final merger of the objects into a single one. This system evolution produces a characteristic chirping gravitational-wave signal, as shown in Fig. 1. This is the only type of gravitational–wave signal detected to date.

The detection of gravitational waves is no easy task. The main issue is that the strain caused by incoming GWs is usually of the order of 10^{-21} or smaller. This means signals are buried in noise and can only be extracted using the coherence or coincidence among detectors (*burst* or *unmodeled* pipelines) and/or prior physical information, like the signal shape from a specific source type (*matched–filter* pipelines). Over the years, several dedicated pipelines have been developed to detect GW signals in the raw interferometer output.

Burst pipelines aim to detect signals using as little physical information as possible, thanks to the coherence or coincidence properties in different detectors, with the goal of detecting generic signals – including Compact Binary Coalescences (CBC). However, unlike the matched–filter pipelines the unmodeled searches cannot return model parameters, and only estimate frequency bandwidth, duration and sky location. With the addition of weak assumptions it is possible to recover additional information, like the chirp mass in CBCs. However most model parameters like the individual component masses, remain hidden in the phenomenological burst waveforms.

In this thesis, I test a novel approach to parameter estimation of gravitational–wave signals that could be applied to gravitational waveforms obtained with burst pipelines,

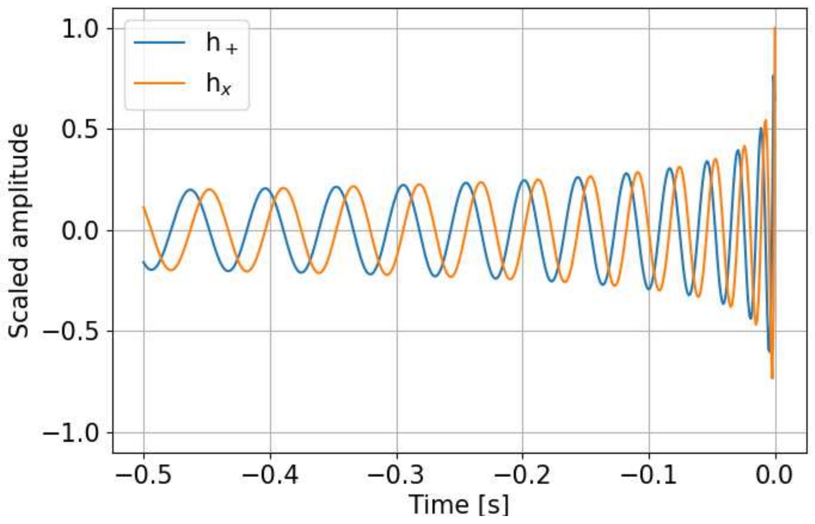


Figure 1: *Newtonian model of the chirping waveform emitted by a system of two merging equal masses $m = 30\ M_\odot$. Both polarizations are shown.*

the Physics Informed Neural Networks (PINNs; Raissi et al. (2019)), first introduced in this field by Keith et al. (2021a). PINNs are a Machine Learning (ML) algorithm capable of estimating parameters or discovering additional terms in ordinary or partial differential equations, starting from experimental data. They are based on the Universal Differential Equation (UDE) formalism (Rackauckas et al., 2021), where the differential equation is partially or fully approximated thanks to NNs. Taking a time–series as training dataset, Keith et al. (2021a) were able to learn equations of motions of a simple CBC. The authors demonstrated that, thanks to PINNs, one can infer source parameters from gravitational-wave data.

While the approach is surely promising it turns out to be inadequate when applied to real data. While testing the code provided by the authors in a public repository (Keith et al., 2021b), I noticed the following limitations:

- there is no provision for different polarization states and the antenna patterns of the detectors;
- the coding environment, using SciML in Julia, is too constraining;
- many constants and assumptions are hard-coded in the program making any modification extremely awkward.

For these reasons, I recoded the algorithm in Python using the PyTorch environment, carefully keeping track of the important parameters in the code, so that the final program would be flexible and adaptable to important changes. In particular, I generated waveforms in the simple Newtonian approximation with no spin and I worked with the phenomenological dependence of the gravitational–wave signal on time, inferring the chirp mass $\mathscr{M}$, the GW phase ϕ and the frequency exponent α (which by default is $11/3$ in the Newtonian approximation). This is a necessary first step to realize parameter inference with PINNs on real GW data.

The thesis is organized as follows. In Chapter 1 I review the GW physics, explaining both the theoretical background and the experimental efforts to detect signals and infer parameters. In Chapter 2 I revise ML techniques, focusing on PINNs and on the SciML environment. In Chapter 3 I report in detail the Keith et al. (2021a) algorithm, in order to introduce my work, which is presented in Chapter 4. Finally, in Chapter 5 I discuss the results my work and I present possible further improvements and applications of it.

Chapter 1

Gravitational waves

The field of Gravitational Waves (GWs) is rapidly emerging. Predicted by Einstein's General Relativity (GR; Einstein (1916)), in September 14th, 2015, the first GW signal from a Compact Binary Coalescence (CBC) was detected by the second generation of GW detectors: laser interferometers (Abbott et al., 2016b). This detection is a milestone of modern physics and more and more physicists are spending their interest in this field.

The first chapter will be a review of GW physics, including basic theoretical concepts about GW production and emission, and a description of modern detectors and data analysis techniques. It will be based mostly on Maggiore (2007) and Foster and Nightingale (2010).

1.1 Einstein equations and their full resolution

Gravity has challenged physicists around the world since its first formal description by Newton (1687). He opened up a whole new field of science by having a brilliant insight into a fundamental force: every massive body exerts this force on another for the simple reason that it has a mass. In his reasoning, the gravitational attraction between two masses m_1 and m_2 is a central conservative force:

$$\mathbf{F} = -\,G\,\frac{m_1 m_2}{r^2}\,\mathbf{r}\,, \tag{1.1}$$

where $G = 6.67 \cdot 10^{-11}$ N m^2 kg^{-2} is the universal gravitational constant and $\mathbf{r}$ is the separation vector between the two objects, with modulus r. Thanks to this, and to Newton's differential calculus, Kepler's laws of dynamics (Kepler, 1609) were developed in their final rigorous form. In particular, given the fact that two objects orbit each other in elliptical orbits, the third law will be useful for the foreseeable future. If a is the semimajor axis, m is the mass of the reference object and ω_o is the angular velocity of the orbit, we can state that

$$a^3 = mG\omega_o^2\,. \tag{1.2}$$

Gravity was thought to propagate instantaneously, with the basic concept of an absolute characterization of the time for different observers. We have to wait for Einstein's General Relativity (GR; Einstein (1916)) for the theory of gravity currently adopted.

Einstein's theory of General Relativity (Einstein, 1916) is based on the equivalence principle, which states that a reference frame in free fall is equivalent to a local inertial reference frame. Indeed, the fundamental key to the behavior of gravity is given by Einstein's field equations:

$$G_{\mu\nu} := R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu}\,. \tag{1.3}$$

Here $G_{\mu\nu}$ is the Einstein tensor, $R_{\mu\nu}$ the Ricci tensor, R the Ricci scalar, $g_{\mu\nu}$ the metric tensor and $T_{\mu\nu}$ the stress-energy tensor. The values of the constants are determined by mathematical consistency[1] and weak field limit. This equation specifies a key physical effect: every possible form of energy causes a warp in space-time. Indeed, the terms in $G_{\mu\nu}$ are related to geometry and $T_{\mu\nu}$ contains the energy components. Since the mass itself is a kind of energy, any object causes a curvature by simply being massive. Gravity is no longer a simple force, but a consequence of this space-time deformation.

Einstein's equations (1.3) are non-linear differential equations in $g_{\mu\nu}$. Indeed, $R_{\mu\nu}$ and R are different contractions of the same Riemann tensor $R^{\sigma}_{\mu\rho\nu}$, given by

$$R^{\sigma}_{\mu\rho\nu} = \partial_{\rho}\Gamma^{\sigma}_{\mu\nu} - \partial_{\nu}\Gamma^{\sigma}_{\rho\mu} + \Gamma^{\sigma}_{\rho\lambda}\Gamma^{\lambda}_{\mu\nu} - \Gamma^{\sigma}_{\nu\lambda}\Gamma^{\lambda}_{\rho\mu}\,, \tag{1.4}$$

where $\Gamma^{\sigma}_{\mu\nu}$ are the Christoffel symbols, dependent on $g_{\mu\nu}$ first parial derivatives. $R_{\mu\nu}$ and R are defined as

$$R_{\mu\nu} := R^{\sigma}_{\mu\sigma\nu}\,, \tag{1.5}$$

$$R := g^{\mu\nu}R_{\mu\nu}\,. \tag{1.6}$$

The terms of the Einstein tensor $G_{\mu\nu}$ depend on the second derivatives of $g_{\mu\nu}$. Moreover, $G_{\mu\nu}$ and $T_{\mu\nu}$ are symmetric tensors, so these equations have 10 degrees of freedom (dofs). We are dealing with nonlinear partial differential equations with 10 dofs.

We can tell from this last statement alone that the solving process is quite difficult. Solving Einstein's equations in their general form is an open task. Some approximate solutions are present in environments with specific symmetries, but current research focuses on two main approaches: the Post-Newtonian (PN) approximation and Numerical Relativity (NR).

1.1.1 Post-Newtonian approximation

The PN approximation is the simplest environment we can operate with. It was developed in the early years just after Einstein's GR thanks to Lorentz and Droste (1937) and it is based on Newtonian relativity, as suggested by the name. The idea is to approximate GR by adding relativistic corrections to the Newtonian laws of motion in the assumption of slowly moving and weakly stressed objects. This is possible thanks to the introduction of the following parameter:

$$\epsilon := \max\left\{\left|\frac{T^{0i}}{T^{00}}\right|, \left|\frac{T^{ij}}{T^{00}}\right|^{1/2}, \left|\frac{U}{c^2}\right|^{1/2}\right\}, \tag{1.7}$$

depending on different components of the stress-energy tensor $T^{\mu\nu}$ and the gravitational potential energy U of the source. It can be seen that ϵ is of the order

$$\epsilon \sim \frac{v}{c}\,, \tag{1.8}$$

where v is the velocity of the considered objects. For $v \ll c$, we get $\epsilon \ll 1$. The parameter ϵ is the index that will lead to different correction orders. More precisely, the PN approximation is an expansion in terms of v^2/c^2 and therefore the n-th order of PN corrections will be denoted as $\frac{n}{2}$PN. We will see that the GWs appear in their simple form with a 2PN quadrupole correction.

This approximation fails when objects and their gravitational interactions become extremely relativistic, since it overlooks a significant amount of physics that must be considered in highly relativistic environments.

[1]Bianchi's identities $\nabla_{\mu}G^{\mu\nu} = 0$ must hold, where ∇_{μ} is the covariant derivative with respect to x^{μ}.

1.1.2 Numerical relativity

NR aims to solve Einstein's equations (1.3) in their complete form. Once the solution is found, it is then applied to real physical situations. The idea is to solve those differential equations using purely numerical techniques. This is an ingenious idea of recent physics, especially thanks to widely available remote supercomputers. Its results could be applied to any possible field of astrophysics due to their generality and they are the most accurate ones available in the literature.

However, the most problematic issue with this approach is the computational cost. A single simplified simulation could take more than a month to run completely and very little time evolution could be simulated. This has narrowed the scope of applicability of this technique. For example, when dealing with coalescences of compact binaries, NR can only simulate the last few orbits.

1.2 Gravitational waves

1.2.1 Linearized general relativity

Gravitational Waves (GW) are linear solutions of Einstein equations. This can be easily demonstrated if we estimate the behaviour of the metric tensor away from the sources. In this case it is possible to use the perturbative approach that is based on the weak field assumption, and Einstein's equations can be linearized. In other words we write the metric tensor $g_{\mu\nu}$ as

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}\,, \qquad |h_{\mu\nu}| \ll 1\,. \tag{1.9}$$

The physical situation speaks for itself: we are in an almost flat space-time. A fundamental note should be made on the degrees of freedom (dof). By choosing this reference system we are already constraining the coordinates. Thus, we are still only allowed to do coordinate transformations not violating the equation (1.9). These are global Lorentz transformations of the type

$$x^\mu \longrightarrow {x'}^\mu = x^\mu + \xi^\mu(x)\,, \tag{1.10}$$

where $\partial_\mu \xi^\mu(x)$ must be as small as $h_{\mu\nu}$. Taking the linear order of each term in (1.3), we have

$$\frac{1}{2}\big[\partial_\mu\partial_\nu h + \Box h_{\mu\nu} - \partial_\rho\partial_\nu h^\rho_\mu - \partial_\mu\partial_\rho h^\rho_\nu - \eta_{\mu\nu}(\Box h - \partial_\rho\partial_\sigma h^{\sigma\rho})\big] = -\frac{8\pi G}{c^4} T_{\mu\nu}\,, \tag{1.11}$$

where $h = h^\rho_\rho$ is the trace of $h_{\mu\nu}$ and $\Box = \partial^\mu\partial_\mu$ is the D'Alembertian operator.

Moving to the Lorenz gauge

$$\Box \xi^\mu = \partial_\rho \overline{h}^{\mu\rho} \tag{1.12}$$

in order to have

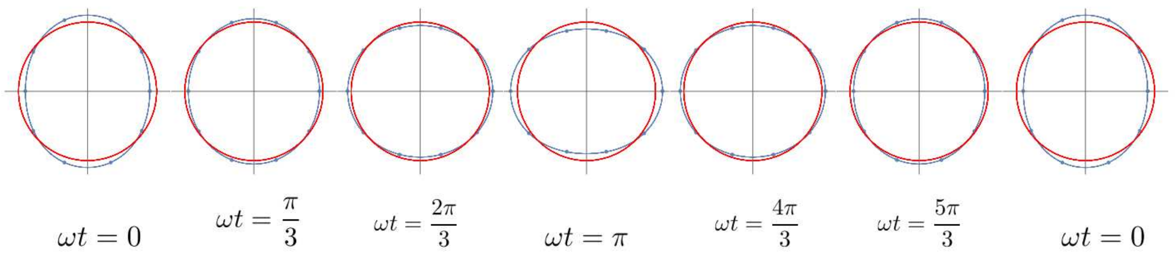
$$\partial_\rho \overline{h}'^{\mu\rho} = 0\,, \tag{1.13}$$

we get the linearized version of Einstein's field equations:

$$\Box \overline{h}_{\mu\nu} = -\frac{16\pi G}{c^4} T_{\mu\nu}\,, \tag{1.14}$$

where $\overline{h}_{\mu\nu}$ is the trace revese of $h_{\mu\nu}$. Expanding the D'Alembertian operator, we get

$$\left(\frac{1}{c^2}\frac{\partial^2}{(\partial x^0)^2} - \nabla^2\right)\overline{h}_{\mu\nu} = -\frac{16\pi G}{c^4} T_{\mu\nu}\,. \tag{1.15}$$



$\omega t = 0$
$\omega t = \frac{\pi}{3}$
$\omega t = \frac{2\pi}{3}$
$\omega t = \pi$
$\omega t = \frac{4\pi}{3}$
$\omega t = \frac{5\pi}{3}$
$\omega t = 0$

Figure 1.1: *Configurations for a ring of particles with a GW propagating orthogonally to the plane with only h_+ polarization as a function of $\omega_o t$. The underlined circle is the rest position of the particles, without the GW passage* (Milotti, 2022b).

This is a wave equation. In linearized general relativity, with Lorenz gauge, the little curvature perturbation $h_{\mu\nu}$ has the characteristic behaviour of a wave propagating at the speed of light c.

Let us focus now on a local vacuum ($T_{\mu\nu} = 0$). We can thus look for a plane wave solution of the type

$$\overline{h}_{\mu\nu} = A_{\mu\nu} e^{ik_\lambda x^\lambda} , \tag{1.16}$$

where $A_{\mu\nu}$ is the amplitude tensor and k_λ is the 4-wave vector $k = (\omega_o/c, \mathbf{k})$. Thanks to the Lorenz gauge (1.13), we find the wave to be transverse, which means that the amplitude is non-zero only orthogonally to the direction of propagation. For what concerns the dofs, we started with the 10 dofs of Einstein's equations and set 4 constraints to switch to the Lorentz gauge. We are now free to constrain another four dofs by moving to the Transverse-Traceless (TT) gauge:

$$\overline{h}^{0i}_{TT} = 0 , \tag{1.17}$$

$$\overline{h}_{TT} = 0 . \tag{1.18}$$

Now we have only two degrees of freedom left. These become explicit if we consider a plane wave propagating along x^3:

$$h^{TT}_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}_{\mu\nu} e^{i\omega_o(t-z/c)} . \tag{1.19}$$

They corresponds to the two polarizations h_+ and $h_\times$.

Consider a ring of particles in the xy plane separated by $\xi(t) = (x_0+\delta x(t), y_0+\delta y(t), 0)$ and a GW propagating along x^3, as (1.19). One can demonstrate that a GW passing by acts on their proper distances as

$$\begin{aligned} h_+: &\quad \delta x(t) = \frac{1}{2} h_+ x_0 \cos(\omega_o t) , &\quad \delta y(t) = -\frac{1}{2} h_+ y_0 \cos(\omega_o t) , \\ h_\times: &\quad \delta x(t) = \frac{1}{2} h_\times x_0 \cos(\omega_o t) , &\quad \delta y(t) = \frac{1}{2} h_\times y_0 \cos(\omega_o t) . \end{aligned} \tag{1.20}$$

For what concerns h_+, we can see the physical effect on the ring of particles in Fig. 1.1.

1.2.2 Generation

So far we have talked about GWs as already present in space-time. What can we say on their production? To answer this question, we need to solve the linear form of

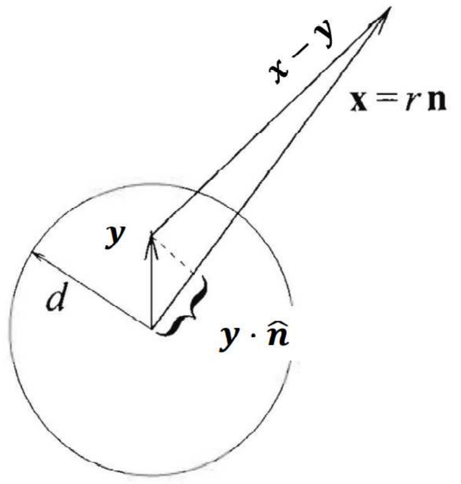


Figure 1.2: *Geometric sketch for the GW emission* (Maggiore, 2007).

Einstein's equations (1.14) in the presence of some form of energy. Setting $T_{\mu\nu} \neq 0$ is necessary to have a perturbation of the space-time curvature[2]. This means that we want to solve

$$\Box \bar{h}_{\mu\nu} = -\frac{16\pi G}{c^4} T_{\mu\nu}\,. \tag{1.21}$$

In this case, we are in the linear order of expansion, considering (1.9) for $g_{\mu\nu}$ and assuming a weak field regime. We can also assume a non-relativistic velocity $v \ll c$ for the source. For a self-gravitating system, the virial theorem states that the kinetic energy of the system must be equal to half of its gravitational potential energy

$$E_{kin} = -\frac{1}{2}U \qquad \Longrightarrow \qquad \frac{1}{2}\mu v^2 = \frac{1}{2}\frac{G\mu M}{R}\,, \tag{1.22}$$

where U is the gravitational potential energy of the source, $\mu = m_1 m_2/(m_1 + m_2)$ is the reduced mass, $M = m_1 + m_2$ is the sum of the two masses and R is the object radius. Isolating v^2 and dividing everything by c^2, we get

$$\frac{v^2}{c^2} = \frac{GM}{c^2 R} = \frac{R_S}{2R}\,, \tag{1.23}$$

where R_S is the Schwarzschild radius, defined as

$$R_S = \frac{2GM}{c^2}\,. \tag{1.24}$$

It is the event horizon radius for a non-rotating BH. We see that v^2/c^2 is an indicator of the relativistic level of the self-gravitating system. The closer the ratio is to 1/2, the more the object is similar to a BH. The closer the ratio is to 0, the more the object is non-relativistic. We can choose to deal with an infinitesimal v^2/c^2 and use it as an expansion parameter, exactly as the PN approximation does. An accurate calculation for this resolution is beyond the scope of this thesis and can be found in Maggiore (2007). In this section I will present only the fundamental results.

Consider the sketch in Fig. 1.2, where we can see the extended source of dimension d, at a distance $\mathbf{x} = r\hat{\mathbf{n}}$ from the observer. Every point of the source can be mapped thanks to the $\mathbf{y}$ coordinate. The TT gauge solution is

$$h_{ij}^{TT}(t, \mathbf{x}) = \frac{4G}{c^4 r}\Lambda_{ij}^{kl} \int d^3y\, T_{kl}\left(t - \frac{r}{c} + \frac{\mathbf{y}\cdot\hat{\mathbf{n}}}{c}, \mathbf{y}\right)\,, \tag{1.25}$$

[2]In the far field limit that we considered before, $T_{\mu\nu} = 0$ locally, but at the source $T_{\mu\nu}$ must be non-zero.

where Λ_{ij}^{kl} is the projector from a general tensor to its TT gauge form, defined as

$$A_{ij}^{TT} = \Lambda_{ij}^{kl} A_{kl} := \left(P_k^i P_l^j - \frac{1}{2} P^{ij} P_{kl} \right) A^{kl} , \tag{1.26}$$

with

$$P_{ij}(\hat{\mathbf{n}}) = \delta_{ij} - n_i n_j . \tag{1.27}$$

Note that T_{kl} in (1.25) is calculated at the time of emission. t is the time of the detection, while the $(r + \mathbf{y} \cdot \hat{\mathbf{n}})/c$ correction is the so-called retarded time. Following Fig. 1.2, the idea is that the GW detected at instant t is the superimposition of several waves produced by δ-like sources at different times. These δ-like sources are precisely the different points of the extended source, which are $\mathbf{y} \cdot \hat{\mathbf{n}}$ more or less distant from the observer. The time lag due to this difference in path can be seen as the following parameter

$$\xi := \frac{\mathbf{y} \cdot \hat{\mathbf{n}}}{c} \sim \frac{d}{c} \ll \frac{d}{v} . \tag{1.28}$$

Thanks to this term, we can expand T_{kl}. Taking only the leading order, we can get

$$h_{ij}^{TT}(t) \approx \frac{2G}{c^4 r} \Lambda_{ij}^{kl} \ddot{M}_{kl} = \frac{2G}{c^4 r} \Lambda_{ij}^{kl} \ddot{Q}_{kl} \left(t - \frac{r}{c} \right) , \tag{1.29}$$

where M^{ij} is the second moment for $T^{00}(t, x)$ and Q^{kl} is the quadrupole moment of mass (ρ is the mass density, defined as

$$M^{ij} := \frac{1}{c^2} \int d^3x \, T^{00}(t, x) x^i x^j , \tag{1.30}$$

$$Q^{kl} := \left(M^{kl} - \frac{1}{3} \delta^{kl} M_i^i \right) = \int d^3x \, \rho(t, x) \left(x^i x^j - \frac{1}{3} r^2 \delta^{ij} \right) . \tag{1.31}$$

Every moving object with a non-zero second derivative for the quadrupole moment of mass emits GWs. Note that we are not projecting the waves in the TT gauge, but are dealing directly with the projection of the moment of mass. This is because what matters in GW emission is the transverse motion of it (we are projecting in the plane orthogonal to the line of sight). In particular, for a wave propagating along x^3, we have

$$h_{ij}^{TT} \approx \frac{G}{c^4 r} \begin{pmatrix} \ddot{M}_{11} - \ddot{M}_{22} & 2\ddot{M}_{12} & 0 \\ 2\ddot{M}_{21} & -(\ddot{M}_{11} - \ddot{M}_{22}) & 0 \\ 0 & 0 & 0 \end{pmatrix} , \tag{1.32}$$

where M_{11}, M_{22} and M_{12} are easy to calculate for this configuration.

1.3 Compact binary coalescences

As we concluded in Sec. 1.2.2, thanks to (1.29), the GWs are emitted by the second derivative of a mass quadrupole. Many systems in the Universe are possible candidates as GW emitters. Compact binary stars are the most likely ones. These binary systems consist of Black Holes (BH) and/or Neutron Stars (NS). Both are extremely heavy objects: they have a radius equal (BH) or similar (NS) to their Schwarzschild radius. They originate from the SuperNova (SN) death of a massive star (a star with mass $m > 8\ \mathrm{M}_{\odot}$). Once the original star begins to synthesize the Fe-group elements in its core, the energy production from nuclear fusion is suppressed. This event triggers the entire structure to collapse because no source of energy can counterbalance the gravitational central pull. Anyway, the core of the star can stop this collapse, restoring the necessary pressure gradient. This can be done thanks to its neutrons. This fundamental particles can become a degenerate gas with a sufficiently high pressure. For this case, the pressure depends only on the mass, not on the temperature of the star. This means that the core as a degenerate gas of neutrons becomes incompressible. The envelope material of the progenitor star falls on the core and bounces back. A series of chain reactions gives rise to the SN explosion leading to the birth of a NS. If neutron degeneracy is instead not enough to stop the nucleus collapse, a BH is formed. In nature we can only find rotating Kerr BHs because of the native angular momentum of the stars.

BHs may have other origins besides stellar ones. In particular, they can often be found in the center of bigger galaxies as SuperMassive Black Holes (SMBHs). This kind of objects are BHs with a mass of the order of 10^9 $\mathrm{M}_\odot$. Their initial formation is still under debate. Banik et al. (2018) proposed the possibility for SMBH to be originated from stellar BHs coming from stars born so early in the history of the Universe to have only Hydrogen and Helium as constituents (population III). On the other hand, Shinohara et al. (2023) studied primordial BHs (BHs originated just after the inflation due to mass perturbations) as seeds for SMBHs. Astrophysicists anyway agrees on the fact that these stellar or inflationary seeds have largely grown thanks to both a huge accretion from the surrounding material and mergers with other BHs. In this thesis I will refer to compact stellar objects, unless otherwise specified.

Most of the known stars are present in a binary system, where the two masses orbit each other. If both stars form a compact remnant, they are likely to form a compact binary. Furthermore, the gravitational encounter between two compact objects can bound in a binary system two objects previously separated. The evolution of these compact binary systems has challenged physicists for more than half a millennium. Although the solution for Newtonian relativity is known (Kepler's three laws), the GR solution is quite difficult to achieve. Using PN approximation and NR, as we saw in Sec. 1.1, physicists have tried to solve Einstein's equations for this configuration with different approaches. With both these techniques we conclude that compact binaries are the best setup to produce GW. We will see that, due to this emission, the fate of compact binaries is to merge into a larger object.

1.3.1 Gravitational wave emission from a binary system

We consider the situation in Fig. 1.3: two objects with equal mass m orbiting around their common Center of Mass (CM) with distance r from it and angular velocity ω_o. We set the origin of the coordinates at the CM. Thus, the positions of the masses are given by

$$x^i = \pm(r\cos\omega_o t, r\sin\omega_o t, 0)\,. \tag{1.33}$$

Using (1.30), we find

$$M^{ij} = 2mr^2 \begin{pmatrix} \cos^2\omega_o t & \cos\omega_o t\sin\omega_o t & 0 \\ \cos\omega_o t\sin\omega_o t & \sin^2\omega_o t & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ret}, \tag{1.34}$$

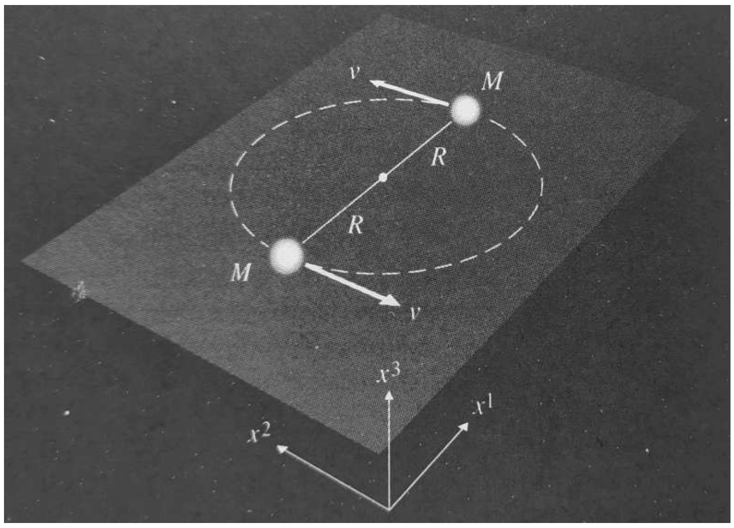


Figure 1.3: *Sketch of the two-body problem* (Milotti, 2022a).

where the subscript "*ret*" indicates quantities calculated at the retarded time. Following (1.29), one can find

$$\begin{aligned} h_{ij}(ct,\mathbf{r}) &= -\frac{4GMr^2}{c^4D}\frac{d^2}{dt^2}\begin{pmatrix} \cos^2\omega_o t & \cos\omega_o t\sin\omega_o t & 0\\ \cos\omega_o t\sin\omega_o t & \sin^2\omega_o t & 0\\ 0 & 0 & 0\end{pmatrix}_{ret}, \\ &= \frac{8GMr^2\omega_o^2}{c^4D}\begin{pmatrix} \cos(2\omega_o t+\phi) & \sin(2\omega_o t+\phi) & 0\\ \sin(2\omega_o t+\phi) & -\cos(2\omega_o t+\phi) & 0\\ 0 & 0 & 0\end{pmatrix} \end{aligned} \tag{1.35}$$

where the additional phase ϕ takes into account the retarded time and D is the luminosity distance of the observer from the source. Introducing different masses and radii for the two objects and using the Kepler third law (1.2) we can write

$$h_{ij}(ct,\mathbf{r}) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}\omega_o^{2/3}\begin{pmatrix} \cos(2\omega_o t+\phi) & \sin(2\omega_o t+\phi) & 0\\ \sin(2\omega_o t+\phi) & -\cos(2\omega_o t+\phi) & 0\\ 0 & 0 & 0\end{pmatrix}, \tag{1.36}$$

where $\mathscr{M}$ is the *chirp mass*, a parameter dependent on the two masses, defined as

$$\mathscr{M} = \frac{(m_1m_2)^{3/5}}{(m1+m_2)^{1/5}}\,. \tag{1.37}$$

Note the fundamental fact that the GW pulsation is twice the orbital frequency of the binary, as there is no difference on the two objects when considering the projected mass quadrupole. This equation represents the GW emission from a compact binary system considering Newtonian dynamics only.

One can demonstrate that GW emission subtracts energy from the emitting system at the rate

$$\mathcal{P}_{GW} = \frac{32G^{7/3}}{5c^5}\mathscr{M}^{10/3}\omega_o^{10/3}\,, \tag{1.38}$$

where $\mathscr{M}$ is in units of $\mathrm{M}_\odot$. On the other hand, the time derivative of the total energy of the binary system is

$$\frac{dE_{tot}}{dt} = -\frac{1}{3}G^{2/3}\mathscr{M}^{5/3}\omega_o^{-1/3}\frac{d\omega_o}{dt}\,. \tag{1.39}$$

Assuming no other loss of energy, the radiated power is opposite to the energy loss of the binary and so

$$\mathcal{P}_{GW} = -\frac{dE_{tot}}{dt} \qquad \Longrightarrow \qquad \frac{32G^{7/3}}{5c^5}\mathscr{M}^{10/3}\omega_o^{10/3} = \frac{1}{3}G^{2/3}\mathscr{M}^{5/3}\omega_o^{-1/3}\frac{d\omega_o}{dt}\,, \tag{1.40}$$

which becomes

$$\frac{d\omega_o}{dt} = \frac{96}{5}\left(\frac{GM_\odot}{c^5}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}\omega_o^{11/3}\,. \tag{1.41}$$

The derivative of the orbital frequency is positive, so the GW emission causes the increasing of the orbital frequency.

Now we can express Kepler's third law (1.2) as

$$\omega_o^2 = \frac{GM}{R^3}\,, \tag{1.42}$$

where $M = m_1 + m_2$ and $R = r_1 + r_2$. Taking the time derivative and arranging the terms, one has

$$\frac{dR}{dt} = -\frac{2}{3}\frac{R^4}{GM}\omega_o\frac{d\omega_o}{dt} = -\frac{2}{3}\frac{R^{5/2}}{(GM)^{1/2}}\frac{d\omega_o}{dt}\,. \tag{1.43}$$

Using Eq. (1.41), we find the evolution for the radial distance:

$$\frac{dR}{dt} = -\frac{64}{5}\frac{G^{7/6}}{c^5}\frac{\mathscr{M}^{5/3}}{M^{1/2}}R^{5/2}\omega_o^{11/3}\,. \tag{1.44}$$

Since ω_o continues to increase thanks to Eq. (1.41) and the first derivative of orbital separation is negative, we can infer that the orbit is shrinking. The GW emission causes a loss of energy, which in turns causes the orbit to shrink, following Eq. (1.44). This shrinkage increases the orbital velocity which consequently boosts the GW production. A runaway loop is started and it continues until the two objects merge into a larger one.

Considering two BHs, we can have an estimate of the maximum orbital angular velocity. Let me impose the orbital distance equal to the sum of the Schwarzschild radii:

$$r_1 + r_2 = \frac{2G}{c^2}(m_1 + m_2)\,. \tag{1.45}$$

At this separation we have a maximum for the emitted power (1.38) and so the maximum value for the angular velocity is given by

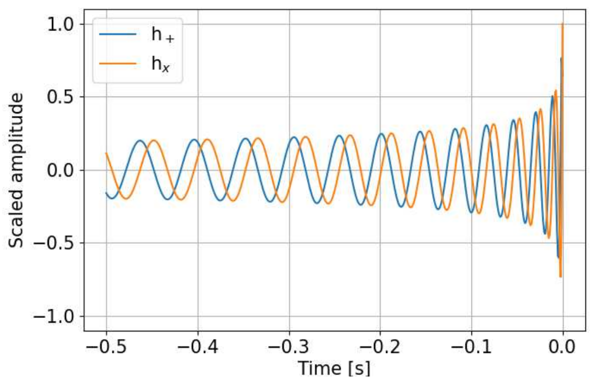
$$\omega_c = \frac{1}{2\sqrt{2}}\left(\frac{GM_\odot}{c^3}\right)^{-1}\left(\frac{M}{M_\odot}\right)^{-1}\,, \tag{1.46}$$

where $M = m_1 + m_2$.

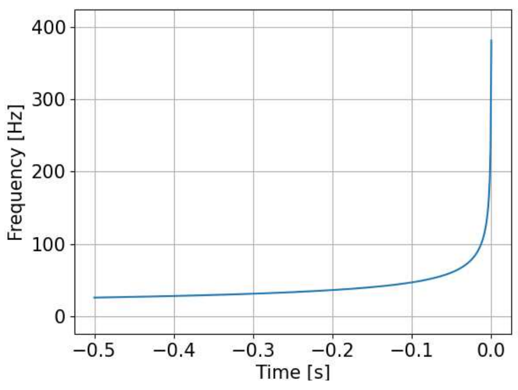
1.3.2 Chirping waveform

This orbital evolution has a characteristic imprint on the shape of the emitted GW, so also on the waveform of the GW signal. Considering that the orbital frequency is twice the GW f frequency, as we saw in Eq. (1.36), we can write

$$f = \frac{\omega_o}{\pi}\,, \tag{1.47}$$



(a)



(b)

Figure 1.4: (a) *Amplitude and* (b) *frequency evolution for a system of two merging BHs of equal mass* $m = 30\ M_\odot$*, with original phase* $\phi = 1$*. The amplitude is normalized to the maximum value. The origin of the time axis is fixed at the coalescence time.*

where ω_o is the orbital velocity. From this, we can modify Eq.(1.41) and express the frequency evolution of the GW as

$$\frac{df}{dt} = \frac{96}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3} f^{11/3}. \tag{1.48}$$

By integrating this last expression, we have the GW frequency as a function of time t

$$f(t) = \left[f_c^{-8/3} - \frac{256}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(t - t_c)\right]^{-3/8}, \tag{1.49}$$

where t_c is the coalescence time and $t \leq t_c$. f_c is the maximum frequency of GW, corresponding to the merger at $t = t_c$. Therefore, from Eq.s (1.46) and (1.47) we can deduce that the maximum frequency is given by

$$f_c = \frac{1}{2\pi\sqrt{2}}\left(\frac{GM_\odot}{c^3}\right)^{-1}\left(\frac{M}{M_\odot}\right)^{-1}. \tag{1.50}$$

The frequency f of the GW and its time derivative df/dt can be measured in a precise instant of time. Thanks to (1.49) and to these f and df/dt measurements, one can calculate the chirp mass $\mathscr{M}$. Furthermore, we can measure the maximum frequency f_c, too. From Eq. (1.50) the total mass M is deduced and we can infer the two masses m_1 and m_2 of the objects thanks to the two combinations ($\mathscr{M}$ and M).

Finally, the GW amplitude (1.36) could be expressed in terms of frequency as

$$h_{ij}(ct, \mathbf{r}) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\begin{pmatrix}\cos(2\pi f t + \phi) & \sin(2\pi f t + \phi) & 0\\ \sin(2\pi f t + \phi) & -\cos(2\pi f t + \phi) & 0\\ 0 & 0 & 0\end{pmatrix}. \tag{1.51}$$

In particular, using the TT gauge as defined in (1.19), we can deduce the following expressions for the several components:

$$h_+ = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\cos(2\pi f t + \phi), \tag{1.52}$$

$$h_\times = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\sin(2\pi f t + \phi). \tag{1.53}$$

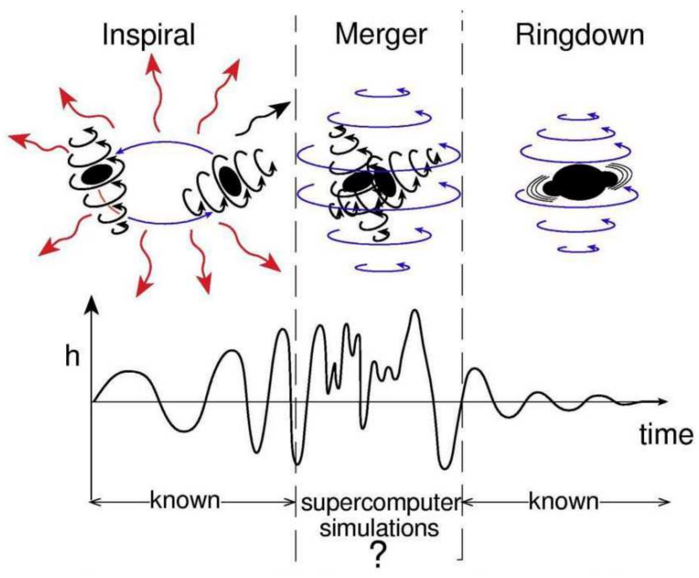


Figure 1.5: *Compact binary sketch representing different phases of coalescence and the corresponding imprint on the waveform* (Schutz, 2004).

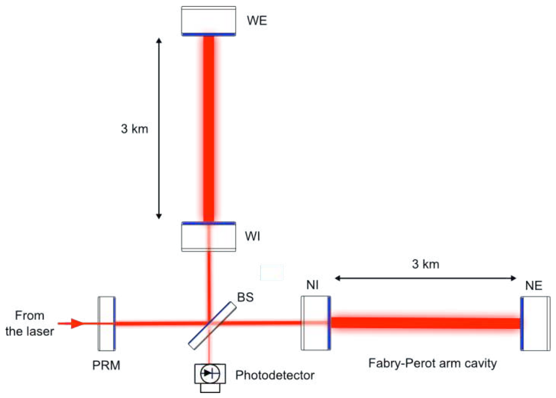
In panel (a) of Fig 1.4 we can see the characteristic chirping waveform produced by a Compact Binary Coalescence (CBC). In this figure I am considering two non-rotating BHs with equal masses $m = 30\ \mathrm{M}_\odot$. The time left to the merge, situated at $t = 0$, is shown on the abscissa, while the amplitudes h_+ and $h_\times$ are shown on the ordinate, calculated respectively by (1.52) and (1.53) with initial phase $\phi = 1$. In panel (b) the evolution of the frequency over time is plotted. As we can see, we have an increasing frequency over time up to the merger phase. This causes an increase in amplitude thanks to the term $f^{2/3}$ in (1.51), too.

In conclusion, a typical GW signal waveform is a chirping waveform, as seen in panel (a) of Fig 1.4. This is due to the merger of a compact binary system whose objects spiral towards their fusion. The energy loss due to the GW emission reduces the orbital separation between the objects. This increases the orbital frequency of the motion, causing the frequency trend over time that we can see in panel (b) of Fig 1.4. When the two objects start getting too close, they merge into a single object, which slows its movement in the ring-down phase. As we will discuss in Sec. 1.5, understanding the physics behind this waveform will be extremely useful for both detection and parameter inference. On the other hand, the detection and characterization of these transient events is indeed a key for testing theoretical models.

1.3.3 Merger and ringdown

In the real world, things are not that simple. As discussed in Section 1.1, Einstein's equations are not easy to solve and our approach has overlooked many physical phenomena. The NR and PN approximation lead to the scenario that we can see in Fig. 1.5. The inspiral phase is the perfect field of study for the PN approximation, with a relatively weak field and slow movements. However, when the BHs are approaching their ISCO, NR must be considered because objects become extremely relativistic. This merger phase can only be studied thanks to supercomputer simulations, taking into account the full numerical solutions of Einstein's equations. In this phase, GWs with maximum power are emitted. After the merger, a single massive object is formed. Any possible deviation of this final object from a spherical configuration causes the emission of GWs with decreasing

(a)



(b)

Figure 1.6: (a) *Virgo interferometer in Cascina, Italy* (Wikipedia, 2023). (b) *Optical configuration of a laser interferometer* (the Virgo scientific collaboration, 2016).

frequency and amplitude (ring-down phase). In fact, for a single non-spherical object, the loss of energy due to the emission of GW results in a slowdown of the rotational motion and in an alignment of the axis of rotation with the main axis of inertia.

1.4 Detectors

The exciting work of detecting GWs started in the late '60s with the idea of the first generation of detectors. Many steps have been taken since then, leading to the first detection of a GW from a CBC on September 14, 2015 (Abbott et al., 2016b). This was possible by the second generation of detectors, operating right now. In this section I will briefly revise modern detectors, highlighting the hardware structure and the expected and obtained results. A greater emphasis will be given to ground-based interferometers, a key topic for the work of this thesis.

At the beginning of our century, the idea of the need for a wider frequency window of sensitivity was gaining ground. This led to the second generation of detectors, right now operating, based on laser interferometry. The basic optical setup is the same of the Michelson interferometer and it is outlined in panel (b) of Fig. 1.6. The laser coming from the left side in the figure is separated into two different beams by a special mirror called a Beam Splitter (BS). The beams return to the BS thanks to two far mirrors, and recombine in one resulting beam sent to a photodetector. The distances between the mirrors and the BS define the so-called arms of the interferometer and are set in such a way that, if nothing is detected, destructive interference is seen at the photodetector. When a GW passes by, the far mirrors act as test masses and follow a motion like the one in Fig. 1.1. As one arm increases in length, the other decreases (and vice versa). This produces a phase shift between the two incoming beams so that their interference is not always destructive and some amount of ligth is detected by the photodetector . From the photodetector measurement, the differential arm length can be inferred and it is converted to a digital output.

The relative deformation (the strain) caused by the incoming GW is of the order

$$h = \frac{\Delta L}{L} \approx 10^{-21} \,. \tag{1.54}$$

where L is the arm length and ΔL is the displacement of the arms due to the GW. The value of L for an interferometer built on the Earth surface can be of a few kms, so the displacement to be measured is of the order of

$$hL = \Delta L \approx 10^{-18} m\,. \tag{1.55}$$

Currently, four interferometers are operational in the fourth observational run: the two Laser Interferometer Gravitational-wave Observatories (LIGO) in Livingston, USA, and in Handford, USA, the European Gravitational Observatory VIRGO (EGO-VIRGO, panel (b) in Fig 1.6) in Cascina, Italy, and the KAmioka GRAvitational wave detector (KAGRA) in the Kamioka mine, Japan[3]. For more recent reviews, the interested reader could refer to Cahillane and Mansell (2022), Nguyen (2021) and Abe et al. (2022), respectively. The first GW signal from a CBC was detected on September 14, 2015 (Abbott et al., 2016b). In total about90 CBC GW signals have been detected during three different observing runs (O1, O2 and O3), up to the Corona virus pandemic in 2020. The joint collaboration published three catalogs with every detected event: GWTC-1 on O1 and O2 data (Abbott et al., 2019), GWTC-2 on O3a data (the first part of O3) (Abbott et al., 2021a) and GWTC-3 on O3b data (Abbott et al., 2021b). All these catalogs have detailed analysis of the event's signal and source. At the time of each publication also the raw GW strain data are made public. A quick catalog of new candidate events for the current O4 run is available as the GraceDB database (the LIGO-Virgo-Kagra scientific collaboration, 2015).

Ground-based interferometers are in their golden age. Now that the CBC statistic is rapidly growing its sample, the search for new sources of GW is challenging physicists worldwide. New transients (supernova explosions) or continuous sources (single neutron stars, GW background) have already dedicated researches. For these reasons, the third generation of detectors is now taking its first steps. The main effort for a ground-based instrument is Einstein Telescope (ET), a triangular-shaped interferometer that will be placed underground for seismic noise reduction. Its location is currently under discussion between the Sos Enattos mine in Sardinia, Italy, and the Meuse-Rhine Euroregion between Belgium, the Netherlands and Germany. ET scientific objectives are described in Branchesi et al. (2023).

1.4.1 Antenna pattern

This subsection aims to answer the following related questions: What does an interferometer measure about the GW? Why is more than one interferometer needed? Observing the panel (b) of Fig 1.7, we can define the detector frame with the two arms of the interferometer parallel to the x and y axes. Another frame may be defined at the source of GWs, with z' the direction to the origin of the detector frame. The detector tensor for this system is

$$D_{ij} = \frac{1}{2}(\hat{\mathbf{x}}_i\hat{\mathbf{x}}_j - \hat{\mathbf{y}}_i\hat{\mathbf{y}}_j) \tag{1.56}$$

and therefore the detector output is given by

$$h(t) = \frac{1}{2}\big(\ddot{h}_{xx} - \ddot{h}_{yy}\big)\,, \tag{1.57}$$

where h_{xx} and h_{yy} are GW components in the detector frame. They depend on the usual h_+ and $h_\times$ polarizations defined in the source frame. Considering the two angles θ and ϕ in the drawing, we can apply a rotation on the two polarizations, to obtain

$$h_{xx} = h_+\big(\cos^2\theta\cos^2\phi - \sin^2\phi\big) + 2h_\times\cos\theta\sin\phi\cos\phi\,, \tag{1.58}$$

$$h_{yy} = h_+\big(\cos^2\theta\cos^2\phi - \cos^2\phi\big) - 2h_\times\cos\theta\sin\phi\cos\phi\,. \tag{1.59}$$

[3]Only the two LIGO interferometers are operational when this thesis was published. Virgo and Kagra will join the observing run in the coming months.

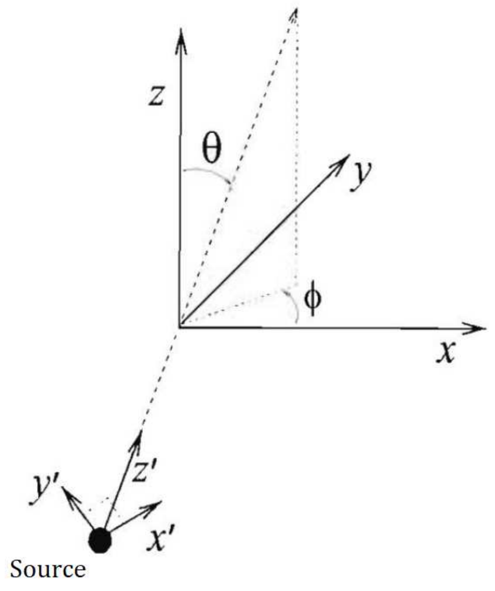


Figure 1.7: *Antenna pattern configuration.* (Maggiore, 2007)

Thus, the output of a single interferometer could be expressed as

$$h(t) = F_+(\theta,\phi)\,h_+ + F_\times(\theta,\phi)\,h_\times\,, \tag{1.60}$$

where

$$F_+(\theta,\phi) = \frac{1}{2}\big(1+\cos^2\theta\big)\cos 2\phi\,, \qquad F_\times(\theta,\phi) = \cos\theta\sin 2\phi\,. \tag{1.61}$$

In (1.60), we can measure $h(t)$ and reconstruct θ and ϕ thanks to statistical analysis. However, with a single interferometer we still have a single equation with two unknowns (h_+ and $h_\times$). We have to work with at least one other interferometer, to have another equation like (1.60), with different $h(t)$, θ and ϕ and resolve the polarizations.

1.4.2 Interferometers' noise budget

Interferometers have outputs dominated by noise, given by unwanted stochastic or deterministic signal. In Fig. 1.8 we can see the noise level measured by LIGO-Handford (LIGO-H). On the abscissa we have the considered frequency, while on the ordinate we have the amplitude spectral density of the interferometer output, which will be defined in sec. 1.5.1. Characterizing the various components of this noise budget is not simple: years of commissioning and upgrading phases are required to reduce it.

Let me follow the order of the plot legend.

- *Measured O3 noise.* This is the measured noise from the third observation run, the actual sensitivity to GWs. Notice the wide frequency band between $10-10^4$ Hz.
- *Advanced LIGO design sensitivity.* The sensitivity that during the O3 commission phase LIGO-H engineers wanted to reach.
- *Total controls noise.* It is related to the read-out noise on the photodetector.
- *Quantum shot noise and quantum radiation pressure noise.* These sources of noise are due to quantum fluctuations in the output vacuum. Shot noise arises from the Poisson fluctuations in the arrival time of the photons on the photodetector, which alter the detected power. The photons impinging on the mirrors cause also a displacement of the mirrors not distinguishable from the one due to the effect on a GW. This is the so-called radiation pressure noise.

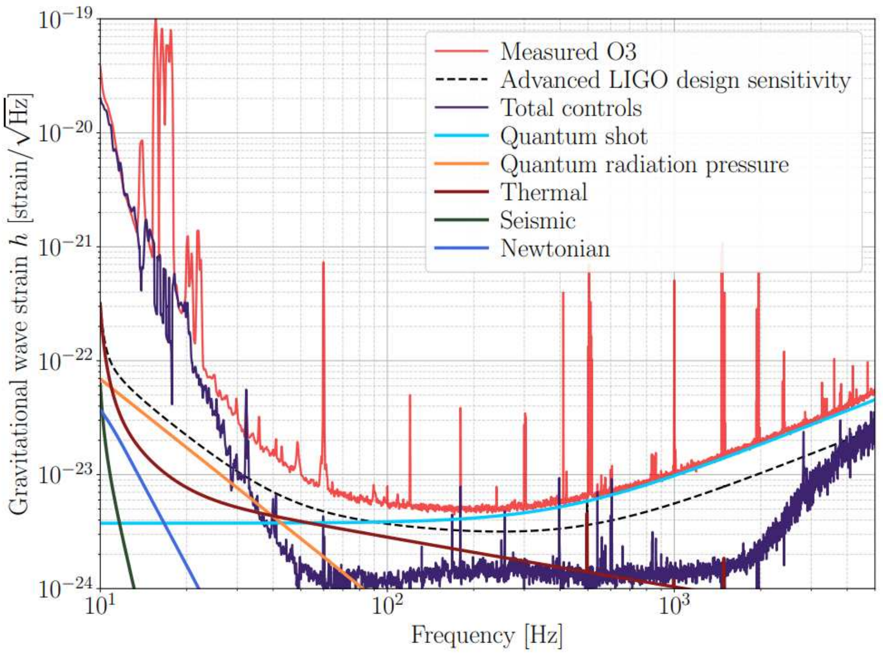


Figure 1.8: *LIGO-Handford noise budget.* (Cahillane and Mansell, 2022)

- *Seismic noise.* I introduce seismic noise first because thermal noise is related to it. It represents ground vibrations, which could be huge (as in the case of earthquakes), but more often they do not have a characteristic behavior due to random human activity. Inside the interferometer, each mirror is suspended thanks to a series of pendulums that attenuate the vibrations of the ground, in order to reduce seismic oscillations as much as possible.
- *Thermal noise.* Thermal noise is due to the thermal fluctuations of the different constituent materials of the interferometer, for example of suspension fibers mentioned before or of the fine coatings that cover several mirrors for better reflection.
- *Newtonian noise.* This type of noise is due to local events on the ground which can cause a false GW signal. For example landslides on a nearby mountain, tidal effects on a nearby sea, and a simple change in the position of a group of people cause a change in the gravitational background. Several accelerometers are distributed throughout the building to record these effects and remove this noise in a post-processing step. This kind of impurity is what currently motivates the concept of constructing underground or space interferometers.

The discussed noise budget show how the noise integrated in a certain time window evolves in frequency. However, GW data contains also non-Gaussian transient noise artifacts, called glitches, that can mask or mimic true astrophysical signals (Abbott et al., 2016a).

1.5 Data analysis

In this section, I will introduce the main problem this thesis wants to address. After a brief summary on basic signal analysis (Sec. 1.5.1), we will see in Sec. 1.5.2 an example of raw interferometric data. Next, in Sec. 1.5.3 I will present several techniques used for the rapid online detection of new signals from raw ground-based interferometer data. In particular, I will focus on the analysis of burst transients, which is of interest for my thesis. Finally, in Sec. 1.5.4 I will present, for completeness, how Bayesian analysis can lead to accurate, but slow, offline parameter inference.

1.5.1 Recap on signal analysis

The output $o(t)$ of a generic physical instrument can be expressed as

$$o(t) = s(t) + n(t)\,, \tag{1.62}$$

where $s(t)$ is the deterministic signal we want to measure and $n(t)$ is any unwanted output other than the signal: the noise of the experiment. Often $n(t)$ is reserved for random processes. For this reason we introduce the concept of a random variable: a number x associated with a possible experimental outcome. The intervals of this outcome have a probability associated:

$$\mathscr{P}(x_0 \leq x \leq x_1) = \int_{x_0}^{x_1} dx\, f(x)\,, \tag{1.63}$$

where $f(x)$ is the probability density. The fundamental statistical quantities for a random variable x are the mean value, the mean value of a function of x itself and the variance, respectively defined as

$$\langle x \rangle := \int_{-\infty}^{+\infty} dx\, x\, f(x)\,, \tag{1.64}$$

$$\langle g(x) \rangle := \int_{-\infty}^{+\infty} dx\, g(x)\, f(x)\,, \tag{1.65}$$

$$\sigma^2 := \langle x^2 \rangle - \langle x \rangle^2\,. \tag{1.66}$$

A random process in an experiment produces a signal, which is a time series of the random variable $x(t)$. Since a different value for x is found for each repetition of the experiment, we need to analyze the statistical properties to characterize the process. Some of them are

- *ergodicity*, which means that the statistical properties of time and ensemble coincide (the experiment could be repeated at a different time and the new result is uncorrelated to the previous one as it comes from a completely new experimental environment);
- *stationarity*, which means that the statistical properties are independent of time;
- *Gaussianity*, which means that $x(t)$ is normally distributed for a fixed time t (if the process is Gaussian, it is completely characterized by its mean value and its variance.).

In our case the signal will be the deterministic incoming gravitational wave signal, while we will consider as noise anything that could contaminate this signal (even deterministic events such as glitches).

Fourier transform As we will see in Sec. 1.5.2, the typical GW signal exits the interferometer buried in a much higher noise. Since it has a characteristic oscillatory behavior, we will certainly be interested in working on the frequency domain. Any signal can be represented as the superposition of several sinusoidal components with a characteristic frequency. These components appears in a series each with a different coefficient. We can use the Fourier transform to go from the time domain to the frequency domain. Considering a generic $s(t)$ signal, if it is true that

$$\int_{-\infty}^{+\infty} dt\, |s(t)|^2 < \infty\,, \tag{1.67}$$

we can define the Fourier transform $\tilde{s}(\omega_o)$ of the signal $s(t)$ as

$$\tilde{s}(\omega) = \int_{-\infty}^{+\infty} dt\, s(t)\, e^{i\omega t}\,, \tag{1.68}$$

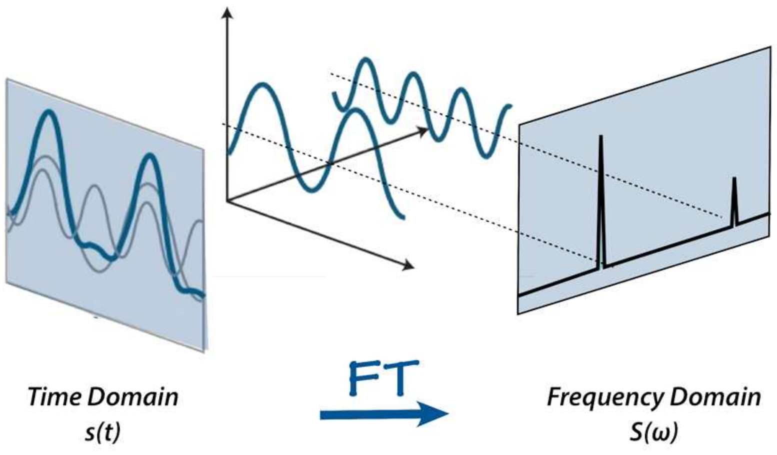


Figure 1.9: *Fourier transform action on the convolution of two sinusoidal functions* (Chaudhary, 2020)

while the inverse Fourier transform is defined as

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} d\omega\, \tilde{s}(\omega)\, e^{-i\omega t}\,. \tag{1.69}$$

This frequency decomposition is based on the linearity of the Fourier transform, which in turn is given by the superposition principle of different frequency components. If we take the Fourier transform of a sinusoidal function, we will have a single frequency component with a δ-like coefficient, as we can see in Fig 1.9.

Power spectral density Speaking about deterministic signal, the Fourier transform is the optimal tool to analyze the system. Anyway, this technique makes little sense when we are dealing with a stochastic signal, because the results will be stochastic, too. However, for a random process the statistical properties are well defined. Thus, we can transfer these quantities into the frequency domain. For this purpose we can define the Power Spectral Density (PSD) of the signal as the Fourier transform of the auto-correlation function $R(\tau)$

$$S_x(\omega) = \int_{-\infty}^{+\infty} d\tau\, R(\tau)\, e^{i\omega\tau}\,, \tag{1.70}$$

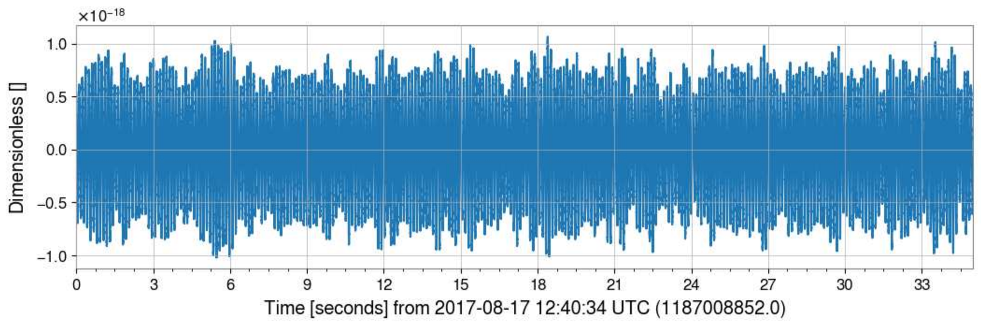
with

$$R(t,t') = \langle x(t)\, x(t')\rangle = R(\tau := t - t')\,, \tag{1.71}$$

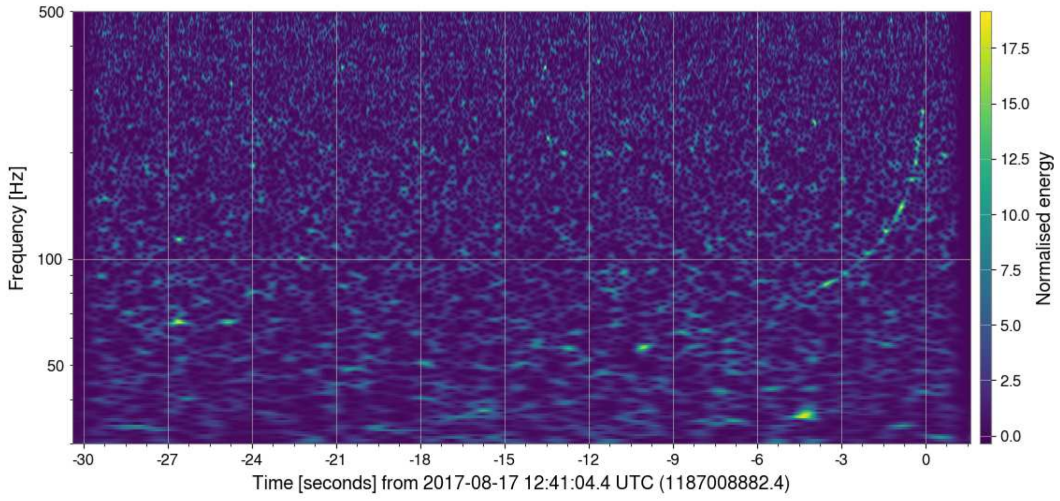
where the last equivalence is valid for stationary processes. Usually the PSD is replaced by the Amplitude Spectral Density $S_x^{1/2}$ (ASD), defined as the square root of the PSD.

1.5.2 Raw interferometer's data

As previously mentioned, the detected events have been published in three different catalogues: GWTC-1 with data from O1 and O2 (Abbott et al., 2019), GWTC-2 with data from O3a (Abbott et al., 2021a) and GWTC-3 with data from O3b (Abbott et al., 2021b). All the GW strain data are publicly available on the Gravitational Wave Open Science Center (GWOSC, the LIGO-Virgo-Kagra scientific collaboration (2019)). In panel (a) of Fig 1.10 we can see a typical raw output of the interferometer: a time series of strain data of the order $\sim 10^{-18}$. The sample rate is 16kHz, but it can easily be reduced to 4kHz.



(a)



(b)

Figure 1.10: (a) *Time series and* (b) *spectrogram around the GW170817 event. The coalescence time is at $t \sim 33s$ in panel* (a) *and at $t = 0s$ in panel* (b).

Zooming in a bit, we could be able to see that the signal is dominated by low frequency components. This time series is taken from LIGO-Handford detector and shows the event GW170817 merging at $t \sim 33$ s: as we can see, no signal can be easily detected by eye from the time series. We will focus on how to highlight the signal in Sec. 1.5.3, but we can anticipate that the easiest way is pass to the frequency domain. For example, we can apply the q-transform (Chatterji et al., 2004) to the time series. This technique is a discrete version of the Fourier transform superimposing on the detector output a small window function whose length depends on the frequency. The window shifts in time to have a temporal resolution. By doing so, we find the spectrogram in panel (b) of Fig 1.10. We can see in the x axis the time to coalescence and in the y axis the frequency component. The intensity of the color indicates the normalized power for each frequency at any given time. We can see the chirping waveform by eye, with the frequency increasing over time.

1.5.3 Detection

During these years several pipelines have been developed to detect the signals. The main differences between them concern the kinds of signal to detect. The different techniques can be seen in tab. 1.1, based on the presence or absence of prior physical information and on the signal length. In general, the term CBC search is referred to pipelines based on matched filtering. Burst searches, instead, look for an excess of energy coherent in different detectors. This would allow to detect possible new sources such as

	Transient	*Continuos*
Modeled	CBC searches	Continuous searches
Unmodeled	Burst searches	Stochastic GWB search

Table 1.1: *Detection techniques for GWs signals.*

gravitational waves from SNs or other unknown sources. The continuous waves research technique is developed to detect binary systems far from coalescence or single NSs rotation, while the research of a stochastic GW background aims to detect this fundamental phenomenon that will have enormous astrophysical and cosmological applications. In this thesis I will focus on a specific pipeline called coherent Wave Burst (cWB), which is part of the Burst searches category. In this section, I will briefly describe each technique.

Matched filtering The basic idea of the matched filtering technique is to cross-correlate a numerically generated (or approximated) template waveform with the raw output of the interferometer (see Vio and Andreani (2021) for a review of matched filters). Suppose we have an output signal of the form

$$s(t) = h(t) + n(t). \tag{1.72}$$

The ideal template waveform matches exactly the GW signal $h(t)$. So, multiplying everything by the optimal template $h(t)$ and integrating over the time interval $t \in [0, T]$, we get

$$\frac{1}{T}\int_c^T dt\, s(t)\, h(t) = \frac{1}{T}\int_c^T dt\, h^2(t) + \frac{1}{T}\int_c^T dt\, n(t)\, h(t)\,. \tag{1.73}$$

Since $n(t)$ and $h(t)$ are not correlated, the dominant term is the first, positive definite. Therefore,

$$\frac{1}{T}\int_c^T dt\, s(t)\, h(t) \quad \xrightarrow{T\to 0} \quad \langle h^2(t)\rangle\,. \tag{1.74}$$

More formally, we are looking for a Kernel function $K(t)$ which outputs “high” when $h(t)$ is present and “low” when it is absent. One can demonstrate that the signal-to-noise ratio (SNR) in the frequency domain is given by

$$\frac{S}{N} = \frac{\int_{-\infty}^{+\infty} df\, \tilde{K}^*(f)\, \tilde{h}(f)}{\left[\int_{-\infty}^{+\infty} df\, |\tilde{K}(f)|^2\, S_n(f)/2\right]^{1/2}}\,, \tag{1.75}$$

where $\tilde{h}(f)$ and $\tilde{K}(f)$ are the Fourier transform respectively of $f(t)$ and $K(t)$. Notice that the SNR depends both on the noise PSD $S_n(f)$ and on the Kernel $\tilde{K}(f)$. To maximize the SNR, we need

$$\tilde{K}(f) = \text{const}\,\frac{h(f)}{S_n(f)} \tag{1.76}$$

and therefore

$$\frac{S}{N} = 2\biggl(\int_c^\infty df\, \frac{|h(f)|^2}{S_n(f)}\biggr)^{1/2}\,. \tag{1.77}$$

The noise PSD $S_n(f)$ determines the maximum SNR that we can get considering the incoming GW signal $h(t)$ and the noise $n(t)$. The actual measured SNR depends instead on the chosen template function $K(t)$.

In practice, we do not know the shape of $h(t)$, nor its arrival time. What we can do is to build a series of chirping template waveforms spanning the plausible parameter space

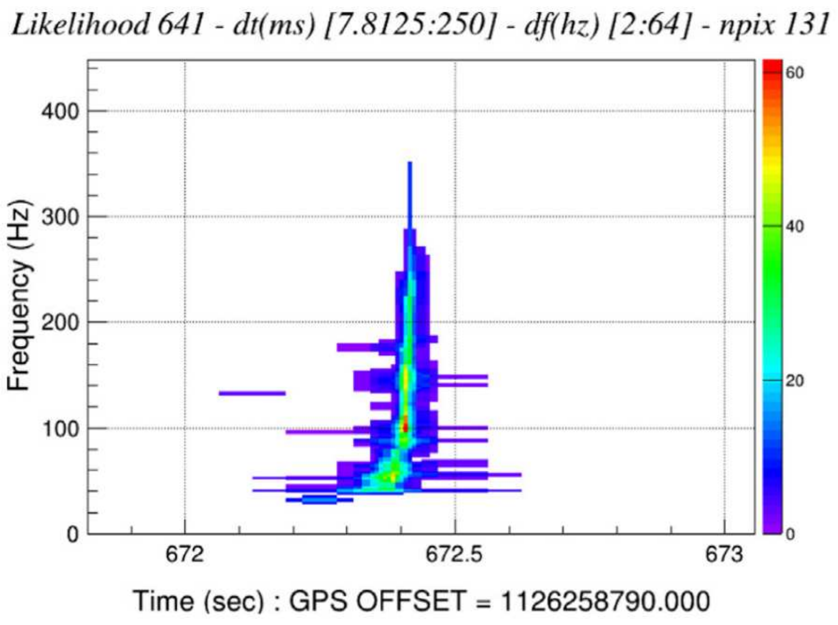


Figure 1.11: *cWB reconstruction of the waveform of GW150914 as a colored time-frequency map.* (Salemi et al., 2021)

and cross-correlate them with the output data. The one with the highest correlation value is the best estimate. Coincidence studies between different interferometers are essential to confirm the detection and avoid false signals, taking into account both the time of flight of the GW between different detectors and the antenna pattern of each interferometer.

The matched filtering technique is the most developed and currently the most efficient tool for detecting CBCs. It is able to detect the GW passing by and makes a rough estimate of the source parameters, thanks to the set of template waveforms built for reference purposes. This procedure must be accomplished with a short computational time in order to notify dedicated telescopes for the detection of a possible electromagnetic counterpart. However, a strong theoretical background is required to perform these detection and inference tasks. In these techniques theory is fundamental to produce or approximate chirping waveforms to be used as templates.

cWB As we said, matched filtering is a brilliant tool for target modeled CBCs, but it requires a huge amount of prior physical informations to create simulated or approximated template waveforms to compare with the raw data. In contrast to this approach, the *coherent Wave Burst* (cWB; Salemi et al. (2021)) pipeline is devoted to the search for unmodeled transient signals. cWB was designed to detect a broad class of GW signals, not just CBCs. It reconstructs their waveforms with minimal assumptions on the source theoretical model. The idea of this pipeline has been developed since the early years of laser interferometers and was designed to detect first events. Indeed, this ingenious code led to the first detection ever of a GW signal on September 14, 2015 (Abbott et al., 2016b). It was later acknowledged in the official description of the 2017 Nobel Prize awarded to R. Weiss, B. C. Barish and K. S. Thorne (the Nobel Committee for Physics, 2018). CBCs are still searched by cWB, but the latter was also used to search for SN GW signal in the entire available data set. No SN signal has been detected so far, but this pipeline will surely be the key tool to detect unknown burst sources.

Taking advantage of the presence of more than one detector since the beginning of observations, cWB searches for coherent signals in the outputs of several interferometers.

The first filter is a regression algorithm removing persistent lines and noise artifacts in the single detector output. Data is then converted into the time-frequency domain thanks to the Wilson-Daubechies-Meyer (WDM) wavelet transform (Necula et al., 2012). This transform relies on a Fourier transform with a window function translating in time, to account for both the frequency and time resolution. Data is after whitened, i.e. normalized by the PSD, to highlight the signal on the now flattened noise. Those pixels whose energy is greater than a given threshold are kept and then compared with selected pixels in other detectors. Together they are combined in a constrained likelihood (a statistical tool that I will describe in Sec. 1.5.4) which takes into account the different antenna patterns and the time delays between interferometer couples. A candidate event is identified maximizing this likelihood, if the maximum is above a certain threshold. After finding the candidate, cWB repeats the search thousands of times on the shifted data to create background noise. By placing specific waveforms in the background, it estimates statistical fluctuations for the reconstruction process. In this phase it determines phenomenological parameters such as the chirp mass $\mathscr{M}$. An output example can be seen in Fig 1.11 as a reconstruction of the waveform in the time-frequency map, where the colored pixels are those with maximum likelihood values above the threshold. In particular, this plot derives from the analysis of the first event GW150914.

The goal of detecting signals with minimal physical preliminary information is a double-edged sword. Thanks to the cWB pipeline we can search for new transient signals without having a precise kind of source as a target. Anyway, not knowing which object is emitting the detected gravitational wave translates into the impossibility of making parameter inference on the source. This issue can be neglected if one wants to exploit cWB to detect new signals kinds. Anyway, for what concerns CBCs, this pipeline is now lacking a parameter inference on the source. Now that matched filtering techniques detect most of the CBC signals, for which cWB was originally thought, this lack starts to be really a defect. As we will see in Sec. 1.5.4, this is the problem that the work that begins with my thesis wants to solve. We will see how to handle a fast parameter inference with as less prior physical information as possible.

Continuous waves search The search for modeled continuous waves is mainly based on monochromatic signals. This is the case for compact binaries far from the coalescence, which still evolves with a low frequency derivative, or for spinning asymmetric neutron stars. The idea of these searches is to be carried out on the entire time series at once. As a first approximation, the sensitivity around a certain frequency f is $S_n^{1/2}/T^{1/2}$, where T is the time interval of the measurement. The SNR instead increases as $T^{1/2}$. Thanks to the likelihood that takes into account antenna patterns, considering in this way in the search the different directions of the incoming GW, we can “point” the interferometers in a certain direction and look for the emission from known sources. Currently, no detection was found and only upper limits on the detectable strain have been set (Abbott et al., 2022).

Stochastic gravitational wave background search Like the continuous wave search, the search for a gravitational wave background relies only on analyzing the entire time series at once. In this case we are not looking for a modelled signal, but we want to universally correlate the same apparent noise source. The idea of the investigation is simply to subtract the noise from the interferometers and look at what is left. In practice, characterizing the whole noise is quite difficult. A triangular-insensitive GW configuration (as the one for LISA; Amaro-Seoane et al. (2017)) could definitely be a key ingredient to this step. The gravitational wave background will be a fundamental ingredient for testing cosmological and astrophysical models. Also in this case, only the upper limits to the detectable strain from ground-interferometers studies have been placed (Abbott et al., 2021c). A 4σ evidence for a gravitational wave background has been found from the 15-year PTA database (Lynch (2015); Agazie et al. (2023)).

1.5.4 Parameter inference

After the detection, high relevance events are saved in the GraceDB (the LIGO-Virgo-Kagra scientific collaboration, 2015) database. An offline analysis can now start to infer the 15 parameters that the GW signal could provide in case of a binary black hole: 2 masses, 6 spin components, the orbit eccentricity, the inclination of the orbit with respect to the line of sight, the luminosity distance and 2 coordinates for the position of the source in the sky. For binary neutron stars other parameters are needed. I will briefly introduce the basic statistics behind these methods and how they are implemented.

Bayesian statistics This brilliant way of doing statistics is founded on the Bayes' theorem defined by T. Bayes in the XVIII century. Unlike a Frequentist approach, Bayes statistics introduces a subjectivity into the game through the use of statements coupled with random variables. The Bayes' theorem states that: *given a dataset d and a hypothesis H, the probability that the hypothesis H is true is given by the probability of having the data d given the true hypothesis times the probability that the hypothesis is true, normalized by the entire probability of having the data.* In formulas, the Bayes' theorem is given by

$$\mathscr{P}(H|d) = \frac{\mathscr{P}(d|H)\,\mathscr{P}(H)}{\mathscr{P}(d)}\,. \tag{1.78}$$

This becomes clearer if we think of a physical model M as hypothesis, with associated parameters θ. The probability of having the model M with those parameters θ given the observed data d is represented by the probability of having d given the true M multiplied by the probability for the M, normalized by the whole probability of having the data set d, which results in

$$\mathscr{P}(\theta|d) = \frac{\mathscr{P}(d|\theta)\,\mathscr{P}(\theta)}{\mathscr{P}(d)}\,. \tag{1.79}$$

$\mathscr{P}(\theta)$ is the *prior* information on the model with associated parameters and must be given by the user himself: here the subjectivity of statistics comes into play (in the literature there are several prior optimized forms). $\mathscr{P}(d|\theta)$ is called the *likelihood* for the parameters. It represents the plausibility of the data given the true model. It mostly depends on the kind of problem one wants to solve, but one could make several assumptions to reduce to simplified forms (for example, considering the logarithm of the probability). $\mathscr{P}(d)$ is called evidence and normalizes the numerator. It is defined through the marginalization on the parameters, which means

$$\mathscr{P}(d) = \int_{\omega_o} d\theta\, \mathscr{P}(d|\theta)\,\mathscr{P}(\theta)\,, \tag{1.80}$$

where ω_o represents the entire parameter space. Substituting this into (1.79), we can see that this marginalization might actually be a normalizing factor. Since there is usually no calculation available for this, we can estimate the numerator (1.79) and scale it later to useful values. Last but definitely not least, $\mathscr{P}(\theta|d)$, opposite to the prior, is the posterior of the parameters and it is the object to determine. It represents the plausibility of our model given the data. It depends on the choice done for the prior $\mathscr{P}(\theta)$, but it can be shown that the more data we have, the less informative the prior is.

The idea for a parameter inference is to evaluate the target posterior probability distribution $\mathscr{P}(\theta|d)$ and to sample parameter values from it. The problem is that the posterior is usually impossible to integrate to get a sample. Numerical sampling techniques, such as rejection sampling or Markov Chain Monte Carlo (MCMC), come to the rescue.

Gravitational wave parameter inference The GW signals from CBCs are a perfect environment to apply this kind of statistics. The developed pipelines for parameter estimation relies on enormous prior theoretical knowledge to have strong structured priors on the 15 parameters. Several PYTHON libraries, such as BILBY (Ashton et al., 2019), have built-in functions to perform this parameter inference. This type analysis must be done off-line, after the full understanding of the event phenomenology. This is required because the Bayesian pipelines need an order of six months of work to operate on a single event. The main reason is due to the MCMC covering a large range of parametric spaces with great detail. Despite the time length of this work, at the end of the analysis the fifteen parameters are inferred much more strongly than using matched filtering.

Need for a data-driven parameter inference Most of the currently popular pipelines are based on numerically generated waveforms, as previously explained. This can be done only with an extremely solid physical background. Matched filtering, for example, uses these templates to highlight signal over noise. On the other hand, Bayesian parameter estimation utilizes these models to infer CBC source parameters from the data captured by the matched filtering. The former is a fast-running pipeline that detects the signal and performs rapid parameter estimation to contact dedicated observatories for an electromagnetic counterpart. The latter is instead a computationally heavy pipeline that operates offline in the next months after the detection. Both of these pipelines need solid physical models, developed in the last century, as we saw in Section 1.1.

On the other hand, the purpose of cWB is to detect signals from new sources, avoiding the need for an extensive theoretical background. Anyway, this algorithm can only lead to infer phenomenological parameters of the signal. At the moment it is not possible to make any profound estimation on all the 15 parameters.

This is the problem that this thesis wants to address. In the next chapter I will introduce Physics Informed Neural Networks (PINN) as a Machine Learning (ML) technique to learn the basic physics behind the input data without the need for a huge physics background. PINNs are able to reconstruct physical differential equations and estimate parameters in the meantime. The reconstructed waveform of cWB could be used as input to the PINNs, which will learn the differential equations behind it, estimating the source parameters. The key tool of PINNs is to work really well even with a little physical ansatz.

Chapter 2

Machine learning

In this chapter I introduce the methodology to solve the problem presented above. We want to achieve a fast data-driven method for parameter inference. From now on, we will introduce the strong assumption to have a BBH as source of gravitational waves. Anyway, we will start considering only Newtonian dynamics, testing and exploiting data-driven algorithms velocity and accuracy to compensate the missing physics.

I will present Machine Learning (ML) based algorithms as key tools to address this problem. In particular, I will show how the recently developed Physics Informed Neural Networks (PINNs) allow to learn partial differential equations (PDEs) from experimental data. This task would be carried out optimizing the desired parameters.

The aim of this chapter is to be a review on ML and on PINN in particular. It will be organized as follows. I will introduce basic ML concepts, statistics, and techniques in sec. 2.1, always with an eye to parameter inference. In sec. 2.2 I will examine the Deep Learning (DL) techniques of Neural Networks (NNs). In sec. 2.3 I will talk about PINNs as a physical implementation on the NNs themselves. In sec. 2.4 I will briefly talk about the ML techniques already implemented in GW data analysis. Sec. 2.1 and sec. 2.2 are based on ML's review for physics by Mehta et al. (2019), sec. 2.3 mainly on articles by Raissi et al. (2019) and Rackauckas et al. (2021) and sec. 2.4 on the ML for GW review done by Cuoco et al. (2020).

2.1 Basic concepts

Machine Learning (ML) is a field of information technology that has emerged in recent decades mainly for industrial interests and to tackle scientific problems. The working principle of ML techniques is to develop self-improving algorithms by learning information from input data. Based on these latter and their interpretation, ML algorithms are able to make predictions, make choices and/or determine the model implicit in the data itself (infer its parameters). Other brilliant applications of ML can be found in information engineering (image recognition), bioinformatics and genomics. Anyway, its basis is on statistical studies. We will see in this section the statistical work behind a ML algorithm, introducing then the ML techniques mainly used in physics.

2.1.1 Learning process

ML is not a single technique, but a set of different techniques that share the same basic idea to learn information from data and draw conclusions (classify objects, make predictions, make choices, ...). To set up a ML problem, the following ingredients must be set:

- a *dataset* $\mathcal{D} = (\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of independent variables, while $\mathbf{y} = (y_1, \ldots, y_n)$ is a vector of dependent variables;
- a *model* $f(\mathbf{x}; \boldsymbol{\xi})$, which is a function $f : \mathbf{x} \to \mathbf{y}$ with the set of parameters $\boldsymbol{\xi}$;
- a *cost function* $\mathscr{C}(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\xi}))$.

We assumed that the independent variables $\mathbf{x}$ and $\mathbf{y}$ are constrained by an unknown model. f is the function approximating this unknown model thanks to the parameters $\boldsymbol{\xi}$. How can we optimize the $\boldsymbol{\xi}$ parameters to know if f is a good approximation for the model behind the data or not? We can minimize the associated cost function $\mathscr{C}$ which compares the actual data $\mathbf{y}$ with the predicted data, calculated thanks to the approximation $f(\mathbf{x}; \boldsymbol{\xi})$. A classical example of $\mathscr{C}$ is the squared error, given by

$$\mathscr{C}(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\xi})) = \sum_{i=1}^{n} \big(y_i - f(x_i; \boldsymbol{\xi})\big)^2 , \tag{2.1}$$

where n is the total number of variables in $\mathbf{x}$ (and in $\mathbf{y}$). By optimizing $\boldsymbol{\xi}$, we can modify the approximation f and look for a minimum of $\mathscr{C}(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\xi}))$.

Now it is essential to specify that, before carrying out any type of evolution or statistical analysis, the dataset must be randomly divided into two subsets: the training set $\mathbf{y}_{train}$ and the test set $\mathbf{y}_{test}$. The cost function $\mathscr{C}$ will be minimized only considering the training dataset. The idea is to find the values for the components $\boldsymbol{\xi}$ that minimize $\mathscr{C}$, i.e.

$$\hat{\boldsymbol{\xi}} = \arg\min_{\boldsymbol{\xi}} \big\{\mathscr{C}(\mathbf{y}_{train}, f(\mathbf{x}_{train}; \boldsymbol{\xi}))\big\} . \tag{2.2}$$

This search for the minimum is like performing a fit of f on the data $\mathbf{y}_{train}$. Once a value for $\hat{\boldsymbol{\xi}}$ is fixed, the act of calculating the $\mathscr{C}$ value with the test dataset permits us to validate our f optimization thanks to new data. We can define in this reasoning two different errors, the so-called *loss functions*:

$$E_{in} = \mathscr{C}(\mathbf{y}_{train}, f(\mathbf{x}_{train}; \boldsymbol{\xi})) , \qquad E_{out} = \mathscr{C}(\mathbf{y}_{test}, f(\mathbf{x}_{test}; \boldsymbol{\xi})) , \tag{2.3}$$

where E_{in} is called the in-sample error and is calculated with the training dataset only, while E_{out} is called the out-of-sample error and is calculated with the test dataset only. Usually one could think that the minimization of $\mathscr{C}(\mathbf{y}_{train}, f(\mathbf{x}_{train}; \boldsymbol{\xi}))$ is sufficient to have the full model behind every possible $\mathbf{x}$ data. Anyway, optimizing on a fraction of the data could lead to an overfitting of this particular subset. Therefore, a test set is needed to verify new data, different from the training one.

2.1.2 Linear regression

In this section I will introduce a simple application for ML that will be useful for understanding the much more complex algorithms of the next chapters: linear regression.

Suppose we have n samples $\mathcal{D} = \{(y_i, \mathbf{x}^{(i)})\}_{i=1}^{n}$, where $\mathbf{x}^{(i)}$ is the ith observation vector of size p, equal to the number of observables. y_i is its scalar response. We can insert the observation samples of $\mathbf{x}^{(i)}$ into a $n \times p$ matrix $\mathbf{X}$. Let g be the real (unknown) function that did generate the sample y_i as

$$y_i = g(\mathbf{x}^{(i)}; \boldsymbol{\xi}_{true}) + \epsilon_i , \tag{2.4}$$

with ϵ_i some white random noise. What we want to do is to find a function f which best fits the data y_i, approximating g. In this thesis we are interested in parameter estimation, in the sense that we want to find the optimal $\hat{\boldsymbol{\xi}}$ such that $f(\mathbf{x}; \hat{\boldsymbol{\xi}})$ is the best approximation

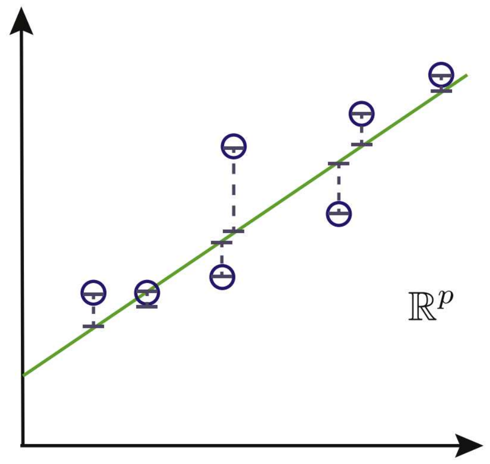


Figure 2.1: *Result of the linear regression for a one-dimensional* $\boldsymbol{x}$*. The circles are* (x_i, y_i) *points and* f *is the red line.* (Mehta et al., 2019)

that we can get for g. Once that is done, we can use this f to make predictions about the response y_0 given the new test data $\mathbf{x}_0$.

To solve this problem, we can minimize the following cost function

$$\mathscr{C}(\mathbf{X}, \boldsymbol{\xi}) = ||\mathbf{X} \cdot \boldsymbol{\xi} - \mathbf{y}||_2^2\,, \tag{2.5}$$

where $||\mathbf{x}||_p = (|x_1|^p + \cdots + |x_2|^p)^{1/p}$. Let $f(\mathbf{x}^{(i)}, \boldsymbol{\xi})$ be defined as

$$f(\mathbf{x}^{(i)}, \boldsymbol{\xi}) = \boldsymbol{\xi}^T \mathbf{x}^{(i)}\,. \tag{2.6}$$

Minimizing this $\mathscr{C}$ is equivalent to minimizing the sum of all the residuals of $\mathbf{x}^{(i)}$ with respect to the hyperplane defined by this f, as seen in fig. 2.1 for $p = 1$. This first application is the basis of ML reasoning and is used to fit linear problems like the one we have seen. Usually several smoothing techniques are used to mitigate the high-dimension limit.

From this simple example, we can understand another fundamental thing about ML operations. Considering the definitions of E_{in} and E_{out} from (2.3), it can be shown that the difference of their means is given by

$$|\overline{E}_{in} - \overline{E}_{out}| = 2\sigma^2 \frac{p}{n}, \tag{2.7}$$

where σ^2 is the variance of the intrinsic noise ϵ_i, p is the number of observables in each sample and n the number of samples. The more data we have, the more we can learn from them. The more observables we consider, the less we can learn from the same amount of data.

2.1.3 Optimization methods

One of the main problems in ML is to find a way to minimize the cost function $\mathscr{C}$ with respect to each component of $\boldsymbol{\xi}$ (each parameter of model) at the same time. Usually $\mathscr{C}$ is not as simple as (2.1) and the derivative calculations could be very computationally expensive. We will see in this section the routes developed to simplify it.

Newton's method The goal of the presented different methods is always the same: to find an intelligent way to update the parameters $\boldsymbol{\xi}$ to minimize $\mathscr{C}$. The distance between

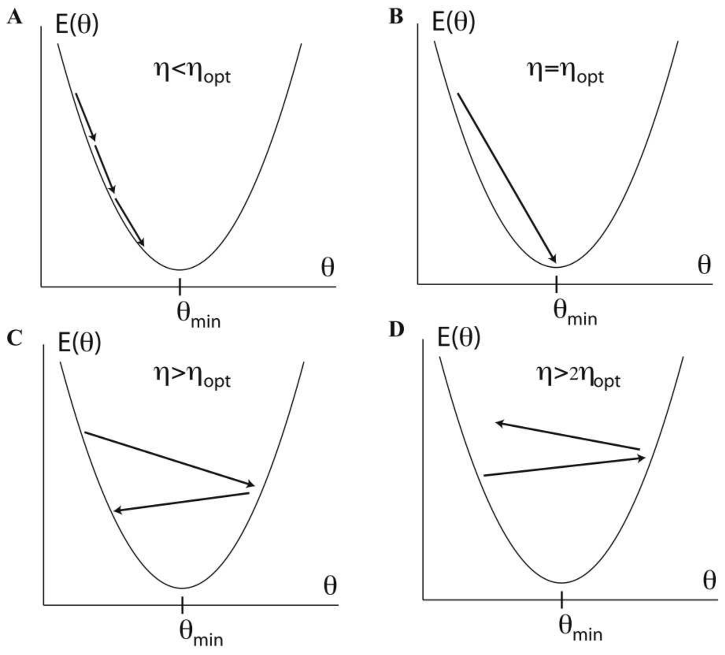


Figure 2.2: *Effect of the learning rate η on convergence.* (Mehta et al., 2019)

two values of $\boldsymbol{\xi}$ at iterations t and $t+1$ is called *step* $\mathbf{v}$, with the same dimension as $\boldsymbol{\xi}$. In Newton's method, which inspires every other method, we choose the $\mathbf{v}$ step as expansion term for $\mathscr{C}$. What we want to minimize is the second-order expansion

$$\mathscr{C}(\boldsymbol{\xi}+\mathbf{v}) \approx \mathscr{C}(\boldsymbol{\xi}) + \nabla_{\boldsymbol{\xi}}\mathscr{C}(\boldsymbol{\xi})\,\mathbf{v} + \frac{1}{2}\mathbf{v}^T H(\boldsymbol{\xi})\,\mathbf{v}\,, \tag{2.8}$$

where $H(\boldsymbol{\xi})$ is the Hessian matrix of $\mathscr{C}$ (note that we changed notation from $\mathscr{C}(\mathbf{y}, f(\mathbf{x};\boldsymbol{\xi}))$ to the simplified one $\mathscr{C}(\boldsymbol{\xi})$). With the optimal value of $\mathbf{v}$ ($\mathbf{v}_{opt}$) we step right on the minimum. So, differentiating the previous formula we have

$$\nabla_{\boldsymbol{\xi}}\mathscr{C}(\boldsymbol{\xi}) + H(\boldsymbol{\xi})\,\mathbf{v}_{opt} = 0\,. \tag{2.9}$$

Rearranging this expression, we have the desired rule for updating the $\boldsymbol{\xi}$ parameters:

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \mathbf{v}_t\,, \qquad \text{with} \qquad \mathbf{v}_t = H^{-1}(\boldsymbol{\xi}_t)\nabla_{\boldsymbol{\xi}}\mathscr{C}(\boldsymbol{\xi}_t)\,, \tag{2.10}$$

where t is the iteration index.

Gradient descent A simplified version of this method is given by the simplest and most used optimization method: the gradient descent. In this case the update rule is given by

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \mathbf{v}_t, \qquad \text{with} \qquad \mathbf{v}_t = \eta_t\nabla_{\boldsymbol{\xi}}\mathscr{C}(\boldsymbol{\xi}_t)\,. \tag{2.11}$$

The inverse of the Hessian matrix is replaced by the so-called *learning rate* η_t. We update $\boldsymbol{\xi}$ thanks to the use of η_t, which drives the step together with the gradient. The sign of the step is given by the sign of the gradient, while the absolute value is governed by the learning rate. In the case of (2.11), we have a fixed value, while for Newton's method (2.10) it is regulated thanks to the Hessian matrix. Having a fixed value is quite risky for the reason we see in fig. 2.2, where in Mehta et al. (2019)'s notation $E(\theta) = \mathscr{C}(\boldsymbol{\xi})$. If η is exactly equal to its optimal value η_{opt}, on the first step we arrive at the minimum, while if η is too small, the computational cost becomes higher because we need many passes. Furthermore, if η is too small, we may be constrained to local minima, but if too large, we may skip the real minimum or diverge. It can be shown that η_{opt} is just the inverse of the Hessian matrix $H^{-1}(\boldsymbol{\xi})$ because, as we know from mathematical analysis, it provides fundamental informations about the curvature of the function. This can drive the step v much better than a constant value.

ADAM ADAM (Kingma and Ba, 2017) is a rather advanced optimization algorithm which introduces learning rate adaptivity for several parameters. It is based on the running averages of the first ($\mathbf{m}_t$) and second ($\mathbf{s}_t$) moment of the gradient and on the bias corrections to account for the running averages. The update rule is given by

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \eta_t \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}\,, \tag{2.12}$$

with

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-(\beta_1)^t}\,, \qquad \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-(\beta_2)^t}\,, \tag{2.13}$$

where

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\mathbf{g}_t^2\,, \qquad \mathbf{s}_t = \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)\,\mathbf{g}_t^2\,, \tag{2.14}$$

with

$$\mathbf{g}_t = \nabla_{\boldsymbol{\xi}} \mathscr{C}(\boldsymbol{\xi})\,. \tag{2.15}$$

A sort of memory of the past updates of $\boldsymbol{\xi}$ is defined, depending on the two parameters β_1 and β_2, to avoid local minima using an inertia factor. ϵ is instead a regularization parameter with the same intent.

RMSprop RMSprop works mostly with mini-batches, which are subsets of the training data. The algorithm chooses stochastically which mini-batches to select. Within this new simpler dataset, a running average of the gradient is used to update parameters (Hinton et al., 2018). Following the notation used by Graves (2014), the update rule is

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t + \Delta_t\,, \tag{2.16}$$

whit

$$\Delta_t = \beth \Delta_{t-1} - \gimel \frac{\epsilon_t}{\sqrt{n_t - g_t^2 + \daleth}}\,, \tag{2.17}$$

where

$$n_t = \aleph n_{t-1} + (1-\aleph)\epsilon_t\,, \tag{2.18}$$
$$g_t = \aleph g_{t-1} + (1-\aleph)\epsilon_t \tag{2.19}$$

and

$$\epsilon_t = \frac{\partial \mathscr{C}}{\partial \boldsymbol{\xi}_t}\,. \tag{2.20}$$

$\beth$, $\gimel$, $\daleth$ and $\aleph$ are parameters to be set by hand. Zaman et al. (2021) demonstrated that RMSprop is the best optimization algorithm when dealing with Recurrent Neural Networks (see Sec. 2.2.3).

Broyden–Fletcher–Goldfarb–Shanno (BFGS) Since the Hessian matrix is usually too computationally heavy to calculate, several techniques are used to approximate it. In this procedure introduced by Fletcher (2000), an approximation for $H(\boldsymbol{\xi})$ also evolves. In this case we use the quasi-Newton equation

$$B_t \mathbf{v}_t = \nabla_{\boldsymbol{\xi}} \mathscr{C}(\boldsymbol{\xi}_t)\,, \tag{2.21}$$

where B_t is the approximation for $H(\boldsymbol{\xi})$. The update for $\boldsymbol{\xi}$ is given by

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \gamma_t \mathbf{v}_t, \tag{2.22}$$

where $\gamma_t > 0$. The direction of $\mathbf{v}_t$ is found in the computation thanks to a line search minimizing $\mathscr{C}(\boldsymbol{\xi}_t + \gamma_t \mathbf{v}_t)$ to different directions. B_t must satisfy the secant condition

$$B_{t+1} \mathbf{s}_t = \mathbf{a}_t\,, \tag{2.23}$$

where

$$\mathbf{s}_t = \boldsymbol{\xi}_{t+1} - \boldsymbol{\xi}_t\,, \qquad \mathbf{a}_t = \nabla_{\boldsymbol{\xi}} \mathscr{C}(\boldsymbol{\xi}_{t+1}) - \nabla_{\boldsymbol{\xi}} \mathscr{C}(\boldsymbol{\xi}_t)\,, \tag{2.24}$$

and is updated thanks to

$$B_{t+1} = Bt + \frac{\mathbf{a}_t \mathbf{a}_t^T}{\mathbf{a}_t^T \mathbf{s}_t} - \frac{B_t \mathbf{s}_t \mathbf{s}_t^T B_t^T}{\mathbf{s}_t^T B_t \mathbf{s}_t}\,. \tag{2.25}$$

This useful algorithm will be the optimisation algorithm used in chap. 3.

2.2 Neural networks

A large branch of ML is Deep Learning (DL), consisting of Neural Network (NN) techniques. These NNs, as the name suggests, are built as a network of neurons communicating between each other. At the moment of their creation the inspiration was trying to recreate the human brain thanks to computer science (Minsky, 1975). Anyway, they gained the attention of the ML community only after decades, thanks to the brilliant work done by Krizhevsky et al. (2012). They lowered the error rate of an image recognition algorithm by twelve percent. Since then, NNs have emerged as one of the most powerful and widely used supervised learning techniques. Despite the large applicability for industrial purposes (image recognition, decision making, . . .), many branches of physics are exploiting NNs for their own purposes. Statistical physics is closely related to their internal structure and many different tasks have been achieved, together with quantum computing. Furthermore, astrophysics and cosmology can both apply convolutional neural networks (CNNs) to image recognition (Carleo et al., 2019).

In this section we will see the basic concepts of NNs, focusing the discussion on the algorithms that will be used in the next chapters of this thesis.

2.2.1 Neuron

As anticipated, the building block of NNs are the so-called *neurons*. They are defined as functions taking the d input features $\mathbf{x} = (x_1, \dots, x_d)$ and producing a scalar output y. The formula behind each neuron can be expressed as

$$y = \mathrm{NN}(\mathbf{x}; \boldsymbol{\xi}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)\,, \tag{2.26}$$

where we can define σ as the neuron activation function (usually hand-given), $\mathbf{w}$ as the vector of different weights for different inputs, b as the neuron bias and $\boldsymbol{\xi} = (b, \mathbf{w})$. The

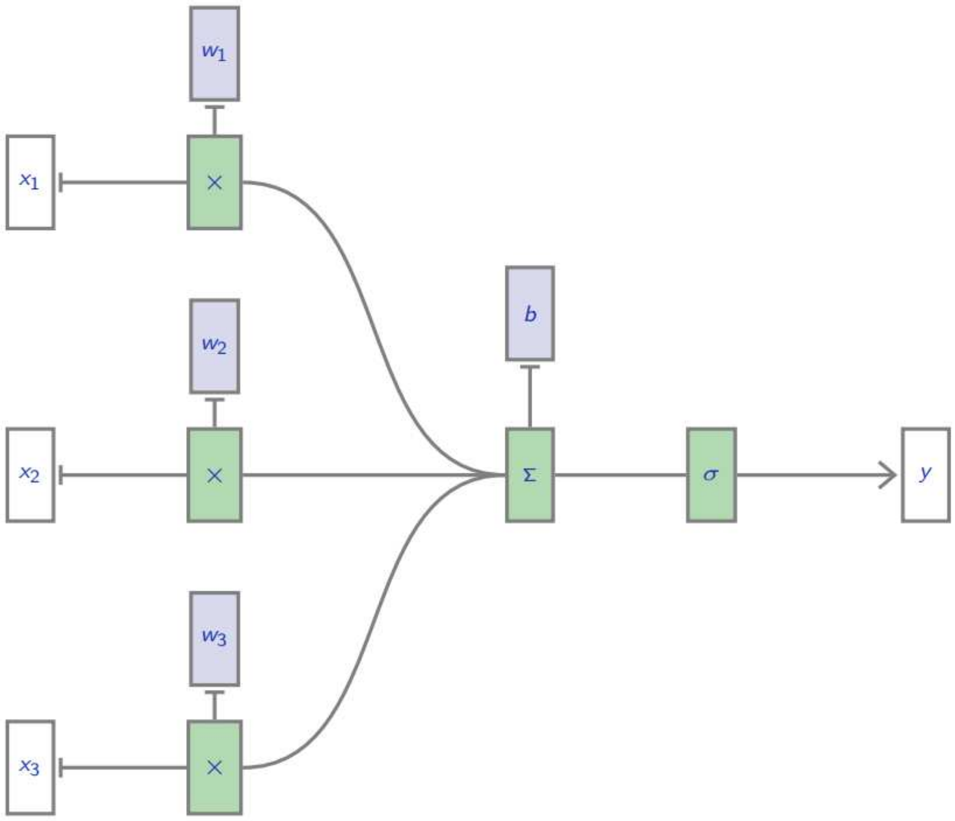


Figure 2.3: *Single neuron scheme. In white we can see different values, in light violet parameters and in green operations.* (Tefas and Nousi, 2023)

idea of the calculation can be seen in Fig. 2.3: several inputs x_i are combined with weights w_i and then summed together with the bias. This is the linear part of the neuron. Instead, when the activation function σ is applied, non-linearity is introduced to compute y.

This simple neuron concept is really a key tool, because different weights could be modified to have different y results. In any case, a single neuron is sufficient to handle a linear regression, but not to solve complex tasks such as object classification or non-linear fitting. For this reason we have to introduce networks of neurons.

2.2.2 Multi-layer network

The structure of a NN is shown in Fig. 2.4, where each circle represents a neuron. Each output y of a single neuron is the input of the next one. A network with as many layers of neurons as desired can be built following this logic. In the first layer inputs are hand-given to the network and it is called *input layer*. Intermediate layers are called *hidden layers* and the last layer is the *output layer* returning the scalar output of the network. Because of the multilayer structure we usually speak of DL referring to NNs. Each neuron works as (2.26) and so we have

$$y_i = \mathrm{NN}_i(\mathbf{x}_i; \boldsymbol{\xi}_i) = \sigma_i(\mathbf{w}_i y_{i-1} + b_i)\,, \tag{2.27}$$

where i is the index of the neuron. The y_i term is the input for the next neuron. For example, a feed-forward neural network of three layers in total, with one neuron per layer, input $\mathbf{x}$ and scalar output y is given by

$$y = \mathrm{NN}(\mathbf{x}; \boldsymbol{\xi}) = \mathbf{w}_3\sigma_2(\mathbf{w}_2\sigma_1(\mathbf{w}_1 \cdot \mathbf{x} + b_1) + b_2) + b_3\,, \tag{2.28}$$

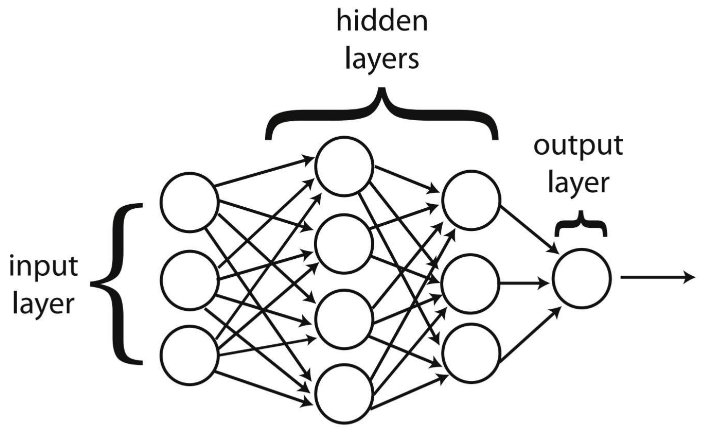


Figure 2.4: *Neural network scheme. The circles represent different neurons, while the arrows represent the output-input connections between each neuron.* (Mehta et al., 2019)

where we used $\sigma_3 = \mathbb{1}$.

The key feature differentiating NNs from other ML techniques in terms of performance is the use of hidden layers. As the number of both parameters and activation functions increases, the NN can reconstruct every possible type of function. Hidden layers generate step functions with arbitrary offsets and heights, which are added together to approximate each arbitrary function. For this reason, this layering of neurons is extremely useful for solving complex non-linear tasks, such as image reconstruction thanks to Convolutional Neural Networks (CNNs), parameter estimation for different features and solving of differential equations. Defining the optimal network architecture is really an art left to the developer, but some PYTHON and JULIA packages have been created to easily build such complex codes: PYTORCH (Paszke et al., 2019) , TENSORFLOW (Abadi et al., 2016) and SCIML (Rackauckas et al., 2021). We will see the latter in detail in sec. 2.3.4.

2.2.3 Recurrent neural networks

Feedforward NNs, as we have seen, have the ability to approximately reproduce any type of function exploiting the interconnections of neurons between layers adjusting weights. Back-links are usually not implemented and special networks were built to take advantage of this particular type of connection: the so called Recurrent Neural Networks (RNNs). They are thought to learn from sequences. The idea is to have loops within the network, as we can see on the left of Fig. 2.5. We can understand how the algorithm works unrolling the loop. Having as input the series of values x_t, the neuron A first accepts the initial condition x_0 to return the output h_0. Also, the neuron has a connection with itself: an internal state is updated. When the second input x_1 is given, A gives the different result h_1 thanks to the evolution. This allows to have in output the evolved series h_t.

RNNs are imagined to implement memory. The hidden state of A is updated during the loop: this is like saying that A has memory of past evolution. This memory approach allows us to train the network unrolling the loop and using the backpropagation algorithm with a cost function.

RNNs are much more complicated than this and a dedicated section should be written concerning this argument, but it is out of the purposes of this thesis. Noteworthy to mention that RNNs are certainly suitable for audio identification and for evolving differential equations in order to find their solution. The interested reader may refer to Lipton et al. (2015) for a review of RNN.

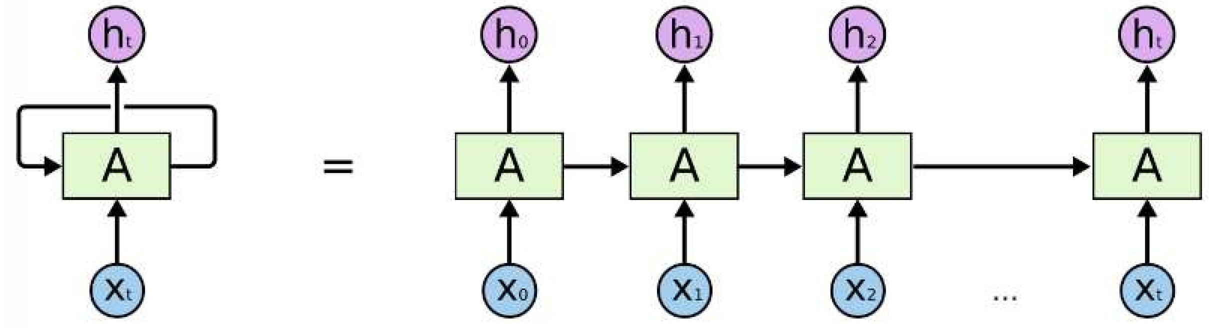


Figure 2.5: *Unrolled recurrent neural network.* (Olah, 2015)

2.2.4 Training neural networks

Like any other ML algorithm, NNs need to be trained. The procedure follows the same locgic as illustrated in sec. 2.1.2 for linear regression. We define a cost function $\mathscr{C}$ to be minimized with respect to the parameters.

More formally, given n data point $(\mathbf{x}_i, y_i)$, the neural network makes a prediction $\hat{y}_i(\boldsymbol{\xi})$, where i is the index of the data point and $\boldsymbol{\xi}$ the parameters of the NNs. Using the mean squared error as in (2.1) as reference for qualifying the NN performance

$$\mathscr{C}(\mathbf{y}, f(\mathbf{x};\boldsymbol{\xi})) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{y}_i(\boldsymbol{\xi})\big)^2\,, \tag{2.29}$$

we solve the following expression

$$\hat{\boldsymbol{\xi}} = \arg\min_{\boldsymbol{\xi}}\big\{\mathscr{C}(\mathbf{y}, f(\mathbf{x};\boldsymbol{\xi}))\big\}\,. \tag{2.30}$$

In the next chapters we will also use the following expression as a cost function, called the mean-absolute error:

$$\mathscr{C}(\mathbf{y}, f(\mathbf{x};\boldsymbol{\xi})) = \frac{1}{n}\sum_{i=1}^{n}\big|y_i - \hat{y}_i(\boldsymbol{\xi})\big|\,. \tag{2.31}$$

The goal is to find the minimum (2.30). Being the parameter number of NN (the $\boldsymbol{\xi}$ dimension) usually huge and the gradient computation could be really expensive. Hence, we have to use a dedicated algorithm for the purpose of deriving the whole network.

The backpropagation algorithm The backpropagation algorithm is a chain rule for partial differentiation. It was thought to deal with the difficulties arising in deriving NN, due to the presence of more than one connection per layer. The connection between the k-th neuron in layer $l-1$ and the j-th neuron in layer l is given by

$$y^l_j = \sigma(z^l_j)\,, \qquad \text{with} \qquad z^l_j = \sigma\left(\sum_k w^l_{jk} y^{l-1}_k + b^l_j\right)\,. \tag{2.32}$$

Let me define the error Δ^l_j of the jth neuron in the lth layer as the variation of the cost function with respect to z^l_j, which means

$$\Delta^l_j = \frac{\partial\mathscr{C}}{\partial z^l_j}\,. \tag{2.33}$$

We can use the chain rule in different ways:

$$\Delta_j^l = \frac{\partial \mathscr{C}}{\partial y_j^l}\sigma'(z_j^l)\,, \tag{2.34}$$

$$\Delta_j^l = \frac{\partial \mathscr{C}}{\partial b_j^l}\,, \tag{2.35}$$

$$\Delta_j^l = \left(\sum_k \Delta_k^{l+1} w_{kj}^{l+1}\right)\sigma'(z_j^l)\,. \tag{2.36}$$

Finally, differentiating the cost function with respect to the weights, we obtain

$$\frac{\partial \mathscr{C}}{\partial w_j^l} = \frac{\partial \mathscr{C}}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_j^l} = \Delta_j^l y_k^{l-1}\,. \tag{2.37}$$

Together (2.34), (2.35), (2.36) and (2.37) define the four backpropagation equations. The algorithm proceeds as follows:

1. *Activation at input layer*: calculate the activation y_j^l of all the neurons in the input layer;
2. *Feedforward*: starting with the first layer, exploit the feedforward architecture through (2.32) to compute z^l and a^l for each subsequent layer;
3. *Error at top layer*: calculate the error of the top layer using (2.34) (this requires to know the expression for the derivative of both $\mathscr{C}$ and σ);
4. *Backpropagation of the error*: use (2.36) to propagate the error backwards and calculate Δ_j^l for all layers;
5. *Calculate gradient*: use (2.35) and (2.37) to calculate $\partial\mathscr{C}/\partial b_j^l$ and $\partial\mathscr{C}/\partial w_j^l$.

This algorithm allows to compute $\mathscr{C}$ derivatives with respect to many parameters in a shorter time than performing the usual numerical derivation. We will refer to this detailed description in order to highlight the differences in the implementation of this algorithm within different environments (sec. 2.3.4).

2.3 Physics-informed neural networks

In Sec. 2.2 I introduced NNs as useful tools for solving physical problems. However, all of these algorithms have been developed for engineering, bioinformatics, and genomics purposes. They are simply applied as-is and tuned to handle the desired physical activities. They are trained on specific datasets and then tested on related others. In this process the internal structure of the ML and NN algorithms does not change at all. Furthermore, studying physical (and especially astrophysical) problems means working with a lack of large amounts of data. ML techniques are instead designed to work with “big data”, being trained with every possible data available online[1]. In physics, this large data set cannot be generated to this level of variety. However, physics has from its side the theoretical and phenomenological laws in the form of partial differential equations. These can be used to augment the information carried by a data set, not necessarily that large. This is the idea behind Raissi et al. (2019)’s pioneering work which introduced Physics Informed Neural Networks (PINN). They are NNs in which the a priori physical information is not given as a bias, but it is implemented directly into the algorithm structure. They are a ML technique born precisely to deal with physical problems.

[1]Think of image generation algorithms or modern text generating artificial intelligences like CHAT-GPT.

PINNs are capable of approximating and solving any physical law starting from input data. The only necessary condition is that the physical principle must be expressed as a partial differential equation. This environment could be useful for the purposes of this thesis because PINNs can infer parameters of the system behind training data starting only from a little ansatz.

In Subsec. 2.3.1, I will introduce the basic mathematical structure on which PINNs are based, while in Subsec. 2.3.2 and in Subsec. 2.3.3 I will follow Raissi et al. (2019) examples explaining PINNs' applicability. In Subsec. 2.3.4 I will introduce the interesting SciML workbench developed by Rackauckas et al. (2021) to handle PINNs. We will see the implementation of the so-called *reverse Automatic Differentiation* (AD), a generalization of the backpropagation algorithm. Subsec. 2.3.4 will also be where I will compare the performance of PINNs in the literature versus usual ML techniques. Finally, in Subsec. 2.3.5 I will present an hybrid approach to PINNs that will be useful in my future work, developed by Nascimento and Viana (2020).

2.3.1 Universal differential equations

PINNs exploit the NNs capability of approximating any possible function. PINNs are based on the (Rackauckas et al., 2021) formalism of Universal Differential Equations (UDEs). A UDE is a differential equation defined in whole or in part by a Universal Approximator (UA), which is a parameterized object capable of representing any possible function in a parameter space limit. A NN is the perfect tool to be considered as UA. The most general UDE is a forced stochastic delay Partial Differential Equation (PDE) defined with the UA embedded. It is given by the expression

$$\mathscr{O}\big[u(t), u(\alpha(t)), W(t), U(u, \beta(t); \boldsymbol{\xi})\big] = 0\,, \tag{2.38}$$

where $\mathscr{O}$ is a generic nonlinear operator, $u(t)$ is the solution of the differential equation, $\alpha(t)$ and $\beta(t)$ are delay functions, $W(t)$ is the Wiener process and U is the UA (the NN) with the parameters $\boldsymbol{\xi}$. A stochastic differential equation is a differential equation whose coefficients are random numbers or random functions of independent variables, while a stochastic delay differential equation is a differential equation where the increment of the random process depends on the past values of the process itself. However, this generic UDE can be specified depending on the problem we want to solve. The simplest form one can use is

$$u' = \mathrm{NN}(u, t; \boldsymbol{\xi})\,, \tag{2.39}$$

a one-dimensional UDE defined completely by a neural network. As we will see in the next sections with some simple examples, this NN can be trained as a data-driven solver of the UDE (forward training) or to learn the structure of the UDE itself (backward training). This means that this UDE can be the basis of every possible implementation of physical laws, requiring these latter only to be expressed as PDEs.

2.3.2 Solving partial differential equations

We will see as PINNs can be established following Raissi et al. (2019) reasoning. Let's express (2.38) as

$$\frac{\partial u}{\partial t} + \mathscr{N}[u] = 0\,, \tag{2.40}$$

where $x \in \Omega \subset \mathbb{R}^d$, $t \in [0, T]$, $\mathscr{N}[\cdot]$ is a nonlinear differential operator and $u(t, x)$ is the hidden solution of the differential equation. We can therefore define the function $f(t, x)$

as the left side of (2.40):

$$f := \frac{\partial u}{\partial t} + \mathscr{N}[u]\,. \tag{2.41}$$

Approximating $u(t,x)$ with a deep NN means constructing a PINN on $f(t,x)$. The two NNs share the same parameters, but with different activation functions due to the presence of $\mathscr{N}[\cdot]$. These parameters can be learned by minimizing the mean squared error cost function

$$\mathscr{C} = \mathscr{C}_u + \mathscr{C}_f\,, \tag{2.42}$$

where

$$\mathscr{C}_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u(t_u^i,x_u^i) - u^i\right|^2, \tag{2.43}$$

$$\mathscr{C}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f(t_f^i,x_f^i)\right|^2. \tag{2.44}$$

In particular, $\mathscr{C}_u$ compares the initial and boundary data, while $\mathscr{C}_f$ applies the structure (2.40). As training data we need both $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ (the initial and limit data for $u(t,x)$) and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ (the placement points for $f(t,x)$). Once $\mathscr{C}$ has been minimized to (2.42), we can learn both $f(t,x)$ and $u(t,x)$.

Note that $\mathscr{N}[\cdot]$ is known and can contain time and space derivatives for $u(t,x)$. Their calculation is necessary to compute $f(t,x)$. Using the PDE structure is the key ingredient to implement the physics on the NN approximating $u(t,x)$. To derive the NNs with respect to their coordinates, in addition to the parameters, we need a more general algorithm than the backpropagation one: Automatic Differentiation (AD; we will see it in Sec. 2.3.4) . This introduces a smoothing mechanism that allows to handle small training datasets with relatively simple feed-forward neural network architectures.

2.3.3 Data-driven discovery of partial differential equations

The most exciting thing PINNs are able to do is approximate the partial differential equations underpinning the training data. The idea is to start from (2.40) in the form

$$\frac{\partial u}{\partial t} + \mathscr{N}[u;\lambda] = 0\,, \tag{2.45}$$

where here $\mathscr{N}[\cdot]$ depends also on some unknown parameters λ. Setting again (2.41) as

$$f := \frac{\partial u}{\partial t} + \mathscr{N}[u;\lambda]\,, \tag{2.46}$$

we approximate $u(t,x)$ as a deep NN. This hypothesis constructs f as PINN, with the same parameters as the u NN plus λ set. Again the aim is to minimize the cost function

$$\mathscr{C} = \frac{1}{N}\sum_{i=1}^{N}\left|u(t^i,x^i) - u^i\right|^2 + \frac{1}{N}\sum_{i=1}^{N}\left|f(t^i,x^i)\right|^2, \tag{2.47}$$

where we are using the training data $\{t^i, x^i, u^i\}_{i=1}^{N}$. By minimizing this $\mathscr{C}$, we can learn both the NNs parameters of u and the λ parameters. The computational approach we can use to have the data-driven learning of a physical law hidden behind the data is to set up a number of different ansatz functions in approximating $\mathscr{N}$, this time unknown, and optimize its respective parameters to have the final form of the equation. We will exploit this wonderful utility in the work of chap. 3.

2.3.4 SciML environment

PINNs are based on the UDE formalism as already explained. We can choose the differential operator $\mathscr{O}$ however we like, creating a huge variety of differential equations. How can we handle this large amount of possibilities? Rackauckas et al. (2021) introduced the SciML ecosystem of macros packages as a set of tools to handle the wide range of possible UDEs, considering all possible cases of adaptivity, rigidity, stochasticity, delays and more. It is coded within the Julia programming language and its main packages can simply be imported from the Pkg terminal. In the next chapters we will mainly use the DifferentialEquations.jl solvers, the DiffEqSensitivity.jl additional methods and the DiffEqFlux.jl helper functions. This environment is really a complete programming tool to handle any kind of possible PINN problem.

Automatic differentiation Training an UDE means minimizing the desired cost function $\mathscr{C}$ with respect to the desired parameters and coordinates. The usual choice for $\mathscr{C}$ is shown in (2.42) and (2.47), with discrete data points as input. Thanks to the chain rule, the calculation of the derivative of this $\mathscr{C}$ is given by the derivative of u and f themselves. The problem of efficiently training a UDE boils down to computing the gradients of the solution of the differential equation u with respect to the desired parameters and coordinates. The issue, as I anticipated, lies in how to compute these derivatives. Since we are dealing with derivatives with respect to the coordinates of the system, in addition to the parameters of the NNs, we need a generalization of the back-propagation algorithm: the Automatic Differentiation (AD; Baydin et al. (2018)), also known as the added derivative.

The idea of AD is to perform a non-standard interpretation of a computer program where this interpretation involves augmenting the standard calculus by computing various derivatives. All numerical calculations are just compositions of elementary operations. Combining the derivative of the constituent operations through the chain rule gives the derivative of the overall composition. In other words, AD in inverse addition mode propagates derivatives backwards from a given output. This consists in the construction of a generic function $f : \mathbb{R}^n \to \mathbb{R}^m$ thanks to intermediate variables v_i such that $v_{i-n} = x_i$ with $i = 1, \dots, n$ are the input variables, v_i with $i = 1, \dots, l$ are the internal working variables and $v_{l-i} = y_{m-i}$ with $i = m-1, \dots, 0$ are the output variables. Each intermediate variable v_i can be completed with an adjoint

$$\overline{v}_i = \frac{\partial y_j}{\partial v_i} \,, \tag{2.48}$$

which represents the sensitivity of an output y_j to variations of v_i. For the back-propagation case $y_j = \mathscr{C}$. The derivative of the generic function f is determined in two steps: a forward step where f is computed to populate the intermediate variables v_i and then a reverse step where the derivative is actually calculated.

Let's take an example by deriving the following $f : \mathbb{R}^2 \to \mathbb{R}$

$$y = f(x_1, x_2) = \ln x_1 + x_1 x_2 - \sin x_2 \,. \tag{2.49}$$

The forward phase is calculated in the left part of the Tab. 2.1 and follows the calculation of the graph in Fig. 2.6. The backward calculus is shown on the right side of Tab. 2.1 and corresponds to each operation on the left side. What we want to calculate as a last result is the influence of the inputs (v_0 and v_{-1}) on the output y. Taking for example v_0 and looking at Fig. 2.6, we can notice that its influence on y occurs only through v_2 and v_3 and therefore

$$\frac{\partial y}{\partial v_0} = \frac{\partial y}{\partial v_2}\frac{\partial v_2}{\partial v_0} + \frac{\partial y}{\partial v_3}\frac{\partial v_3}{\partial v_0} \qquad \Longrightarrow \qquad \overline{v}_0 = \overline{v}_2 \frac{\partial v_2}{\partial v_0} + \overline{v}_3 \frac{\partial v_3}{\partial v_0} \,, \tag{2.50}$$

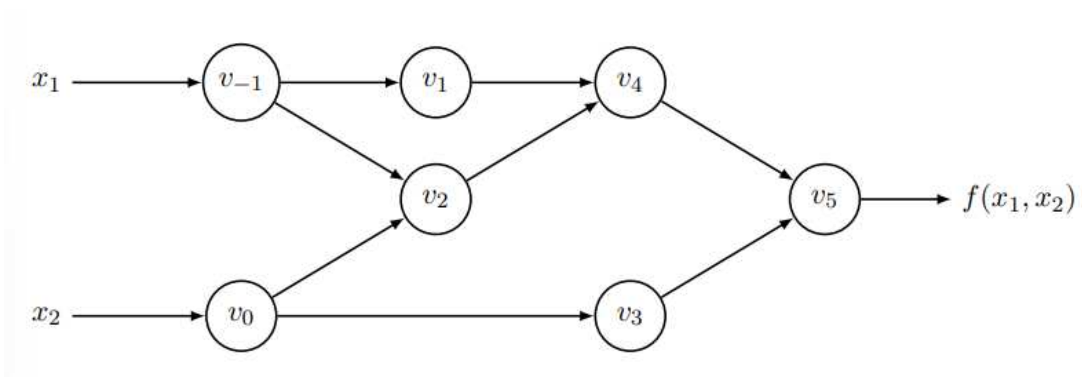


Figure 2.6: *Computational graph of the example $f(x_1, x_2) = \ln x_1 + x_1x_2 - sinx_2$.* (Baydin et al., 2018)

Forward primal trace			*Reverse adjoint trace*			
v_{-1}	$= x_1$	$= 2$	$\overline{x}_1$		$= \overline{v}_{-1}$	$= 5.5$
v_0	$= x_2$	$= 5$	$\overline{x}_2$		$= \overline{v}_0$	$= 1.716$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\overline{v}_{-1}$	$= \overline{v}_{-1} + \overline{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \overline{v}_{-1} + \frac{\overline{v}_1}{v_{-1}}$	$= 5.5$
v_2	$= v_{-1} \cdot v_0$	$= 2 \cdot 5$	$\overline{v}_0$	$= \overline{v}_0 + \overline{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \overline{v}_0 + \overline{v}_2 \cdot v_{-1}$	$= 1.716$
			$\overline{v}_{-1}$	$= \overline{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \overline{v}_0 + \overline{v}_2 \cdot v_0$	$= 5$
v_3	$= \sin v_0$	$= \sin 5$	$\overline{v}_0$	$= \overline{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \overline{v}_3 \cos v_0$	$= -0.284$
v_4	$= v_1 + v_2$	$= 0.693 + 10$	$\overline{v}_2$	$= \overline{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \overline{v}_4$	$= 1$
			$\overline{v}_1$	$= \overline{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \overline{v}_4$	$= 1$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$	$\overline{v}_3$	$= \overline{v}_5 \frac{\partial v_5}{\partial v_3}$	$= -\overline{v}_5$	$= -1$
			$\overline{v}_5$	$= \overline{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \overline{v}_5$	$= 1$
y	$= v_5$	$= 11.652$	$\overline{v}_5$	$= \overline{y}$		$= 1$

Table 2.1: *Automatic differentiation calculation for the example $f(x_1, x_2) = \ln x_1 + x_1x_2 - sinx_2$.* (Baydin et al., 2018)

as we can see on the right-hand side of Tab. 2.1 in the two steps

$$\overline{v}_0 = \overline{v}_3 \frac{\partial v_3}{\partial v_0}\,, \tag{2.51}$$

$$\overline{v}_0 = \overline{v}_0 + \overline{v}_2 \frac{\partial v_2}{\partial v_0}\,. \tag{2.52}$$

This calculation takes advantage on the fact that the derivative's computation is significantly less expensive to evaluate than both the normal numerical implementation and the backpropagation algorithm. In the extreme case of a function $f : \mathbb{R}^n \to \mathbb{R}$, a single computation is sufficient to determine the entire gradient ∇f, given by

$$\nabla f = \left(\frac{\partial y}{\partial x_1}, \ldots, \frac{\partial y}{\partial x_n} \right)\,, \tag{2.53}$$

with a significantly reduced computational cost.

SciML performances Thanks to the implemented UDE formalism and the bunch of packages developed just for this environment, SciML is able to handle every kind of possible differential equation that can be constructed from (2.38). As we can see in Tab. 2.2, it can support stiff ordinary differential equations (ODE), differential algebraic equations (DAE), stochastic differential equations (SDE) and delay differential equations (DDE). Furthermore, it is compatible with most of the modern high computing hardwares

Feature	*Stiff*	*DAEs*	*SDEs*	*DDEs*	*Stabilized*	*DtO*	*GPU*	*Dist*	*MT*
SciML	✓	✓	✓	✓	✓	✓	✓	✓	✓
TORCHDIFFEQ	-	-	-	-	-	✓	✓	✓	✓
TORCHSDE	-	-	✓	-	-	✓	✓	✓	✓
TFDIFFEQ	-	-	-	-	-	-	✓	-	-

Table 2.2: *Feature comparison between different ML-augmented differential equation libraries. Columns correspond to a support for stiff ordinary differential equations, differential algebraic equations, stochastic differential equations, delayed differential equations, stabilized adjoints, discrete-then-optimize methods, compatibility with graphic processing unit (parallelization), compatibility with distributed computing and compatibility with multi-threading.* (Rackauckas et al., 2021)

# of ODEs	3	28	768	3072	12288	49152	196608	786432
SciML	1×	1×	1×	1×	1×	1×	1×	1×
TORCHDIFFEQ	4900×	1900×	840×	280×	82×	31×	24×	7×

Table 2.3: *Relative time to solve ordinary differential equations for ML-augmented differential equation libraries (smaller is better).* (Rackauckas et al., 2021)

available. On the other hand, thanks mainly to AD, SciML has the ability to solve a large system of stiff ODEs much faster than the usual solvers available in PyTorch and in TensorFlow, which have approximately the same speed (±2%; Abadi et al. (2016)).

2.3.5 Hybrid physics-informed neural networks

Alongside regular PINNs, Nascimento and Viana (2020) developed a hybrid version of them. As we saw, Raissi et al. (2019)'s PINNs are used to solve and reconstruct physical laws thanks to the use of a cost function. No limit on the order of the PDE is anyway set (it depends on the particular case of study that one wants to handle). If instead one does not want to learn the structure of the equation, but wants only to solve a reduced-order known differential equation, it can be done thanks to hybrid PINNs. In particular, the authors chose to use a RNN implemented directly inside the integration algorithm. Having a training dataset for the solution (or for a related quantity) can lead to an inference of the differential equation's parameters. Nascimento and Viana (2020) applied this idea to improve the accuracy of cumulative damage models used to predict wind turbine main bearing fatigue, solving in a fast way a first order differential equation.

This hybrid PINN idea will be the base for the final result of this thesis, and will be presented more in detail in Sec. 4.4.

2.4 Gravitational-wave data analysis with neural networks

As we have seen, GWs were first detected in 2015 (Abbott et al., 2016b). On the other hand, ML has emerged in the last decade. Thus, their synergy is quite an open task. Currently many efforts are underway to implement ML and NN in GW data analysis for different purposes. In this section I will summarize them briefly, in order to introduce the background behind the next few chapters. For more details, see Cuoco et al. (2020) and the constantly updated web page Wang (2019).

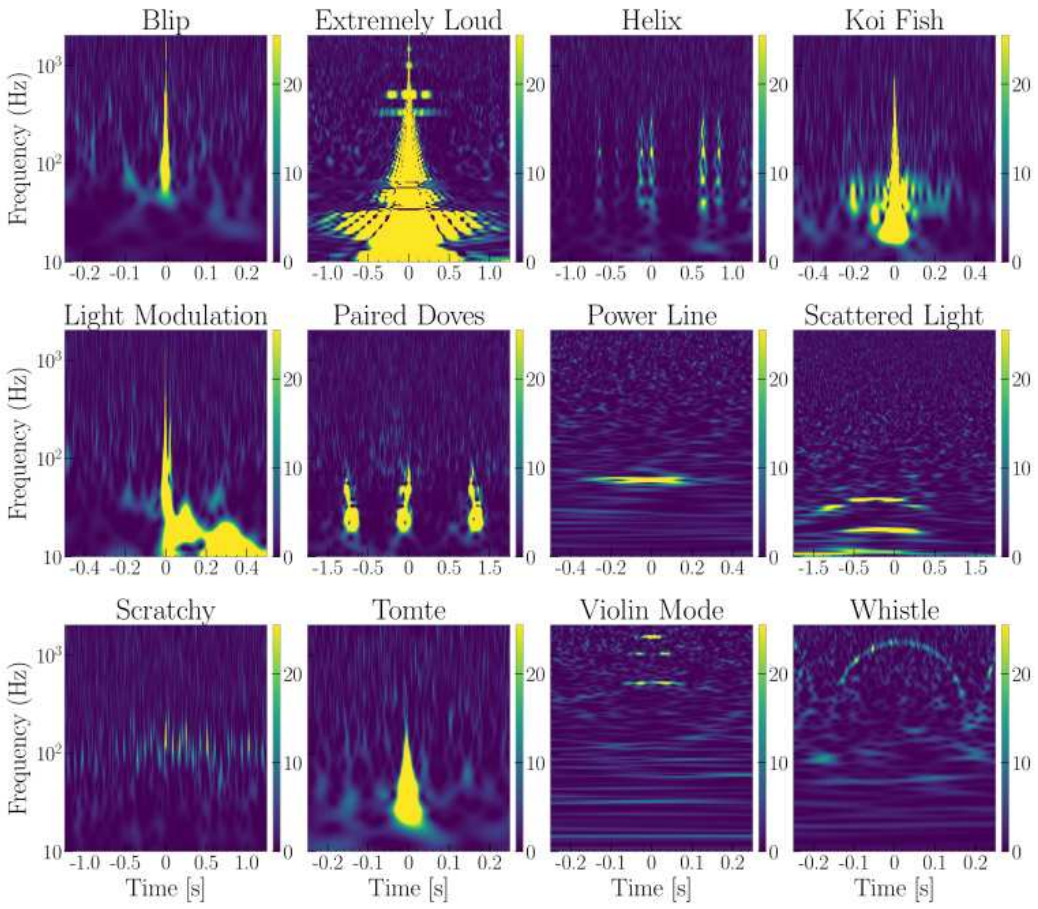


Figure 2.7: *Examples of glitches in the time-frequency spectrogram.* (Cuoco et al., 2020)

2.4.1 Data quality

As we saw in Sec. 1.5.2, GW signals are buried within the detector noise of the interferometer. The noise usually is stochastic, but it can also appear as an unwanted deterministic signal, called a glitch. In Fig. 2.7 we can see several examples of glitch classes. Since they always have the same characteristic pattern in the time-frequency spectrogram, Convolutional Neural Networks (CNNs) could be successfully used for glitch classification and simulation (Razzano and Cuoco, 2018), thanks to their optimal behavior with image recognition. We will go deeper in the argument in Sec. 3.4.2

ML could also be useful for reconstructing nonlinear noise sources. Even with the most powerful filtering, a significant amount of noise pollutes the interferometer output. Using the ability of NNs to reconstruct non-linear functions, they can have as input environmental and control data streams to reconstruct the transfer function for these noise sources. Once trained in this way, they can be used to subtract new nonlinear couplings from the output data (see e.g. Bacon et al. (2023) and reference therein).

2.4.2 GW signal's modelling

As we saw in Sec. 1.5, GW searches from CBC relying on matched filtering and Bayesian parameter inferences require accurate waveform models. They are usually produced through NR simulations or solving the Effective One Body (EOB) problem (sec. 3.1.1). In particular, these EOB models first find the orbital dynamics by solving a complex system of ordinary differential equations (known only from the precise configuration) and then obtain the waveform as a second step.

2.4.3 Sensitivity improvements

The searches for the signal are divided into four branches: CBCs, bursts, continuous waves and gravitational wave background. In each of these researches, ML is increasing its presence with already tested or developed algorithms. In this paragraph I will talk briefly about each of them.

As far as CBCs are concerned, decision trees were tested as an alternative detection method alongside matched filtering, showing good improvements (Kapadia et al., 2017). Bursts searches are instead focusing their ML efforts on Core Collapse SuperNovae (CCSN) searches, primarily via CNNs and CCSN simulations for a mock training signal, so far without success. Concerning continues waves, the difficulty for ther searches is based on the huge computational cost. Machine learning certainly holds promise in two ways: a direct replacement of parts of the analysis pipeline (Morawski et al., 2020) and a comprehensive analysis of strain data (Dreissigacker et al., 2019). Furthermore, for a non-Gaussian GW background, nested sampling is usually implemented. ML techniques could certainly improve their speed and accuracy (Wysocki et al., 2019).

2.4.4 Parameter inference

As we saw in Sec. 1.5.4, parameter inference for CBCs relies on long Bayesian techniques that compare an underlined model to the actual signal. The main problem with these methods is the computational time required (almost six months). Some efforts to speed up these estimation thanks to ML have been implemented, such as random forest regressions to approximate the likelihoods. Bayesian neural networks (neural networks based on probability statements) were also tested. The results are certainly promising.

On the other hand, a quick estimate of the parameters for low latency inference is needed to decide whether to send an alert for EM counterpart lookup. Some ML techniques are now implemented on matched filter pipelines, but as already stated, this technique has strong physical background behind it (Gabbard et al., 2018).

As a concluding remark to this chapter, I resume the basic utilities for understanding the work of the next few chapters. We want to build an algorithm for fast parameter inference from simple waveform measurements, with as little physical a priori information as possible. PINNs are definitely the best tool to handle this, mostly due to their ability to have data-driven discovery of differential equations from a simple initial ansatz. In the next chapter I will present the work done by Keith et al. (2021a). They were able to reconstruct with high precision the dynamics of binary black holes only from (mock) waveform measurements and an initial Newtonian ansatz.

Chapter 3

Learning binary black hole's dynamics from gravitational wave data

The main goal of this thesis is to build a fast algorithm for parameter estimation and we want it to be as more data-driven as possible. This aspect will be extremely useful especially for GW burst searches, such as for example the cWB pipeline now operational in modern interferometric detectors. This environment aims to detect new GW signals basing its investigation on as little previous physical information as possible (see Sec. 1.5.3). The starting approach is to switch from the time-domain output of the interferometer to the time-frequency domain, as we saw in panel (b) of Fig. 1.10. cWB looks for pixels in this spectrogram with an energy density above a certain threshold. If several groups of highlighted pixels are coherent between two or more interferometers, a candidate event is triggered and a waveform reconstruction is performed. Thanks to this highlighting of only relevant pixels, cWB reconstructs a fairly sharp waveform, with little random noise. This method should also allow to detect other GW signals different from CBCs, for example CCSNs signals. However, regarding CBCs, no parameter estimation is implemented. Currently only phenomenological parameters can be deduced. In other words, we have a fantastic detection tool for GW signals, but being as uninformed as possible about preemptive physics permits to easily detect new signals, but without inferring so much about the source.

From the cWB pipeline we have as output a clear reconstructed waveform. Using this waveform (its individual data points) as a training dataset for a PINN is the leading idea of this work. As we saw in Sec. 2.3.3, a PINN is capable of reconstructing the basic physical differential equations behind a data set, providing to it only a small initial ansatz. The algorithm is also able to infer parameters of the system. This useful ability of a NN to approximate every possible function is exactly what we are looking for.

This thesis is inspired by the work done by Keith et al. (2021a). As we will see in this chapter, they were able to reconstruct the orbital dynamics of spiral Binary Black Holes (BBHs) by creating a system of four UDEs to be learned from a PINN. Simulated waveform measurements were used as a training dataset, minimizing the difference between the expected and measured waveforms. We will see that they were able to reconstruct the orbits with a level of NR precision, starting only from a Newtonian ansatz for the governing equations. The algorithm was implemented using the SciML workbench in the Julia programming language and is available at Keith et al. (2021b).

This chapter is organized as follows. In Sec. 3.1 I will introduce the physics to be studied and explain how the PINN algorithm is set up following it. In Sec. 3.2 I will show their main results, extending them in Sec. 3.4. Studying and reproducing this algorithm in detail, I came across some hidden parametric constraints that were greatly reducing the space of the parameters considered: they were working with non-physical values. In Sec. 3.5 I will report this problem, while in chap. 4, I will describe my work to overcome it and handle a more phenomenological parameter space.

Through the chapter I will follow Keith et al. (2021a) geometric units where both the speed of light c and the gravitational constant G are set to unity.

3.1 Binary black hole modelling

Keith et al. (2021a)'s main goal was to reconstruct the orbital dynamics of a coalescent BBH system from waveform measurements alone. The idea is to construct a set of UDEs as a system of PDEs governing the dynamics of the spiral system, with PINNs as approximators of part of the equations. These UDEs will be trained thanks to a mean squared error between the measured and the theoretical waveform (calculated using the UDE system).

In this section I will talk about the configuration of the problem. Briefly, we will see the basic physical assumptions behind the algorithm in Sec. 3.1.1. In Sec. 3.1.2 I will illustrate how a UDE system for orbital reconstruction is set up to be trained, while in Sec. 3.1.3 we will see how GW physics is implemented to train the UDE system.

3.1.1 Effective one body problem

The problem of finding the trajectories $\mathbf{r}_1(t)$ and $\mathbf{r}_2(t)$ of two objects of mass m_1 and m_2 orbiting each other is the well known two-body problem, studied since the birth of Newtonian physics. Kepler (1609)'s three laws are the well-known solutions to this problem in the context of Newtonian gravity. With Einstein (1916)'s GR, this problem has been revisited. As we saw in Sec. 1.1, solving Einstein's equations (1.3) in their complete form is an open task. Different ways of proceeding are used: NR techniques, which use numerical relativistic methods to find a complete resolution for specific cases, and PN approximations, which add relativistic terms such as v/c orders to the Keplerian equations.

In Keith et al. (2021a)'s case, they considered slow-moving objects ($v \ll c$). In this case, the PN formalism provides the justification for treating the two-body problem as a real one-body problem. The "effective" body rotates around the center of mass (CM) of the system at a distance $\mathbf{r}$ and with a mass M, respectively given by

$$\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2 \,, \tag{3.1}$$

$$M = m_1 + m_2 \,, \tag{3.2}$$

where $\mathbf{r}_i$ and m_i are the position and mass of the ith real object. This object is orbiting around a fixed central object with spherical symmetry. From the definition for the position of the CM,

$$\mathbf{r}_{CM} = \frac{m_1\mathbf{r}_1 + m_2\mathbf{r}_2}{M} \,, \tag{3.3}$$

we can reconstruct the two-body motion from the effective one-body using the relations

$$\mathbf{r}_1 = \frac{m_2}{M}\mathbf{r} \,, \tag{3.4}$$

$$\mathbf{r}_2 = -\frac{m_1}{M}\mathbf{r} \,. \tag{3.5}$$

3.1.2 Universal differential equations for binary black holes' dynamics

The idea is to build a UDE system of the type

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}, \mathscr{F}(\mathbf{u}; \boldsymbol{\xi}))\,, \qquad \text{with} \qquad \mathbf{u}(0) = \mathbf{u}_0\,, \tag{3.6}$$

where $\mathbf{u}(t)$ is the solution vector and $\mathbf{u}_0$ the initial conditions. Looking at the generic definition of UDE (2.38), $\mathscr{F}(\mathbf{u}; \boldsymbol{\xi}))$ is the NN, which approximates a part or the whole function $\mathbf{f}$. Remember that a feed forward NN with 3 layers in total is given by (2.28). The PN approximation, through the effective one-body problem, allows to develop a physical framework to build a system like (3.6).

The authors assumed that the trajectory of the real object around the fixed central one with spherical symmetry lies in the equatorial plane. Taking this plane perpendicular to a z axis, one can define the orbital phase angle ϕ as the angle between $\mathbf{r}$ and a reference axis $\hat{\mathbf{x}}$. Other parameters of the orbits are the anomaly χ, the semilatus rectus p and the eccentricity e. In the Newtonian case the last two parameters are constant, while for the PN approximation they depend on time for a reason that we will see shortly. Using the following parametrization for the Euclidean norm of $\mathbf{r}$, one gets

$$r(t) = \frac{M\,p(t)}{1 + e(t)\cos\chi(t)}\,. \tag{3.7}$$

Expressing (3.4) and (3.5) as

$$\mathbf{r}_1(t) = \frac{m_2}{M} r(t)\left(\cos\phi(t), \sin\phi(t), 0\right) \tag{3.8}$$

$$\mathbf{r}_2(t) = -\frac{m_1}{M} r(t)\left(\cos\phi(t), \sin\phi(t), 0\right), \tag{3.9}$$

and substituting (3.7) into them, one has

$$\mathbf{r}_1(t) = \frac{m_2\,p(t)}{1 + e(t)\cos\chi(t)}\left(\cos\phi(t), \sin\phi(t), 0\right), \tag{3.10}$$

$$\mathbf{r}_2(t) = -\frac{m_1\,p(t)}{1 + e(t)\cos\chi(t)}\left(\cos\phi(t), \sin\phi(t), 0\right). \tag{3.11}$$

In this way Keith et al. (2021a) have a parametrization of the two BH trajectories, whose evolution is governed by $\mathbf{u} = (\phi, \chi, p, e)$. They proposed the following set of UDEs to describe the dynamics:

$$\begin{cases} \dot{\phi} = \frac{(1+e\cos\chi)^2}{Mp^{3/2}}\left(1 + \mathscr{F}_1(\cos\chi, p, e)\right) \\ \dot{\chi} = \frac{(1+e\cos\chi)^2}{Mp^{3/2}}\left(1 + \mathscr{F}_2(\cos\chi, p, e)\right) \\ \dot{p} = \mathscr{F}_3(p, e) \\ \dot{e} = \mathscr{F}_4(p, e) \end{cases} \qquad \text{with} \qquad \mathbf{u}(0) = \begin{pmatrix} \phi_0 \\ \chi_0 \\ p_0 \\ e_0 \end{pmatrix}. \tag{3.12}$$

Here we see a set of four UDEs of the form (3.6), where $\mathscr{F}_i$ are the NNs to train. Note that these expressions are rotationally invariant because their right-hand side does not depend on ϕ. Furthermore, we can see that, for $\mathscr{F}_3 = \mathscr{F}_4 = 0$, the orbital energy E and the orbital angular momentum L are conserved. However, since GW has back-reaction on the system, we need to consider always $\dot{E}, \dot{L} < 0$.

These UDEs are not randomly set, but are thought to recover Newtonian orbits for $\mathscr{F}_i = 0$. With this structure, Keith et al. (2021a) can learn new interaction terms for

UDEs, working exactly as with the PN approach. The 0th order PN approximation and quadrupole formula, that I will discuss in Sec. 3.1.3, should neglect a large amount of the physics that the $\mathscr{F}_i$ NNs are arranged to recover. THe authors start from a simple Newtonian ansatz and ask our PINNs to learn more complex physics from experimental data.

3.1.3 Training the algorithm: gravitational wave implementation

These UDEs need to be trained to be useful. As we have seen in Sec. 1.4, the only measurement we have available is the strain amplitude $h(t)$ of a GW emitted by the system. We can then set up a cost function $\mathscr{C}$ which depends on the difference between this measured data and the data predicted by the UDE evolution.

More formally, we can take a different representation of the quadrupole mode (1.29) of a GW. Very far from the source, the gravitational radiation field is an outgoing spherical wavefront. We can therefore expand the far-field radiation thanks to the spherical tensor harmonics Y_{lm} with respective harmonic indices (l, m) as

$$h(t,\theta,\varphi) = h_+(t,\theta,\varphi) - ih_\times(t,\theta,\varphi) = \sum_{l=2}^{\infty}\sum_{m=-l}^{l} h^{lm}(t)\,{}_{-2}Y_{lm}(\theta,\varphi)\,, \tag{3.13}$$

where (θ, φ) are the polar and azimuthal angles, h_+ and $h_\times$ are respectevely the real and imaginary parts of the complex field h, and $h^{lm}(t)$ are harmonic coefficients. We can focus our interest on the main quadrupole component $(2, 2)$ of the waveform, defined as

$$h^{22}(t) = \frac{1}{r}\sqrt{\frac{4\pi}{5}}\left(\ddot{M}_{xx} - 2i\ddot{M}_{xy} - \ddot{M}_{yy}\right). \tag{3.14}$$

This definition is similar to the expression (1.32). From the last formula, we can isolate the polarization h_+ as

$$h_+(t) = \frac{r}{M}\mathrm{Re}\{h^{22}(t)\}\,. \tag{3.15}$$

Using the M_{ij} definition (1.30), in this case we have

$$M_{xx}(t) = m_1x_1^2(t) + m_2x_2^2(t)\,, \tag{3.16}$$
$$M_{yy}(t) = m_1y_1^2(t) + m_2y_2^2(t)\,, \tag{3.17}$$
$$M_{xy}(t) = m_1x_1(t)y_1(t) + m_2x_2(t)y_2(t)\,, \tag{3.18}$$

where the index refers to the corresponding BH. By substituting (3.16 - 3.18) in (3.14) and then the latter in (3.15), we can see that the time evolution of h_+ depends on the positions of BHs $\mathbf{r}_1 = (x_1, y_1, z_1)$ and $\mathbf{r}_2 = (x_2, y_2, z_2)$ as defined in (3.8) and (3.9). Thus, h_+ depends on time through the evolution of $\mathbf{u} = (\phi, \chi, p, e)$ and so on the PINN $\mathscr{F}_i$ within it.

This is the basis for constructing the cost function $\mathscr{C}$. Having our waveform data measurements as ordered pairs $(t_k, h_{+,k})$, Keith et al. (2021a) defined $\mathscr{C}$ as

$$\mathscr{C}(\mathbf{u}) = \langle J(\mathbf{u},\cdot)\rangle := \frac{1}{T}\int_0^T J(\mathbf{u},t)\,dt\,, \tag{3.19}$$

with

$$J(\mathbf{u},t) = \sum_k (h_{+,k} - h_+(t))^2\,\delta(t-t_k)\,, \tag{3.20}$$

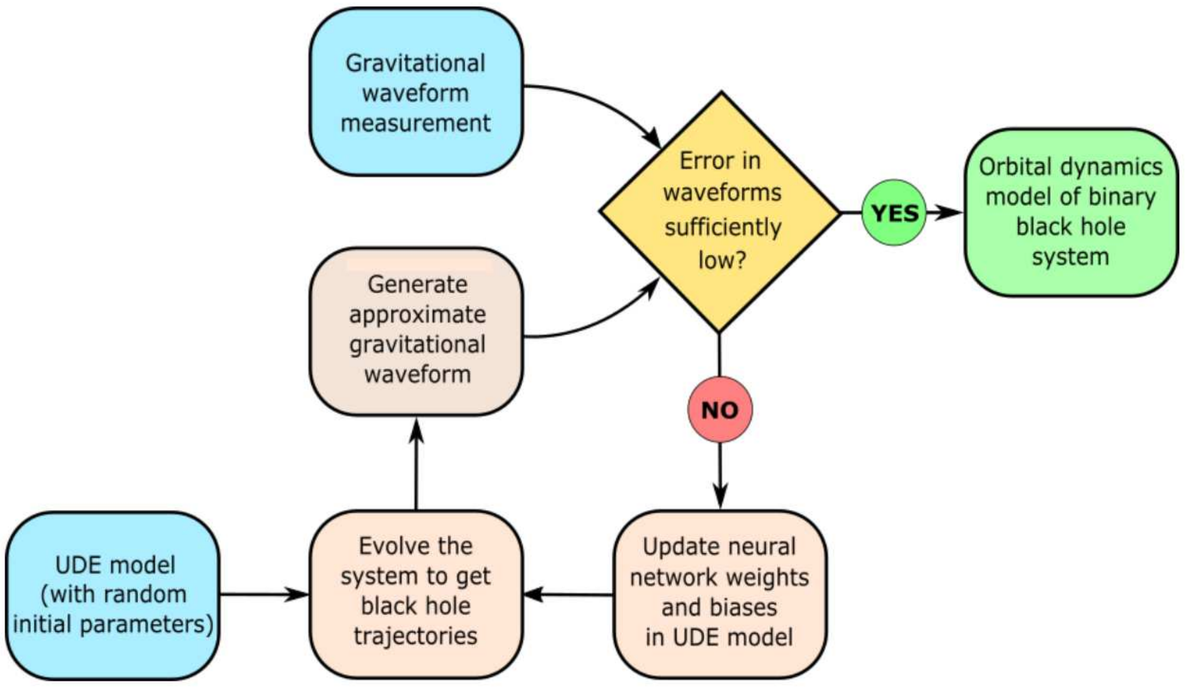

Gravitational waveform measurement
Error in waveforms sufficiently low?
YES
Orbital dynamics model of binary black hole system
Generate approximate gravitational waveform
NO
UDE model (with random initial parameters)
Evolve the system to get black hole trajectories
Update neural network weights and biases in UDE model

Figure 3.1: *Keith et al. (2021a)'s algorithm flowchart.*

where $t_k \in [0, T]$ and $\langle \cdot \rangle$ represent an average over time. Here $h_+(t)$ is the one defined in (3.15) and can be modified to minimize $\mathscr{C}$ varying the parameters of $\mathscr{F}_i$. What we want to solve is the following optimization problem

$$\hat{\boldsymbol{\xi}} = \arg\min_{\boldsymbol{\xi}} \{\mathscr{C}(\mathbf{u})\} \,. \tag{3.21}$$

The structure of the algorithm is summarized by the flowchart in Fig. 3.1. We start with the UDE model (3.12) with random $\boldsymbol{\xi}$ parameters (fluctuating around $\boldsymbol{\xi} = \mathbf{0}$ with a Gaussian behaviour with variance 10^{-5}). This beginning seems obvious, but once implemented it results in an actual coding step where we are building the ODE. We solve this initial system to obtain the solution $\mathbf{u} = (\phi, \chi, p, e)$. This step is performed thanks to a Runge-Kutta method at order 4^{th}, a classic ODE solver that we will describe in Sec. 4.4. Once we have found this solution $\mathbf{u}$, we can calculate our theoretical $h_+(t)$ thanks to (3.10), (3.11) and (3.14 - 3.18). This $h_+(t)$ is then compared with the GW measurements calculating the in-sampling error $E_{in} = \mathscr{C}$ from (3.19). If this E_{in} is low enough, we can stop training and define the current set of $\boldsymbol{\xi}$ as the optimal value $\hat{\boldsymbol{\xi}}$. If instead the error is still too big, we have to update $\boldsymbol{\xi}$ to better values to minimize E_{in}. For this $\boldsymbol{\xi}$ optimization, Keith et al. (2021a) has chosen to use the BFGS algorithm (see Sec. 2.1.3). Once the error is low enough, or after a certain number of iterations (of epochs), we can stop training and find $\hat{\boldsymbol{\xi}}$ to get the best approximation for the whole UDE system (3.12) behind the measured data.

3.2 Algorithm implementation

In their algorithm, Keith et al. (2021a) considered numerically generated waveforms from an eccentric BBH system of equal masses. Noteworthy is the fact that here the authors optimized also two parameters of the system, alongside the $\boldsymbol{\xi}$ ones. This will surely be useful for the work of this thesis: here we can learn how to infer the parameters of the source. The authors have set $m_1 = m_2 = 0.5$ (without units of measure), with the possibility of changing the masses in the code, but with the constraint that the sum of the two must be $M = 1$. As we will see, this severely limits the parameter space.

The algorithm at which I'm referring can be found as SXS2.jl in Keith et al. (2021b)

Initial conditions The only fixed initial condition is $\phi_0 = 0$. e_0 and χ_0 are treated as parameters to be learned, exactly like $\boldsymbol{\xi}$. The initial guesses are $e_0 = 0.085$ and $\chi_0 = \pi$. For what concerns p_0, its initial condition is also free to vary, but assuming that r_0 is known, it depends on the values of e_0 and χ_0 thanks to

$$p_0 = \frac{r_0}{M}(1 + e_0 \cos \chi_0)\,, \tag{3.22}$$

derived from Eq. (3.7).

Training dataset The training data set consists of 10^3 data points $(t_k, h_{+,k})$. The waveform is collected from an NR database. The simulation was performed using the Spectral Einstein Code (SpEC) developed by the Simulating eXtreme Spacetimes (SXS) collaboration (the SXS collaboration, 2000) and made publicly available through the Gravitational Waveform Database (GWD; Boyle et al. (2019), the SXS collaboration (2019)). The collected waveform is imported from an input file that contains 20350 points with $t \in [-107.8, 6951.7]$s, separated by $\Delta t \sim 0.1$s. This is clipped to the range $t \in [5263.7, 6769.0]$s, where the most recent data occurs just before the merge. The interval is then set as $t \in [0, T]$ and the data is resampled to have 10^3 data points, separated in time by $t = 1.50$ s. We can see this collected and resampled data in panel (a) of Fig. 3.2. The amplitude modulations are due to the fact that with a high eccentricity the objects are faster when closer and slower when farther.

Neural network architecture Following (3.12), Keith et al. (2021a) have implemented two NNs: one for $\mathscr{F}_1$ and $\mathscr{F}_2$ and one for $\mathscr{F}_3$ and $\mathscr{F}_4$. Each of them has a fully connected hidden layer with 32 neurons and two outputs. Following (2.28), the authors set each activation function σ_j as tanh in the middle layers, while in the input layer they insert 9 different neurons, each with its own activation function.

Regularization strategies To regularize the algorithm, Keith et al. (2021a) changed the cost function (3.19) to

$$\mathscr{C}(\mathbf{u}) = \langle J(\mathbf{u}, \cdot)\rangle + \mathscr{P}_1(\mathbf{u}) + \mathscr{P}_1(\mathbf{u}) + \mathscr{R}(\boldsymbol{\xi})\,, \tag{3.23}$$

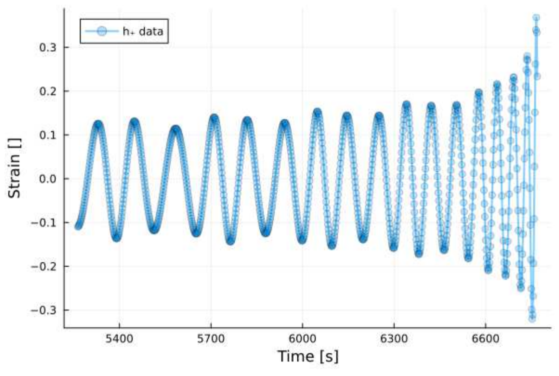
where

$$\mathscr{P}_1(\mathbf{u}) = \gamma_1\big\langle(\dot{p})_+^2\big\rangle + \gamma_2\big\langle(\ddot{p})_+^2\big\rangle\,, \tag{3.24}$$
$$\mathscr{P}_2(\mathbf{u}) = \gamma_3\big\langle(-e)_+^2\big\rangle + \gamma_4\big\langle(e - e_0)_+^2\mathbf{1}_{\{p>6+2e_0\}}\big\rangle\,, \tag{3.25}$$
$$\mathscr{R}(\boldsymbol{\xi}) = \gamma_5||\boldsymbol{\xi}||^2 \tag{3.26}$$

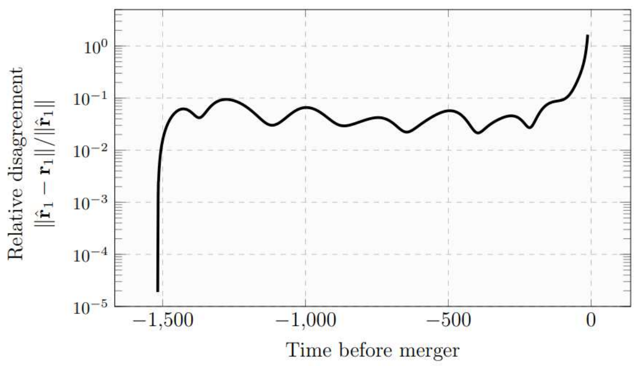
and

$$(f(t))_+ = \max\{0, f(t)\}\,, \tag{3.27}$$

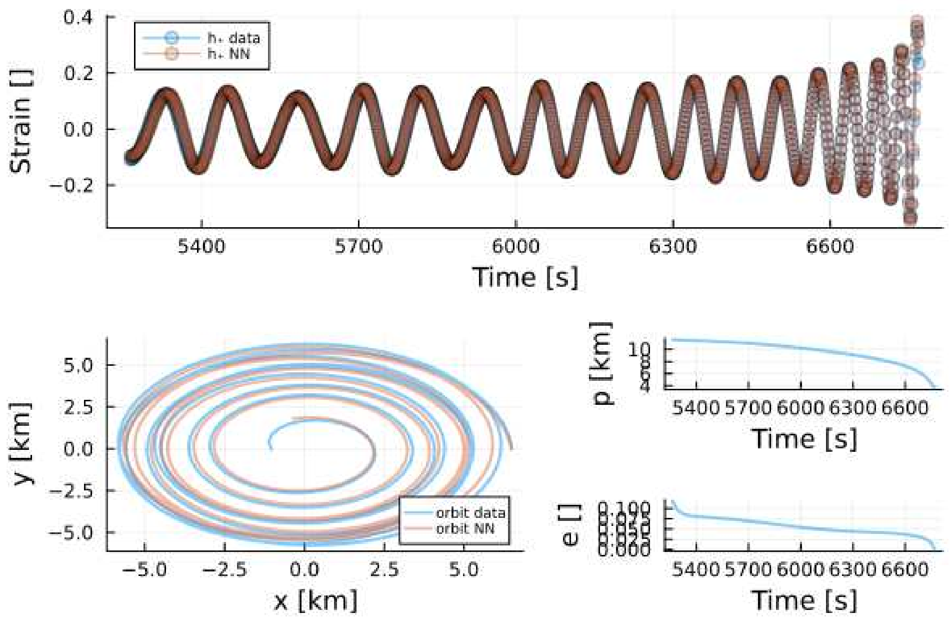
while $\mathbf{1}_\Omega$ denotes the indicator function on the set $\Omega \subset [0, T]$. All these penalty and regularization terms to help the algorithm to avoid non-physical local minima. In order, (3.24) establishes a $p(t)$ (and so a $r(t)$) converging to 0 at an increasing rate over time, (3.25) encourages the selection of $e(t) > 0$ and the last term constrains the stability of the system by imposing $p \geq 6 + 2e$, which must be true for constrained orbits. Since the



(a)



(b)



(c)

Figure 3.2: SXS2.jl *specifics.* (a) *Starting waveform: "h_+ data" is the training dataset. Results after the training:* (b) *relative disagreement between the NR waveform and the learned one* (Keith et al., 2021a) *and* (c) *results for orbits and waveform reconstruction. p and e as a function of time are also plotted.*

real minimum of $\mathscr{C}$ has all these conditions satisfied, they act as a guardrail throughout the optimization process.

On the other hand, imposing $t = 0$ at the beginning of the time series data and $t = T$ at the end of it, the training is done on a sequence of increasing time intervals $[0, T_0] \subsetneq [0, T_1] \subsetneq \cdots \subsetneq [0, T]$. The values of T_i start from the 30% of T and proceed with 10% steps up to 99% and 100%. The optimized parameters $\boldsymbol{\xi}$ which exit as output of the training on the first interval are provided as an initial guess for parameters of subsequent interval. The objective is to train the algorithm on a shorter time interval to more easily reach the real minimum and avoid local minima.

3.3 Results

The final results are shown in panel (c) of Fig. 3.2. In the top plot we can see the excellent agreement that the learned waveform has with the data. In the lower left panel

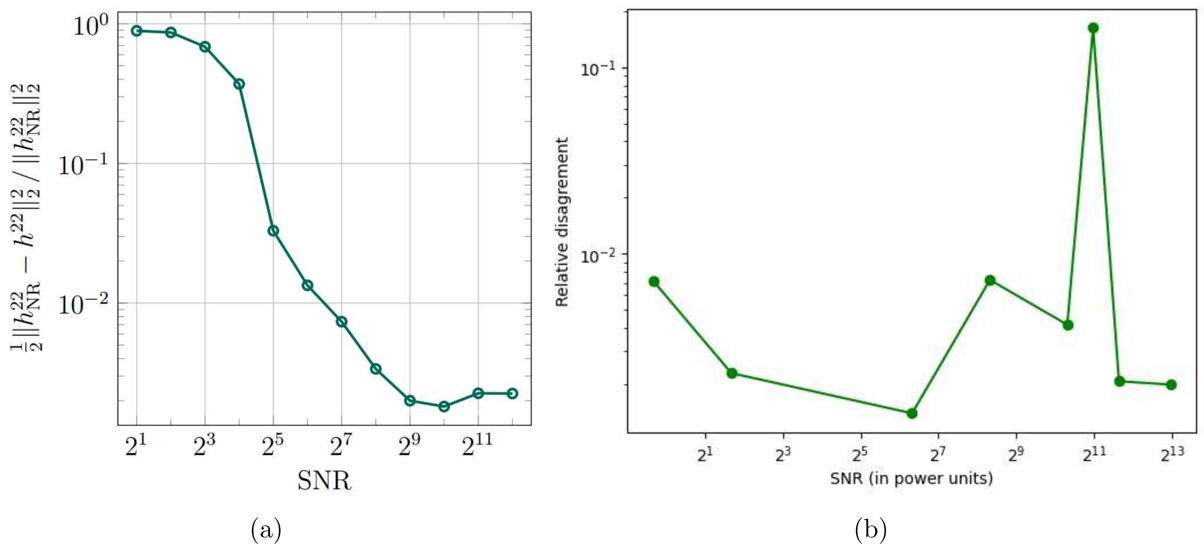


Figure 3.3: (a) *Keith et al. (2021a)'s and* (b) *my plot of the relative disagreement between the NR waveform and the learned one as a function of the SNR.*

we can see the orbit of one of the two objects. Thanks also to the p and e plots at the bottom right of the panel, we can see that the algorithm reproduces both the spiral and the plunge phase very well. p decreases faster and faster, while e decreases as predicted by Peters (1964). In this case, no plunge phase is seen for e. The actual performance of the algorithm can be seen in panel (b) of the same figure. Here the relative disagreement between the NR orbit and the learned one

$$\text{Relative disagreement} = \frac{||\mathbf{r}_{NR} - \mathbf{r}_{GW}||}{||\mathbf{r}_{NR}||} \tag{3.28}$$

is plotted against time. However, this cannot be considered as a relative error because the two are calculated in two different gauges.

3.4 Noise implementation

3.4.1 Gaussian noise

For the latter example, it was necessary to investigate how well the machine is able to learn the parameters. Keith et al. (2021a) tested their parameter inference performance by learning $\boldsymbol{\xi}$, e_0 and χ_0 from the same waveform they analyzed in the SXS2.jl algorithm, but adding several levels of zero-mean Gaussian white noise. Following the formalism (1.62) we can express the output $o(t)$ of an instrument as

$$o(t) = s(t) + n(t)\,, \tag{3.29}$$

where $s(t)$ is the signal (h_+ in our case) and $n(t)$ the noise. The values inferred from the parameter output are then fed as initial guesses to the same algorithm which fits the same waveform with the same noise level, but all at once. After performing the second fit, they plotted the relative disagreement between the NR waveform data and the learned waveform ($||\cdot||_2 = \sqrt{(\cdot)^2}$)

$$\text{Relative disagreement} = \frac{1}{2}\frac{||h_{+,NR} - h_{+,NN}||_2^2}{||h_{+,NR}||_2^2} \tag{3.30}$$

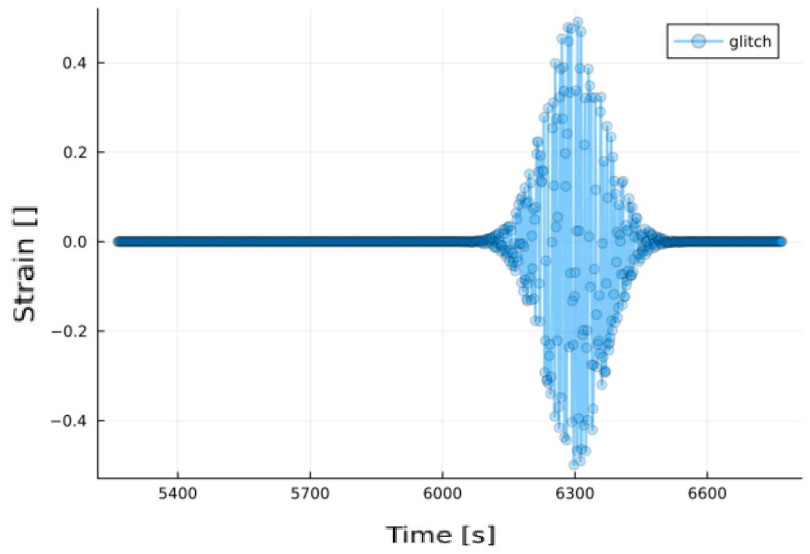


Figure 3.4: *Whistle glitch in time domain.*

with respect to the SNR, as can be seen in panel (a) of Fig. 3.3. In this case the SNR is in power units, defined as

$$\mathrm{SNR} = \frac{\mathcal{P}(s(t))}{\mathcal{P}(n(t))} = \frac{||s(t)||_2^2}{N}\frac{N}{||n(t)||_2^2} = \frac{||s(t)||_2^2}{||n(t)||_2^2}\,, \tag{3.31}$$

where $\mathcal{P}(\cdot)$ is the power of the argument and N is the number of data points. In panel (b) there is the same plot I reproduced. In the Keith et al. (2021a) algorithm, the performance is good up to SNR$\sim$32. As the authors said, this is a critical result, because only one GW event has such a high SNR in the history of observations. My plot represents only a quick estimate of the relative disagreement and it is based only on one single run per point, not considering the second run with learned guesses in input. For this reason, no error bars was set on the points and my results for this plot are less informative than Keith et al. (2021a) ones.

Before analyzing the main issues of this algorithm, I will introduce a part of my work, which concerned the reproduction of the same study of this section, but with deterministic transient noise (glitch).

3.4.2 Glitches

In modern interferometers, not only the Gaussian random noise is present. Unwanted deterministic features called glitches appear, too. They can be brought about by various known or unknown factors but mostly appear at a random time and are divided in families showing similar patterns in the spectrograms as can be seen in Fig. 2.7. Their typical signature in the spectrograms can be used to classify them, using both visual inspection and CNNs (Razzano and Cuoco, 2018), thanks mostly to the latters' excellent performance with image recognition.

In general glitches cannot be modelled easily but attempts to reproduce them using analytical formulas exists (Razzano and Cuoco, 2018) and have been used in the past to train CNNs and test their ability to correctly classify the glitches. In Fig. 3.4 a whistle glitch is plotted, and it is given by the following formula:

$$h(t) = h_0 \sin\left(\phi_{SL}\right) e^{-(t-t_0)^2/2\tau}\,, \tag{3.32}$$

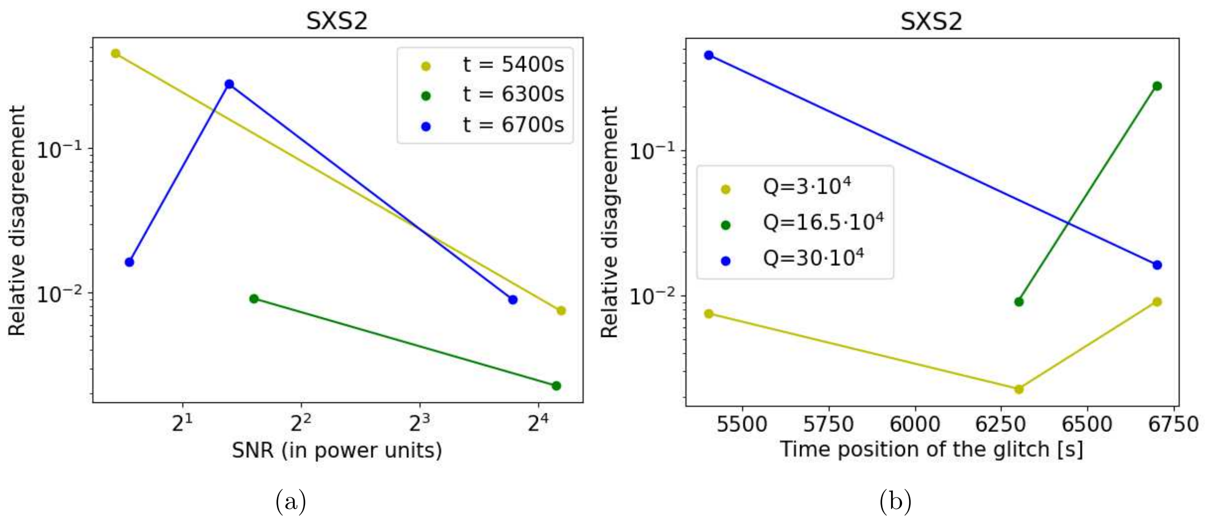


Figure 3.5: (a) SXS2.jl *performance with respect to SNR at fixed time positions for the glitch.* (b) SXS2.jl *performance with respect to the time position of the glitch at fixed widths Q.*

where h_0 is a normalization factor, t_0 is the center time, τ is the variance and ϕ_{SL} is given by

$$\phi_{SL} = 2\pi f_0(t - t_0)\left[1 - 3\tau(t - t_0)^2\right] . \tag{3.33}$$

Basically, this is a sin function modulated with a Gaussian centered in t_0 and with variance τ. This τ depends on the characteristic frequency f_0 thanks to the quality factor Q

$$\tau = \frac{Q}{\sqrt{2}\pi f_0} . \tag{3.34}$$

In this particular plot I set $h_0 = 0.5$, $f_0 = 10^3$ Hz, $t_0 = 6300$ s and $Q = 3 \cdot 10^5$. These are unrealistic values for this type of glitch (they usually last ~ 0.5 s), but this choice was made to cover one cycle of the considered waveform and due to the very low sample rate.

In this first work, I superimposed a series of nine whistle glitches onto the SXS2.jl waveform. This set is created with three different values for both t_0 and Q. The choice of t_0 was guided by the influence this glitch would have on the waveform signal: $t_0 = 5400$ s for a glitch occurring at the initial spiral phase, $t_0 = 6300$ s to approach the glitch coalescence and $t = 6700$ s to place the glitch at a time comparable to the coalescence one. For each of these t_0 sources, I chose three values for Q to extend the glitch over time: $Q \in (\{0.3, 1.65, 3\} \cdot 10^5)$. I created set of nine "waveform + glitches" superimposing each glitch on a SXS2.jl waveform. The set was provided as a training dataset for the algorithm. Being the waveform reconstruction the goal of the algorithm, glitches are treated as noise and should not appear in the results.

Exactly like in Sec. 3.4.1, I calculated the relative disagreement between the input NR waveform and the learned one. In panel (b) of Fig. 3.5, we can see the same type of plot where the relative disagreement is plotted against the SNR, considering the three different positions in time. Despite the lower SNR, this graph appears to agree with the one of Gaussian noise. In panel (c) of the same figure the relative disagreement is plotted against the temporal position of the glitch, with different values of Q. Form the graph we can infer that the proximity of the glitch to the coalescence brings more uncertainty. The loudest glitch case is not relevant because a local minimum was reached in the case $t_0 = 5400$. I must say that, in this algorithm, the presence of a strong glitch sometimes leads to the divergence of the system, with orbits no longer bound.

3.5 Discussion

The work by Keith et al. (2021a) surely brings a wonderful idea to have a relatively fast parameter inference from waveform measurements. The algorithm takes the waveform as a training dataset and, trying to minimize a cost function, learns the physical laws behind the data thanks to a PINN. In particular, the last example shows how to deduce the parameters of the emitting system, optimizing them together with the parameter of NN. The idea is certainly exciting and the algorithm seems stable enough, but hides some issues.

As we saw in Sec. 1.5.2 and in particular in the spectrogram of Fig. 1.10, a typical waveform measured by a ground-based interferometer remains in the detector for a few seconds if the signal comes from a BNS, or even for a few milliseconds if it comes from a BBH. The waveform considered by Keith et al. (2021a) instead extends over more than 10^3 s, with a sampling frequency of ~ 1 Hz [1]. Surely this choice was done to avoid a huge computational time, but this led me to ask how this waveform was generated.

The key ingredient to this issue is that the only parameter governing the two mass values is the mass ratio $q = m_1/m_2$. The two masses are defined as $m_1 = q/(1+q)$ and $m_2 = 1/(1+q)$. No other information is provided. However, having a mass ratio close to infinity in the first example and unitary in the last two, the values of the masses are constrained to vary in the interval $m_i \in [0, 1]$. Also, the code returns an error if $M = m_1 + m_2 \neq 1$. Furthermore, no units of measure for masses have been mentioned during this discussion. This constraint on the parameter space is very critical because, even considering two NS, the sum of the two masses should exceed 1 $M_\odot$.

However, as I said, the idea is certainly brilliant and in the next chapter I will start by implementing the same algorithm on the phenomenological formula (1.48) for the evolution in frequency of GWs.

[1]The sampling frequency of interferometers can be set to 4 kHz , but the default is 16 kHz.

Chapter 4

Learning the evolution of gravitational-wave frequency

The idea of my work, as I said in previous chapters, is to face the problem of inference of parameters about GW sources. In particular, pipelines currently operating on modern ground-based detectors needs to do this in a fast way, in order to trigger alerts for electromagnetic counterparts' searches. Some of them have already implemented a rough parameter estimation (matched filtering). From the waveform signal of a CBC, we can infer fifteen parameters of the source, in case of a BBH: the two masses, six spin components, an angle describing the polarization of the wave, the inclination θ of the binary with respect to the line of sight, two values for the sky position (right ascension and declination), the luminosity distance D of the source, the reference time and the phase of the orbit at reference time. After the detection, all these parameters can be precisely inferred with offline Bayesian techniques described in Sec. 1.5.4, but with an extremely long computational time (approximately six months) and a a precise model of the expected waveform. What lacks between detection and parameter estimation is a relatively fast parameter inference which can enter in the transient burst searches, using as less physical prior information as possible. How can we implement a fast parameter inference on this latter technique?

We can face this problem learning and solving physical laws behind the waveform data as PDEs, thanks to PINNs. The idea of Keith et al. (2021a) that we want to exploit is to optimize parameters of a user-defined UDE in order to minimize a PINNs' cost function, possibly learning new physics in the meantime. As training data, the reconstructed cWB waveform can be used. This approach is really promising because we can both infer parameters of the source and learn unknown terms of physical laws, starting only from a simple physical ansatz.

Anyway, the work of spanning in a fast way all fifteen parameters is extremely long. This is why this thesis is only the starting point of it, putting the base for future works. I will concentrate my study on the phenomenological differential equation (1.48) for the frequency evolution of a chirping GW signal:

$$\frac{df}{dt} = \frac{96}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3} f^{11/3}\,, \tag{4.1}$$

where $\mathscr{M}$ is the chirping mass defined in (1.37). We start from a simple, known differential equation and infer the chirp mass $\mathscr{M}$, the phase of the signal ϕ, which enters in the amplitude formula (1.51), and/or the 11/3 exponent for the frequency term. In this way we can analyse the behavior of the algorithm, handling a small 3D parameter space within a known formula. The goal is to put the base in order to extend the dimensionality of the space in a future work. In particular, learning the 11/3 exponent is surely a fundamental first step in order to extend the learning to higher PN orders.

After a short recap on the basic physics in sec. 4.1, in this chapter I will present my work. I implemented two different algorithms, in order to analyse which approach performs better. Firstly, I tried with a SciML implementation (sec. 4.3), learning a user-built UDE. Secondly, I concentrated my study on a PyThorch implementation of Hybrid PINNs (HPINNs; sec. 4.4). The idea is to test both these ways of proceeding to understand how to continue in a future work, stating good results if they are present and discussing issues to overcome if no convergence is reached (sec. 4.4.4).

4.1 Gravitational wave frequency evolution

In sec. 1.3, we derived main phenomenological equations that describes the evolution of a signal coming from a CBC. I summarize here the main expressions. Because of the back reaction on the compact binary system, the GW frequency evolution is given by (4.1). In the notation I am using the GW frequency is given by $f = \omega/\pi$, where ω is the orbital angular velocity of the binary and f is twice the orbital frequency. The (4.1) expression has the known solution (1.49):

$$f(t) = \left[f_c^{-8/3} - \frac{256}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(t-t_c)\right]^{-3/8}, \tag{4.2}$$

where t_c is the time of coalescence and f_c is the maximum frequency (1.50), given by

$$f_c = f(t_c) = \frac{1}{2\pi\sqrt{2}}\left(\frac{GM_\odot}{c^3}\right)^{-1}\left(\frac{M}{M_\odot}\right)^{-1}, \tag{4.3}$$

where $M = m_1 + m_2$ is the sum of the two masses. Furthermore, we were able to describe the GW amplitude h_{ij} as a function of the frequency f as (1.51):

$$h_{ij}(ct, \mathbf{r}) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\begin{pmatrix} \cos(2\pi f t + \phi) & \sin(2\pi f t + \phi) & 0 \\ \sin(2\pi f t + \phi) & -\cos(2\pi f t + \phi) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{4.4}$$

where D is the luminosity distance of the system and ϕ is the waveform phase. This latter depends on the retarded time of production and in particular on the phase of the orbit. Considering the two GW's polarizations, one gets

$$h_+ = h_{11}(ct, \mathbf{r}) = -h_{22}(ct, \mathbf{r}) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\cos(2\pi f t + \phi), \tag{4.5}$$

$$h_\times = h_{12}(ct, \mathbf{r}) = h_{21}(ct, \mathbf{r}) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\sin(2\pi f t + \phi). \tag{4.6}$$

We have seen in sec. 1.4 that the interferometer's output is a linear combination of these h_+ and $h_\times$, given by

$$h(t) = F_+(\theta,\phi)\, h_+ + F_\times(\theta,\phi)\, h_\times . \tag{4.7}$$

Simplifying this approach, we can consider the single interferometer output as

$$h(t) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\big[\cos(2\pi f t + \phi) + \sin(2\pi f t + \phi)\big]. \tag{4.8}$$

Equally we can write

$$h(t) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\cos\left(2\pi f t + \phi\right), \tag{4.9}$$

by shifting ϕ by a reasonable amount. With the algorithms discussed in th next sections we will exploit (4.1) and (4.9) to infer the chirp mass $\mathscr{M}$, the phase ϕ and the 11/3 exponent. This simple approach is the fundamental first step to test algorithms that in future aims to infer a much larger parameter space.

Noteworthy is that these expressions (4.1) and (4.9) are only the leading Newtonian terms of a full GR solution. As we have seen in sec. 1.1.1, a PN expansion is possible with v/c terms. In particular, Blanchet et al. (1996) expanded the $f(t)$ solution (4.2) as

$$\begin{aligned} f(t) = -\frac{c^3}{8\pi GM}\Bigg[&\Theta^{-3/8} + \left(\frac{743}{2688} + \frac{11}{32}\eta\right)\Theta^{-5/8} - \frac{3\pi}{10}\Theta^{-3/4} + \\ &+ \left(\frac{1855099}{14450688} + \frac{56975}{258048}\eta + \frac{1855}{2048}\eta^2\right)\Theta^{1/8}\Bigg], \end{aligned} \tag{4.10}$$

where here $\eta = m_1 m_2/M^2$ and the expansion term is the dimensionless time variable

$$\Theta = \frac{c^3\eta}{5GM}(t - t_c). \tag{4.11}$$

η and M are bound to the chirp mass as $\mathscr{M} = \eta^{3/5}M$. In (4.10) we can see that the exponent of Θ changes between terms. Reconstructing the 11/3 exponent in (4.2) is a first tentative to learn new expansion terms.

4.2 Runge-Kutta at 4th order

Before introducing the two implementations of my algorithm, I anticipate here an integration technique to solve differential equations. This will be useful in next sections.

The task of numerically solve a differential equation is a problem that arises since the first differential calculus era (Leibniz (1684); Newton (1745)). Through the centuries different solutions was developed (see Davis et al. (2014) for a review), but the most general and accurate one was given by Runge and König (1924). This is an iterative method where every iteration permits to calculate the subsequent value of the solution within a time discretization.

Suppose that we want to solve the following first-order differential equation

$$\frac{dy}{dt} = f(y, t). \tag{4.12}$$

To integrate this expression, we can numerically use the following Taylor expansion

$$y_{t+h} = y_t + h\frac{dy}{dt} + \mathcal{O}(h^2) = y_t + hf(y_t, t) + \mathcal{O}(h^2), \tag{4.13}$$

where here h is the time increment. Taking only the first order in h we have the well known Euler method. This means that, if we know the value of y at the time t and we define the desired time step h, we can calculate the y value at time $t + h$. Following this path for the whole time range, we have a numerical solution for (4.12). As we can see, the error scales as h^2.

If we instead Taylor expand the solution around $t + h/2$, we have

$$y_{t+h} = y_{t+h/2} + \frac{h}{2}\left(\frac{dy}{dt}\right)_{t+h/2} + \frac{h^2}{8}\left(\frac{d^2y}{dt^2}\right)_{t+h/2} + \mathcal{O}(h^3)\,, \tag{4.14}$$

$$y_t = y_{t+h/2} - \frac{h}{2}\left(\frac{dy}{dt}\right)_{t+h/2} + \frac{h^2}{8}\left(\frac{d^2y}{dt^2}\right)_{t+h/2} + \mathcal{O}(h^3)\,. \tag{4.15}$$

Subtracting the first equation from the second one, we find

$$y_{t+h} = y_t + h\left[\frac{dy}{dt}\right]_{t+h/2} = y_t + hf\left(y_{t+h/2}, t + \frac{h}{2}\right) + \mathcal{O}(h^3)\,. \tag{4.16}$$

Here the error scales as h^3, having cancelled out second-order terms, but we need to calculate y at $t + h/2$. This must be done via the Euler's method

$$y_{t+h/2} = y(t) + \frac{h}{2}f(y(t), t). \tag{4.17}$$

Cancelling out with the same method the h^3 and h^4 terms, we have the Runge-Kutta method at 4th order (RK4), which is given by

$$k_1 = \frac{h}{2}f(y_t, t)\,, \tag{4.18}$$

$$k_2 = \frac{h}{2}\left(y_t + k_1, t + \frac{h}{2}\right), \tag{4.19}$$

$$k_3 = h\left(y_t + k_2, t + \frac{h}{2}\right), \tag{4.20}$$

$$k_4 = h(y_t + k_3, t + h)\,, \tag{4.21}$$

$$y_{t+h} = y_t + \frac{1}{6}(2k_1 + 4k_2 + 2k_3 + k_4)\,. \tag{4.22}$$

Simply knowing the y value at time t and setting the time step h, we can calculate the value of y at time $t + h$ with an error which scales as h^5.

4.3 SciML implementation

This first algorithm is based on the same concept of Keith et al. (2021a), with the aim of approximating a PDE with the use of the UDE formalism. In particular, I used the SciML environment written in Julia programming language. I decided to construct the following UDE, inspired by (4.1),

$$\frac{df}{dt} = \frac{96}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}\mathscr{F}(f, \boldsymbol{\xi})\,, \tag{4.23}$$

where $\mathscr{F}$ is a feed-forward PINN defined as in (2.28), with $\boldsymbol{\xi}$ parameters. The goal is to learn the $f^{11/3}$ term thanks to the UA behavior of the NN and to treat the chirp mass $\mathscr{M}$ as a parameter to infer. The training dataset, which will be described later in detail, is given by 2048 data points $\{(t_k, h_k)\}_{k=1}^{n}$, where t_k is the independent time variable, h_k are amplitude strain values for (4.9) and $n = 2048$.

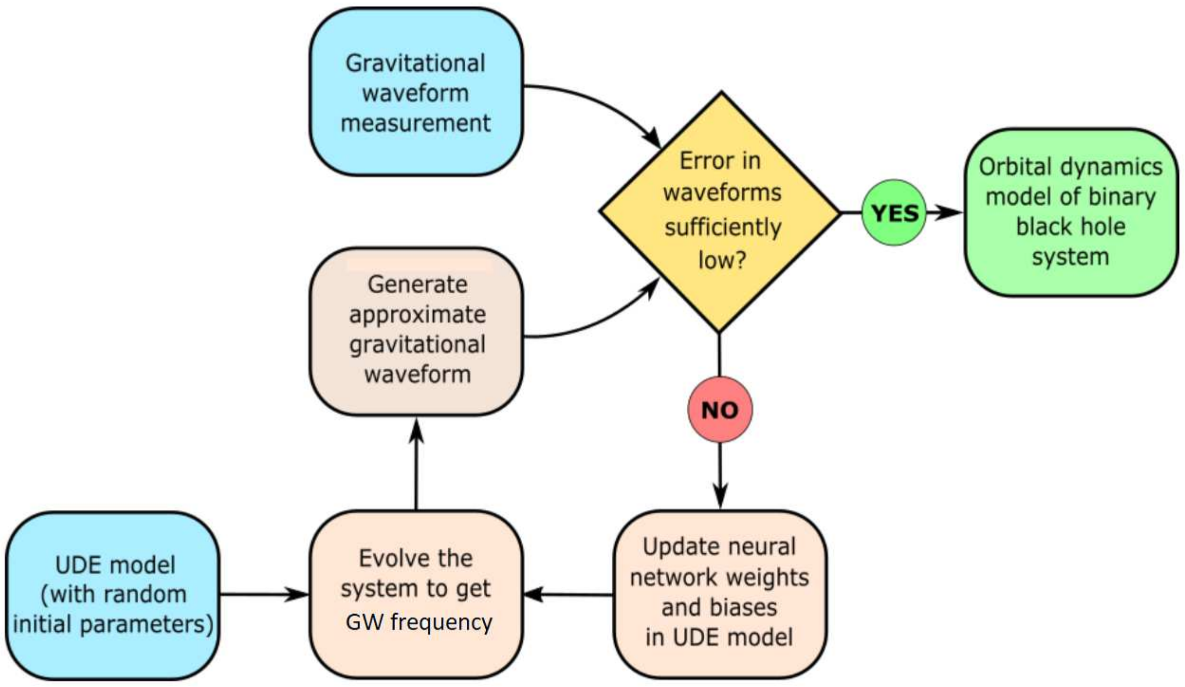

Gravitational waveform measurement
Error in waveforms sufficiently low?
YES
Orbital dynamics model of binary black hole system
Generate approximate gravitational waveform
NO
UDE model (with random initial parameters)
Evolve the system to get GW frequency
Update neural network weights and biases in UDE model

Figure 4.1: SciML *algorithm's training loop flowchart.*

4.3.1 Algorithm implementation

The basic structure of the algorithm can be seen in Fig. 4.1. Once the UDE model (4.23) is set, the $\boldsymbol{\xi}$ and $\mathscr{M}$ parameters set to guess values and are slightly randomized thanks to a Gaussian random noise generator with variance $\sigma^2 = 10^{-2}$. This differential equation is solved thanks to a RK4 method, producing an f array evolving with time. Substituting this in (4.9), we can have predicted values for h. To compare this predicted $h(t)$ with the training data h_k, the following cost function was built:

$$\mathscr{C} = \frac{1}{n}\sum_{k=1}^{n} ||h_k - h(t_k)||^2. \tag{4.24}$$

Notice that the RK4 step can have a precise arbitrary time-step (smaller than $t_{k+1} - t_k$), but to have a useful comparison $\mathscr{C}$, $h(t)$ must be sampled precisely at t_k. If the calculation of this cost function results in a sufficiently low value, different guesses for $\boldsymbol{\xi}$ and $\mathscr{M}$ are fixed and a reconstruction for (4.1) via (4.23) is performed. If instead the value for $\mathscr{C}$ is too high, $\boldsymbol{\xi}$ and $\mathscr{M}$ are optimized with BFGS algorithm to modify the UDE guess in (4.23) and to have better values for the predicted $h(t)$. Notice that $\mathscr{M}$ enters in both the UDE (4.23) and (4.9): the optimization algorithm can act both on the approximation of the PDE and on the amplitude formula. This will effect enormously the results, as we will see. The training loop continues until convergence (or a maximum number of epochs) is reached.

Initial conditions The first thing to say is that, differently from Keith et al. (2021a), I've chosen to use the International System's units of measure, with the Newton's universal gravitational constant $G = 6.67 \cdot 10^{-11}$ N m^2 kg^{-1}, the light velocity $c = 3 \cdot 10^8$ m s^{-1} and the solar mass $M_\odot = 1.99 \cdot 10^{30}$ kg. Furthermore, I took the luminosity distance for the object as $D = 3.09 \cdot 10^{22}$ m (= 1 Mpc), a reasonable value considering CBC events already detected (Abbott et al., 2021b). Because of this, a factor of 10^{19} is multiplied with (4.9) to avoid purely numerical errors. This is not fully arbitrary because usually

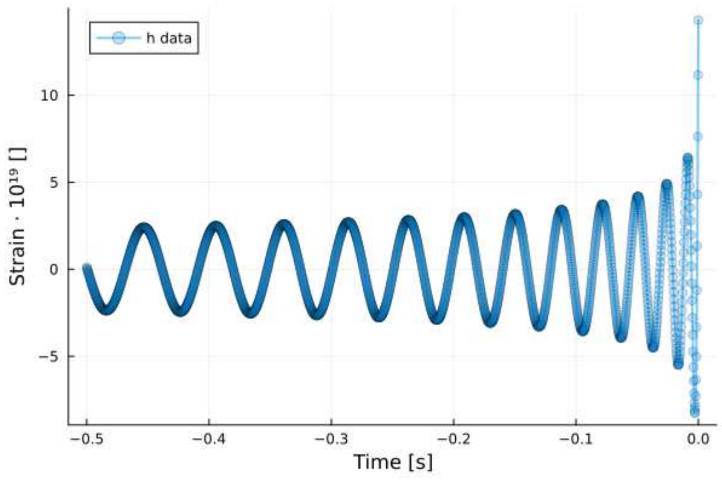


Figure 4.2: *Training dataset for the* SciML *algorithm.*

the whitening phase, to have a noise not dependent from the frequency, passes through a ratio between the detector's output and its amplitude spectral density, which is in the order of $\sim 10^{-19}$.

Since the RK4 solver for the f differential equation (4.1) needs the initial condition for the solution, the initial (exact) value for f is given to the algorithm, calculated thanks to (4.2), and remains fixed. If in future this algorithm is taken into account this constrain is one of the first things to remove, learning the initial value for f, too. Here we know the exact solution, but to make this algorithm more general, one must learn the initial condition. This will be discussed in the PyTorch algorithm and in sec. 4.4.4.

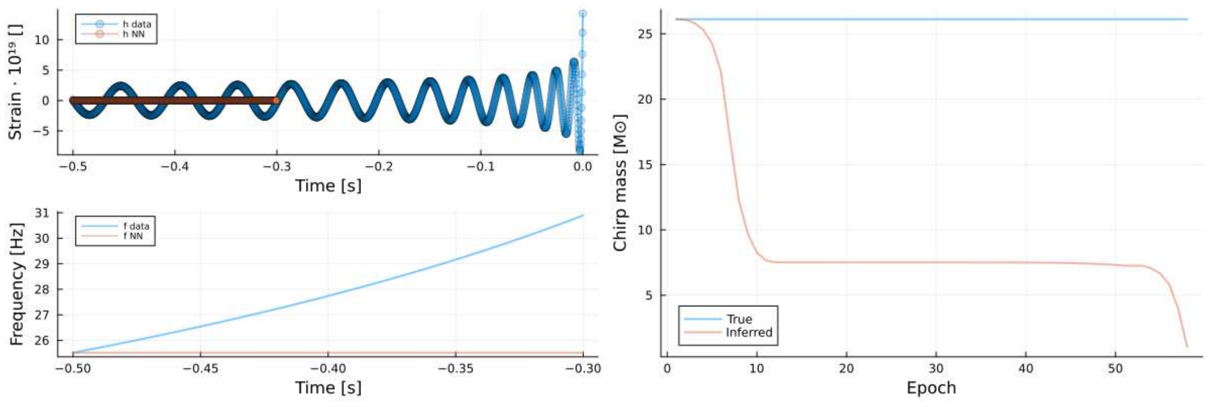
Training dataset The training dataset, which can be seen in Fig. 4.2 as "h data", is generated starting from the set of 2048 time points $t_k \in [t_c - 0.5\ \mathrm{s}, t_c]$ where t_c is the time of coalescence, that I set to 0 s. This number of points is considered because the actual interferometer sampling rate can be set at 4096 Hz. Thanks to (4.2) and to (4.9), I did generate the training dataset as $\{(t_k, h_k)\}_{k=1}^{n}$. For this particular case, I considered a system of equal masses $m_1 = m_2 = m = 30\ \mathrm{M}_\odot$, with a target chirp mass $\mathscr{M}_T$ given by

$$\mathscr{M}_T = \frac{m}{2^{1/5}} \approx 26.11\ \mathrm{M}_\odot\,, \tag{4.25}$$

and $\phi = 0$.

Neural network architecture The neural network is defined as a feed-forward NN with two fully connected hidden layers, each with 32 neurons. All hidden layers' activation functions are set as tanh. The first layer is instead constituted by 5 neurons with f^α activation function, with $\alpha = 1, \dots, 5$. The output layer is finally given by a single neuron with a linear activation function.

Regularization strategies In this case no regularization term on $\mathscr{C}$ was added, but I took advantage of the already coded Keith et al. (2021a) method of dividing the dataset in subsequent increasing subsets. Starting the training only on a small subset, we can avoid local minima. Indeed, since the function to fit is sinusoidal, working with a small amount of cycles helps one can avoid frequencies that causes a 2π extra phase inside the training waveform time range.



(a) (b)

Figure 4.3: SciML *algorithm divergence:* (a) *amplitude and frequency behaviour at the last useful epoch and* (b) *chirp mass divergence.*

4.3.2 Results

Despite the solid computational background, the results for this SciML algorithm were pretty bad. As we can see in panel (b) of Fig. 4.3, the chirp mass value (even with a right initial guess) keeps increasing its difference with the real value, until a local minimum (the plateau). This is reached with only the 30% of the training data, but when the 40% is then considered, a divergence to negative values of $\mathscr{M}$ is reached and the algorithm crashes. In panel (a) we can see the situation for the strain amplitude and for the frequency solution after the last useful epoch: the NN generated waveform is flat, as well as the frequency solution for the learned UDE.

The main point of this divergence is that the $\mathscr{M}$ parameter is present both in the UDE (4.23) and in the the amplitude formula (4.9). Since $\boldsymbol{\xi}$ parameters are hidden inside $\mathscr{F}$, they are way stiffer than $\mathscr{M}$. For this reason, the algorithm prefers to reach a flat amplitude than to learn the real UDE and modify the frequency derivative.

Different attempts were done to solve this issue. A smaller learning rate was tested, but it gave the same results, only with the need of more epochs because of the smaller size of the steps. An ADAM optimizer was implemented instead of the default BFGS because of the different stiffness between parameters, but again no convergence was seen.

4.3.3 Discussion

In a future work, if one wants to proceed with this idea, this issue will be the first thing to overcome. An important role is played by initial conditions and maybe learning also the initial state of the frequency (possibly dependent on $\mathscr{M}$ itself) could help a lot. Another useful thing one could try is to add regularization terms to the cost function, as Keith et al. (2021a) did for their algorithms. Another tentative that one could do is to create a more complex architecture for the NN to further increase the approximating capability of this UA. Anyway, despite we will see that a local minima issue emerges anytime when fitting a sinusoid, the problem behind this algorithm divergence seems to be more profound. Furthermore, the fact that the algorithm is written using the Julia programming language and the SciML environment doesn't help at all. Both of them are almost new with respect to others, causing a lack of literature and online forums. For this reason, in the next algorithm I implemented the PINN approach in a PyTorch environment, following the work done by Nascimento et al. (2020).

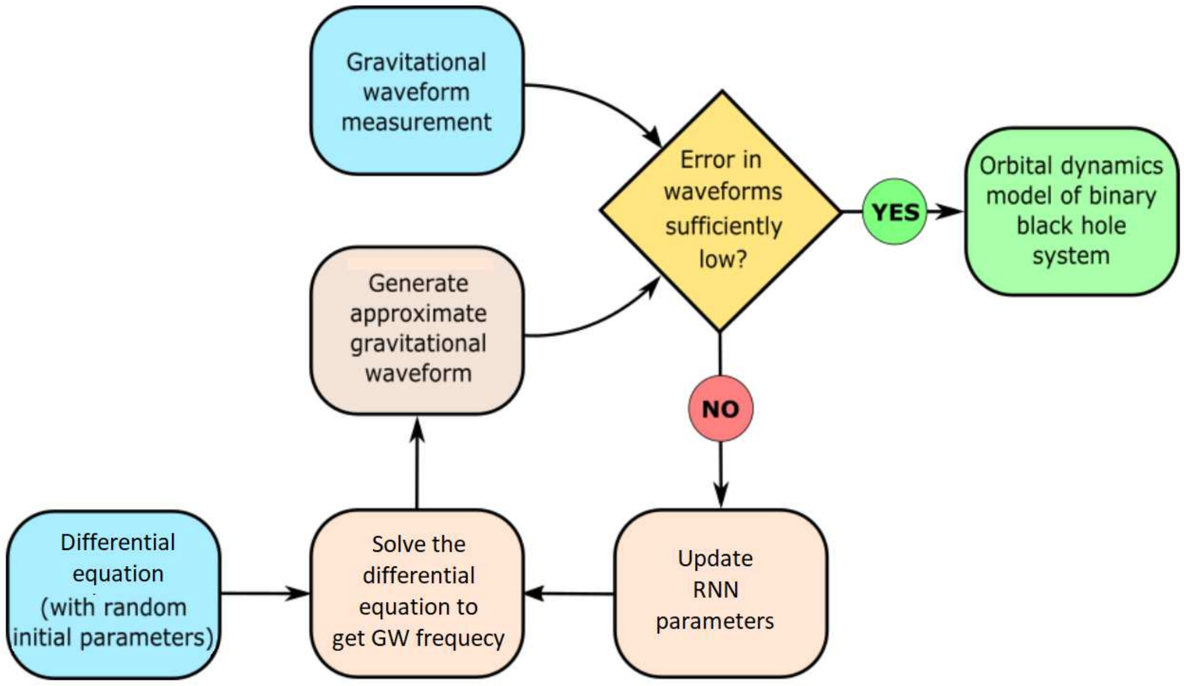

Gravitational waveform measurement
Error in waveforms sufficiently low?
YES
Orbital dynamics model of binary black hole system
Generate approximate gravitational waveform
NO
Differential equation (with random initial parameters)
Solve the differential equation to get GW frequecy
Update RNN parameters

Figure 4.4: PyTorch *algorithm's training loop flowchart.*

As a first conclusion, the approach of this SciML algorithm, once this issue will be overcome, will be really useful because it permits to both reconstruct differential equations with new terms and infer its parameters. Bound with data analysis pipelines that consider as less physical information a priori as possible, this algorithm could provide a tool to have parameter inference. Unfortunately, right now it dramatically diverges. Anyway, this will surely be a future exciting challenge to investigate.

4.4 PyTorch implementation

4.4.1 Hybrid physics-informed neural networks to solve differential equations

In this algorithm I changed a bit the approach to an Hybrid PINN (HPINN) one. This framework was developed by Nascimento and Viana (2020) and was originally implemented in PyTorch for improving the accuracy of cumulative damage models used to predict wind turbine main bearing fatigue. The idea of HPINNs is to implement well developed numerical solvers inside RNNs, physics-informing it. Indeed, if a PDE is known, no approximation on the equation itself is needed and so one can concentrate in solving it. This HPINN environment was built for this purpose. The authors used a Recurrent Neural Network with a Runge-Kutta at 4th order (RK4) method implemented inside the neuron cell. With this, they solved a partial differential equation of the second order. Training this RNN means to optimize the NN parameters in order to better fit solution data to the PDE. Furthermore, Nascimento et al. (2020) have shown that the parameters that appears in the PDE can be optimized as NN parameters and so a parameter inference is possible. The authors provided a tutorial that an interesting reader could find very useful to produce similar algorithms.

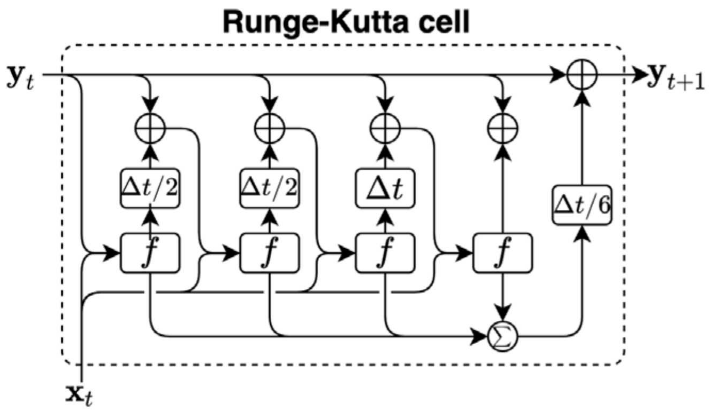


Figure 4.5: *RK4 cell for the RNN.* (Nascimento et al., 2020)

Recurrent Neural Networks As we have seen in sec. 2.2.3, RNN are a way to extend usual feed-forward neural network in order to handle time evolving inputs. Back-links and loops are implemented between different layers. As an extreme case, a single neuron can be used in a loop, as we have seen in the Fig. 2.5, where A is the single cell. More formally, Nascimento et al. (2020) define a RNN as an algorithm which applies a transformation f to a state $\mathbf{y}$ such that

$$\mathbf{y}_t = f(\mathbf{y}_{t-1}, \mathbf{x}_t)\,, \tag{4.26}$$

where $t \in [0, T]$ is the time discretization, $\mathbf{y} \in \mathbb{R}^{n_y}$ are the states representing the quantities of interest and $\mathbf{x} \in \mathbb{R}^{n_y}$ are input variables. As we can see, the output state $\mathbf{y}_t$ will be the input state for the next cell loop.

Hybrid physics-informed neural networks Nascimento et al. (2020) proposed to implement a RK4 inside a RNN. The structure of the cell can be seen in Fig. 4.5. Considering for example the first order differential equation

$$\frac{d\mathbf{y}}{dt} = f(\mathbf{y}, \boldsymbol{\xi}), \tag{4.27}$$

$\mathbf{y}$ is the differential equation solution and $\boldsymbol{\xi}$ are input variables (in Fig. 4.5 $\mathbf{x}_t = \boldsymbol{\xi}_t$). Every cycle the cell accepts as input the $\mathbf{y}$ and $\boldsymbol{\xi}$ values at time t and returns the $\mathbf{y}$ value at time $t+1$. What the algorithm needs to solve (4.27) is the initial condition $\mathbf{y}_0$ and the different values of $\mathbf{x}_t$. After that, the loop starts and the differential equation is solved. At the end of the procedure, the NN produces a set of $\mathbf{y}_t$ values that represents the solution of (4.27) as a function of time.

How can we train this RNN? As usual, we need to define a cost function $\mathscr{C}$ to minimize. Having a $(t_k, \mathbf{y}_k)$ training dataset, $\mathscr{C}$ will compare the predicted $\mathbf{y}$ results with the data $\mathbf{y}_k$. Optimizing the RNN parameters, we can act on the predicted solution $\mathbf{y}$ to minimize $\mathscr{C}$. Noteworthy here is that the only way to optimize different parameters is to make them part of the differential equation to solve. This will be fundamental for our purposes.

4.4.2 Algorithm implementation

The second algorithm is based on (4.1) and (4.9). We can slightly change the f derivative (4.1) as

$$\frac{df}{dt} = \frac{96}{5}\pi^{8/3}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3} 1^\phi f^\alpha\,, \tag{4.28}$$

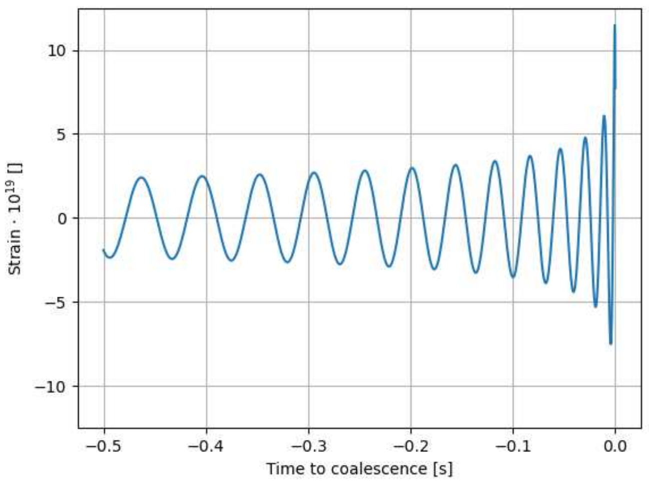


Figure 4.6: PyTorch *algorithm's training dataset.*

where $\mathscr{M}$ is the chirp mass, ϕ is the phase of the GW signal[1] and α is the f exponent. The goal of this algorithm is to infer $\mathscr{M}$, ϕ and α parameters thanks to the HPINN environment described in sec. 4.4.1. In this case, we have a known differential equation that we want to solve. What we will going to learn are parameter values of the differential equation itself, no new terms. Anyway, as we have seen in sec. 4.1, the PN approximation adds new terms with a different exponent for the f time dependence. Anyway, the learning for the exponent α is the first step to extend the approach to new terms. Indeed, I remember that the main goal of my thesis is to put the base study for a future much deeper work, with a bigger parameter space to infer.

The structure of the learning process can be seen in Fig. 4.4. First of all, (4.28) is defined with random guesses for the parameters. The second passage is to solve the differential equation thanks to the RNN with the RK4 cell implemented. Here in particular the initial condition for f is needed to start the integration. This is a problem because the initial frequency must be obtained knowing the solution (4.2) to the already optimized differential equation. If we suppose to don't know the analytical solution, this situation leads to an information that we can't give to the algorithm. For this reason, I changed the direction of the RK4 integration, setting the time increment $h < 0$ in (4.18 - 4.22). In my algorithm the integration is done supposing to know the maximum frequency f_c and evolving the RK4 cell backward in time. This is helpful for two reasons. Firstly, the information on the initial value of f is not analytically calculated, but with real data it can be measured. Secondly, since the derivative explodes at $t = 0^-$, starting from the end permits to avoid numerical divergences when integrating. Once the f solution is obtained, the $h(t)$ GW amplitude is calculated thanks to (4.9):

$$h(t) = \frac{4c}{D}\left(\frac{GM_\odot}{c^3}\right)^{5/3}\left(\frac{\mathscr{M}}{M_\odot}\right)^{5/3}(\pi f)^{2/3}\cos\left(2\pi f t + \phi\right), \tag{4.29}$$

where we see how the ϕ parameter influences the physics. The training dataset will be again a set of 2048 points (t_k, h_k), with $t \in [t_c - 0.5s, t_c]$, where t_c is the coalescence time. It can be seen in Fig. 4.6. In particular, the luminosity distance D was set at 1 Mpc $= 3,086 \cdot 10^{22}$ m. To avoid purely numerical error, in the generation of $h(t)$ I multiplied the (4.29) for a 10^{19} factor. Thanks to the predicted $h(t)$ and to the h_k measurements,

[1]It is the same ϕ which appears in (4.9).

	$\mathscr{M}$	ϕ	α
Target value	26.117 $\mathrm{M}_{\odot}$	1 rad	11/3

Table 4.1: *Target values behind the dataset.*

we can calculate the loss function as a mean squared error:

$$\mathscr{C} = \frac{1}{n}\sum_k (h_k - h(t_k))^2 \,, \tag{4.30}$$

where n is the total number of training data points. If the loss function value is sufficiently low or the number of iterations is bigger than the desired one, the algorithm stops and parameters are inferred. On the other hand, if the loss is too high and the number of iterations is lower than the desired value, parameters are updated thanks to the RMSprop algorithm dependent on the squared root of the gradient, more useful when RNNs are trained (the PyTorch collaborators, 2023). For this, I've chosen a learning rate $\eta_t = 10^{-2}$ for the reasons that I will discuss later. The (4.28) with updated parameters is solved and another solution to f is found. When the loss value is sufficiently low or the number of iterations is bigger than the desired one, the algorithm stops and the parameters are inferred.

Training dataset As we can see in Fig. 4.6, the training dataset is given by 2048 (t_k, h_k) data points, spanning 0.5 s before the coalescence of the system. They are generated starting from t_k points sampled at 4096 Hz. Thanks to (4.2) and (4.29), h_k values was found. This mock signal was generated assuming equal masses $m_1 = m_2 = m = 30\ \mathrm{M}_{\odot}$, and so $\mathscr{M} = m/2^{1/5} \approx 26.117\ \mathrm{M}_{\odot}$. Furthermore, the target values for the rest of the parameters are $\phi = 1$ rad and obviously $\alpha = 11/3$, as we can see in Tab. 4.1.

Initial conditions For what concerns initial conditions, the only needed quantity is the maximum frequency f_c. This is necessary for the inverse RK4, as I previously described. This is a not trivial information, since f_c depends on the sum of the masses thanks to (4.3). Anyway, this quantity can be easily measured for present CBC signals, which lasts in the interferometers for fractions of seconds. This will not be the case for signals detected by LISA, but for the purpose of this thesis, we can stay in a framework where f_c is known.

Parameter guesses 124 runs in total were performed considering both different initial conditions and constrained parameters. For $\mathscr{M}$ were chosen the ensemble $\mathscr{M}_G = \{20, 25, 26.11, 30\}\ \mathrm{M}_{\odot}$, for ϕ I chose to use $\phi_G = \{0.1, 0.9, 1., \pi\}$ rad, while for α, $\alpha_G = \{3, 10/3, 11/3, 4\}$ values was used. This work has the goal to understand how the algorithm performs with different conditions, so initially 4 runs for every parameter were performed with only the corresponding parameter free. For example, considering the $\mathscr{M}$ parameter, 4 runs were performed spanning $\mathscr{M}_G$ and considering ϕ and α fixed to the target value. This was done for every subset, for a total of 12 runs with only one parameter to learn. 16 runs was then processed with two free parameters, spanning their respective guess space, with the third parameter fixed at target value (for a total of 48 runs with two three parameters). Finally, 64 runs were done spanning the three ensembles, with all the parameters free to vary.

4.4.3 Results

Results are shown in Tab. A.1 in App. A. For every run, one can see the guess starting value for parameters, separated by the type and the number of parameters to learn. At each guess is associated its inferred value after 100 epochs. The final value for the loss function is also shown. In the last column I noted the right guess as "RG", the minimum loss value for the set of runs as "mL" and the divergences as *Div.*

In Fig. A.1 are shown histograms of the inferred value separated in color by their initial guess. Also, for the analysis of this results was really useful to produce surface plots and a 3D plot for the loss value as a function of parameters (Fig. A.2 - Fig. A.5). For every run, a plot for the loss and for the evolving parameter values along the epochs were performed. Furthermore, plots showing the initial and the final solution for both the frequency and the strain components were produced. From Fig. A.6 to Fig. A.9, we can see the plots produce for the RG runs. Plots from Fig. A.10 to Fig. A.14 shows useful runs for the properties of the algorithm that we will discuss later.

4.4.4 Discussion

From a first look at Tab. A.1, we can notice that the final values doesn't deviate so much from the starting guess. This is definitely more evident in the Fig. A.1. Considering for example the chirp mass $\mathscr{M}$ in panel (a), we can see that the four peaks in the global distributions are given exactly by the four different guesses. The same is seen for the phase ϕ and for the exponent α. The reason for this behavior of the algorithm is in the figures from given from Fig. A.2 to Fig. A.5. These maps represents the loss value as a function of the different parameters. For producing these plots, I simply selected an array of 10^3 values for each parameter and created a color map calculating the loss value for every point. Taking $\mathscr{M}$ as reference, we can see in the top panel of Fig. A.2 that the loss function is full of local minima. The same is true for α. For what concerns ϕ, as expected, the loss behavior has a 2π periodicity. In the surface plots, we can see how these minima depends on couples of parameters. Fig. A.2 and Fig. A.4 are pretty similar, since different minima for the top panel are in reality long and thin oblique minima, shifted by the *sin* behavior for the ϕ loss. Fig. A.3 is much more interesting because we can see a profound minimum corresponding to the right guesses. This absolute minimum anyway behaves as a thin hyperbola. Relative minima are hyperbolas as well, highlighting a dependence of one parameter on the other. In Fig. A.5 is shown the loss value in the three parameters volume. The absolute minimum is evident as a thin hyperbolic surface, which is wiggled by the ϕ sinusoidal behavior. This abundance of local minima is what guides the algorithm behavior, producing the little displacement between guesses and inferred values.

Different approaches are possible in a near future to jump out from a minimum. The main reason why the algorithm stacks into local minima is the small value for the learning rate $\eta_t = 10^{-2}$. This is imposed because of the sensibility on the exponent α. Anyway, one can fine-tune the learning rate in order to balance α sensitivity and local minima. Another way can be a change in the optimization algorithm to a momentum-guided one. For both $\mathscr{M}$ and α, the loss reaches high values at the edges of the considered domain. Using a momentum-guided algorithm and starting from an high-edge (from high $\mathscr{M}$ and low α), the algorithm can have some inertia to avoid local minima. The last improvement that I propose is a change in the cost function that we want to minimize (in the loss value calculation). Instead of a square in (4.30), one can use an higher even power, in order to increase local minima's loss values and decrease the absolute minimum's one. This absolute minimum will be more marked and a plateau will form at the bottom of the positive curvature, but the algorithm will be facilitated to enter in it and don't move.

Single parameter inference Runs from 1 to 12 in Tab. A.1 are performed learning only one parameter, taking the other two fixed to target values.

If we start with an initial $\mathscr{M}$ guess close to the target value $\mathscr{M}_T = 26.117$ M$_\odot$ or at least within the global minimum, we can reach really low values for the loss function (in the order of $\sim 10^{-4}$) in few epochs (see also panel (a) of Fig. A.6). If we instead start with an $\mathscr{M}$ guess too distant, we fall into a local minimum.

For what concerns ϕ, Its learning it's much more difficult because of the large minimum that we can see in the right panel of Fig. A.2. Anyway, the gradient at the minimum is so small that if we start with the right guess, no update is done on the parameter.

The parameter on which the algorithm is most sensible is surely α. Since it is the exponent of the f term in the differential equation, from it depends the whole behavior of the frequency solution. We can see this great dependence in Fig. A.10, where with a difference in α of 0.5 between the first and the last epoch, the frequency solution changes a lot. This sensitivity on α is also causing a divergence when starting with $\alpha = 4$. We can see this in Tab. A.1. This is because, for such values of α, the derivative nearly diverges to high values, causing $f = -\infty$ (numerically) in the differential equation integration.

Two parameters inference Runs from 13 to 60 in Tab. A.1 are performed learning two parameters, taking the third one as fixed to target values.

As discussed above, in this case the local minima are much more complicated than for the single parameter case. This is both better or worse. We have to deal with two parameters and so the $\eta_t = 10^{-2}$ condition must be applied not only to α. In different runs shown in this thesis, in all of them the η_t parameter has always the same value for coherence, but one can see that changing the learning rate to $\eta_t = 10^{-1}$ inferred values for $\mathscr{M}$ and ϕ are stabilized better and faster. Anyway, learning rate relatively this big is impossible to handle because of the α sensibility and divergence. On the other hand the α sensibility can help to jump out of local minima. In Fig. A.11 we can see on the left the run 1 and on the right the run 30. The starting guess for $\mathscr{M}$ is the same, but in the 30th run α is inferred, too. On the left a local minimum at $\mathscr{M} = 20.202$ M$_\odot$ is easily reached and the algorithm stacks there. Introducing the α learning, this minimum is avoided. Looking at the surface in Fig. A.3, we can see that the starting point $(\mathscr{M}, \alpha) = (20\ M_\odot, 3.333)$ is a point in the bottom-left violet plateau. The sensitivity of the algorithm on α is sufficient to permit to $\mathscr{M}$ to jump outside the local minimum. Despite the fact that $\mathscr{M}$ is then blocked in another minimum, this principle can be studied in a near future to have better performances.

Fig. A.12 shows the run 2 in panel (a) and the run 19 in panel (b). Again the $\mathscr{M}$ guess value is the same, but in the panel (b) also ϕ is learnt. We can clearly see that, as for the run with the single ϕ guess equal to $\phi_T = 1$, the ϕ value doesn't change and the $\mathscr{M}$ learning proceeds in the same way.

Three parameters inference Runs from 61 to 124 in Tab. A.1 are performed learning all three parameters.

In this case the loss variation with respect to the different parameters can be seen in the plot in Fig. A.5. In this case we have a volume plot because of the three parameter space. We can see the absolute minimum as the hyperbolic surface. It slightly wiggles vertically with a 2π phase because of the sin behavior of the loss function with ϕ.

No new behavior of the algorithm was found in this case. Anyway, we can see again the big dependence of the algorithm on α in Fig. A.13 and the stability of ϕ when the guess is $\phi = \phi_T = 1$ in Fig. A.14.

Chapter 5

Conclusions

In my thesis work, I tested a PINN-based method on GW waveforms. The method could be applied to GW waveforms obtained by burst pipelines like cWB.

To do so, I analyzed in detail the algorithm developed by Keith et al. (2021a). To test its robustness, I introduced different levels of Gaussian noise and whistle glitches. The algorithm is robust in this sense. However, analysing it in detail, I discovered several hidden constraints which dramatically reduce the parameter space spanned by model parameters. Moreover, their approach could not be implemented directly on the interferometers' raw data.

For this reason, I recoded the algorithm from scratch, changing the coding environment from JULIA with the SCIML library to the PYTORCH package in PYTHON. In my newly developed code, the phenomenological Newtonian evolution for GW frequency is implemented as (4.28). I selected three parameters of interest: the chirp mass $\mathscr{M}$, the GW signal phase ϕ and the frequency exponent α, for which the target value is 11/3 in the Newtonian approximation. This is performed thanks to an Hybrid version of PINNs (HPINNs), as developed by Nascimento et al. (2020). A Recurrent Neural Network (RNN) with a 4^{th} order Runge-Kutta method is used to solve the differential equation that describes the frequency evolution with time. The resulting frequency function is necessary to calculate the predicted solution for the strain amplitude in the Newtonian approximation. The computed strain is compared with the training waveform dataset and the cost function is evaluated thanks to (4.30). The inferred model parameters correspond to those that minimize the cost function, and this is done by training the PINN.

Although this is only an initial test version, the algorithm has performed well in 1-, 2- and 3-dimensional parameter spaces. $\mathscr{M}$, ϕ and α could be inferred as in Tab. A.1 from a training dataset of numerically generated data points.

The main issue of this algorithm lies in the modified cost function. Since we are trying to learn the frequency evolution of an oscillating function, the cost function is characterized by many local minima. Each minimum corresponds to a dominant frequency which closely matches the actual one. The local minima are reached when the amplitude function is nearly in-phase with the initial and the terminal part of the dataset, although not with the central part of it. Here, I remark that a full 3D analysis must still be completed with the aim of detecting near-minima which are actually saddle points. Looking ahead to a burst implementation, in particular for the cWB pipeline, this study is a step forword in the path to the introduction of parameter inference of CBC sources, and it is important to remark that the cWB output representation may also help avoiding local minima. This could be done by using the time-frequency map in addition to the reconstructed waveform for the definition of the cost function.

The analysis reported here was performed with a maximum of three parameters only. Expanding the parameter space can bring new unexpected opportunities and difficulties. Furthermore, the RMSprop use as optimization algorithm brings a stochasticity inside the algorithm. This means that a statistical analysis must be performed in order to put error bars on the inferred parameter values. Finally, I note that yet another difference with the work of Keith et al. (2021a) is that the latter is written in the SciML environment, built precisely with the purpose to handle UDE problems and which utilizes Automatic Differentiation (AD), thanks to which derivatives are calculated in a more accurate and fast way. To obtain the same level of performance, AD needs to be implemented on PyTorch as well.

In summary, this work is only the first step of a more extensive effort to implement a PINN-based parameter inference module following a burst analysis. Many different aspects of the algorithm can be tuned to approach the accuracy levels of CBC pipelines.

Appendix A

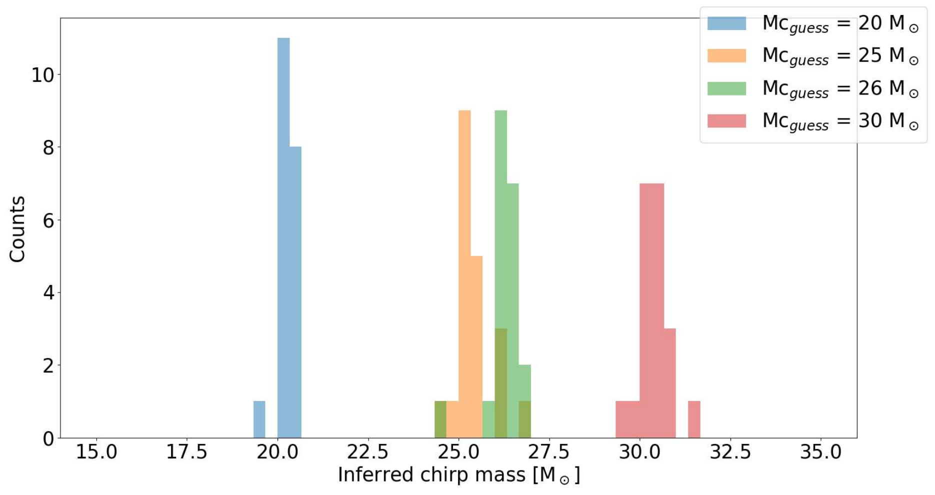
Extended results

ID	$\mathscr{M}$ [M$_\odot$]		ϕ [rad]		α [ ]		*Loss*	*Note*
	Guess	*Inferred*	*Guess*	*Inferred*	*Guess*	*Inferred*		
1	20.000	20.202	-	-	-	-	7.663	
2	25.000	26.107	-	-	-	-	0.001(608)	
3	26.110	26.121	-	-	-	-	0.000(451)	RG - mL
4	30.000	30.728	-	-	-	-	11.642	
5	-	-	0.100	0.010	-	-	4.146	
6	-	-	0.900	0.934	-	-	0.030	
7	-	-	1.000	1.000	-	-	0.000(591)	RG - mL
8	-	-	π	3.071	-	-	16.233	
9	-	-	-	-	3.000	3.504	12.747	
10	-	-	-	-	3.333	3.508	12.320	
11	-	-	-	-	3.667	3.749	8.840	RG - mL
12	-	-	-	-	4.000	NaN	NaN	Div
13	20.000	20.468	0.100	0.162	-	-	8.375	
14	20.000	20.298	0.900	0.689	-	-	7.926	
15	20.000	20.202	1.000	1.000	-	-	7.663	
16	20.000	19.515	π	3.102	-	-	7.277	
17	25.000	26.797	0.100	-0.103	-	-	2.245	
18	25.000	26.136	0.900	0.960	-	-	0.007(323)	
19	25.000	26.108	1.000	1.000	-	-	0.001(608)	
20	25.000	24.504	π	3.170	-	-	5.634	
21	26.110	26.883	0.100	0.088	-	-	1.361	
22	26.110	26.179	0.900	0.928	-	-	0.012	
23	26.110	26.121	1.000	1.000	-	-	0.000(451)	RG - mL
24	26.110	24.520	π	3.370	-	-	6.638	
25	30.000	31.385	0.100	-0.081	-	-	10.694	
26	30.000	30.922	0.900	0.662	-	-	11.164	
27	30.000	30.728	1.000	1.000	-	-	11.642	
28	30.000	29.563	π	2.970	-	-	14.064	
29	20.000	20.474	-	-	3.000	3.514	9.650	
30	20.000	20.427	-	-	3.333	3.855	6.448	
31	20.000	20.120	-	-	3.667	3.872	6.316	mL
32	20.000	NaN	-	-	4.000	NaN	NaN	Div
33	25.000	25.510	-	-	3.000	3.606	10.036	
34	25.000	25.178	-	-	3.333	3.612	9.819	
35	25.000	25.051	-	-	3.667	3.759	8.334	
36	25.000	NaN	-	-	4.000	NaN	NaN	Div
37	26.110	26.601	-	-	3.000	3.552	12.151	
38	26.110	26.353	-	-	3.333	3.602	10.623	
39	26.110	26.152	-	-	3.667	3.739	8.908	RG
40	26.110	NaN	-	-	4.000	NaN	NaN	Div

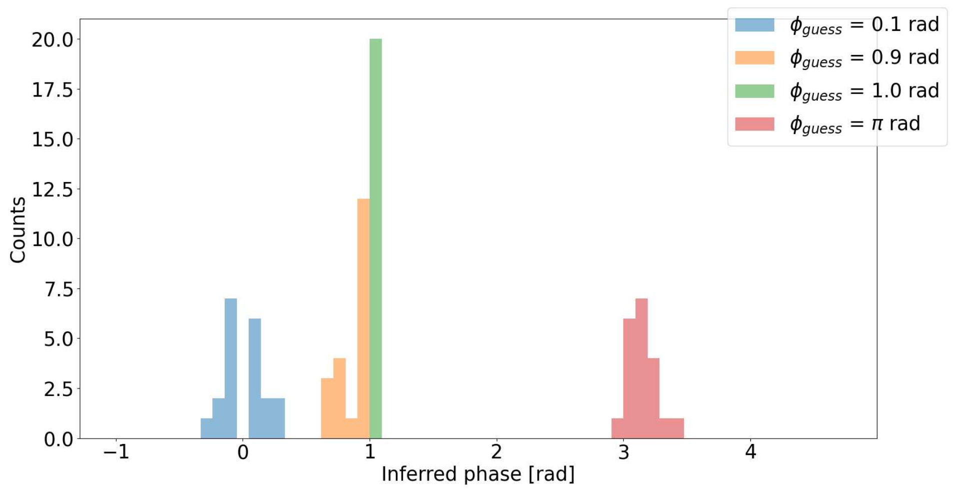
41	30.000	30.426	-	-	3.000	3.456	17.657	
42	30.000	30.384	-	-	3.333	3.667	11.590	
43	30.000	30.024	-	-	3.667	3.746	11.163	
44	30.000	NaN	-	-	4.000	NaN	NaN	Div
45	-	-	0.100	-0.274	3.000	3.570	12.619	
46	-	-	0.100	0.278	3.333	3.507	12.838	
47	-	-	0.100	-0.106	3.667	3.751	7.914	mL
48	-	-	0.100	NaN	4.000	NaN	NaN	Div
49	-	-	0.900	0.896	3.000	3.517	12.441	
50	-	-	0.900	0.758	3.333	3.597	10.752	
51	-	-	0.900	0.808	3.667	3.607	10.704	
52	-	-	0.900	NaN	4.000	NaN	NaN	Div
53	-	-	1.000	1.000	3.000	3.504	12.747	
54	-	-	1.000	1.000	3.333	3.508	12.320	
55	-	-	1.000	1.000	3.667	3.749	8.840	RG
56	-	-	1.000	NaN	4.000	NaN	NaN	Div
57	-	-	π	3.083	3.000	3.533	11.696	
58	-	-	π	3.058	3.333	3.591	10.038	
59	-	-	π	3.137	3.667	3.590	10.141	
60	-	-	π	NaN	4.000	NaN	NaN	Div
61	20.000	20.482	0.100	-0.165	3.000	3.503	9.648	
62	20.000	20.281	0.100	-0.092	3.333	3.621	8.816	
63	20.000	20.163	0.100	0.130	3.667	3.871	6.037	mL
64	20.000	NaN	0.100	NaN	4.000	NaN	NaN	Div
65	20.000	20.469	0.900	0.916	3.000	3.512	9.656	
66	20.000	20.191	0.900	0.956	3.333	3.594	8.724	
67	20.000	20.112	0.900	0.946	3.667	3.863	6.336	
68	20.000	NaN	0.900	NaN	4.000	NaN	NaN	Div
69	20.000	20.474	1.000	1.000	3.000	3.514	9.650	
70	20.000	20.427	1.000	1.000	3.333	3.855	6.448	
71	20.000	20.120	1.000	1.000	3.667	3.872	6.316	
72	20.000	NaN	1.000	NaN	4.000	NaN	NaN	Div
73	20.000	20.456	π	3.156	3.000	3.493	9.706	
74	20.000	20.255	π	3.063	3.333	3.664	7.806	
75	20.000	20.171	π	3.112	3.667	3.917	7.077	
76	20.000	NaN	π	NaN	4.000	NaN	NaN	Div
77	25.000	25.620	0.100	0.162	3.000	3.621	11.439	
78	25.000	25.149	0.100	-0.194	3.333	3.528	12.359	
79	25.000	25.122	0.100	-0.114	3.667	3.779	7.481	
80	25.000	NaN	0.100	NaN	4.000	NaN	NaN	Div
81	25.000	25.388	0.900	0.733	3.000	3.445	13.419	
82	25.000	25.219	0.900	0.682	3.333	3.613	10.294	
83	25.000	25.051	0.900	0.967	3.667	3.759	8.319	
84	25.000	NaN	0.900	NaN	4.000	NaN	NaN	Div
85	25.000	25.510	1.000	1.000	3.000	3.606	10.036	
86	25.000	25.178	1.000	1.000	3.333	3.612	9.819	
87	25.000	25.051	1.000	1.000	3.667	3.759	8.334	
88	25.000	NaN	1.000	NaN	4.000	NaN	NaN	Div
89	25.000	25.523	π	3.268	3.000	3.588	10.052	
90	25.000	25.222	π	3.161	3.333	3.744	9.715	
91	25.000	24.825	π	3.254	3.667	3.555	10.756	
92	25.000	NaN	π	NaN	4.000	NaN	NaN	Div
93	26.110	26.562	0.100	-0.107	3.000	3.507	13.954	
94	26.110	26.276	0.100	0.285	3.333	3.550	12.387	
95	26.110	26.208	0.100	-0.111	3.667	3.749	7.962	
96	26.110	NaN	0.100	NaN	4.000	NaN	NaN	Div
97	26.110	26.687	0.900	0.909	3.000	3.587	11.043	
98	26.110	26.483	0.900	0.731	3.333	3.735	8.789	
99	26.110	26.167	0.900	0.905	3.667	3.750	8.743	

100	26.110	NaN	0.900	NaN	4.000	NaN	NaN	Div
101	26.110	26.601	1.000	1.000	3.000	3.552	12.151	
102	26.110	26.352	1.000	1.000	3.333	3.602	10.623	
103	26.110	26.152	1.000	1.000	3.667	3.734	8.908	RG
104	26.110	NaN	1.000	NaN	4.000	NaN	NaN	Div
105	26.110	26.592	π	3.169	3.000	3.541	12.210	
106	26.110	26.278	π	3.198	3.333	3.577	10.390	
107	26.110	25.993	π	3.032	3.667	3.593	9.914	
108	26.110	NaN	π	NaN	4.000	NaN	NaN	Div
109	30.000	30.496	0.100	0.067	3.000	3.498	16.836	
110	30.000	30.180	0.100	0.134	3.333	3.683	10.026	
111	30.000	30.076	0.100	0.120	3.667	3.755	10.441	
112	30.000	NaN	0.100	NaN	4.000	NaN	NaN	Div
113	30.000	30.364	0.900	0.943	3.000	3.412	19.406	
114	30.000	30.119	0.900	0.999	3.333	3.497	15.813	
115	30.000	30.039	0.900	0.938	3.667	3.756	11.067	
116	30.000	NaN	0.900	NaN	4.000	NaN	NaN	Div
117	30.000	30.426	1.000	1.000	3.000	3.456	17.657	
118	30.000	30.384	1.000	1.000	3.333	3.667	11.589	
119	30.000	30.024	1.000	1.000	3.667	3.746	11.163	
120	30.000	NaN	1.000	NaN	4.000	NaN	NaN	Div
121	30.000	30.435	π	3.392	3.000	3.510	14.600	
122	30.000	30.215	π	3.092	3.333	3.730	13.070	
123	30.000	30.708	π	3.227	3.667	3.659	14.659	
124	30.000	NaN	π	NaN	4.000	NaN	NaN	Div

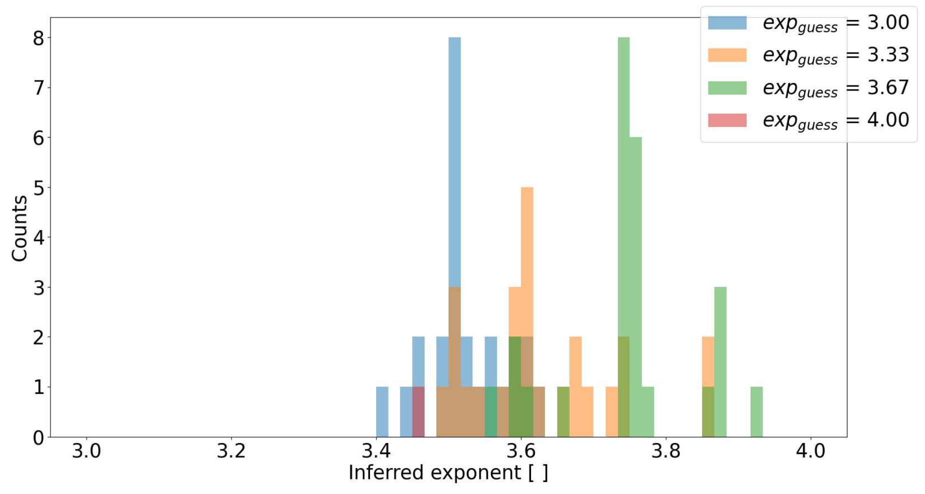
Table A.1: *Parameter inference results for all the runs (RG = Right Guess*; mL = *minimum Loss*; Div = *Divergence).*



(a)



(b)



(c)

Figure A.1: *Inference dependence on the initial guess.*

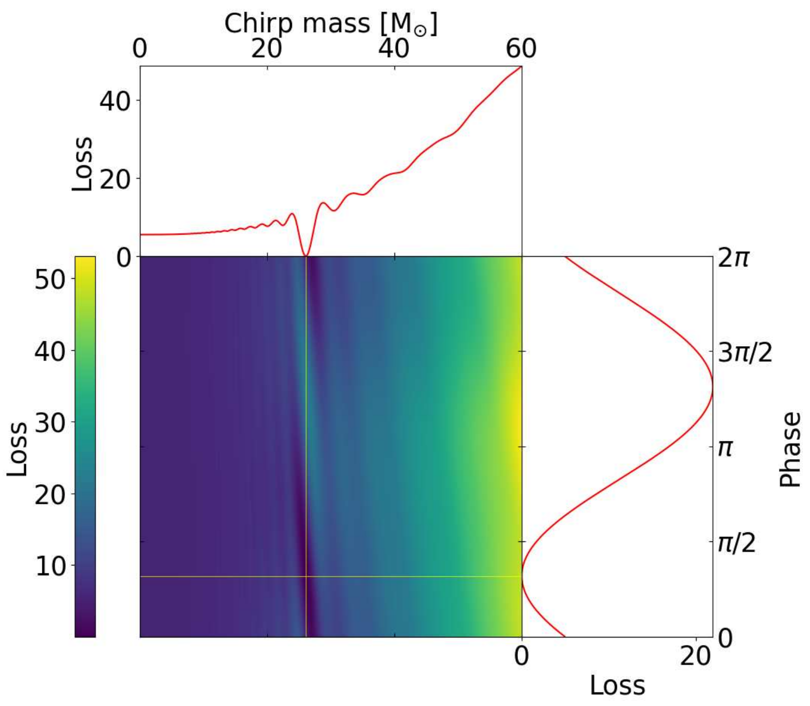


Figure A.2: *Loss dependence on the chirp mass $\mathscr{M}$, on the phase ϕ and on both of them. The two side plot are the loss values taken when the other parameter is fixed at target value, i.e. are the loss values at the yellow lines.*

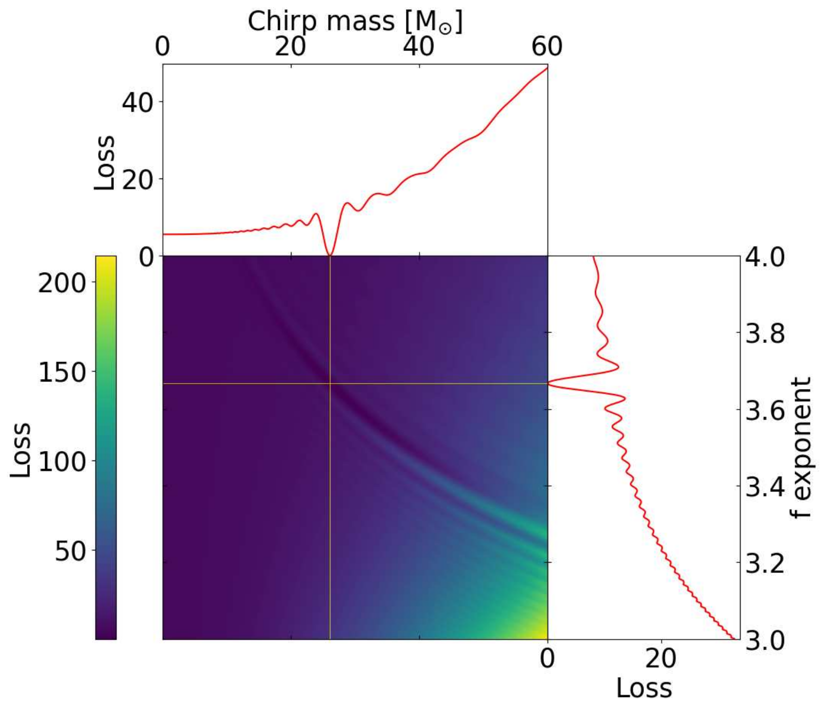


Figure A.3: *Loss dependence on the chirp mass $\mathscr{M}$, on the f exponent α and on both of them. The two side plot are the loss values taken when the other parameter is fixed at target value, i.e. are the loss values at the yellow lines.*

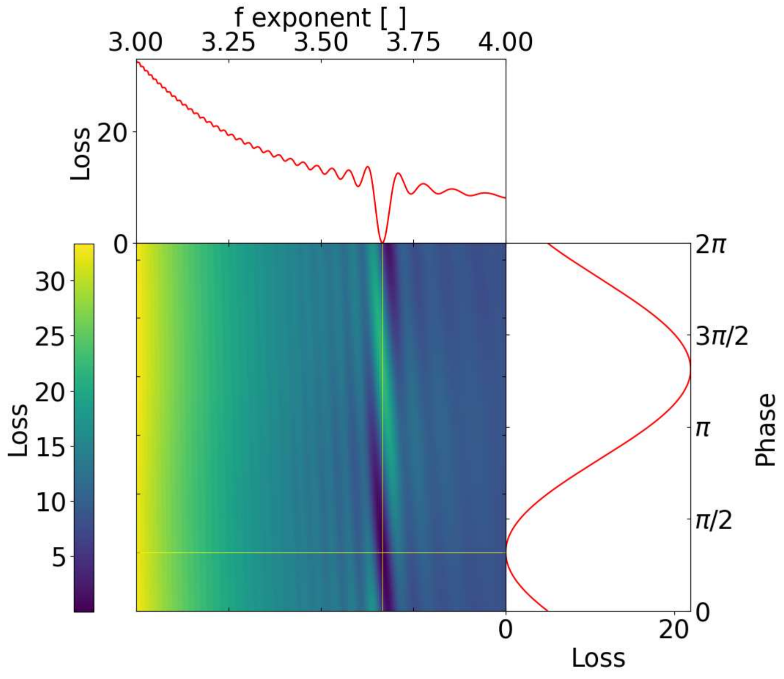


Figure A.4: *Loss dependence on the f exponent $\mathscr{M}$, on the phase ϕ and on both of them. The two side plot are the loss values taken when the other parameter is fixed at target value, i.e. are the loss values at the yellow lines.*

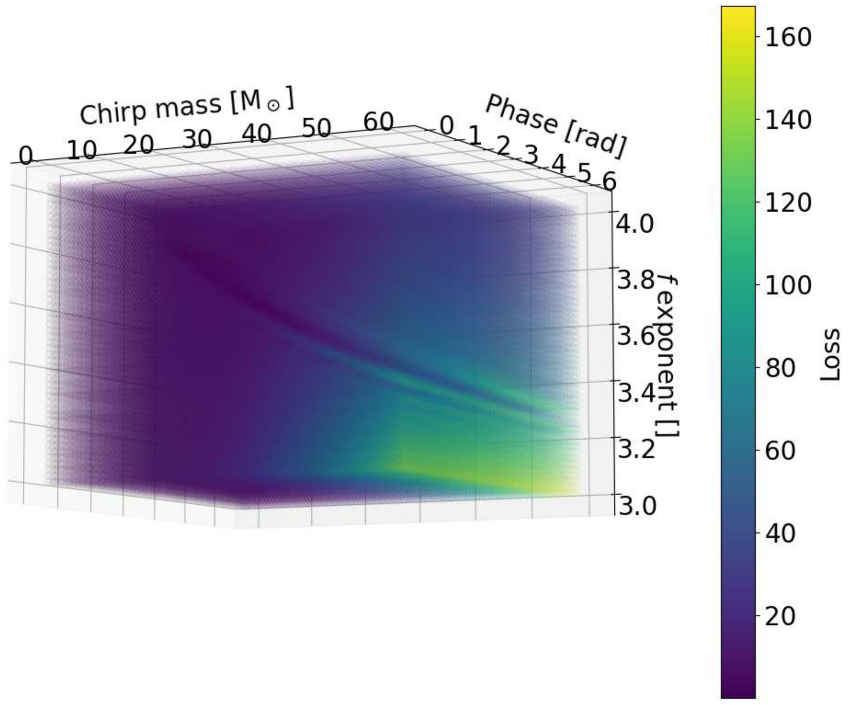
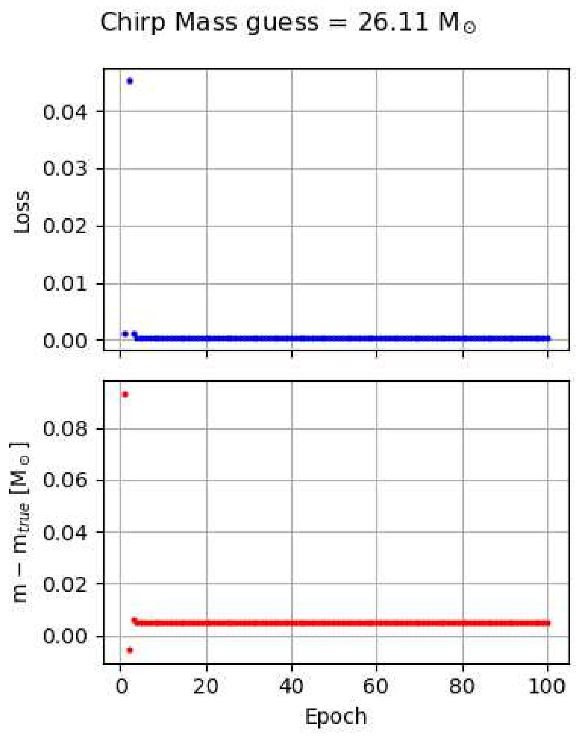
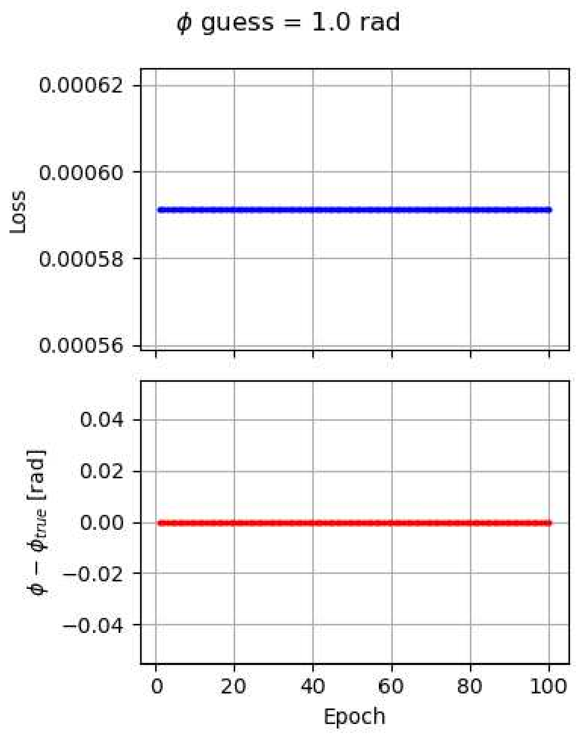


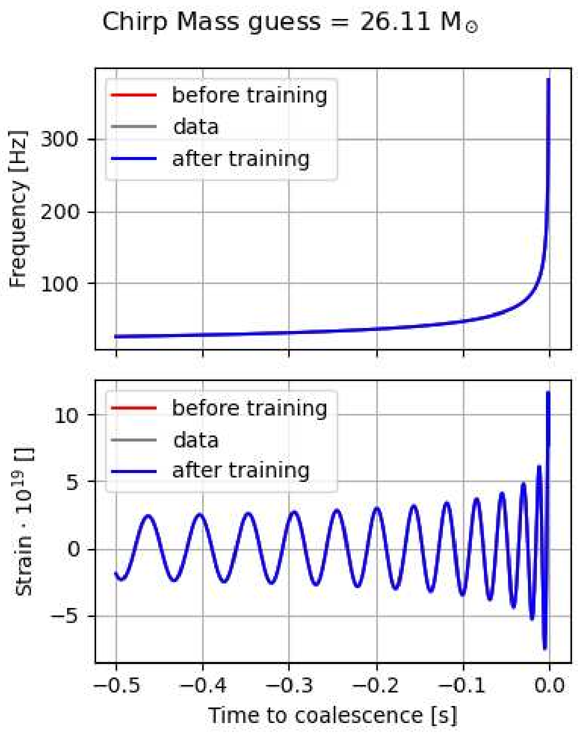
Figure A.5: *Loss volume dependent on the chirp mass $\mathscr{M}$, on the phase ϕ and on the f exponent α.*



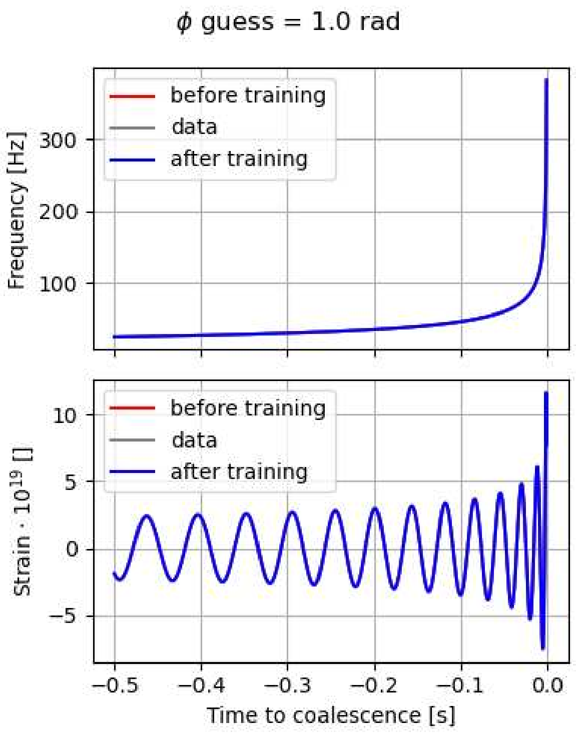
(a)



(b)

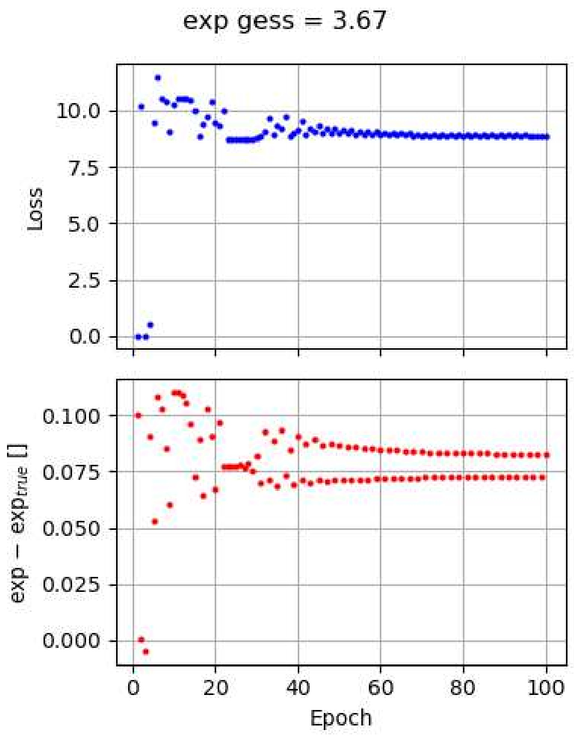


(c)

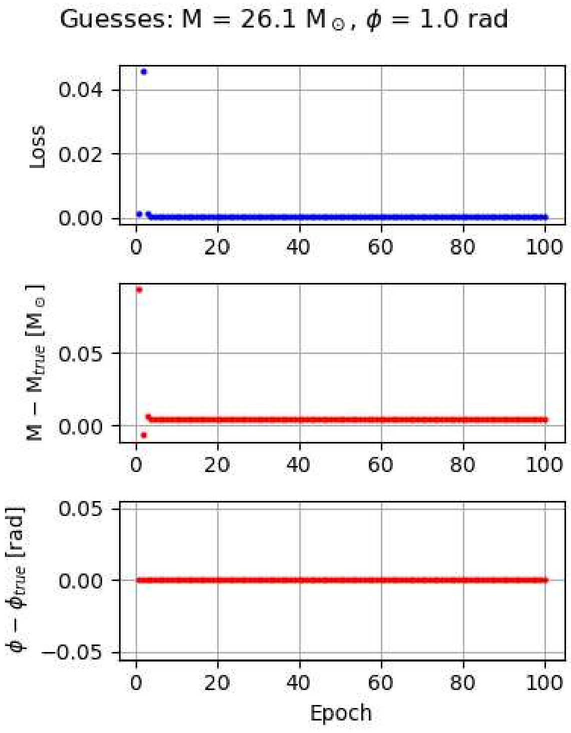


(d)

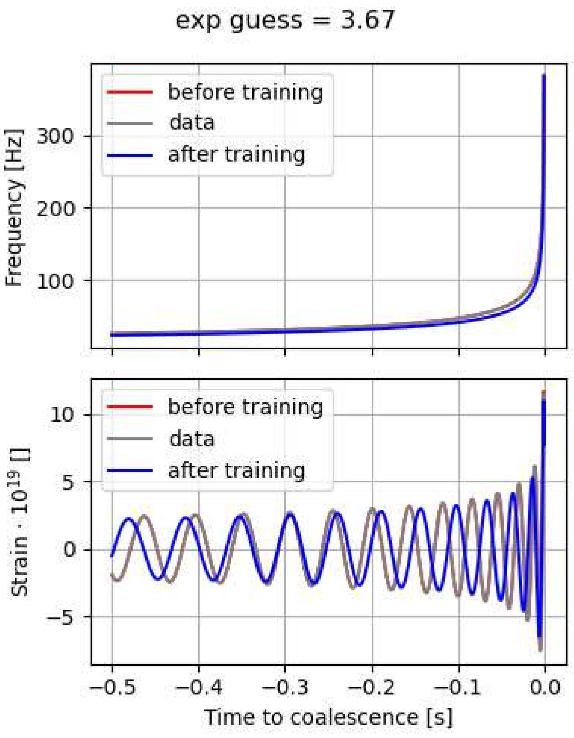
Figure A.6: *RGs. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: RG run for $\mathscr{M}$ (ID 3);* (b) *and* (d)*: RG run for ϕ (ID 7).*



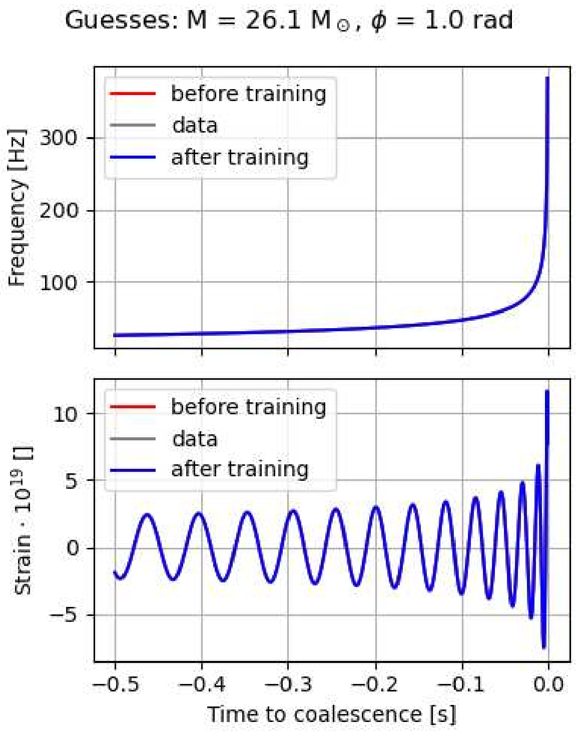
(a)



(b)

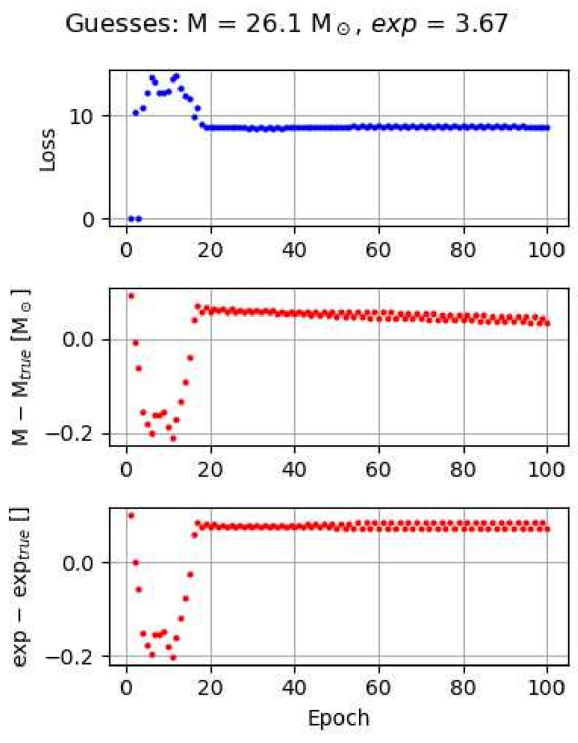


(c)

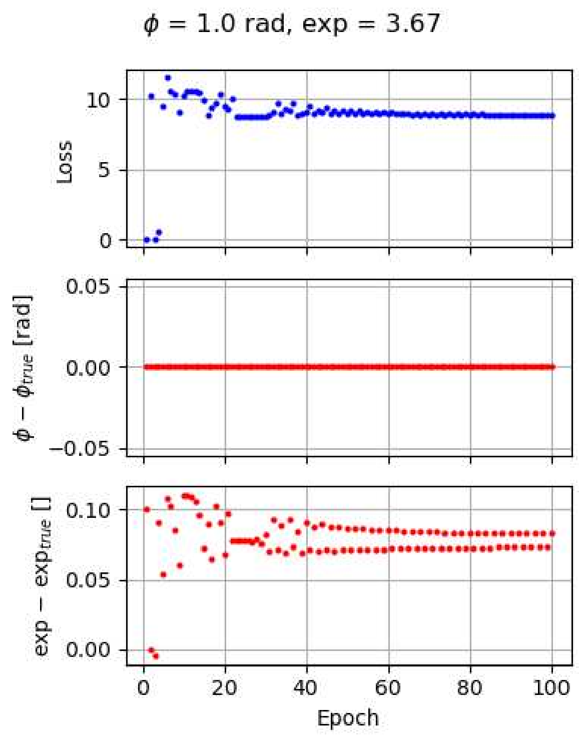


(d)

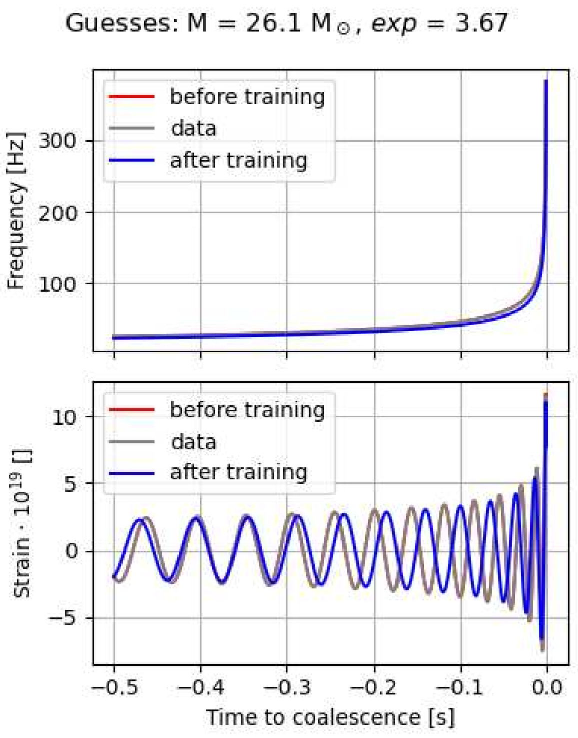
Figure A.7: *RGs.From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: RG run for f exponent (ID 11);* (b) *and* (d)*: RG run for the couple $\mathscr{M}$ and ϕ (ID 23).*



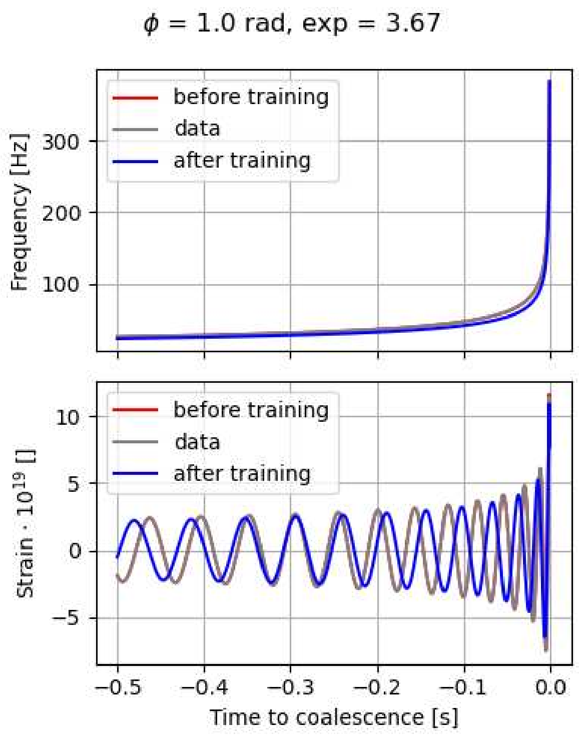
(a)



(b)

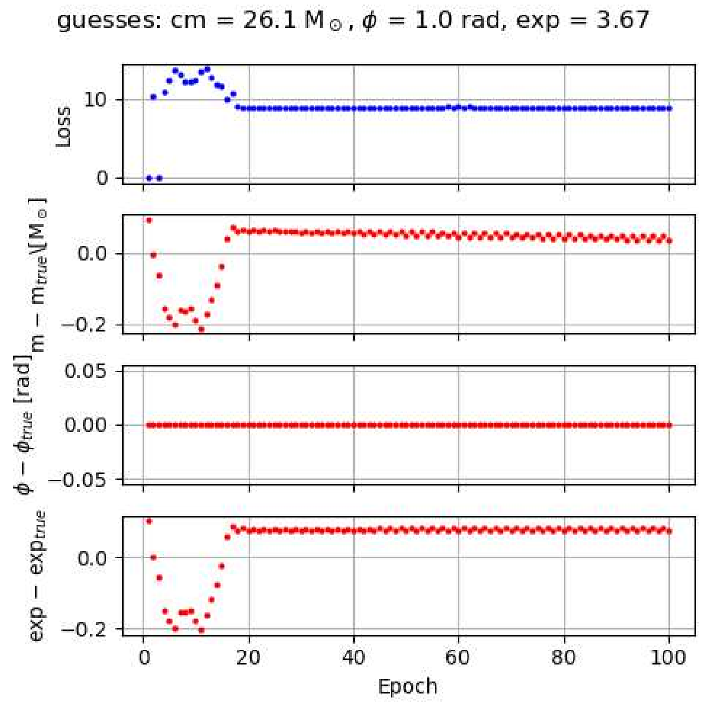


(c)

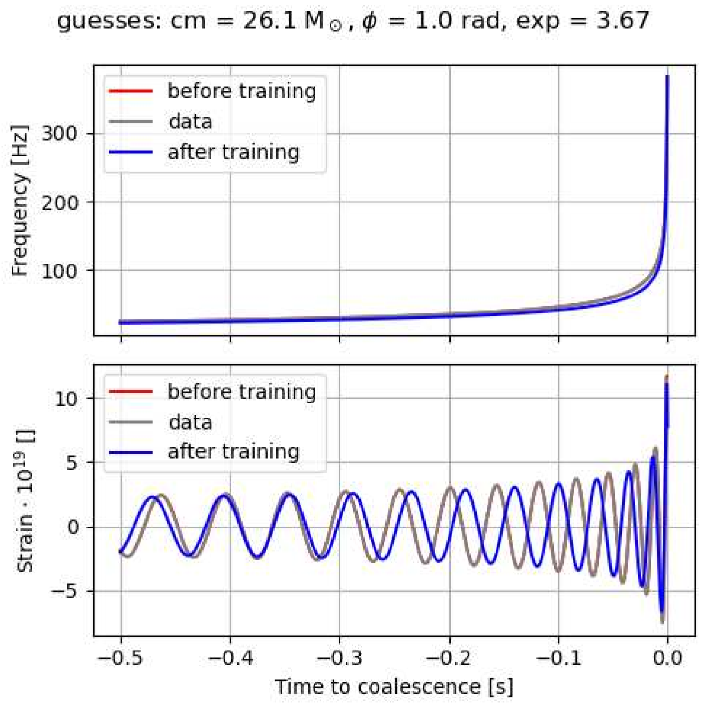


(d)

Figure A.8: *RGs. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: RG run for the couple $\mathscr{M}$ and f exponent (ID 39);* (b) *and* (d)*: RG run for the couple ϕ and f exponent (ID 55).*

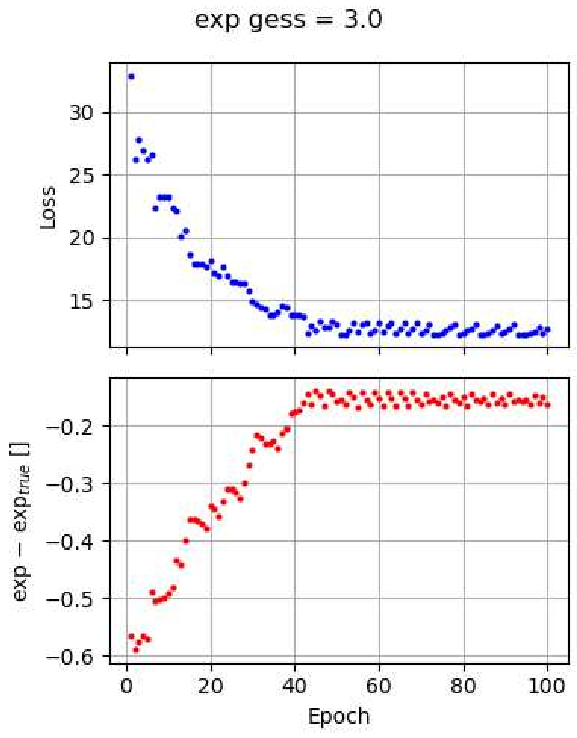


(a)

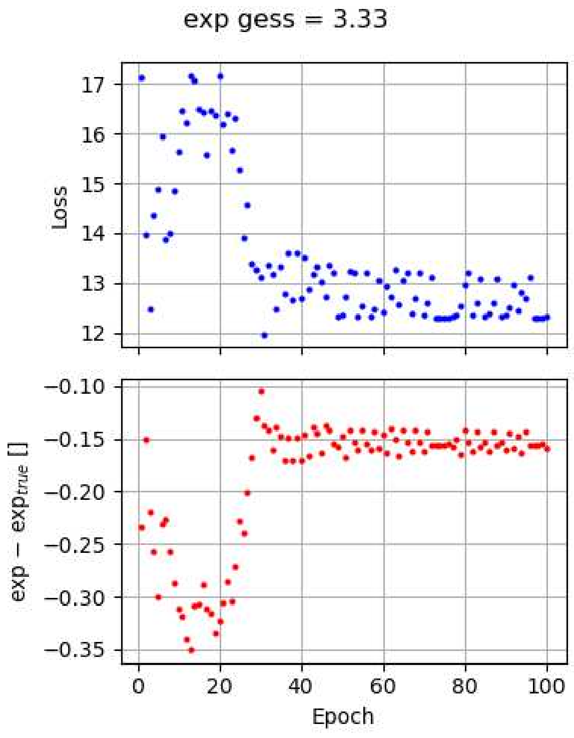


(b)

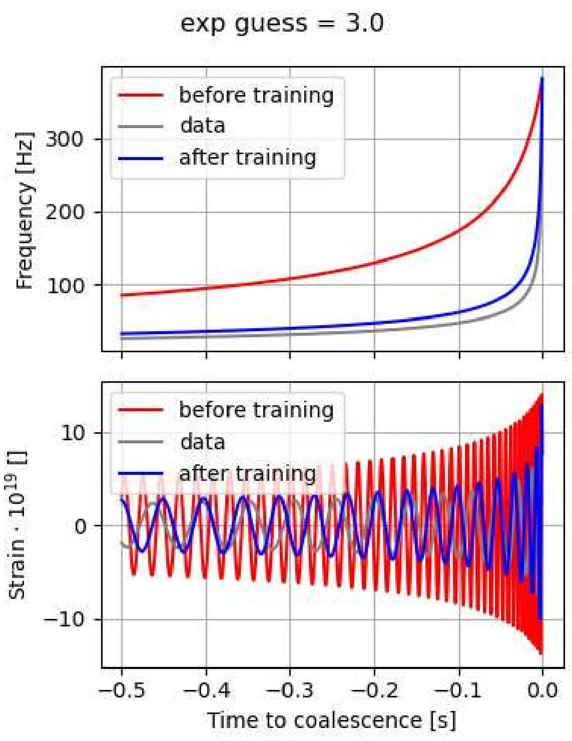
Figure A.9: *RGs. From the top to the bottom: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (b)*: RG run for the tern $\mathscr{M}$, phi and f exponent (ID 103).*



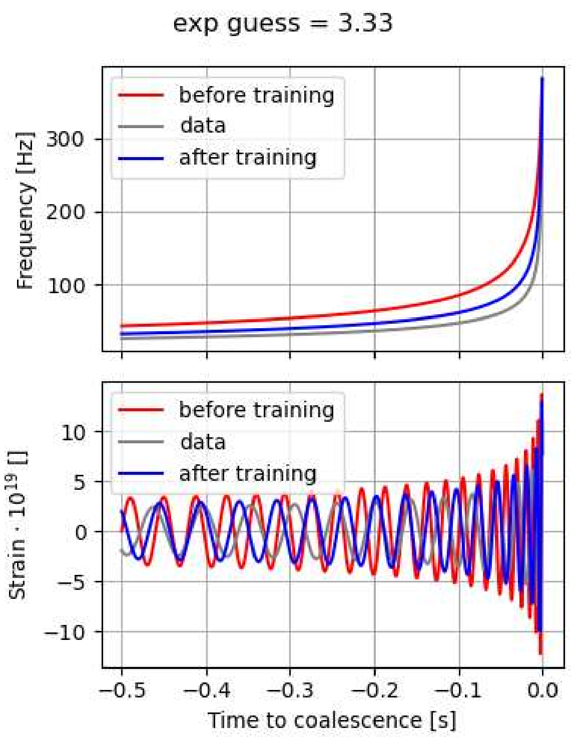
(a)



(b)

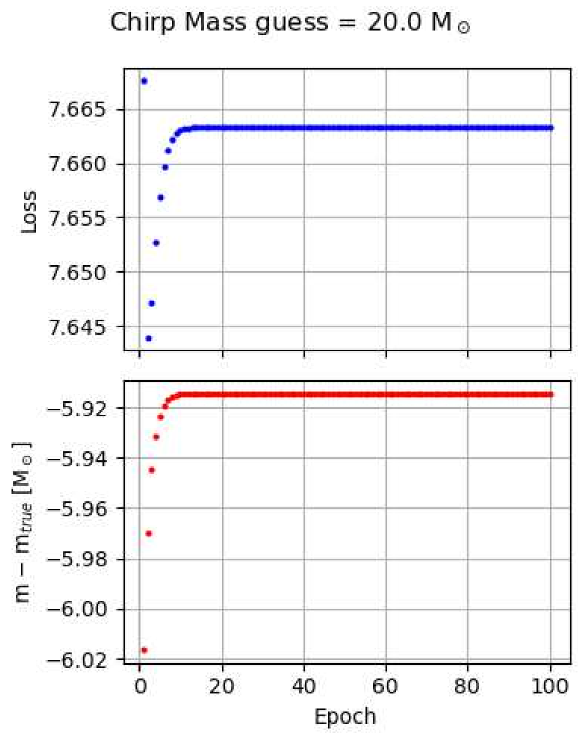


(c)

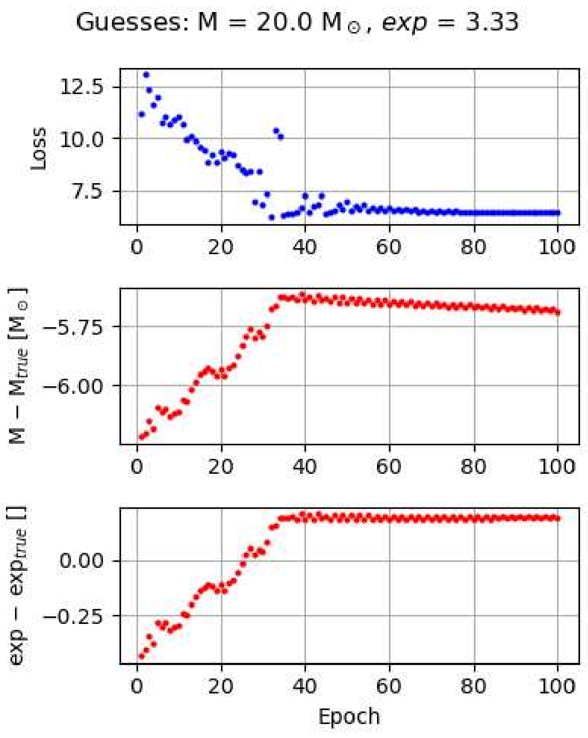


(d)

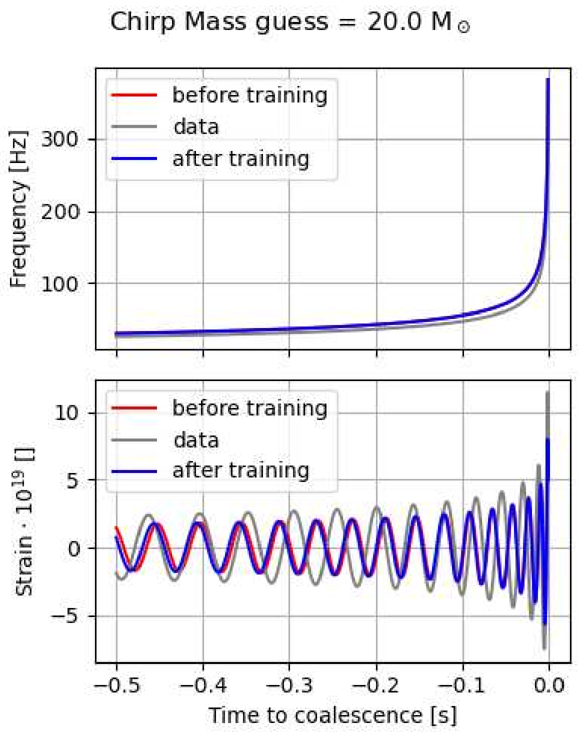
Figure A.10: *α dependence. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: ID 9;* (b) *and* (d)*: ID 10.*



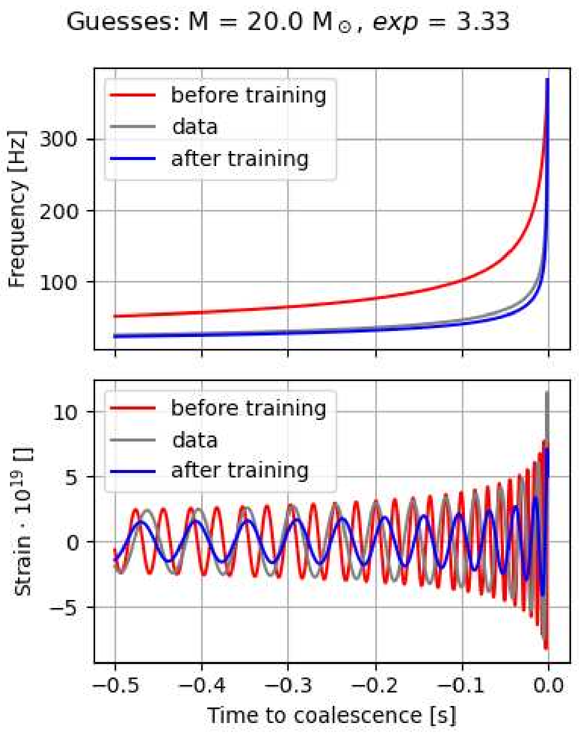
(a)



(b)

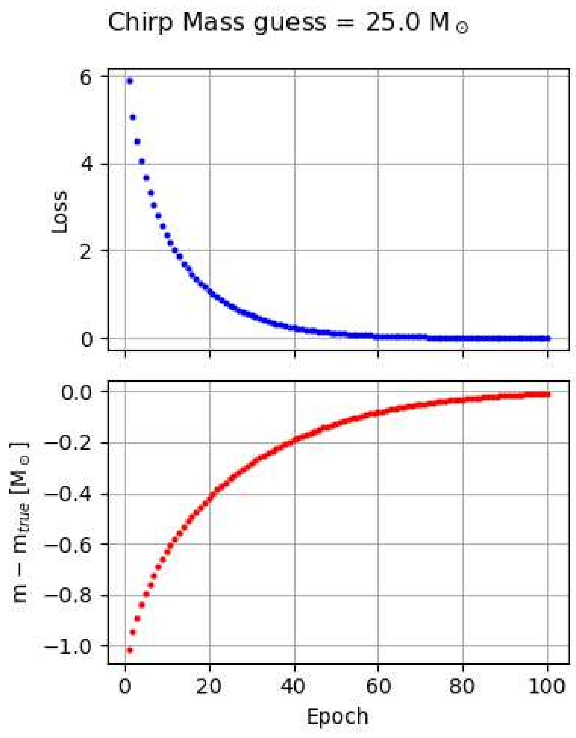


(c)

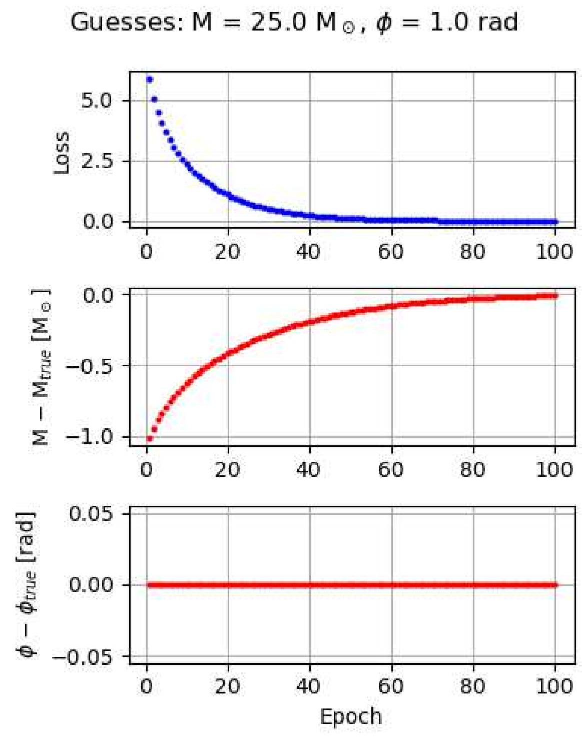


(d)

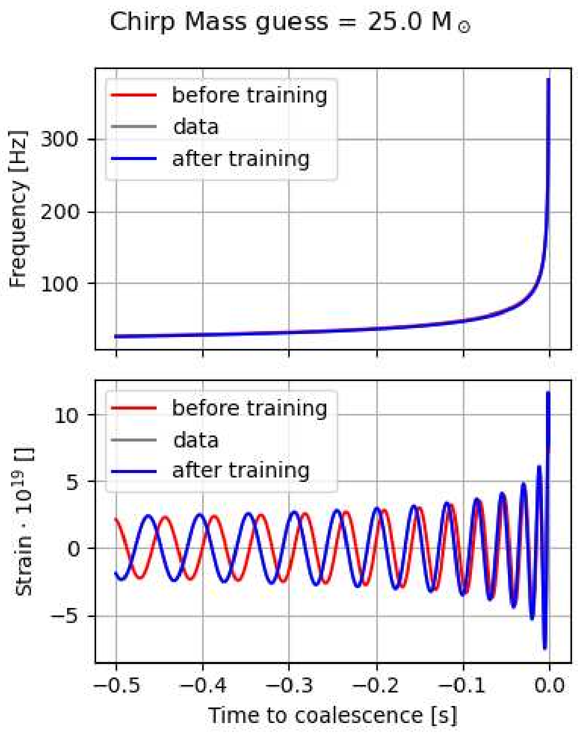
Figure A.11: *α helps exit from the minima. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: ID 1;* (b) *and* (d)*: ID 30.*



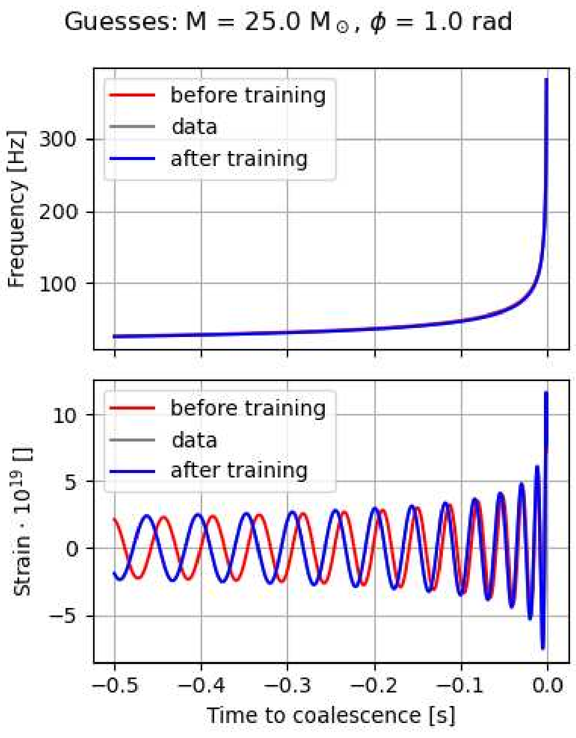
(a)



(b)

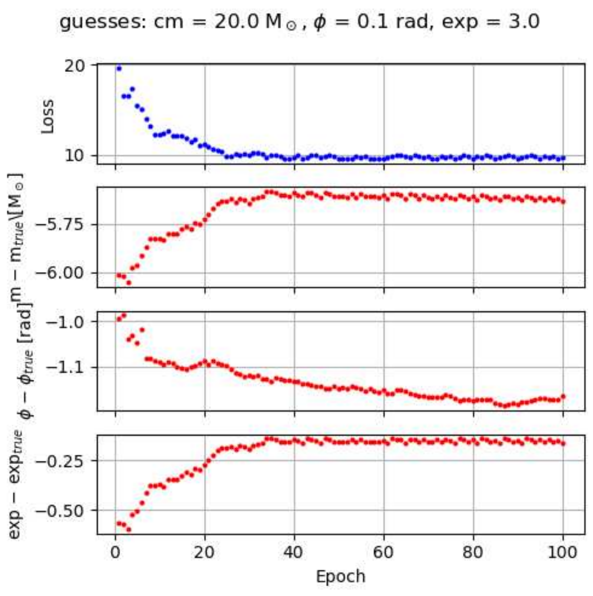


(c)

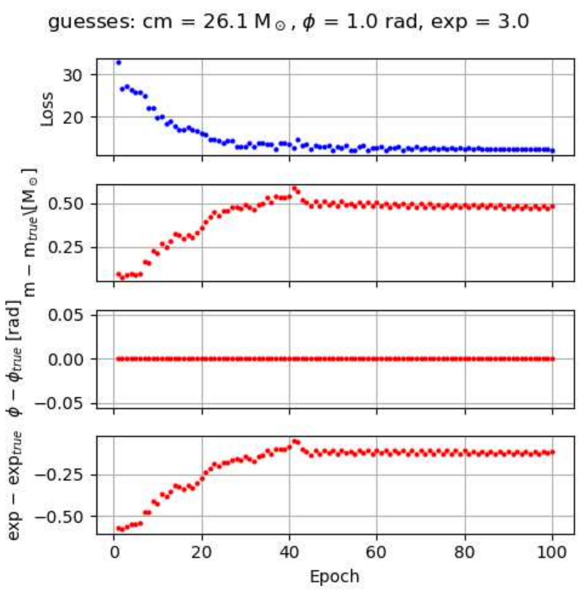


(d)

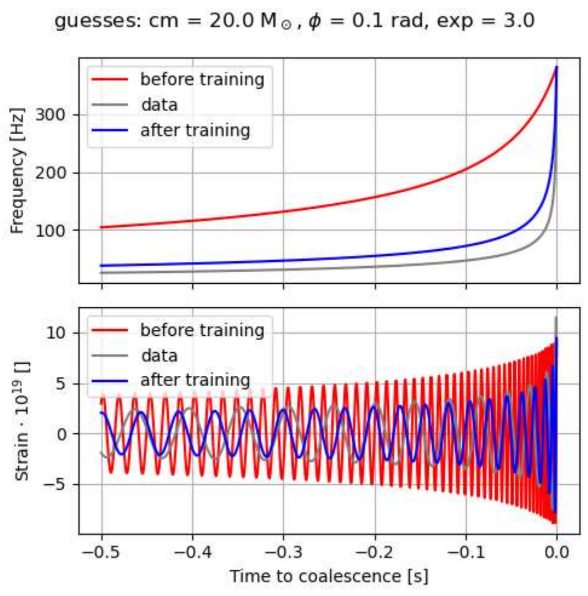
Figure A.12: *ϕ independence when $\phi = \phi_T = 1$. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: ID 2;* (b) *and* (d)*: ID 19.*



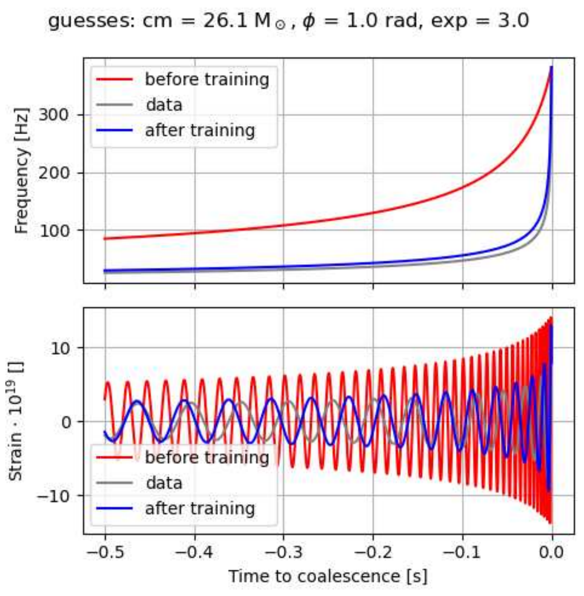
(a)



(b)

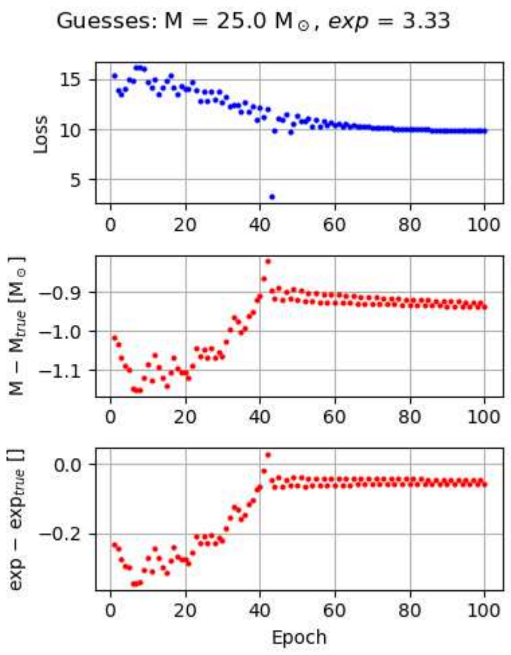


(c)

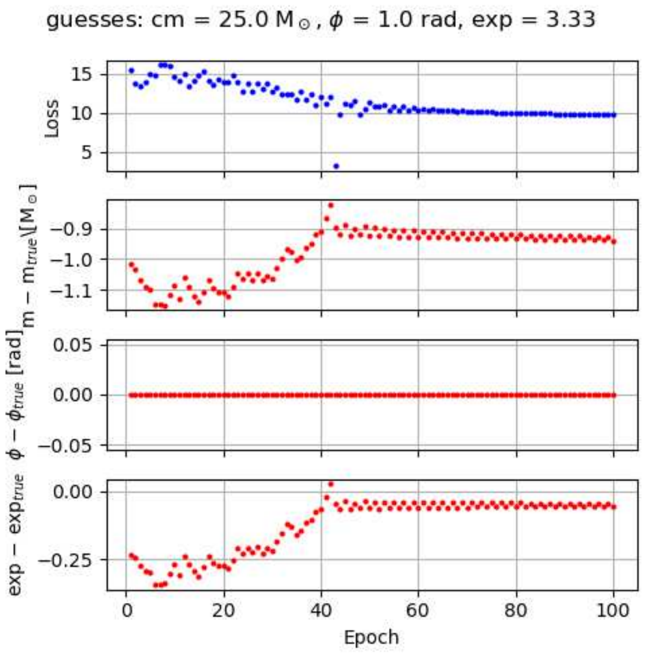


(d)

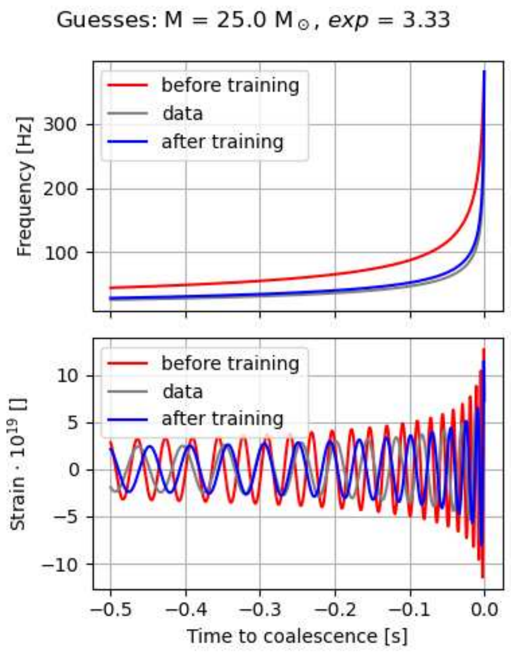
Figure A.13: *α dependence. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: ID 61;* (b) *and* (d)*: ID 101.*



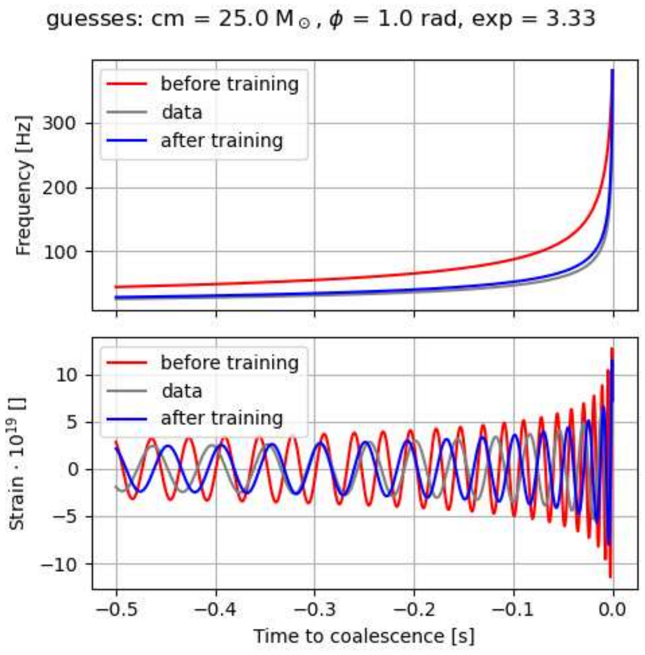
(a)



(b)



(c)



(d)

Figure A.14: *ϕ independence when $\phi = \phi_T = 1$. From the top to the bottom of every column: loss with respect to the epoch number, parameter errors with respect to the epoch number, frequency learning, waveform learning.* (a) *and* (c)*: ID 34;* (b) *and* (d)*: ID 86.*

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning.

Abbott, B. et al. (2019). GWTC-1: A gravitational-wave transient catalog of compact binary mergers observed by LIGO and virgo during the first and second observing runs. *Physical Review X*, 9(3).

Abbott, B. P. et al. (2016a). Characterization of transient noise in advanced ligo relevant to gravitational wave signal gw150914. *Classical and Quantum Gravity*, 33(13):134001.

Abbott, B. P. et al. (2016b). Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102.

Abbott, R. et al. (2021a). GWTC-2: Compact binary coalescences observed by LIGO and virgo during the first half of the third observing run. *Physical Review X*, 11(2).

Abbott, R. et al. (2021b). Gwtc-3: Compact binary coalescences observed by ligo and virgo during the second part of the third observing run.

Abbott, R. et al. (2021c). Upper limits on the isotropic gravitational-wave background from advanced LIGO and advanced virgo's third observing run. *Physical Review D*, 104(2).

Abbott, R. et al. (2022). All-sky search for continuous gravitational waves from isolated neutron stars using advanced LIGO and advanced virgo o3 data. *Physical Review D*, 106(10).

Abe, H. et al. (2022). The current status and future prospects of kagra, the large-scale cryogenic gravitational wave telescope built in the kamioka underground. *Galaxies*, 10(3).

Agazie, G. et al. (2023). The nanograv 15-year data set: Search for anisotropy in the gravitational-wave background.

Amaro-Seoane, P. et al. (2017). Laser interferometer space antenna.

Ashton, G., Hübner, M., Lasky, P. D., Talbot, C., Ackley, K., Biscoveanu, S., Chu, Q., Divakarla, A., Easter, P. J., Goncharov, B., Vivanco, F. H., Harms, J., Lower, M. E., Meadors, G. D., Melchor, D., Payne, E., Pitkin, M. D., Powell, J., Sarin, N., Smith, R. J. E., and Thrane, E. (2019). Bilby: A user-friendly bayesian inference library for gravitational-wave astronomy. *The Astrophysical Journal Supplement Series*, 241(2):27.

Bacon, P., Trovato, A., and Bejger, M. (2023). Denoising gravitational-wave signals from binary black holes with a dilated convolutional autoencoder. *Machine Learning: Science and Technology*, 4(3):035024.

Banik, N., Tan, J. C., and Monaco, P. (2018). The formation of supermassive black holes from population III.1 seeds. i. cosmic formation histories and clustering properties. *Monthly Notices of the Royal Astronomical Society*, 483(3):3592–3606.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43.

Blanchet, L., Iyer, B. R., Will, C. M., and Wiseman, A. G. (1996). Gravitational waveforms from inspiralling compact binaries to second-post-newtonian order. *Classical and Quantum Gravity*, 13(4):575–584.

Boyle, M., Hemberger, D., Iozzo, D. A. B., Lovelace, G., Ossokine, S., Pfeiffer, H. P., Scheel, M. A., Stein, L. C., Woodford, C. J., Zimmerman, A. B., Afshari, N., Barkett, K., Blackman, J., Chatziioannou, K., Chu, T., Demos, N., Deppe, N., Field, S. E., Fischer, N. L., Foley, E., Fong, H., Garcia, A., Giesler, M., Hebert, F., Hinder, I., Katebi, R., Khan, H., Kidder, L. E., Kumar, P., Kuper, K., Lim, H., Okounkova, M., Ramirez, T., Rodriguez, S., Rüter, H. R., Schmidt, P., Szilagyi, B., Teukolsky, S. A., Varma, V., and Walker, M. (2019). The SXS collaboration catalog of binary black hole simulations. *Classical and Quantum Gravity*, 36(19):195006.

Branchesi, M. et al. (2023). Science with the einstein telescope: a comparison of different designs.

Cahillane, C. and Mansell, G. (2022). Review of the advanced LIGO gravitational wave observatories leading to observing run four. *Galaxies*, 10(1):36.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., and Zdeborová , L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4).

Chatterji, S., Blackburn, L., Martin, G., and Katsavounidis, E. (2004). Multiresolution techniques for the detection of gravitational-wave bursts. *Classical and Quantum Gravity*, 21(20):S1809–S1818.

Chaudhary, K. (2020). Understanding audio data, fourier transform, fft and spectrogram features for a speech recognition system. .

Cuoco, E., Powell, J., Cavaglià , M., Ackley, K., Bejger, M., Chatterjee, C., Coughlin, M., Coughlin, S., Easter, P., Essick, R., Gabbard, H., Gebhard, T., Ghosh, S., Haegel, L., Iess, A., Keitel, D., Márka, Z., Márka, S., Morawski, F., Nguyen, T., Ormiston, R., Pürrer, M., Razzano, M., Staats, K., Vajente, G., and Williams, D. (2020). Enhancing gravitational-wave science with machine learning. *Machine Learning: Science and Technology*, 2(1):011002.

Davis, P., Rabinowitz, P., and Rheinbolt, W. (2014). *Methods of Numerical Integration*. Computer Science and Applied Mathematics. Elsevier Science.

Dreissigacker, C., Sharma, R., Messenger, C., Zhao, R., and Prix, R. (2019). Deep-learning continuous gravitational waves. *Physical Review D*, 100(4).

Einstein, A. (1916). Näherungsweise Integration der Feldgleichungen der Gravitation. *Sitzungsberichte der K&ouml;niglich Preussischen Akademie der Wissenschaften*, pages 688–696.

Fletcher, R. (2000). *Nonlinear Programming*, chapter 12, pages 277–330. John Wiley & Sons, Ltd.

Foster, J. and Nightingale, J. (2010). *A Short Course in General Relativity.* Springer New York.

Gabbard, H., Williams, M., Hayes, F., and Messenger, C. (2018). Matching matched filtering with deep networks for gravitational-wave astronomy. *Physical Review Letters*, 120(14).

Graves, A. (2014). Generating sequences with recurrent neural networks.

Hinton, G., Srivastava, N., and Swersky, K. (2018). Neural networks for machine learning - lesson 6e.

Kapadia, S., Dent, T., and Canton, T. D. (2017). Classifier for gravitational-wave inspiral signals in nonideal single-detector data. *Physical Review D*, 96(10).

Keith, B., Khadse, A., and Field, S. E. (2021a). Learning orbital dynamics of binary black hole systems from gravitational wave measurements. *Phys. Rev. Res.*, 3:043101.

Keith, B., Khadse, A., and Field, S. E. (2021b). Three examples of learning orbital dynamics of binary black hole systems from gravitational wave measurements with Julia. https://doi.org/10.5281/zenodo.4477649.

Kepler, J. (1609). *Astronomia nova.* .

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Leibniz, G. W. (1684). Nova methodus pro maximis et minimis, itemque tangentibus, quae nec fractas, nec irrationales quantitates moratur, et singulare pro illis calculi genus. *Acta Eruditorum*, pages 467–473.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning.

Lorentz, H. A. and Droste, J. (1937). *The Motion of a System of Bodies under the Influence of their Mutual Attraction, According to Einstein's Theory*, pages 330–355. Springer Netherlands, Dordrecht.

Lynch, R. S. (2015). Pulsar timing arrays. *Journal of Physics: Conference Series*, 610(1):012017.

Maggiore, M. (2007). *Gravitational Waves. Vol. 1: Theory and Experiments.* Oxford University Press.

Mehta, P., Bukov, M., Wang, C.-H., Day, A. G., Richardson, C., Fisher, C. K., and Schwab, D. J. (2019). A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124.

Milotti, E. (2022a). https://wwwusers.ts.infn.it/ milotti/didattica/gravitational-waves/handouts/cbc.pdf.

Milotti, E. (2022b). https://wwwusers.ts.infn.it/ milotti/didattica/gravitational-waves/handouts/sequence.pdf.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P., editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York.

Morawski, F., Bejger, M., and Ciecielkag, P. (2020). Convolutional neural network classifier for the output of the time-domain $\mathcal{f}$-statistic all-sky search for continuous gravitational waves. *Machine Learning: Science and Technology*, 1(2):025016.

Nascimento, R. and Viana, F. (2020). Cumulative damage modeling with recurrent neural networks. *AIAA Journal*, 58:1–13.

Nascimento, R. G., Fricke, K., and Viana, F. A. (2020). A tutorial on solving ordinary differential equations using python and hybrid physics-informed neural network. *Engineering Applications of Artificial Intelligence*, 96:103996.

Necula, V., Klimenko, S., and Mitselmakher, G. (2012). Transient analysis with fast wilson-daubechies time-frequency transform. *Journal of Physics: Conference Series*, 363(1):012032.

Newton, I. (1687). *Philosophiae naturalis principia mathematica.*. Jussu Societatis Regiae ac Typis Josephi Streater.

Newton, I. (1745). *De analysi per aequationes numero terminorum infinitas.* .

Nguyen, C. (2021). Status of the advanced virgo gravitational-wave detector.

Olah, C. (2015). https://colah.github.io/posts/2015-08-understanding-lstms/.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Peters, P. C. (1964). Gravitational radiation and the motion of two point masses. *Phys. Rev.*, 136:B1224–B1232.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. (2021). Universal differential equations for scientific machine learning.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Razzano, M. and Cuoco, E. (2018). Image-based deep learning for classification of noise transients in gravitational wave detectors. *Classical and Quantum Gravity*, 35(9):095016.

Runge, C. and König, H. (1924). *Vorlesungen über numerisches Rechnen.* Springer.

Salemi, F., Drago, M., Klimenko, S., Lazzaro, C., Milotti, E., Mitselmakher, G., Necula, V., O'Brian, B., Prodi, G. A., Szczepanczyk, M., Tiwari, S., Tiwari, V., V, G., Vedovato, G., and Yakushin, I. (2021). coherent waveburst, a pipeline for unmodeled gravitational-wave data analysis. *SoftwareX*, 14:100678.

Schutz, B. (2004). The art and science of black hole mergers. .

Shinohara, T., He, W., Matsuoka, Y., Nagao, T., Suyama, T., and Takahashi, T. (2023). Supermassive primordial black holes: a view from clustering of quasars at $z \sim 6$.

Tefas, A. and Nousi, P. (2023). Introduction to deep learning. Aristotle University of Thessaloniki.

the LIGO-Virgo-Kagra scientific collaboration (2015). https://gracedb.ligo.org/.

the LIGO-Virgo-Kagra scientific collaboration (2019). https://gwosc.org/.

the Nobel Committee for Physics (2018). https://www.nobelprize.org/uploads/2018/06/advanced-physicsprize2017.pdf.

the PyTorch collaborators (2023). https://pytorch.org/docs/stable/generated/torch.optim.rmsprop.html.

the SXS collaboration (2000). https://www.black-holes.org/code/spec.html.

the SXS collaboration (2019). https://data.black-holes.org/waveforms.

the Virgo scientific collaboration (2016). http://public.virgo-gw.eu/wp-content/uploads/2014/10/adv_virgob.png.

Vio, R. and Andreani, P. (2021). Everything you always wanted to know about matched filters (but were afraid to ask).

Wang, H. (2019). https://iphysresearch.github.io/survey4gwml/.

Wikipedia (2023). Interferometro virgo — wikipedia, l'enciclopedia libera. [Online; checked on 7-september-2023].

Wysocki, D., O'Shaughnessy, R., Lange, J., and Fang, Y.-L. L. (2019). Accelerating parameter inference with graphics processing units. *Physical Review D*, 99(8).

Zaman, S. M., Hasan, M. M., Sakline, R. I., Das, D., and Alam, M. A. (2021). A comparative analysis of optimizers in recurrent neural networks for text classification. In *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pages 1–6.

Acknowledgments

At the end of this Master degree, I want to thank all the people that supported me and allowed me to grow as a scientist and as a person.

First of all, I thank my supervisor Prof. Giacomo Ciani, and my co-supervisors in Trieste University, Prof. Edoardo Milotti and Prof. Agata Trovato. The new experience in a research group was the engine for my personal and professional growth.

I thank the degree commission for the autumn session in a.y. 2022/23, chaired by Prof. Sabino Matarrese, and the refree Dr. Tiziano Zingales.

I thank Prof. Paola Marigo and Prof. Roberto Turolla. We worked together to build this Master degree course. I really did appreciate the effort that both of them put in listening to students opinions and in make choices for a welcoming and formative place. Thanks to them, I was able to work as a student representative really as a service to my colleagues.

I thanks the LIGO-Virgo-Kagra collaboration to host #5 and #6 open data GWOSC workshops which make me passionate on gravitational wave data analysis and led me to the other side of the main desk for the first time.

I thank the 4th COST Action G2Net 2023 in Thessaloniki, Greece, to have introduced me in the machine learning world and in an international way of doing scientific research.

I thank my family, which supported me from the first moment.

Last, but not least, I thank Camilla, always present companion of my days. She always looked at me in the best way, ready to cheer me up in the heavy moments and to share the joy of main life changes.