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)

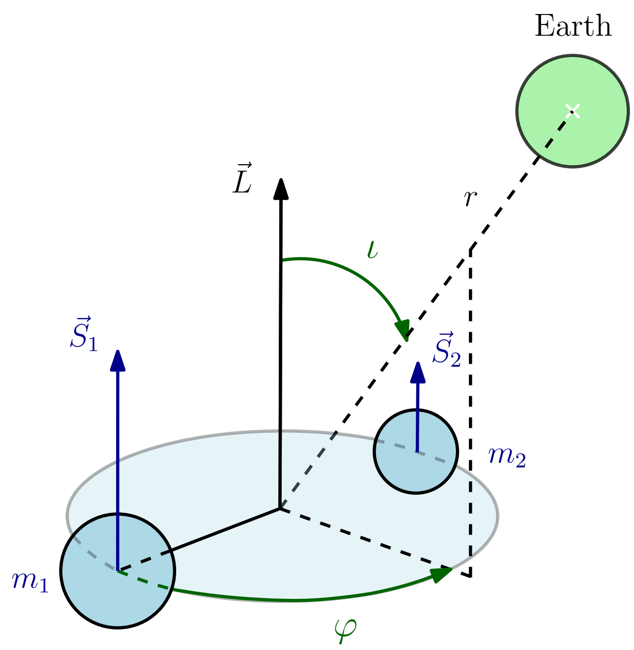
Preprint : https://tds.virgo-gw.eu/ql/?c=16877
Git : https://git.ligo.org/cyril.cano/gw-generation

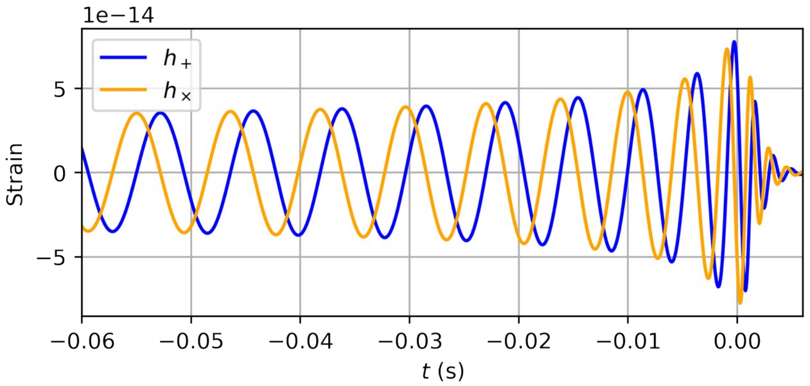
Motivations

- Bayesian inference **needs large amount of waveforms** (~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 ~10^{-4}) [Schmidt et al. 2020]

- Proposed model: **principal component regression**

BBH parameters





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

ι, φ : line of sight

r : luminosity distance

Generative model

SEOBNRv4

Only (2,2) mode

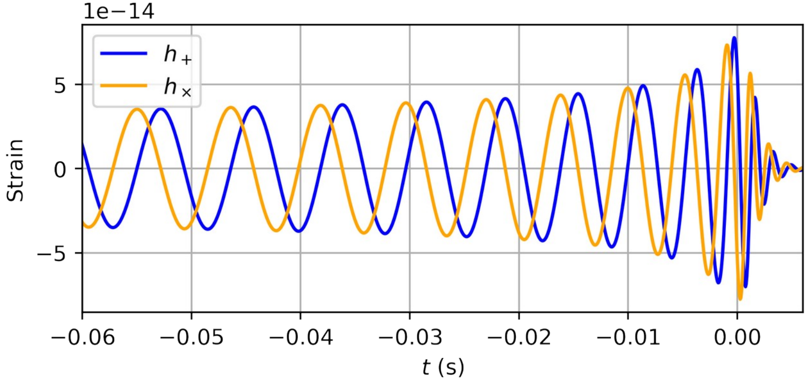
$$\mathrm{h}(\mathrm{t;m}_1,\, m_2,\, S_{1z},\, S_{2z},\, \iota,\, \varphi,\, r)$$

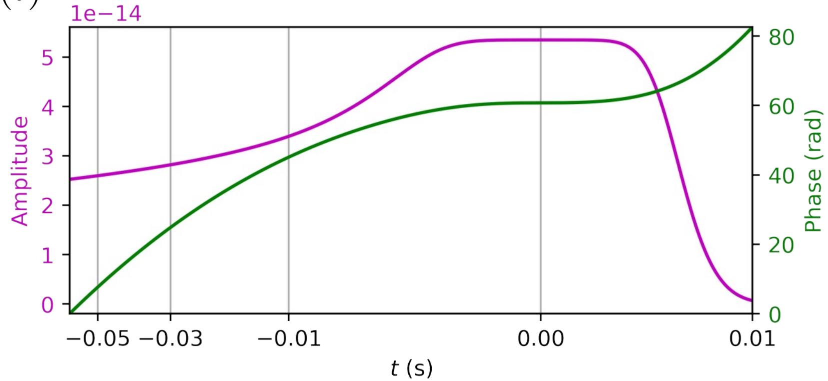
$$\mathrm{h}(\mathrm{t}) = \mathrm{h}_+ - i h_\times(t)$$

Waveform attributes

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

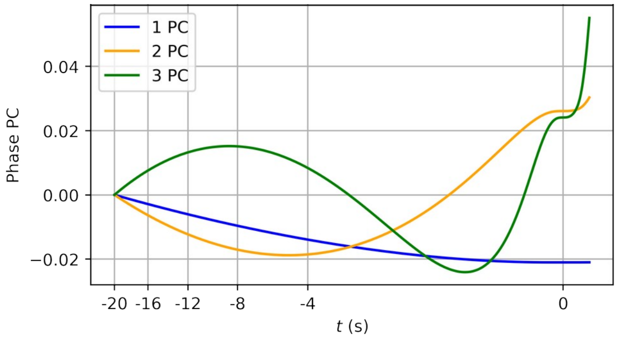
$$\Phi(t) = \arctan \frac{h_+(t)}{h_\times(t)}$$





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

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

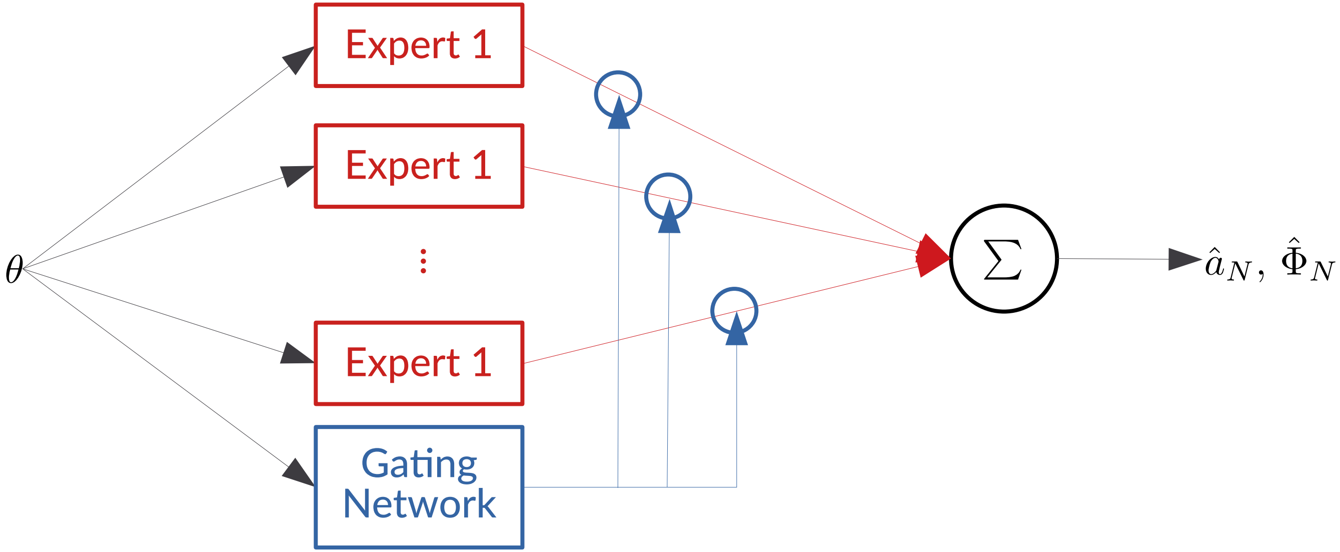


Schmidt's model [Schmidt et al. 2020]

Input — **Output**

Binary parameters → Model features → Expanded features → Reduced coeff. → Attributes → Waveform

$$\{m_1,\, m_2,\, \chi_{1z},\, \chi_{2z}\} \longrightarrow \{q,\, \chi_{1z},\, \chi_{2z}\} \xrightarrow{\text{Polynomial expansion}} \theta \xrightarrow{\text{Mixture of Expert}} \hat{a}_N,\, \hat{\Phi}_N \xrightarrow{\text{PCA invertion}} \hat{a},\, \hat{\Phi} \xrightarrow{\text{Post-processing}} \hat{h}$$



Overview of our model

Generation

Input → **Output**

Binary parameters	Model features	Expanded features	Reduced coeff.	Attributes	Waveform
$\{m_1,\, m_2,\, \chi_{1z},\, \chi_{2z}\}$	$\{q,\, m_2,\, \chi_{2z},\, \chi_{\text{eff}}\}$	θ	$\hat{a}_N,\, \hat{\Phi}_N$	$\hat{a},\, \hat{\Phi}$	$\hat{h}$

$$\{m_1,\, m_2,\, \chi_{1z},\, \chi_{2z}\} \longrightarrow \{q,\, m_2,\, \chi_{2z},\, \chi_{\text{eff}}\} \xrightarrow{\text{Polynomial expansion}} \theta \xrightarrow{\text{Linear regression}} \hat{a}_N,\, \hat{\Phi}_N \xrightarrow{\text{PCA invertion}} \hat{a},\, \hat{\Phi} \xrightarrow{\text{Post-processing}} \hat{h}$$

Linear regression: $L\theta$

PCA invertion: $\hat{a}_N V_N,\, \hat{\Phi}_N W_N$

Goodness-of-fit metric:

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

Overview of our model

Fitting

Training binary parameters | Model features | Expanded features | Reduced coeff. | Attributes | Training waveform

$$\{m_1,\, m_2,\, \chi_{1z},\, \chi_{2z}\} \longrightarrow \{q,\, m_2,\, \chi_{2z},\, \chi_{\rm eff}\} \xrightarrow[\text{Polynomial expansion}]{} \theta \xrightarrow[\text{Linear regression}]{} a_N,\, \Phi_N \xleftarrow[\text{PCA reduction}]{} a,\, \Phi \longleftarrow \hat{h}$$

$L\theta$

$a_N V_N^T,\, \Phi_N W_N^T$

Goodness-of-fit metric:

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

Hyperparameters tuning

Feature set

- **Tested features:**

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

- **Tested feature sets:**

$$\begin{array}{cccc} \{\mathrm{m}_1\} & \{\mathrm{m}_1,\ m_2\} & \{\mathrm{m}_1,\ m_2,\ \chi_{1z}\} & \dots \\ \{\mathrm{m}_2\} & \{\mathrm{m}_1,\ \chi_{1z}\} & \{\mathrm{m}_1,\ m_2,\ q\} & \dots \\ \vdots & \vdots & \vdots & \end{array}$$

- **Several good choices:**

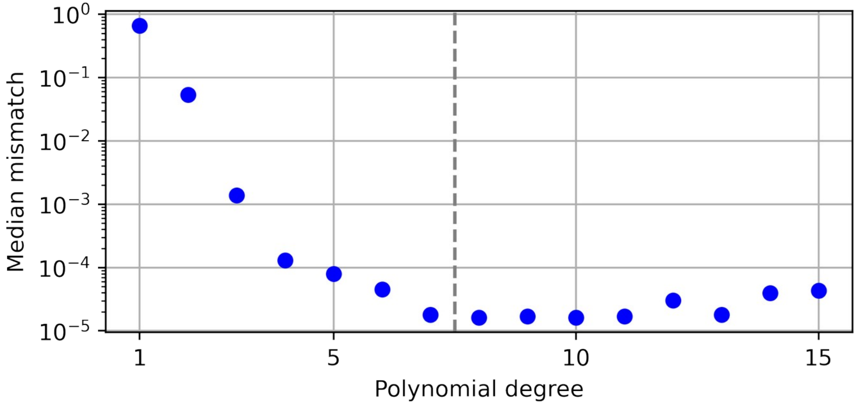
$$\begin{array}{cc} \{\chi_{2z},\ \chi_{\rm eff},\ \mathcal{M}\} & \{\mathrm{q}, \chi_{2z},\ \chi_{\rm eff},\ \mathcal{M}\} \\ \{\mathrm{q}, \mathrm{m}_2,\ \chi_{2z},\ \chi_{\rm eff}\} & \{\chi_{2z},\ \chi_{\rm eff},\ m_1,\ {m_2}^{-1}\} \end{array}$$

Selected

Polynomial degree

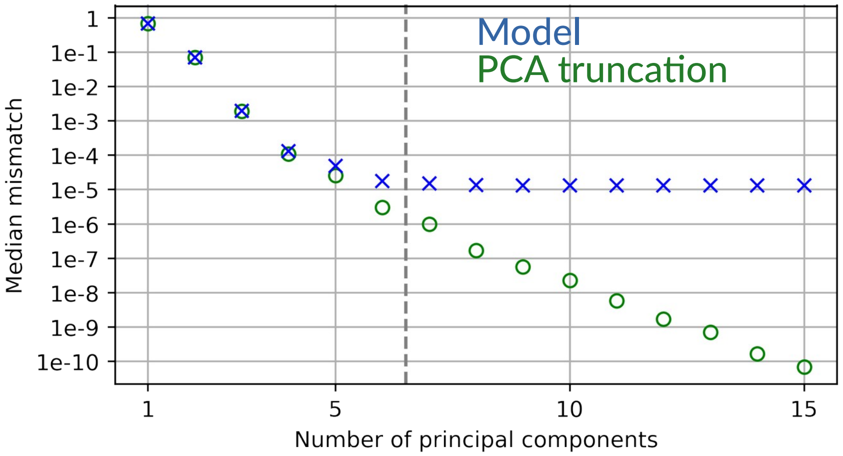
First order	$\{\mathrm{a}, \mathrm{b}\}$
Second order	$\{\mathrm{a}, \mathrm{b}, \mathrm{ab}, \mathrm{a}^2,\ b^2\}$
Third order	$\{\mathrm{a}, \mathrm{b}, \mathrm{ab}, \mathrm{a}^2,\ b^2,\ a^2b,\ ab^2,\ a^3,\ b^3\}$
	$\vdots$

Seventh order used for regression

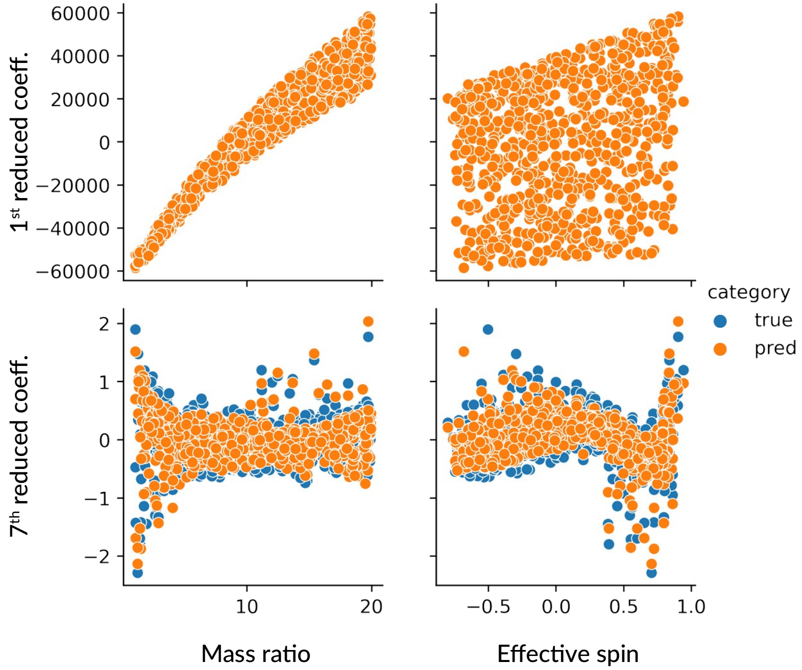


Hyperparameters tuning

Number of PC for the **phase**



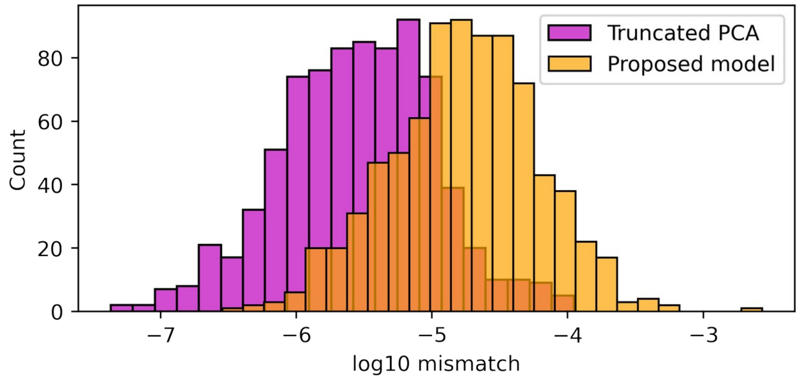
Six PC used for dimension reduction



Results on SEOBNRv4

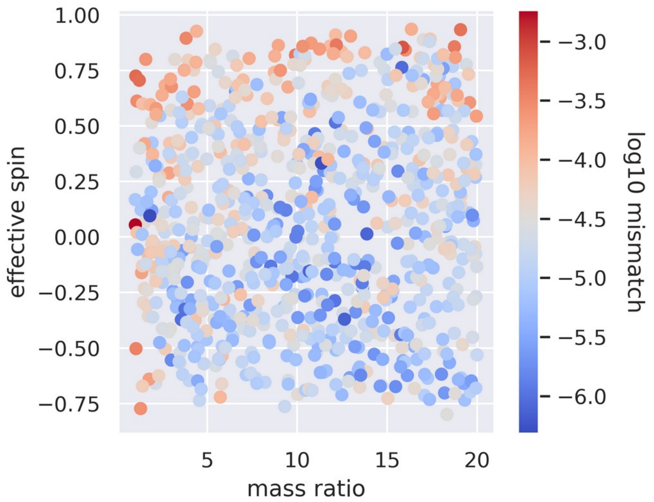
Dataset properties

- 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

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

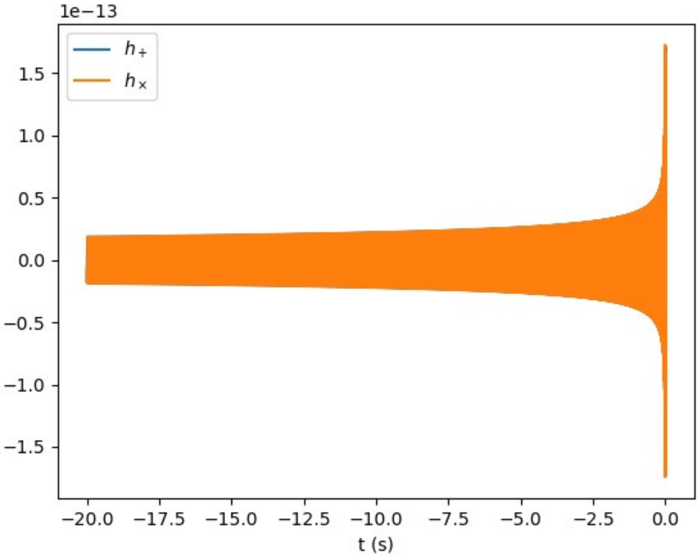
In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import mlpgw

%matplotlib notebook
```

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

In [3]:
```python
# Make prediction
h_pred = model.predict(m1=15, m2=5, s1z=0.9, s2z=0.2)
```

In [4]:
```python
# Plot it
plt.figure()
plt.plot(h_pred['time'], h_pred['hp'], label=r'$h_+$')
plt.plot(h_pred['time'], h_pred['hc'], label=r'$h_\times$')
plt.xlabel('t (s)')
plt.legend()
plt.show()
```



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

Conclusion/perspectives

Take home messages :

- **Fast** and **accurate** GW generation with **principal component regression**
- **Applicability up to SNR ~ 225** (18 in the worst case) *: $\text{mismatch} < \frac{N}{2\text{SNR}^2}$
- **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^2 scores

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

PC	1	2	3	4	5	6
a	1.67e-06	0.00231	0.0214	0.00728	1.42	0.177
Φ	1.65e-09	9.44e-07	0.000248	0.00322	0.00401	0.0326