

# Fast and accurate gravitational-wave modelling with principal component regression

GWMull  
2021

Cyril Cano (Gipsa-lab)  
Nicolas Le Bihan (Gipsa-lab)  
Éric Chassande-Mottin (APC)

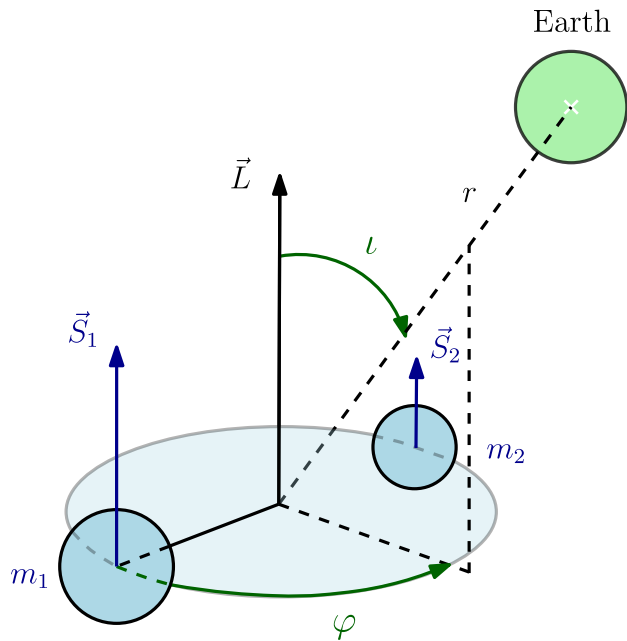
Preprint : <https://tds.virgo-gw.eu/ql/?c=16877>  
Git : <https://git.ligo.org/cyril.cano/gw-generation>



# Motivations

- Bayesian inference **needs large amount of waveforms** ( $\sim 10^5$ )
- Time domain GW generation is **computationally expensive**
- **Need for fast and accurate generative model**
- Reduced Order Models [Pürrer 2016]
- ML with Mixture of Experts (med. mismatch  $\sim 10^{-4}$ ) [Schmidt et al. 2020]
  
- Proposed model: **principal component regression**

# BBH parameters



$m_1, m_2, S_{1z}, S_{2z}$  : binary parameters

$\iota, \varphi$  : line of sight

$r$  : luminosity distance

**Generative model**

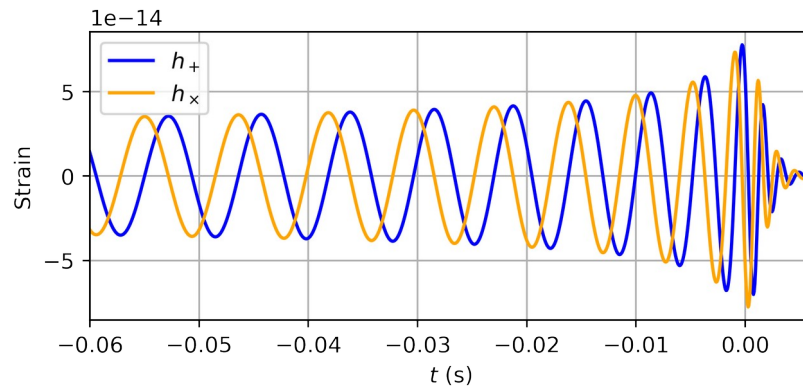


**SEOBNRv4**

**Only (2,2) mode**

$$h(t; m_1, m_2, S_{1z}, S_{2z}, \iota, \varphi, r)$$

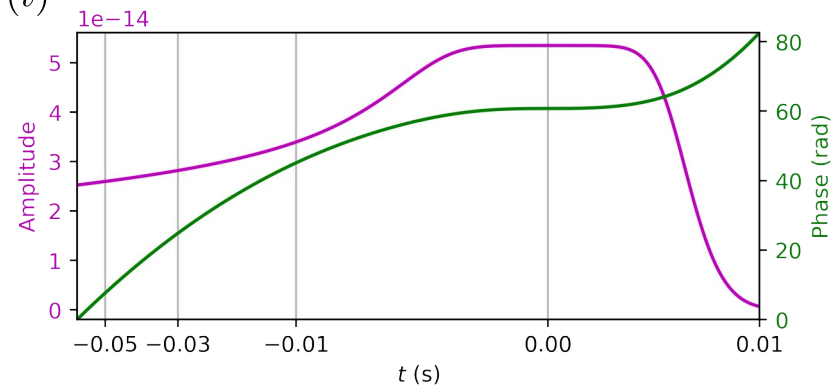
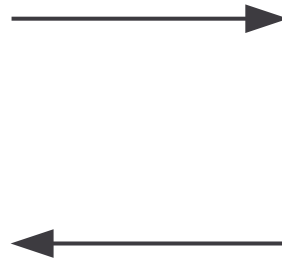
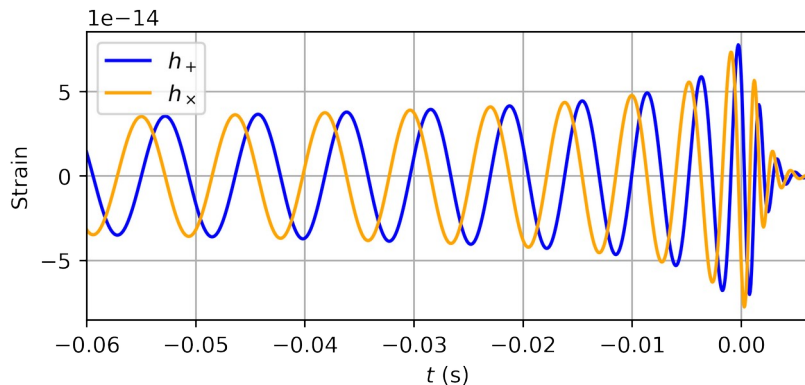
$$h(t) = h_+ - ih_\times(t)$$



# Waveform attributes

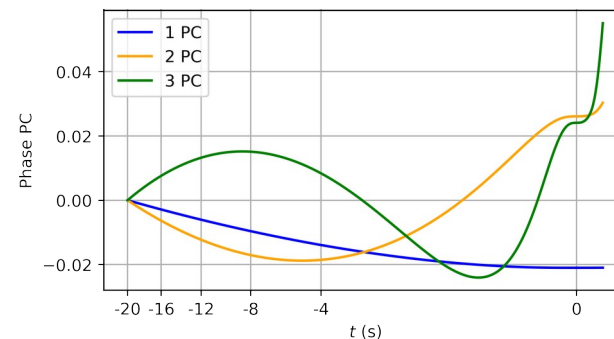
$$a(t) = |h(t)|$$

$$\Phi(t) = \arctan \frac{h_+(t)}{h_\times(t)}$$

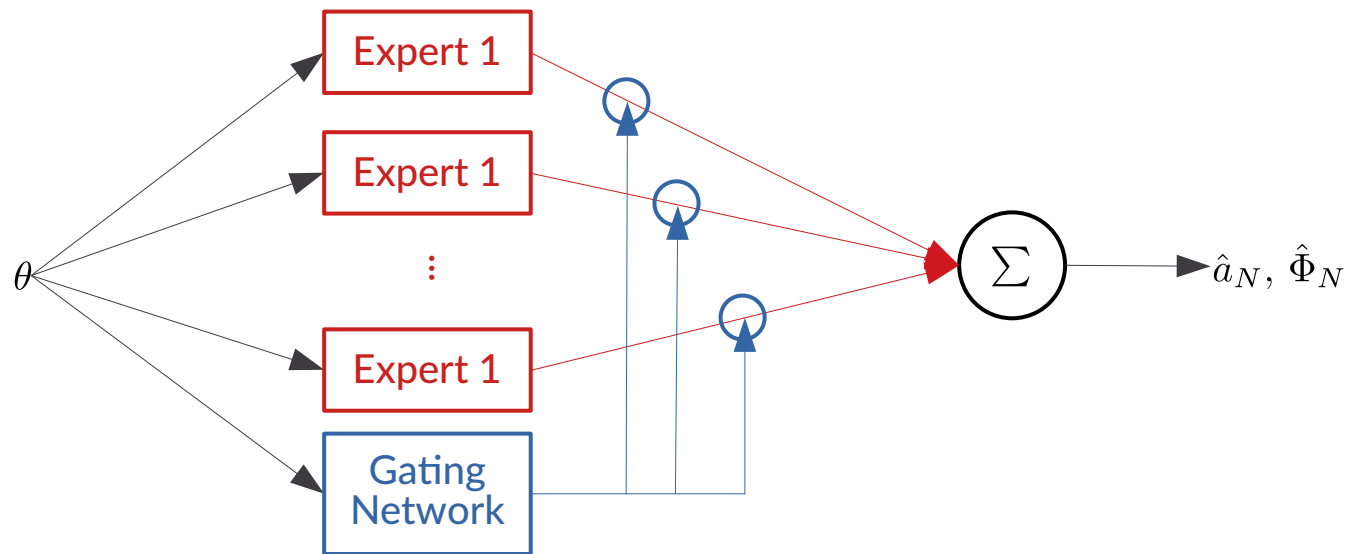
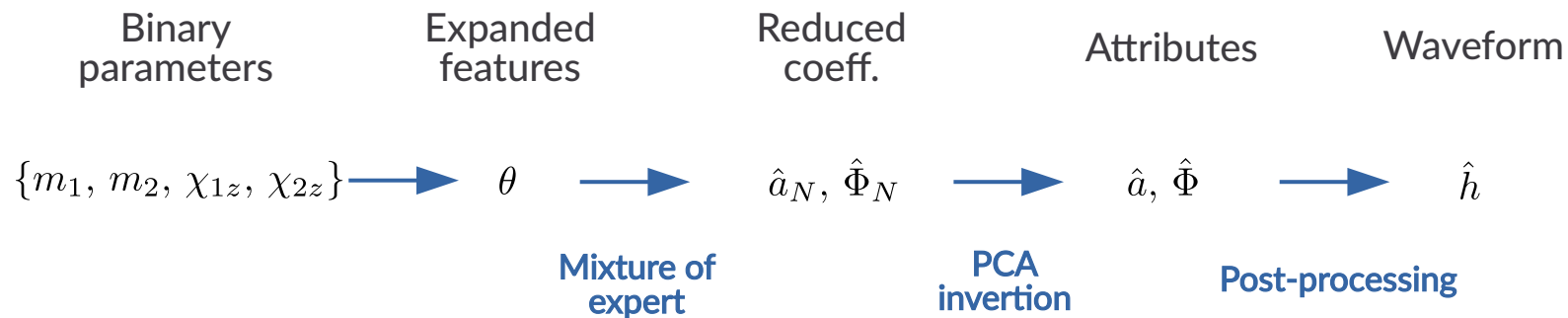


$$h(t) = a(t)e^{-i\Phi(t)}$$

- ML model generates amplitude and phase
- Non uniform time grid  $\text{sign}(t)|t|^{\frac{1}{\alpha}}$
- Needs dimension reduction: **truncated PCA**

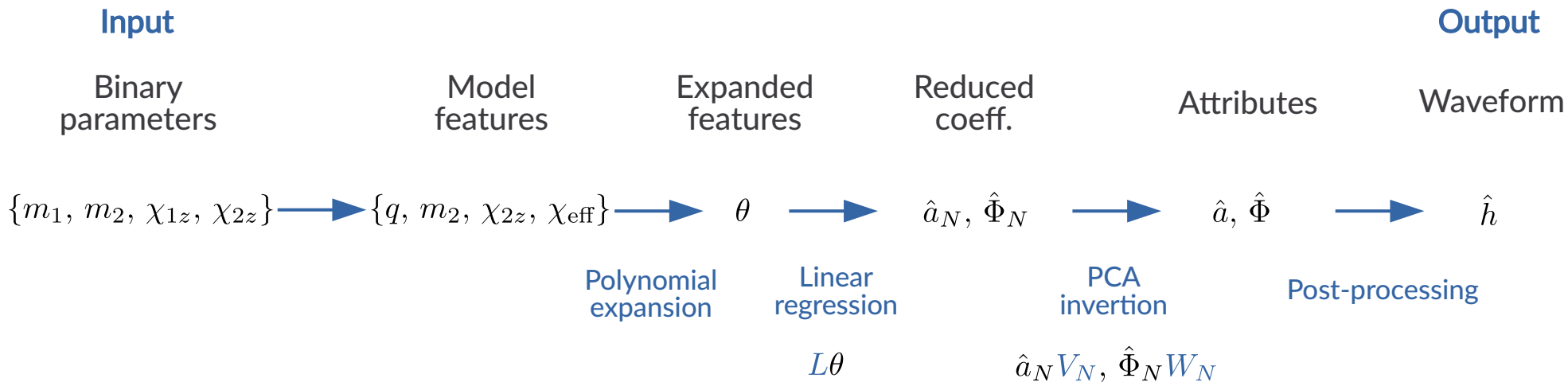


# Schmidt's model [Schmidt et al. 2020]



# Overview of our model

## Generation

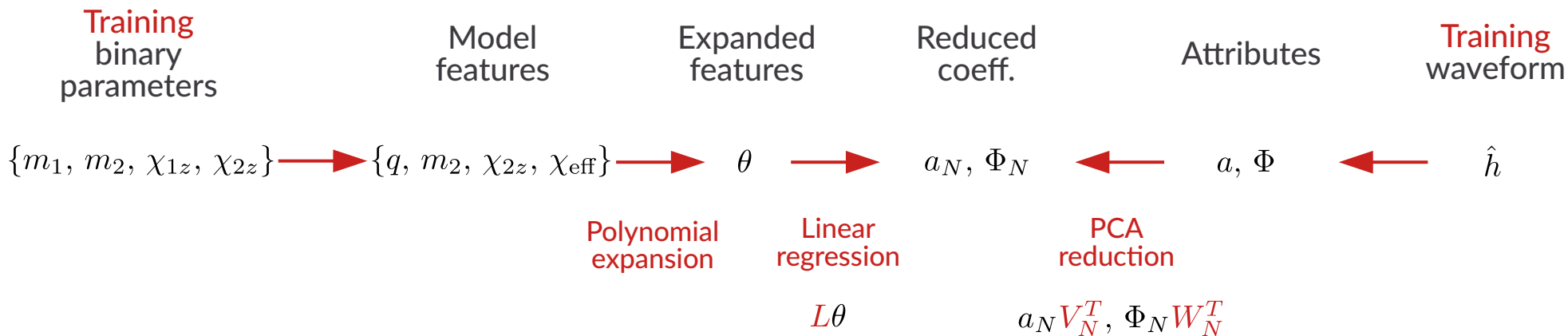


**Goodness-of-fit metric:**

$$\text{mismatch}(h, g) = \min_{\tau \in \mathbb{R}} \left[ 1 - \frac{|\langle h_\tau, g \rangle|}{\|h_\tau\| \|g\|} \right] \quad \text{with} \quad \langle f, g \rangle = \int \frac{h(f)g^*(f)}{S(f)} df$$

# Overview of our model

## Fitting



Goodness-of-fit metric:

$$\text{mismatch}(h, g) = \min_{\tau \in \mathbb{R}} \left[ 1 - \frac{|\langle h_\tau, g \rangle|}{\|h_\tau\| \|g\|} \right] \quad \text{with} \quad \langle f, g \rangle = \int \frac{h(f)g^*(f)}{S(f)} df$$

# Hyperparameters tuning

## Feature set

- Tested features:

$$m_1, m_2, \chi_{1z}, \chi_{2z}, q, \mathcal{M}, \chi_{\text{eff}}, m_1^{-1}, m_2^{-1}$$

- Tested feature sets:

$\{m_1\}$	$\{m_1, m_2\}$	$\{m_1, m_2, \chi_{1z}\}$	$\dots$
$\{m_2\}$	$\{m_1, \chi_{1z}\}$	$\{m_1, m_2, q\}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	

- Several good choices:

$\{\chi_{2z}, \chi_{\text{eff}}, \mathcal{M}\}$	$\{q, \chi_{2z}, \chi_{\text{eff}}, \mathcal{M}\}$
$\{q, m_2, \chi_{2z}, \chi_{\text{eff}}\}$	$\{\chi_{2z}, \chi_{\text{eff}}, m_1, m_2^{-1}\}$

Selected

## Polynomial degree

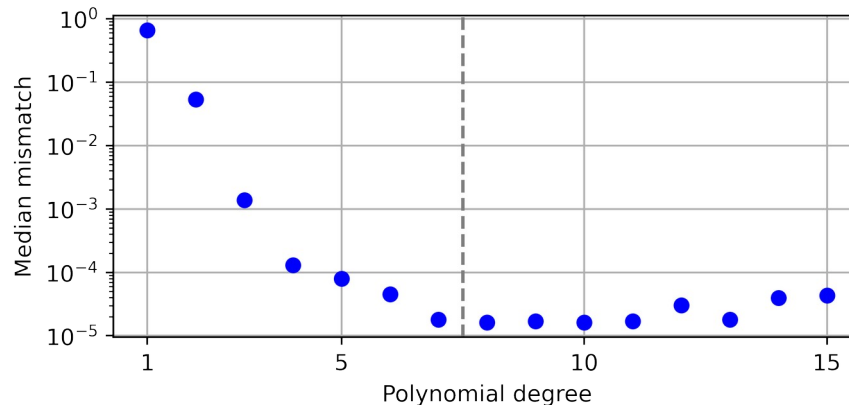
First order      $\{a, b\}$

Second order    $\{a, b, ab, a^2, b^2\}$

Third order      $\{a, b, ab, a^2, b^2, a^2b, ab^2, a^3, b^3\}$

$\vdots$

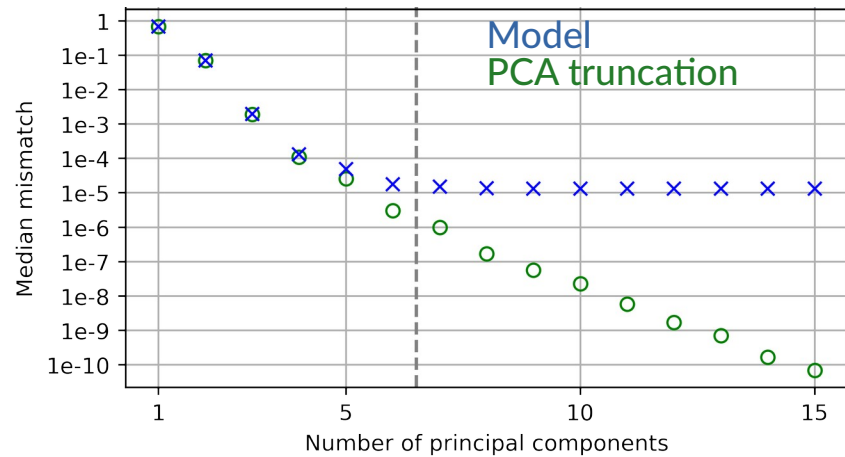
Seventh order used for regression



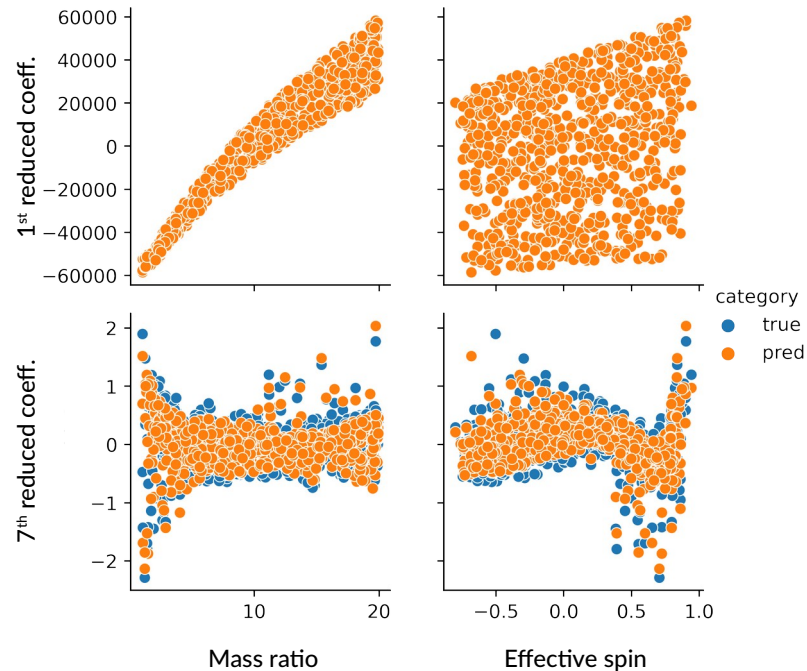


# Hyperparameters tuning

Number of PC for the phase



Six PC used for dimension reduction



# Results on SEOBNRv4

## Dataset properties

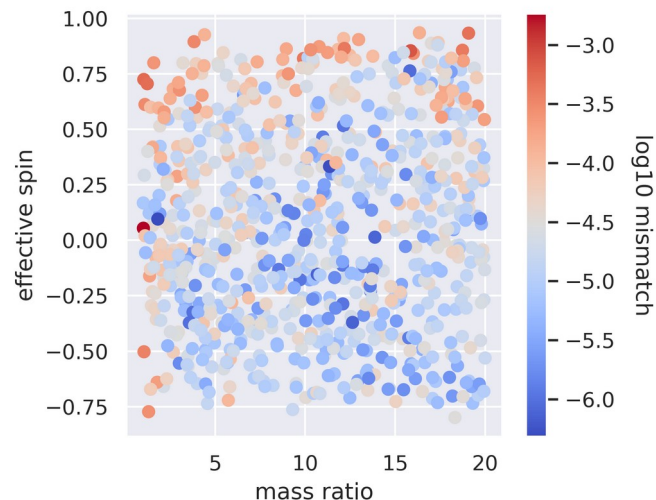
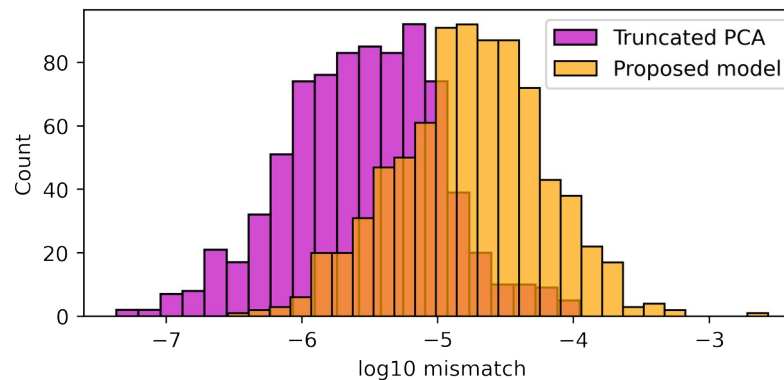
- Training size: 3200    Testing size: 800
- Mass ratio:  $U([1, 20])$
- Dimensionless spins:  $U([-0.8, 0.95])$

## Accurate

- $Q_{50\%} = 2.10^{-5}$
- $Q_{5\%} = 2.10^{-6}$
- $Q_{95\%} = 1.5.10^{-4}$

## Fast

- **~100 times faster than SEOBNRv4**
- **Can be faster without interpolation**  
(from non-uniform to regular time grid)



# Python library

## README

---

Contact : [cyril.cano@gipsa-lab.fr](mailto:cyril.cano@gipsa-lab.fr)

### Overview

---

This Git repo provides the library `mlpgw.py` that allows to train a machine-learning model able to regress gravitational-wave waveform from a set of examples as described in this [article](#).

This Git repo includes several notebooks that allow to reproduce the results presented in the paper.

The learning part is mainly based on [scikit-learn](#). This package is included in the required [environment](#).

Take care that in the notebook a gravitational waveform  $h(t)$  is denoted There was an error rendering this math block but There was an error rendering this math block in the paper.

### Installation

---

Clone this Git repo and create the environment `gw-generation` by running:

```
conda env create -f environment.yml
```

Activate the environment

```
conda activate gw-generation
```

... and run the following command line from this folder:

```
conda develop .
```

### How to generate a waveform using a pre-computed ML model?

---

This [notebook](#) shows how to generate a waveform with a pre-computed ML model.

The pre-computed model is stored in a set of Pickle files (see [data/](#))

Git : <https://git.ligo.org/cyril.cano/gw-generation>

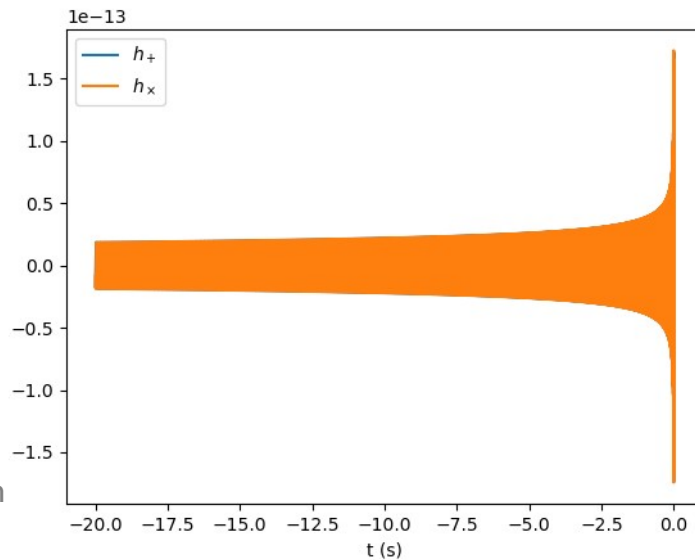
# Python library

```
In [1]: 1 import numpy as np
        2 import matplotlib.pyplot as plt
        3 import mlpgw
        4
        5 %matplotlib notebook
```

```
In [2]: 1 # Download the model
        2 model = mlpgw.load_obj('../data/model')
```

```
In [3]: 1 # Make prediction
        2 h_pred = model.predict(m1=15, m2=5, slz=0.9, s2z=0.2)
```

```
In [4]: 1 # Plot it
        2 plt.figure()
        3 plt.plot(h_pred['time'], h_pred['hp'], label=r'$h_+$')
        4 plt.plot(h_pred['time'], h_pred['hc'], label=r'$h_-$')
        5 plt.xlabel('t (s)')
        6 plt.legend()
        7 plt.show()
```



# Conclusion/perspectives

- Take home messages :
- **Fast and accurate** GW generation with **principal component regression**
  - **Applicability up to SNR  $\sim 225$**  (18 in the worst case) \* : mismatch  $< \frac{N}{2\text{SNR}^2}$
  - **Non conventional features** lead to better results
  - Simple method with off-the-shelf algorithms from scikit-learn

- Perspectives :
- Subdominant modes
  - Comparison with other ML state of the art algorithms
  - **Precessing BBH**

\* see [Chatziioannou 2017]

# R<sup>2</sup> scores

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

PC	1	2	3	4	5	6
<i>a</i>	1.67e-06	0.00231	0.0214	0.00728	1.42	0.177
$\Phi$	1.65e-09	9.44e-07	0.000248	0.00322	0.00401	0.0326