UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Laurea Magistrale in Fisica

Gravitational wave modelling with machine learning

Relatore interno:
Prof. Pierre PIZZOCCHERO

Relatore esterno:
Prof. Walter DEL POZZO

Elaborato finale di:
Stefano SCHMIDT
Matricola n° 920715

Anno Accademico 2018-2019

Abstract

As the observations of gravitational waves from coalescence of compact objects are expected to be more frequent in the near future, the need for an accurate and fast gravitational waves data analysis framework is becoming more and more relevant. Any analysis requires a waveform model capable of predicting the shape of the gravitational wave signal, given the physical parameters of the source. Currently, several models exist, but they are cursed by the fact that they are slow to compute, hence they are the bottleneck of the task of extracting physical information from the raw data.

In our work, we apply Machine Learning methods to build a model designed to generate a gravitational waveform in the time domain as produced by a binary black hole coalescence. Our model matches in accuracy the performance of the state-of-the-art direct computations while, at the same time, it provides a speed up of a a factor of ~ 30 in the generation of the waveform. Furthermore, our model provides a closed form expression for the waveform and its gradient with respect to the orbital parameters. This open the possibility to further improve the sampling algorithms used for the parameter estimation.

We train our model on a number of waveforms computed by the `SEOBNRv2` generator and we infer a relation between the waveform and the masses m1,m2 and spins s1, s2 of the two BHs. In this work, we consider only the case in which the two spins are aligned to the orbital angular momentum. We reduce the dimensionality of our problem by decomposing each waveform in amplitude and phase and it is further represented in a lower dimensional space using a Principal Component Analysis. The regression from orbital parameters to principal components is performed using a Mixture of Expert model. Our implementation is publicly available as a `Python` package `mlgw` as `https://pypi.org/project/mlgw/`. `mlgw` has all the features required for performing a full parameter estimation (including the waveform dependence on geometrical parameters). We demonstrate the faithfulness of `mlgw` by successfully reproducing the inference on GW150914 made by the LIGO-Virgo collaboration.

Contents

Chapter 1

Introduction

The detection of gravitational waves (GW) has an important role in our ability to observe the universe and to understand the underlying physics. After the first detection on September 2015 [1] of a binary Black Hole coalescence, a number of signals were recorded and they allowed for a great number of tests of General Relativity, cosmology and astrophysics [2], [3]. With the scheduled improvements in detector sensitivity, a higher detection rate is expected in the near future, giving GW astronomy a central role among the observational channels of astrophysical objects.

The GW detection has been possible thanks to the joint effort of a number of different fields of expertise, taking care of different parts of the sophisticated detection process. Among them, data analysis is mainly concerned in detecting a GW signal in the raw detector output (*matched filtering*) and to produce physical prediction based on it (*parameter estimation*). In order to accomplish its goal, GW data analysis relies on accurate numerical waveforms, to compare the detector signal with. The predicted waveforms emitted by a physical system is obtained by solving Einstein equations for the relevant physical scenario. Once they are available, one can compare them to the observed detector output and state whether a physical signal is recorded by the detector and, if so, which sources are more likely to have produced the signal.

Data analysis requires a huge number of numerical waveforms. An accurate parameter estimation needs as many as 10^5 waveforms, generated for a variety of physical situations, and a similar number is required for matched filter. While for matched filtering the computation can be done off-line (i.e. before the detection), the parameter estimation requires the waveform to be generated on-line, as the physical parameters of the source are not known in advance. It is then manifest that a reliable and fast to run model for generating theoretical GW signal is a crucial part of the GW detection.

With the current interferometers, coalescing compact objects are expected to be the most observed GW sources and, for this reason, many efforts are devoted to solve Einstein equation for two coalescing objects and to predict the gravitational radiation emitted. As Einstein equation are highly non-linear and difficult to solve, an exact solution to the two body problem is still unknown and one must solve it numerically. A full solution of the Einstein equation is the realm of Numerical Relativity (NR) but it is unfeasibly slow to run (up to 2 weeks for a single waveform!) and undergoes many technical difficulties. For this reason, it cannot be effectively employed in GW detection, where speed is crucial.

An approximate semi-analytical solution is obtained by expanding in series the Einstein equations. This is called Post-Newtonian (PN) approach. It gives accurate results and, as it is moderately fast to run, it is currently employed in GW detection. However, for the late stage of coalescence the

method cannot give accurate results [1] and must use inputs from numerical relativity. Furthermore, despite being much faster than numerical relativity, such models are still the bottleneck of GW data analysis. The parameter estimation accuracy is limited in its precision by the time to generate such waveforms. For instance, if $O(1\,\mathrm{s})$ is required to build a single waveform, a parameter estimation can take $O(1\,\mathrm{week})$!

A Machine Learning model which generates a GW waveform is a promising alternative to the state-of-the-art waveforms generators. Machine Learning (ML) is a branch of statistics which is devoted to reproduce patterns read from data. A ML algorithm needs very little human input and, by a automatically solving an optimization problem, it is able to choose the best performing element among a large class of parametric models for the solution. This is called training procedure. The ML flexibility in modelling data and reproducing trends is appealing: with a proper model choice and with an appropriate training procedure, we can hope to have a reliable, fast to execute, generator of GW waveforms.

The advantages are manifold. First of all, one needs not to solve a differential equation to generate a waveform but a single closed form function is available for this purpose. This can result in a speed up of the time to generate a waveform and a significant increase in parameter estimation accuracy. Furthermore, a reliable ML scheme could be fed only with NR waveforms, which arises from solving the exact Einstein equation. This in principle will allow to bypass the PN approximation and to work directly with the full equations. Nowadays, too few NR waveforms are available and for training cannot avoid PN surrogate models. However, in a close future, NR waveforms might be used for the purpose. A third advantage is more technical, but still important. A closed form expression for a waveform yields a closed form for the gradients of the waveforms with respect to the physical quantities used to generate the wave. Information about gradients can be successfully used for a speed up of the parameter estimation, by providing an estimation of the regions where more probability mass is concentrated (for a proof proof of principle of such framework see [4]).

In this work, we explore this opportunity and we create a ML model which is able to produce GW signal generated by a binary Black Hole coalescence. We train a function that outputs a waveform when given masses (m_1, m_2) and z components of spins (s_1, s_2) of the two merging BHs. We consider only the case in which the two spins are aligned with the orbital angular momentum: this is called non-precessing case. The model is trained with waveform by the surrogate PN models, currently used in LIGO/Virgo data analysis, and it is able to faithfully reproduce them. The waveforms generated comply with the LIGO/Virgo standard and are ready to use for a real life application: an interested user can use the function as a "black box". The model is distributed as a Python package `mlgw` It is available in the PyPI repository and can be installed with `pip install mlgw`.

Our work is organized as follows.

In chapter 2 we review some basics concepts in GW theory, detection and data analysis. After a brief order of magnitude estimation of the relevant physical quantities, we show how a wave equation arises from Einstein equations and show some GW properties. Next we consider GW emission in the particular case of compact object coalescence, which is of special interest for GW detection. Finally we present basics of GW interferometry and GW data analysis. We will put a particular emphasis on the latter.

In chapter 3 we discuss some ML basics and present some ML models which might be successfully applied to our problem. In particular we discuss regression algorithms and dimensional reduction

[1] As the two compact objects gets closer, gravity gets stronger and eventually the PN approximation will not be adequate anymore.

models. The first are useful to establish a relation between orbital parameters and a (low dimensional) wave representation; the latter is useful to reduce the number of parameters used to fully characterize a wave.

Chapter 4, is devoted to a thorough discussion of the ML model and its details. We present every "building block" of the model and the main issues that must be solved. We then summarize the functioning of the `mlgw` package. To put our work into a broader context, the chapter opens with a brief review of earlier works on ML and GW.

In chapter 5 we present some results regarding the model performances. We will asses the model dependence on its hyperparameters (i.e. non trainable parameters that must be set by the user) and we will measure model ability to reproduce state-of-the-art waveforms. In the last part we will measure run time for ML generation of waveforms and we will compare it with standard surrogate models.

Chapter 6 holds some final remarks about our work and an outline of future developments.

Chapter 2

Basics of Gravitational Waves

In this chapter we present some basics concepts on Gravitational Wave (GW) theory and detection. By no means the chapter is intended to be exhaustive; rather it should be regarded as a summary of the physical ideas used throughout the work. For a deeper discussion the reader can refer to [5], [6], [7] [8] and [9].

In the first section, we provide some order of magnitude estimations of the typical physical quantities we will work with. In the following section a general theoretical introduction to gravitational waves is presented: the wave equation (Pauli-Fierz equation) and the famous quadrupole formula are derived. The third section is devoted to the coalescence of compact object and gravitational waves emitted by the system. Newtonian description of inspiral is treated in some details; late inspiral, merger and ring-down phase are just outlined. A fourth section deals with the way we observe GWs and learn something from them. Basic operation of interferometers is described and some of the most common data analysis techniques are presented.

In what follows we set $G = c = 1$ where convenient.

2.1 Order of magnitude estimation

Before diving into calculations and lengthy formulas, it is interesting to have a taste for the magnitude of the physical quantities which we talk about. For this reason we present here some order of magnitude estimation relevant to the field of GW and binary compact object coalescence. The discussion is specialized to the case of the compact object coalescence and numerical estimations will be given for systems with their typical magnitudes; the discussion however can be made more general and applied to other radiating systems (such as a perturbed neutron star).

A gravitational wave is a perturbation to the spacetime metric which propagates far away from its source. Like the metric tensor it is dimensionless. At a given point away from the source, the magnitude h of the wave (also called *strain*) depends on the the distance r of the observer, on the source mass M and on the source size R. [1]. Building together a dimensionless quantity h for this is straightforward:

$$h \sim \frac{M^2}{rR} \sim \frac{M}{R}\frac{M}{r} \sim \varphi_{int}\varphi_{ext} \tag{2.1}$$

thus the magnitude of a GW is proportional to the product of the internal gravitational field (which

[1] In the context of inspiral, M is the total mass of the binary system and R is the distance of the two coalescing objects.

binds together the system) and of the external gravitational field. Inserting the numerical values we get:

$$h \sim \frac{G^2}{c^4}\frac{M^2}{rR} \sim 10^{-21}\left(\frac{M}{M_\odot}\right)^2\left(\frac{1\,\mathrm{Mpc}}{r}\right)\left(\frac{100\,\mathrm{km}}{R}\right) \tag{2.2}$$

10^{-21} is a extremely small number: GW at our astronomical distances has a tiny effect.

We now consider the matter of the wave frequency. The emission frequency must be close to the system proper frequency of a system. For a gravitationally bound systems the proper frequency is given by [2]:

$$f \sim \sqrt{\frac{M}{R^3}} \tag{2.3}$$

As the wave propagates along the spacetime no change in frequency should be observed and thus for a coalescence of compact object we should measure a frequency of:

$$f \sim 300\,\mathrm{Hz}\left(\frac{M}{M_\odot}\right)^{\frac{1}{2}}\left(\frac{100\,\mathrm{km}}{R}\right)^{\frac{3}{2}} \tag{2.4}$$

For BBH coalescence, the frequency emitted falls in the human audible frequency range.

To obtain the luminosity (energy radiated per unit time) of GW emission, we note that $\frac{c^5}{G}$ has the same dimension as the luminosity, regardless system features. This means that GR has a typical built in luminosity $L_0 = \frac{c^5}{G} \simeq 10^{52}\mathrm{W}$. It is reasonable to expect that a source will emit a fraction of L_0 and since the only dimensionless combination of the physical dimensions of the system is $\frac{M}{R}$, the total luminosity will depend on a power α of $\frac{M}{R}$. Furthermore α is expected to be positive: for big compactness $\frac{M}{R}$, gravity plays a big role and thus the system should have higher luminosity. Unfortunately the right value of α cannot be determined and one must look at more specific calculations (see (2.51)). The correct formula turns out to be:

$$L \sim L_0\left(\frac{M}{R}\right)^5 \sim 10^{47}\mathrm{W}\left(\frac{M}{M_\odot}\right)^5\left(\frac{10\,\mathrm{km}}{R}\right)^5 \tag{2.5}$$

L_0 is a huge value: gravitational radiation is a very effective system to radiate away energy.

With a formula for luminosity we can give an estimate of the time duration of a compact object coalescence signal. The effect of GW emission is to shrink the orbit radius to compensate for potential energy loss. As radius is reduced, the emitted luminosity increases making the GW emission faster: the increase of frequency and amplitude is called *chirp* and is typical of a coalescence of binary systems. The time to coalescence can be estimated as follows:

$$t_{coal} \sim \frac{E_{tot}}{L} \sim \frac{GM^2/R}{L_0(M/R)^5} \sim 100\,\mathrm{s}\left(\frac{M_\odot}{M}\right)^3\left(\frac{R}{100\,\mathrm{km}}\right)^4 \tag{2.6}$$

Using (2.6) one can eliminate the R dependence in (2.4) and compute the emission frequency as a function of time to coalescence:

$$f \sim 300\,\mathrm{Hz}\left(\frac{M_\odot}{M}\right)^{\frac{5}{8}}\left(\frac{100\,\mathrm{s}}{t_{coal}}\right)^{\frac{3}{8}} \tag{2.7}$$

It is interesting to have an estimate of the maximum frequency generated by a coalescence. The inspiral ends when the two compact object are closer than the innermost stable circular orbit (ISCO)

[2] It is remarkable to note how this formula is able to reproduce (a part from small constants) Kepler's law: $\omega^2 = \frac{GM}{R^3}$.

at $R = 6M$; at that point the two compact objects are so close that they plunge quickly into each other. Signal at (much) higher frequencies is not expected. The ISCO the frequency is:

$$f_{ISCO} \sim 10\,\mathrm{kHz}\left(\frac{M_\odot}{M}\right) \tag{2.8}$$

The ISCO frequency sets an upper limit for the emission frequency and depends only on the mass and it can be regarded as the natural frequency scale for the late inspiral.

We now want to to quantify the effect on GW that can be measured by an observer. A metric perturbation has the effect of changing distances between test masses. The relevant quantities here are the size L of a system and its variation δL due to the arrival of GW. It is natural to relate the two length with the strain as follows:

$$h \sim \frac{\delta L}{L} \tag{2.9}$$

Thus the strain is proportional to the fractional variation in the test object size. Since strain magnitude is usually tiny, the effect of GW is little and hard to measure.

2.2 The wave equation

Einstein equations allow naturally for a wave like solution. In this section some basic results are presented.

2.2.1 Pauli-Fierz equations

In order to write Einstein equations as a wave equation, we perform an expansion of the metric around flat space time. We assume that the metric, far away from every source, can be written as Minkovski metric plus a small perturbation:

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \quad |h_{\mu\nu}| << 1 \tag{2.10}$$

where we define $\eta_{\mu\nu} = \mathrm{diag}(-1, +1, +1, +1)$. This is the *linearized theory* [3]. Condition (2.10) can always be obtained by a suitable coordinate transformation (space-time is always locally flat!!). With linearized theory we mean that (2.10) is valid for a large portion of space-time and not only locally. The region of spacetime where (2.10) holds is called *radiation zone*.

By inserting the metric (2.10) in Einstein's equations and expanding at first order in $h_{\mu\nu}$, Einstein's equations can be reduced to a wave equation for the perturbation. It is convenient to define the traceless perturbation $\bar{h}_{\mu\nu}$:

$$\bar{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h \tag{2.11}$$

One then finds that:

$$G_{\mu\nu} \simeq -\frac{1}{2}\partial^\rho\partial_\rho\bar{h}_{\mu\nu} + \partial^\rho\partial_{(\mu}\bar{h}_{\nu)\rho} - \frac{1}{2}\partial^\rho\partial_\sigma\bar{h}_{\rho\sigma}\eta_{\mu\nu} = 8\pi T_{\mu\nu} \tag{2.12}$$

where the parenthesis on two indices denote symmetrization: $\partial_{(\mu}\bar{h}_{\nu)\rho} = \frac{1}{2}(\partial_\mu\bar{h}_{\nu\rho} + \partial_\nu\bar{h}_{\mu\rho})$.

Equation (2.12) can be simplified by looking at the symmetries that linearized theory inherits from

[3] Since in linearized theory only first order terms are retained, the flat metric is used to raise and lower the indices within the theory.

the symmetry under coordinate transformation of the general theory. A coordinate transformation is a diffeomorphism of the form:

$$x^\mu \to x'^\mu \tag{2.13}$$

Under (2.13) the unperturbed metric transforms as follows:

$$g_{\mu\nu}(x) \to g'_{\mu\nu}(x') = \frac{\partial x^\alpha}{\partial x'^\mu}\frac{\partial x^\beta}{\partial x'^\nu} g_{\alpha\beta}(x) \tag{2.14}$$

When applied to the perturbed metric, the coordinate transformation must be small (otherwise the transformation breaks the condition in (2.10)) and thus must be on the form of:

$$x^\mu \to x'^\mu = x^\mu + \xi^\mu(x) \tag{2.15}$$

where for consistency $\partial\xi$ must be at most of the same order of h. With this, equation (2.14) specialises to:

$$h_{\mu\nu}(x) \to h'_{\mu\nu}(x') = h_{\mu\nu}(x) - (\partial_\mu \xi_\nu + \partial_\nu \xi_\mu) \tag{2.16}$$

Equation (2.16) is the gauge transformation law for the first order perturbation to the metric.

We can thus use this freedom to set the *Lorenz gauge* in which [4]:

$$\partial^\nu \bar{h}_{\mu\nu}(x) = 0 \tag{2.17}$$

In Lorentz gauge, the perturbed Einstein equation (2.12) takes the much simpler form:

$$\Box \bar{h}_{\mu\nu} = -\frac{16\pi G}{c^4} T_{\mu\nu} \tag{2.18}$$

where as usual $\Box = \partial_\mu \partial^\mu$. The equations are called Pauli-Fierz equations and together with the Lorentz gauge specification they form the starting point for the analytical treatment of gravitational radiation.

2.2.2 GW propagation in vacuum

We now turn to the problem of how a GW propagates in *vacuum* and therefore we set $T_{\mu\nu} = 0$.

In order to make any computation, a frame of reference must be chosen. Setting the Lorentz gauge (2.17) does not fully specify a frame and a further gauge choice must be performed. In fact, Lorentz condition still holds if a transformation (2.15) is performed with $\Box\xi^\mu = 0$ [5]. The usual choice is to set a frame in which [6]:

$$\bar{h} = 0 \ , \ \bar{h}_{0i} = 0 \tag{2.19}$$

Note that $\bar{h} = 0$ implies that $\bar{h}_{\mu\nu} = h_{\mu\nu}$.

The coordinate frame in which (2.19) is valid is called *transverse-traceless gauge* (TT gauge). In TT gauge the dynamics of the perturbed metrics is fully specified by Einstein eq (2.18) plus some

[4] This is possible to achieve using (2.16):

$$\bar{h}'_{\mu\nu} = \bar{h}_{\mu\nu} - (\partial_{(\mu}\xi_{\nu)} - \eta_{\mu\nu}\partial^\rho \xi_\rho) \to \partial^\nu \bar{h}'_{\mu\nu} = \partial^\nu \bar{h}_{\mu\nu} - \Box\xi_\mu$$

and the left hand side of the last equation can be always set to 0 since one can always choose ξ_μ such that $\Box\xi_\mu = \partial^\nu \bar{h}_{\mu\nu}$.

[5] This is due to the fact that under (2.15) $\partial^\nu \bar{h}'_{\mu\nu} = \partial^\nu \bar{h}_{\mu\nu} - \Box\xi_\mu$ and thus if $\Box\xi^\mu = 0$ the value of $\partial^\nu \bar{h}_{\mu\nu}$ is not affected by the transformation.

[6] The existence of such a frame is not trivial to prove: one must solve eq (2.16) to find the explicit values of ξ^μ satisfying the condition (2.19). The computation works only outside the source.

gauge specifications. In summary:

$$\Box h_{ij} = 0 \qquad \text{Dynamic equation (Pauli-Fierz)} \tag{2.20}$$

$$h^{0\mu} = 0\ ,\ h^i_i = 0\ ,\ \partial^j h_{ij} = 0 \qquad \text{TT gauge specification} \tag{2.21}$$

Note that TT gauge can only be chosen outside source where $T_{\mu\nu} = 0$. In fact, while one can still choose Lorentz gauge inside the source, one cannot set to zero any component of $h_{\mu\nu}$ and thus cannot make the simplification above.

The equations above are useful to understand the nature of gravitational radiation. First of all, since $\Box = -\frac{1}{c^2}\partial_t^2 + \nabla^2$, a GW propagates at the speed of light. Furthermore, solving (2.20) can give information about the polarization. As usual, it is convenient to solve equation (2.20) with plane wave ansatz:

$$h_{ij} = H_{ij} \cdot e^{ik^\mu x^\mu} \tag{2.22}$$

where H_{ij} is a constant polarization and k^μ is the wave vector. Using (2.21) one finds:

$$H^i_i = 0\ ,\ k^j H_{ij} = 0 \tag{2.23}$$

Due to its symmetry, the polarization H_{ij} has 6 free parameters and equation (2.23) gives 4 constraints: this means that there are only two independent polarization for a gravitational wave. We learn from the second equality in (2.23) that a GW is a transverse wave.

Taking $k^\mu = (\omega, \mathbf{k})$ and wave vector $\mathbf{k}$ along z axis a generic plane wave can be parameterized as follows:

$$h_{ij}(t,z) = \begin{pmatrix} h_+ & h_\times & 0 \\ h_\times & -h_+ & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ij} \cos(\omega(t-z)) \qquad = (h_+ \cdot H_{+ij} + h_\times \cdot H_{\times ij}) \cos(\omega(t-z)) \tag{2.24}$$

where $H_+ = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ and $H_\times = \left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$ are the two independent polarizations called respectively "plus" and "cross".

In this frame the whole space-time metric takes the following form:

$$\mathrm{d}s^2 = -\,\mathrm{d}t^2 + \mathrm{d}z^2 + [1 + h_+ \cos(\omega(t-z))]\,\mathrm{d}x^2 + [1 - h_+ \cos(\omega(t-z))]\,\mathrm{d}y^2 + 2h_\times \cos(\omega(t-z))\,\mathrm{d}x\,\mathrm{d}y \tag{2.25}$$

So far we have considered a wave propagating along the z-axis. In the more general case, one can choose a wave propagating in a generic direction $\hat{\mathbf{n}}$ and setting an orthonormal coordinate system $(\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{n}})$. In this frame $\mathbf{k} = k\hat{\mathbf{n}}$ and the polarizations are:

$$H^+_{ij}(\hat{\mathbf{n}}) = \hat{u}_i \hat{u}_j - \hat{v}_i \hat{v}_j \tag{2.26}$$

$$H^\times_{ij}(\hat{\mathbf{n}}) = \hat{u}_i \hat{v}_j + \hat{v}_i \hat{u}_j \tag{2.27}$$

Effects of GW on spacetime It is instructive to compute the geodesic equation and the geodesic deviation equation in the metric (2.25).

We evaluate at $\tau = 0$ the geodesic equation for a particle initially at rest in TT frame:

$$\left.\frac{\mathrm{d}^2 x^i}{\mathrm{d}^2\tau^2}\right|_{\tau=0} = -\Gamma^i_{\mu\nu} \left.\frac{\mathrm{d}x^\mu}{\mathrm{d}\tau}\frac{\mathrm{d}x^\nu}{\mathrm{d}\tau}\right|_{\tau=0} = -\Gamma^i_{00} \left.\left(\frac{\mathrm{d}x^0}{\mathrm{d}\tau}\right)^2\right|_{\tau=0} \tag{2.28}$$

The relevant Christoffel symbols for the perturbed metric are:

$$\Gamma^i_{00} = \frac{1}{2}(2\partial_0 h_{0i} - \partial_i h_{00}) \tag{2.29}$$

Since this vanishes in the TT gauge, we find that a free falling particle initially at rest in the TT gauge will not accelerate and will remain at rest at the transit of a GW. Thus the TT frame reference is a *local inertial frame* [7] where gravity has no effect on particle's motion. A free-falling observer does not notice the passage of a gravitational wave.

However, the arrival of a gravitational wave has a measurable physical effect. Consider the geodesic deviation equation which measures tidal acceleration among two nearby freely falling bodies. Let X be their (small) separation and let the two be at rest (or at least slowly moving such that $\frac{\mathrm{d}x^\mu}{\mathrm{d}\tau} = (1,0,0,0)$) in the coordinate system chosen. The equation for their separation is:

$$\frac{\mathrm{d}^2 X^\mu}{\mathrm{d}^2 t^2} = R_{\nu 00}{}^{\mu} X^\nu = \frac{1}{2}\frac{\partial^2 h^\mu{}_\nu}{\partial t^2} X^\nu \tag{2.30}$$

From here the physical effect of a gravitational wave is clear: it stretches the proper separation s_i between two freely falling frames according to equation [8]:

$$\ddot{s}_i = \frac{1}{2}\ddot{h}_{ij} s_j \tag{2.31}$$

This effect is exploited by any attempt to detect gravitational waves (see section 2.4.1): if the distance between two points is measured at different times with enough precision, the effect in eq (2.31) can be observed.

2.2.3 GW sources: multipole expansion and the quadrupole formula

We now address the issue of wave generation. The relevant formula is eq. (2.18); we work in Lorentz gauge (not yet in TT gauge). As it is customary in electrodynamics, the equation is solved by means of Green's function. The Green's function of D'Alambertian operator satisfies $\Box_x G(x^\sigma - y^\sigma) = \delta^{(4)}(x^\sigma - y^\sigma)$ hence the solution to (2.18) can be formally written as:

$$\begin{aligned}\bar{h}_{\mu\nu}(x) &= -16\pi \int \mathrm{d}^4 y\, G(x-y) T_{\mu\nu}(y) \\ &= 4\int \mathrm{d}^3 y \frac{1}{|\mathbf{x}-\mathbf{y}|} T_{\mu\nu}(t - |\mathbf{x}-\mathbf{y}|, \mathbf{y})\end{aligned} \tag{2.32}$$

where we used the fact that for $\Box$ operator the *retarded* Green's function has the form: $G(x^\sigma - y^\sigma) = -\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|}\delta[|\mathbf{x}-\mathbf{y}| - (x^0 - y^0)]\theta(x^0 - y^0)$.

To move further it is convenient to work in Fourier space with respect to time. Let's denote

[7] A local inertial frame at point P is a frame in which $\frac{\mathrm{d}^2 x^i}{\mathrm{d}^2\tau^2}|_P = 0$. It is a well known property of gravity that such a frame always exist for every given point.

[8] Equation (2.30) and (2.31) are valid only up to first order in h and for small separation s. The sign = should be more precisely replaced by $\simeq$.

$\tilde{\varphi}(\omega, \mathbf{x}) = \frac{1}{\sqrt{2\pi}} \int \mathrm{d}t\, e^{i\omega t}\varphi(t, \mathbf{x})$ the Fourier transform of $\varphi(t, \mathbf{x})$. The transformed metric is:

$$\begin{aligned}\tilde{\bar{h}}_{\mu\nu}(\omega, \mathbf{x}) &= \frac{1}{\sqrt{2\pi}} \int \mathrm{d}t\, e^{i\omega t} \bar{h}_{\mu\nu}(t, \mathbf{x}) \\ &= \frac{4}{\sqrt{2\pi}} \int \mathrm{d}t_{ret}\, \mathrm{d}^3y\, e^{i\omega t_{ret}} e^{i\omega|\mathbf{x}-\mathbf{y}|} \frac{T_{\mu\nu}(t_{ret}, \mathbf{y})}{|\mathbf{x}-\mathbf{y}|} \\ &= 4 \int \mathrm{d}^3y\, \frac{e^{i\omega|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|} \tilde{T}_{\mu\nu}(\omega, \mathbf{y}) \end{aligned} \tag{2.33}$$

The problem with this formula is that the term $\frac{e^{i\omega|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|}$ within the integral is hard to evaluated. In the general case, it can be expanded in terms of spherical harmonics and Bessel function: the spherical harmonics expand the angular part while Bessel functions take care of the radial dependence of vector **x** - **y** (see e.g. [10, ch 16.1]).

Here however we will pursue a less general approach by making three simplifying hypothesis about the source:

- Source is *localized*: source has a well defined dimension $O(d^3)$, where d is its typical scale.
- Source is *far away*: the observer of the emitted GW is at a distance $r = |\mathbf{x}| >> d$. We are then allowed to write: $|\mathbf{x} - \mathbf{y}| \simeq r - \mathbf{y} \cdot \hat{\mathbf{n}} + O\left(\frac{d^2}{r}\right)$, where $\hat{\mathbf{n}} = \frac{\mathbf{x}}{r}$
- Source is *slowly moving*: the typical internal velocity of the source v is small, $v << 1$. Under this hypothesis, the typical frequency of emission $\omega \sim v/d$ satisfies: $\omega d << 1$.

With the approximation above [9]. , we can expand $\frac{e^{i\omega|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|}$ in powers of $\omega\mathbf{y} \cdot \hat{\mathbf{n}} \lesssim \omega d << 1$:

$$\frac{e^{i\omega|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|} \simeq \frac{1}{r} e^{i\omega(r - \mathbf{y}\cdot\hat{\mathbf{n}})} \simeq \frac{e^{i\omega r}}{r} \left[1 - i\omega y_i \hat{n}_i + \frac{(-i\omega)^2}{2} y_i y_j \hat{n}_i \hat{n}_j + O((\omega d)^3)\right] \tag{2.34}$$

We can plug the series in (2.33) and we obtain:

$$\begin{aligned}\tilde{\bar{h}}_{\mu\nu}(\omega, \mathbf{x}) &= 4\frac{e^{i\omega r}}{r} \left[\int \mathrm{d}^3y \tilde{T}_{\mu\nu}(\omega, \mathbf{y}) - i\omega n_i \int \mathrm{d}^3y \tilde{T}_{\mu\nu}(\omega, \mathbf{y}) y_i + \frac{(-i\omega)^2}{2} n_i n_j \int \mathrm{d}^3y \tilde{T}_{\mu\nu}(\omega, \mathbf{y}) y_i y_j + \ldots\right] \\ &= 4\frac{e^{i\omega r}}{r} (\tilde{Q}_{\mu\nu} - i\omega n_i \tilde{Q}_{\mu\nu,i} + \frac{(-i\omega)^2}{2} n_i n_j \tilde{Q}_{\mu\nu,ij} + \ldots) \end{aligned} \tag{2.35}$$

The series (2.35) provides an approximate solution to the gravitational wave in the the far zone. The right hand side is usually called *multipole expansion* of the source. Transforming in time domain [10] , we get:

$$\bar{h}_{ij}(t, \mathbf{x}) = 4\frac{1}{r}\left[Q_{ij}(t-r) + n_k \dot{Q}_{ij,k}(t-r) + \frac{1}{2} n_k n_l \ddot{Q}_{ij,kl}(t-r) + \ldots\right] \tag{2.36}$$

In the last equation we consider only space-like components of h because, as discussed above, they are the only nonvanishing components in the far zone when imposing TT gauge. It is important to remember that the multipole expansion above is valid only in the approximation of slowly moving isolated and far sources. Furthermore, the formula neglects gravity back-action on the source: this

[9] The approximation is reasonable. For example, we usually observe a binary Black Hole (BBH) from a distance $r \simeq 1\,\mathrm{Mpc}$ while the internal structure of the system is of order $d \simeq 100\,\mathrm{km}$. The typical frequency is of order $100\,\mathrm{Hz} \simeq 1000\,\mathrm{km}$ which yields a distance much larger than typical size of the system.

[10] The term $e^{i\omega r}$ ensures that Q_{ij} is evaluated at retarded time, thus respecting causality.

cannot be captured by the first order expansion of the flat metric and requires a more accurate formalism (see 2.3.2). However, it still shows many important features of the GW emission and is useful as a starting point for more detailed computations.

We define the following momenta of masses M and linear momentum P^i:

$$\begin{aligned}
M &= \int \mathrm{d}^3 y T^{00} & P^i &= \int \mathrm{d}^3 y T^{0i} & \\
M^i &= \int \mathrm{d}^3 y T^{00} y^i & P^{i,j} &= \int \mathrm{d}^3 y T^{0i} y^j & \text{Dipole momentum} \\
M^{ij} &= \int \mathrm{d}^3 y T^{00} y^i y^j & P^{i,jk} &= \int \mathrm{d}^3 y T^{0i} y^j y^k & \text{Quadrupole momentum} \\
M^{ijk} &= \int \mathrm{d}^3 y T^{00} y^i y^j y^k & P^{i,jkl} &= \int \mathrm{d}^3 y T^{0i} y^j y^k y^l & \text{Octupole momentum} \\
&\dots & &\dots &
\end{aligned}$$

Note that the quantity above depends only on time and not on space. Their physical meaning is clear: they summarize how the mass (linear momentum) distribution affects the wave at large distance, integrating out the dependence on the details of the source. To make eq. (2.36) more interpretable, it is common to express the quantities $Q_{\dots}$ as a function of the above momenta. We do this by observing that in Fourier space, it holds $i\omega \tilde{T}_{0\nu} = -\partial^i \tilde{T}_{i\nu}$, by 4-momentum conservation $\partial^\mu T_{\mu\nu} = 0$. For example:

$$\begin{aligned}
\tilde{Q}^{ij} = \int \mathrm{d}^3 y \; \tilde{T}^{ij}(\omega, \mathbf{y}) &= \int \mathrm{d}^3 y \; \partial_k (y^i \tilde{T}^{kj}) - y^i \partial_k \tilde{T}^{kj} = -\int \mathrm{d}^3 y \; y^i \partial_k \tilde{T}^{kj} \\
&= i\omega \int \mathrm{d}^3 y \; y^i \tilde{T}^{0j} = \frac{i\omega}{2} \int \mathrm{d}^3 y \; (y^i \tilde{T}^{0j} + y^j \tilde{T}^{0i}) \\
&= \frac{i\omega}{2} \int \mathrm{d}^3 y \; \partial_l (y^i y^j T^{0l}) - y^i y^j \partial_l \tilde{T}^{0l} \\
&= \frac{(i\omega)^2}{2} \int \mathrm{d}^3 y \; y^i y^j \tilde{T}^{00} = \frac{(i\omega)^2}{2} \tilde{M}^{ij}
\end{aligned} \tag{2.37}$$

Similar expression can be obtained for other moments $\tilde{Q}^{ij,\cdots}$. For example, the next-to-leading order is:

$$i\omega \tilde{Q}^{ij,k} = \frac{(i\omega)^3}{6} \tilde{M}^{ijk} + \frac{(i\omega)^2}{3} (\tilde{P}^{i,jk} + \tilde{P}^{j,ik} - 2\tilde{P}^{k,ij}) \tag{2.38}$$

and it is composed by a mass octupole term and a combination of current quadrupole terms.

From now on, we specialise equation (2.36) by considering only the first term of the expansion. Using (2.37) and transforming in time domain, we find $S_{ij} = \frac{1}{2}\ddot{M}_{ij}$ and obtain the famous quadrupole formula for GW emission by a distant source:

$$\bar{h}_{ij}(t, \mathbf{x}) = 2\frac{1}{r} \ddot{M}_{ij}(t - r) \tag{2.39}$$

The expression (2.39) is written in Lorentz gauge (with $\partial^\mu \bar{h}_{\mu\nu} = 0$) but not in TT gauge. It is possible to obtain the expression in TT gauge by means of projection. We define the projector $P_{ij}(\hat{\mathbf{n}}) = \delta_{ij} - n_i n_j$ on the space orthogonal to $\hat{\mathbf{n}}$. We then define the *Lambda tensor*:

$$\Lambda_{ij,kl}(\hat{\mathbf{n}}) = P_{ik} P_{jl} - \frac{1}{2} P_{ij} P_{kl} \tag{2.40}$$

The $\Lambda_{ij,kl}$ tensor is traceless w.r.t. indices ij and kl and transverse $n^i \Lambda_{ij,kl} = 0$ on all its indices.

Using this one can prove that, if h_{ij} is a plane wave in Lorentz gauge, moving in direction $\hat{\mathbf{n}}$, the corresponding metric in TT gauge is given by:

$$h_{ij}^{TT} = \Lambda_{ij,kl}(\hat{\mathbf{n}})h_{kl} \tag{2.41}$$

We now obtain the angular pattern of emission of a gravitational wave in the quadrupole mode. Let the wave travel in direction $\hat{\mathbf{n}}$ and let M_{ij} be the quadrupole moment computed in a generic frame defined by coordinates $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$. In the orthogonal coordinate system $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}$, the wave has the usual form:

$$h'_{ij}(t, \mathbf{x}) = \frac{2}{R} \begin{pmatrix} \ddot{M}'_{11}(t) & \ddot{M}'_{12}(t) & 0 \\ \ddot{M}'_{21}(t) & \ddot{M}'_{22}(t) & 0 \\ 0 & 0 & 0 \end{pmatrix}_{ij} e^{i\omega \mathbf{n}\cdot\mathbf{x}} \tag{2.42}$$

The polarizations in TT gauge are obtained by rotating the matrix in eq. (2.42) and projecting the result with Lambda tensor (like in eq. (2.41)). They are expressed in terms of the quadrupole moment computed in $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ frame:

$$\begin{aligned} h_+ = \frac{1}{R}[&\ddot{M}_{11}(\cos^2\varphi - \sin^2\varphi\cos^2\theta) + \ddot{M}_{22}(\sin^2\varphi - \cos^2\varphi\cos^2\theta) - \ddot{M}_{33}\sin^2\theta \\ &- \ddot{M}_{12}\sin 2\varphi(1+\cos^2\theta) + \ddot{M}_{13}\sin\varphi\sin 2\theta) + \ddot{M}_{23}\cos\varphi\sin 2\theta)) \end{aligned} \tag{2.43}$$

$$h_\times = \frac{1}{R}[(\ddot{M}_{11} - \ddot{M}_{22})\sin 2\varphi\cos\theta + 2\ddot{M}_{12}\cos 2\varphi\cos\theta - 2\ddot{M}_{13}\cos\varphi\sin\theta + 2\ddot{M}_{23}\sin\varphi\sin\theta] \tag{2.44}$$

Where θ and φ represents the polar angles of $\hat{\mathbf{n}}$ in the $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ frame: $\theta = \hat{\mathbf{n}} \angle \hat{\mathbf{z}}$ and $\varphi = \hat{\mathbf{n}}_{xy} \angle \hat{\mathbf{y}}$ and $\hat{\mathbf{n}}_{xy} = \hat{\mathbf{n}} - < \hat{\mathbf{n}}, \hat{\mathbf{z}} > \hat{\mathbf{z}}$ is the projection of $\hat{\mathbf{n}}$ over the $\hat{\mathbf{x}} - \hat{\mathbf{y}}$ plane. The expression above is the typical emission pattern of the dipole radiation: it is due only to the quadrupole approximation of the source and not to the source features.

2.2.4 Energy radiated by a GW in the quadrupole approximation

The matter of energy of a gravitational wave (and indeed of energy in GR) is thorny and requires some caution. On the one hand, we know that an incoming gravitational wave is able to make work: if a spring is put where a GW is passing, it will be elongated and compressed thus acquiring potential energy. On the other hand, there is no meaningful notion of local energy density in general relativity, since there is not a clear distinction among background and dynamical part of the metric. Indeed with a gauge transformation one is able to make the metric flat in an arbitrary point, thus making the definition of local energy problematic.

A formal definition of energy can be given in GR for an isolated system by examining the effect of system mass at large distance. Once the total energy is defined, we have a natural way to define the flux of energy carried away by radiation. The formal definition of total energy can be given in terms of Komar integral but won't be discussed here (see e.g. [7, ch. 11]).

In what follows, we will adopt a different approach. The energy of a gravitational wave can be considered as the ability of the gravitational field perturbation to curve the space-time. Since in vacuum at first order in h, we have $G^{(1)}_{\mu\nu}[h_{\rho\sigma}] = 0$, the nonzero curvature generated by the "energy" of h must be due to higher order terms. For this reason, we consider the Einstein equation up to second

order in h. We obtain:

$$G_{\mu\nu}[h_{\rho\sigma}] \simeq G^{(1)}_{\mu\nu}[h_{\rho\sigma}] + G^{(2)}_{\mu\nu}[h_{\rho\sigma}] = 0 \tag{2.45}$$

where the right hand side is set to zero because we consider equation in vacuum. The first term vanishes identically and gives the usual wave propagation of perturbation. If we do not neglect the second order term, we can interpret it in two ways. A first option is to consider it as a correction to the propagation term which adds nonlinearities due to self interactions of the h field. Another option is to move $G^{(2)}_{\mu\nu}[h_{\rho\sigma}]$ to the right hand side and to consider it as an effective energy momentum tensor. If we adopt the second strategy, we obtain the definition of an energy momentum tensor for a gravitational wave, thus quantifying its energy. The Einstein equation can be written as:

$$G^{(1)}_{\mu\nu}[h_{\rho\sigma}] = 8\pi\, t^{gw}_{\mu\nu} = -G^{(2)}_{\mu\nu}[h_{\rho\sigma}] \tag{2.46}$$

When compared to the typical timescales of compact objects such as planets or stars, a GW is a fast oscillating object. For this reason, in order to quantify the effects of a GW on larger scales, it makes sense to average over some wavelengths: this is equivalent to integrate out degrees of freedom impossible to resolve at the large scale of observation [11]. We can now obtain the expression for $t^{gw}_{\mu\nu}$ in terms of the metric perturbation $h_{\mu\nu}$ [12]:

$$t^{gw}_{\mu\nu} = -\frac{1}{8\pi}\langle G^{(2)}_{\mu\nu}[h_{\rho\sigma}]\rangle = -\frac{1}{8\pi}\langle R^{(2)}_{\mu\nu}[h_{\rho\sigma}] - \frac{1}{2}\eta_{\mu\nu}R^{(2)}[h_{\rho\sigma}]\rangle \tag{2.47}$$

$$= \frac{c^4}{32\pi G}\langle \partial_\mu h_{\rho\sigma}\partial_\nu h^{\rho\sigma}\rangle \tag{2.48}$$

We stress the fact that, the formula is valid only at 2nd order in h. For more accurate formula one must include higher order terms.

We can now compute the (averaged) energy density of a gravitational wave as the t^{gw}_{00} component of $t^{gw}_{\mu\nu}$:

$$t^{gw}_{00} = \frac{1}{32\pi}\langle \dot{h}^{TT}_{ij}\dot{h}^{TT}_{ij}\rangle = \frac{1}{16\pi}\langle \dot{h}^2_+ + \dot{h}^2_\times\rangle \tag{2.49}$$

It is interesting to note that this expression is gauge invariant: this means that adding a gauge mode ξ^μ (see eq. (2.15) and (2.16)) the temporal average do not change.

With this in hands it is useful to compute the flux of energy radiated away by a GW. This will be helpful when computing energy balance in the coalescence of a binary BH. Starting from $\partial_\mu t^{\mu\nu} = 0$ we get:

$$\frac{\mathrm{d}E_V}{\mathrm{d}t} = -\int_V \mathrm{d}V\,\partial_i t^{0i} = -\int_S \mathrm{d}A\, n_i t^{0i} = -\int_S \mathrm{d}\Omega r^2 t^{0r} = \frac{r^2}{32\pi}\int_S \mathrm{d}\Omega \langle \dot{h}^{TT}_{ij}\dot{h}^{TT}_{ij}\rangle \tag{2.50}$$

where in the last equality we used $t^{0r} = \frac{1}{32\pi}\langle \dot{h}^{TT}_{ij}\partial_r h^{TT}_{ij}\rangle$ and $\partial_r h^{TT}_{ij}(t-r) = -\partial_t h^{TT}_{ij}(t-r) + O(1/r^2)$. We considered a surface element $\mathrm{d}A = r^2\,\mathrm{d}\Omega$ oriented in the radial direction. This is the required expression for the change of energy within a given volume V of space.

If h is computed in the quadrupole approximation, we can obtain a closed form expression for the energy radiated as a function of the total mass M of the system. Using the quadrupole formula and performing an integral over the solid angle, we get:

$$\frac{\mathrm{d}E_V}{\mathrm{d}t} = \frac{1}{5}\langle \dddot{M}^T_{ij}\dddot{M}^T_{ij}\rangle \tag{2.51}$$

[11] Indeed, introducing a scale for the observations and integrating out faster modes is rather natural from the point of view of renormalization group.

[12] See [5, ch 1.4] for the full computation.

where $M_{ij}^T = \int \mathrm{d}^3 y T^{00}(t,\mathbf{y})(y^i y^j - \frac{1}{2} r\delta_{ij})$ is the traceless quadrupole moment [13]. In SI units the luminosity acquires a prefactor $L_0 = \frac{c^4}{G} \simeq 10^{52}\mathrm{W}$. This is a huge amount of energy: gravitational radiation is very efficient in radiating away energy.

With a similar reasoning one can quantify the momentum carried away by a gravitational wave:

$$\frac{\mathrm{d}P^k}{\mathrm{d}t} = -\frac{r^2}{32\pi}\int_S \mathrm{d}\Omega \langle \dot{h}_{ij}^{TT} \partial^k \dot{h}_{ij}^{TT} \rangle \tag{2.52}$$

2.2.5 General parametrization for a GW

In GW data analysis community, it is important to choose an effective parametrization to the waveforms in the far (radiation) zone. The most common strategy exploits a *multipole expansion* of the strain h [11]. The general idea is to factor out the r dependence and express the remaining angular dependence as linear combination of the spherical harmonics basis [14]. The time dependence is then encoded in the linear coefficients in the basis expansion.

To be more specific, let us define a coordinate system (t,x,y,z) and express it in a spherical coordinate system (t,r,ι,φ). With these coordinates, a natural set of orthogonal basis for the spatial part of the tangent space is given by:

$$\hat{e}_r = \sin\iota\cos\varphi\ \hat{e}_x + \sin\iota\sin\varphi\ \hat{e}_y + \cos\iota\ \hat{e}_z \tag{2.53}$$

$$\hat{e}_\iota = \cos\iota\cos\varphi\ \hat{e}_x + \cos\iota\sin\varphi\ \hat{e}_y - \sin\iota\ \hat{e}_z \tag{2.54}$$

$$\hat{e}_\varphi = -\sin\varphi\quad \hat{e}_x + \cos\varphi\quad \hat{e}_y \tag{2.55}$$

In TT gauge, we have $h_{ij} = h_+\hat{e}_+ + h_\times\hat{e}_\times$, where, as in (2.26) and (2.27), we define:

$$\hat{e}_+ = \hat{e}_\iota \otimes \hat{e}_\iota - \hat{e}_\varphi \otimes \hat{e}_\varphi \tag{2.56}$$

$$\hat{e}_\times = \hat{e}_\iota \otimes \hat{e}_\varphi + \hat{e}_\varphi \otimes \hat{e}_\iota \tag{2.57}$$

Using the basis above, a wave can be expressed by the a complex function $h(t,r,\iota,\varphi)$ [15] as follows:

$$h = h_+ + ih_\times = \frac{M}{r}H(t,\iota,\varphi) + O(r^{-2}) = \frac{M}{r}\sum_{\ell=2}^{\infty}\sum_{m=-\ell}^{\ell} H_{\ell m}(t)\cdot {}^{-2}Y_{\ell m}(\iota,\varphi) \tag{2.58}$$

where ${}^{-2}Y_{\ell m}(\iota,\varphi)$ are the spin-weighted spherical harmonics of weight -2.

In equation (2.58) we performed an expansion in powers of r^{-1} and retains only the first order term (which is the usual power law decay for a 3D wave). This is well justified by the fact that in the far zone, r is (much) big compared with the source size (with scale M). The angular dependence of H is expanded in spherical harmonics. As the solution is complex, ordinary spherical harmonics are not suitable for the expansion and one must use spin-weighted spherical harmonics. The needs for -2 weight stems from the fact that h behaves as a spin-2 waves. The discussion is beyond of our scope.

[13] The need of the traceless quadrupole moment in the formula arises from the fact that in (2.51) only quantity in TT gauge appears. Thus when one projects the quadrupole formula (2.39) on TT gauge with $\Lambda_{ij,kl}$ tensor, the traceless quadrupole M^T appears in the formula.

[14] The multipole expansion here is different from the multipole expansion of the source. The latter is used to expand in series the energy momentum tensor $T_{\mu\nu}$ and it is used to solve the wave equation in the *near zone*. The wave multipole expansion refers to the radiation zone and it decompose a wave in the base of spherical harmonics.

[15] Here we define $h \equiv h_+ + ih_\times$. However this is a matter of convention: indeed the notation $h = h_+ - ih_\times$ is rather common in literature.

For reference, when $\ell = 2$:

$$ {}^{-2}Y_{2m}(\iota,\varphi) = e^{im\varphi} \begin{cases} \sqrt{\frac{15}{32\pi}}(\sin\iota)^2 & m=0 \\ \sqrt{\frac{5}{16\pi}}\sin\iota(1\pm\cos\iota) & m=\pm1 \\ \sqrt{\frac{5}{64\pi}}(1\pm\cos\iota)^2 & m=\pm2 \end{cases} \tag{2.59} $$

The expansion (2.58) is valid for every source of GW and it provides a very general and powerful method for writing a waveform. Almost any numerical waveform is written in this framework.

2.2.6 Waves in curved space time

So far we have considered a gravitational wave as a perturbation to flat spacetime. While this framework is useful and conceptually clear, it is not the most general one in which to develop the theory. Indeed, a GW can propagate also in curved spacetime. The crucial point here is to define what is the background and what is the perturbation. We can write the metric as sum of the two contribution:

$$ g_{\mu\nu}(x) = \bar{g}_{\mu\nu}(x) + h_{\mu\nu}(x) \qquad |h| << 1 \tag{2.60} $$

We call $\bar{g}_{\mu\nu}(x)$ the background metric and $h_{\mu\nu}(x)$ the perturbation (i.e. gravitational wave).

Of course, deciding what is $\bar{g}$ and h is a matter of convention. However, in some situations a physical motivated distinction can be performed. Suppose that the metric have two contributions from two timescales completely different from each other - say $t_A << t_B$. We could then call the slowly oscillating part (B in our notation) the background metric and, as long as it is small, we might denote the fast oscillating part (A) as the perturbation. The background will set an upper limit to the timescale t of the measurements we perform: $t_A << t << t_B$. On this time scale, the background metric can be deemed as static and the fast oscillating part actually behaves as a wave, being a rapid oscillation in an otherwise static background.

Once this distinction is made clear, it is possible to write every relevant geometric quantity as a sum of the two contributions from the high and the low frequency modes. The approach is useful not only because it is more general but also because it allows for introducing the back-action of GW on the background. Indeed the presence of a GW will deform the background $\bar{g}$: to account for this we must allow $\bar{g}$ to be more than flat spacetime. The calculation of the GW energy-momentum tensor (eq. (2.47)) is better done in this framework.

Once the mode separation is performed, one obtains a generalization of Pauli-Fierz equations in vacuum (2.18) to the non flat background. To proceed further, one should split Einstein equations into a two set of equations: one referring to high frequency (low timescale) modes and the other referring to low frequency (high timescale). The computations to solve the high frequency part are quite involved and will not be done here; however the result is simple:

$$ \nabla^\rho \nabla_\rho h_{\mu\nu} = 0 \tag{2.61} $$

where ∇^ρ is the covariant derivative induced by the background metric $\bar{g}_{\mu\nu}$. The interpretation is straightforward: as the wave is propagating into a curved space, the derivative operator must be adapted to this difference $\partial^\mu \to \nabla^\mu$. Apart from this difference, Pauli-Fierz equation are still valid.

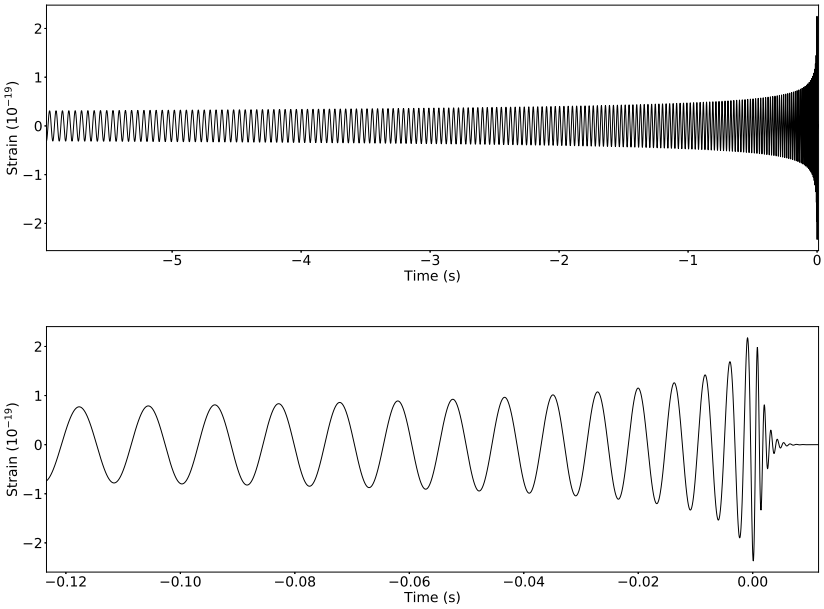

Figure 2.1: In the figure the h_+ polarization resulting from the coalescence of two non-rotating Black Holes. The two Black Holes have equal masses $m_1 = m_2 = 10\,\mathrm{M}_\odot$. In the upper panel the wave starts at $f_{gw} = 20\,\mathrm{Hz}$ (i.e at $t \simeq 6\,\mathrm{s}$ before merger). Many inspiral cycles can be observed; merged and ringdown are too short to be seen. The characteristic amplitude increase - the *chirp* - can be noted. In the lower panel, the wave starts at $f_{gw} = 80\,\mathrm{Hz}$ ($t \simeq 0.12\,\mathrm{s}$). Merger and ringdown stages can be resolved.

2.3 Compact objects coalescence

In this section we discuss some features on the GW generation by a coalescing system of compact objects. We will achieve this by the use different computational techniques, each adequate for describing the coalescence at different stages. As the computations are quite involved, only a sketch of them will be provided. The present works consider only coalescence of two BHs. If a neutron star takes part to the coalescence, the phenomenology can be very different (especially in the late stages of the coalescence) because of matter deformability, magnetic fields and other kind of phenomena. A coalescence with neutron stars won't be discussed here: for more information see [12].

The coalescence of two BHs is commonly divided into three main stages.

- *Inspiral*. This is the steady approach of the two object. The two BHs are far apart: their separation is much larger than their size. They follow circular (elliptical) orbits centered on their center of mass. The radius of orbits gradually shrinks because the system dissipates energy by means of gravitational radiation. To describe the early stages of this phase quasi-Newtonian theory is adequate (see section 2.3.1); however as the distance reduces more and more GR correction to the Newtonian theory must be taken into account (see 2.3.2) and a new formalism must be introduced. This is called post-Newtonian formalism.

- *Merger*. As the distance between the objects gets similar to the size of the BHs, the linear regime is abandoned and one needs to solve the full Einstein equations. This task must be performed numerically: it is the realm of numerical relativity (NR) [13]. Among other goals (see [14] for many examples of NR applications), NR tries to compute reliable waveforms for the

merger. Current state-of-the-art models for generating waveforms make use of the outputs of NR computations.

- *Ringdown.* The result of the binary black hole (BBH) merger stage is another BH. However the child BH is not on its "ground state" represented by the Kerr solution: some residual energy is stored into the perturbed modes of the metric. This energy is released trhough gravitational radiation emitted by the damped oscillations of the modes. The machinery to treat this stage is called Black Hole Perturbation Theory (see [6, ch 12]) and it is developed enough to have reliable waveforms for this stage. A proper theoretical understanding of ringdown and its observations are important since they might reveal some unknown feature of BHs and their horizon.

We report in figure 2.1 a waveform which shows the stages above.

A correct modeling of the waveform is crucial for testing GR in the compact objects coalescence scenario. Once a GW signal is detected, detailed theoretical predictions can be compared to the observed signal: see [3] for a number of GR tests done by the LIGO-Virgo collaboration with their catalog of observed GWs.

In what follows some of the common methods for computing the gravitational waveforms in the three stages are summarized: particular care will be devoted to the inspiral phase; merger and ring-down will only be sketched.

2.3.1 Newtonian inspiral

The early inspiral can be described by the following idealized situation. Two point masses are on a circular orbit and their motion is described by Newtonian gravity: as frequency and radius are related by Kepler's law and the motion is planar, only one parameter R (or f) is required to describe the state of the orbit and thus the energy of the system. The system has a non vanishing quadrupole moment and according to quadrupole formula (2.39) it emits gravitational waves. The emission of gravitational waves draws away energy from the binary system and results in a reduction of orbit radius (or increse in orbital frequency) which in turns changes the spectrum of the emitted radiation. Eventually all the initial potential energy will be radiated away and the two body will merge together.

We now analyse the system in two steps. In the first we will derive the radiation from two bodies in a circular orbit; in the second we will relate the radiation to the energy loss of the system. The goal of this section is to obtain an analytical formula for the gravitational wave radiation emitted by such system.

Gravitational radiation from a bynary system As usual a binary system of point masses m_1 and m_2 is equivalent (when seen from its center of mass frame) to a single particle in circular orbit with mass $\mu = \frac{m_1 m_2}{m_1+m_2}$. Let's consider a particle in a circular orbit on the $x-y$ plane:

$$\begin{aligned} x(t) &= R\cos(\omega_s t + \frac{\pi}{2}) \\ y(t) &= R\sin(\omega_s t + \frac{\pi}{2}) \\ z(t) &= 0 \end{aligned} \tag{2.62}$$

Our task is to compute the emitted radiation in quadrupole approximation. From mass quadrupole definition, the only non vanishing components are:

$$\begin{aligned} M_{11}(t) &= 3\mu R^2 \frac{1-\cos 2\omega_s t}{2} \\ M_{22}(t) &= 3\mu R^2 \frac{1+\cos 2\omega_s t}{2} \\ M_{12}(t) &= -\frac{3}{2}\mu R^2 \sin 2\omega_s t \end{aligned} \tag{2.63}$$

Computing the second time derivative and plugging the result into equations (2.43) and (2.44) one obtains:

$$h_+(r,\theta,t) = 4\mu R^2 \omega_s \cdot \frac{1}{r} \; \frac{1+\cos^2\theta}{2} \; \cos 2\omega_s t \tag{2.64}$$

$$h_\times(r,\theta,t) = 4\mu R^2 \omega_s \cdot \frac{1}{r} \; \cos\theta \; \sin 2\omega_s t \tag{2.65}$$

where it was set $\varphi = 0$ for convenience. The angle θ is usually named *inclination* (sometimes denoted by ι): it represents the angle between the normal to the orbital plane and the direction of view. r is the distance at which the wave is observed. We note that such system radiates GW at a frequency $f_{gw} = 2f_s$, which is twice the orbital frequency.

The power radiated per solid angle is readily computed with (2.50):

$$\frac{\mathrm{d}P}{\mathrm{d}\Omega} = \frac{2\mu^2 R^4 \omega^6}{\pi} \cdot \left[\left(\frac{1+\cos^2\theta}{2} \right)^2 + \cos^2\theta \right] \tag{2.66}$$

It is important to note that the modulation in θ is only due to the quadrupole nature of the radiation and do not depend on the details of the source.

As previously mentioned, it is possible to use Kepler's law to eliminate R in favor of $f_{gw} = \omega_{gw}/2\pi$. Kepler's law states:

$$\omega_s^2 = \frac{m}{R^3} \tag{2.67}$$

where $m = m_1 + m_2$. It is convenient to define the chirp mass:

$$\mathcal{M}_c = \mu^{\frac{3}{5}} m^{\frac{2}{5}} = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1+m_2)^{\frac{1}{5}}} \tag{2.68}$$

With the two expressions above the polarizations (2.64) and (2.65) turn into:

$$h_+(r,\theta,t) = \frac{4}{r}\mathcal{M}_c^{\frac{5}{3}} (\pi f_{gw})^{\frac{2}{3}} \; \frac{1+\cos^2\theta}{2} \; \cos(2\pi f_{gw} t) \tag{2.69}$$

$$h_\times(r,\theta,t) = \frac{4}{r}\mathcal{M}_c^{\frac{5}{3}} (\pi f_{gw})^{\frac{2}{3}} \; \cos\theta \; \sin(2\pi f_{gw} t) \tag{2.70}$$

The total radiated power averaged over the solid angle is:

$$P = \int_{4\pi} \frac{\mathrm{d}\Omega}{4\pi} \frac{\mathrm{d}P}{\mathrm{d}\Omega} = \frac{32}{5} \left(\pi \mathcal{M}_c f_{gw}\right)^{\frac{10}{3}} \tag{2.71}$$

Energy balance and full GW spectrum The equations above refers to the case in which the orbit radius (or orbital frequency) is constant. This is not the case since, as explained before, radiation

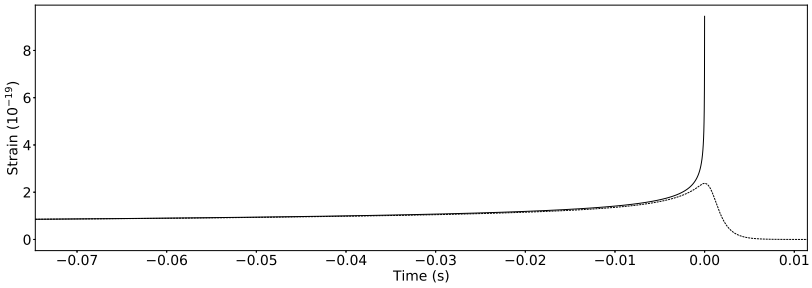

Figure 2.2: In solid line, it is shown chirp amplitude computed with quasi-circular Newtonian approximation ($h \sim \tau^{-\frac{1}{4}}$). Its divergence at merger is manifest as well as its failure to capture post merger emission. This is compared with the wave (dashed line) computed with state-of-the-art methods which reliably captures also merger and ringdown stages. Far from merger the two lines agree well: Newtonian approximation is accurate to describe early inspiral stages. Both waves are generated by a coalescence of two BHs with $m_1 = m_2 = 10\,\mathrm{M}_\odot$. They start when $f_{gw} = 95\,\mathrm{Hz}$.

emission carries away the total energy $E_{tot} = -\frac{1}{2}\frac{m_1 m_2}{R}$ [16]. As the system loses energy, the orbital frequency ω_s increase and thus the GW emission changes as well as the radiated power. The process gets faster (P in eq (2.71) is monotonic of $\omega_{gw} = 2\omega_s$) and eventually leads to coalescence of the two objects. The expressions above are valid as long as the timescale for the variation in orbital frequency is much smaller than the motion timescale $\dot{\omega_s} << {\omega_s}^2$: this means that the energy loss is slow. This regime is called *quasi-circular motion*. When the condition is not met, Newtonian approximation is not valid anymore and more powerful computation schemes (such as PN expansion) must be exploited.

The total energy can be written in terms of the frequency and the chirp mass:

$$E_{tot} = -\left(\frac{1}{32}\mathcal{M}_c^5 \omega_{gw}^2\right)^{\frac{1}{3}} \tag{2.72}$$

The energetic balance is readily obtained (with eq. (2.71)) and can be used to compute the rate of increase of frequency:

$$-\dot{E}_{tot} = P \quad \rightarrow \quad \dot{f}_{gw} = \frac{96}{5}\pi^{8/3}\mathcal{M}_c^{\frac{5}{3}} f_{gw}^{\frac{11}{3}} \tag{2.73}$$

Integrating the above equation one can find the dependence of frequency upon time:

$$f_{gw}(\tau) \simeq 151\,\mathrm{Hz}\left(\frac{M_\odot}{\mathcal{M}_c}\right)^{\frac{5}{8}}\left(\frac{1\,\mathrm{s}}{\tau}\right)^{\frac{3}{8}} \tag{2.74}$$

where $\tau = t_{coal} - t$ is the time to coalescence.

This equation, together with the two polarization (2.69) and (2.70) for a circular motion, fully specifies the waveform of the inspiral of two BHs. One must only pay attention to a time varying frequency. To account for this it is necessary to replace the time independent phase with:

$$\varphi(\tau) = \int_{t_0}^{t} \mathrm{d}t' \omega_{gw}(t') = -2\left(\frac{\tau}{5\mathcal{M}_c}\right)^{\frac{5}{8}} + \varphi_0 \tag{2.75}$$

The equation (2.74) has a divergence for $\tau \to 0$. This is due to the fact that Newtonian interaction diverges as the separation of two bodies approaches 0. This is clearly unphysical and marks the break

[16] We used the virial theorem for gravitational systems to compute the total energy: $2\langle E_{kin}\rangle = \langle E_{pot}\rangle$.

out of newtonian physics: for short distances and high velocities GR is much more accurate. To overcome this limit, while still using an analytical approach, is the goal of PN expansion and it will be outlined in the next section. As previously mentioned, quasi-circular motion breaks up when $\dot{\omega}_s \sim \omega_s^2$. Using previous equations this happens when $f_{gw} \simeq 5\,\mathrm{kHz}\left(\frac{M_\odot}{\mathcal{M}_c}\right)$. This is reached for $\tau \simeq 10^{-4}\mathrm{s}\left(\frac{\mathcal{M}_c}{M_\odot}\right)$ before merger. Of course, the problems in Newtonian physics arise at earlier times than this order of magnitude estimation.

In figure 2.2 a wave amplitude computed in quasi-circular approximation is compared to the waveform computed with the best available methods: the agreement in the early inspiral give way to the disagreement at later stages.

So far, we only focused on the case of circular orbits. However, this is not a great limitation. Detailed computations, which allow for ellyptical orbits, show that the orbital eccentricity decreases quickly during the inspiral. Heuristically, the eccentricity is "radiated away" faster than the potential energy. This means that, even for highly eccentrical orbits, by the time of late inspiral most of the orbits will have became circular. This simplifies dramatically the phenomenology: one can safely focus on the case of circular orbits without (much) loss of generality.

2.3.2 Late inspiral

In the late stages of the inspiral, the above Newtonian approximation breaks down. The non linearity of Einstein equation cannot be neglected anymore and we must develop more sophisticated methods to treat this case. A vast amount of work was done on the problem.

It turns out that it is too difficult to solve both for the system dynamics and the radiation at infinity. Thus the solution is obtained in three steps. First of all, the field equation are solved (approximately) in the near zone, with a method called Post-Newtonian approximation. The method is able to capture the two body dynamics and the metric nearby, but breaks down at larger distance. Second, a general solution in vaccum is obtained in the far zone (well outside the source) by expanding around the Minkowskian metric. Finally, the two solutions are matched together by setting appropriate boundary conditions in the region where the two solutions overlap.

PN expansion in the near zone PN expansion [15] tries to overcome the difficulties of the quasi-circular approximation. Indeed, the computation above has essentially two problems.

- It only considers the first term of multipole expansion. Multipole expansion (2.35) expands the source term in power of $\frac{v}{c}$ where v is the internal velocity of the source [17] : considering the leading term (quadrupole term) means that only slow moving sources are considered. While this is adequate for early inspiral, at later stages the speed of sources increase and thus higher and higher multipole terms shall be included.

- It treats background space-time as flat. The approximation is adequate as long as the two coalescing bodies are far apart (Schwarzschild solution is asymptotically flat) but as the separation decrease, a Schwarzaschild background must be considered.

The first problem can be overcome by including more and more multipole modes [18]; however this approach is doomed to fail: eventually a non flat background must be included. The problem here is

[17] In BBH coalescence the internal velocity of the source is the speed of the two bodies or (this is the same) the orbital frequency.

[18] Indeed this is the strategy employed for solving other problems, such as GW emission by an accelerated point mass, where the background can always be considered flat.

that the speed of the sources and the potential energy (which control flatness of space-time) are not independent. For bounded gravitational systems we can use the virial theorem to obtain the following relation between typical speed and typical size d of the source:

$$E_{kin} = -\frac{1}{2}E_{pot} \quad \rightarrow \quad \frac{1}{2}\mu v^2 = \frac{1}{2}\frac{\mu m}{d} \quad \rightarrow \quad \left(\frac{v}{c}\right)^2 \sim \frac{R_S}{2d} \tag{2.76}$$

where $R_S = 2m$ was used to simplify the expression. From the above equation we learn that, in order to be consistent, we must include also corrections to the flat space-time as we include corrections for high speed of the sources. The PN approach is the proper framework to do so.

The idea underlying the PN expansion is that the metric and the energy momentum tensor (and any other relevant quantity) must be expanded in series of a small PN parameter:

$$\epsilon \sim \left(\frac{R_s}{d}\right)^{\frac{1}{2}} \sim \frac{v}{c} \tag{2.77}$$

The ϵ expansion affects also the time derivative operator: $\frac{\partial}{\partial t} = O(v)\frac{\partial}{\partial x^i}$. The d'Alambertian operator $\Box$ is then approximated by a Laplacian: to the lowest order we obtain $\Box = -\frac{\partial^2}{\partial t^2} + \boldsymbol{\nabla}^2 = (1+O(\epsilon^2))\boldsymbol{\nabla}^2$. This means that any wavelike equation turns into a Poisson equation and the effects of gravity can be described by instantaneous potentials: the PN expansion absorbs retardation effects (due to finite speed of light) into Newton like instantaneous potentials. For this reason, PN approximation is only valid in the near zone, where retardation effects are small and can be captured by a series expansion. When $r\omega/c >> 1$ the PN approximation breaks down.

The PN can be successfully applied to describe the dynamics of a the late BBH inspiral. Once the usual separation from relative motion and center of mass motion is performed, the equation of motion for the BBH separation $\mathbf{x} = r\hat{\mathbf{n}}$ are written as:

$$\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = -\frac{m}{r^2}\hat{\mathbf{n}}\left[1 + O(\epsilon) + O(\epsilon^{3/2}) + O(\epsilon^2) + O(\epsilon^{5/2}) + \ldots\right] \tag{2.78}$$

$$= \mathbf{a_N} + \mathbf{a_{PN}} + \mathbf{a_{SO}} + \mathbf{a_{2PN}} + \mathbf{a_{SS}} + \mathbf{a_{RR}} + \ldots \tag{2.79}$$

In the second line we separate contribution to the acceleration from different kind of interactions. They are Newtonian interaction (N), first/second PN order ($PN/2PN$), spin-orbit interaction (SO), spin-spin interaction (SS) and radiation-reaction force (RR), due to the emission of gravitational radiation. We included terms up to 2.5PN but the framework is general and the expansion can be pushed forward.

A number of sophisticated techniques have been developed to compute the actual form of the acceleration terms: we refer the reader to [5, ch 5] for the details. The PN series is analytically solved up to 3.5 PN order, where ready-to-use analytical waveforms are available (see [16] for an example). It is interesting to note that at high PN order exotic effects, such as spin-spin interaction and spin-orbit interaction, arise from the Einstein equation and they bring to a rich phenomenology.

If the BHs spins and the orbital angular momentum are all parallel, the scenario does not change much from the spinless case. The orbit lies on a single plane orthogonal to the orbital angular momentum, as $\mathbf{L}$ does not change direction. Spins introduce some modulation in the rate of radiation emission and in the system dynamics. Heuristically, the case $\mathbf{s}_1 \cdot \mathbf{s}_2 < 0$ results in a more attractive interaction, which cause a shorter merger, and viceversa for the case $\mathbf{s}_1 \cdot \mathbf{s}_2 > 0$.

The non-aligned spins scenario has a more complex phenomenology [18], [17]. An important

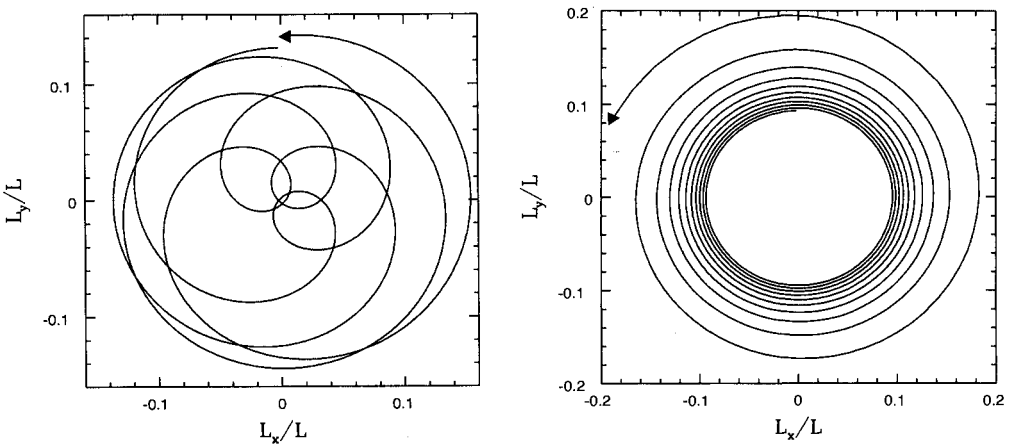

Figure 2.3: The effect of precession on the orbital angular momentum $\mathbf{L}$ of a BBH system which emits gravitational radiation. The total angular momentum $\mathbf{J}$ is on the z-axis, directed out of the page. BHs have mass ratio $\frac{m_1}{m_2} = 2$. In the left panel, we show the case in which the two BHs have equal spins, initially aligned with each other. In the right panel, only the larger BH has a spin. Figure adapted from [17].

quantity is the total angular momentum $\mathbf{J} = \mathbf{L} + \mathbf{S}$, where $\mathbf{L}$ is the orbital angular momentum (which includes PN corrections to the Newtonian expression and spin-orbit contribution) and $\mathbf{S} = \mathbf{s}_1 + \mathbf{s}_2$ is the total spin of the two BHs. $\mathbf{J}$ is always conserved in absence of gravitational radiation and, even if GW occurs, it never changes direction. However, single components of angular momentum (i.e. $\mathbf{L}, \mathbf{s}_1, \mathbf{s}_2$) change with time, giving rise to precession phenomena. The relevant equation at 2.5 PN order are:

$$\dot{\mathbf{S}} = \frac{1}{r^3}\left[\mathbf{L}_N \times \left(\frac{7}{2}\mathbf{S} + \frac{3}{2}(m_1 - m_2)(\mathbf{s}_2/m_2 - \mathbf{s}_1/m_1)\right) + 3(\mathbf{s}_1 \cdot \hat{\mathbf{n}})(\hat{\mathbf{n}} \times \mathbf{s}_2) + 3(\mathbf{s}_2 \cdot \hat{\mathbf{n}})(\hat{\mathbf{n}} \times \mathbf{s}_1)\right] \tag{2.80}$$

$$\dot{\mathbf{L}} = -\dot{\mathbf{S}} \tag{2.81}$$

where $\mathbf{L}_N = \frac{1}{2}\mu\mathbf{x} \times \mathbf{v}$ is the newtonian angular momentum. The cross product $\times$ yields a precessing solution for the time dependence of spins and orbital angular momentum. To an observer, the change of direction of $\mathbf{L}$ results in a time dependent modulation of amplitude, due to a time varying inclination angle ι. The phenomenon is complex and it is still not completely understood at the level of PN expansion: PN models for a precessing systems are not able to include all the effects seen in NR simulations [19]. Effects of precession on the orbital angular momentum are shown in figure 2.3

Matching the solutions in the intermediate zone The PN expansion provides a powerful tools for an approximate solution for the GR metric in the near zone. However it turns out to be inadequate for the far zone, for $r > \lambda\!\!\!^{-} = \frac{d}{v}$ (d being the source size) [19]. In the far zone, where one expects the metric to be nearly flat, one can use a method called post-Minkowskian approximation to take into account iteratively deviations from flat metric. With this machinery, the most general solution to the Einstein equation in vacuum can be written as a multipole expansion of the post-Minkowskian series. Post-Minkowskian expansion breaks down where the spacetime cannot be approximated as

[19] As in 2.2.3, $\lambda\!\!\!^{-}$ is the typical wavelenght of emission. It is given by $\lambda\!\!\!^{-} = \frac{c}{\omega} \simeq \frac{c}{v/d}$.

flat, i.e. close to the source, and it is valid only in the far zone. A number of multipole coefficients parametrise the general solution to the Einstein equation and their value must be set by imposing adequate boundary conditions close to the near zone.

PN and post-Minkowskian allows to solve Einstein equation in the near zone and in the far zone respectively. The two expansions have an overlapping region (intermediate zone) and the two solutions can be arranged so as to match together in the intermediate zone. This powerful formalism, due to Blanchet and Damour is the key ingredient for computing the metric perturbation (i.e. a GW wave) at infinity. The actual computations are quite involved and will not be presented here.

We limit ourselves to quote the result for the waveform computed in frequency domain. This will allow us to have a taste of what are the relevant quantities to determine the structure of the waveform. We write $h(f) = A(f)e^{i\Phi(f)}$. The phase $\Phi(f)$ of the wave is expanded as follows [20]:

$$\Phi(f) = \frac{3}{128\eta\nu^5}\left\{1 + \nu^2\left[\frac{3715}{756} + \frac{55\eta}{9}\right] - \nu^3\left[16\pi - \left(\frac{113}{3} - \frac{76\eta}{3}\right)\chi_s - \frac{113}{3}(m_1 - m_2)\chi_a\right]\right\} \quad (2.82)$$

where $\nu = (\pi M f)^{1/3}$ and $\chi_{s/a} = \frac{s_1 \pm s_2}{2}$ is the symmetric/antisymmetric combination of the spins. We denote by

$$\eta = \frac{m_1 m_2}{M^2} \quad (2.83)$$

the symmetric mass ratio. We see that at leading order, the phase depends on the chirp mass $\mathcal{M}_c$: indeed, we have that $\frac{1}{\eta\nu^5} = \frac{1}{(\pi\mathcal{M}_c f)^{5/3}}$. This was already predicted by the newtonian theory (see eq. (2.74)).

The dependence on η appears at the next-to-leading order and it was not predicted by the quasi-circular approximation.

The spin dependence enters at next-to-next-to-leading order as a combination of spins $\chi_{PN} = \chi_s + (m_1 - m_2)\chi_a - \frac{76}{113}$ Thus, in a first approximation the waveform depends on the *effective spin parameter*:

$$\chi_{eff} = \chi_s + (m_1 - m_2)\chi_a = \frac{1}{2}(1 + \sqrt{1 - 4\eta})s_1 + \frac{1}{2}(1 - \sqrt{1 - 4\eta})s_2 \quad (2.84)$$

This has deep implication on the measures of spins. By observing a GW, we can only measure the linear combination of spins $\chi_{PN} \simeq \chi_{eff}$ and the individual values of spins are not well constrained. As suggested in [21], the degeneracy might be eventually removed by observing the merger and ringdown. Indeed, they depend on final BH mass and spin, which depends on different combination of initial spins s_1, s_2. A precise detection of merger and ringdown phase could allow to disentagle the spin correlation and provide an accurate measure of the two individual spins.

Interestingly, the dependence on parameters of the PN series can be recovered by statistical methods. In [22] Ohme et al. performed a PCA expansion of the Fisher matrix of the mismatch. They found that the principal components of the Fisher matrix corresponded to the quantity that enters the PN expansion.

2.3.3 After inspiral: Merger and Ringdown

After the inspiral, the system enters the merger phase. During this phase the two black holes plunge into each others and their horizons merge together before settling down in a newly formed black hole. As already mentioned, during merger the two coalescing objects are so close that there is no hope to use a perturbative expansion to describe their motion: we enter the realm of numerical relativity

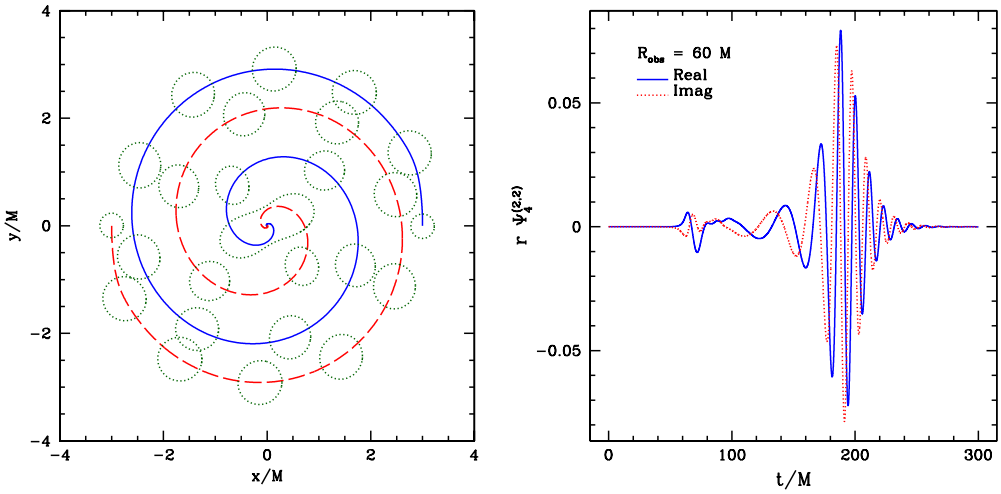

Figure 2.4: Results from a BBH NR simulation. In the left panel the trajectories of the two BHs as well as their horizons are represented. Note that in the last stage there is a single, highly deformed BH. In the right panel, the resulting real and imaginary part of the Weyl scalar computed at a distance $R_{obs} = 60M$ are reported. As is custom in NR, gravitational radiation is extracted by the metric computing the Weyl scalar $\Psi_4 = -\ddot{h}_+ + i\ddot{h}_\times$. Picture reproduced from [23]

where the full Einstein equation must be solved numerically. The output of such simulation is used to set some unknown quantities in the models that generate the full waveforms.

NR makes use of the initial value formulation of GR. Space-time is foliated into a set of space-like hypersurfaces Σ_t which have a time-like normal vector. The aim of any simulation is to compute the "time evolution" of the metric, given its value on an initial hypersurface. The choice of the metric on the initial hypersurface is not trivial since it must satisfy some conditions. In figure 2.4 the output of a NR simulation of a BBH merger is shown; the simulation is run using the "Einstein Toolkit" software [23].

As the two black merge together, the final BH is not described exactly by Schwarzschild (or Kerr) metric but rather by a perturbed state, in which some tiny oscillation modes are excited. Since an excited BH has a non vanishing quadrupole moment, it emits gravitational radiation by releasing the energy of the excited modes. Once all the modes are decayed, the BH eventually will settle down in its fundamental Schwarzschild (or Kerr) ground state. This phase is called ring-down.

The general metric decomposes into a Schwarzschild (or Kerr) $\bar{g}_{\mu\nu}$ metric plus a small perturbation $h_{\mu\nu}$:

$$g_{\mu\nu}(x) = \bar{g}_{\mu\nu}(x) + h_{\mu\nu}(x) \tag{2.85}$$

We see that, in the ring-down stage, the system is again in the linear regime and can be solved analytically.

Once Einstein equations are linearized, one can obtain describe the time evolution of the perturbation: this is done by the Teukolsky's equation ([24] is the original work). Usually the solution is expressed in terms of a set of basis function:

$$h_+ + ih_\times = \frac{M}{d}\sum_{lmn}\left[A_{lmn}e^{i\varphi_{lmn}}e^{it\omega_{lmn}}{}_{-2}S_{lmn} + c.c.\right] \tag{2.86}$$

where functions ${}_{-2}S_{lmn}$ are a basis of functions called *spin-weighted spheroidal harmonics*. The coefficients A_{lmn} depend on the perturbation imposed at $t = 0$. The frequencies ω_{lmn} at which the

system oscillates are characteristic of the system and are called *quasinormal modes*. This is the starting point of the *Black Hole perturbation theory* which develops the machinery to deal with the problem. With BH perturbation theory is possible a computation of radiation emitted during the ring-down phase of coalescence. The effect is included in the commonly used routine for computing the BBH waveform.

The field of study is called BH *spectroscopy* and, in analogy to atomic spectroscopy, is devoted to analyse the emission spectra of BHs. The hope is that, studying BHs emission spectra, one is able to infer the properties of the event horizon and possibly of the BH. For this reason, a detailed model for the ringdown together with a precise observation of a ringdown signal could allow for some GR tests or even for the discovery of new physics. The literature on the matter is vast: for more information see [25], [26], [27], [28] and [29] and references therein.

2.3.4 Surrogate models

A reliable data analysis on a detected signal (see section 2.4) requires accurate templates for gravitational waveforms. For this reason, a number of gravitational wave generator have been developed. They include accurate PN computation of the inspiral (with the matching procedure in the intermediate region) but, in order to push their accuracy up to the merger and ringdown, phenomenological inputs are given from NR simulation. For this reason they are also called *surrogate models*.

Basically two families of surrogate models were developed in the last decades:

- *Effective One-Body* (EOB): the model ([30], [31]) provides a mapping from the two body problem to a single body problem motion in an effective metric. The metric is described in terms of an effective Hamiltonian which combines PN expansion and resummation techniques to provide the maximum accuracy possible. The waveform is thus obtained by solving the dynamics under the effective Hamiltonian of the state variables p^i and x^i of the reduced problem. In the solution it is also included an energy balance to take into account the outward radiation flow. The late inspiral and the merger phase are incorporated with some phenomenological coefficients which fits the model to results from NR simulations. Although accurate, the models are quite slow to run. See [32] for an implementation.

- *IMR Phenomenological Models*: the class of models provides a waveform in frequency space. The waveform is built using PN expansion up to the last order to be known analytical. PN coefficients for the higher orders are fitted by looking at NR simulations. The model is known to be less accurate than EOB model but faster in its execution. See [33] for an implementation.

Software suite `lalsuite` [34] by the LIGO Scientific Collaboration provides a good implementation of a variety of methods for both families and it will be used in our work.

2.4 GW detection

We now turn to the matter of how a GW signal can be detected and how interesting physical information can be extracted from it. A smattering on the topic is useful to understand the context in which the present work is applied.

The standard method for detecting a GW signal relies on laser interferometry: the arrival of a GW changes the distance between test masses and this effect is observed with a change in interference pattern of a suitable laser system. A brief description is given in section 2.4.1. Since the GW wave

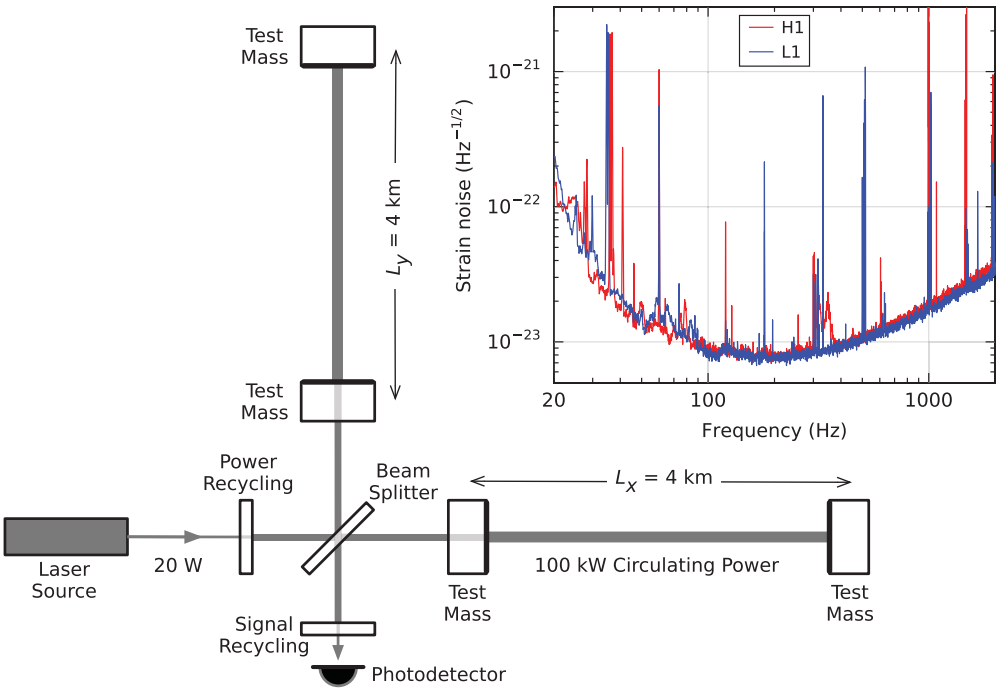

Figure 2.5: A scheme for the layout of Advanced LIGO interferometer. As more components were added to deal with technical issues, the set-up is more complicated than that describe in text. It is clear however the two arms configuration with beam splitter and mirrors at their edge. As mentioned in text light circulates in a Fabry–Pérot cavities and at their ends mirrors are placed. In the inset, the measured power spectral density $S_n(f)$ of the instrument noise (also called sensitivity window) is plotted for the two Advanced LIGO interferometers at Hanford, WA (H1) and Livingston, LA (L1), both in the US. The narrow lines corresponds to the resonances of some components of the interferometer. Figure adapted from [1].

signal is weak, the raw signal is dominated by noise. For this reason a thorough statistical analysis of the datastream must be performed both to detect a GW signal and to extract physical measures. We focus on this on sections 2.4.2 and 2.4.3.

2.4.1 Interferometers

The idea behind any GW detector is to measure the effect of a GW on (the variation of) the distance between two test masses. In fact according to (2.31), the effect of a GW is to stretches the proper separation between points of space-time. The most popular method to achieve this is to use a Michelson laser interferometer [20]. The experimental setup (see figure 2.5) is composed by a laser which travels along a two arms structure. Once generated a beam is passed through a beam splitter which separates the light into two beams; each beam travels toward a mirror and it is reflected back to the splitter. The two beams are then collected by a photodetector and their interference pattern is analysed.

An interferometer is able to measure with precision differences in the distance traveled by light in the two arms. Thus the physical idea behind GW detection is clear: as a GW arrives, it affects the length of the two arms differently and thus a measure of laser signal can reveal the properties of the GW. Usually the two arms are built to have similar length: this is to make the two arms as similar as

[20] Another alternative is provided by the so called resonant-mass detector, where the arrival of a GW excites the resonant modes of a carefully crafted object. This option is less accurate and more problematic and it has never been fully developed.

possible and to cancel out many common noise.

To make the statement quantitative, consider at time t two monochromatic beams of frequency ω_L that have recombined after traveled back and forth on the arms. The two arms have length L_x and L_y. Each beam entered the beam splitter (and thus started the travel in the arm) at time:

$$t_0^{(x/y)} = t - \frac{2L_{x/y}}{c} \tag{2.87}$$

This means that when the two beams entered the splitter, they had different phases. Since the phase is conserved during the propagation along the arm, at recombination the electric fields of the two can be written as follows [21]:

$$E_{1/2} = \mp\frac{1}{2}E_0 e^{-i\omega_L t_0^{(x/y)}} = \mp\frac{1}{2}E_0 e^{-i\omega_L t + 2ik_L L_{x/y}} \tag{2.88}$$

where of course $k_L = \omega/c$. The photodetector will measure the field $E_{tot} = E_1 + E_2$:

$$E_{tot} = iE_0 e^{-i\omega_L t + ik_L(L_x+L_y)} \sin[k_L(L_y - L_x)] \tag{2.89}$$

This means that any variation of the length difference of the two arms will correspond to a change in the power measured by the detector.

The next step is to link the difference $L_x - L_y$ to a GW signal. The calculation can be performed in two different (almost equivalent) coordinate frames: TT gauge and detector frame. As already discussed the TT gauge labels freely falling observer with constant coordinates. In our case, the relevant objects are the mirrors and the beam splitter [22]. By definition, in TT gauge the coordinates of free falling objects do not change as GW passes. However, the metric does change and this affects the way light propagates. In the detector frame the situation is the opposite. ST is considered flat - thus light propagates always in the same way - and the coordinates of the arms are time varying. Of the course the two approaches agree on the proper length of the arms.

Strictly speaking, making computations in the detector frame is not correct because space-time is not flat on scales of GW wavelength λ_{gw}. However, at smaller length scales the approximation above works well. Thus computations regarding the interaction detector and radiation can be performed in detector frame as long as:

$$\frac{\omega_{gw} L}{c} << 1 \tag{2.90}$$

The advantage of detector frame is that Newtonian intuition about forces and flat space-time can be used: a gravitational wave can be approximated by a Newtonian force which stretches each arm of the interferometer according to (2.31).

In what follows we will develop calculations in the detector frame [23]. We consider a wave with only plus polarization $h_+(t) = h_0 \cos(\omega_{gw} t)$. We start from (2.31) and compute the time variation of the x-arm lenght ξ_x in terms of its unperturbed lenght L_x (i.e. measured in a flat spacetime):

$$\ddot{\xi}_x = \frac{1}{2}\ddot{h}_+ \xi_x \simeq \frac{1}{2}\ddot{h}_+ L_x \quad \rightarrow \quad \xi_x(t) = L_x + \frac{h_0 L_x}{2}\cos\omega_{gw} t \tag{2.91}$$

[21] The sign difference arises from proper modelling of the beam splitter. In the reflection, a beam gets a π phase shift (resulting in a $-$ sign). The same does not happen for the beam which is transmitted by the beam splitter.

[22] Strictly speaking those are not in free fall since they are suspended against Earth gravity. However as long as only horizontal degrees of freedom are considered a free fall approximation can hold.

[23] For typical interferometers $L \sim 5\,\text{km}$ and $\omega_{gw} \sim 100\,\text{Hz}$ giving $\frac{\omega_{gw} L}{c} \sim 10^{-3}$. Thus the condition (2.90) is met and calculations in the detector frame are justified.

And for y-arm the + must be replaced by a − sign. In this equation we read again a well known property of the strain h: the amplitude of a GW corresponds to the ratio between the variation of length produced by the wave and the length itself: $h(t) \sim \frac{\Delta L}{L}$

We can now compute the time-varying difference in length of the two arms:

$$\Delta L(t) = \xi_y(t) - \xi_x(t) \simeq L_y - L_x + h_0 L \cos \omega_{gw} t \tag{2.92}$$

where we used $L = \frac{L_x + L_y}{2}$. Usually, we build the interferometers such that $L \simeq L_x \simeq L_y$.

If we plug (2.92) into equation (2.89) we obtain the desired formula which relates the field observed at the detector with the frequency of the incoming gravitational wave:

$$E_{tot} = iE_0 e^{-i\omega_L t + 2ik_L L} \sin[\varphi_0 + \Delta\varphi(t - L/c)] \tag{2.93}$$

where

$$\Delta\varphi(t) = h_0 k_L L \cos \omega_{gw} t \tag{2.94}$$

$$\varphi_0 = (L_y - L_x) k_L \tag{2.95}$$

$\Delta\varphi(t)$ is the phase shift between the two beams when a gravitational wave of frequency ω_{gw} and only plus polarization enters the detector: this is the effect that is measured by a detector. Note that in (2.93) $\Delta\varphi$ is evaluated when the laser hits the mirror (time $t - L/c$) and not when it exit the arm: this provides a better point estimation of the time integrated effect of the wave on the laser over the time $2L/c$, which the beams spend in the arms.

As previously said, the formula above is only an approximation for small values of $\omega_{gw} L/c$. For higher values one must do the computation above in TT gauge. The result is only slightly different:

$$\Delta\varphi(t) = h_0 \, k_L \, L \, \mathrm{sinc}(\omega_{gw} L/c) \cos \omega_{gw} t \tag{2.96}$$

where $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$. In TT framework is also possible to compute the optimal length of the arms for maximizing the magnitude of $\Delta\varphi(t)$. This results to be $L \simeq 750\,\mathrm{km} \cdot \left(\frac{100\,\mathrm{Hz}}{f_{gw}}\right)$: for the frequencies of interest this is a huge distance that can not be achieved in ground based observatories. For this reason in each arm a Fabry–Pérot cavity is implemented: this expedient is useful for overcoming the difficulty posed by a 750 km long arm.

Since a GW has a very weak amplitude, an interferometer must measure distances very precisely: for $h \sim 10^{-19}$, it must be able to resolve a variation in length of $\delta L \sim hL \sim 10^{15}$m. This is an incredible precision and requires to implement sophisticated experimental setup and have great control over possible sources of noise. See [35] for more details on the experimental challenges related to the interferometers.

Angular sensitivity So far, we have considered the effect on the interferometer of the only plus polarization. Of course, in the general case, the interferometer will rather observe a linear combination of plus and cross. The combination coefficients depend on the orientation of the GW polarizations with respect to the detector. A study of it will give the angular sensitivity of the detector as well as a quantitative indication of what the detector actually measures.

Before starting any computation we make the approximation that the detector size is small compared to the wavelength of the incoming wave. This is useful to remove spatial dependence of the incoming

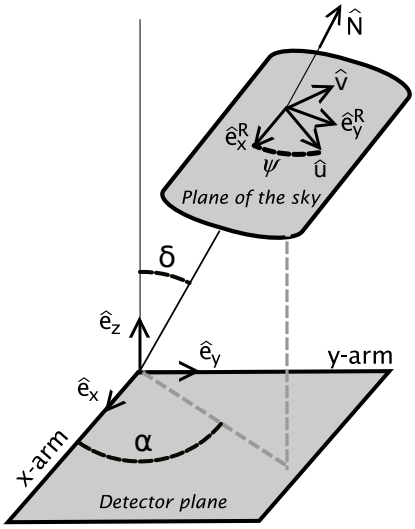

Figure 2.6: Relation between the detector frame and the TT gauge (also plane of the sky) in which wave is propagating is represented. As in text the two frames are denote by basis $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ and $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}$. Figure reproduced from [8].

GW: $h(t, \mathbf{x}) \simeq h(t, \mathbf{x}_{detector}) \equiv h(t)$. The experimental input is:

$$h(t) = \sum_{P=+,\times} D^{ij} H^P_{ij} h_{P\,ij}(t) = F_+(\hat{\mathbf{n}}) h_+(t) + F_\times(\hat{\mathbf{n}})\; h_\times(t) \tag{2.97}$$

where H^P_{ij} are given in equations (2.26) and (2.27) and $F_{+/\times} = D^{ij} H^{+/\times}_{ij}(\hat{\mathbf{n}})$. The functions $F_{+/\times}(\hat{\mathbf{n}})$ are called detector patterns functions and denoted detector sensitivity in different directions. Their specific form depends on the particular kind of detector.

For the computation of pattern functions two orthonormal coordinate systems are defined: the detector frame denoted by $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$ and the source frame denoted by $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{n}}$. In the detector frame, the arms of the interferometers are along the $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ directions and the wave propagates along the $\hat{\mathbf{n}}$ direction. This is shown in figure 2.6.

It can be proved that in the case of an interferometer, the detector tensor has the form

$$D^{ij} = \frac{1}{2}(\hat{\mathbf{x}}_i \hat{\mathbf{y}}_j - \hat{\mathbf{y}}_i \hat{\mathbf{y}}_j) \tag{2.98}$$

From here, a straightforward (even though boring) computation leads to the antenna patterns for an interferometer:

$$\begin{aligned} F_+(\delta, \alpha, \Psi) &= \frac{1}{2}(1 + \cos^2\delta)\cos 2\alpha \cos 2\Psi - \cos\delta \sin 2\alpha \sin 2\Psi \\ &= F_+(\delta, \alpha, 0)\cos 2\Psi - F_\times(\delta, \alpha, 0)\sin 2\Psi \end{aligned} \tag{2.99}$$

$$\begin{aligned} F_\times(\delta, \alpha, \Psi) &= \frac{1}{2}(1 + \cos^2\delta)\cos 2\alpha \sin 2\Psi + \cos\delta \sin 2\alpha \cos 2\Psi \\ &= F_+(\delta, \alpha, 0)\sin 2\Psi + F_\times(\delta, \alpha, 0)\cos 2\Psi \end{aligned} \tag{2.100}$$

where δ and α are the polar angle of vector $\hat{\mathbf{n}}$ in the detector frame. As they locate the source position in the sky, they are called, following the convention in astronomy, *declination* and *right ascension*. Ψ is called *polarization angle* and is defined as $\Psi = \hat{\mathbf{x}} \angle \hat{\mathbf{u}}$. It is interesting to note that the Ψ dependence

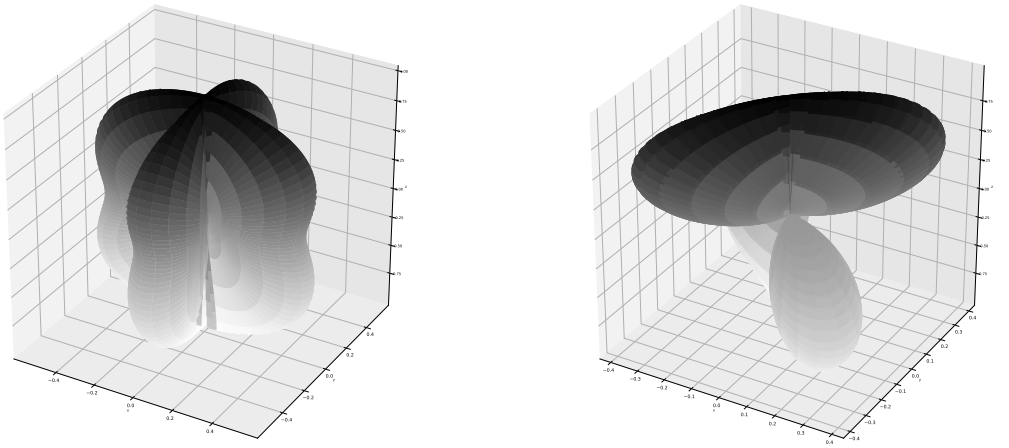

Figure 2.7: Antenna pattern function for the two polarizations: $F_+(\theta, \varphi)$ on the left panel and $F_\times(\theta, \varphi)$ on the left. The three dimensions represent spatial directions; the magnitude of the antenna pattern in a certain direction is proportional to the distance to the plotted function in that direction. A good sensitivity along all directions can be observed for the plus polarization.

of detector pattern is only due to the properties under rotation of tensor h_{ij} and do not depend on the specificity of the detector.

In figure 2.7 the angular dependence of the antenna pattern for the two polarizations are reported (Ψ is set to zero). The sensitivity peak is at $\delta = 0, \pi$ when wave incides orthogonally over the interferometer arms. It can be noted that an interferometer has a good sensitivity on many directions: this feature is useful since it allows the detector to receive signals from a bigger region in the sky; on the other hand this limit the ability of detecting the localization of the source. This is among the reasons why an array of interferometers is more effective than a single one.

Sources of noise Each source of noise mimics or mask a true GW signal. The window in which the noise is low enough to allow for detection is called sensitivity window. For a typical interferometer it spans the range from 10 Hz to 10 kHz (see figure 2.5). To reliably detect a wave, the different sources of noise that affect the interferometers must be known and its impact must be assessed: this task is commonly performed by the maintainers of current interferometers (see eg. [36]).

The topic is vast and we will not treated in this work. However, below we give a brief list of the most important sources of noise in the detector and of the strategies adopted to cope with them.

- *Seismic noise*: any noise generated by the motion of the ground. Typically it is a low frequency (up to 10Hz) signal and it can be generated by any ground motion such as sea waves, earthquakes, wind and human activities. In the low frequency it can be 10 orders of magnitude higher than the target sensitvity. To reduce its magnitude, mirrors are suspended by a complicated system of pendulums mounted on multistage active platforms. The pendulums inertia provides a effective filter for seismic noise.

- *Thermal noise*: any noise generated by random thermal motion of atoms in the mirrors and their

suspensions. The levels of thermal noise are analytically computed and this makes its treatment easier. Thermal noise is more effective in the range 10 Hz to 500 Hz within the sensitivity window; its magnitude however is below the current sensitivity target.

- *Gas noise*: noise generated by the interaction between laser and environment gas molecules. This is mainly due by scattering. To limit this noise, all the optics is placed in vacuum chambers and pressure is maintained below 10^{-6}Pa. However a residual noise is still present and must be studied. Models for laser-gas interaction were developed and tested and are effectively applied in the noise for model estimation.

- *Shot noise*: noise arising from the quantized nature of photons. When a low number of photons reaches the photodetector, fluctuations on the arrival rate will result in fluctuations in detected brightness. The phenomenon can be modeled as a Poisson process. Currently, shot noise affects the sentitivity above around 100 Hz.

- *Quantum noise*: noise driven by fluctuations of the vacuum field in the optical component of the detector. Vacuum can interact with the laser; however this is expected to be significant below 40 Hz and thus it is not a concern in present detector setup.

- *Scattered light*: noise arising from small imperfection in the mirrors. They can scatter light towards other components; if scattered light recombines with the signal, small spurious oscillations in the signal will occur.

- *Laser and electronics noises*: noises which arise from unideal laser or electronic components.

2.4.2 GW data analysis: detecting a wave

A typical GW signal is a few order of magnitude lower than the detector noise. For this reason, the extraction of an (eventual) signal from raw data requires a sophisticated statistical analysis.

The first step detects the presence of a GW signal buried in the detector output. This is done by comparing many GW signal templates with the raw signal and trying to judge whether there is a match between the signal and one among the templates. This procedure is called *matched filtering* and must be performed *on-line*, i.e. as soon as the data are acquired. Performing matched filtering on a single detector doesn't give enough statistical evidence for a GW signal detection. For this reason, a GW detection is claimed only if at least two detectors are deemed to have registered a GW signal: this is called a coincidence. We discuss it in the current section.

Once we are reasonably sure to have recorded a gravitational wave signal, physical information must be recovered from it. This is the second stage of statistical analysis, called *parameter estimation* (PE). It makes use of Bayesian statistics to recover a probability distribution $p(\theta|\mathcal{D})$ for all the relevant physical parameter θ required to fully describe the incoming signal. This is not a trivial task and is discussed in the next section.

We start with an hypothesis about the nature of GW detector signal and we split the raw detector output $s(t)$ into two parts: (eventual) GW signal $h(t)$ and noise $n(t)$.

$$s(t) = h(t - t_a) + n_t \tag{2.101}$$

where t_a is the time of arrival of the GW in the detector. The noise here is regarded as a stochastic process, i.e. n_t is a random variable at each time t. It is common to consider the case of gaussian

noise, where the probability distribution function of the noise magnitude n_t is a gaussian. Actual noise is only approximately gaussian and has a more complex structure. As non-gaussian noise in detectors is not completely understood and more difficult to model, a gaussian noise for n_t is usually adopted: this makes the data analysis easier and it is commonly used in practical applications. Another crucial assumption is that noise and GW signal are factorized and don't affect each other; formally, $p(n_t|h) = p(n_t)$. In this framework, the goal of matched filtering is to detect in $s(t)$ statistical evidence for a component $h(t)$.

Unfortunately, the usual situation in interferometers is that $|h(t)| << |n_t|$. In this situation a direct observation of the signal is impossible because any signal is washed out by noise. However, a signal can be recognized if one is looking for a known template. More precisely, by convolving the observed signal and the template, noise can be averaged out and only the signal terms dominates:

$$\frac{1}{T}\int_0^T \mathrm{d}t s(t)h(t) = \frac{1}{T}\int_0^T \mathrm{d}t(h^2(t) + n(t)h(t)) \simeq \frac{1}{T}\int_0^T \mathrm{d}t h^2(t) \tag{2.102}$$

where the last equality holds if T is large enough to allow for the noise average to go to zero. This technique is called matched filtering.

In order to perform matched filtering, the first step is to characterize statistical properties of the noise n_t. Noise is modeled as a time series (i.e. a sequence of random observations whose value depends on the earlier ones) and it is fully characterized by autocorrelation function $\kappa(t_1, t_2)$. If noise is zero mean at every time, autocorrelation has the following form:

$$\kappa(t_1, t_2) = \langle n^*_{t_1} n_{t_2} \rangle \tag{2.103}$$

where $\langle \cdot \rangle$ denotes average over different noise realizations. Autocorrelation measures the correlations of the two random variables n_{t_1} and n_{t_2} and is a function of the times at which noise is evaluated. If the noise is "invariant under time translations" it is called *stationary* and $\kappa(t_1, t_2) = \kappa(\tau = |t_2 - t_1|)$. This condition is only approximately valid in real detectors but it is widely used because of its simplicity. Under the stationariety condition, the noise can be fully described by the Fourier transform of its autocorrelation function $\kappa(\tau)$. This is called noise *power spectral density* (PSD):

$$S_n(f) = \frac{1}{2}\int_{-\infty}^{+\infty} \mathrm{d}\tau\ \kappa(\tau) e^{i2\pi\, f\, \tau} \tag{2.104}$$

The meaning of $S_n(f)$ can be understood by writing autocorrelation in Fourier space:

$$\langle \tilde{n}(f')\tilde{n}^*(f)\rangle = \frac{1}{2} S_n(f)\delta(f - f') \tag{2.105}$$

This means that $\frac{1}{2}S_n(f)$ is the non vanishing matrix element of the autocorrelation in Fourier space. Note that $\langle \tilde{n}(f')\tilde{n}^*(f)\rangle$ is diagonal (i.e. $\sim \delta(f - f')$) because of the stationariety assumption. In figure 2.5 the PSDs for a LIGO interferometers are represented.

Once the noise of the detector is fully characterized, we can focus on the problem of assessing whether a given detector output $s(t)$ contains a GW signal. Like in equation (2.102), let

$$c(\tau) = \int_{-\infty}^{+\infty} \mathrm{d}t\ s(t)q(t + \tau) \tag{2.106}$$

be the convolution between the signal. Function $q(t)$ called filter and encodes the knowledge of the

function $h_\star(t)$ which we are looking for in the signal. We then define the *signal-to-noise ratio* (SNR) ρ as:

$$\rho = \frac{S}{\sqrt{N^2}} = \frac{\bar{c}}{\sqrt{\langle (c-\bar{c})^2 \rangle}} \tag{2.107}$$

where $\bar{c} = \langle c(\tau) \rangle$ is the average of the convolution. Here S is the expected value of c when a signal $h(t)$ (which doesn't need to be the same of $h_\star(t)$) is present and N^2 is the variance of c in the case of no GW signal. The template function $h_\star(t)$ is likely to be in the signal if ρ is high.

So far the filter function $q(t)$ was left undefined. We now derive its form by maximizing the signal-to-noise ratio with respect to the filter. Suppose that in the signal is present the template wave $h_\star(t)$. Then we can compute S and N:

$$S = \int_{-\infty}^{+\infty} \mathrm{d}f \; \tilde{h}_\star(f) \tilde{q^*}(f) e^{-2\pi i f(\tau - t_a)} \tag{2.108}$$

$$N^2 = \int_{-\infty}^{+\infty} \mathrm{d}f \; \frac{1}{2} S_n(f) |\tilde{q^*}(f)|^2 \tag{2.109}$$

In the expression for S the noise part in the signal vanishes because n_t has zero mean. It is natural to define a scalar product between waveform:

$$\langle a, b \rangle = 4\mathrm{Re} \int_{-\infty}^{+\infty} \mathrm{d}f \; \frac{\tilde{a}(f)\tilde{b}^*(f)}{S_n(f)} \tag{2.110}$$

This is a measure of the overlap between waves made in frequency space where the noisier component are less important.

With the scalar product above ρ can be written as:

$$\rho^2 = \frac{\langle \tilde{h}_\star e^{-2\pi i f(\tau - t_a)}, S_n q \rangle}{\langle S_n q, S_n q \rangle} \tag{2.111}$$

From here it is manifest that the value for the filter q which maximizes SNR is:

$$q(f) = \gamma \frac{\tilde{h}_\star(f) e^{-2\pi i f(\tau - t_a)}}{S_n(f)} \tag{2.112}$$

where γ is an unrelevant multiplicative constant. q is called *optimal filter*.

With the optimal filter, we can address the problem of detecting a physical signal into the detector signal. A bank of precomputed templates is set up and for each short time series the SNR ρ is computed using the optimal filter. As the SNR exceeds a threshold value (typical values are 8, 16, 32) a signal detection claim is done. If this is confirmed by other interferometers, then a signal is deemed to contain as physical inputs and a further statistical analysis is performed.

It is important to stress the fact that it is important to have a reliable, highly accurate templates. Without them a detection is really difficult and the risk of not recognizing physical signals is high. For this reason modeling accurate waveform is really important and it is still a lively research branch.

2.4.3 GW data analysis: getting physical information

As a detection claim is done, we hope to determine the physical parameters of the system which have generated the wave. Of course we cannot compute their actual value but only an estimation of how probable each parameter is. As already mentioned, the procedure is called *parameter estimation* (see

[37] [38], [39] or [40]).

We denote by ϑ the set of parameters that we want to give an estimate of. For a BBH coalescence there are 15 of them:

- *Componenent masses*: masses m_1, m_2 of the two coalescing BHs.
- *Componenent spins*: spins $\mathbf{s_1}$, $\mathbf{s_2}$ of the two coalescing BHs.
- *Arrival time*: time t_0 at which the wave entered the detector
- *Source position*: declination δ and right ascension α of the source as well as its luminosity distance d_L.
- *Polarization angle*: angle Ψ setting a rotation of the polarization basis around the propagation direction (see eqs. (2.99) and (2.100))
- *Inclination*: inclination angle between orbital angular momentum and direction of propagation
- *Reference phase*: reference phase φ_0 at the arrival time t_0. [24]

Note that the angles α, δ and Ψ are those defined in equations (2.99) and (2.100): they parametrize the different orientation between the TT gauge of the wave and the detector frame. The resulting detected signal in Fourier space is parametrized as follows:

$$\hat{s}(f,\vartheta) = [F_+(\delta,\alpha,\Psi)\,\hat{h}_+(f,\vartheta') + F_\times(\delta,\alpha,\Psi)\,\hat{h}_\times(f,\vartheta')]e^{2\pi i f t_0} \tag{2.113}$$

The dependence on the remaining parameters $\vartheta' = (m_1, m_2, \mathbf{s_1}, \mathbf{s_2}, d_L, \iota, \varphi_0)$ must be included in the waveform model and it is an output of the theory.

Posterior distribution After having looked at the data $\mathcal{D}$, it is natural to encode all our beliefs on the parameters ϑ on a probability distribution function $p(\vartheta|\mathcal{D})$. Bayesian inference provides a natural way to compute the required distribution [37]. In this framework any prior knowledge about the parameters to infer is encoded by a probability distribution $p(\vartheta)$ called prior distribution. A model H for the observation is then built as a form of likelihood function $p(\mathcal{D}|\vartheta, H)$: this encodes how probable is the observation $\mathcal{D}$, if it was generated by the parameters set ϑ.

Combining the two ingredients, one can take advantage of Bayes rule [25] to obtain a model $p(\vartheta|\mathcal{D})$ for the inferred parameters. This is the *posterior distribution*:

$$p(\vartheta|\mathcal{D}, H) = \frac{p(\mathcal{D}|\vartheta, H)p(\vartheta)}{p(\mathcal{D}|H)} \tag{2.114}$$

where the $p(\mathcal{D}|H)$ is called evidence and it is defined by:

$$p(\mathcal{D}|H) = \int \mathrm{d}\vartheta p(\vartheta, \mathcal{D}|H) = \int \mathrm{d}\vartheta p(\mathcal{D}|\vartheta, H)p(\vartheta) \tag{2.115}$$

[24] This is an arbitrary phase shift that only changes the value of the phase. Modulo 2π, φ_0 is the azimuthal angle of the direction of propagation. Equivalently it is the reference value (i.e. at a given initial time) of the angular variable of the effective one body problem of coalescence. The three definitions are equivalent. Together with ι, φ_0 appears in the harmonics decomposition of strain h.

[25] As a reminder, Bayes rule for two events A and B is as follows:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

. This arises from definition of conditional probability $p(A|B) = \frac{P(A,B)}{P(B)}$.

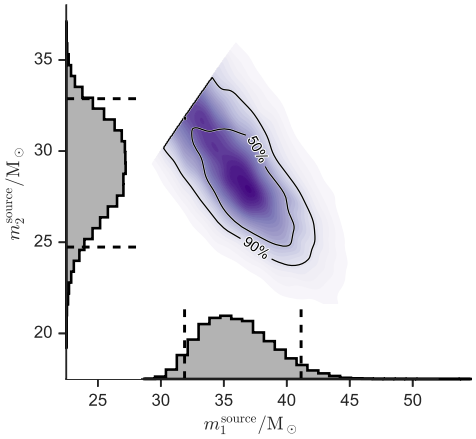

Figure 2.8: The (marginalized) posterior for the signal of GW150914. In the two axis, the histograms represents the one-dimensional marginalized probability distribution for the BHs mass values. The two-dimensional contour plot represents the 90% and 50% credible regions for the two-dimensional marginalized distribution $p(m_1^{source}, m_2^{source}|\mathcal{D})$. The Bayes factor for the model is $\log \mathcal{B} = 288.7 \pm 0.2$ (with model EOBNR). We define the Bayes factor as $\mathcal{B} = \frac{Z}{Z_n}$ where Z_n is the evidence of a model with only random noise. Figure adapted from [40].

The evidence is a normalization constant for the posterior and it measures the probability that the data are explained by the model choice H. The evidence can be used to make (Bayesian) model selection: if two competing models H_1 and H_2 try to explain the data, it is wise to pick the one with higher evidence. In the GW context, model selection can be useful to decide the physical origin of the signal: different sources yield different models; however only the "true" source gives a high value for the evidence. For this reason every posterior must come with (an approximated) value of model evidence.

In order to make some statement about the data, a model must be specified. Specifically this is done by defining a functional form for the likelihood $p(\mathcal{D}|\vartheta, H)$. In the context of GW data analysis the data is the signal received by the detector in eq. (2.101): $\mathcal{D} = s(t)$. The choice for the likelihood is as follows:

$$p(\mathcal{D}|\vartheta, H) \sim \cdot \exp\left[-\frac{1}{2}\langle s - \bar{s}(\vartheta), s - \bar{s}(\vartheta)\rangle\right] \tag{2.116}$$

where $\bar{s}(\vartheta)$ is defined in (2.113) and $\langle \cdot\,, \cdot \rangle$ is the standard scalar product between waves (2.110). We strees again the necessity of having reliable and accurate models for $h(\vartheta)$: without them, the likelihood term would not be able to describe data correctly. We note that an overall multiplicative constant (which do not depend on ϑ) do not affect the posterior. Indeed, a constant in the likelihood (or in the prior) is canceled by the same multiplicative factor in the evidence.

The prior reflects prior knowledge about the system. If nothing is known in advance, as it is the case in GW detection, usually an *uninformative* prior is used. Uninformative priors put equal probability mass on each part of the parameter space: a flat distribution is a common choice. The choice of the prior is as important as the model specification: the posterior depends on the prior and great care must be taken to identify a proper prior. A wrong choice can result in unmeaningful posteriors.

Monte Carlo inference The goal of Bayesian analysis is to compute the posterior distribution (2.114) together with the evidence (2.115) of the model. Writing down the posterior is an easy task; the hard task is to make from it some statement about the data and model effectiveness. This process is called *inference*. Giving an estimate for ϑ with errorbars, computing correlation between

variables, marginalizing over some variables or performing model selection are all common inference problems. Usually, exact inference is impossible to performs and thus approximate way must be developed.

A strategy called Monte Carlo inference is usually exploited. Monte Carlo inference consists in drawing a set $\{\bar{\vartheta}_i\}_{i=1}^N$ of N samples from the posterior $\bar{\vartheta}_i \sim p(\vartheta|\mathcal{D}, H)$. They can be used to compute an approximation of the quantities of interests as well as of the evidence.

For example, if one wants to compute the probability $p(\vartheta \in A)$ that ϑ lies in a given set A, Monte Carlo approximation works as follows:

$$p(\vartheta \in A) = \int_A \mathrm{d}\vartheta \; p(\vartheta|\mathcal{D}, H) \simeq \frac{|\{\bar{\vartheta}_i\}_{i=1}^N \cap A|}{|A|} \tag{2.117}$$

where $|\cdot|$ denotes the cardinality of a set. The expression above has a straightforward interpretation: $p(\vartheta \in A)$ is approximated by the ratio between the random samples which satisfy $\bar{\vartheta}_i \in A$ and the total number of samples. The hard task of evaluating a multidimensional integral is reduced to a trivial counting task (plus a non trivial sampling from the posterior).

There are a number of standard methods for effectively creating a set $\{\bar{\vartheta}_i\}$ of samples from the posterior. The majority of them relies on a standard technique called *Markov Chain Monte Carlo* (MCMC), or Metropolis-Hastings algorithm, [41] which was first developed for simulating a many-body system [42] and rapidly spread to many other fields. The basic idea (see e.g. [43, ch 23]) is to perform a random exploration of the parameter space. At each iteration t, the algorithm tries to update the position $\bar{\vartheta}^t$ of an agent. It draws randomly a new position $\bar{\vartheta}^{t+1}$ from a proposal distribution $Q(\bar{\vartheta}'|\bar{\vartheta})$. The proposed move is accepted with probability $p = \mathtt{min}(1, \alpha)$ with:

$$\alpha = \frac{Q(\bar{\vartheta}|\bar{\vartheta}')p(\vartheta'|\mathcal{D}, H)}{Q(\bar{\vartheta}'|\bar{\vartheta})p(\vartheta|\mathcal{D}, H)} \tag{2.118}$$

Following this simple prescriptions the algorithm is guaranteed to generate samples $\bar{\vartheta}^t$ drawn from the posterior distribution. Crucially, the acceptance α does not depend on an overall normalization of the posterior. The hard task of computing the evidence is bypassed and only the ability to evaluate $p(\mathcal{D}|\vartheta, H)p(\vartheta|H)$ is required.

Of course a full implementation of the algorithm is more complex and must deal with some technical issues to avoid correlations between consecutive drawn samples. Furthermore one must input an effective move proposal $Q(\bar{\vartheta}'|\bar{\vartheta})$ and tune a number of hyperparameters. However the general idea is simple and works for a surprisingly wide class of problems: this is the strength of Metropolis algorithm and a huge number of variants were proposed and developed [44].

As previously said, usually one is interested in computing the evidence $Z = \int \mathrm{d}\vartheta \; p(\mathcal{D}|\vartheta, H)p(\vartheta)$ of a model. Once a set $\{\bar{\vartheta}_i\}$ of samples is available, this could be done with some forms of Monte Carlo integration (one commonly used option is thermodynamic integration [45]). However methods from this family are known to be slow and unreliable.

A popular variant to Markov Chain Monte Carlo is the *Nested-Sampling* method [46] and it is designed to output, together with the samples, an approximation for the evidence. The basic idea is to replace the multidimensional integral of the evidence with a one dimensional one. Call $L(\vartheta) = p(\mathcal{D}|\vartheta, H)$ and define:

$$X(\lambda) = \int_{L(\vartheta)>\lambda} p(\vartheta) \, \mathrm{d}\vartheta \tag{2.119}$$

$X(\lambda)$ represents the prior volume in which points with likelihood higher than λ are enclosed. The idea is to compute the evidence Z as a function of the one dimensional variable $X(\lambda)$

$$Z = \int_0^1 L(X)\,\mathrm{d}X \simeq \sum_{i=1}^{M} \frac{1}{2}(X_{i-1} - X_{i+1})L_i \tag{2.120}$$

Where $L(X)$ is the inverse of $X(\lambda)$ eq. (2.119) and it is a monotonic function in X. In the second equality an approximation to the integral is done if M samples of X are available. The algorithm provides a way to extract randomly points X_i such that the evidence is well approximated.

Starting from a set of M_0 points θ_i drawn from the prior, the algorithm sorts them by decreasing likelihood $L_i = L(\theta_i)$. Then one should iteratively pick the lowest likelihood one and replace it with another randomly drawn points with higher likelihood (for this task variants of MCMC are available). For each point, it it straightforward to compute $X_i = X(L_i)$. All the points generated create a collection of M elements which can be used to perform evidence estimation. They can also be used as starting point for MCMC to perform posterior sampling.

Monte Carlo inference is slow to run. A typical parameter estimation requires $O(10^6) - O(10^8)$ sampled points to achieve a satisfactory accuracy: this task can take up to weeks, even in large computers! Therefore any speed-up is of crucial importance. The slow part in the process is the likelihood computation of equation (2.116). For each sampled points $\bar{\vartheta}$, a likelihood term must be evaluated for running the Markov chain and a full waveform $h(\bar{\vartheta})$, required for it, must be generated; as a wave can take up to 1 s for being generated, the need for a speed up in this part is manifest. As explained in chapter 4, our work tries to solve this problem with a quick ML wave generation.

Variational Inference When addressing inference problems, *Variational Inference* (VI) [47] is a valid alternative to Monte Carlo. In this framework, one tries to approximate the posterior (or in general any interesting PDF) with another function chosen within a parametric family. The approximating functions are expected to be easier to handle than the original distribution, while being "similar" enough to the original.

The "closeness" of two distributions is usually measured by the KL-divergence, defined as:

$$\mathbb{KL}(a||b) = \sum_{\mathbf{x}} a(\mathbf{x}) \log \frac{a(\mathbf{x})}{b(\mathbf{x})} = \mathbb{E}_a \left[\log \frac{a(\mathbf{x})}{b(\mathbf{x})} \right] \tag{2.121}$$

KL is positive and satisfies $\mathbb{KL}(a||a) = 0$; however, it is not a proper distance (in the metric sense), because it is not symmetric and it does not satisfy triangular inequality. Despite this, KL-divergence is ubiquitous in Variational Inference and other fields of statics.

With the KL-divergence, the VI problem can be turned into an optimization problem. One tries to approximate the PDF $p(\mathbf{x})$ with the best function $q_{\tilde{\alpha}}(\mathbf{x})$ chosen within a family $\{q_\alpha(\mathbf{x})\}_{\alpha \in \mathcal{Q}}$. The inference problem is stated as follows: find the best $\tilde{\alpha} \in \mathcal{Q}$ sucht that the distribution $q_{\tilde{\alpha}}(\mathbf{x})$ is the closest to $p(\mathbf{x})$, according to KL divergence; in symbols:

$$\tilde{\alpha} = \arg\min_{\alpha \in \mathcal{Q}} \mathbb{KL}(q_\alpha(\mathbf{x})||p(\mathbf{x})) \tag{2.122}$$

The solution is computed with standard optimization techniques. Interestingly, the problem of VI is similar to a ML problem (see next chapter): many ML techniques can be effectively employed in VI.

Once the best approximating distribution is found, VI is much faster to run: this is why it is so

attractive. However, it can be much less accurate than Monte Carlo inference. While Monte Carlo is guaranteed to converge to an optimal solution in the limit of long chains, VI lacks a similar theoretical foundation. The choice of the variational family strongly influences the performances of VI and it is impossible to make a choice that is valid in every situation.

Due to its heavy limitations, it is not commonly exploited in GW, where one wants robust physical predictions. However, as ML state-of-the-art provides flexible parametric models for functions and efficient optimization methods, some attempts to apply VI to GW were done. Some recent works [48] [49] use ML methods to model a parametric family for variational inference on the signal posterior. The reward is a many order of magnitude faster parameter estimation. Promising results were found and in the future VI is likely to be a standard tool in GW data analysis. See section 4.1 for more details.

Chapter 3

Basics of Machine Learning

In this chapter we will outline the Machine Learning techniques that has been explored in the present work. The explanation is by no means complete but is intended to give the reader the minimum information to understand how our model works. For a thorough discussion, the reader can refer to [43] from which the present chapter is inspired.

In the first section, we present a very brief introduction of the basics ML reasoning. The concept of learning will be made clear and the standard procedure of fitting will be described.

A second part will be devoted to Regression Methods. Such methods aim to learn a relation between a set of variables. In the context of present work, they are useful to create a relation from BBH parameters to the (low dimensional representation of) waveforms.

In the third section some techniques to reduce data dimensionality will be presented. All of them were tried with the GW dataset created. A particular attention will be given to Principal Component Analysis (PCA) as it is the most effective model tried and it is used in the final release of the code.

3.1 Basics of Machine Learning

> We define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!).[1].

The quote captures well the main goal of Machine Learning (ML): dealing automatically with data. Nowadays a huge amount of data can be acquired with ease and thus using them to learn something is pressing: ML is mainly concerned with extracting as much information as possible from a (huge) set of data. The basic idea is to set up a (usually probabilistic) model to encode all the information we have about the data. The model must have a number of unknown parameters, which must be set after looking at the available information. ML learning provides an effective way to set those parameters with the least handwork possible by fitting the model to data. The human user does not need to know anything about them and their meaning; the only important point is that the model gives good performances: only machine must "learn" something and humans can sit and use the model as a black box. This is the difference with physics. Also a physicist uses data and fitting procedures to understand the underlying patterns; but unlike ML, the outcome of the understanding must be in an human understandable form.

[1] By K.P. Murphy. See [43, section 1.1]

ML is useful when it is applied to boring tasks that require very repetitive work to be done and from which humans have nothing to learn about. Examples of those tasks are image and speech recognition, text prediction, personalized advertising, anti-spam software etc... All of them are easy and boring for humans but still it would be nice to have them done. That is were ML enters.

Every statement based on data is intrinsically uncertain and an estimate of that uncertainty is advisable. For this reason, probabilistic models are the most appropriate framework for automatic learning. Thus the goal of ML is to create a parametric model of a probability distribution $p_\theta(\mathbf{x})$ and to adapt the parameters to data $\mathcal{D}$. The learning procedure is an *optimization procedure* which minimizes an objective $L(\mathcal{D}, \theta)$ (usually called loss function) which measures how well the model agrees with data. The minimization is performed with respect to the model parameters θ. The fitted model is $p_{\tilde{\theta}}(\mathbf{x})$, where:

$$\tilde{\theta} = \arg\min_{\theta} L(\mathcal{D}, \theta) \tag{3.1}$$

In what follows some basic issues in Machine Learning are presented. First the main ML categories of supervised learning, unsupervised learning and reinforcement learning are described. Second the standard way of preparing datasets and perform the fit is discussed. In the third part some common issues regarding the abiliy of the model to learn pattern is presented.

3.1.1 Supervised vs Unsupervised Learning

When dealing with data, it is custom to create a dataset of observations $\mathbf{x}$. Each component $(\mathbf{x})_j$ of $\mathbf{x}$ is called feature. Features can be very heterogeneous, depending on the kind of information they are encoding: they can be real or integer numbers, they can be strings, they can sequences etc... In order to be able to use them within a probabilistic model, they usually are preprocessed and mapped into some real numbers. Thus in what follows we will assume real valued features, i.e. $\mathbf{x} \in \mathbb{R}^D$.

Machine Learning is mostly divided into three broad categories. *Supervised* (or predictive) learning is mainly concerned in making predictions y based on some observations $\mathbf{x}$. A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ must be built with empirical observations $\mathbf{x}$ and empirical quantities y that must be predicted based on $\mathbf{x}$. The model then aims to fit to data a probability distribution for $p(y|\mathbf{x})$. Once trained, the model is able to give a probability distribution for the prediction y. Usually a single prediction is required and one can make the prediction $\widehat{y} = \mathbb{E}[p(y|\mathbf{x})]$. However since a full probability distribution is returned, the model incorporates also information about the confidence of the prediction - a point which is important to stress. The performances of a supervised learning model can be measured by a loss function, which quantify of much the prediction $\widehat{y}$ differs from true value y. The fitting procedure aims to minimize the loss function with respect to model parameters.

Unsupervised learning is less common than supervised and it is mainly devoted to find patterns or unknown features among data. In this case a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ holding only observations is available and the goal is to find a probability distribution $p(\mathbf{x})$ which explains well the data. This category encompasses clustering algorithm, dimensional reduction algorithm and generative models for data[2]. Usually the performance of an unsupervised learning algorithm are more difficult to assess. The problem here is that there is no *a priori* knowledge of the "right piece of information" to gain about the data. The effectiveness of the model must be evaluated when the model is applied to a real world problem.

[2] A generative model tries to generate a data which is very similar to the previous given. An example of that is sentence completion: given a context of a few word a model should generate a plausible continuation.

A third part of Machine Learning is the so called reinforcement learning. This is much less common and exploited in usual applications. Its aim is to learn how to act or behave in some *environments* under different *situations*. The learning is done by trials and errors by trying to maximize a reward for the choice of the proper action to take under given conditions. Reinforcement learning was not used in the current work and thus won't be treated in what follows.

3.1.2 Datasets: training, validation and test

As sketched above, a dataset consists of a set of observations and predictions $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ for the case of supervised learning and of a set of observations $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ for the unsupervised case. Once features of observations are mapped to real number (and possibly preprocessed) they can be conveniently stored in a matrix X where D is the number of features for each observation. The common convention is that $X \in \mathbf{Mat}(N, D)$ and

$$X_{ij} = (\mathbf{x}_i)_j \tag{3.2}$$

i.e. the i-th row of X holds the feature vector of the i-th observation. If predictions are given in the dataset, they are stored in a vector $\mathbf{y}$ such that $(\mathbf{y})_i = y_i$. The notation is standard and it will be used throughout this work.

Depending on the application, it might be convenient to preprocess the data. This means to create a map $X \longmapsto \Xi$ from the dataset to a new dataset. The new dataset will be used for all the learning purposes. A part from dealing with categorical features [3] , preprocessing are useful to obtain "cleaner" data and to make them suitable for a computer. The most common preprocessing aims to scale the data so that every feature is $O(1)$. This is important to ensure that any number within the ML model is not too big or too small, so as to avoid numerical error due to finite computer precision. Virtually every ML model takes advantage of such preprocessing. Another common preprocessing is called data augmentation. Other features are added to the dataset based on previous features: this could be useful to "help" the learning. An example of data augmentation will be discussed in the context of linear regression section 3.2.1 and more details will be provided.

Once a model is trained, it is mandatory to evaluate its performances on data never seen during the fitting procedure. This is required to assess whether the model is able to generalize fully the relation among data learned during fitting procedure. It is common (actually compulsory) to split the main dataset into three smaller datasets, each of which serves a different purpose. They are: training set, validation set and test set.

Training set is the set which is used for the fitting procedure. Every fittable parameter must be set by looking only at the data in the training set. Usually around 70% of the data available are included in the training set, although the percentage can increase if few data are available and/or many train data are required.

Usually a model has some parameters that cannot be fitted but must be chosen by the user before fitting. Those kind of parameters are called *hyperparameters*. The choice of hyperparameters is not trivial and usually must be done by comparing performances on unseen data of trained model with different values of hyperparameters. Another usual situation is that a number of competing model are fitted and one must decide which is the best model to use. Again the solution is to evaluate their performances on unseen data. The tests required by the two situations above are performed by looking at data in the *validation set*. Depending on the needs it can include around 15% of datapoints.

[3] A categorical feature has value in a discrete set $C = \{c_1, \ldots, c_K\}$. Common preprocessings map it either into a single discrete feature or into K different "binary" features.

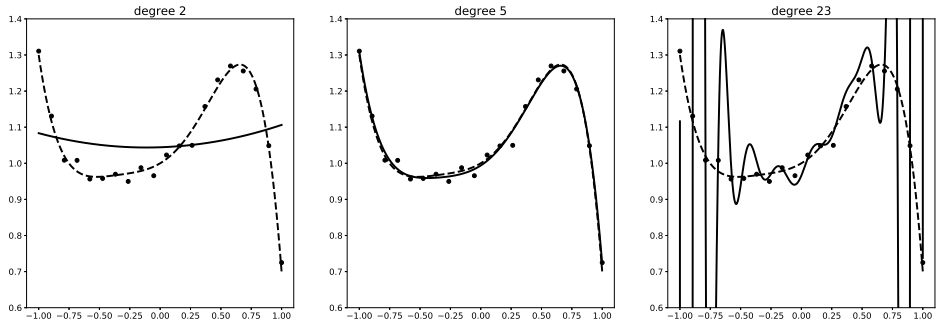

Figure 3.1: 20 noisy observations (dots) drawn from a 5-th order polynomial (dashed line). Observations were fitted fit polynomials (solid line) of increasing order (2, 5, 23). The optimal polynomial (5-th order) is able to capture the trend. Order 2 polynomial underfits the data while order 23 polynomial overfits.

Test set is only used for reporting and communicating the performances of the chosen model. Since this has to be done to data never used to take any decision, a new set must be created. Again a 15% of all the data is common choice for that. If a validation process is not required, test and validation set can correspond. However this situation is rare and in practice the two sets are always formed.

3.1.3 Overfitting and Underfitting

The choice of the right model is not trivial. If one chooses too few parameters for a model or make a wrong choice of model type, they might incur in *underfitting*: the model is not suitable to learn a pattern in the data and it will make poor predictions. However, choosing a model that is too flexible (with too many parameters) might not be the right approach: *overfitting* can occur. When a model overfits it has a very low training error and thus it is able to reproduce training data very well. However, when looking at unseen data, it gives bad predictions[4]: the model learned only training data and it is not able to generalize the underlying pattern.

A simple example can be seen in figure 3.1. A simple polynomial regression was done trying to learn noisy observations from a polynomial of 5-th degree. If the model is fitted with a polynomial of second order, its performances are very poor. On the other hand, results are even worse if one tries to fit the model with a polynomial of order 23. Only training data are predicted and the resulting pattern is absurd.

In choosing the correct model one must trade off between flexibility and the need to avoid overfitting. This is not trivial and requires some work at validation time. In supervised learning, besides model validation, often a good strategy is to *regularize* the model. This means that the loss function is modified in order to discourage extreme values for the parameters: in the example above this would help to avoid the wiggly behavior of the function and to achieve a more reasonable trend. This is extensively applied in Neural Networks were millions of parameters are often used and overfitting is a serious issue.

From the discussion above, it must be clear that a model which works with every kind of problem do not exist. A 23th order polynomial might overfit, if we cope with noisy observation from a 5th order polynomial. However, for data generated by a 20th order polynomial, a 23th polynomial could

[4]This is another reason why having a validation and test set to check model performances is a good idea.

yield a perfectly acceptable result, whereas a 5th order polynomial will underfit. The general lesson is that, depending on the data, one must carefully choose the model which works better in the particular case. Another dataset might require a completely different approach.

This statement is made (slightly) more formal by the popular "No free lunch theorem". The original formulation is about optimization algorithms and states that "any two algorithms are equivalent when their performance is averaged across all possible problems" [50]. This means that, in order to achieve better than random search when looking for an *optmimum*, one must adopt different strategies for different problems. The theorem applies also to ML: a model do not have any guarantee to have acceptable performance on a large set of problems. Thus, when solving a problem with ML, human expertise must play a fundamental role in choosing (and implementing) the right model. Indeed a variety of different models has been proposed to tackle a huge number of problems and, as the applications of ML will change, more and more ML models will need to be developed.

3.2 Regression methods

A regression aims to learn a mapping from inputs $\mathbf{x} \in \mathbb{R}^D$ to outputs $\mathbf{y} \in \mathbb{R}^K$. This is basically a prediction task: given an input $\mathbf{x}$ one needs to predict the best guess for the output $\mathbf{y}$ after having seen a number of "past" observations (the training set). Usually this is done specifying a functional form for $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$. Here $\boldsymbol{\theta}$ represents a set of parameters required to fully specify the probability distribution function (PDF) chosen. A choice of PDF is called model.

With $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ available, making prediction is trivial: the best guess $\widehat{\mathbf{y}}$ for $\mathbf{y}$ is the expected value of the PDF. Furthermore, specifying a full probability distribution for $\widehat{\mathbf{y}}$ rather than a single value can give important information about how confident the prediction is.

Paramers $\boldsymbol{\theta}$ must be chosen by looking at the data: this procedure is usually called fitting or learning procedure. As explained above, usually training data are stored in the matrices $X \in \mathbf{Mat}(N, D)$ and $Y \in \mathbf{Mat}(N, K)$. Each row holds one of the N observation required for training. The parameter are chosen by minimizing (or maximizing) a function L of the data (X, Y) and of the parameters $\boldsymbol{\theta}$. L is called *loss function*. Once specified a model and a loss function[5], a fitting strategy must be adopted in order to find among the (usually huge) space of parameters those who gives the best value of the loss function.

Below we outline some common regression methods that has been tried and implemented during the work.

3.2.1 Linear Regression

The most common regression method is called *Linear Regression*. It consist on a model in the form[6]:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2) \tag{3.3}$$

Here the vector $\mathbf{w}$ holds the parameters to fit. σ^2 models the noise of observations and can be chosen at will. The choice does not affect the determination of $\mathbf{w}$ but only the confidence of predictions of the model.

[5] Clearly the same model can be fitted with different loss functions. The results of the fit (i.e. a choice for parameters $\boldsymbol{\theta}$) might depend on that choice.

[6] Here we will consider only the case of one dimensional outputs y. The generalization is straightforward since every component of the output can be predicted separately.

The objective to maximize is the log-likelihood (LL) of the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &:= \log(p(\mathcal{D}|\boldsymbol{\theta})) = \sum_{i=1}^{N} \log(p(y_i|\mathbf{x}_i, \boldsymbol{\theta})) \\ &= \sum_{i=1}^{N} \log\left[(\frac{1}{2\pi\sigma^2})^{\frac{1}{2}} \exp(-\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T\mathbf{x}_i)^2)\right] \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 - \frac{N}{2}\log(2\pi\sigma^2) \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 \end{aligned} \tag{3.4}$$

Where in the last equality we dropped a unrelevant constant, not depending on the parameters $\mathbf{w}$. The form of equation (3.4) is the usual sum of squared errors that one tries to minimize.

The best parameters are computed as follows:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \log(p(\mathcal{D}|\boldsymbol{\theta})) \tag{3.5}$$

This procedure is called maximum likelihood estimation (MLE). For the case of the linear regression a closed form for (3.5) is available [7] . Computing gradient of (3.4) and equating them to zero, one obtains the following equation for the problem in (3.5):

$$\mathbf{w}_{best} = (X^T X)^{-1} X^T \mathbf{y} \tag{3.6}$$

The equation gives the best choice of parameters for the linear model (given the data).

A simple (yet powerful) generalization of linear regression is known as *basis function expansion* and is useful to model non linearities. It consists in the replacement

$$\mathbf{x} \to \boldsymbol{\xi}(\mathbf{x})$$

where $\boldsymbol{\xi} : \mathbb{R}^D \to \mathbb{R}^{D'}$ is any continuous function. The components $\xi_i(\mathbf{x})$ are called basis functions. With this replacement the model (3.3) becomes:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\boldsymbol{\xi}(\mathbf{x}), \sigma^2) \tag{3.7}$$

Equation (3.6) is still valid with the obvious replacement $X \to \Xi$ where $\Xi_{ij} = \xi_j(\mathbf{x}_i)$. The trick of adding new features to data (or replacing them with other features) is called *data augmentation* and it is widely used in Machine Learning, to make the model more flexible or to make the data more easier to fit.

Increasing the number of basis functions increases model complexity and thus its flexibility. However the increase might not always be advisable: simpler models can be more interpretable, still showing good performance. Furthermore when many basis functions are used, overfitting becomes an issue and model can generalize very poorly to unseen data (see discussion above on overfitting).

[7] This is a particular case and is due to the simplicity of the model. For more complex model a closed form is not available and one can only obtain an *estimate* of the optimal $\boldsymbol{\theta}$

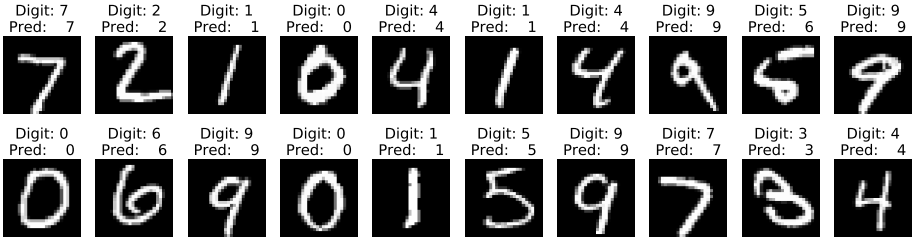

Figure 3.2: 20 test examples of handwritten digits from MNIST dataset with reported true and predicted label. Digits were classified with a softmax regression model implemented within the present work. Misclassification rate is: 7.73%. The current state of the art misclassification rate is more than one order of magnitude lower (0.18% [51]) and it uses of sophisticated Deep Learning models. However, the result obtained with softmax regression is still remarkable due to the simplicity of the model implemented: with a easy to code model, a reliable classification is achieved. See `https://paperswithcode.com/sota/image-classification-on-mnist` (consulted on April 2020) for a review of the state-of-the-art model performances of MNIST.

3.2.2 Softmax Regression

Softmax regression is a popular method to solve a classification problem. A classification problem is a special kind of regression where the outputs can assume only one of C different value $y = k; k = 1, .., C$. The value k of y_i specify which class data point $\mathbf{x}_i$ belongs to. The regression problem consists in giving the proper label to each test (i.e. unseen) input $\mathbf{x}$.
As usual every classification is embedded within a probabilistic framework. A model specifies the probability $p(y = k|\mathbf{x}, \boldsymbol{\theta})$ which denotes the probability that point $\mathbf{x}$ belongs to class k. The guessed class assignment $\widehat{y}$ can be done as follows:

$$\widehat{y} = \arg\max_k p(y = k|\mathbf{x}, \boldsymbol{\theta}) \tag{3.8}$$

In the particular case of *Softmax Regression*, the model is chosen to be:

$$p(y = k|\mathbf{x}, W) = \frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_{k'=1}^{C} e^{\mathbf{w}_{k'}^T \mathbf{x}}} \equiv \mathcal{S}(W^T \mathbf{x})_k \tag{3.9}$$

Here $W \in \mathbf{Mat}(D, C)$ is the matrix of the weights defined as $W_{jk} = (\mathbf{w}_k)_j$ and holds all the weights to be fitted. Since an overall multiplicative factor does not affect the result, it can be assumed without loss of generality that $\mathbf{w}_C = \mathbf{0}$. This ensures that the result is unique.

As in the case of linear regression, the model is fitted maximizing the log likelihood of the data. For notational convenience, let $\widehat{Y}_{ik} = p(y = k|\mathbf{x}_i, W)$ be the model predictions and $Y_{ik} = \delta_{y_i,k}$ the train labels. Then the log likelihood of the model is:

$$\mathcal{L}(W) = \log \prod_{i=1}^{N} \prod_{k=1}^{C} \widehat{Y}_{ik}^{Y_{ik}} = \sum_{i=1}^{N} \sum_{k=1}^{C} Y_{ik} \cdot \log(\widehat{Y}_{ik}) \tag{3.10}$$

$$= \sum_{i=1}^{N} \left[\left(\sum_{k=1}^{C} Y_{ik} \mathbf{w}_k^T \mathbf{x}_i \right) - \log \left(\sum_{k'=1}^{C} e^{\mathbf{w}_{k'}^T \mathbf{x}} \right) \right] \tag{3.11}$$

The function $f(W) = -\mathcal{L}(W)$ is called cross-entropy loss function and is widely used as minimization

target for many classifications problems. The function makes sense: if a data point $\mathbf{x}_i$ is correctly classified, its contribution to the sum in (3.10) is 0; on the other hand if it is misclassified, the contribution to the loss function will blow up to ∞ (there will a term like $-1 \cdot \log(0)$).

Unlike linear regression, a closed form expression for the weights W is not available. For this reason one must use a gradient descent algorithm for finding a (local) minimum of the loss function $f(W)$. Due to its popularity a huge variety of methods have been developed: see [52] for a review of them. For softmax regression, the gradient is readily computed:

$$\nabla_{\mathbf{w}_k} f(W) = \sum_{i=1}^{N} (\widehat{Y}_{ik} - Y_{ik}) \mathbf{x}_i \tag{3.12}$$

In figure 3.2 is reported an application of softmax regression for classification of handwritten digits of MNIST dataset[8].

3.2.3 Gaussian Discriminant Analysis

Softmax regression is primarly concerned in finding the proper probability distribution for $p(y = k|\mathbf{x}, W)$. Another popular approach to classification relies on creating a probability model $p(\mathbf{x}|y = k, \boldsymbol{\theta})$ for each target class. Methods following the latter approach are called *generative classifiers* as opposed to *discriminative classifiers*.

While in discriminative classifiers one tries to learn a relation between the inputs and the class labels choosing a functional form for it, in generative classifiers the idea is different: one tries to learn the features which distinguish each class. A test time, it compares an input with the features of each class and decides a (probabilistic) class assignment. More formally, a generative classifiers makes a prediction $\widehat{y}$ using Bayes rule:

$$\widehat{y} = \arg\max_k \log p(y = k|\mathbf{x}, \boldsymbol{\theta}) = \arg\max_k \left[\log p(y = k|\boldsymbol{\theta}) + \log p(\mathbf{x}|y = k, \boldsymbol{\theta})\right] \tag{3.13}$$

where in the last equality an overall constant independent of k was dropped. $p(y = k|\boldsymbol{\theta})$ is a trainable PDF which represents *a priori* probability for class labels and, as above, $p(\mathbf{x}|y = k, \boldsymbol{\theta})$ is the generative model for each class.

The model has a joint PDF:

$$p(y, \mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} p(y = k|\boldsymbol{\theta}) p(\mathbf{x}|y = k, \boldsymbol{\theta}) \tag{3.14}$$

Thus the joint probability is a linear combination of class assignment probability $p(\mathbf{x}|y = k, \boldsymbol{\theta})$ weighted by the class "importance" $p(y = k|\boldsymbol{\theta})$. It is common to set $p(y = k|\boldsymbol{\theta}) \propto$ const.: in this case all classes are *a priori* equally probable and the class assignment is determined only by the generative part.

A simple (but still effective) generative classifier is called *Gaussian Discriminant Analysis* (GDA). It chooses a gaussian distribution as the probability model for the each class:

$$p(\mathbf{x}|y = k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{3.15}$$

The model is easy to fit with maximum likelihood: $\boldsymbol{\mu}_k$ and Σ_k are set to the empirical mean and

[8] `https://en.wikipedia.org/wiki/MNIST_database` for more information

covariance of data belonging to class k. For data with many feautures, the model might have a dangerously high number of parameters: overfitting becomes likely. To avoid it, a number of strategies are usually adopted. In a typical situation, one imposes a constraint on covariance matrix (which limits the number of free parameters) or "ties" the parameters across the classes ($\forall k \;\; \Sigma_k = \Sigma$).

3.2.4 Mixture of Experts

In actual regressions it is very common the case in which the relation to fit has only local linear trends. In such situations a linear regression from the full parameter space can generalize poorly since it does not capture different slopes at different points. However, it might work well a model with many linear regressions, each applied to a different part of parameters space. This idea motivates the introduction of a the *Mixture of Experts* (MoE) model [53].

In the model, a number of regression models are called *experts*. A function called *gating function* decide which expert to use based on the input. Thus, each expert specializes in accurate prediction in a small region of the parameter space. The intuition is formalized by the introduction of an unseen (i.e. artificial) categorical variable $z_i \in \{0, ..., K\}$ for each input $\mathbf{x}_i$. Based on the value of z_i, a different linear combination of experts is used. Formally the expert and the gating function are modeled as follows:

$$p(y|\mathbf{x}, z = k, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}_k^T\mathbf{x}, \sigma_k^2) \qquad \text{(Expert model)} \tag{3.16}$$

$$p(z = k|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{G}(\mathbf{x}, \boldsymbol{\theta})_k \qquad \text{(Gating model)} \tag{3.17}$$

Here $\mathcal{G}(\cdot)$ represents any classifier. The most common choice is the softmax function (3.9) but more complex classifiers (such as neural networks) can be used depending on the problem. The meaning of gating function is to provide a probabilistic model (3.17) for variable z which is used to determine with (3.16) the weights of each expert's prediction.

The overall probabilistic model is[9]:

$$\begin{aligned} p(y|\mathbf{x}, \boldsymbol{\theta}) &= \sum_{k=1}^{K} p(y|\mathbf{x}, z = k, \boldsymbol{\theta}) p(z = k|\mathbf{x}, \boldsymbol{\theta}) \\ &= \sum_{k=1}^{K} \mathcal{N}(y|\mathbf{w}_k^T\mathbf{x}, \sigma_k^2) \mathcal{S}(V^T\mathbf{x})_k \end{aligned} \tag{3.18}$$

From here the meaning of the gating function is clear: it switch on (in a smooth way) expert contributions whenever this is required. An example of a fit made with a MoE model is reported in 3.3

We can gain more insight on the model by writing the joint PDF $p(y, z|\mathbf{x}, \boldsymbol{\theta})$ for y and z. By Bayes theorem, it holds $p(y, z|\mathbf{x}, \boldsymbol{\theta}) \propto p(y|\mathbf{x}, z, \boldsymbol{\theta})p(z|\mathbf{x}, \boldsymbol{\theta})$ and the MoE prediction arises naturally by marginalizing over z:

$$p(y, |\mathbf{x}, \boldsymbol{\theta}) = \sum_{z \in \{1,...,K\}} p(y, z|\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(y, z = k|\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(y|\mathbf{x}, z = k, \boldsymbol{\theta}) p(z = k|\mathbf{x}, \boldsymbol{\theta}) \tag{3.19}$$

[9]For definiteness Softmax Function (3.9) is used as gating function from now on.

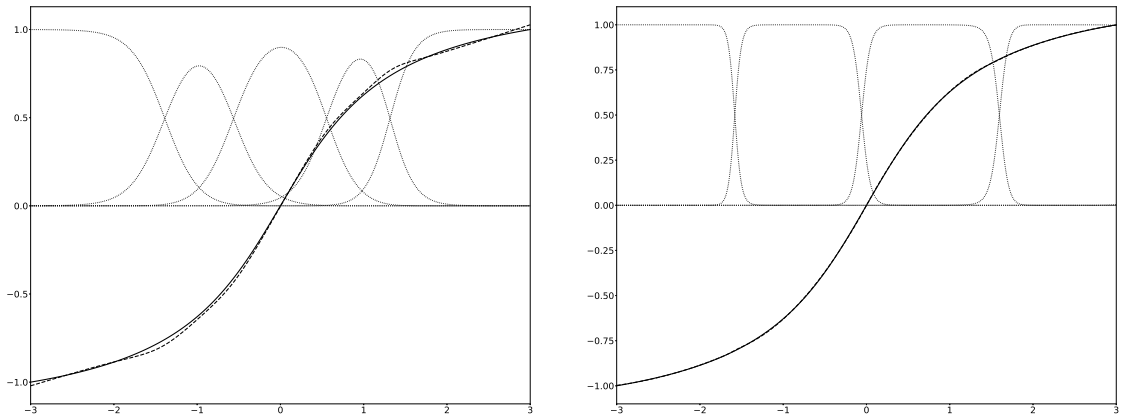

Figure 3.3: The function $y = \frac{1}{\arctan 3} \arctan x$ is fitted with a MoE model. The true function (solid line) is compared with the predicted function (dashed). The dotted lines represents the value of the gating function $p(z = k|\mathbf{x}, \boldsymbol{\theta})$ for different experts (labeled by k): the gating function is able to effectively divide the input space. In the left panel, we set a simple linear regression for the five experts. In the right panel we set a quadratic regression for the four experts (achieved with basis function expansion). The lack of non-linearities in the expert model is balanced by a higher number of experts.

As many probabilistic models, MoE is fitted by maximizing the log-likelihood of the observations:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \sum_{z_i} p(y_i, z_i|\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} p(y_i|\mathbf{x}_i, z_i = k, \boldsymbol{\theta}) p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}) \right] \end{aligned} \tag{3.20}$$

Although $\mathcal{L}(\boldsymbol{\theta})$ can be maximized with gradient descent, the Expectation Maximization algorithm (EM) provides a more efficient way for the fit. EM algorithm makes use of an iterative method to maximize an increasing accurate lower bound of $\mathcal{L}(\boldsymbol{\theta})$ and it is described below.

Fitting MoE with EM algorithm Equation (3.20) is hard to maximize because the log symbol is inside the sum; this prevents us from having a closed form expression for the MLE. The idea behind EM algorithm is that *if* we could observe the variables z_i, there would be no reason to marginalise over them and thus the log-likelihood would be tractable. We define the *complete data log-likelihood* as:

$$\mathcal{L}'(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(p(y_i, z_i|\mathbf{x}_i, \boldsymbol{\theta}) \right) \tag{3.21}$$

$\mathcal{L}'(\boldsymbol{\theta})$ would be the model log-likelihood, if we observed also the z labels. It can be shown that for each choice of $\boldsymbol{\theta}$, $\mathcal{L}'(\boldsymbol{\theta})$ is a lower bound of $\mathcal{L}(\boldsymbol{\theta})$, i.e. $\mathcal{L}'(\boldsymbol{\theta}) < \mathcal{L}(\boldsymbol{\theta})$. For this reason, maximizing (3.21) we get closer to a maximum of (3.20).

Of course we don't know the actual values of the hidden variables z_i and, for every training point i, they should be guessed using $p(z_i|\mathbf{x}_i, \boldsymbol{\theta})$. Once the z_is are known, we can compute MLE for the parameters maximasing the expected value of $\mathcal{L}'(\boldsymbol{\theta})$ eq. (3.21). Then one can give another estimation to the hidden variable values and fit again the model. This is the core of the EM algorithm. It runs as

follows:

1. Initialize $\boldsymbol{\theta}^0$. Set $t = 0$. Choose threshold ϵ for minimum log-likelihood increase that can be tolerated before stopping.

2. **E step** - Compute the "responsibility of class k for point i": $r_{ik} = p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}^t)$

3. **M step** - Set optimal $\boldsymbol{\theta}^{t+1}$ by maximizing *expected* complete data log-likelihood $\mathbb{E}\left[\mathcal{L}'(\boldsymbol{\theta}^t)\right]$:

$$\begin{aligned}\boldsymbol{\theta}^{t+1} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}\left[\mathcal{L}'(\boldsymbol{\theta}^t)\right] = \arg\max_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_i \log p(y_i, z_i|\mathbf{x}_i\boldsymbol{\theta})\right] \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i,k} \left[r_{ik} \log p(y_i, z_i = k|\mathbf{x}_i\boldsymbol{\theta})\right] \end{aligned} \tag{3.22}$$

   where the expectation is taken with respect to the complete data $(\mathbf{x}_i, y_i, z_i)$. The factor $r_{ik} = p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}^t)$ takes into account the uncertainty about the value of z_i.

4. Compute $\Delta\mathcal{L} = \mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathcal{L}(\boldsymbol{\theta}^t)$. If $\Delta\mathcal{L} > \epsilon$ go to step 2.; exit otherwise.

With this algorithm a local maximum of an approximation of $\mathcal{L}'(\boldsymbol{\theta}^{t+1})$ is computed at each step. Since $\mathcal{L}'$ is a lower bound of $\mathcal{L}$ this results in an increase of log-likelihood $\mathcal{L}$. At each step the algorithm is guaranteed to improve the value of the data likelihood $\mathcal{L}$ (i.e. $\Delta\mathcal{L} > 0$) and to reach a local maximum of the true LL.

The result of the EM algorithm strongly depends on the initialization. Indeed EM, is guaranteed only to converge towards a local maximum; different extrema can be obtained with different choices of initial parameters. Thus, the initialization part is not trivial at all and some care is required. An heuristic method for a good first guess of the initial weights is provided by the *farthest point clustering*. Let K be the number of experts. The algorithm aims to choose K "centroids" $\boldsymbol{\mu}_k$, so that they are as spread as possible in the parameter space. After a random choice of the first centroid among the training data, one iteratively picks at random a new centroid. Each training point is chosen with probability proportional to its distance from the closest centroids already extracted. After K centroids are chosen, an initial guess of responsibility is done as $r_{ik}^{(guess)} \propto d_{\mathbf{x}_i,\boldsymbol{\mu}_k}$, where d is the usual euclidean distance. Responsibilities $r_{ik}^{(guess)}$ are input to an initial M step which sets model weights $\boldsymbol{\theta}^0$ accordingly.

The EM algorithm works for a large class of probabilistic models called mixture models where a PDF for the observed data can be written as

$$p(\mathcal{D}|\boldsymbol{\theta}) = \sum_k p(z = k|\mathcal{D}, \boldsymbol{\theta}) p(\mathcal{D}|z = k, \boldsymbol{\theta}) \tag{3.23}$$

Obviously different models have different objectives to maximize in eq. (3.22). Note that the equation above is very similar to (3.14) for generative classifiers. There is an important difference: in a generative classifier the class assignment are known and given in the dataset. Here a class assignment is determined by unobserved variables - indeed the goal of EM algorithm is to guess a plausible value for the unobserved variables.

MoE is a particular case of Mixture Model. For MoE the objective in equation (3.22) takes the

form:

$$\mathcal{Q}(\boldsymbol{\theta}) = \mathbb{E}\left[\mathcal{L}'(\boldsymbol{\theta})\right] = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log\left[\pi_{ik}\mathcal{N}(y_i|\mathbf{w}_k^T\mathbf{x}, \sigma_k^2)\right] \tag{3.24}$$

$$= \sum_{i,k} r_{ik} \log\left(\pi_{ik}\right) + \sum_{i,k} r_{ik}\left[-\frac{1}{\sigma_k^2}\left(y_i - \mathbf{w}_k^T\mathbf{x}_i\right)^2\right] \tag{3.25}$$

where:

$$\pi_{ik} = \mathcal{S}(V^T\mathbf{x})_k \tag{3.26}$$

$$r_{ik} = \pi_{ik}^{old}\mathcal{N}(y_i|\mathbf{w}_k^T\mathbf{x}, \sigma_k^2) \tag{3.27}$$

The E step consists in computing r_{ik} with (3.27). The M step performs an update of the experts weights $\mathbf{w}_k$ and one for the softmax weigths. The update of $\mathbf{w}_k$ can be done by solving a weighted linear regression that appears in the second term of (3.25). Defining $R_k \in \mathbf{Mat}(N, K)$, with $(R_k)_{ij} = \delta_{i,j}r_{ik}$, the result analog to that in section 3.2.1 is:

$$\mathbf{w}_k = (X^T R_k X)^{-1} X^T R_k \mathbf{y} \tag{3.28}$$

Regarding the updates of the softmax weights, one can note that the first term in (3.25) is the categorical cross entropy loss function eq (3.10) with targets r_{ik}. As in section 3.2.2 softmax model can be fitted by gradient descent as described in section 3.2.2.

3.2.5 Neural Networks

Neural Networks (NNs) are today one of the most popular regression method. Their popularity arises from great model flexibility together with a number of effective software libraries, which make the implementation and fitting procedure easy and quick. Many different modifications to the basic model has been tried and successfully applied to several fields of artificial intelligence. A (much) broader discussion can be found in [54]. In what follows only a very brief introduction will be provided.

A (feed-forward) neural network is a series of simple regression models stacked on top of each other. The regression is made from inputs $\mathbf{x} \in \mathbb{R}^D$ to outputs $\mathbf{y} \in \mathbb{R}^K$ and each basic regression model is called layer. A layer l takes input $\mathbf{h}_l \in \mathbb{R}^{D_l}$ from the previous layer and return its output $\mathbf{h}_{l+1} \in \mathbb{R}^{D_{l+1}}$ to the $l+1$-th layer. Usually the first layer is called input layer the last is called output layer while the others are the hidden layers. Each of them has the following structure:

$$\mathbf{h}_{l+1} = f_l(V_l \cdot \mathbf{h}_l) \tag{3.29}$$

where $V_l \in \mathbf{Mat}(D_{l+1}, D_l)$ are the weights of the model. The function $f_l : \mathbb{R} \to \mathbb{R}$ is called activation function and usually is taken as a monotonous increasing function. Some common choice for activations are the sigmoid function, ReLU function, linear function and hyperbolic tangent (see fig. 3.4). The common convention is that the activation function is applied element-wise to its input: $(f(\mathbf{x})_i)_j \equiv f((\mathbf{x}_i)_j)$.

The input $\mathbf{x}$ undergoes a series of deterministic non-linear transformation through the layers until the output $\widehat{\boldsymbol{y}}$ is returned[10] . Setting the weights $\{V_l\}_{l=0,..,N_l}$ is performed by gradient descent. A

[10]Unlike previous models here the model is not presented in a probabilistic framework. This is because a probabilistic

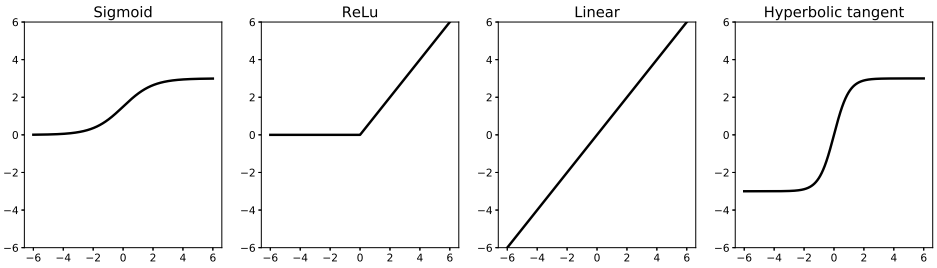

Figure 3.4: Graph of some of the activations functions most commonly used in NNs. Functions plotted are sigmoid, ReLu, linear and hyperbolic tangent. Sigmoid and hyperbolic tangent are multiplied by an overall factor of 3.

famous algorithm called backpropagation computes the gradients with respect to each of the the weights $(V_l)_{jk}$, once a loss function is specified. The computation relies on the chain rule for derivatives and, although straightforward in principle, it is not trivial, due to the messy notation to introduce: we will not discuss it here.

The layer described above is called dense (or fully connected) layer. Other layers type are possible such as convolutional layers, pooling layers, downsampling layers etc... Each of them serves different purpose and is applied to different problems. Choosing the exact architecture (i.e. number and type of layers as well as their dimensionality) of a NN is not straightforward and requires some expertise: there are no general rules for doing so and a lot of work must be done at validation time to pick the best model among those tried.

The original motivation for the Neural Networks come from the study of how human brain processes information. In the brain a number of processing units (called *neurons*) are connected with each other by means of synapses which transmits electric signals generated by neurons. Broadly speaking, each neuron takes some input signals and, if the overall magnitude of input signals is above a certain threshold, it sends an output signals to all neurons it is connected with. The topology of the network, and the activation threshold are what make each part of the brain different. (Artificial) Neural Network were originally developed to emulate this behavior. Connections between layers (matrix V_l) resembles the synapses and activation functions usually have horizontal asymptotes at infinity to mimic a continuous version of real neurons activations.

It was soon realized that the model was powerful and efficient: it was proved that any continuous, multi-input/multi-output function can be approximated with arbitrary accuracy by a neural network with a single hidden layer and a feed-forward architecture. The theorem goes by the name of *Universal approximation theorem* [55] and the result does not depend on the choice of activation (see e.g. [56]). Strong heuristic and theoretical foundations are the starting point of a lively field called Deep Learning. Deep Learning is focussed on Neural Networks with a large number of hidden layers. The idea is that each layer is able to learn features from data at an increasing level of abstraction (see e.g [57]): thus complicated tasks such as object recognition within images requires deep networks. The field is very open and rich in applications and developments, which we do not consider here.

formulation does not provide much insight on the problem and makes the notation less clear.

3.3 Dimensionality reduction methods

The aim of any dimensional reduction method is to represent a dataset of high dimensional data (in $\mathbb{R}^D$) with suitable low dimensional data (in $\mathbb{R}^K$ with $K << D$). An effective reduction model is able to remove unrelevant features from data and thus to build a smaller dataset out of a bigger one. The advantage of this procedure is clear: not only it makes every computation with data less time consuming but also it makes any pattern among data more evident and easy to learn. In the case of current work the advantage is manifest: a wave sampled on a grid of ~ 3000 points can be fully reconstructed starting from a point in a low dimensional space with a number of features as low as 5.

More formally, a *dimensional reduction method* is any couple of function $f : \mathbb{R}^D \longmapsto \mathbb{R}^K$ and $g : \mathbb{R}^K \longmapsto \mathbb{R}^D$ with the following property:

$$\begin{aligned} g \circ f :& \mathbb{R}^D \longmapsto \mathbb{R}^D \\ & \mathbf{v} \longmapsto \mathbf{h}(\in \mathbb{R}^K) \longmapsto \widehat{\mathbf{v}} \simeq \mathbf{v} \end{aligned} \tag{3.30}$$

Usually the the similarity between $\widehat{\mathbf{v}}$ and $\mathbf{v}$ is quantified with the specification of a loss function $L(\widehat{\mathbf{v}}, \mathbf{v})$ for the model. Two values are deemed similar if $L(\widehat{\mathbf{v}}, \mathbf{v})$ is below a certain threshold. Of course, a reliable method must work with a low threshold.

$\mathbf{h} = f(\mathbf{v}) \in \mathbb{R}^K$ is said to be the low dimensional representation of $\mathbf{v}$; $\widehat{\mathbf{v}} = g(\mathbf{h}) \in \mathbb{R}^D$ is said to be the reconstruction of $\mathbf{v}$. f is called *encoder*; g is called *decoder*.

A model consists in specifying a functional form for g and f which depends on many parameters. A fitting procedure is required to make the best choice of them (i.e. the set of parameters which minimize the average reconstruction error $L(\widehat{\mathbf{v}}, \mathbf{v})$). Several methods has been developed to achieve such reduction. They mainly pertain to unsupervised learning: in the dataset there is no indication of what the right $\mathbf{h}_i$ should be and this must be inferred from data. In the following we outline some models that were tried during the work and proved of some utility.

3.3.1 Principal Component Analysis

Principal Component Analysis (PCA) is one the most popular method for dimensional reduction. The algorithm is both effective and easy to implement. The basic idea is to seek a linear relation between high dimensional and low dimensional data: high dimensional data are projected onto a low dimensional subspace of $\mathbb{R}^D$ with an orthogonal projection. The right form of the projection matrix is chosen by looking at covariance matrix of the data. For this reason PCA can also be seen from an interesting probabilistic point of view: this will be done in a dedicated subsection below.

We define $\mathbf{h}$ as follows:

$$\mathbf{h} = W \cdot (\mathbf{v} - \boldsymbol{\mu}) \tag{3.31}$$

with $W \in \mathbf{Mat}(K, D)$. Where $\boldsymbol{\mu} \in \mathbb{R}^D$ is an arbitrary constant vector. The best reconstruction $\widehat{\mathbf{v}}$ must be as well a linear transformation:

$$\widehat{\mathbf{v}} = \widehat{W} \cdot \mathbf{h} + \boldsymbol{\mu} \tag{3.32}$$

Here $\widehat{W} \in \mathbf{Mat}(D, K)$. A square error loss function is used t specify the reconstruction error:

$$L(\widehat{\mathbf{v}}, \mathbf{v}) = \|\widehat{\mathbf{v}} - \mathbf{v}\|^2 \tag{3.33}$$

Under this framework a well known theorem can be proved. It provides a method to compute the best reduction matrix W as well as the relation between matrices $\widehat{W}$ and W. Some common notation shall first be introduced. We denote $V \in \mathbf{Mat}(N, D)$ a dataset of N observations of vectors $\mathbf{v} \in \mathbf{R}^D$. In similar way we define the matrix $H \in \mathbf{Mat}(N, K)$.

We can now state and prove the theorem [11].

Theorem 3.3.1 (PCA)**.** *Given a dataset V, suppose we want to find the matrix $\widehat{W} \in \mathbf{Mat}(D, K)$ and the reduced dataset $H \in \mathbf{Mat}(N, K)$ such that the mean square reconstruction error is minimized:*

$$J(H, \widehat{W}) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{v}_i - \widehat{\mathbf{v}}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{v}_i - \widehat{W} \cdot \mathbf{h}_i\right\|^2 = \left\|V - H \cdot \widehat{W}^T\right\|_F^2 \tag{3.34}$$

Let $Z \in \mathbf{Mat}(D, K)$ be a matrix such that $Z_{ij} = (\boldsymbol{\lambda}_j)_i$ where $\boldsymbol{\lambda}_j \in \mathbb{R}^D$ is the j-th largest eigenvalue of the empirical covariance matrix of the data, $\Sigma = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i \cdot \mathbf{v}_i^T = V^T \cdot V \in \mathbf{Mat}(D, D)$.
The minimum of J is obtained setting $\widehat{W} = Z$. Furthermore the best low-dimensional reduction of the data is given by $H = V \cdot \widehat{W}$. This means that $\mathbf{h}_i = W \cdot \mathbf{v}_i = \widehat{W}^T \cdot \mathbf{v}_i$.

Once computed the matrix $\widehat{W}$ of the first K eigenvectors of covariance matrix of high dimensional data, reducing and reconstructing data is a trivial task that can be done with little computational effort using formulae (3.31) and (3.32) with $W = \widehat{W}^T$. To ensure the uniqueness of the $\widehat{W}$ decomposition it is common to normalize to 1 the eigenvectors of the covariance matrix. Moreover in the theorem above, as it is custom to do, data are assumed to be zero mean in the computation of the covariance matrix. This requirement is trivial because data can always have zero mean by setting the constant $\boldsymbol{\mu}$ to the empirical mean of the visible variables.

Choosing the first K eigenvector means choosing the K directions in the parameters space $\mathbb{R}^D$ along which data show more variability - the eigenvectors $\boldsymbol{\lambda}_j$ are also called Principal Components (PC) of the data. This is an heuristic explanations of why the projections along those directions are the most relevant for explaining the data. For a rigorous proof, the reader can refer to [43, sec. 12.2.2]

An easy improvement to the model can be done with a simple change to the algorithm. Instead of considering covariance, one can consider correlation $\rho \in \mathbf{Mat}(D, D)$ defined as

$$\rho_{jk} := \frac{\Sigma_{jk}}{\sigma_j \sigma_k} \tag{3.35}$$

where $\sigma_j = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{v}_i)_j^2 = \frac{1}{N}\sum_{i=1}^{N} (V_{ij})^2$ is the standard deviation of j-th feature of data (assumed zero mean). This can be useful to avoid that directions with small variability display high variance due to a large measurement scale.

PCA as perturbative expansion One can have a deeper insight on PCA considering the following formula:

$$\widehat{\mathbf{v}} = \widehat{W} \cdot \mathbf{h} = \sum_{j=1}^{D} h_j \cdot \boldsymbol{\lambda}_j \tag{3.36}$$

[11] Here $\|\cdot\|_F^2$ denotes the Frobenius Norm $\|A\|_F^2 = \sum_i \sum_j |A_{ij}|^2$

where, again, $\boldsymbol{\lambda}_j$ is the j-th eigenvalue of the covariance (or correlation) matrix. Thus the reconstructed value of $\mathbf{v}$ can be seen as a linear combination of K basis vector $\boldsymbol{\lambda}_j$ (i.e. the principal components) with linear combination coefficients h_j. According to (3.31),

$$h_j = (\widehat{W}^T \cdot \mathbf{v})_j = \langle \boldsymbol{\lambda}_j, \mathbf{v} \rangle \tag{3.37}$$

Since less important PCs are more orthogonal to data, the typical magnitude of h_j increases with the eigenvalue of $\boldsymbol{\lambda}_j$. For this reason PCA can be seen as a perturbative expansion on the basis vectors $\boldsymbol{\lambda}_j$ where the accuracy is roughly measured by the eigenvalues of the first neglected PC. Increasing the number K of PCs considered increases the accuracy of the reconstruction (but also the complexity of the model). Very regular datasets can be explained by a few PCs; on the other hand complicated dataset which don't show clear patterns requires a big number of PCs to have a realiable estimation.

Looking at eigenvalues E_j can give an estimation of how important is the corresponding eigenvector $\boldsymbol{\lambda}_j$ when compared to the others. If a few E_j are different orders of magnitude higher than the others, PCA works well; if all E_j have the same magnitude, then all PCs have the same importance and one must retain many PCs: the gain from dimensional reduction is poor.

The number K of PCs to consider depends on the threshold on the reconstruction error that the user can tolerate and there is no general rule for deciding. K must be set at validation time, trying different values and check for performances. Other methods exist in literature but are not considered here.

PCA as probabilistic model PCA can be seen as a probabilistic model where relation between variables ($\mathbf{h}$ and $\mathbf{v}$ in the case of PCA) are encoded with a probability distribution function. This is a standard way of reasoning in ML. When inserted in this broader probabilistic context, PCA can be seen as a special case of Factor Analysis (FA) as it will be clear later.

The key idea of FA is to explain all correlations among features $\mathbf{v}_i$ of an observation in terms of a lower number of hidden (unobserved) variables $\mathbf{h}_i$. The relations between variables are represented by a probability distribution function $p(\mathbf{v}_i, \mathbf{h}_i)$. It is useful to write the full joint probability distribution with Bayes theorem (an overall normalization constant is dropped).

$$p(\mathbf{v}_i, \mathbf{h}_i) \propto p(\mathbf{v}_i|\mathbf{h}_i, \boldsymbol{\theta}) \cdot p(\mathbf{h}_i) \tag{3.38}$$

Once a particular form of PDF is chosen, the model is full specified and its parameters $\boldsymbol{\theta}$ can be fitted.

FA choses to use normal distributions:

$$p(\mathbf{v}|\mathbf{h}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{v}|\widehat{W}\mathbf{h} + \boldsymbol{\mu}, \Psi) \tag{3.39}$$

$$p(\mathbf{h}) = \mathcal{N}(\mathbf{h}|\boldsymbol{\mu_0}, \Sigma_0) \tag{3.40}$$

where again $\widehat{W} \in \mathbf{Mat}(D, K)$ and $\Psi \in \mathbf{Mat}(D, D)$.

Here Ψ encodes the some additional correlation among visible variables $\mathbf{v}_i$ which is not explained by the hidden variables. If $\Psi = \sigma^2 \mathbb{I}$ the model is called *probabilistic PCA* (PPCA); as it will be shown below in the limit $\sigma \to 0$ we recover PCA. Setting $\Psi \sim \mathbb{I}$ means that we are forcing the hidden variables $\mathbf{h}_i$ to explain all the correlations. We can assume (without loss of generality as it is explained later) that $\boldsymbol{\mu_0} = 0$ and $\Sigma_0 = \mathbb{I}$

Within this framework, given a low dimensional representation $\mathbf{h}_i$ of $\mathbf{v}_i$, it is straightforward to

compute the best reconstruction $\widehat{\mathbf{v}}_i$ of $\mathbf{v}_i$ as the conditional expectation $\mathbb{E}[\mathbf{v}|\mathbf{h}=\mathbf{h}_i]$:

$$\begin{aligned}
\widehat{\mathbf{v}}_i &= \mathbb{E}[\mathbf{v}|\mathbf{h}=\mathbf{h}_i] \\
&= \int_{\mathbb{R}^D} \mathrm{d}\mathbf{v}\ \mathbf{v} p(\mathbf{v}, \mathbf{h}=\mathbf{h}_i) \\
&= \frac{1}{\int_{\mathbb{R}^D} \mathrm{d}\mathbf{v}\ p(\mathbf{v}|\mathbf{h}=\mathbf{h}_i, \boldsymbol{\theta}) p(\mathbf{h}=\mathbf{h}_i)} \int_{\mathbb{R}^D} \mathrm{d}\mathbf{v}\ \mathbf{v} p(\mathbf{v}|\mathbf{h}=\mathbf{h}_i, \boldsymbol{\theta}) p(\mathbf{h}=\mathbf{h}_i) \\
&= \int_{\mathbb{R}^D} \mathrm{d}\mathbf{v}\ \mathbf{v} p(\mathbf{v}|\mathbf{h}=\mathbf{h}_i, \boldsymbol{\theta}) \\
&= \widehat{W}\mathbf{h}_i + \boldsymbol{\mu}
\end{aligned} \tag{3.41}$$

This shows that FA is equivalent to PCA in the way it encodes relation between hidden (low-dimensional) and visible (full dimensional) variables.

Using standard Bayes rule for gaussians, we can compute the posterior $p(\mathbf{h}_i|\mathbf{v}_i, \boldsymbol{\theta})$ of latent factors $\mathbf{h}_i$ over data. It is still a gaussian $\mathcal{N}(\mathbf{h}|\mathbf{m}_i, \Sigma_i)$ with:

$$\Sigma_i = (\mathbb{I} + \widehat{W}^T \Psi^{-1} \widehat{W})^{-1} \tag{3.42}$$

$$\mathbf{m}_i = \Sigma_i (\widehat{W}^T \Psi^{-1} (\mathbf{v}_i - \boldsymbol{\mu})) \tag{3.43}$$

This shows that FA does not agree with PCA in the way latent factors are computed from visible variables. However in case $\Psi = \sigma^2 \mathbb{I}$ and $\sigma \to 0$, we find $\mathbf{m}_i = \widehat{W}^T(\mathbf{v}_i - \boldsymbol{\mu})$ which is equivalent to eq (3.31) for PCA.

It is interesting to have a look at the way the model is fitted. Marginalising over latent variables one finds:

$$p(\mathbf{v}_i|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{v}_i|\widehat{W}\boldsymbol{\mu_0} + \boldsymbol{\mu}, \Psi + \widehat{W}\Sigma_0\widehat{W}^T) \tag{3.44}$$

From this we see that setting $\boldsymbol{\mu_0} = 0$ and $\Sigma_0 = \mathbb{I}$ does not affect model's predictivity since it is possible to redefine $\widehat{W}$ and $\boldsymbol{\mu}$ in order to encompass any choice of the prior $p(\mathbf{h})$. The variance of the data according to (3.44) is given by:

$$\mathtt{cov}[V] = \widehat{W}\widehat{W}^T + \Psi \tag{3.45}$$

From this equation we learn that, if we set $\Psi = 0$ as is the case of PCA, the variance is approximated by the low rank matrix $\widehat{W}\widehat{W}^T$. By minimizing Frobenius norm, we find the best low rank approximation for covariance matrix and we recover the expression for $\widehat{W}$ previously obtained. It is interesting to observe that if we don't put any constrain on matrix Ψ, low dimensional representation of data is not required to explain data: we could set Ψ to the empirical covariance matrix and choose $\widehat{W} = 0$ which would mean that no relation between high and low dimensional variables is learned. This is the reason why usually in FA Ψ is forced to be non full rank.

3.3.2 Greedy Approximation

We briefly discuss here Greedy Approximation (GA) ([58], [59]) which is an useful numerical approximation method for any function in an Hilbert space. In what follows we will focus on L^2 space of square integrable functions. Strictly speaking, GA does not belong to Machine Learning since it is not a probabilistic model with some free parameters to fit. However, since it achieves very similar results to that of PCA, it is presented here.

It is well known that every function in L^2 can be written as linear combination of an orthonormal

basis of L^2.

$$f = \sum_{k=1}^{\infty} \langle f, \Psi_k \rangle \Psi_k \tag{3.46}$$

where $\{\Psi_k\}_{k=0}^{\infty}$ is an orthonormal basis of L^2 and $\langle \cdot, \cdot \rangle$ is the scalar product of L^2 $\langle a, b \rangle = \int_{-\infty}^{\infty} \mathrm{d}x a^*(x) b(x)$.

The goal of GA is to write a function f as a linear combination of a small set of basis function. GA provides an efficient way to choose which basis function to consider in the expansion. A dictionary $\mathcal{D}$ of functions is created with the condition that $\forall f \in \mathcal{D} \Rightarrow \|f\| = 1$. A number K of the maximum number of basis function to use is also decided by the user. Then each function is approximated by:

$$\widehat{f} = \sum_{g \in \Lambda} c_g g, \qquad \Lambda \subset \mathcal{D}, \qquad |\Lambda| \leq K \tag{3.47}$$

where $|\cdot|$ denotes cardinality of a set. The purpose of an iterative greedy algorithm is the choice of set Λ (i.e. of the approximant function) as well as the coefficients c_g. To find the best approximation of f it runs as follows:

1. Choose K; choose $\mathcal{D}$; set $f_0 := f$; set $m = 0$

2. Choose the best approximant in $\phi_m \in \mathcal{D}$ for f_m satisfying:

$$\phi_m = \arg\max_{g \in \mathcal{D}} \langle g, f_m \rangle \tag{3.48}$$

3. Set $f_{m+1} = f_m - \langle \phi_m, f_m \rangle \phi_m$; update m

4. If $m < K$ go to 2.

5. Define the set of approximant $\Lambda = \{\phi_m\}_{m=0}^{K-1}$

The algorithm tries to iteratively approximate the residuals of f with the best function in the dictionary. It outputs a set Λ of approximant functions. Once Λ is computed, the best approximation $\widehat{f}$ to f si given by:

$$\widehat{f} = \sum_{g \in \Lambda} \langle g, f \rangle \cdot g = \sum_{m=0}^{K-1} \langle \phi_m, f \rangle \cdot \phi_m \tag{3.49}$$

Under certain hypothesys on $\mathcal{D}$, the algorithm is guaranteed to converge to the right solution that is to make arbitrary small the residual magnitude $\|f_K\| = \left\| \widehat{f} - f \right\|$. The choice of $\mathcal{D}$ is crucial. The performances of algorithm strongly depends on that. Using a number of elements from a basis of L^2 (such as Legendre polynomials) can be effective strategy to choose a dictionary.

Greedy approximation is intimately connected to PCA when functions are discretized on a fixed grid: in this case the substitution $f \to \mathbf{f} \in \mathbb{R}^D$ must be done. If one choose a dictionary composed of the eigenvalues $\boldsymbol{\lambda}_j$ of covariance matrix, the GA eq. (3.49) is equivalent to (3.36). Thus we can see PCA as a "smart" way to choose a dictionary. The choice is effective if something is known in advance about the class of functions to be approximated: if so PCA finds general patterns in data and thus functions in dictionary which have more overlaps with a generic function in the dataset. Using a dictionary of PCs is valid as long the function f to approximate is similar to those used for computing PCA. If f is very different, GA might work very poorly and a different (more generic) dictionary must be chosen.

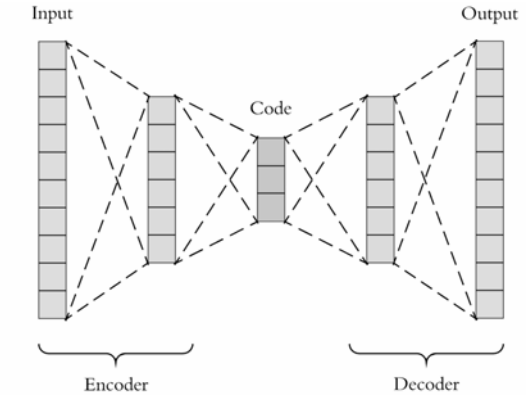

Figure 3.5: Schematic representation of an autoencoder as a sequence of layers which decrease dimensionality until a low dimensional representation of data is learned. It then increase dimensionality again in order to reproduce the input. Picture downloaded from: `https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368`

3.3.3 Autoencoder

Autoencoders are a popular way of reducing data dimensionality. Autoencoders are a particular kind of Neural Network (see section 3.2.5 for more on Neural Networks) which aims to reproduce the identity function. This means that they try to learn a regression in which the inputs are equal to the targets. The task would be trivial if the hidden layers had higher dimensionality than the inputs. However, if the hidden layers have (much) smaller dimension than the input a trained autoencoder can be successfully used for learning reduced representations of data.

Usually an autoencoder is built of two different parts: an *encoder* and a *decoder*. The encoder is a Neural network that is built of layers such that $\dim\mathbf{h}_l > \dim\mathbf{h}_{l+1}$. The dimension of the last layer is the dimension of the reduction. The output of the encoder is then inputted to the decoder which gradually increases its dimensionality until the original data is recovered (for decoder it holds $\dim\mathbf{h}_l < \dim\mathbf{h}_{l+1}$).

A trained encoder can be used to create a reliable low dimensional representation of a dataset. A decoder is then able to reconstruct data. Because of the flexibility of Neural Networks, autoencoders are commonly used in many Machine Learning applications. The main drawback of this approach is the same as neural networks: finding a good architecture is not trivial and fitting the model is an expensive and tricky procedure.

Chapter 4

Machine Learning for Gravitational Waves signal generation

In this chapter, we present our work on GW signal generation with ML. As we will explain below, the aim of the work is to create a ML model which outputs the GW strain generated by a compact object coalescence, when given the orbital parameters of the BBH.

In section 4.2 we will describe thoroughly the model developed for this purpose, as well as the way the model is fitted; furthermore, we will discuss how dataset is prepared. In section 4.3 we will present the `Python` package `mlgw` which gives a complete implementation of the model above. The same code was used to performs all the tests and experiments required for its development. A description of those will be the matter of chapter 5.

Before discussing our work, it might be useful to have a larger insight on previous works on Machine Learning and Gravitational Waves. This is done in the next section 4.1

4.1 Earlier works on Machine Learning and Gravitational Waves

Although very recently, a number of works on the application of Machine Learning to the field of GW data analysis has been proposed. In this section we briefly review some of them so as to describe the broader context in which the present work is located.

In the context of GW data analysis, ML techniques has been successfully applied to perform the following tasks:

- *Signal modelling*: given some source parameters, one aims to produce the GW signal generated by the source. The present work is devoted to this.
- *Signal detection*: given a raw stream of data output by a detector, one wish to state whether the stream contains a physical signal from an astrophysical source.
- *Physical predictions*: once an observed timeseries is deemed to contain a physical signal, one wants to extract some physical information (not necessarily in a Bayesian framework as we did in the previous chapter).

We present below some strategies adopted to perform each task.

Signal modelling Several works (see e.g. [60] and [61]) explored a reduction method for the waveforms called *Reduced ordered modelling* (ROM). Such models represent a waveform as linear combination of a small set of basis waves so that "the continuum of gravitational waves can be represented by a finite and comparatively compact basis". However, such works only perform a reduce order representation and do not provide a regression from the space of orbital parameters to fully reconstruct a wave.

An interesting work on this is in [62]. Using a greedy algorithm (see section 3.3.2) to perform ROM, they represent a waveform onto a reduced basis of function. Once a wave is represented in the low-dimensional space, a NN is trained to learn a relation from the sources parameters to the low dimensional representation. In their work, they are not interested in the problem of reconstructing the full waveform but rather in creating a reliable reduced order representation. The main advantage is that "statistical inference proceeds directly in coefficient space, where it is theoretically straightforward and computationally efficient". They are able to give a fast estimation for the likelihood term in eq. (2.116): the scalar product is not computed in the waveform space (with a high dimension) but in the reduced waveform dimension (with dimension $d = 141$). With this modification, the MCMC algorithm in parameter estimation significantly speeds up. Furthermore, as will be explained below, they use the result in [49] to provide an approximation to the posterior.

A comparative study of different regression methods for generating a BBH coalescence signal were performed by Setyawati et al. in [63]. With different regression methods, they learn a relation from a vector λ of parameters to the coefficients of waveform in an orthonormal basis. They try different regression methods: straightforward fitting (i.e. linear or basis expansion fit), interpolation methods, Gaussian process regression [1] and neural networks. They then compare their results from the point of view of speed and accuracy. They report that, for the case of aligned spins, a simple regression is adequate; for a precessing system results are less clear and are not conclusive.

Besides the ROM approach, Tiglio and Villanueva proposed in [64] a different paradigm. They seek to find a *closed form* expression for the a GW signal using a genetic algorithm. A genetic algorithm is devoted to find a solution to an optimization problem. It works with a "population" of "individuals", each of them carry a possible solution to the problem. Individuals are evaluated according to their *fitness* (i.e. ability to solve the problem) and a mechanism for breeding is set up. If the breeding mechanism favors the fitter individuals, by iterating over many generations a (closely) optimal solution can be found. Within this framework, they created a "population" of functions which describe a GW: this method is called symbolic regression. They were able to find reliable waveforms in closed form for a non-spinning BBH coalescence.

Signal detection Signal detection is essentially a classification problem: given a number of time series on a discrete time grid, each data point is assigned one of the two labels "noise" and "signal". Different classifiers can be trained on such a dataset and they should allow for a speed up in the matched filtering procedure. A commonly used approach makes use of Convolutional Neural Network (CNN): due to convolutional layers, they are most suited for dealing with signals. Dataset are commonly generated using synthetic noise and waveform generated by EOB or IMMR models.

Gabbard et at. in [65] developed a Convolutional Neural Network for performing the same task as matched filtering. They created a dataset of synthetic signals [2] for non spinning BBH coalescence.

[1] A gaussian process is an effective strategy for dealing with interpolation of an unknown function $f(\mathbf{x})$. It models the joint probability $p(f(\tilde{\mathbf{x}}), f(\bar{\mathbf{x}}))$ of training $\tilde{\mathbf{x}}$ and test observations $\bar{\mathbf{x}}$ as jointly gaussian. The prediction of the function at test points are obtained by conditioning over the training points, i.e. using the quantity $p(f(\bar{\mathbf{x}})|f(\tilde{\mathbf{x}}))$

[2] The distribution of datapoint in the dataset reflects the expected distribution of the events to detect. This ensures

Each signal lasted 1 s. Noise magnitude were tuned to achieve a predefined SNR. Time series were given to the CNN as 1D picture and the network were trained to properly classify the signal. They chose a complex architecture with 9 layers of different types (dense, convolutional etc.) and obtained results compared to those of matched filtering. The true alarm probabilities (the same as the false alarm probability) were found very close to those obtained by matched filtering.

Other similar works in this direction are in [66] and [67]. The latter, besides CNN, also tries a random forest approach [3]. They found the two methods to work well with no significant differences between the two approaches. In [68] a CNN is used to perform both signal detection and quick parameter estimation. All the results above are proof of principles as they do not deal with a real case scenario. However, the results obtained show that the application of ML to matched filtering is promising and could save a lot of computation time in the near future.

Physical predictions When it is established that a BBH coalescence signal reached the detector, ML can provide a quick estimation of the source parameters of the event. ML techniques can perform this task in two different ways.

A first option is to provide a "point estimation" of the source parameters. A dataset of waves is created and they are labeled with the true source parameters. As in supervised learning, a NN (usually convolutional) is trained to provide an estimation of the labels. It is important to stress that this method provides the uncertainties due only to the probabilistic model (and not obtained in a Bayesian framework); they are not able to give any information about physical uncertainties of a signal. This approach is followed in [68] (for non-spinning BBH) and in [69] (generalized to the aligned spin case). They built large CNNs and trained them to predict some interesting source quantities such as BHs masses and spins. In [70] a NN provides a point estimation for the source localization in the sky. A quick knowledge of that will help to perform post detection observation in the electromagnetic band, possibly yielding invaluable physical information.

Another option is to use a ML model (often a NN) to represent the posterior $p(\theta|\mathbf{s})$, given the observed signal $\mathbf{s}$: we enter the realm of *Variational Inference*, a promising alternative to Monte Carlo Inference. This option is far more interesting than the previous one because, by providing an approximated posterior, it returns a physically motivated (and Bayesian) prediction.

Chua and Vallisneri in [71] used a Deep Neural Network for this task. The network takes as input a noisy signal and outputs an approximation to the true posterior, either by means of an histogram either parametrically. The network is trained using the KL-divergence $D_{KL}(p||q)$ eq. (2.121) to measure the distance between the actual posterior and the NN approximation $q(\theta|\mathbf{s})$. Crucially, the training does not require the knowledge of the whole posterior $p(\theta|\mathbf{s})$ but only the ability to sample from the prior $p(\theta)$ and from the conditional likelihood $p(\mathbf{s}|\theta)$. Indeed, it can be shown that the KL-divergence can be written as

$$L \simeq -\sum_j \log q(\theta_j|\mathbf{s}_j) \tag{4.1}$$

as long as $\theta_j \sim p(\theta)$ and $\mathbf{s}_j \sim p(\mathbf{s}|\theta_j)$. The first sampling can be achieved by a suitable choice of the prior; the latter is readily done by a surrogate model, using the explicit form of the likelihood eq (2.116). To speed up the sampling procedure, Chua and Vallisneri used the ROM developed in

that the dataset is the closest possible to the observations.

[3] A random forest makes prediction by averaging predictions of a number of decision trees. A decision tree is a popular ML algorithm which provides a cover of the input set by making binary choices. In each set of the cover a single prediction model is fitted. Given an input, the tree is able to recover which regression model to use and thus giving a very accurate prediction.

[62]. They addressed a simplified case with a (very) small parameter space. However, as their results indicate that the approach is feasible and yields meaningful results, the work suggests that more work can be done to address the full case.

A closely related work is done by Gabbard at al. in [48]. They built a Conditional Variational Auto-Encoder (CVAE) [72] designed to give an approximation $q(\theta|\mathbf{s})$ of the true posterior. A CVAE is built of two stacked NNs which provides the required output. An encoder $r_E(\mathbf{z}|\mathbf{s})$ provides a latent, low-dimensional representation $\mathbf{z}$ of the signal; a decoder $r_D(\theta|\mathbf{z},\mathbf{s})$ is able to map the latent variable $\mathbf{z}$ to the high dimensional parameter space. The approximation of the posterior is obtained by marginalizing over $\mathbf{z}$:

$$q(\theta|\mathbf{s}) = \int \mathrm{d}\mathbf{z}\; r_D(\theta|\mathbf{z},\mathbf{s}) r_E(\mathbf{z}|\mathbf{s}) \tag{4.2}$$

To make the loss function tractable, one must introduce a second encoder $\bar{r}_E(\mathbf{z}|\mathbf{s},\theta)$ at training time. Heuristically, $\bar{r}_E$ helps to learn a proper latent representation by inputting also the results of the decoding θ. At test time however, this network must not be exploited. Like Chua and Vallisneri, they obtained a training set by sampling parameters θ from the prior: this is enough to guarantee that the learnt relation agrees with the true posterior. Using this methods, Gabbard at al. were able to obtain a reliable estimation of the true posterior. Their results are similar to those by Chua and Vallisneri.

We briefly mention that some work has been done also in other fields than parameter estimation for BBH coalescence. For instance, Haegel et al. in [73] used a Deep Neural Network to provide a point estimate for the properties of BBH merger remnants. They created a dataset built from the publicly available catalog of NR simulation and learnt a regression from the initial orbital parameters to the mass and spin of the resulting Kerr's BH. On the other hand, Chan et al. [74] addressed the "detection and classification of Supernova GW signals". They used a publicly available dataset of NR generated supernova GW signals; the signals were buried in the typical LIGO interferometer noise. A CNN were trained to discern between signal and noise and to provide a classification of their explosion mechanism. Interestingly, their results suggests that a supernova signal can be effectively recognized with a network of advanced interferometers.

4.2 The model

We now discuss the details of the ML model we use to generate a GW signal, starting from the BBH orbital parameters. Our first task will be to explain the main motivation to build such a model. After, we will give a formal definition of the problem we want to solve. We will then explain how the problem is solved.

4.2.1 Motivation

In order to describe a BH binary system, one must specify the properties of the two orbiting BHs. Because of no hair theorem, each BH (labeled by $i = 1, 2$) of the binary is fully specified by its mass m_i and dimensionless spin parameter $\mathbf{s}_i = \frac{\mathbf{J}_i}{m_i^2}$. J_i denotes the angular momentum of the BH. As mentioned in section 2.3, the time evolution of the system and the emitted gravitational radiation depend non trivially on those quantities. As we are interested in computing the waveform of the gravitational radiation of this system, we need to cope with this difficulty. A number of physically motivated computational techniques were developed to this extent and they give a reasonably accurate solution to the problem.

However they have at least two shortcomings. First of all, none of them provides a closed form solution for the waveform. Furthermore - and this is even worse - in order to achieve high accuracy, they consume a lot of computational power. Unfortunately any GW data analysis relies heavily on these models. For example, every signal detection requires an accurate parameter estimation (see section 2.4.3) that is performed by running a long Markov-Chain Montecarlo for sampling the posterior. Each point drawn from the chain requires the generation of a waveform. The orbital parameters required to generate the waveform are not known *a priori* and thus the waveform generation must be done online. This can be very expensive: a precise parameter estimation can take up to two weeks.

Of course, this sets a great limitation in the ability to extract physical information from a signal. The overcoming of such drawbacks is the main motivation for developing a fast Machine Learning model to generate a GW waveform. If one manages to create such a model, a significant speed up can be reached: this can result either in less computing time either in more accurate predictions. As the number of GW is expected to increase (while detectors sensitivity raises), the issue becomes more and more urgent.

A closed form expression for the model can be useful also in gradient based sampling methods. Methods in this class are a variant of MCMC which makes use of likelihood gradient to effectively explore the parameter space (see e.g. Hamiltonian Montecarlo [75]). The gradients are used to move towards high likelihood regions of the parameter space and the sampling gets more efficient. With standard methods, the computation of gradients is expensive and it is not routinely performed. An analytical model, providing a closed form gradient, will enhance the effectiveness of such methods: a further speed up in parameter estimation is likely to be achieved.

Of course, another interesting alternative for a speed up in parameter estimation is to build a ML model which approximates the posterior. However, pursueing this path gives no guarantee for high accuracy results. Variational inference has no theoretical assurance to provide an arbitrary approximation; furthermore, approximating a complicated function of 15 parameters is not trivial at all. Our choice is somewhat less radical. We choose to keep the Monte Carlo inference for dealing with the posterior and we focus on its bottleneck, i.e. the waveform generation. This might give worse time performance when compared to the promises of variational inference. However, the approach is more straightforward and has more chances to achieve both speed-up in runtime and high accuracy results.

4.2.2 Problem specification

We now provide a precise statement of the objective of our work. A binary black hole is parametrized by a vector $\boldsymbol{\vartheta} = (m_1, m_2, \mathbf{s}_1, \mathbf{s}_2)$, where m_i are the BHs masses and $\mathbf{s}_i = \frac{\mathbf{S}_i}{m_i^2} < 1$ are the *dimensionless* spin. We call them the *orbital parameters*. Let the wave direction of propagation be identified with the spherical coordinates (d_L, ι, φ_0), where d_L is the luminosity distance, ι the polar angle and φ_0 the azimuthal angle. The polar angle ι is also called inclination and it is measured with respect to the normal to the orbital plane. A GW is parametrized as in eq. (2.58) [4]:

$$h(t, d_L, \iota, \varphi_0; \boldsymbol{\vartheta}) = h_+ + i h_\times = \frac{1\,\mathrm{Mpc}}{d_L} \sum_{m=-2}^{2} {}^{-2}Y_{2m}\,(\iota, \varphi_0) \cdot H_{2m}(t; \boldsymbol{\vartheta}) \tag{4.3}$$

In the expression we included only the lowest order term in the multipole expansion, which is the dominant term in every realistic case. We call $(m_1, m_2, \mathbf{s}_1, \mathbf{s}_2, d_L, \iota, \varphi_0)$ *physical parameters* and they

[4] In eq. (2.58), a dependence on total mass is factored out together with d_L dependence. Here, in order to keep the $\boldsymbol{\vartheta}$ dependence on a single function, we included the total mass term in H_{2m}.

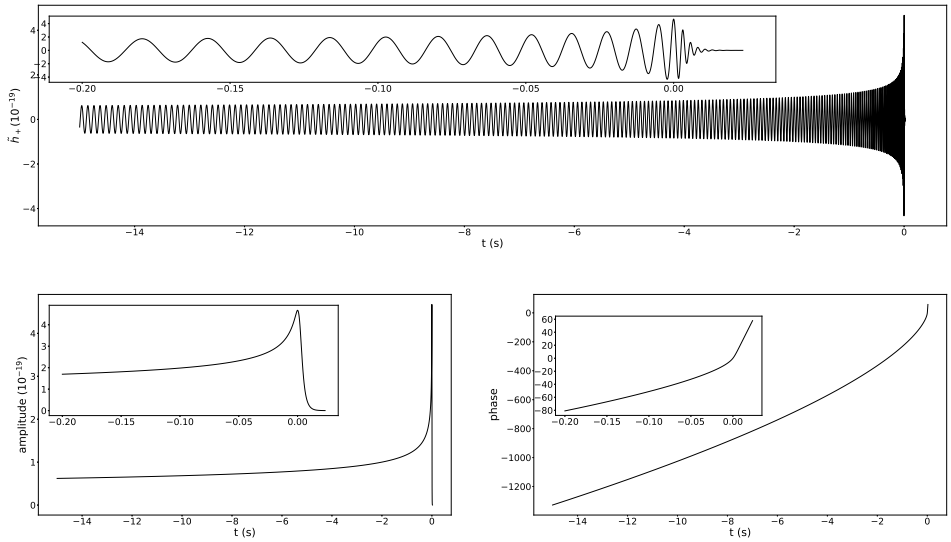

Figure 4.1: An example of the amplitude $A(t;\boldsymbol{\vartheta})$, phase $\phi(t;\boldsymbol{\vartheta})$ and strain $\tilde{h}_+(t;\boldsymbol{\vartheta}) = \mathrm{Re}\left[A(t;\boldsymbol{\vartheta})e^{i\phi(t;\boldsymbol{\vartheta})}\right]$. Reproducing them with a ML model (see eqs. (4.6)) is the objective of the present work. They represent a wave seen at a distance of 1 Mpc at an inclination angle $\iota = 0$ and with $\varphi_0 = 0$. Waves are generated with orbital parameters $\boldsymbol{\vartheta} = (m_1 = 30\,\mathrm{M}_\odot, m_2 = 15\,\mathrm{M}_\odot, s_1 = 0.4, s_2 = -0.3)$ and start 15 s before merger. In the above panel the full wave is represented; in the left lower panel, the amplitude is represented, while in the right lower panel we plot the phase. In the insets, we show the last 200 ms before merger. The goal of the present work is to reproduce GW signals like these.

fully express the source proprieties as well as its orientation and position.

In what follows, we will limit to the case in which spins $\mathbf{s}_1$ and $\mathbf{s}_2$ are aligned. In fact, when a BBH has non-aligned spins, the physical scenario gets (much) more complicated: the orbital angular momentum changes with time. This means that the orbital plane is not constant and the system precedes. Such systems are very complicated to tackle, since there is not a straightforward way to parametrize the direction of orbital angular momentum (orthogonal to the orbital plane). Our working hypothesis allow us to avoid the (huge) complication of dealing with precessing systems and to focus on the modelling of the waveform. The expertise learnt with this task could eventually lead to create a model for the full precessing case. However, it is important to stress that the case of aligned spins is not unrealistic: it still includes a lot of phenomenology and can be successfully used to perform a full parameter estimation [5].

It is crucial to realise that, in the case of aligned spins, equation (4.3) can be further simplified. It turns out that $|H_{2\pm2}| >> |H_{2\pm1}|, |H_{20}|$ and this means that the $|m| \neq 2$ modes of the multipole expansion can be safely neglected. Furthermore, because of parity invariance, it holds $H_{22} = H^*_{2-2}$. Thus a waveform can be fully expressed by the single complex quantity H_{22}. We define:

$$\tilde{h}(t;\boldsymbol{\vartheta}) \equiv {}^{-2}Y_{22}\,(0,0) \cdot H_{22}(t;\boldsymbol{\vartheta}) = 4 \cdot \sqrt{\frac{5}{64\pi}} H_{22}(t;\boldsymbol{\vartheta}) \tag{4.4}$$

[5]Indeed, the LIGO/Virgo collaboration uses routinely models of class "IMRPhenom" to compute their physical predictions (see e.g. [40]). Models in this class only deal with non-precessing systems and their outputs are comparable with those of EOB methods, which are able to tackle the precessing case.

With this definition, the full waveform can be expressed as:

$$\begin{aligned} h(t, d_L, \iota, \varphi_0; \boldsymbol{\vartheta}) &= \frac{1\,\mathrm{Mpc}}{d_L} \left[{}^{-2}Y_{22}\left(\iota, \varphi_0\right) \cdot H_{22}(t; \boldsymbol{\vartheta}) + {}^{-2}Y_{2-2}\left(\iota, \varphi_0\right) \cdot H^*_{22}(t; \boldsymbol{\vartheta}) \right] \\ &= \frac{1\,\mathrm{Mpc}}{d_L} \cdot \left\{ \frac{1+\cos^2\iota}{2} \left[\cos 2\varphi_0 \cdot \mathrm{Re}(\tilde{h}) - \sin 2\varphi_0 \cdot \mathrm{Im}(\tilde{h}) \right] \right. \\ &\left. + i \cdot \cos\iota \left[\sin 2\varphi_0 \cdot \mathrm{Re}(\tilde{h}) + \cos 2\varphi_0 \cdot \mathrm{Im}(\tilde{h}) \right] \right\} \end{aligned} \tag{4.5}$$

where, in the last equality, we split the real and the imaginary part of the strain and dropped the dependence on $(t; \boldsymbol{\vartheta})$.

We express H_{22} in terms of two functions $A(t; \boldsymbol{\vartheta})$ and $\phi(t; \boldsymbol{\vartheta})$ such that:

$$\tilde{h}(t; \boldsymbol{\vartheta}) = A(t; \boldsymbol{\vartheta}) e^{i\phi(t; \boldsymbol{\vartheta})} \tag{4.6}$$

We may also write $f_{\boldsymbol{\vartheta}}(t)$ to denote a function $f(t; \boldsymbol{\vartheta})$ of time with parametric dependence on $\boldsymbol{\vartheta}$. The goal of the present work is to provide an accurate ML model for functions A and ϕ. We should be able to reproduce the current state-of-the-art time domain waveforms (i.e. from EOB surrogate model, see subsection 2.3.4). Once they are known, any physical prediction regarding GWs can be done, as well as the analysis of a detector data stream.

Functions $A(t; \boldsymbol{\vartheta})$ and $\phi(t; \boldsymbol{\vartheta})$ encode the non trivial dependence of the wave on the orbital parameters. The dependence on geometric parameters ι, φ_0 and d_L is analytically known and thus can be factored out. It is interesting to compare equation (4.6) to the detected signal in (2.113). In eq (2.113) the full dependence of signal $\hat{s}$ from all the 15 parameters is written explicitly; the equations above parametrize only the part depending on the source and its orientation with respect to the observer and they do not consider the detector pattern (which varies from different detectors).

The physical meaning of A and ϕ is readily understood. Apart from a numerical constant, the quantity $\tilde{h}(t; \boldsymbol{\vartheta}) = A(t; \boldsymbol{\vartheta}) e^{i\phi(t; \boldsymbol{\vartheta})}$ is the $h = h_+ + i h_\times$ strain, generated by a BBH with orbital parameters $\boldsymbol{\vartheta}$, when observed from the direction orthogonal to the total angular momentum ($\iota = 0$), at a luminosity distance $d_L = 1\,\mathrm{Mpc}$ and with $\varphi_0 = 0$. $A(t)$ represent the amplitude of $\tilde{h}_+$ as a function of time, while $\phi(t)$ represent the phase of $\tilde{h}_+$ as a function of time. In figure 4.1, we show an example of such quantities.

We choose to parametrize a wave with amplitude and phase. Another natural choice is to work with the real and imaginary part of the polarization. The latter choice, however, is most prone to numerical inaccuracies: the polarization is a fast oscillating function and tracking its behavior is difficult. Amplitude and phase, on the other hand, are slowly varying functions and it is easier to capture their trend.

Having restricted to the case of aligned spins, the set of independent orbital parameters to consider is $\boldsymbol{\vartheta} = (m_1, m_2, s_1, s_2)$ where s_i is the non zero component of the i-th BH. Without loss of generality the spins are aligned to the z-direction. For reasons made clear in below, it is convenient to give the following parametrization for the parameter space. Let us define the mass ratio q and the total mass M.

$$q = \frac{m_1}{m_2} > 1 \qquad M = m_1 + m_2 \tag{4.7}$$

with the convention that $m_1 > m_2$. A useful parametrization is then $\boldsymbol{\vartheta} = (q, M, s_1, s_2)$. We can invert

the relation above to find:

$$m_1 = \frac{q}{1+q}M \qquad m_2 = \frac{1}{1+q}M \tag{4.8}$$

With the above definitions we state our problem as follow. We wish to find a ML model that reliably represents the following map:

$$(q, M, s_1, s_2) \longmapsto A_{(q,M,s_1,s_2)}(t) \tag{4.9}$$

$$(q, M, s_1, s_2) \longmapsto \phi_{(q,M,s_1,s_2)}(t) \tag{4.10}$$

where $A(t)$ and $\phi(t)$ are as in equation (4.6).

4.2.3 Building blocks of the model for signal generation

Modelling dependence on total mass Before developing the full ML model, the problem in equations (4.9) and (4.10) can be further simplified using the global scale invariance of general relativity in vacuum. This will allow us to model analytically the dependence on total mass M.

Following [76], we recognize that the free theory lacks of a built in scale. For this reason, a scale coordinate transformation $x^\mu \to \lambda x^\mu$ must not affect the physics as long as the other quantities (i.e. times, masses frequencies etc.) are scaled accordingly. With some dimensional considerations (supported by rigorous computations in GR), one finds, in the case of a BBH merger, that the following scaling must be simultaneously applied when scaling the coordinates:

$x \to \lambda x$	(distance)	$t \to \lambda t$	(time)
$M_i \to \lambda M_i$	(mass)	$p_i \to \lambda p_i$	(momentum)
$J_i \to \lambda J_i$	(angular momentum)	$s_i \to s_i$	(spin parameter)
$E \to \lambda E$	(energy)	$f \to \lambda^{-1} f$	(frequency)
$h \to h$	(strain)	$v \to v$	(velocity)

This means that, if one performs simultaneously all the global transformation above, the equation are invariant. This can be applied to the case of GW generated by a BBH coalescence.

$$h'(r', t'; m_1', m_2', s_1', s_2') = h(\lambda r, \lambda t; \lambda m_1, \lambda m_2, s_1, s_2) = h(r, t; m_1, m_2, s_1, s_2) \tag{4.11}$$

The relation is really useful: once a waveform is computed for a system with total mass $M = m_1 + m_2$ and arbitrary spins, it is straightforward to compute the wave for a system with unchanged spins but different total mass, as long as the two masses keep the same ratio.

Factoring out the dependence on distance ($\propto \frac{1}{d_L}$) in eq. (4.11), one can find how scale invariance can be used to model the total mass dependence of amplitude $A_{\boldsymbol{\vartheta}}$ and phase $\phi_{\boldsymbol{\vartheta}}$:

$$\frac{1}{M'} A_{(q,M',s_1,s_2)}\left(\frac{t}{M'}\right) = \frac{1}{M} A_{(q,M,s_1,s_2)}\left(\frac{t}{M}\right) \tag{4.12}$$

$$\phi_{(q,M',s_1,s_2)}\left(\frac{t}{M'}\right) = \phi_{(q,M,s_1,s_2)}\left(\frac{t}{M}\right) \tag{4.13}$$

where M, M' are arbitrary total mass. Thus, we need to develope a ML model only to fit the relation

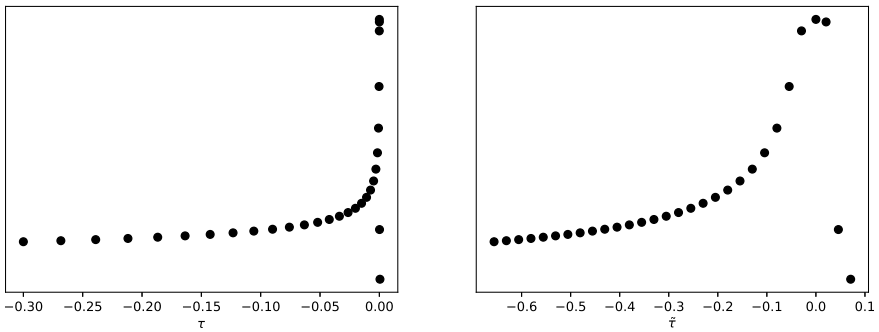

Figure 4.2: Amplitude (arbitrary units) of a GW wave represented on a grid with 30 points. On left panel, amplitude values are plotted in the reduced time grid $\boldsymbol{\tau}$ (units in $\frac{s}{M_\odot}$). On the right panel, the same function is represented as a function of the transformed grid $\tilde{\boldsymbol{\tau}}$ (units in $\left(\frac{s}{M_\odot}\right)^\alpha$). It is set $\alpha = 0.35$. It is manifest that $\boldsymbol{\tau}$ grid is finer around the merger. On the other hand in the transformed grid $\tilde{\boldsymbol{\tau}}$, amplitude has a smoother behaviour.

$(q, s_1, s_2) \longmapsto A_{(q,\tilde{M},s_1,s_2)}(t)$ (and $\phi_{(q,\tilde{M},s_1,s_2)}(t)$) with a fixed $\tilde{M}$. Once this model is fitted, one can use equations (4.12) and (4.13) to capture the dependence on total mass. With this, the only parameters to consider in a ML model will be just $\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2)$: now we are ready to tackle our problem.

The time grid The first task to build a waveform generator is to choose a representation of the waveforms. Each function to fit f (i.e. amplitude and phase) must be represented by its values $\mathbf{f} \in \mathbb{R}^D$ on a discrete grid of points $\mathbf{t} \in \mathbb{R}^D$. In equations (4.12) and (4.13), only dimensionless time $\tau = \frac{t}{M}$ appears and it is then convenient to work in a grid of reduced time $\boldsymbol{\tau}$. The time grid is chosen with the convention that at $\tau = 0$ the merger happens (i.e. the amplitude has a maximum). Even though in our units (where $G = c = 1$) τ is dimensionless, we use the more understanble units $\frac{s}{M_\odot}$. Both amplitude and phase are represented by:

$$\mathbf{f}(\tilde{\boldsymbol{\vartheta}})_i = f_{\tilde{\boldsymbol{\vartheta}}}(\boldsymbol{\tau}_i) \tag{4.14}$$

where f stands for any of the functions $A_{\tilde{\boldsymbol{\vartheta}}}(t)$ and $\phi_{\tilde{\boldsymbol{\vartheta}}}(t)$.

Of course, representing a function with a discrete vector is an approximation to the real function. The value at an arbitrary time must be found by interpolation and introduces an error in the value of the function. The spacing of the grid must be small enough so that the error is small; however, it can not grow too much, as it gets unfeasible work with such a large amount of data. To make the interpolation the effective, the points in the (reduced) time grid must be chosen by looking at the functions to represent on the grid. Heuristically, we want a finer grid when the function changes much, and viceversa. Clearly an equally spaced grid over times is not the best choice since the amplitude has a very narrow peak at $\tau = 0$ [6]. A good solution is to create an equally spaced grid $\tilde{\boldsymbol{\tau}}$ for the variable

$$\tilde{\tau} \equiv \operatorname{sign}\tau \cdot (|\tau|)^\alpha; \quad \alpha < 1 \tag{4.15}$$

and build the τ grid $\boldsymbol{\tau}$ as:

$$\boldsymbol{\tau}_i = \operatorname{sign}\tilde{\boldsymbol{\tau}}_i (|\tilde{\boldsymbol{\tau}}_i|)^{\frac{1}{\alpha}} \tag{4.16}$$

[6] As the phase has a rather regular behavior, it is not important to choose the right grid. For this reason a single grid for amplitude and phase, tuned on the behavior of amplitude, is used.

We call α *distortion parameter*. As can be seen in figure 4.2, this choice ensures that more points are accumulated around the merger and provides an adequate resolution for this part. Far from the merger the amplitude changes slowly and a few grid points are required to reliably sample the function. As we will see in the next chapter, a good choice is to set $\alpha = 0.35/0.5$. In order to represent faithfully a wave, a grid must have around 2500 points; see section 5.1.1 for a discussion on how the decision is made.

In this framework, our problem reduces to performing the regression:

$$\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2) \in \mathcal{P} \longmapsto \boldsymbol{A}_{\tilde{\boldsymbol{\vartheta}}} \in \mathbb{R}^D \tag{4.17}$$

$$\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2) \in \mathcal{P} \longmapsto \boldsymbol{\phi}_{\tilde{\boldsymbol{\vartheta}}} \in \mathbb{R}^D \tag{4.18}$$

where $\mathcal{P} \subset \mathbb{R}^3$ is the domain in which the physical parameters are allowed to vary. Spins must be in range $[-1, 1]$ and the mass ratio must satisfy $q > 1$: the choice of $\mathcal{P}$ must satisfy those requirements. The domain $\mathcal{P}$ depends on the limitations of the numerical models (in our case EOB) used to train the ML model. Usually, PN models are calibrated for domains $\mathcal{P} \simeq [1, 20] \times [-0.9, 0.9] \times [-0.9, 0.9]$. A conservative choice leads to use a subset of the full calibration domain.

Dataset creation As in any ML method, we must create a dataset in order to perform training. The dataset $X \in \mathbf{Mat}(N, 3 + 2D)$ of N waveform, useful to train this model, has the following form:

$$X_{i:} = [q, s_1, s_2, \boldsymbol{A}^T_{\tilde{\boldsymbol{\vartheta}}}, \boldsymbol{\phi}^T_{\tilde{\boldsymbol{\vartheta}}}] \tag{4.19}$$

where $X_{i:}$ denotes the i-th row of the dataset matrix. Of course the dataset dimensionality depends on the size D of the time grid.

Each row holds a random sample $\tilde{\boldsymbol{\vartheta}}_i \sim \text{Unif}(\mathcal{P})$ [7] and the amplitude and phase for the wave generated with that orbital parameters $\tilde{\boldsymbol{\vartheta}}_i$. To generate the training waves a standard surrogate time-domain model can be used. In our work, we used model from EOB family. Waves are generated with inclination angle $\iota = 0$ and are evaluated at a standard luminosity distance $d_0 = 1\,\text{Mpc}$. All the sources have $M = 20\,\text{M}_\odot$. Surrogate models generate the strain h of the wave; amplitude and phase must be extracted from there. The output of a surrogate model must be interpolated to the chosen grid and must be trimmed wherever it provides a zero amplitude or a nonphysical signal.

It is important to ensure that all waves have zero phase at a constant time point. This is crucial in order to have a continuous dependence of the phase components on the orbital parameters; a random alignment would wash out every relation, making impossible to learn any regression. Phases can be aligned either at merger (i.e. $\phi(\tau = \tau_{merger}) = 0$), either at the beginning of the signal (i.e. $\phi(\tau = \tau_0) = 0$). None of the two options was found better than the other.

To avoid numerical inaccuracies, the waves in the dataset must be computed as precisely as possible. For EOB models, this means that a very small integration step for the equation of motion must be chosen ($O(10^{-6}\text{s})$). Accurate models minimise numerical errors in the phase and amplitude of the waves and, for this reason, they lower the noise in the relation to learn.

Dimensionality reduction Once we are able to represent waveforms, performing a regressions $\tilde{\boldsymbol{\vartheta}} \longmapsto \boldsymbol{A}_{\tilde{\boldsymbol{\vartheta}}}, \boldsymbol{\phi}_{\tilde{\boldsymbol{\vartheta}}}$ is unfeasible. The dimension of the target space is too large for this task. Luckily, the components of $\boldsymbol{A}, \boldsymbol{\phi}$ are strongly correlated with each other: the independent amount of information,

[7] We denote by Unif($\mathcal{P}$) a uniform probability distribution on the set $\mathcal{P}$. This means that $\forall A \subseteq \mathcal{P} \to p(A) = \frac{\text{vol}(A)}{\text{vol}(\mathcal{P})}$

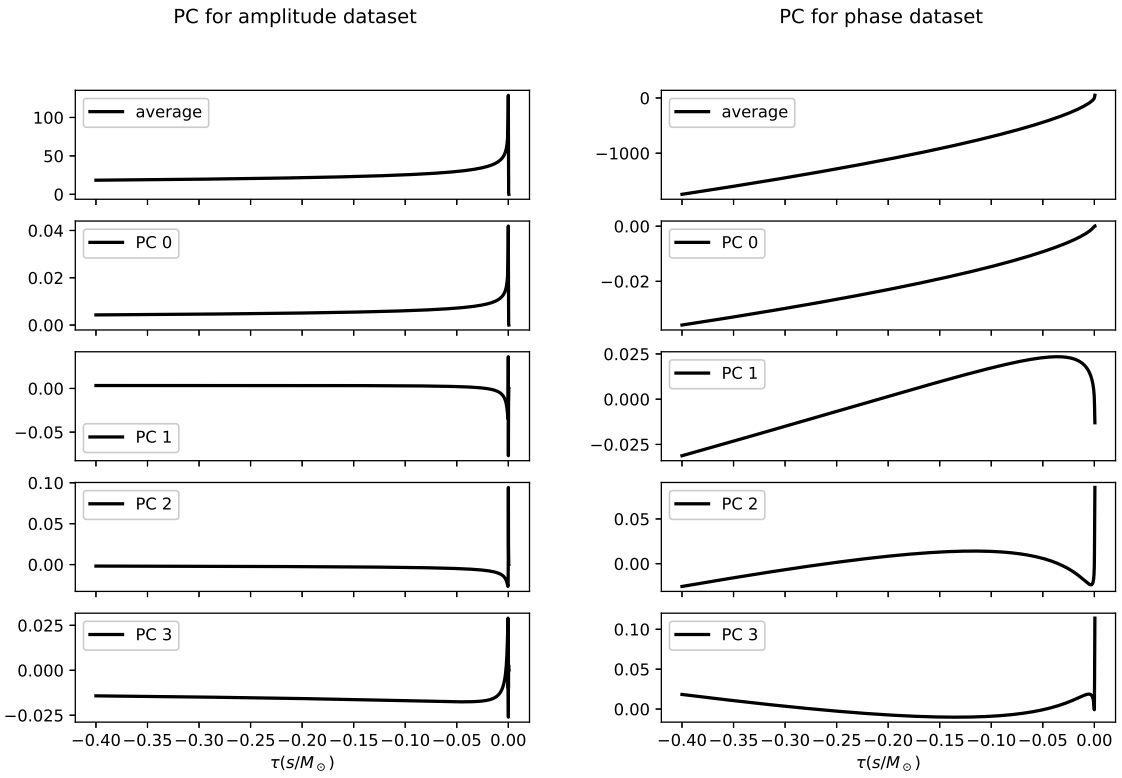

Figure 4.3: The average μ and the first 4 Principal Components for the amplitude (left) and phase (right) dataset. They are plotted as a function of reduced time $\tau = \frac{t}{M}$. Dataset is generated with 2039 waves in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$. PCs are scaled so that they form an orthonormal set. Note that amplitude has little variation before merger: its PCs are zero before merger as they show little variance in that part of the waveform.

required to fully reconstruct the wave, can be stored in a low dimensional vector. A number of ML techniques to perform such a task are available (see section 3.3). Among them, Principal Component Analysis (see 3.3.1) was found to be particularly effective.

A PCA model is trained: it represents an (approximate) bijective map between the high dimensional waveform $\mathbf{f} = \boldsymbol{A}, \boldsymbol{\phi} \in \mathbb{R}^D$ and the low-dimensional representation $\mathbf{g} = \mathbf{g}_A, \mathbf{g}_\phi \in \mathbb{R}^K$. Here D is the number of grid points; K is the dimension of low-dimensional representation (i.e. the number of principal components considered). In what follows, $\mathbf{f}$ will be the high order representation of any of the function $A_{\tilde{\boldsymbol{\vartheta}}}(t)$, $\phi_{\tilde{\boldsymbol{\vartheta}}}(t)$, while $\mathbf{g}$ will be the PCA reduced lower order representation of $\mathbf{f}$. The relation takes the following form:

$$\mathbf{g} = W(\mathbf{f} - \boldsymbol{\mu}) \tag{4.20}$$

$$\mathbf{f} = W^T \mathbf{g} + \boldsymbol{\mu} \tag{4.21}$$

where, as usual in PCA, $W \in \mathbf{Mat}(K, D)$ is the matrix whose colums are the first K eigenvalues of the empirical covariance matrix. $\boldsymbol{\mu}$ is the empirical mean vector: $\mu_j = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{f}_i)_j$. In the equations above, the $\tilde{\boldsymbol{\vartheta}}$ is omitted for notational simplicity.

Equation (4.21) will be used to reconstruct the full waveform, once the reduced coefficients are guessed starting from the orbital parameters $\tilde{\boldsymbol{\vartheta}}$. It is clear that two PCA models must be fitted, one

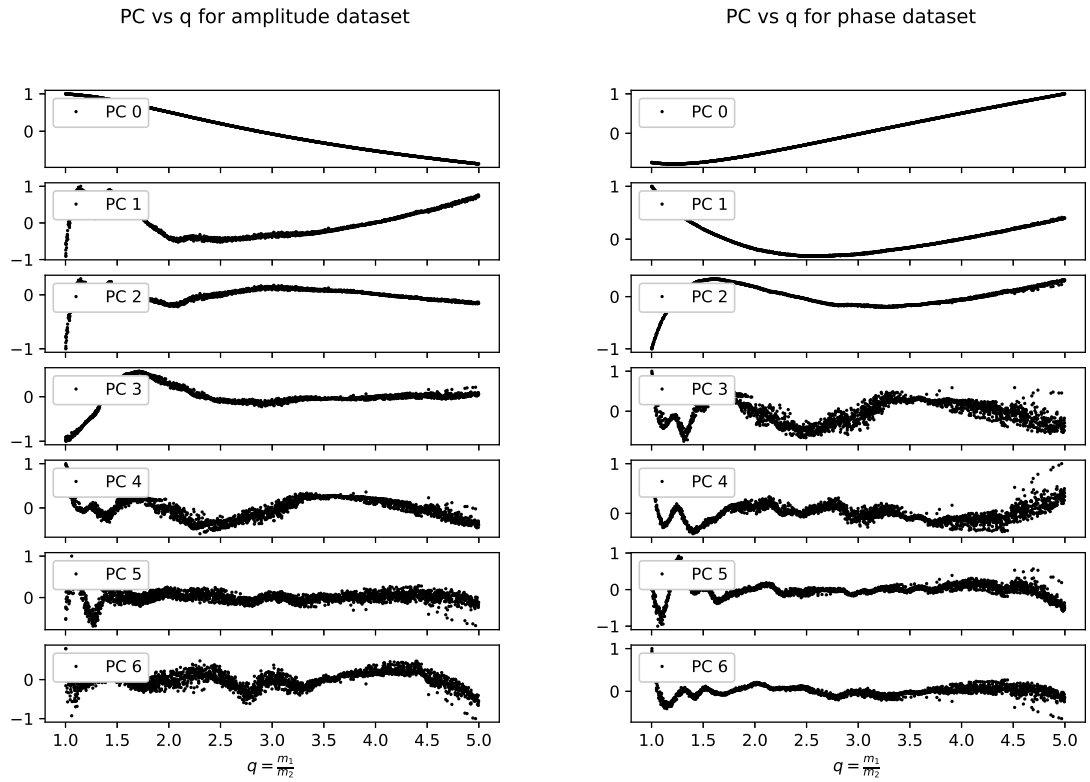

Figure 4.4: Here we plot the projection of GWs on the first 6 PCs as a function of the mass ratio $q = \frac{m_1}{m_2}$; more formally, we plot the quantities $(\mathbf{g}(q))_i$ in eq. (4.20) for both amplitude (left panel) and phase (right panel). The index i labels the order of the PC which the wave is projected to. The relation above are the objective of regression in eq. (4.22). To make the relation more clear, we chose to use a dataset where spins are fixed: $\mathcal{P} = [1., 5.] \times \{-0.3\} \times \{0.2\}$. EOB integration step is chosen to be 10^{-6}s. It is interesting to note that, as the order of the PC increases, the relation becomes more and more noisy. This is explained as follows: as the PCA perturbative expansion goes to higher order, it is able to reconstruct the waves at smaller and smaller scales. Eventually, it will reach the noise scale and thus the observed relation $PC(q)$ will be noisy as well. To push the noise errors to higher order PCs, the waves must be generated with increasing precision (see section 5.1.1 and figure 5.3 for more on this).

for amplitudes, one for phases. In figure 4.3 a number of PC for phase and amplitude dataset are represented.

Regression Once a dimensional reduction (and reconstruction) scheme is available, the aim is to perform the regression

$$\tilde{\boldsymbol{\vartheta}} \longmapsto \boldsymbol{g}(\tilde{\boldsymbol{\vartheta}}) \tag{4.22}$$

As described in section 3.2, a number of ML models are avaiable for this purpose. The model Mixture of Experts (see 3.2.4), with *softmax* gating function, is found to be a good choice.

To fit the MoE, one must reduce the dimensionality of the dataset, using PCA. The new dataset consists in two matrices X and G such that:

$$X_{ij} = (\tilde{\boldsymbol{\vartheta}}_i)_j \qquad G_{ij} = (\boldsymbol{g}_i)_j = (\boldsymbol{g}(\tilde{\boldsymbol{\vartheta}}_i))_j \tag{4.23}$$

A MoE model will be fitted using this dataset. The EM algorithm is a good choice for the training. The user must choose a number of hyperparameters: the number L of experts to use, the basis functions features $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \in \mathbb{R}^M$ to use and some fitting parameters (see next section 4.3).

As in section 3.2.1, the basis functions expansions allows to use the machinery for a linear regression to a more flexible polynomial regression. Vector $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}})$ is defined as:

$$\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) = [\xi_1(\tilde{\boldsymbol{\vartheta}}), \dots, \xi_M(\tilde{\boldsymbol{\vartheta}})]^T \tag{4.24}$$

where ξ_i $i = 1, \dots, M$ are the basis functions, chosen by the user. A careful choice of basis functions can really make a difference in fit performances. Indeed, the more similar is the basis function behaviour to the actual relation, the more reliable is the regression. This would bring the user to include as many basis functions as possible to make the model more flexible. However, due to overfitting concerns, one must not include too many of them. A choice of basis function must be done at validation time, by comparing performances of different models. It is found that including in the ξ_i all every monomial up to 3/4th order in the three components of $\tilde{\boldsymbol{\vartheta}}$ is a good working choice.

As MoE model deals with single dimensional outputs, a single independent regression must be performed for each component g_k of $\mathbf{g} \in \mathbb{R}^K$ [8]. In general, a regression will be a collection of MoE weights $\{\{\mathbf{w}_l^{(k)} \in \mathbb{R}^M\}_{l=1}^{L_k}, V^{(k)} \in \mathbf{Mat}(M, L_k)\}_{k=0}^K$, where index k labels different regressions for each PC and index l labels the expert of a given model. As in eq (3.18), $\mathbf{w}_l^{(k)}$ are the linear regression coefficients and $V^{(k)}$ is the softmax regression weight matrix.

Once the model is fitted, one can use eq (3.18) to predict each of the K component g_k of $\boldsymbol{g}$, given an input $\tilde{\boldsymbol{\vartheta}}$:

$$g_k = (\boldsymbol{g}(\tilde{\boldsymbol{\vartheta}}))_k = \sum_{l=1}^{L_k} \mathbf{w}_l^{(k)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \cdot \mathcal{S}(V^{(k)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}))_l \tag{4.25}$$

where $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}})$ is the basis function expansion chosen for the regression (see section 3.2.1) and L_k denotes the number of experts chosen for the k-th regression. Figure 4.4 shows an example of the relations that must be fitted by the MoE model.

4.2.4 Summary of the model

We now summarise our ML to generate the GW waveforms for BBH coalescence.

As in equations (4.5) and (4.6), a waveform is represented by two functions $A(t; \boldsymbol{\vartheta})$ and $\phi(t; \boldsymbol{\vartheta})$ where $\boldsymbol{\vartheta} = (q, M, s_1, s_2)$. Each function is approximated by a vector $\mathbf{f} \in \mathbb{R}^D$ which holds their values on a (reduced) time grid. The dependence on total mass M can be factored out, using scale invariance of GR. This reduces the number of independent parameters to 3; we denote them by $\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2)$. The regression $\tilde{\boldsymbol{\vartheta}} \longmapsto \mathbf{f}$ is performed in two steps. First a MoE regression which outputs a low dimensional representation $\mathbf{g}$ of $\mathbf{f}$; then a PCA model which reconstruct the waveform $\mathbf{f}$, based on $\mathbf{g}$.

[8] This is not a great limitation, because, due to orthogonality of PCs, each g_j is independent from the other: we do not miss correlation among different regressions.

The model has the following explicit form:

$$\text{model} : \mathbb{R}^3 \to \mathbb{R}^K \to \mathbb{R}^D \tag{4.26}$$

$$\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2) \longmapsto \mathbf{g}(\tilde{\boldsymbol{\vartheta}}) = \begin{pmatrix} \sum_{l=1}^{L_1} \mathbf{w}_l^{(1)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \cdot \mathcal{S}(V^{(1)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}))_l \\ \vdots \\ \sum_{l=1}^{L_K} \mathbf{w}_l^{(K)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \cdot \mathcal{S}(V^{(K)\,T} \boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}))_l \end{pmatrix} \longmapsto \mathbf{f}(\tilde{\boldsymbol{\vartheta}}) = W^T \mathbf{g}(\tilde{\boldsymbol{\vartheta}}) + \boldsymbol{\mu} \tag{4.27}$$

where $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) \in \mathbb{R}^M$ are the chosen basis function for the regression and $\mathcal{S}(\cdot)_k$ is the *softmax* function eq (3.9). Two relations of the same type must be fitted, one for the amplitude, the other for the phase.

Once weights are set properly, this messy expression provides an estimation for the waveform $\tilde{h}$ in (4.4). The fitted expression for $\tilde{h}$ is evaluated at constant mass $M = 20\,\mathrm{M}_\odot$; the dependence on total mass must be included with equations (4.12) and (4.13). The dependence on (d_L, ι, φ_0) is computed with eq. (4.5). With this method, we are able to obtain a complex waveform $h(t; m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$ which reproduces closely a waveform from the training surrogate model.

We report below some remarks about the model.

In principle, $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}})$ can be any continuos function. However, it is practical to choose it within a (much) smaller set. In our work, we will consider only the following functional form for $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}})$:

$$\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}}) = \boldsymbol{\xi}(\log q, s_1, s_2) = \begin{pmatrix} (\log q)^{A_{0,0}} \cdot (s_1)^{A_{0,1}} \cdot (s_2)^{A_{0,2}} \\ \vdots \\ (\log q)^{A_{M-1,0}} \cdot (s_1)^{A_{M-1,1}} \cdot (s_2)^{A_{M-1,2}} \end{pmatrix} \tag{4.28}$$

This means that every basis function is a monomial of the variables $\log q$, s_1 and s_2. A matrix $A \in \mathbf{Mat}(M, 3)$ fully specifies the function $\boldsymbol{\xi}$. We take A_{ij} to have positive integer values. Equation (4.28) has a simple interpretation: we take the expert to be a polynomial of the input variable. This choice ensures that, for a polynomial of high enough order, the expert can model any function with arbitrary precision. It is a good idea to include in the basis function expansion, every monomial with order less than a chosen order $\bar{n}$. This implies that each expert performs a Taylor expansion of order $\bar{n}$.

The choice of working with variable $\log q$ rather than q is for convenience: it was noted that working with logarithms gave much better validation results. Indeed variable q spans at least one order of magnitude and thus values of $\boldsymbol{\xi}(\tilde{\boldsymbol{\vartheta}})$ for different values of q can take very different values. For example, a fourth order monomial spans approximately 4 order of magnitude if q change of a factor of 10. This is an unwanted property, because large (or small) number value within the input are more prone to numerical errors. Using $\log q$ prevents the values of the data features to vary too much within the range of interest, yielding improved results. Of course, this is just a heuristic. We use $\log q$ because it performs better at validation time: this is the real justification for our choice [9].

The waves produced by this model are valid in the range of parameters $\mathcal{P}$ and times in which the dataset was built. The model can generate a waveform also outside the training parameters space, however the waveform is not guaranteed to provide a reliable approximation to the true wave. Outside the fitted time domain, the model cannot give a meaningful estimation for the wave behavior.

As all the waves have the same duration in the grid $\boldsymbol{\tau}$ computed in reduced time $\tau = \frac{t}{M}$, the signal duration depends on the total mass of the system. If the training waves are generated starting from

[9] Actually, almost every choice in our model (and in ML in general) is justified by looking at validation performances of different competing alternatives. This is unavoidable, as we are ignorant about the relation among data and we do not know *a priori* which model to use.

$\tau = -\tau_{min}$, the time to coalescence for a signal from a BBH with M is $t_{coal} = \tau_{min} \cdot M$. Note that $\tau_{min} > 0$ is the time to merger in reduced grid. It is interesting to compute the minimum frequency in the signal as a function of M and τ_{min}. Using eq. (2.74) and writing $M_c = \left[\frac{q}{(1+q)^2}\right]^{\frac{3}{5}} M$, one finds a simple expression for f_{min} as a function of total mass and mass ratio:

$$f_{min} = f_{gw}(t_{min} = \tau_{min} \cdot M) = 151\,\mathrm{Hz} \left(\frac{(1+q)^2}{q}\right)^{\frac{3}{8}} \left(\frac{M_\odot}{M}\right) \left(\frac{1\,\frac{\mathrm{s}}{\mathrm{M}_\odot}}{\tau}\right)^{\frac{3}{8}} \tag{4.29}$$

The expression is approximate because it is obtained within a Newtonian framework and does not consider spin effects. However, it is still an useful order of magnitude estimation for the lowest frequency in the wave. The equation is useful to decide how to set the value of τ_{min}. By looking at a detector sensitivity, one can find out which is the lowest frequency to be measured and set τ_{min} accordingly.

It is important to realise that we could also have worked with waves in frequency domain (FD). At first sight, this choice seems more efficient, as the likelihood term eq. (2.116) in the Bayesian probabilistic model for parameter estimation requires computations in frequency domain. However, this approach hides some issues that makes this framework unfeasible and unadvisable. First of all, surrogate models which work in FD are less accurate than EOB models in time domain. This poses a serious limitation to the theoretical accuracy that the model can reach. Furthermore, performing the total mass scaling to model the M dependence is not trivial. The straightforward relation in eqs. (4.12) and (4.13) becomes messier when expressed in frequency domain and a mass scaling cannot be performed reliably. The total mass dependence should be fitted together with the other quantities and this might result in worse regression performances.

Parameters and hyperparameters of the model Equation (4.27) represents a class of models because many hyperparameters are left unspecified. Before fitting the model, the user must make set them. We list them below.

- D: dimension of the (reduced) time grid.
- τ_{min}: time to merger (in reduced grid) at which the waves are generated
- α: distorsion parameter for the time grid
- (K_{amp}, K_{ph}): number of PCs to be considered in the low dimensional representation of amplitude and phase
- $(\mathbf{L}_{amp}, \mathbf{L}_{ph})$: number of experts to use for the regression of each PC component; $\mathbf{L}_{amp/ph} \in \mathbb{N}^{K_{amp/ph}}$. Setting th same value for every component was seen to perform well.
- (A_{amp}, A_{ph}): matrix for the basis function expansion (as in eq (4.28)). Setting $A_{amp/ph} \in \mathbf{Mat}(M_{amp/ph}, 3)$ sets implicitely the number (M_{amp}, M_{ph}) of independent variables used for regression.

Beside these, there are a number of other hyperparameters that are required for a good fit of the model, such as EOB integration step, steps for gradient descent in softmax regression etc.. Strictly speaking, they are not part of the model and we will not discuss them here. The user can use the default values provided in the model implementation.

The best value of the parameters above must be chosen at validation time by looking at model performances. We do this in section 5.1. The following choice of hyperparameters is found to perform well:

$$D = 2000/5000 \qquad \alpha = 0.3/0.5 \qquad \tau_{min} = 0.4\,\mathrm{s}/\mathrm{M}_\odot{}^{10}$$
$$K_{amp} = 4 \qquad L_{amp} = 4 \text{ (for all PCs)} \qquad M_{amp} = 34 \text{ (every monomial up to 4th order)}$$
$$K_{ph} = 5 \qquad L_{ph} = 4 \text{ (for all PCs)} \qquad M_{ph} = 34 \text{ (every monomial up to 4th order)}$$

It is useful to count the number of hyperparameters for the model. As it is common in ML, this will provide an estimation of model complexity. The model is fully specified by the following quantities:

- $W_{amp/ph} \in \mathbf{Mat}(K_{amp/ph}, D)$: matrix of PCs.
- $\boldsymbol{\mu}_{amp/ph} \in \mathbb{R}^D$: average value of amplitude/phase waveform
- For each MoE model for reconstructing PC with index $i = 1, \ldots, K_{amp/ph}$:
  - $V^{(i)} \in \mathbf{Mat}(M_{amp/ph} + 1, (\mathbf{L}_{amp/ph})_i)$ weights for the gating function (softmax regression) model
  - $\tilde{W}^{(i)} \in \mathbf{Mat}(M_{amp/ph} + 1, (\mathbf{L}_{amp/ph})_i)$ for the expert model.

  The $+1$ in the dimension arises from the constant feature (bias) employed by both the expert and the gating function

The total number of parameters required by the model is

$$N_{tot} = N_{PCA} + N_{MoE} = (K_{amp} + 1) \cdot D + \sum_{i=1}^{K_{amp}} 2 \cdot (M_{amp} + 1) \cdot (\mathbf{L}_{amp})_i + (amp \to ph) \tag{4.30}$$

The first term refers to PCA parameters, while the second term refers to MoE model. The two terms are the same for phase. With the recommended values of hyperparameters we have: $N_{PCA} \simeq 16000/40000$ parameters for the PCA model and $N_{MoE} \simeq 2400$ for the regression model. It is important to realise that N_{PCA} is high because D is high: this is compulsory to obtain a reliable representation of the waveform. However, as it uses a number of parameters which is a small multiple of D, PCA is not a complex model. It is remarkable how small is N_{MoE}. In modern NN architecture, the number of free parameters can be as big as 10^6. The relation to fit is simple so that it can be reproduced in such a small number of parameters.

4.2.5 Gradients of the waveform

A ML model for producing waveform offers a substantial advantage because it provides a closed form expression for the waveforms and of its gradients with respect to the physical parameters. This, in turn, might help the speed up of the parameter estimation, using gradient based sampling techniques. The computation of the gradients are quite involved and we will just sketch them. The released implementation of the model provides a method for the full calculation.

We first need the gradients of the MoE model. We start by computing the jacobian of the softmax function:

$$\frac{\partial \mathcal{S}(V^T\mathbf{x})_k}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\frac{e^{(V^T\mathbf{x})_k}}{\sum_{k'} e^{(V^T\mathbf{x})_{k'}}}\right] = \mathcal{S}(V^T\mathbf{x})_k V_{ik} - \mathcal{S}(V^T\mathbf{x})_k \sum_l \mathcal{S}(V^T\mathbf{x})_l V_{il} \tag{4.31}$$

The gradient of the MoE model $y(\mathbf{x})$ is readily computed:

$$\frac{\partial y(\mathbf{x})}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\sum_k (\mathbf{w}_k^T\mathbf{x} + b_k)\mathcal{S}(V^T\mathbf{x})_k\right] = \sum_k (\mathbf{w}_k^T\mathbf{x} + b_k)\frac{\partial \mathcal{S}(V^T\mathbf{x})_k}{\partial x_i} + \sum_k (\mathbf{w}_k)_i \mathcal{S}(V^T\mathbf{x})_k \tag{4.32}$$

where the jacobian $\partial \mathcal{S}(V^T \mathbf{x})_k / \partial x_i$ of the softmax regression is computed above.

Of course, in any realistic case, one performs a basis function expansion. Thus the objective of differentiation is the composite MoE function $y(\boldsymbol{\xi}(\boldsymbol{\vartheta}))$. The derivative can be trivially computed using the chain rule together with the MoE derivative (4.32) and the jacobian $\frac{\partial \xi_i}{\partial \vartheta_j}$ of the basis function expansion:

$$\frac{\partial y(\boldsymbol{\xi}(\boldsymbol{\vartheta}))}{\partial \vartheta_i} = \sum_j \frac{\partial y}{\partial \xi_j} \frac{\partial \xi_j}{\partial \vartheta_i} \tag{4.33}$$

The gradients above allow us to recover the derivative of the low dimensional representation $\mathbf{g}(\tilde{\boldsymbol{\vartheta}})$ of amplitude/phase with respect to the parameters $\tilde{\boldsymbol{\vartheta}} = (q, s_1, s_2)$. This step is straightforward. The PCA reconstruction does not depend on $\tilde{\boldsymbol{\vartheta}}$ and it is easy to recover the gradient of $\mathbf{f}(\tilde{\boldsymbol{\vartheta}})$ evaluated on the high dimensional grid.

$$\frac{\partial f_i}{\partial \tilde{\vartheta}_j} = \sum_k W_{ki} \frac{\partial g_k}{\partial \tilde{\vartheta}_j} + \mu_i \qquad j = 1, 2, 3 \tag{4.34}$$

We stress that the index i on the vector $\mathbf{f}$ denotes a discretized time dependence. Therefore, the gradients can be evaluated on any time grid by means of interpolation.

Let M_0 be the mass at which the training waveform are given; the PCA model will yield wave A_0 and ϕ_0 with $M = M_0$. The derivative of the strain $h(t; M, \tilde{\boldsymbol{\vartheta}})$ with respect to the total mass can be computed as follow (see eq. (4.12) and (4.13)):

$$\frac{\partial A(t; M, \tilde{\boldsymbol{\vartheta}})}{\partial M} = \frac{\partial}{\partial M}\left[\frac{M}{M_0} A_0\left(t\frac{M}{M_0}\right)\right] = A_0\left(t\frac{M}{M_0}\right) + \frac{t}{M_0}\frac{\partial}{\partial t'} A_0(t')\Big|_{t'=\frac{M}{M_0}\cdot t} \tag{4.35}$$

$$\frac{\partial \phi(t; M, \tilde{\boldsymbol{\vartheta}})}{\partial M} = \frac{\partial}{\partial M}\left[\phi_0\left(t\frac{M}{M_0}\right)\right] = \frac{t}{M_0}\frac{\partial}{\partial t'} \phi_0(t')\Big|_{t'=\frac{M}{M_0}\cdot t} \tag{4.36}$$

The time derivatives $\frac{\partial A_0}{\partial t}$ and $\frac{\partial \phi_0}{\partial t}$ can be performed numerically with finite difference methods.

So far we have computed $\nabla_{\boldsymbol{\vartheta}} A(t; \boldsymbol{\vartheta})$ and $\nabla_{\boldsymbol{\vartheta}} \phi(t; \boldsymbol{\vartheta})$. As usual the time dependence is discretized on a time grid. We now use equation (4.5) to compute the gradients of the full waveform. Let us call $h_R = \mathrm{Re}[h(t; \boldsymbol{\vartheta})] = A(t; \boldsymbol{\vartheta}) \cos\phi(t; \boldsymbol{\vartheta})$ and $h_I = \mathrm{Im}[h(t; \boldsymbol{\vartheta})] = A(t; \boldsymbol{\vartheta}) \sin\phi(t; \boldsymbol{\vartheta})$. We have:

$$\frac{\partial h_R}{\partial \vartheta_i} = \frac{\partial A}{\partial \vartheta_i}\cos\phi - A\sin\phi\frac{\partial \phi}{\partial \vartheta_i} \tag{4.37}$$

$$\frac{\partial h_I}{\partial \vartheta_i} = \frac{\partial A}{\partial \vartheta_i}\sin\phi + A\cos\phi\frac{\partial \phi}{\partial \vartheta_i} \tag{4.38}$$

We are now ready to compute the gradient of the full complex waveform (4.5) $h(t; \boldsymbol{\vartheta}, d_L, \iota, \varphi_0)$ with respect to physical parameters $(q, M, s_1, s_2, d_L, \iota, \varphi_0)$. For convenience, we drop the explicit dependence

of h.

$$\frac{\partial h}{\partial \vartheta_i} = \frac{1\,\mathrm{Mpc}}{d_L} \cdot \left\{ \frac{1+\cos^2\iota}{2} \left[\cos 2\varphi_0 \cdot \frac{\partial h_R}{\partial \vartheta_i} - \sin 2\varphi_0 \cdot \frac{\partial h_I}{\partial \vartheta_i} \right] + i \cdot \cos\iota \left[\sin 2\varphi_0 \cdot \frac{\partial h_R}{\partial \vartheta_i} + \cos 2\varphi_0 \cdot \frac{\partial h_I}{\partial \vartheta_i} \right] \right\} \qquad i = 1,2,3,4 \tag{4.39}$$

$$\frac{\partial h}{\partial d_L} = -\frac{1}{d_L} h \tag{4.40}$$

$$\frac{\partial h}{\partial \iota} = -\cos\iota \sin\iota \cdot \mathrm{Re}[h(t;\boldsymbol{\vartheta}, d_L, \iota=0, \varphi_0)] - i \sin\iota \cdot \mathrm{Im}[h(t;\boldsymbol{\vartheta}, d_L, \iota=0, \varphi_0)] \tag{4.41}$$

$$\frac{\partial h}{\partial \varphi_0} = \frac{1\,\mathrm{Mpc}}{d_L} \cdot \Big\{ -(1+\cos^2\iota)\,[\sin 2\varphi_0 \cdot h_R + \cos 2\varphi_0 \cdot h_I] + 2i \cdot \cos\iota\,[\cos 2\varphi_0 \cdot h_R - \sin 2\varphi_0 \cdot h_I] \Big\} \tag{4.42}$$

This fully specify the waveform gradient.

4.3 Package `mlgw`

The ML model described above is implemented in a `Python` package `mlgw`. It is publicy avaiable at PyPI [11] under the same name. It can be easily downloaded and installed from `https://pypi.org/project/mlgw/` or typing the standard command `pip install mlgw`. In this section, we describe the main features of the package, the criteria behind its implementation and we provide a description of its modules. The discussion here is at high-level and, as it is intended to provide an overview of the package, it does not require pior knowledge of `Python`. An interested user can access detailed documentation of the low-level details with the `help Python` built-in function.

Package `mlgw` provides the code to merge every part of the model described above. It is built up of several *modules*, each of which perform a well defined task. We list them below and summarize their functioning.

`GW_generator`

This is the main core of the package. It provides the class `GW_generator` to generate a wave according to the model so that it can be exploited in real-word application such as parameter estimation. A fitted model is provided and it ready to use. `GW_generator` gathers the different parts of the model. It contains the Mixture of Experts model and the PCA model, used to reproduce relation (4.27). Furthermore, it provides the code to include the known dependence on total mass M, inclination angle ι, reference phase φ_0 and luminosity distance d_L. The latter functionality is implemented within the class. It uses the PCA model provided in `ML_routines` and the MoE model in `EM_MoE`.

A wave can be generated with method `get_WF` or `__call__`. The orbital parameters required to generate the waves can be specified by the user in the form of a `numpy` [77] array with shape (N, D). Each of the N rows refers to a different wave to generate. Each row has D features and must have one of the following layout:

[11] PyPI is a public repository for `Python` projects and it hosts the code by everyone who want to share it. Among the `Python` community, it is standard practice to upload there any open source software. See `https://pypi.org/`.

- (q, s_1, s_2)
- (m_1, m_2, s_1, s_2)
- $(m_1, m_2, s_1, s_2, d_L)$
- $(m_1, m_2, s_1, s_2, d_L, \iota)$
- $(m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$
- $(m_1, m_2, (\mathbf{s}_1)_{x/y/z}, (\mathbf{s}_2)_{x/y/z}, d_L, \iota, \varphi_0, \Omega, \epsilon, \pi_m)$

In the latter line Ω is the longitude of the ascending node, ϵ the orbital eccentricity and π_m is the mean periastron anomaly. The dependence on these quantities, as well as on the transverse spin components, is not implemented. It were included to make the prototype of the function compatible with those provided by `lalsuite`. The user can specify a custom time grid and can choose whether it represents physical or reduced times. Waves are returned in a `np.array` with shape (N, N_{grid}), where N_{grid} is the number of grid points chosen by the user. Two arrays are returned and, at user choice, they hold either amplitude and phase or the waves polarizations. The way data are organized is typical of ML (see e.g. eq (3.2)), where every row of a matrix holds different observation of data. Working with numpy arrays gives access to a number of highly optimized routines for matrix manipulation and provides a significant speed up in wave generation.

Essentially, the generation of a WF is performed in three steps (by function `__get_WF`):

- *Generation of "raw" WF.* In this stage, the waveform in eq. (4.27) is reconstructed. It only depends on (q, s_1, s_2) (function `get_raw_WF`).
- *Interpolation to the user grid.* The wave is interpolated to the user grid. At this stage, the total mass dependence is included with equations (4.12) and (4.13) (interpolation is performed with `np.interp`).
- *Post processing.* If required by the user, the dependence on d_L, ι, φ_0 is computed with eq. (4.5) (function `__set_d_iota_phi_dependence`).

A pre-trained model is released and can be used to promptly generate waveforms. However, the class implements the possibility to use user-created models. The pre-trained model is fitted with a maximum (reduced) time to coalescence $\tau_{min} = 8\,\mathrm{s/M_\odot}$. Orbital parameters (q, s_1, s_2) are allowed to vary in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$. Both amplitude and phase are reconstructed with 4 PCs. MoE models employs 4 experts for the amplitude fit and 4 experts for the phase fit. A full 4-th order polynomial in the three variables is used both for amplitude and phase.

A fitted model is stored in a single folder and it is composed by the following files:

- `amp(ph)_PCA_model` PCA model for amplitude(phase), as output by class `PCA_model`
- `amp(ph)_gat_<PCnumber>` model for the gating function of the `<PCnumber>` for amplitude(phase). `<PCnumber>` holds the number of PC the MoE refers to.
- `amp(ph)_exp_<PCnumber>` weights for the experts of the `<PCnumber>` for amplitude(phase). `<PCnumber>` holds the number of PC the MoE refers to.
- `amp(ph)_feat` features ξ_i to use for the basis function expansion. They are in the same data format of `add_extra_features`.
- `times` points of time grid at which full dimensional waves are generated by the PCA model

There can be an arbitrary number of experts and gating functions as long as their number is less than PCA components. The files above can be generated by the routines provided in module `fit_model`.

As the typical magnitude of a GW is very small, all the ampltidues the model works with are scaled by a factor 10^{-21}. The scaling ensures that they are $O(1)$. Class `GW_generator` performs the inverse scaling to real amplitudes.

Function `get_grads` provides the gradients of the waveform (evaluated on an arbitrary user given grid) with respect to the variables $(m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$.

A call to method `model_summary` gives information about the features of the model in use.

`EM_MoE`

The module holds a Mixture of Experts model (class `MoE_model`). It accepts a generic gating function model. A softmax regression model (class `softmax_regression`) is implemented as standard gating function.

`MoE_model` provides methods for fitting with the EM algorithm and to make predictions. If the default softmax model is not chosen, the user must provide a valid model for gating functions (i.e. must make predictions $\in [0, 1]$ and use cross entropy loss function) which implements methods `fit`, `predict`, `save` and `load`. A dataset $\{X \in \mathbf{Mat}(N, D), y \in \mathbb{R}^N\}$ of N datapoint must be provided at training time. At test time, the model returns predictions $\hat{y} = \mathbb{E}[y|X_{test}] \in \mathbb{R}^N$. It can also output predictions of each expert or values of the gating function. At training time a number of hyperparameters must be chosen by the user.

`softmax_regression` provides a method for fitting and evaluating a softmax regression model. A dataset $\{X \in \mathbf{Mat}(N, D), Y \in \mathbf{Mat}(N, K)\}$ of N datapoint must be provided when training the model. The model is fitted with gradient descent performed with an implementation of Adam algorithm [78]. A wrapper to `scipy.optimize.fmin_bfgs` [79] is also available for minimisation of loss function. However, the latter method, was found to have poorer performance. At test time user can obtain model classifications probabilities $p(y = k|\mathbf{x})$ and can evaluate the log-likelihood of a test set.

`ML_routines`

The module adds a number of useful classes and routines, commonly used in ML. It provides an implementation of a PCA model (class *PCA_model*), a Gaussian Discriminant Analysis classifier (class `GDA`) and routine `add_extra_features` for data augmentation.

Class `PCA_model` provides an implementation of a standard PCA model. It must be trained with a dataset of N real observations $X \in \mathbf{Mat}(N, D)$. The user sets K at training time and, once trained, the module outputs the low-dimensional reduction $H \in \mathbf{Mat}(N, K)$ of any test set. PCA models are saved to a single file, that can be input to class `GW_generator`. An option allows the user to receive low dimensional data scaled to be in range $[-1, 1]$. This preprocessing is crucial for the MoE regression, as it is always convenient to work with $O(1)$ data to avoid numerical errors. The PCA model stores also the eigenvalues of the PC. Although not strictly necessary, this is an useful information provided to a future user.

In order to provide an alternative to softmax, a GDA model is available in class `GDA`. As usual, it has methods for fitting and predicting. Since the model is simple, it is not expected to work well in a difficult classification problem, as that posed by MoE can be. However, its MLE is known in closed form and this makes the training procedure extraordinary fast to run. This is the main reason why an user might want to use it.

A routine for basis function expansion is provided by `add_extra_features`. This routine is used by `GW_generator` to enlarge test data and make them suitable to be given as input to the MoE model:

it is a crucial part of the model. As data augmentation is useful also at training time, we chose to keep this task separate from `GW_generator`. The function accepts as input a dataset and a list of features. It adds only monomial features of the form $x_{i_1} \cdot \ldots \cdot x_{i_m}$ where x_j is the j-th component of the input vector; to denote such feature one must add "$i_1 \ldots i_m$" to the list of features. By default, it replaces $q \to \log q$ in the feature vector before any computation.

`GW_helper`

The module gathers some routines that are useful to deal with GW signal.

It provides a routines `compute_scalar` for computing the scalar product (2.110) between two waves. It is computed in frequency domain and an optional noise PSD can be given. It can work also with waves in time domain as long as a no noise is considered [12]. The routine is useful to computation of mismatch function $\mathcal{F}[a,b] = 1 - \frac{\langle a,b\rangle}{\sqrt{\langle a,a\rangle \langle b,b\rangle}}$ which measure how two waves are different from each other. Some deeper discussion of its meaning will be given in the next chapter. Function `compute_mismatch` provides a tool for such computation.

Function `compute_optimal_mismatch` provides tool for computing the optimal mismatch (see next section) for two waves in time domain. Unlike the above function, the user must input a complex strain h to represent the two waves.

The module provides also methods for generating a dataset of waveforms in time domain. It uses a surrogate model in the EOB familiy, as provided by `lalsuite` sotware suite [34]. This is a software library designed for GW data analysis and provides (among other utilities) an easy-to- use implementations of EOB models. The user must install a version of `lalsuite` before using the routine.

Function `create_dataset_TD` generates a dataset (4.19) in time domain. User is required to input:

- The range of orbital parameters in which random waves must be created
- The time to merger (in reduced grid τ) which the wave begins at
- Number of waves to generate
- Number of grid points
- The distorsion parameter α
- The integration step for the EOB integration model

The output dataset is redirected to file or returned as an `np.array`.

`fit_model`

It provides an useful tool for fitting the whole model. Once a dataset of waves is generated, this is done in two steps. The first step consists in fitting a PCA model and creating a dataset of PCA reduced waveform (routine `create_PCA_dataset`). In the second step, the reduced form dataset is used to perform a number of MoE regression (routine `fit_MoE`).

`create_PCA_dataset` requires as input the path to a dataset of waveform. Furthermore, the user must choose how many PCs shall be used for the dimensional reduction, both for the amplitude and phase dataset. The routine performs a split between training and test set and the user can decide the fraction of training points to use. The routine outputs a number of files where a PCA reduced dataset, as well as the PCA model, are stored both for amplitude and phase. They are:

[12] Indeed, if noise $S_n(f) = 1$, $\langle a,b\rangle = \int \mathrm{d}f\ \tilde{a}^*(f)\tilde{b}(f) = \int \mathrm{d}t\ a^*(t)b(t)$.

- `amp(ph)_PCA_model`: model for PCA for amplitude(phase)
- `PCA_train(test)_theta.dat`: orbital parameters for the reduced training(test) set
- `PCA_train(test)_amp(ph).dat`: reduced representation of training(test) amplitude(phase). Waves are generated with orbital parameters at stored in file above. Amplitude are multiplied by a factor 10^{-21}.
- `times`: times at which the high dimensional waves are saved. This part is not necessary for the MoE regression but it is required by `GW_generator`.

User must specify a folder to store the output. The routines computes the mismatch $\mathcal{F}$ between test waveforms and test waveforms reconstructed by PCA.

Routine `fit_MoE` requires a PCA dataset as input. User must indicate an input folder where it is stored (as output of `create_PCA_dataset`). The routine fits a MoE model for amplitude or phase, depending on user's request. User must specify the PCs to include in the model, the number of experts to use in each regression and the basis functions for basis function expansion. Furthermore, a number of hyperparameters for the fitting procedure must be set: a threshold for the maximum LL increase in the EM algorithm and some parameters for the softmax regression gradient descent.

The routine saves the MoE models to a number of files, gathered in a custom folder. Output files have the same structure as the input for `GW_generator` (i.e. for each regression a file for the gating function and the experts as well as a list of features used in the basis function expansion). The routine copies to the output folder also the PCA models and the time list. Once MoE models for amplitude and phase are fitted, the common output folder gathers a valid model which can be handled by `GW_generator`.

Chapter 5

Results

This chapter presents the results of some tests on the model for signal generation. We first study how its performance depend on the choice of hyperparameters. Furthermore, to make sure that it can be effectively used in real-life application, we assess the model accuracy and its limitations. We perform three kinds of tests:

- *Tests to assess model dependence on hyperparameters.* Such tests are meant to explore how the model depends on the different hyperparameters. Indeed a number of choices (eg. number of PCs to consider, number of training points, number of experts in MoE, features to use for regression etc...) must be done by the user and it is important to base them on numerical experiments. Furthermore, even if no parameter setting is required, it is useful to have an idea of how the model performs. In ML common language, this task is called *validation* of the model. This will be done in section 5.1.

- *Tests to assess model accuracy.* These tests measure the discrepancy between the ML generated waves and the surrogate waves that we seeks to reproduce. They include a computation of test error (mismatch) for different values of orbital parameters. Furthermore a full parameter estimation is performed to check on the performances in a real application. We discuss the results in section 5.2

- *Tests to assess runtime.* We aim to measure the time required to generate a wave with our method. This must be compared with the time taken by EOB models and gives an estimation of the potential gain of our work. See section 5.3 for the results.

Before presenting the results of our investigations, it is important to define the *mismatch* $\mathcal{F}[h_1, h_2]$ between two waves $h_1(t)$ and $h_2(t)$. This is the standard measure of wave "similarity" in GW data

analysis.

$$\mathcal{F}[h_1, h_2] \equiv 1 - \frac{\langle h_1, h_2\rangle}{\sqrt{\langle h_1, h_1\rangle \langle h_2, h_2\rangle}} \tag{5.1}$$

$$= 1 - \frac{4\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{h}_1(f)\tilde{h}_2^*(f)}{S_n(f)}}{\sqrt{\left(4\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{h}_1(f)\tilde{h}_1^*(f)}{S_n(f)}\right)\left(4\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{h}_2(f)\tilde{h}_2^*(f)}{S_n(f)}\right)}}$$

$$= 1 - \frac{\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{A}_1\tilde{A}_2}{S_n} e^{i(\phi_1-\phi_2)}}{\sqrt{\left(\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{A}_1^2}{S_n}\right)\left(\mathrm{Re}\int \mathrm{d}f\ \frac{\tilde{A}_2^2}{S_n}\right)}} \tag{5.2}$$

where $\langle \cdot, \cdot \rangle$ is the standard GW scalar product eq. (2.110) and in the last equality we dropped the f dependence for simplicity.

Equation (5.2) allows for an straightforward interpretation of the mismatch. Mismatch can be written as

$$\mathcal{F} = 1 - (\text{overlap})$$

The overlap measures how much two waves are similar. Two waves agree if their phase difference is small: this the reason for the factor $e^{i(\phi_1-\phi_2)}$ in the numerator, which grows quickly for high phase differences. The disagreement in phase is weighted more if the wave amplitude is high: this is taken into account by the term $\frac{\tilde{A}_1\tilde{A}_2}{S_n}$. Amplitudes are weighted by noise: if the noise level is high at a given frequency, the phase difference is less significant. When two waves agree well in phase but not in amplitude, the overlap will drop together with the term $\int \mathrm{d}f A_1 A_2^*$. Finally, the denominator provides a normalization for the amplitudes, to make the overlap in the range $[0, 1]$. Obviously, when $h_1 = h_2$, $F[h_1, h_2] = 0$; more generally, the lower mismatch the two waves have, the more similar they are. Crucially, a discrepancy in phase has more effect than a discrepancy in amplitudes: this means that a ML model for reproducing waveforms must be more careful when fitting the phase.

In the definition of $\mathcal{F}$, we implied that the waves are aligned so as to minimise the value of $\mathcal{F}$. Indeed the addition of a constant phase does not affect the physics but changes dramatically the mismatch. This means that we compare only waves that are aligned, neglecting any mismatch due to wrong phasing. For this reason, the quantity above is better called *optimal mismatch* and it is defined as:

$$\mathcal{F}_{optimal}[h_1, h_2] = \min_{\phi_0 \in \mathbb{R}} \{\mathcal{F}[h_1, h_2 e^{i\phi_0}]\} \tag{5.3}$$

In what follows, we will imply that the minimisation above is performed, when reporting a mismatch value.

The overlap $\mathcal{O}[h_1, h_2] = \frac{\langle h_1, h_2\rangle}{\sqrt{\langle h_1, h_1\rangle \langle h_2, h_2\rangle}}$ is closely related to the likelihood (2.116) for the parameter estimation Bayesian model. In fact, if s is the detected signal and h is the model for the waveform, we get:

$$\log p(s|\vartheta, H) = \langle h(\vartheta), s\rangle - \frac{1}{2}\langle h(\vartheta), h(\vartheta)\rangle - \frac{1}{2}\langle s, s\rangle \tag{5.4}$$

$$\log \mathcal{O}[s, h] = \log\langle h(\vartheta), s\rangle - \frac{1}{2}\log\langle h(\vartheta), h(\vartheta)\rangle - \frac{1}{2}\log\langle s, s\rangle \tag{5.5}$$

The expressions above are identical a part from the log in each term of lhs of (5.5). More interestingly, whenever $\log \mathcal{O}[s, h]$ is maximised, also $\log p(s|\vartheta, H)$ is maximised. This means that the likelihood term is higher for models with higher overlap (lower mismatch) with data: this is a very natural property to

ask for. Another insight can be obtained by scaling both the signal s and the wave models $h(\vartheta)$ so that their norm is equal to one [1]. Under this condition, we can drop any multiplicative constant in the likelihood to get:

$$p(s|\vartheta, H) \sim e^{\mathcal{O}} \sim e^{-\mathcal{F}} \tag{5.6}$$

which provides a simple statistical interpretation for the overlap and the mismatch.

The threshold on $\mathcal{F}$, below which two waves are deemed similar depends on the application. Usually different surrogate models seek an accuracy with each other below 10^{-2}. This will be the reference value for the assessment of model performances.

5.1 Tests on hyperparameters

In this section, we report validation results to asses the dependence on model performances on hyperparameters. We study separately three parts of the model where the user must choose some parameters: dataset generation, PCA dimensionality reduction and MoE regression.

5.1.1 Setting hyperparameters for dataset generation

Firstly, the creation of a dataset of waveforms requires the choice of the domain $\mathcal{P}$ for the physical parameter, a time τ_{min} (in reduced grid) and the number of waves to generate. The choice of those depends on user's needs and can be chosen at will. The choice of number of grid points N_{grid}, of distortion parameter α and the integration step for EOB can be optimized by looking at test mismatch.

Setting α and N_{grid} We first evaluate the model ability to fully reproduce a wave, as a function of number of grid points N_{grid} and distortion parameter α. Let $\mathbf{f}_{N_{grid},\alpha}$ the wave stored in a dataset where τ_{min} and $\mathcal{P}$ are fixed. We compare it with the output of the EOB model $\mathbf{f}_{EOB}$, generated with a fixed integration step. To compare the two waves, $\mathbf{f}_{N_{grid},\alpha}$ must be evaluated on the dense time grid of EOB; this is done with a linear interpolation. We then vary N_{grid} and α and compute the resulting mismatch $\mathcal{F}[\mathbf{f}_{EOB}, \mathbf{f}_{N_{grid},\alpha}]$. We report the results in figure 5.1.

As expected, we note that, by increasing the number of grid points, the mismatch decreases. Furthermore, using more than $\sim 10^3$ grid points, does not bring any improvement to mismatch. The result is dominated by numerical errors in the interpolations [2]. This is a lower-bound for the performances of the fit: average fit mismatch cannot be better than $\sim 10^{-4}$. We note that a dataset grid with $\alpha \simeq 0.35 - 0.5$ is the most effective choice if one has few grid points to use. With 100 grid points, this means a two order of magnitude improvement in the mismatch, as compared with the equally spaced time grid $\alpha = 1$. This means that we are able to concentrate points wherever the waveform changes quicker. However, for a high number of grid points, the choice of α does not affect much the performances and all the choices are almost equivalent. A good choice for dataset hyperparameters could be: $N_{grid} \simeq 3/4 \cdot 10^3$ and $\alpha \simeq 0.3/0.5$.

It is interesting to study how the mismatch depends on the orbital parameter. The results are shown in figure 5.2, where we report $\mathcal{F}[\mathbf{f}_{EOB}, \mathbf{f}_{\tilde{N}_{grid},\tilde{\alpha}}]$ as a function of the orbital parameters of the waves generated. We chose fixed value for the hyperparameters: $\tilde{N}_{grid} = 2000$ and $\tilde{\alpha} = 0.5$. We note

[1] This seems a bit unnatural from a physical point of view. However, if s and h are regarded as variables in a probability space, the scaling has the natural meaning of a change of variable and must not affect the overall results.

[2] Two interpolations are required to produce the result: the first when creating the waveform for the dataset, the second when getting back the wave in a high dimensional grid.

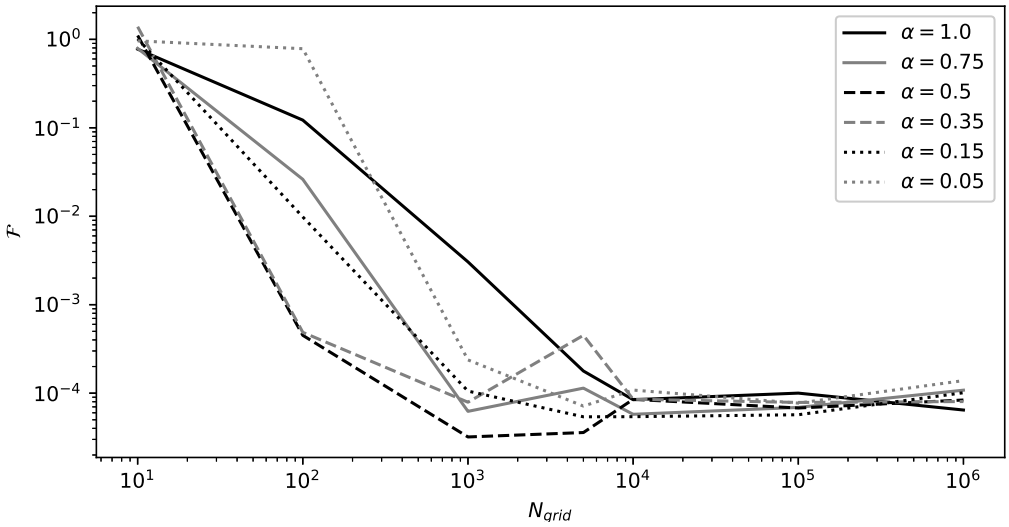

Figure 5.1: Mismatch between waves $\mathbf{f}_{N_{grid},\alpha}$ and raw waves from EOB model. The mismatch is represented as a function of time grid size N_{grid}. Different series are reported for different values of α. The average mismatch is computed on 30 test waves. All waves are generated in $\mathcal{P} = [1, 10] \times [-0.8, 0.65] \times [-0.8, 0.8]$. Waves are generated with $\tau_{min} = 0.2\,\mathrm{s/M_\odot}$ and the integration step is held constant at 10^{-5}. All events have total mass $M = 20\,\mathrm{M_\odot}$. The raw wave from EOB is evaluated at the original time grid provided by EOB: this means that it spans $\sim 4\,\mathrm{s}$ sampled at 10^5 step per second. In the generation domain, high spins were left out because of EOB model inaccuracies (see fig. 5.2).

that at high values for s_1 the interpolation is not able to reconstruct the waveform: the mismatch in these regions can be $O = (1)$. The bad performance is probably due to inaccuracies in the EOB model. The same behavior is not observed for s_2: the reason is that s_2 is less important in the BBH dynamics as it refers to the BH with lower mass (remember that in our convention (4.7) $m_1 > m_2$). The observation will be useful when assessing the range of validity of the model. Due to inaccuracy in the training set, the ML model cannot produce meaningful output at higher values of spins. Indeed this trend will be confirmed in section 5.2 where we will note that model performances at higher values of s_1 will be much poorer than in the rest of the parameter space.

Here we used linear interpolation to reconstruct wave value in a finer time grid. Of course, this is not the best interpolator available. For example, spline interpolation is known to perform better, as it consider more than two points for computing the unknown values. However, non-linear interpolators are known to be slower to run. In the current work, besides a standard linear interpolation, we tried a spline interpolation but no significant improvements were found. For this reason, linear interpolation is used throughout the model: our choice ensures a quick execution without affecting model performance.

Setting EOB integration step As outlined in 2.3.4, the EOB surrogate models solve the PN equation of motion for the relative coordinate of the BBH. As usual in numerical differential equation, one must set an integration step t_{step} which sets the distance between two consecutive points in the discrete grid. Of course, as using any finite time step is an approximation, the smaller t_{step} is set, the better accuracy is achieved. For this reason, one must set the lowest possible value of t_{step}. However, for practical reasons one need not to choose a value which is too small: otherwise the time required to

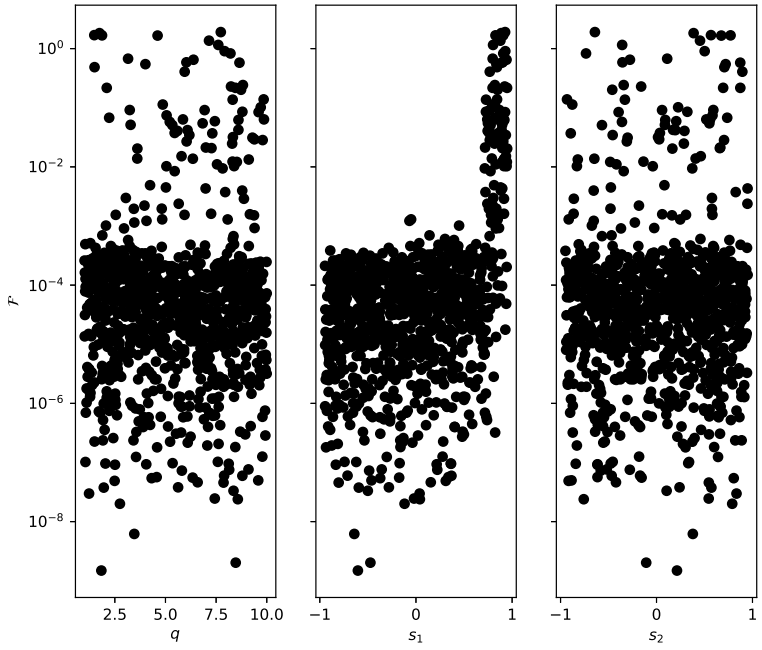

Figure 5.2: Mismatch between 1000 waves $\mathbf{f}_{N_{grid},\alpha}$ and raw waves from EOB model as a function of orbital parameters q (left), s_1 (center) and s_2 (right). We chose optimal values dataset generation hyperparameters: $\tilde{N}_{grid} = 2000$ and $\tilde{\alpha} = 0.5$. All waves are generated in $\mathcal{P} = [1, 10] \times [-0.95, 0.95] \times [-0.95, 0.95]$. Waves are generated with $\tau_{min} = 0.2\,\mathrm{s/M_\odot}$ and the integration step is held constant at 10^{-5}. The EOB implementation is `SEOBNRv2_opt`, provided in `lalsuite`. All events have total mass $M = 20\,\mathrm{M_\odot}$. Note that there are some occasional outliers with very high mismatch. They are gathered at high values positive values of s_1. EOB model, in this region is less reliable. This will affect the ML model performances.

run is too high. Setting $t_{step} = 510^{-6}$ can be a good compromise between speed and accuracy.

The effect of t_{step} on the accuracy can be seen by looking at the "empirical" relation $(\mathbf{g})_i(\tilde{\vartheta})$, where $\mathbf{g}(\tilde{\vartheta})$ is the low dimensional representation of a waveform $\mathbf{f}(q)$ (see eq. (4.20) and figure 4.4). PCA provides a perturbative expansion of each datapoint in a dataset; as the perturbative increase, the reconstruction error decreases and thus the PCA is able to explain the data with an increasing precision. As the dimensional reduction is pushed to higher PCs, numerical errors, due to a finite t_{step}, enters the PCA resolution. This is inevitable as some amount of numerical noise cannot be avoid. Increasing t_{step} increases the noise level of the waveforms: the noise in the function $(\mathbf{g})_i(\tilde{\vartheta})$ will be visible at higher order PCA coefficients. An example of this behavior is shown in figure 5.3.

The relation is precisely what MoE model is meant to reproduce. From the consideration above, we see that MoE (and indeed any regression method) is able to learn a relation, wherever the noise is not too loud. When too much numerical noise arises (as is the case for higher PC in PCA perturbative expansion), the regression is uneffective. For this reason it is important to have numerical noise under control. If it is too high, the fit will stop at lower PCs and thus will generalize the data poorly.

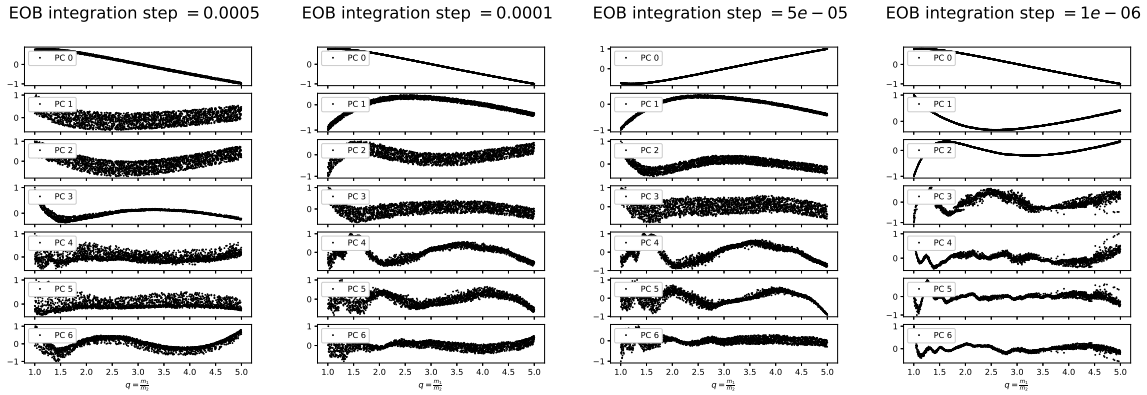

Figure 5.3: We represent here the projection of a waveform on the first 6 PCs as a function of the mass ratio q. Each point represents the projection coefficient on the PC shown in the label. Every waves is generated with $s_1 = -0.3$ and $s_2 = 0.2$. Each columns display a dataset with a different EOB integration time step (the other hyperparameters being fixed). It is manifest that a significant noise level appears at higher order components when the integration step is lowered.

5.1.2 Setting hyperparameters for PCA dimensional reduction

PCA only requires the choice of the number of components to include in the reduction matrix $W \in \mathbf{Mat}(K, D)$. This choice is crucial for the accuracy of the reconstruction of the original waveform. In general one, might want to keep as many PCs as possible to have a reliable reconstruction. However, having a large number of PCs complicates too much the model (i.e. adds many parameters) as compared with the advantages that it brings. Furthermore, from the discussion above, it should be clear that at higher perturbative order, PC projections are very noisy and do not give any useful relation for a regression.

Here we study how the PCA reconstruction accuracy varies with the number of PCs considered. This will be of help to decide how many PCs to choose: once a threshold in accuracy is set, we will use the minimum number of PCs, providing the chosen accuracy. More precisely, our test runs as follows. We generate a waveforms dataset and fit a PCA model with a large number of PCs. We then perform a dimensional reduction and following reconstruction using an increasing number of PCs. We then compute the mismatch between the true wave $\mathbf{h}$ and its reconstruction with k components $\widehat{\mathbf{h}}^{(k)} = W^{(k)\,T} W^{(k)} \mathbf{h}$, where $W^{(k)}$ denotes the matrix of the first k PCs. In figure 5.4, we report an average value for $\mathcal{F}[\widehat{\mathbf{h}}^{(k)}, \mathbf{h}]$ as a function of k. Here we choose the same number of PCs for both phase and amplitude.

Note that the mismatch $\mathcal{F}[\widehat{\mathbf{h}}^{(k)}, \mathbf{h}]$ is different with that computed in 5.1.1. The latter is obtained by interpolating waves on the same grid, while in the first no interpolation is required. Interpolation introduces a big amount of numerical noise and, for this reason, PCA mismatch are much lower than their counterpart in the dataset.

The phase reconstruction of a wave must be more accurate than the amplitude reconstruction. Indeed a small variation in phase affects the mismatch more than a small variation in amplitude. For this reason, in order to fit the amplitude, we need less components than those used for fitting the phase. This observation is shown in figure 5.5 where we reconstruct with PCA only amplitude (phase); the full wave is obtained with reconstructed amplitude (phase) and test phase (amplitude). We obtain two curves with similar behavior and their ratio (approximately constant) is about two order of magnitude high. This means that, at constant number of components, the mismatch introduced by

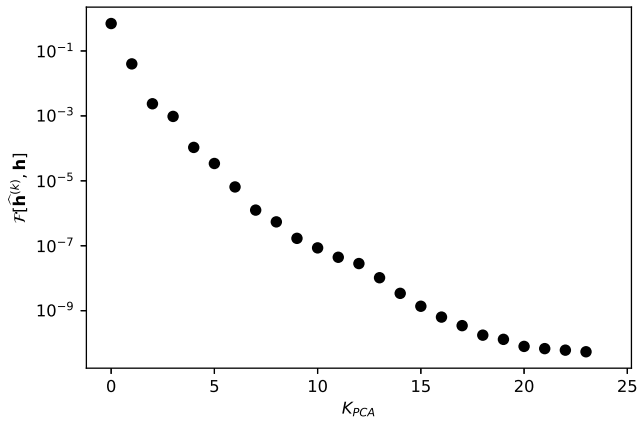

Figure 5.4: We plot, as a function of the number of PC k, the quantity $\mathcal{F}[\widehat{\mathbf{h}}^{(k)}, \mathbf{h}]$, where $\widehat{\mathbf{h}}^{(k)}$ is the PCA reconstruction of $\mathbf{h}$ wave when the first k PCs are used. The mismatch depends strongly on the number of components.

phase is approximately a hundred times higher than that by amplitude: the phase needs more PCs to be reconstructed. The relation can be used to decide the number of PCs to use for each regression. With a mismatch threshold of 10^{-6}, we see that a we should use 4 PCs for the amplitude and 7 for the phase.

5.1.3 Setting hyperparameters for MoE regression

In the Mixture of Experts regression there is a huge number of hyperparameters the user needs to set. They can all affect fit performances and their effects might not be independent. Furthermore, as the EM algorithm strongly depends on a random initialization, the results can vary heavily in different runs of the training: some variation as big as one order of magnitude were observed! For the reasons above, it is virtually impossible to explore deeply the hyperparameters space and to draw a reliable conclusion on an optimal choice of them.

In what follows, we will focus on setting a small number of hyperparameters: the number of experts N_{exp} for each component model, the basis functions $\xi_i(\tilde{\boldsymbol{\vartheta}})$ to use in basis function expansion and the number N_{train} of training points to use. Other parameters, such as the threshold for quitting the EM algorithm, regularization for softmax loss function or the learning rate for softmax gradient descent, will not be considered. We will set plausible values for them, as indicated in the default of the relevant python functions used. This will suffice for our purposes, however it is important to keep in mind that the choices that we will report might not be optimal.

As fitting wave amplitude and phase requires many regression for the different PCs, the mean squared error (mse) of a single regression is not an useful indicator of model performances, unlike the usual case. However, mse still has an immediate and straightforward meaning and cannot be left out. In every numerical experiment, we will consider, together with mismatch of reconstructed function, the mse for the first 2 PCs. This will make our graphs a bit messy, but it will give a larger pool of information.

Choosing expert number and features for basis function expansion The choices of the number of experts and of the features are not independent from each other. In general, one can

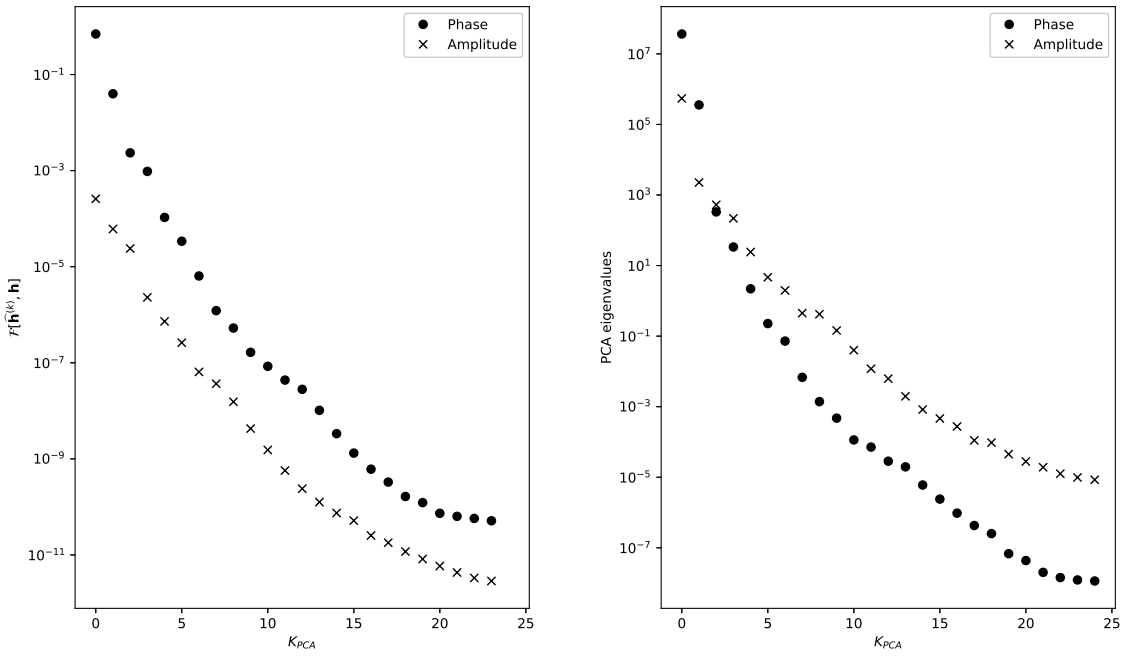

Figure 5.5: On the left panel, we display the mismatch $\mathcal{F}[\widehat{\mathbf{h}}^{(k)}, \mathbf{h}]$ for the reconstruction of amplitude (phase) only. This means that, in order to reconstruct the wave, test phase (amplitude) are used instead of a PCA reconstruction of them. Note that the mismatch depends strongly on the phase and, to achieve a fixed mismatch, one must use more PCs than for the amplitude case. On the right panel, the first k eigenvalues E_k of the dataset covariance matrix are plotted against the index of PC they refer to. We displays values for both amplitude and phase dataset. Both graphs display logarithmic y axis.

reduce the number of experts by incresing the number of basis functions: heuristically, less experts are employed but each of them is more reliable. However, one must pay attention not to overcomplicate the expert model: too many features might not bring significant improvement in the results or, even worse, they might overfit the data. Validation results can help to balance the choice.

In figures 5.6 and 5.7 we present our results. We fitted a model for amplitude (or phase) for different configurations of expert number N_{exp} and polynomial basis function. By label "n-th order", we mean that in the basis function expansion, every monomial up to n-th order is used. We report with a colorbar the value of the mismatch F and of the reconstruction mse of the first 2 PCs. The MoE models for each components share the same number of experts N_{exp}. Test mismatch for amplitude (phase) fit against a test wave is computed by using test phase (amplitude) in the reconstructed wave.

Interpretation of the results is straightforward. Fit performances improves, as the model complexity increases. Indeed, increasing the number of experts and increasing the order of the polynomial always improve the model test performances. This can be seen clearly in the models with $N_{experts} = 1$ and in the models with a third order polynomial. In general, we note that adding more features (i.e. improving expert's flexibility) is more effective than increasing the number of experts: increasing the number of experts of order 1 (linear regression) does not bring any advantage; on the other hand, a single expert with a 6-th order polynomial has great performances. However, model performances

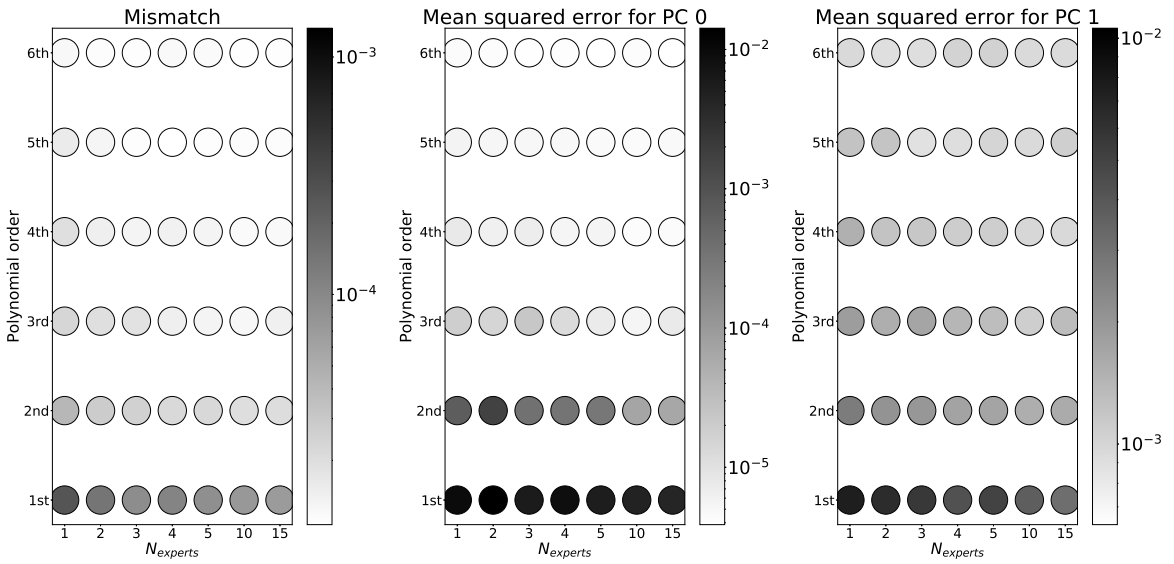

Figure 5.6: Validation results for fit of amplitude. Each point corresponds to a MoE regressions for the amplitude, with a different values of expert number N_{exp} and order of polynomial basis function. The amplitude is represented with 4 PCs. We train the model with 6800 waves in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$ with $\tau_{min} = 0.4\,\mathrm{s/M_\odot}$. In a colorbar, we report mismatch (left), mse for fit reconstruction of the first PC (center) and mse for fit reconstruction of the PC (right). Mismatch is obtained by reconstructing test waves with fitted amplitude and test phase.

do not improve indefinitely. As model complexity increases beyond a threshold, performance does not get better. We can locate the threshold around a model with 4 experts and 4th order polynomial regression. This "simple" model match the performances of much more complex model and thus, for Occam's razor [3], must be deemed as the best choice. We note that all the values for different errors are well correlated: mean squared errors for first is high whenever mse for second PCs is high. The same correlation affects the values of mismatch. In general we note that the reconstruction mse are higher for amplitude rather than for phase. This however have a small impact on the overall mismatch has this is dominated by errors in phase. Performing a regression for amplitude is more difficult but less crucial for good results.

Choosing number of PCs to fit In the previous section, we showed an empirical criterion to decide how many PCs should be used (see left panel of fig. 5.5). We first chose a target mismatch in for reconstructed waves; the number of PCs to be chosen are those that provides the required accuracy. However in practice, due to error in the MoE regression, one cannot reduce the reconstruction mismatch arbitrarily. As previously noted, at high PCA components the relations to fit become very noisy and the regression becomes less accurate: the error introduced by the regression eventually will dominate the mismatch of a low dimensional representation. For this reason, one cannot choose the number of PCs to fit before having seen fit accuracy.

In figure 5.8 we report a numerical study of this. We plot the reconstruction mismatch as a function

[3]Occam's razor is a "philosophical principle" commonly applied in ML when performing model selection. It states that, when considering competing model with comparable performances, the simplest model (i.e. with the lowest number of parameters) has to be chosen.

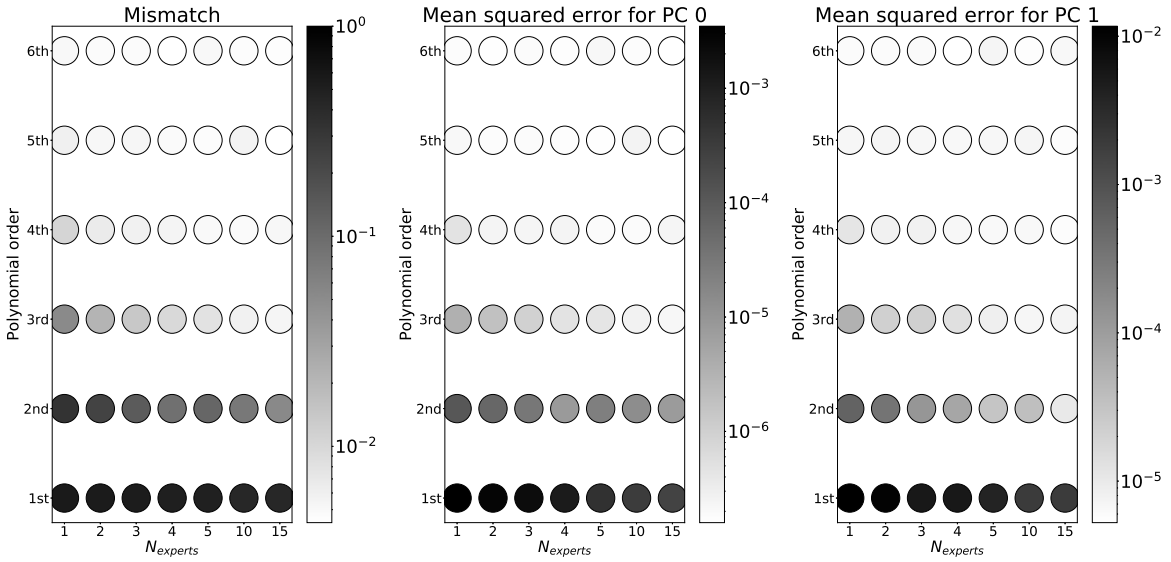

Figure 5.7: Validation results for fit of phase. Each point corresponds to a MoE regressions for the phase, with a different values of expert number N_{exp} and order of polynomial basis function. The phase is represented with 7 PCs. We train the model with 6800 waves in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$ with $\tau_{min} = 0.4\,\mathrm{s/M}_\odot$. In a colorbar, we report mismatch (left), mse for fit reconstruction of the first PC (center) and mse for fit reconstruction of the PC (right). Mismatch is obtained by reconstructing test waves with fitted phase and test amplitude.

of the number of PCs considered. We consider only regression for phase. In one series, we reconstruct the wave using true values of PCs: thus the mismatch is a measure of PCA accuracy. In the other series, we reconstruct a wave using values for PCs as guessed by MoE regression: this is a measure of accuracy of both PCA and regression. It is manifest that for the first two PCs, the regression is accurate enough for reproducing the PCA accuracy: the mismatch introduced by fit residual is negligible. Fitting PCs beyond the 3rd order do not bring any improvement in the reconstructed wave: as noise level is too high, the regression is not able to guess low order coefficients. For this reason, it is useless to include in the PCA model for phase more than 4 PCs: the MoE regression is not able to reconstruct them.

The observation provides a practical way of choosing the number of PCs. We include every PC which yields improvement in MoE mismatch. Of course, this strongly depends on the regression model: the more precise the model is, the more PCs can be included. However, any model cannot increase its accuracy indefinitely. Every surrogate model has an intrinsic noise level, due to numerical error and (mostly) to approximation in the physical model. This sets a lower bound for every improvement in regression performances.

Choosing number of training points The choice of the number of training points N_{train} must trade between accuracy and speed of execution. Too many training points will make the training slow and unfeasible, while too few training points will yield a poor model, which does not generalize data (underfitting). Furthermore, the number of training points depends on the number of model parameters: the more parameters are to fit, the more training points should be used. A general trend is common to many ML models. Increasing the number of training parameters will improve the model

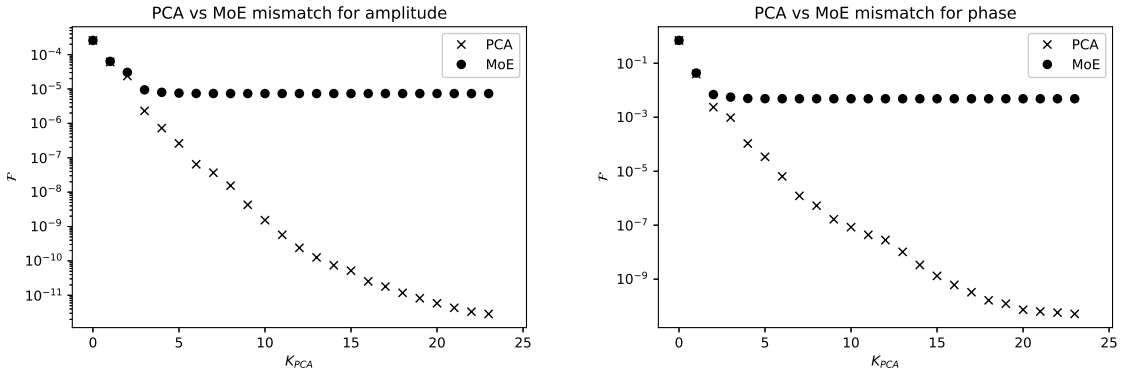

Figure 5.8: Reconstruction mismatch as a function of the number of PCs used in low dimensional representation. The value of PCs are the true ones (label "PCA") and those obtained with a MoE regression (label "MoE"). Data refers to amplitude (left panel) and phase (right panel). MoE model is chosen to be the optimal one (see above) with 4 experts and a fourth order polynomial. It is clear that fitting with a MoE relation beyond the 4th PCA component does not give any improvement to the MoE mismatch, both for amplitude and phase: the noise in the relation of high order PCs is too high for a regression to be performed.

performances, up to a certain threshold. Increasing training points beyond the threshold will not improve model performances but will slow down the training. The optimal value of training points should be slightly higher than the observed threshold. In the choice of number of training points, it is useful to monitor also the training error. The comparison between train and test error will provide important information on how the model is able to generalize the trend.

In figure 5.9 we report train and test value of mismatch and mse of first 3 PCs as a function of the number of training points. Data refers to a MoE model fitted for 7 PCs of the phase dataset. The models has $N_{exp} = 4$ and performs basis function expansion with a fourth degree polynomial. Test mismatch are obtained using test amplitude to reconstruct the waveform; this is not a great limitation as any error in phase reconstruction dominates the overall mismatch (see e.g. 5.5).

The trend is as expected: a steady decrease of erorrs, as N_{train} increases until a plateau is reached. We note that overfitting is not a problem. For a reasonably high number of training points ($N_{train} \gtrapprox 50$) train and test error are always close to each other. The model generalize test data as accurately as for the training points.

In the present model, setting $N_{train} \simeq 2000$ seems a good choice. As compared with standard neural networks, which routinely employ $O(10^5)$ points datasets, this is an incredibly low amount of data. This is due to the fact that MoE is a simple model with a few number of parameters: few data are enough for learning a reliable relation. This is a great strength of MoE.

5.2 Model accuracy

In this section, we present an assessment of model accuracy. Using the results of the previous section, we train a model with the best validation parameters and we evaluate its performance on a test set.

In section 5.2.1, we report test mismatch as a function of the source orbital parameters. This is extremely useful to have an idea of the overall accuracy of the model and to see how it changes for different parameters. It also gives important indications for the choice of fitting domain and the range in which we expect the model to be reliable.

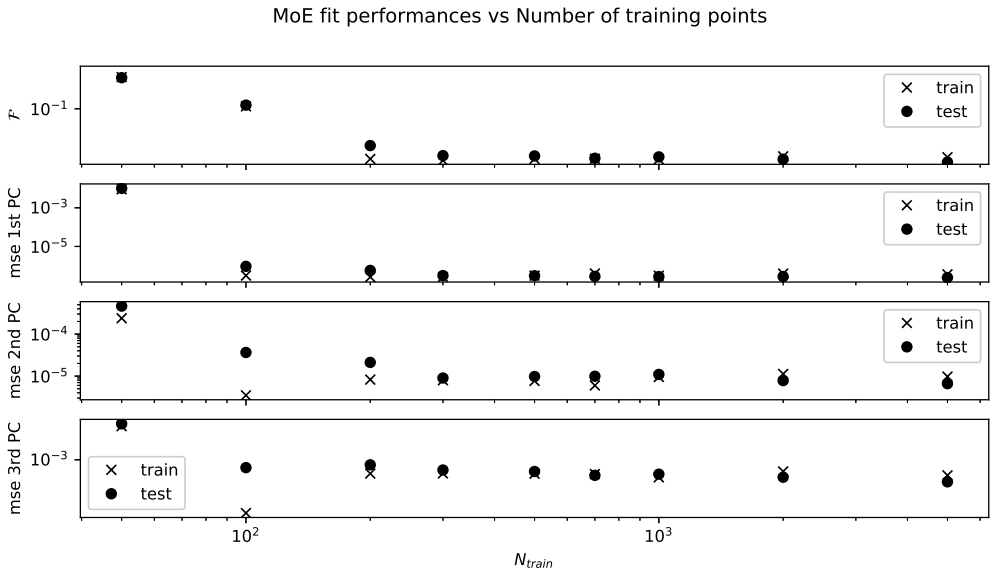

Figure 5.9: Train and test error for MoE fit of 7 PCs of phase, as a function of the number of training points. We report train and test reconstruction mismatch (top) and mse for the first 3 PCs (below). MoE model employes 4 experts and a fourth order polynomial for a basis function expansion. Test error is computed on 1000 test waves. Train and test error decrease steadily when increasing number of training points. For $N_{train} \gtrapprox 800$, the trend stabilises and increasing training points does not affect much model performance.

In section 5.2.2, we perform a full parameter estimation (see section 2.4.3 for the theory) using our model. This is a crucial numerical experiment both for model stability and accuracy. Indeed, for a full PE, a huge number of waves must be generated: implementation weakness of the model can be easily discovered. Furthermore, the test employs the model in a real application scenario and it is the best way to explore whether the model can be effectively used for a reliable physical prediction.

5.2.1 Computing mismatch for test waves

We now measure the accuracy of the best validated model. First of all we compute mismatch value on a number of randomly generated waves. For each wave we extract a random values of the 7 relevant quantities in (4.3) (i.e. $m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0$). This tests the model in the most general case. We report our results in the histogram in figure 5.10. The histogram is built with 3966 waveforms. We report also mean value, standard deviation and median. In the inset the cumulative distribution is plotted. It give us a interesting overall estimation of model accuracy. We report the median value of the mismatch distribution $\mathcal{F}_m = 2 \cdot 10^{-3}$. Furthermore, the 89% of the datapoints lie have a mismatch below 10^{-2}. Such results are similar to the discrepancies between state-of-the-art models.

To understand better model performances, it is interesting to display the accuracy as a function of the orbital parameters $\boldsymbol{\vartheta} = (q, M, s_1, s_2)$. Here we included only the source parameters, leaving out the geometrical parameters. The study allows us to obtain the range of applicability of the model, as well as its limitations. We measure test mismatch $\mathcal{F}$ and mse on reconstruction of the first PC for the phase. Reporting the latter is useful to test the accuracy of the fit before wave reconstruction. Indeed, reconstructing a wave might produce numerical errors which lower the mismatch, especially when interpolating to the user grid (see picture 5.2 for an example). Thus, it is interesting to assess whether,

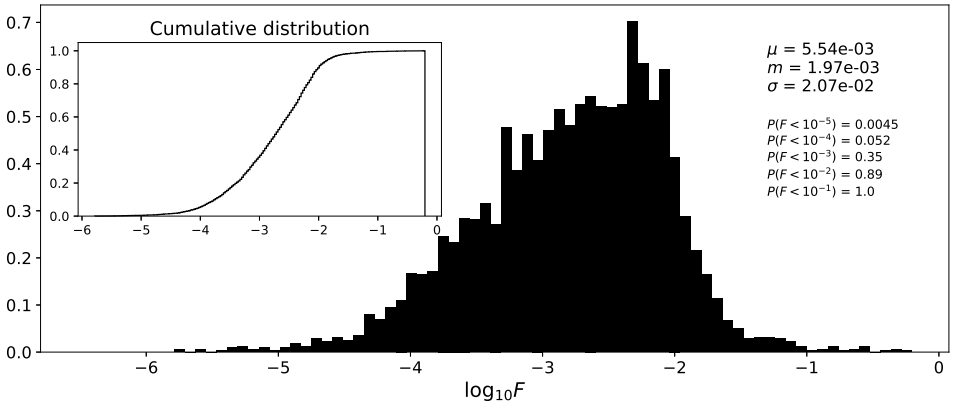

Figure 5.10: Histogram for the logarithm of mismatch values, computed on $N = 3966$ test waveforms. Each WF is generated with random orbital parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \phi_0)$ and with $\tau_{min} = 0.2\,\mathrm{s/M_\odot}$ (you can do better...). In the inset we represent the cumulative distribution of $\log_{10} F$. We report the mean value μ, the standard deviation σ and the median value m as well as the cumulative distribution function at some selected points. Parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \phi_0)$ are randomly chosen in the set $\tilde{\mathcal{P}} = [5., 50.] \times [5., 50.] \times [-0.8, 0.8] \times [-0.8, 0.8] \times [.5, 10.] \times [0, 2\pi) \times [0, \pi)$.

a high mismatch is caused by regression inaccuracies or by an unreliable reconstruction procedure.

We proceed as follows. We set a four dimensional, equally spaced, grid in the space of orbital parameters $\vartheta = (q, M, s_1, s_2)$. At each point of the grid we generate a wave with both mlgw model and EOB method (in the `SEOBNRv2_opt` implementation provided by `lalsuite`); each wave is generated for 8 s before merger. We then compute mismatch $\mathcal{F}$ between waves and the mse on the reconstruction of the first PCs. The results are reported in the contour plots in figure 5.11. In several two dimensional graphs, we report error dependence on selected pairs of variables; the error at each point is obtained with an average on the other quantities. The model is fitted in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$ and with $\tau_{min} = 0.8\,\mathrm{s/M_\odot}$; the hyperparameters of the model were set to be optimal, according the results of previous sections (see caption in 5.11 for more details).

By looking at the top line of 5.11 (mismatch dependence on q and M), we note that the mismatch depends mildly on M. This was to be expected because the total mass dependence is not fitted but it is inserted analytically within the model. Indeed, the mean squared error does not depend on the total mass. We observe however small variation on the mismatch, even at constant M: at lower M values, the mismatch seems to be higher. This can be explained by considering the length of the WFs generated. Each wave lasts approximately 8 s; depending on total mass, each wave has a different frequency at $t = -8\,\mathrm{s}$ (see eq. (2.74)). Waveform with lower M have a higher frequency which makes a bigger value for the phase of the wave eq. (2.75). Thus, a wave with low M has more cycles to accumulate mismatch and this explains the higher values of measured $\mathcal{F}$.

Again on the top line, we note that both $\mathcal{F}$ and mse depend (quite strongly) on q. In general mismatch and mse are well correlated (a part for the mild dependence on M explained above). This means that the reconstruction of the wave and the interpolation does not affect much the mismatch. We note that for low values of q the fit shows poor performance. This is probably due to the failure of MoE model to catch the relevant trend. However, the mismatch is still around 10^{-2}, which still allows for a reliable waveform generation. For $q \gtrsim 10$ we test the ability of the model to extrapolate outside the fitting domain. Of course, the reconstruction mse increases but it is low enough that the model

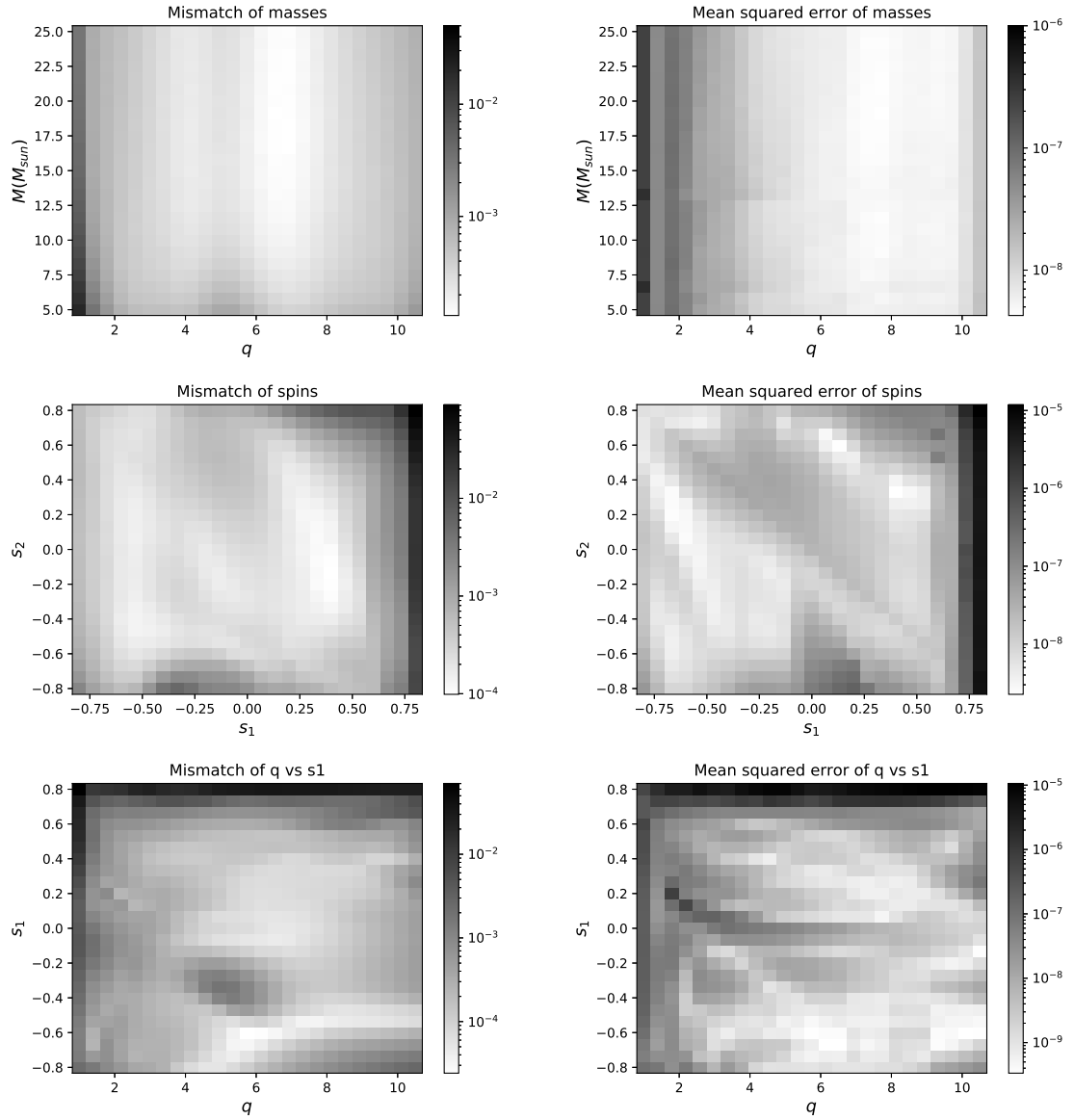

Figure 5.11: We report test mismatch and mse for the first PC of the phase, as a function of the orbital parameters $\vartheta = (q, M, s_1, s_2)$. In the left column, the color mesh refers to test mismatch; in the right column, we report mse for the first PC. On the x-y we display values of two physical quantities. The error reported on the grid is averaged on the two quantities not displayed in the axis. On top row, we display q vs M dependence. On central row, we present spins dependence (s_1 vs s_2). On bottom row, we show q vs s_1 dependence. Each wave starts 8 s before merger. Model is fitted in the domain $\mathcal{P} = [1, 10] \times [-0.8, 0.8] \times [-0.8, 0.8]$ and with $\tau_{min} = 0.8\,\mathrm{s/M_\odot}$; the hyperparameters of the model were set to be optimal, according the results of previous section: we set $t_{step} = 5 \cdot 10^{-6}$s, $\alpha = 0.5$ and $N_{grid} = 3500$. MoE models use a fourth order polynomial with 4 experts both for amplitude and phase. Both phase and amplitude are reconstructed with 4 PCs. Each of the four physical quantitities is sampled on a grid of 25 points. This means that in total $25^4 = 390625$ waves are generated and each point in the two dimensional grid is obtained by averaging on $25^2 = 625$ waves. In the averages, grid points at $s_1 \gtrsim 0.70$ are not considered, as the model is unreliable in that region. Points at $q > 10$ are not considered as well, because they are outside the fitting domain. The overall average mismatch is $\bar{\mathcal{F}} = 1.9 \cdot 10^{-3}$. When excluding unreliable regions, $\bar{\mathcal{F}} = 6.4 \cdot 10^{-4}$.

still provides good results.

In the center line of 5.11, we displayed the dependence on spins. The most striking feature is that the model is completely inadequate for the WF generation, when $s_1 \gtrsim 65$: both $\mathcal{F}$ and mse are too high for our purposes. As mse and mismatch are very well correlated, the problem is due to poor performances of the fit. However, the physics of coalescence at high spins is not completely understood and also the training model EOB does not provide well behaved results. We already showed an example of this in figure 5.2, where for high values of s_1, the EOB waveforms are not consistent with themselves when reconstructed from a smaller grid. The inaccuracies in EOB make a noisy (or even discontinuous) relation to fit. For this reson, it is probable that no model can perform (much) better than mlgw, in the high s_1 region. The same behavior is not observed for high value of s_2. This is expected on physical basis: as s_2 belong to the lower mass BH, it is less important in determining the dynamics of the system.

By looking at mse dependence on s_1 and s_2, it is interesting to note a strong correlation around the main diagonal. This is readily explained [20] by looking at the PN waveform in eq. (2.82). It turns out that the waveform depends (on first approximation) on the *effective spin parameter* $\chi_{eff} = \frac{1}{2}(1+\sqrt{1-4\eta})s_1 + \frac{1}{2}(1-\sqrt{1-4\eta})s_2$ (see eq. (2.84)). As also the mse depends on χ_{eff}, we can explain the observed correlation: wherever χ_{eff} is constant in the $s_1 - s_2$ plane, the mse shows very little variation. The fact the correlation appears is a further verification that mlgw is able to correctly model the waveform, catching the relevant physics. The same structure is recovered in the parameter estimation (see the marginalized posterior $p(s_1, s_2|\mathcal{D})$ in figure 5.12).

In the third row of 5.11, we displayed the dependence on q and s_1, the variables on which the error depends more. We note again the bad behavior of the model at low q and high s_1, both in terms of mismatch and mse. Furthermore, we see that around $q \simeq 5$ and $s_1 \simeq -0.3$ the model worsen its performances (although still in an acceptable level). The same behavior is noted in the mismatch of the other panels. The same clear pattern is not observed in the mse: the problem should be in wave reconstruction. We do not know yet the reason of such problem. However, we stress that the mismatch (being smaller that 10^{-2}) still allows for a reliable wave reconstruction.

5.2.2 Performing a Parameter Estimation with mlgw

A full Parameter Estimation (PE) is crucial test for the model. Indeed, we are able to assess whether the model provides reliable results in a real case scenario and if it is able to match in accuracy the state-of-the-art. Furthermore, as a huge number of WFs are generated, a PE provides a unique stress test for model reliability and stability.

We runned a PE on the signal GW150914 [1], detected by the LIGO interferometers on 14th September 2015. We performed a sampling from the posterior of the signal with Nested Sampling algorithm. We generated 28680 samples. Besides a set of sampled points, the algorithm provides an estimation of the Bayes factor for the model. The Bayes factor $\mathcal{B} = \frac{Z_s}{Z_n}$ is the ratio between the evidence of the model Z_s, where data are assumed to be a superposition of GW signal and noise, and the evidence Z_n of the noise model, where data are assumed to be composed only by noise. The Bayes factor measures in a Bayesian framework the probability that the WF model explain the data. If it is not high enough, the posterior is not meaningful, because we do not believe that the model is able to explain the data. We plot some (marginalized) posterior distribution functions (PDF) in figures 5.12, 5.13 and 5.14. The results shall be compared with those by the LIGO-Virgo collaboration in [40].

In table 5.1, we report our physical predictions for the source parameters and a number of other

	LIGO-Virgo	mlgw
Total mass $\mathcal{M}_c(M_\odot)$	$70.3^{+5.3}_{-4.8}$	$71.4^{+5.2}_{-5.1}$
Chirp mass $\mathcal{M}_c(M_\odot)$	$30.2^{+2.5}_{-1.9}$	$31.0^{+2.3}_{-2.4}$
Primary mass $m_1(M_\odot)$	$39.4^{+5.5}_{-4.9}$	$37.8^{+2.0}_{-3.0}$
Secondary mass $m_2(M_\odot)$	$30.9^{+4.8}_{-4.4}$	$34.1^{+3.5}_{-4.7}$
Inverse mass ratio $\bar{q} = \frac{1}{q} = \frac{m_2}{m_1}$	$0.79^{+0.18}_{-0.19}$	$0.91^{+0.08}_{-0.15}$
Effective inspiral spin parameter s_1	$-0.09^{+0.19}_{-0.17}$	$-0.02^{+0.15}_{-0.21}$
(dimensionless) spin s_1	$0.32^{+0.45}_{-0.28}$	$0.00^{+0.43}_{-0.57}$
(dimensionless) spin s_2	$0.57^{+0.40}_{-0.51}$	$-0.03^{+0.57}_{-0.54}$
Luminosity distance $d_L(Mpc)$	390^{+170}_{-180}	405^{+120}_{-179}
Final mass $M_f(M_\odot)$	$67.1^{+4.6}_{-4.4}$	$68.0^{+4.6}_{-4.5}$
Final spin $s_L(M_\odot)$	$0.67^{+0.06}_{-0.08}$	$0.68^{+0.05}_{-0.08}$
Log Bayes factor $\log \mathcal{B} = \log \frac{Z_s}{Z_n}$	$288.7^{+0.2}_{-0.2}$	$302.0^{+0.2}_{-0.2}$

Table 5.1: Several physical quantities extracted from signal GW150914 by sampling the posterior distribution with a nested sampling algorithm. Results of mlgw model are compared by those published by the LIGO-Virgo collaboration (using EOB model) [40]. We report the measures (i.e. the median value of the marginalized posterior) and their 90% confidence interval. The final BH mass and spins are obtained with the formula from [80].

quantities and compare them with the published LIGO-Virgo results, obtained with the SEOBNR model.

We note that the predictions of mlgw are consistent with those of the LIGO-Virgo collaboration. The mass related parameters (i.e. chirp mass, total mass and individual masses) are very close to their reference value. The same happens for luminosity distance and for the final BH mass and spins (the latter obtained with a phenomenological fit [80] based on NR simulations).

The predicted values for the spins are rather different from the LIGO-Virgo reference. However, the uncertainties on spins are rather high (in both cases) and this makes the mlgw values compatible with the reference values (see fig. 5.12). Indeed, as already noted in section 2.3.2, the measurement of individual spins are difficult due to spin degeneracy of the waveform. Only the effective spin parameter χ_{eff} can be reliably measured because the waveform depends primarily on it. Indeed, the two measures of χ_{eff} agree well with each other and the discrepancy in the spin measures is explained by the difficulty in the measure rather than in a model inaccuracy. The same feature can be observed on the probability distribution $p(s_1, s_2|\mathcal{D})$ in figure 5.12. We note a strong anticorrelation on spins, an ellipse-like 2D distribution function. The slope of the major axis gives a measure of χ_{eff}, while the length of the minor axis roughly estimates its confidence interval. As the ellipse is quite "narrow", a reliable measure of χ_{eff} is possible. However, this constrain poorly the spins. The same correlation is observed on the contour plots for the mismatch (see fig. 5.11) and is already known in literature [20], [21] and [22]. The fact that we observe such correlation is a further confirmation of mlgw accuracy.

It is interesting to explain the strong correlation between M and χ_{eff} that we observe in figure 5.13. Again, the explanation lies in eq. (2.82). We already observed that the two spins enters together the formula with the effective spin paramter χ_{eff} at next-to-next-to-leading order. At that order, the

waveform depends approximately on quantity $\frac{1}{M}\left(16\pi - \frac{11}{3}\chi_{eff}\right)$. It is then explained the observed correlation between χ_{eff} and M. Unlike the two spin case, here we are able to provide a fairly accurate measurement of both quantities: the total mass appears also at next-to-leading-order and this allow for disentangle the correlation and provide an accurate estimation of the two values.

The posterior for sky location (variables α and δ) as well as for the luminosity distance and inclination ι are not really relevant for assessing model accuracy because the waveform dependence on them is inserted analytically. However, for completeness we report them in fig. 5.14. We see that they agree quite well with the results published by the LIGO-Virgo collaboration, both qualitatively (the two shapes are similar to each other) and quantitatively. The reader can refer the paper for comparison.

Finally, we see that the two Bayes factors are similar. The evidence for the two models are comparable: the Bayesian analysis assigns to both a similar probability that the GW150914 is actually described by the model. However, the evidence for mlgw appears to be slightly higher. This is probably due to a prior that covers a smaller region than that in the LIGO-Virgo analysis. The posterior is then forced to stay in a smaller region, putting more probability mass on the high probability region and resulting in higher value for the evidence.

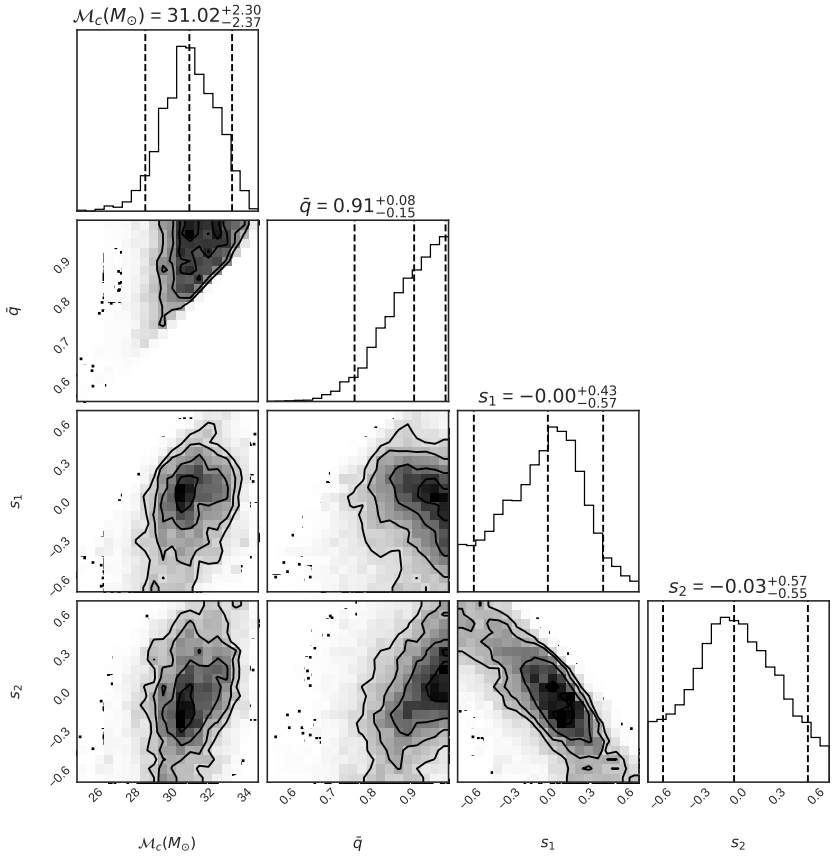

Figure 5.12: Marginalized posteriors for the chirp mass $\mathcal{M}_c$, inverse mass ration $\bar{q} = \frac{1}{q}$ and spins s_1, s_2. We report the histograms for the one dimensional posteriors of each quantity; the 90% credible interval is reported with dashed line as well as the median value. In the contour plots, we report the two dimensional posteriors for each pair of quantities. For each quantity, we report the mean values published by the LIGO-Virgo collaboration: they always lie within the considered confidence interval. Figure generated with Python package `corner` [81].

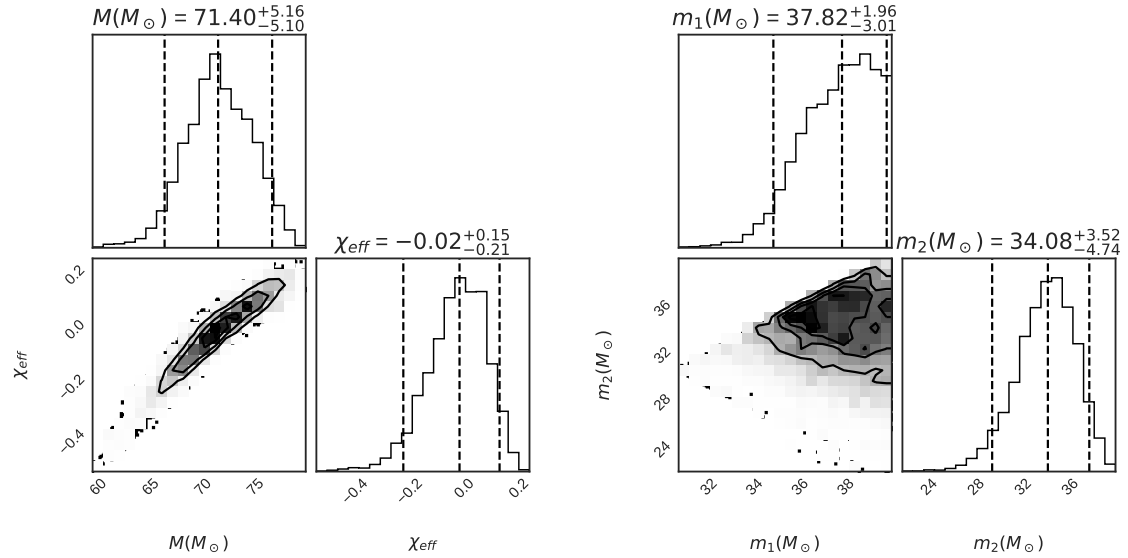

Figure 5.13: Marginalized posteriors for total mass M and effective spin parameter χ_{eff} (left) and for the two BH masses m_1 and m_2 (right). We report the histograms for the one dimensional posteriors of each quantity; the 90% credible interval is reported with dashed lines as well as the median value. In the contour plots, we report the two dimensional posteriors the pair of quantities. Figure generated with Python package `corner` [81].

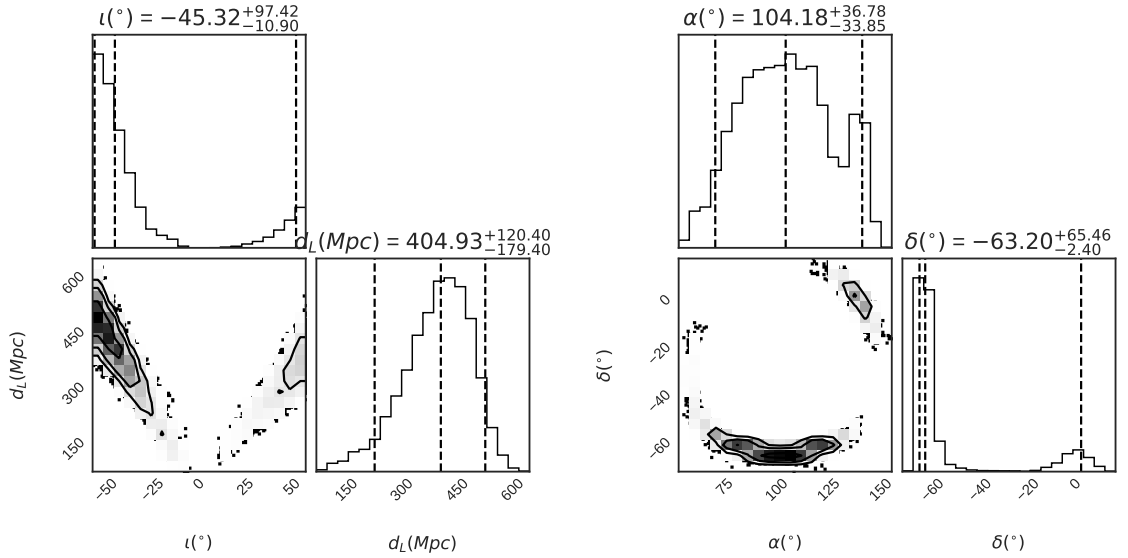

Figure 5.14: Marginalized posteriors for luminosity distance d_L and inclination ι (left) and for the sky location, parametrized by right ascension α and declination δ (right). We report the histograms for the one dimensional posteriors of each quantity; the 90% credible interval is reported with dashed lines as well as the median value. In the contour plots, we report the two dimensional posteriors for each pair of quantities. See figures 2 and 4 in [40] for a comparison. Figure generated with Python package `corner` [81].

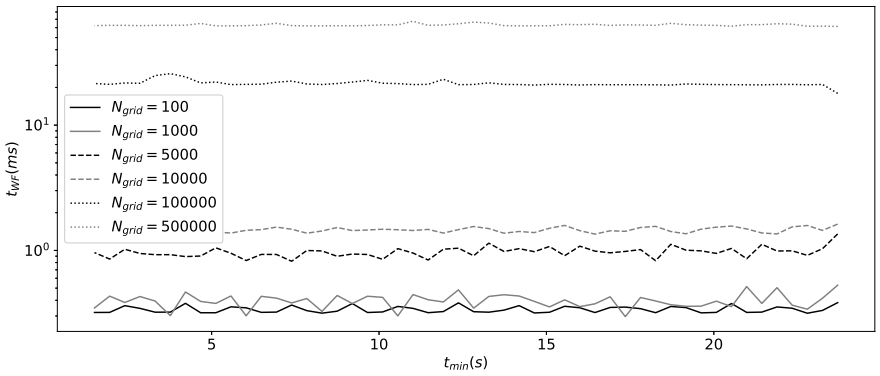

Figure 5.15: Time t_{WF} to generate a single waveform as a function of the time to merger t_{min} which the wave starts from. Waves are computed for different values of N_{grid}, reported on different series on the graph. We generate 100 waves with random parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$ and constant total mass $M = 30\,\mathrm{M}_\odot$ and we compute t_{WF} as the total runtime divided for the number of WFs generated. It is manifest that t_{WF} depends strongly on N_{grid} but does not depend on t_{min}

5.3 Runtime analysis

We now measure the time performances of our model. In section 5.3.1, we consider the time required to generate a waveforms with our model and its dependence on input parameters given by the user. In section 5.3.2, we compare the performance of mlgw with those of EOB.

5.3.1 Time to generate a WF

We now measure the time t_{WF} required to generate a single WF with our model. Essentially, t_{WF} depends only on N_{grid}, the number of points on the time grid chosen by the user, and not by the length of the time grid chosen. Indeed, as explained in section 4.3, the wave generation is performed in three stages: "raw WF generation", "interpolation" and "post-processing". The first stage outputs a WF in the train grid and thus takes a constant time to run. The others deals with WFs represented in the user grid. Their runtime does not depend on the grid extrema but on the grid size N_{grid}.

This is shown in figure]5.15. We generated 100 WFs (in parallel execution, see below) with random orbital parameters and constant total mass $M = 30\,\mathrm{M}_\odot$ [4]. Waves in each series start from different times t_{min} from merger but constant N_{grid}. We repeated the experiment for different values of N_{grid}, reported in different series. It is manifest that the runtime does not depend on t_{min} but depends strongly on N_{grid}. This is an important difference between our code and EOB surrogate models. As the latter integrate a differential equation, the runtime strongly depends on the t_{min}. Our model can be much effective to generate longer WFs.

To investigate further, we measure the runtime t_{WF} as a function of N_{grid}. We report the results in figure 5.16. We generate 100 waveforms with random orbital parameters and constant total mass $M = 30\,\mathrm{M}_\odot$; t_{WF} is obtained by an average. The waveforms are generated on different time grids with same extrema but different number of grid points. We report two series. One, denoted by "parallel", refers to waveforms generated in the a single call of `get_WF`: the use of numpy array allows for a

[4] A total constant mass is required to ensure that all waves fit in the same time grid. This limitation does not affect the runtime.

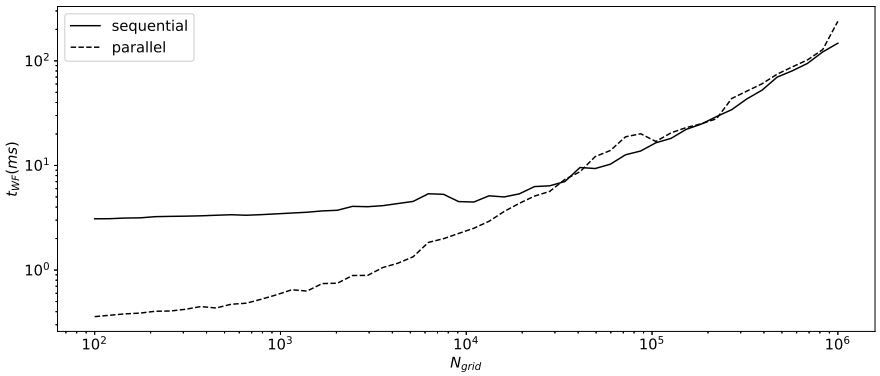

Figure 5.16: Time t_{WF} to generate a single waveform as a function of the number of grid points N_{grid}. Each wave starts at $t_{min} = 21\,\mathrm{s}$ before merger. 100 waves with random parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$ and constant total mass $M = 30\,\mathrm{M}_\odot$ are generated in parallel (label "parallel") and sequentially (label "sequential"). t_{WF} is computed as the total runtime for each method divided for the number of WFs generated. For high values of N_{grid} the two methods have similar performance. For lower values, the sequential execution shows a constant trend which is due to the constant cost of generating the raw waveform with ML model. The parallel execution is faster, as it is able to generate the raw WF quicker.

multicore implementation of matrix operations. In the other series, denoted by "sequential", the waveforms are generated one by one and the optimization is no longer available.

As expected, the time to generate a single waveform, grows dramatically with the number of grid points, approximately following a power law. For large N_{grid}, there is no difference between "parallel" and "sequential" execution: the cost of dealing with a large number of points cannot be parallelized effectively. For smaller number of N_{grid}, the "sequential" execution shows a plateau, where t_{WF} does not depend on N_{grid}. In this region, the "raw WF" generation dominates and the grid size does not affect much the performances. On the other hand, the parallel execution does not show such trend. The raw waveform generation is parallelized and thus has less impact on model performances. The value of t_{WF} is dominated by the cost of interpolation and post-processing, which yields the same relation observed for high N_{grid}.

We conclude that the parallel execution is useful for a speed up in the raw WF generation and it is less effective in the other two tasks. For $N_{grid} \lesssim 20000$, the parallel execution is faster. As 20000 grid points can be adequate to many situations, the parallel execution is a valid option for achieving a speed up of the WF generation.

It is interesting to have a knowledge of the time taken by each stage of the WF generation procedure. This is done by a *profiling* on the code. We generated a number of waves with a single call (i.e. using numpy parallelization methods) and we measure the CPU time spent to execute each function. In the table below we reported the profiling results for the generation of 100 waves. We compare the results for two values of N_{grid}. Results were averaged on 3 runs.

Task	CPU time (ms)	
	$N_{grid} = 10^3$	$N_{grid} = 10^5$
Generation of raw WF	7.0 (40.8%)	7 (1.2%)
Interpolation to the user grid	6.2 (36.4%)	217 (37.7%)
Post processing	2.3 (13.7%)	320 (55.5%)
Total	17.1 (100%)	578 (100.0%)

The profiling confirms that the cost of generating the raw WF does not depend on the number of grid points. The interpolation and the post processing depends on N_{grid} and their cost grows dramatically as the user requires more and more points. It is important to stress that the latter two tasks are slow only because they deal with a huge amount of data. Indeed they perform trivial and quick operations. If such huge amount of datapoints is required, very little space is left for speed up.

5.3.2 Comparison with EOB

We now compare runtime of mlgw against standard EOB model. In our comparison, we use surrogate model `SEOBNRv2_opt` provided in `lalsuite`. The quantity of interest is the *speed-up* S of mlgw model. It is the ratio between runtime of `SEOBNRv2_opt` and runtime of mlgw to produce the same waveform.

We perform different numerical studies on how the speed up depends on different quantities of interests. In each test, we proceed as follows. We generate a waveform with a surrogate model. In order to generate a wave, we must specify a full set of physical parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \varphi_0)$, the sampling rate $f_{sam} = \frac{1}{t_{step}}$ and the wave starting frequency f_{min}. Here t_{step} is the EOB integration step. We express it in terms of the sampling rate because the latter is more relevant to real life application. The wave starting frequency controls the length of the GW signal generated. The lowest frequency is intimately related to τ_{min} (see e.g. eq (4.29)). As it is common in `lal`, we express every result below as a function of f_{min}, rather than of τ_{min}. The surrogate model outputs a waveform, as well as the time grid at which the wave is evaluated. We input to mlgw the physical parameters and the time grid and we compute the waveform. We measure the (clock) time required to generate the waves in the two methods and we compute the speed up. To ensure that the scale is the same for each wave, we compute waves with a constant total mass $M = 30\,\mathrm{M}_\odot$.

We report the following results.

- *Speed up distribution.* We asses the average speed up obtained in a realistic simulation. We randomly choose physical parameters for wave generation and we set a constant length of the signal $t_{min} = 16\,\mathrm{s}$. Furthermore we set a constant $f_{sam} = 4096\,\mathrm{Hz}$. This is the typical situation for a parameter estimation, where one has a signal of constant timelenght sampled at a constant frequency. In EOB wave generation, the wave must be trimmed to the required length. This operation slows down the EOB wave generation because the wave is generated for a (slightly) longer time than necessary. However, the same must be done in a real parameter estimation and thus it must be included in the runtime measurements for EOB. We report in figure 5.17 the histogram for the measured values of the speed up.

- *Speed up dependence on f_{min}.* This is to measure how the speed up depends on the length of the signal generated. We set constant physical parameters for wave generation and set a constant sampling rate $f_{sam} = 4096\,\mathrm{Hz}$ for the EOB model. We measure the speed up with different values of f_{min} in two cases:

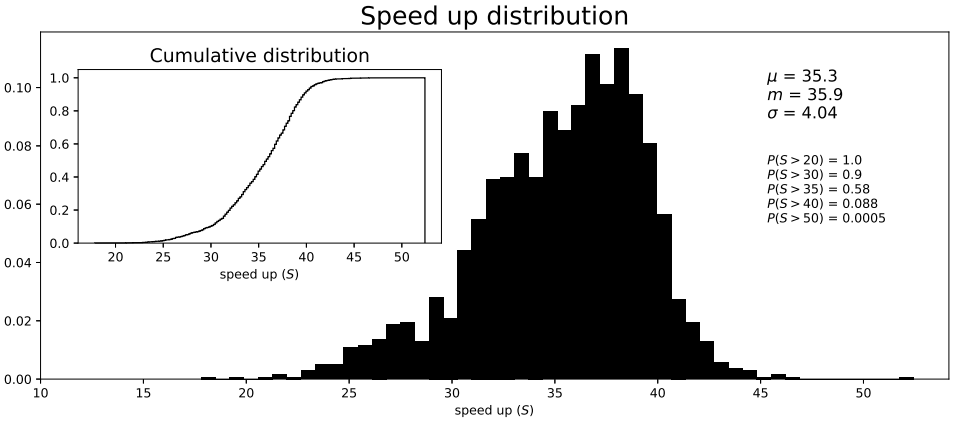

Figure 5.17: Histogram for values of the speed up given by mlgw as compared with standard EOB surrogate models. It is computed on $N = 2000$ test waveforms. Each WF is generated with random orbital parameters $(m_1, m_2, s_1, s_2, d_L, \iota, \phi_0)$ and lasts exactly 16 s. We set a constant total mass $M = 30\,\mathrm{M}_\odot$ and the sampling rate $f_{sam} = 4096\,\mathrm{Hz}$. In the inset we represent the cumulative distribution. We report the mean value μ, the standard deviation σ and the median value m as well as the cumulative distribution function evaluated at some selected points.

1. The time grid has a different number of grid points, depending on the lenght of the signal, as determined by the value of f_{sam}.
2. The time grid has the same number of grid points $N_{grid} = 10^4$, regardless the lenght of the signal

The results are reported in the two panels of figure 5.18.

- *Speed up dependence on sampling rate.* We measure how the speed up depends on the sampling rate f_{sam}. We set constant physical parameters for wave generation and set $f_{min} = 10\,\mathrm{Hz}$. This test is useful to asses how the number of grid points (which is known to be the bottleneck of mlgw) affects the speed up. We show the results in the right panel of figure 5.19.

The speed up reported in the histogram is computed for a typical situation occurring during a Parameter estimation (constant signal length and grid size). In figure 5.17, we note that, in this case, an average speed up as high as 35 is in reach of our mlgw model. The result is remarkable: a fast, yet reliable, GW generation is possible with Machine Learning techniques and allows for a significant speed up in a "real-life" scenario.

We note that the variance of the speed up is rather high. This is due to the fact that EOB is not designed to produce signals with a fixed time lenght but rather with a fixed frequency widow. Indeed the user must specify the starting frequency f_{min} and not the starting time of the wave. Equation (4.29) links the starting frequency to the total mass and the starting time. However, the relation does not take into account the spins. Consequentely, in order to allow for small variation of f_{min} due to spins, one must make a conservative choice and set a value for f_{min} which, in most cases, is lower than required. Thus, depending on spins, the EOB model computes a WF slightly longer than required. On the other hand, mlgw makes exactly the same amount of computation for each wave. This explains the high variance observed in 5.17.

To investigate further, we studied the speed up dependence on the starting frequency. In left panel of fig. 5.18, we computed the speed up with a constant sampling rate f_{sam}. This allow us to address

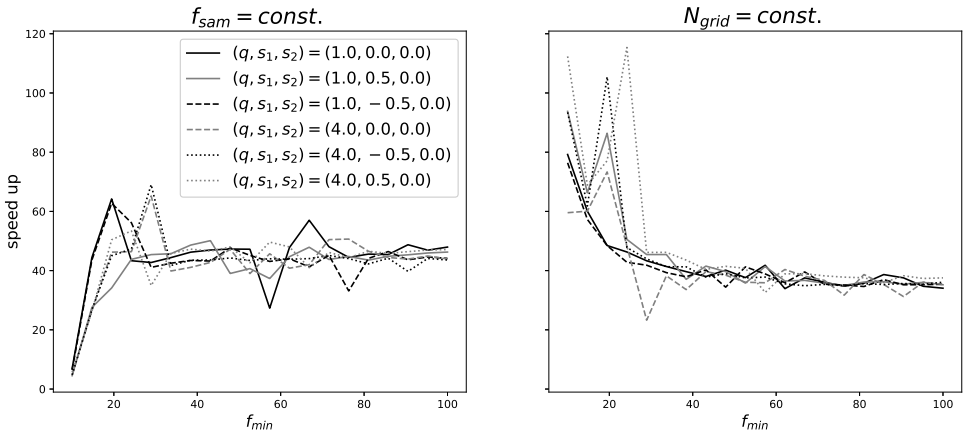

Figure 5.18: Speed up dependence on starting frequency f_{min}. Each series refers to different orbital parameters (q, s_1, s_2); the total mass is set constant $M = 30\,\mathrm{M}_\odot$. The legend refers to data in both panels. Each WF is generated with EOB with a constant integration step $t_{step} = \frac{1}{f_{sam}} = \frac{1}{4096\,\mathrm{Hz}}$. On left panel, we report the speed up in the case where a time grid of points sampled at f_{sam} is considered. We asses here the speed up in performing an identical task. On right panel, we report the speed up in the case where the wave is evaluated on a constant time grid of $N_{grid} = 10^4$ points. This is useful to factor out the dependence on N_{grid} of mlgw performance and to focus only the generation of the "physical signal".

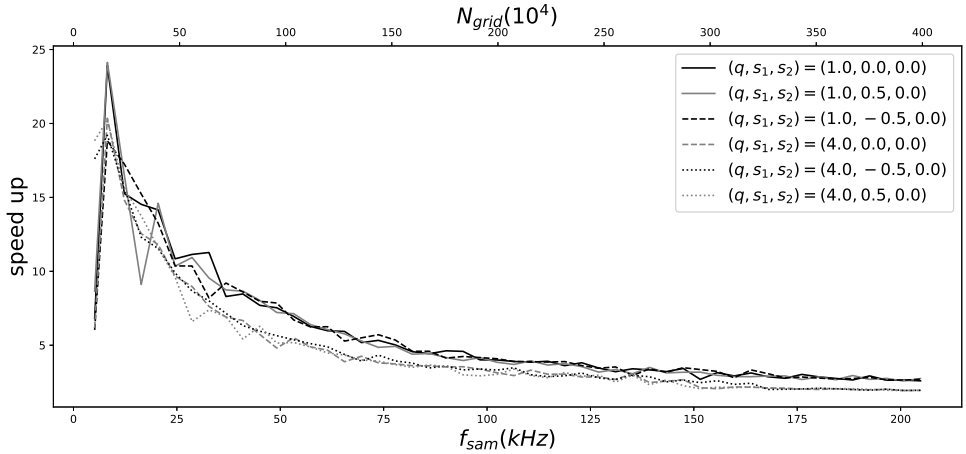

Figure 5.19: Speed up dependence on the sampling rate f_{sam}. Each wave is generated with the same starting frequency $f_{min} = 10\,\mathrm{Hz}$ and has total mass $M = 30\,\mathrm{M}_\odot$. As the length of each generated signal is the same, different sampling rate corresponds to different grid size N_{grid}. The test is useful to assess the speed up dependence on the number of grid points considered. As expected, for high number of N_{grid}, the generation of raw WF with ML has a low share of the total runtime and thus the the speed up decreases.

the case in which EOB model does always the same "amount of work" and in which mlgw performs slightly different tasks at each time (if f_{sam} is constant, the number of grid points need not to be constant). We note that, even in this unfavourable case, mlgw outperforms EOB by at least one order of magnitude. The speed up drops significantly at lower frequency, where the number of grid points grow high and makes mlgw slow.

In the right panel of fig. 5.18, we reported an "opposite situation". We set a constant grid and we generate a wave for the two models on the grid. As EOB is forced to work on a dense time grid, it has the more time consuming task of dealing with a large amount data; in turn, mlgw works directly on the smaller time grid, saving computation time. For small f_{min}, mlgw performs much better than EOB: the advantage of using a smaller grid is more significant when the time grid is long (f_{min} is small). At larger f_{min}, the performance of the two models is similar to the other case: the wave is short and EOB has a (much) smaller number of grid points to deal with. The experiment is useful to remove the strong dependence of mlgw performance on N_{grid}. By giving a constant N_{grid} we focus only on the "time to generate the physical signal" and we remove the variability related different values of N_{grid}. We cannot do anything similar with EOB: to generate a precise physical signal one needs to set t_{step} and thus needs grids with different length.

In figure 5.19 we asses the impact of a different grid length on the speed up. As expected, the speed up is lower if the N_{grid} (or f_{sam}) is high. In this case mlgw spends a large amount of time on performing basic operation on the data. Similar operations are performed by EOB and thus the speed up decrease. We stress however that a sampling rate $f_{sam} = 200\,\mathrm{kHz}$ is extremely high and it is not employed in any realistic experiment.

Chapter 6

Conclusions and future prospects

In this work, we built a Machine Learning model (called mlgw) which generates the gravitational wave signal from a binary Black Hole coalescence. The model considers the case of aligned spins (i.e. the non precessing case) and generates the wave when given the BH's masses m_1, m_2 and (dimensionless) spins s_1, s_2. It implements also the known dependence on source luminosity distance d_L, on inclination angle ι and reference phase φ_0. The model was trained with a dataset of waveforms, generated with EOB surrogate model. It is publicly available as a Python package in the PyPI repository at `https://pypi.org/project/mlgw/` and it can be installed with the command `pip install mlgw`. An interested user can find more relevant code on GitHub: `https://github.com/stefanoschmidt1995/MLGW`

As described in section 5.2, the model shows good accuracy at test times and it is able to widely match EOB generated waveforms. It shows little accuracy in the low q region and is completely unreliable in the high values of s_1 (see fig. 5.11). However, the performance at small mass ratio is still acceptable for our purposes. The behavior at high s_1 is not of great concern: the training model EOB is known to be unreliable in the region and the physics of high spins coalescence is not well understood yet. We used mlgw to perform a full parameter estimation on GW150914 detected signal and we obtained results similar to those published by the LIGO-Virgo collaboration. As assessed in section 5.3, mlgw is quicker to run than any model in the EOB family by a factor of ~ 35.

Now the obvious question is: can the model be improved? From validation results we see that PCA is able to reproduce a high dimensional wave using a few number of variables. The result is remarkable and it is hard to do better. Different conclusion for the MoE model: currently it is the "bottleneck" of the overall model and its performance limits in the accuracy. The fact that few data are required for training might suggest that the model underfits the data: a more complex model might use more data to produce a more accurate relation. For this reason, different regression methods were tried (especially neural networks) but none of them showed a clear sign of better achievements. There is another option: if an underfitting issue is present, it might be caused by an intrinsically noisy or not continuous relation and not (much) by a wrong choice of model. This is plausible: EOB is an approximation to a complex physics and the approximation might not yield a easy to fit relation. Further investigation might be done in this direction. However, as the model matches the EOB performance in the parameter estimation, we believe this is a marginal problem.

The main motivation to apply ML to GW signal modelling lies in the speed up in the runtime that such techniques can achieve. Indeed, the bottleneck of GW data analysis is the generation of waveforms during the Monte Carlo inference stage, which can last even for two weeks. As the detection rate will increase with the improvements of the detector's sensitivity, this will soon be unfeasible. In

this work we proved that a ML to generate GW waveforms can indeed provided the required speed up. With our model, a parameter estimation of two weeks can be easily performed (in principle) in a half a day!

Our work however is far from being over and several issues still require attention.

First of all, the potential speed up of parameter estimation can be even higher, if we take advantage of the closed form expression of the waveform provided by our model. Indeed, a closed form expression for the WF allows for prompt computation of the gradients with respect to the orbital parameters $\boldsymbol{\vartheta} = (m_1, m_2, s_1, s_2)$. Hamiltonian Montecarlo [75], a variant of Markov chain Montecarlo, employs gradients of the waveform to perform an effective sampling of the posterior distibution, which is able to "find quickly" the high density regions. Porter and Carré [4] applied Hamiltonian Montecarlo to parameter estimation. However, they only use analytical waveforms valid only at 3 PN order and completely neglects the late inspiral and the merger. A Machine Learning model for waveforms and its gradients could, in principle, extend the application of Hamiltonian Montecarlo to waveform which included also late inspiral, merger and ringdown. The model will provide a state-of-the-art accurate parameter estimation and, at the same time, it will offer a substantial speed up. We did not explore here this possibility: a full implementation of Hamiltonian Montecarlo algorithm is far from trivial and might require a thesis work itself. However, we believe that this is a promising way to speed up parameter estimation and it is one of the natural continuation of the work we started.

In our model we did not consider precessing system. We made this choice to keep the problem simple. Indeed, a precessing system has a non trivial parametrization of the two-body motion, which has to be included in the ML model. Moreover, the physics of precessing systems is not completely understood and EOB models are not able to catch every feature observed in NR simulations. The expertise gained for the simple non precessing case can be applied to deal with such complicacies.

A good starting point for tackling the problem is given in [82]. In the work, P. Schmidt et al. model the complicated precession problem (which depends on 6 spins components) to a simpler problem where the precession is controlled by a single spin parameter. More specifically, they provide a mapping from the full problem to the reduced problem as follow:

$$(s_{1x}, s_{1y}, s_{1z}) \longmapsto (s_P, 0, s_{1z})$$
$$(s_{2x}, s_{2y}, s_{2z}) \longmapsto (0, 0, s_{1z})$$

where s_P is a function of the mass ratio q and of the 4 spin components in the subspace $x-y$ orthogonal to the (precessing) orbital plane z (see [82, eq. 3]). In the authors' word, the effective spin parameter s_P "accurately captures the dominant precession-induced features in GW signals across the full parameter space". The waveform dependence on the effective spin parameter s_P can be fitted by a ML model in the same fashion of the other orbital parameters. The mapping above simplifies largely the problem: instead of 7 variables (the mass ratio q and 6 spins components), only 4 variables (q and the three spins) are considered in the regression.

As the physics is not completely understood and PN models are not completely reliable for non-precessing case, we expect the result in the precessing case to be less precise than those obtained in the non-precessing case. Despite this, tackling the precessing case is mandatory, in order to fully reproduce the results from LIGO-Virgo collaboration.

All surrogate models provides numerical solutions to approximate form of the Einstein equations. They are useful to catch the relevant physics, leaving out many details, but we expect them to have a

certain degree of inaccuracy, especially close to coalescence [1]. Numerical relativity solves numerically the full (with no approximation) Einstein equation and so far yields the best available solution to them. Clearly, NR waveforms are more accurate and reliable but they are much slow to generate and cannot be used in parameter estimation. A Machine Learning model can be effectively employed to extract as much information as possible from NR waveforms and to generalize them. It could be trained on the publicly available NR waveforms catalogs (see e.g. [83] and [84]) and would provide the best generalization of the numerical waveform. If the model proved to be reliable and effective, one could dispense with the surrogate models altogether and use only NR inputs for parameter estimation. This would ensure that all the relevant physics close to merger is adequately captured. Furthermore, a training set of NR waveforms can be used to address problems where an approximation to Einstein equations (such as PN expansion) is not available. In these cases, it is hard to build a reliable surrogate model and use the "exact" solution to the equation avoids the difficulty.

However, there are at least two problems also with this approach. First of all, at the moment there are too few NR waveforms ($O(10^2)$) available to perform a reliable training: to learn a reliable relation we need at least $O(10^3)$ waveforms. In the future we expect more and more waveforms to be included in the dataset, eventually allowing for a reliable training. Second, NR waveforms are too short as compared to detected signals in interferometers. Probably some form of input from surrogate models is still required to tackle the early inspiral. However, this is not a great limitation because in this region the PN series is more accurate and thus the loss in accuracy can still be small enough.

Of course, the approach we adopted for BBH coalescence can be extended to other compact object coalescence, such as binary neutron star (NS) coalescence or BH-NS coalescence. Our machine learning approach to signal generation is very flexible and it provides a framework which we expect to work for (almost) every source for which a number of training waveforms are available (also those for which no surrogate models are available). Machine learning models for various range of sources can be crucial in the future, where signals from a number of different sources are expected to be detected. In that scenario, a parameter estimation must be able to detect among different source and this will require a lot of computational work. Speed up will be more pressing.

[1] Indeed, every surrogate model has phenomenological inputs from NR simulation to properly model the physics close to merger.

Bibliography

[1] A. et al., "Observation of gravitational waves from a binary black hole merger," *Physical Review Letters*, vol. 116, Feb 2016.

[2] B. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, and et al., "Tests of general relativity with gw150914," *Physical Review Letters*, vol. 116, May 2016.

[3] A. et al., "Tests of general relativity with the binary black hole signals from the LIGO-Virgo catalog GWTC-1," *Physical Review D*, vol. 100, p. 22, Nov 2019.

[4] E. K. Porter and J. Carré, "A hamiltonian monte–carlo method for bayesian inference of supermassive black hole binaries," *Classical and Quantum Gravity*, vol. 31, p. 145004, Jul 2014.

[5] M. Maggiore, *Gravitational Waves: Volume 1: Theory and Experiments*. Gravitational Waves, OUP Oxford, 2008.

[6] M. Maggiore, *Gravitational Waves: Volume 2: Astrophysics and Cosmology*. OUP Oxford, 2018.

[7] R. M. Wald, *General Relativity*. Chicago, USA: Chicago Univ. Pr., 1984.

[8] B. Sathyaprakash and B. F. Schutz, "Physics, Astrophysics and Cosmology with Gravitational Waves," *Living Reviews in Relativity*, p. 140, Mar 2009.

[9] S. M. Carroll, "Lecture notes on general relativity," *arXiv e-prints*, pp. gr–qc/9712019, Dec 1997.

[10] J. D. Jackson, *Classical electrodynamics; 2nd ed.* New York, NY: Wiley, 1975.

[11] P. Ajith, M. Boyle, D. A. Brown, S. Fairhurst, M. Hannam, I. Hinder, S. Husa, B. Krishnan, R. A. Mercer, F. Ohme, C. D. Ott, J. S. Read, L. Santamaria, and J. T. Whelan, "Data formats for numerical relativity waves," 2007.

[12] L. Baiotti and L. Rezzolla, "Binary neutron star mergers: a review of einstein's richest laboratory," *Reports on Progress in Physics*, vol. 80, p. 096901, Jul 2017.

[13] M. D. Duez and Y. Zlochower, "Numerical relativity of compact binaries in the 21st century," *Reports on Progress in Physics*, vol. 82, p. 016902, Nov 2018.

[14] V. Cardoso, L. Gualtieri, C. A. R. Herdeiro, and U. Sperhake, "Exploring New Physics Frontiers Through Numerical Relativity," *Living Reviews in Relativity*, vol. 18, p. 1, Sep 2015.

[15] H. Asada and T. Futamase, "Chapter 2. post-newtonian approximation," *Progress of Theoretical Physics Supplement*, vol. 128, p. 123–181, 1997.

[16] K. Arun, A. Buonanno, G. Faye, and E. Ochsner, "Higher-order spin effects in the amplitude and phase of gravitational waveforms emitted by inspiraling compact binaries: Ready-to-use gravitational waveforms," *Phys. Rev. D*, vol. 79, p. 104023, May 2009.

[17] L. E. Kidder, "Coalescing binary systems of compact objects to post 5/2-newtonian order. v. spin effects," *Phys. Rev. D*, vol. 52, pp. 821–847, Jul 1995.

[18] T. A. Apostolatos, C. Cutler, G. J. Sussman, and K. S. Thorne, "Spin-induced orbital precession and its modulation of the gravitational waveforms from merging binaries," *Phys. Rev. D*, vol. 49, pp. 6274–6297, Jun 1994.

[19] S. Ossokine, M. Boyle, L. E. Kidder, H. P. Pfeiffer, M. A. Scheel, and B. Szilágyi, "Comparing post-newtonian and numerical relativity precession dynamics," *Phys. Rev. D*, vol. 92, Nov 2015.

[20] E. Baird, S. Fairhurst, M. Hannam, and P. Murphy, "Degeneracy between mass and spin in black-hole-binary waveforms," *Phys. Rev. D*, vol. 87, p. 024035, Jan. 2013.

[21] M. Pürrer, M. Hannam, and F. Ohme, "Can we measure individual black-hole spins from gravitational-wave observations?," *Phys. Rev.*, vol. D93, no. 8, p. 084042, 2016.

[22] F. Ohme, A. B. Nielsen, D. Keppel, and A. Lundgren, "Statistical and systematic errors for gravitational-wave inspiral signals: A principal component analysis," *Physical Review D*, vol. 88, Aug 2013.

[23] Löffler et al., "The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics," *Classical and Quantum Gravity*, vol. 29, p. 115001, Jun 2012.

[24] Teukolsky, Saul A., "Perturbations of a Rotating Black Hole. I. Fundamental Equations for Gravitational, Electromagnetic, and Neutrino-Field Perturbations," *Astrophysical Journal*, vol. 185, pp. 635–648, Oct 1973.

[25] G. Carullo, W. Del Pozzo, and J. Veitch, "Observational black hole spectroscopy: A time-domain multimode analysis of GW150914," *Phys. Rev. D*, vol. 99, p. 123029, Jun 2019.

[26] E. Berti, V. Cardoso, and C. M. Will, "Gravitational-wave spectroscopy of massive black holes with the space interferometer lisa," *Phys Rev. D*, vol. 73, Mar 2006.

[27] M. Isi, M. Giesler, W. M. Farr, M. A. Scheel, and S. A. Teukolsky, "Testing the no-hair theorem with gw150914," *Physical Review Letters*, vol. 123, Sep 2019.

[28] Carullo, Gregorio and van der Schaaf, Laura and London, Lionel and Pang, Peter T. H. and Tsang, Ka Wa and Hannuksela, Otto A. and Meidam, Jeroen and Agathos, Michalis and Samajdar, Anuradha and Ghosh, Archisman and Li, Tjonnie G. F. and Del Pozzo, Walter and Van Den Broeck, Chris, "Empirical tests of the black hole no-hair conjecture using gravitational-wave observations," *Phys. Rev. D*, vol. 98, p. 104020, Nov 2018.

[29] R. Brito, A. Buonanno, and V. Raymond, "Black-hole spectroscopy by making full use of gravitational-wave modeling," *Physical Review D*, vol. 98, Oct 2018.

[30] A. Buonanno and T. Damour, "Effective one-body approach to general relativistic two-body dynamics," *Phys. Rev. D*, vol. 59, p. 084006, Mar 1999.

[31] T. Damour and A. Nagar, "The Effective One Body description of the Two-Body problem," *arXiv e-prints*, p. arXiv:0906.1769, Jun 2009.

[32] A. Nagar, S. Bernuzzi, W. Del Pozzo, G. Riemenschneider, S. Akcay, G. Carullo, P. Fleig, S. Babak, K. W. Tsang, M. Colleoni, and et al., "Time-domain effective-one-body gravitational waveforms for coalescing compact binaries with nonprecessing spins, tides, and self-spin effects," *Phys. Rev. D*, vol. 98, Nov 2018.

[33] P. Kumar, T. Chu, H. Fong, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, M. A. Scheel, and B. Szilagyi, "Accuracy of binary black hole waveform models for aligned-spin binaries," *Phys. Rev. D*, vol. 93, May 2016.

[34] LIGO Scientific Collaboration, "LIGO Algorithm Library - LALSuite." free software (GPL), 2018.

[35] Pitkin, Matthew and Reid, Stuart and Rowan, Sheila and Hough, James, "Gravitational Wave Detection by Interferometry (Ground and Space)," *Living Reviews in Relativity*, vol. 14, p. 5, Jul 2011.

[36] D. Martynov, E. Hall, B. Abbott, R. Abbott, T. Abbott, C. Adams, R. Adhikari, R. Anderson, S. Anderson, K. Arai, and et al., "Sensitivity of the advanced ligo detectors at the beginning of gravitational wave astronomy," *Physical Review D*, vol. 93, Jun 2016.

[37] J. Veitch and A. Vecchio, "Bayesian coherent analysis of in-spiral gravitational wave signals with a detector network," *Phys. Rev.*, vol. D81, p. 062003, 2010.

[38] J. Veitch *et al.*, "Parameter estimation for compact binaries with ground-based gravitational-wave observations using the LALInference software library," *Phys. Rev.*, vol. D91, no. 4, p. 042003, 2015.

[39] J. Aasi *et al.*, "Parameter estimation for compact binary coalescence signals with the first generation gravitational-wave detector network," *Phys. Rev.*, vol. D88, p. 062001, 2013.

[40] B. Abbott, R. Abbott, T. Abbott, M. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, and et al., "Properties of the binary black hole merger gw150914," *Physical Review Letters*, vol. 116, Jun 2016.

[41] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[42] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[43] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, MIT Press, 2012.

[44] R. M. Neal, "Probabilistic inference using markov chain monte carlo methods," technical report crg-tr-93-1, Dept. of Computer Science, University of Toronto, 1993.

[45] A. Gelman and X.-L. Meng, "Simulating normalizing constants: from importance sampling to bridge sampling to path sampling," *Statist. Sci.*, vol. 13, pp. 163–185, 05 1998.

[46] J. Skilling, "Nested sampling for general bayesian computation," *Bayesian Analysis*, vol. 1, pp. 833–860, 12 2006.

[47] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *ArXiv*, vol. abs/1601.00670, 2017.

[48] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, "Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy," *arXiv e-prints*, p. arXiv:1909.06296, Sep 2019.

[49] A. J. K. Chua and M. Vallisneri, "Learning Bayes' theorem with a neural network for gravitational-wave inference," *arXiv e-prints*, p. arXiv:1909.05966, Sep 2019.

[50] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *Trans. Evol. Comp*, vol. 1, p. 67–82, Apr. 1997.

[51] K. Kowsari, M. Heidarysafa, D. E. Brown, K. J. Meimandi, and L. E. Barnes, "RMDL: random multimodel deep learning for classification," *CoRR*, vol. abs/1805.01890, 2018.

[52] S. Ruder, "An overview of gradient descent optimization algorithms," *ArXiv*, vol. abs/1609.04747, 2016.

[53] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[55] C. G., "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[56] H. K., "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[57] Y. Bengio, "Learning deep architectures for ai," *Foundations*, vol. 2, pp. 1–55, 01 2009.

[58] V. N. Temlyakov, "Greedy approximation," *Acta Numerica*, vol. 17, p. 235–409, 2008.

[59] V. Temlyakov, *Greedy Approximation*. New York, NY, USA: Cambridge University Press, 1st ed., 2011.

[60] M. Pürrer, "Frequency-domain reduced order models for gravitational waves from aligned-spin compact binaries," *Classical and Quantum Gravity*, vol. 31, p. 195010, Oct 2014.

[61] S. E. Field, C. R. Galley, F. Herrmann, J. S. Hesthaven, E. Ochsner, and M. Tiglio, "Reduced basis catalogs for gravitational wave templates," *Physical Review Letters*, vol. 106, Jun 2011.

[62] A. J. Chua, C. R. Galley, and M. Vallisneri, "Reduced-order modeling with artificial neurons for gravitational-wave inference," *Physical Review Letters*, vol. 122, May 2019.

[63] Y. Setyawati, M. Pürrer, and F. Ohme, "Regression methods in waveform modeling: a comparative study," *arXiv e-prints*, p. arXiv:1909.10986, Sep 2019.

[64] M. Tiglio and A. Villanueva, "On ab initio closed-form expressions for gravitational waves," *arXiv e-prints*, p. arXiv:1911.00644, Nov 2019.

[65] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, "Matching matched filtering with deep networks for gravitational-wave astronomy," *Physical Review Letters*, vol. 120, Apr 2018.

[66] P. G. Krastev, "Real-Time Detection of Gravitational Waves from Binary Neutron Stars using Artificial Neural Networks," *arXiv e-prints*, p. arXiv:1908.03151, Aug 2019.

[67] K. Kim, T. G. F. Li, R. K. L. Lo, S. Sachdev, and R. S. H. Yuen, "Ranking Candidate Signals with Machine Learning in Low-Latency Search for Gravitational-Waves from Compact Binary Mergers," *arXiv e-prints*, p. arXiv:1912.07740, Dec 2019.

[68] George, Daniel and Huerta, E. A., "Deep neural networks to enable real-time multimessenger astrophysics," *Phys. Rev. D*, vol. 97, p. 044039, Feb 2018.

[69] H. Shen, E. A. Huerta, Z. Zhao, E. Jennings, and H. Sharma, "Deterministic and Bayesian Neural Networks for Low-latency Gravitational Wave Parameter Estimation of Binary Black Hole Mergers," *arXiv e-prints*, p. arXiv:1903.01998, Mar 2019.

[70] C. Chatterjee, L. Wen, K. Vinsen, M. Kovalam, and A. Datta, "Using deep learning to localize gravitational wave sources," *Phys. Rev. D*, vol. 100, p. 103025, Nov 2019.

[71] A. J. K. Chua and M. Vallisneri, "Learning Bayes' theorem with a neural network for gravitational-wave inference," *arXiv e-prints*, p. arXiv:1909.05966, Sep 2019.

[72] F. Tonolini, A. Lyons, P. Caramazza, D. Faccio, and R. Murray-Smith, "Variational Inference for Computational Imaging Inverse Problems," *arXiv e-prints*, p. arXiv:1904.06264, Apr 2019.

[73] L. Haegel and S. Husa, "Predicting the properties of black holes merger remnants with Deep Neural Networks," *arXiv e-prints*, p. arXiv:1911.01496, Nov 2019.

[74] M. L. Chan, I. S. Heng, and C. Messenger, "Detection and Classification of Supernova Gravitational Waves Signals: A Deep Learning Approach," *arXiv e-prints*, p. arXiv:1912.13517, Dec 2019.

[75] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," 2017.

[76] A. Arbey and J.-F. Coupechoux, "Black hole mergers, gravitational waves and scaling relations," 2019.

[77] T. Oliphant, "NumPy: A guide to NumPy." USA: Trelgol Publishing, 2006–.

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Acts of ICLR*, 2014.

[79] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–.

[80] X. Jiménez-Forteza, D. Keitel, S. Husa, M. Hannam, S. Khan, and M. Pürrer, "Hierarchical data-driven approach to fitting numerical relativity data for nonprecessing binary black holes with an application to final spin and radiated energy," *Phys. Rev.*, vol. D95, no. 6, p. 064024, 2017.

[81] D. Foreman-Mackey, "corner.py: Scatterplot matrices in python," *The Journal of Open Source Software*, vol. 24, 2016.

[82] P. Schmidt, F. Ohme, and M. Hannam, "Towards models of gravitational waveforms from generic binaries: Modelling precession effects with a single effective precession parameter," *Physical Review D*, vol. 91, Jan 2015.

[83] A. H. Mroue *et al.*, "Catalog of 174 Binary Black Hole Simulations for Gravitational Wave Astronomy," *Phys. Rev. Lett.*, vol. 111, no. 24, p. 241104, 2013.

[84] J. Healy, C. O. Lousto, Y. Zlochower, and M. Campanelli, "The rit binary black hole simulations catalog," *Classical and Quantum Gravity*, vol. 34, p. 224001, Oct 2017.