Regression methods in waveform modeling: a comparative study

Michael Pürrer
AEI Potsdam-Golm

UNC, October 15, 2020

PAPER • OPEN ACCESS

Regression methods in waveform modeling: a comparative study

Yoshinta Setyawati[1,2], Michael Pürrer[3] and Frank Ohme[1,2]

Published 3 March 2020 • © 2020 The Author(s). Published by IOP Publishing Ltd
Classical and Quantum Gravity, Volume 37, Number 7

https://arxiv.org/abs/1909.10986

MAX-PLANCK-GESELLSCHAFT

Structure of this talk

- Introduction to building models of GWs from compact binaries
- Regression methods
- Setup of this study
- Results
- Conclusion

Introduction to building models of GWs from compact binaries

What do we need to model?

- A **waveform model** is a **parametrized function** of the waveform **polarizations** $h_{+,\times}(t;\,\overrightarrow{\lambda})$ or complex **modes** $h_{lm}(t;\,\overrightarrow{\lambda})$

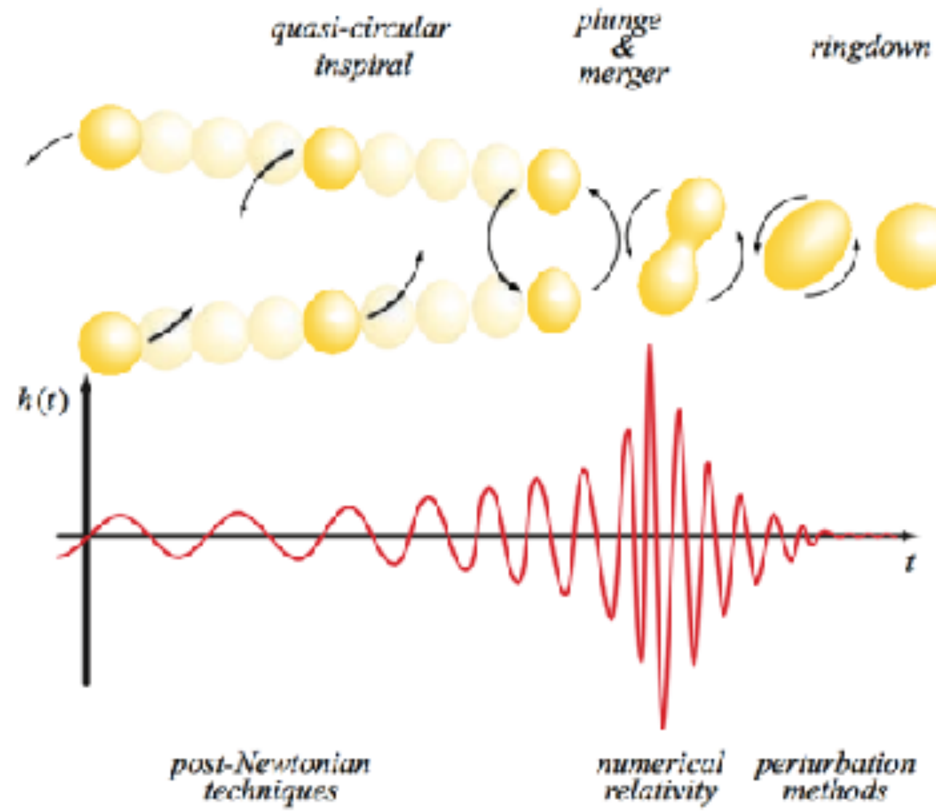
$$h_+ - ih_\times = \sum_{l,m} h_{lm}(t;\,\overrightarrow{\lambda})\,{}^{-2}Y_{lm}(\theta,\phi)$$

- Need to model the **inspiral, merger and ringdown** stages in binary black hole coalescence.

- GW detectors record **GW strain**:

$$h(t;\,\overrightarrow{\theta}) = h_+(t;\,\overrightarrow{\lambda})F_+(\hat{n},\psi) + h_\times(t;\,\overrightarrow{\lambda})F_\times(\hat{n},\psi)$$

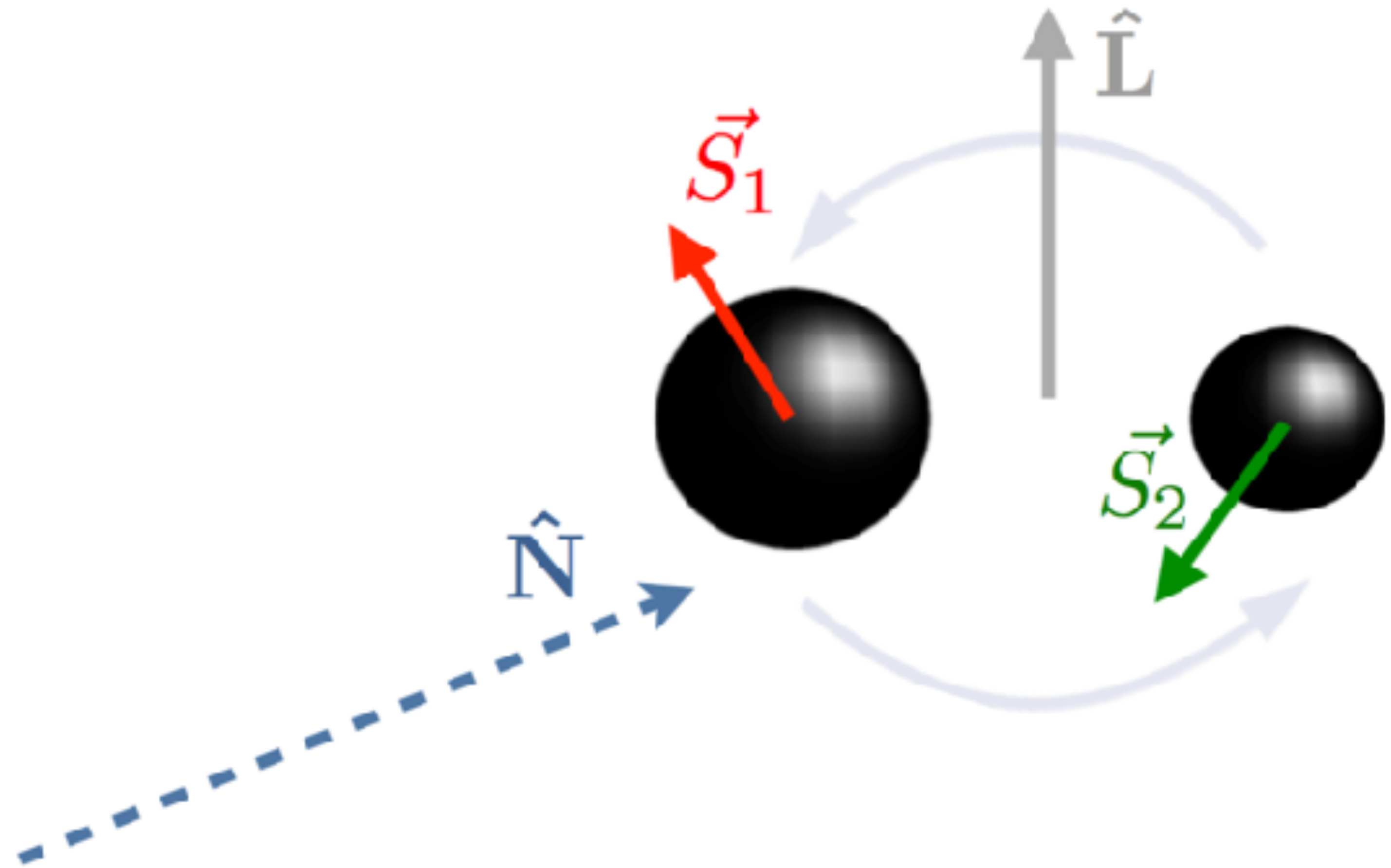
BBH coalescence

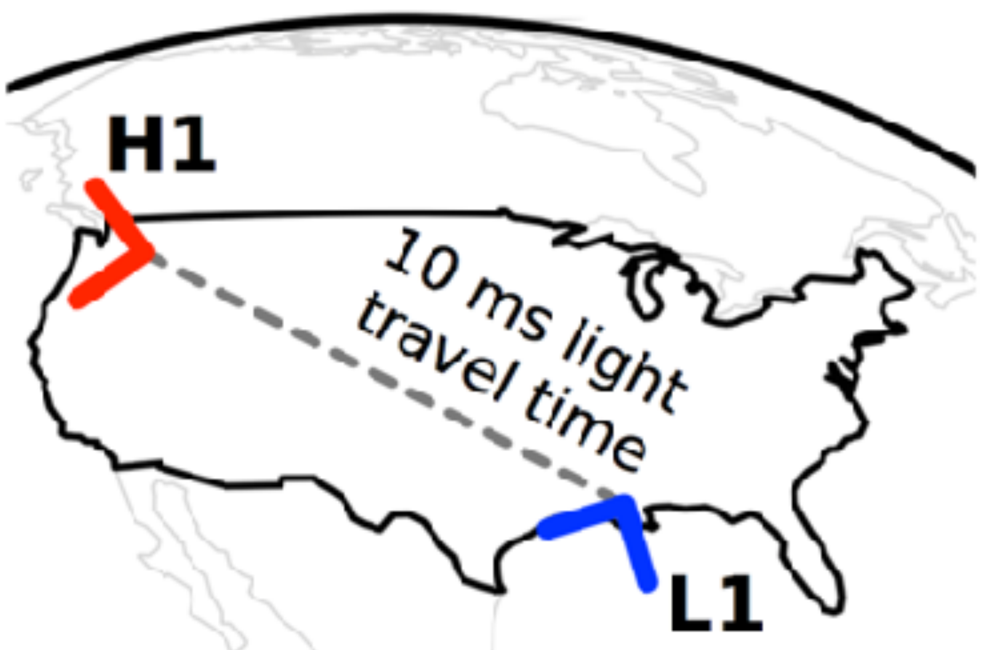


[Baumgarte & Shapiro, Numerical Relativity]

Model parameters

Intrinsic parameters:
masses, spins, eccentricity, tidal deformability





Extrinsic parameters:
time, sky position, distance, orientation, reference phase

Credit: LIGO/Virgo

Types of waveform models

- Analytic:
  - **post-Newtonian**
  - (uncalibrated) **Effective-one body** (EOB) models
- **Numerical relativity** (NR):
  - Numerical solution of Einstein's equations for binary systems
- "Data driven":
  - Semi-analytical models combining analytical information with tuning to numerical relativity (NR) simulations
    - **Phenom(enological)** models
    - **EOBNR** models
  - **Surrogate / reduced order models**
    - Model directly NR or a semi-analytical model

Surrogate / reduced order models

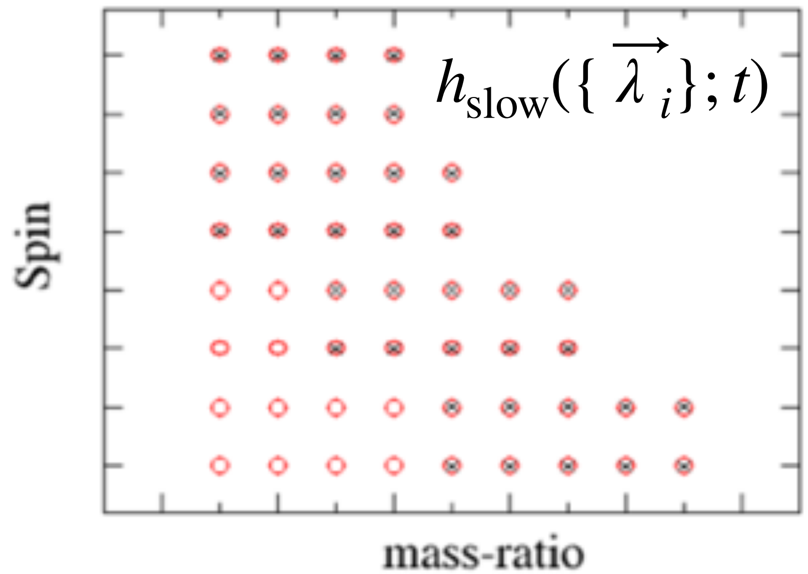
- Start with a ***slow*** **model** (e.g. EOB, NR) $h_{\rm slow}(\overrightarrow{\lambda}; t)$

Surrogate / reduced order models

- Start with a ***slow* model** (e.g. EOB, NR) $h_{\mathrm{slow}}(\vec{\lambda}; t)$
- Build ***fast* and *accurate* surrogate model or ROM**:
  - **surrogate**: *substitute, proxy, replacement*
  - **reduced order model**: *truncated orthonormal basis expansion*

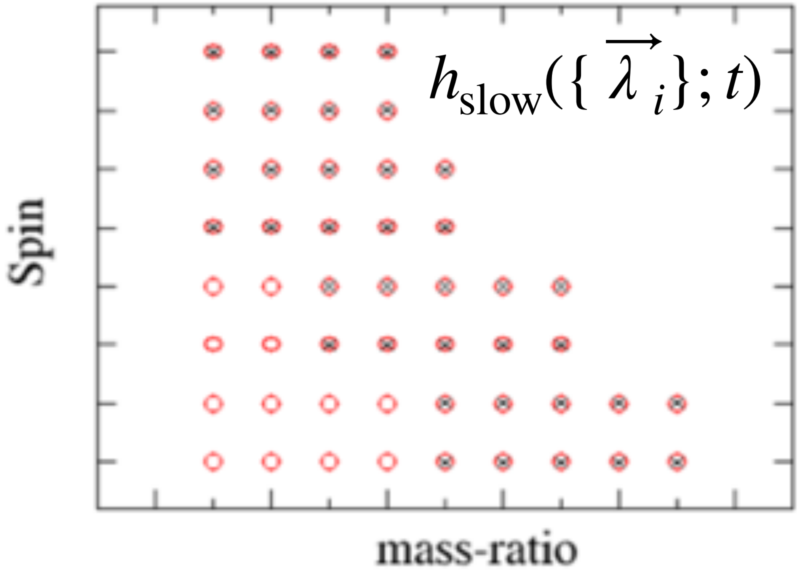
Surrogate / reduced order models

- Start with a ***slow*** **model** (e.g. EOB, NR) $h_{\text{slow}}(\vec{\lambda}; t)$
- Build ***fast*** **and** ***accurate*** **surrogate model or ROM**:
  - Build **training set**:from ***slow*** **model** $h_{\text{slow}}(\{\vec{\lambda}_i\}; t)$



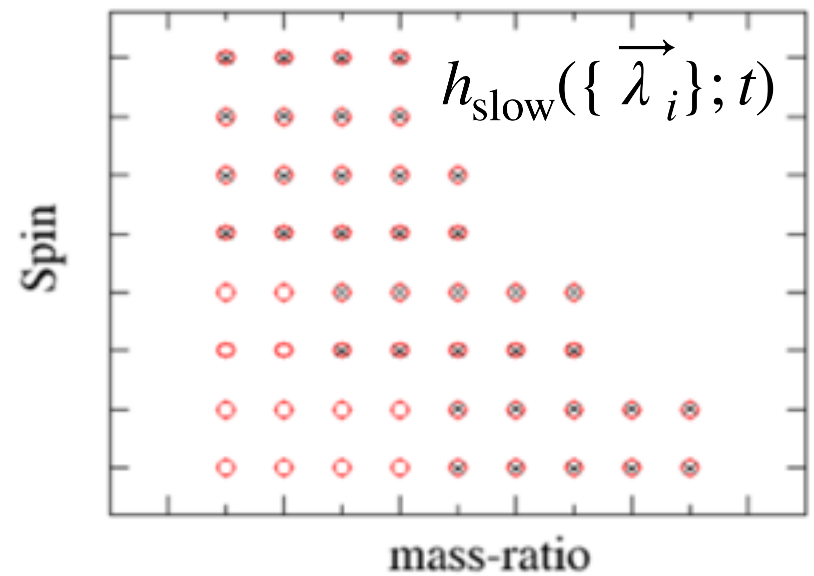
Surrogate / reduced order models

- Start with a ***slow* model** (e.g. EOB, NR) $h_{\rm slow}(\vec{\lambda};t)$
- Build ***fast* and *accurate* surrogate model or ROM**:
  - Build **training set**:from ***slow* model** $h_{\rm slow}(\{\vec{\lambda}_i\};t)$
  - **Decompose** waveform into data pieces and orthonormal bases

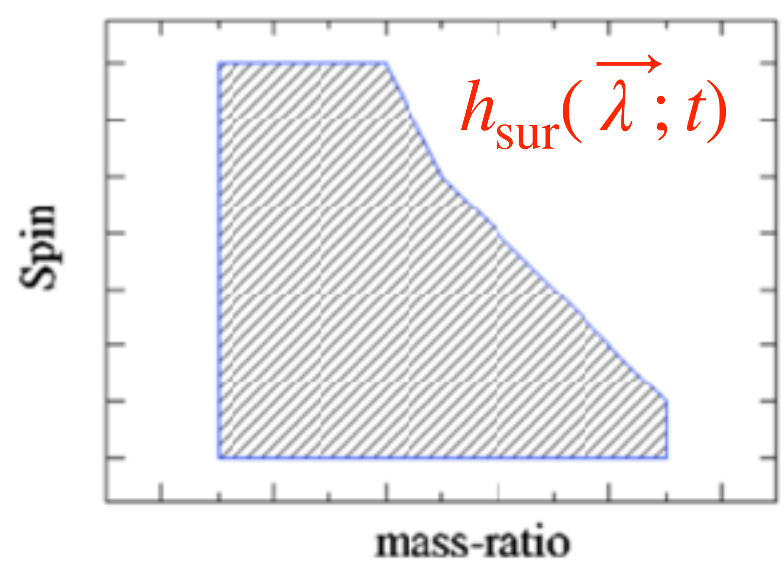


Surrogate / reduced order models

- Start with a ***slow* model** (e.g. EOB, NR) $h_{\mathrm{slow}}(\vec{\lambda};t)$
- Build ***fast* and *accurate* surrogate model or ROM**:
  - Build **training set**:from ***slow* model** $h_{\mathrm{slow}}(\{\vec{\lambda}_i\};t)$
  - **Decompose** waveform into data pieces and orthonormal bases
  - **Interpolate** coefficients over parameter space

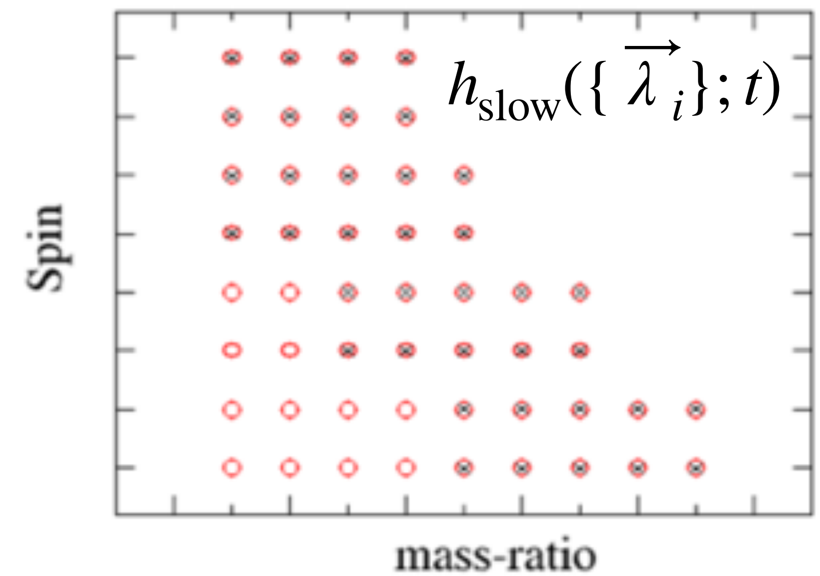


Decompose & **Interpolate**



Surrogate / reduced order models

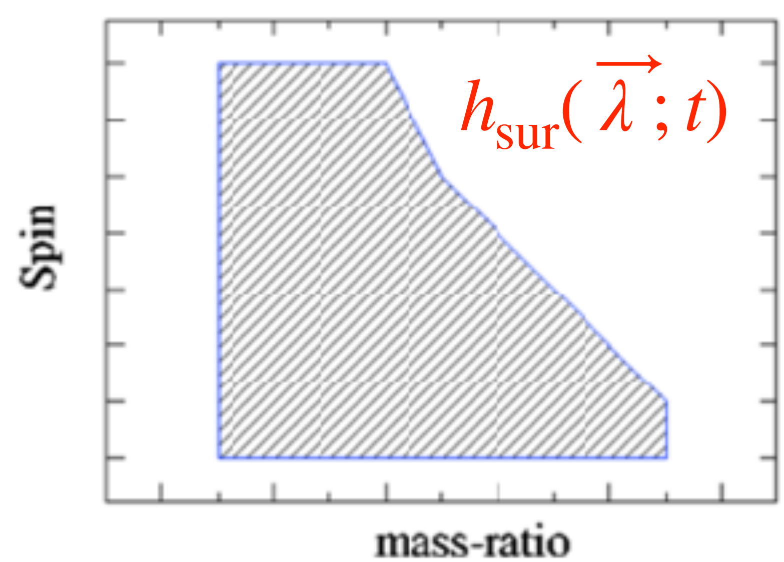
- Start with a ***slow* model** (e.g. EOB, NR) $h_{\mathrm{slow}}(\vec{\lambda};t)$
- Build ***fast* and *accurate* surrogate model or ROM**:
  - Build **training set**:from ***slow* model** $h_{\mathrm{slow}}(\{\vec{\lambda}_i\};t)$
  - **Decompose** waveform into data pieces and orthonormal bases
  - **Interpolate** coefficients over parameter space
  - **Validate** the surrogate

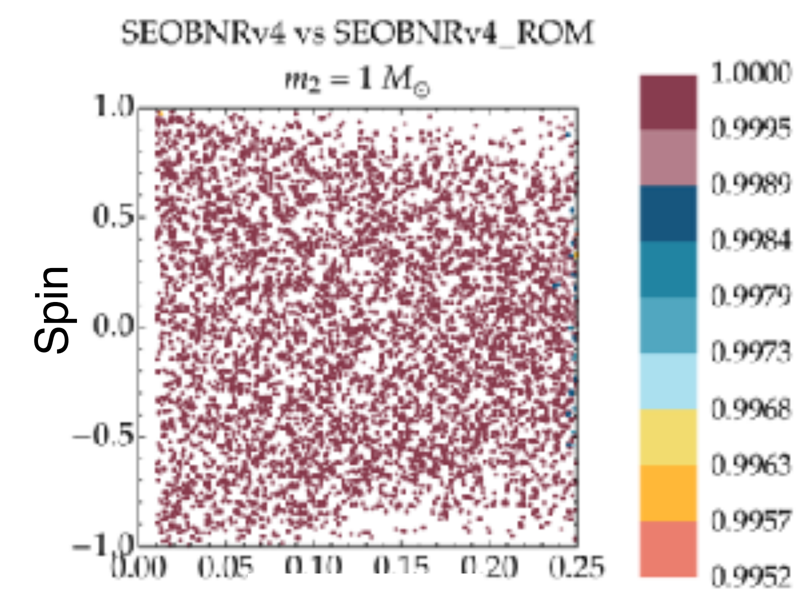


Decompose & **Interpolate**

Validate

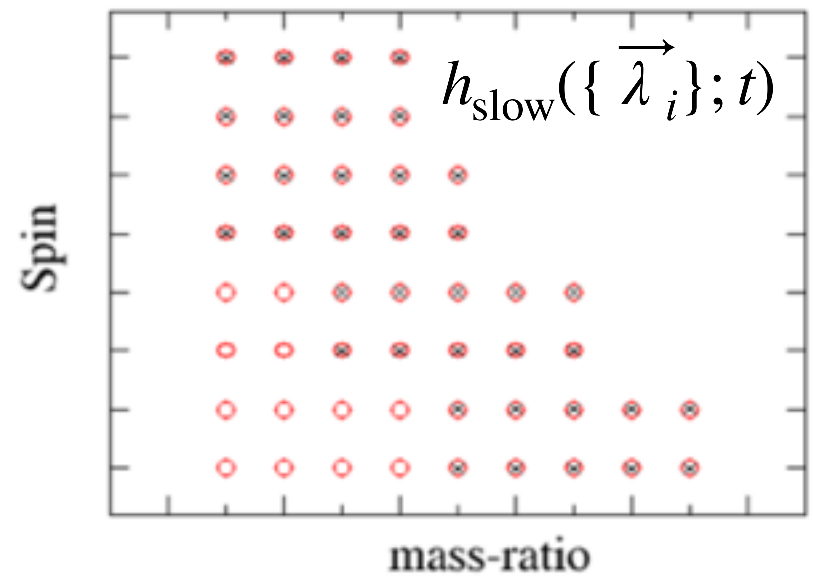
$\langle \hat{h}_{\mathrm{slow}}(\vec{\lambda};t) | \hat{h}_{\mathrm{sur}}(\vec{\lambda};t) \rangle$





Surrogate / reduced order models

- Start with a ***slow* model** (e.g. EOB, NR) $h_{\rm slow}(\vec{\lambda};t)$
- Build ***fast* and *accurate* surrogate model or ROM**:
  - Build **training set**:from ***slow* model** $h_{\rm slow}(\{\vec{\lambda}_i\};t)$
  - **Decompose** waveform into data pieces and orthonormal bases
  - **Interpolate** coefficients over parameter space
  - **Validate** the surrogate
  - **Speedup** ~ O(1000)

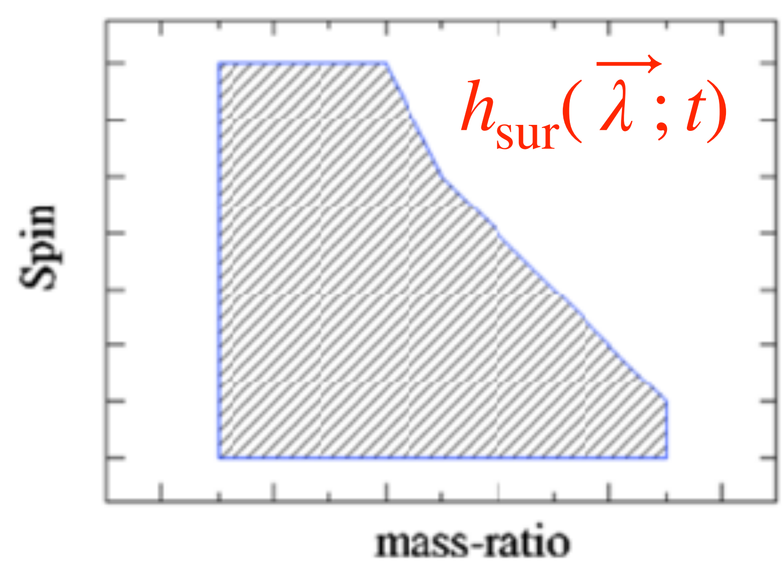


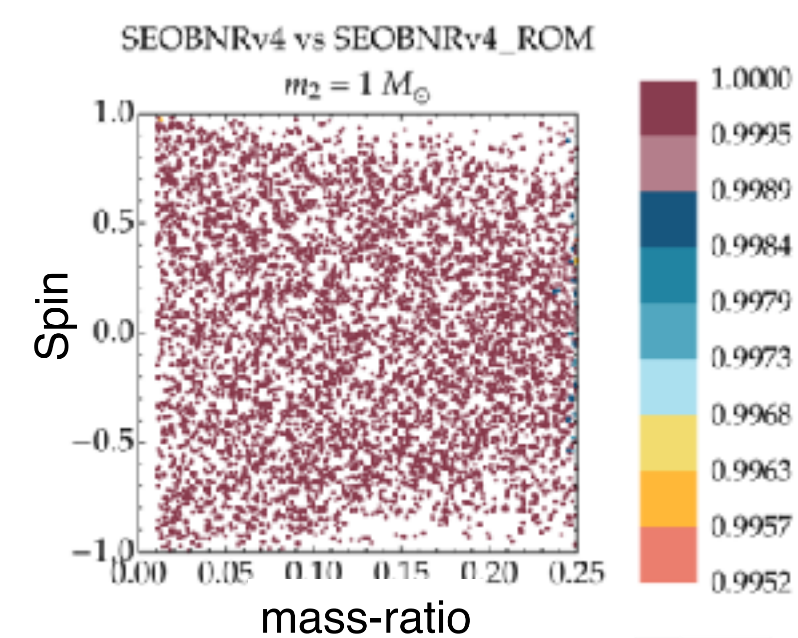
Decompose &

Interpolate

Validate

$\langle \hat{h}_{\rm slow}(\vec{\lambda};t) | \hat{h}_{\rm sur}(\vec{\lambda};t) \rangle$





[Field+13, MP 14, 15, Bohé+…MP 17, Lackey+16, Doctor+17 (MP), Lackey+…MP 19, Blackman+15,+17,+17, Varma+19,+19, Cotesta+ 20]

Reduced order / surrogate models

- Usually **decompose waveform data pieces in a reduced orthonormal basis**
  - **SVD or greedy basis** construction
  - **Work with coefficients**
  - **Empirical interpolation method**: can transform basis such that coefficients are waveform data piece at a certain time or frequency

$$I_N[X](t;\vec{\lambda}) = \sum_{i=1}^{N} c_i(\vec{\lambda}) e^i(t) = \sum_{j=1}^{N} X(T_j;\vec{\lambda}) b^j(t).$$

Reduced order / surrogate models

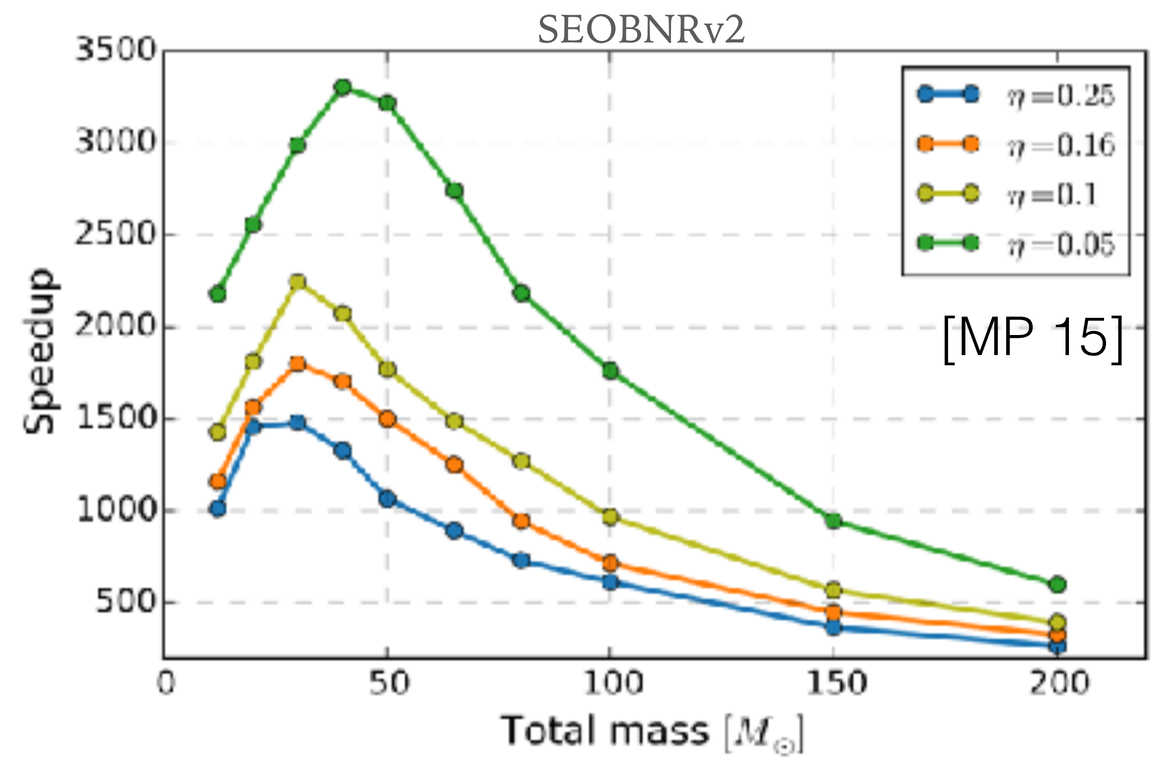
- Usually **decompose waveform data pieces in a reduced orthonormal basis**
  - **SVD or greedy basis** construction
  - **Work with coefficients**
  - **Empirical interpolation method**: can transform basis such that coefficients are waveform data piece at a certain time or frequency

- Use **sophisticated regression methods**:
  - **Greedy step-wise forward** polynomial fits
  - **Tensor-product spline interpolation** (regular grid)
  - **Gaussian process regression**

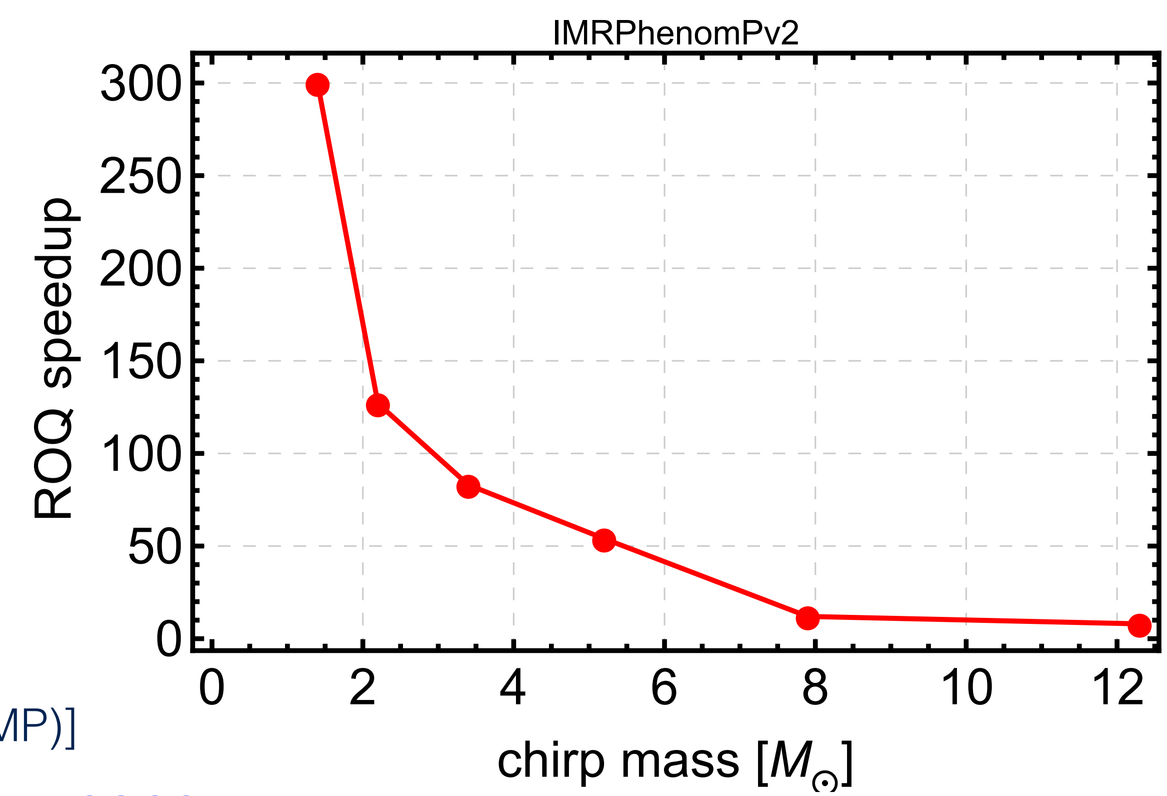
[Field+13, MP 14, 15, Bohé+…MP 17, Lackey+16, Doctor+17 (MP) Lackey+…MP 19, Blackman+15,+17,+17, Varma+19,+19, Cotesta+20 (MP)]

Reduced order / surrogate models

- Usually **decompose waveform data pieces in a reduced orthonormal basis**
  - **SVD or greedy basis** construction
  - **Work with coefficients**
  - **Empirical interpolation method**: can transform basis such that coefficients are waveform data piece at a certain time or frequency



- Use **sophisticated regression methods**:
  - **Greedy step-wise forward** polynomial fits
  - **Tensor-product spline interpolation** (regular grid)
  - **Gaussian process regression**



[Field+13, MP 14, 15, Bohé+…MP 17, Lackey+16, Doctor+17 (MP)
Lackey+…MP 19, Blackman+15,+17,+17, Varma+19,+19, Cotesta+20 (MP)]

Regression methods

PAPER • OPEN ACCESS

Regression methods in waveform modeling: a comparative study

Yoshinta Setyawati[1,2], Michael Pürrer[3] and Frank Ohme[1,2]

Published 3 March 2020 •

Classical and Quantum Gravity, Volume 37, Number 7

https://arxiv.org/abs/1909.10986

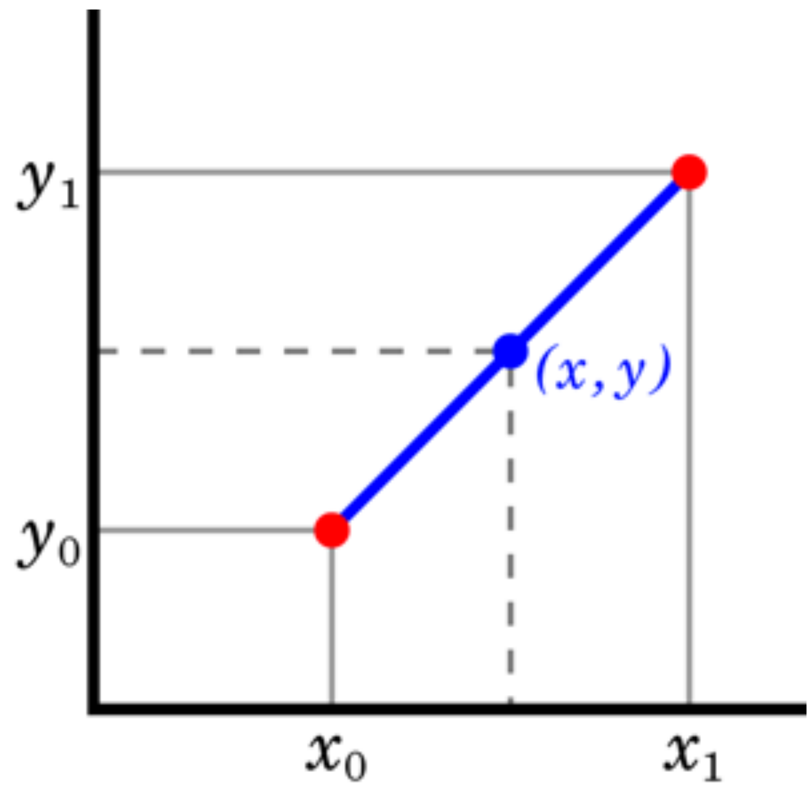
Overview of regression methods

- Polynomial interpolation & polynomial fits
  - Linear interpolation
  - Tensor product interpolation (TPI)
  - Greedy multivariate polynomial fit (GMVP)
- Radial basis functions (RBF)
- Gaussian process regression (GPR)
- Artificial neural networks (ANNs)

Linear interpolation

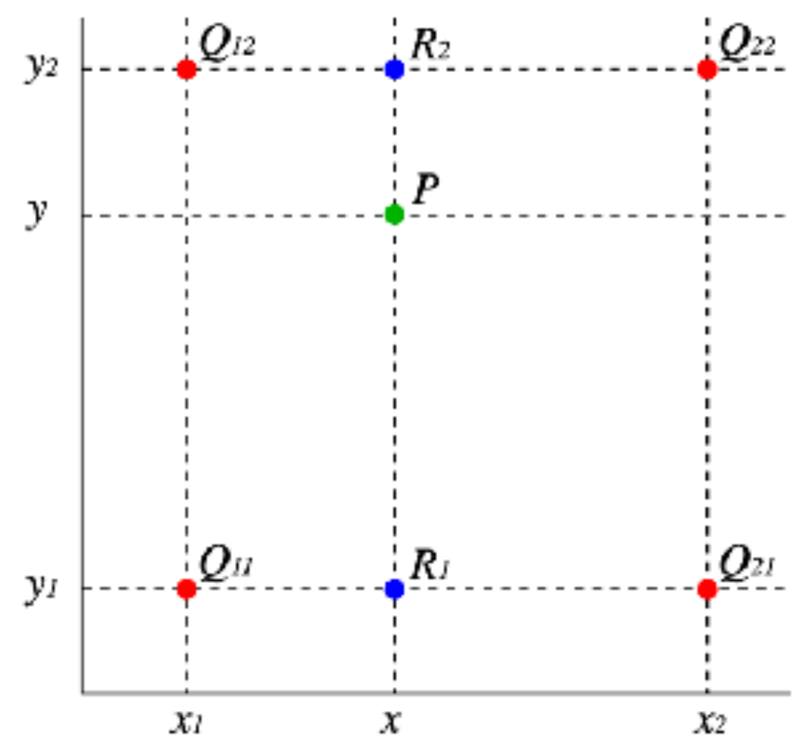
- Multivariate linear interpolation on a *regular grid* using the regular grid interpolator (RGI) in scipy

1D



$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

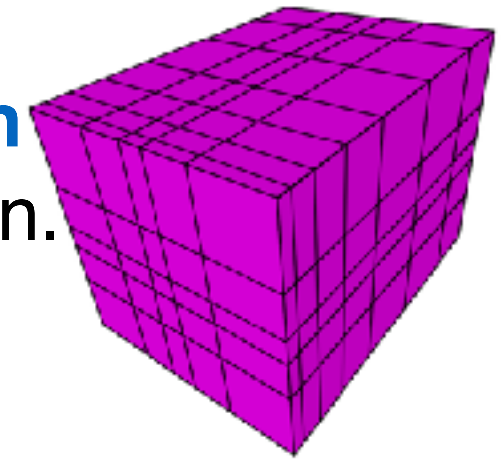
2D



1) Interpolate in x-direction: -> R_1, R_2
2) Interpolate in y-direction: -> P

 [Image credits: wikipedia.org]

Tensor product interpolation (TPI)



A (rectilinear) regular grid [credit: wikipedia.org}

- On a ***regular* grid** can use the same ***univariate* interpolation method** (e.g. splines, spectral interpolation) in each dimension.

$$I[X](t_i; \vec{\lambda}) = \sum_{j_1,\ldots,j_d} a_{j_1,\ldots,j_d} \left(\Psi_{j_1} \otimes \ldots \otimes \Psi_{j_d}\right)(\vec{\lambda}).$$

- **Splines** are piecewise polynomials with continuity conditions.
  - Can be written in terms of B-spline basis functions of degree k with a “knot vector” t (connection points of polynomials)
  - Solve a linear system for the spline coefficients $s = \sum_i s_i B_{i,k,t}(x)$

- If data is smooth **Chebyshev interpolation** is a good option.

- We use **TPI cubic splines** provided by the Cython package at https://github.com/mpuerrer/TPI

Polynomial fits

- A **multiple linear regression model** where the independent variables form a **polynomial**
- Example in 1D:
  - Assume we are given data points $(x_i, y_i)_{i=1}^N$
  - **Polynomial ansatz:** $f(\vec{x}) = c_0 x^k + c_1 x^{k-1} + \cdots + c_{k-1} x + c_k$
  - If ansatz has as many d.o.f. as data points, can solve linear system

$$\begin{pmatrix} x_1^k & x_1^{k-1} & \cdots & x_1 & 1 \\ x_2^k & x_2^{k-1} & \cdots & x_2 & 1 \\ \vdots & & \ddots & & \vdots \\ x_N^k & x_N^{k-1} & \cdots & x_N & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_k \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

  - In general, the linear system may be over- or under-determined such that no unique solution would exist
  - Use discrete **least squares fit** to minimize the error $\Sigma_{j=1}^N |f(x_j) - y_j|^2$
- We use the fitting function polyfitnd() from https://bitbucket.org/chadgalley/rompy

Greedy multivariate polynomial fit (GMVP)

- Procedure (London & Fauchon-Jones, CQG 36, 2019):
  - Assume we have data at $\vec{x}_j = \{x_j^1, x_j^2, \ldots, x_j^d\}$
  - **Ansatz** $f(\vec{x}) \approx \sum_k \mu_k \, \phi_k(\vec{x})$
  - Basis functions $\phi_k(\vec{x})$ are chosen to be **multivariate polynomials** of maximal degree D
  - Select terms from set $\phi_k(\vec{x}) \in \left\{ (x^1)^{\alpha_1} \, (x^2)^{\alpha_2} \ldots (x^n)^{\alpha_d}, \quad \sum_{i=1}^{n} \alpha_i \leq D \right\}$
  - Find least-squares solution
  - Iteratively add terms using a **greedy algorithm** to minimize the error $$\epsilon^2 = \frac{\sum_j \left[f(\vec{x}_j) - \sum_k \mu_k \, \phi_k(\vec{x}_j) \right]^2}{\sum_j \left[f(\vec{x}_j) \right]^2}$$
- A similar method has been used in the construction of NR-surrogate models Blackman+15,+17,+17, Varma+19,+19

Radial basis functions (RBF)

- RBF Method (we use scipy.interpolate.Rbf())
  - Assume we have data $\{(\vec{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R} | i = 1, \ldots, N\}$
  - Make **ansatz** that depends only on **Euclidean distance**

$$s(\vec{x}) = \sum_{i=1}^{N} w_i \varphi(r) \qquad r = \|\vec{x} - \vec{x_i}\|$$

  - Need to solve linear system $\Phi(r)\vec{w} = \vec{Y}$

  - Choose **radial basis function (or kernel)**

    Need to ensure that matrix $\Phi(r)$ is *non-singular*

    Common choice: **multiquadric kernel function**

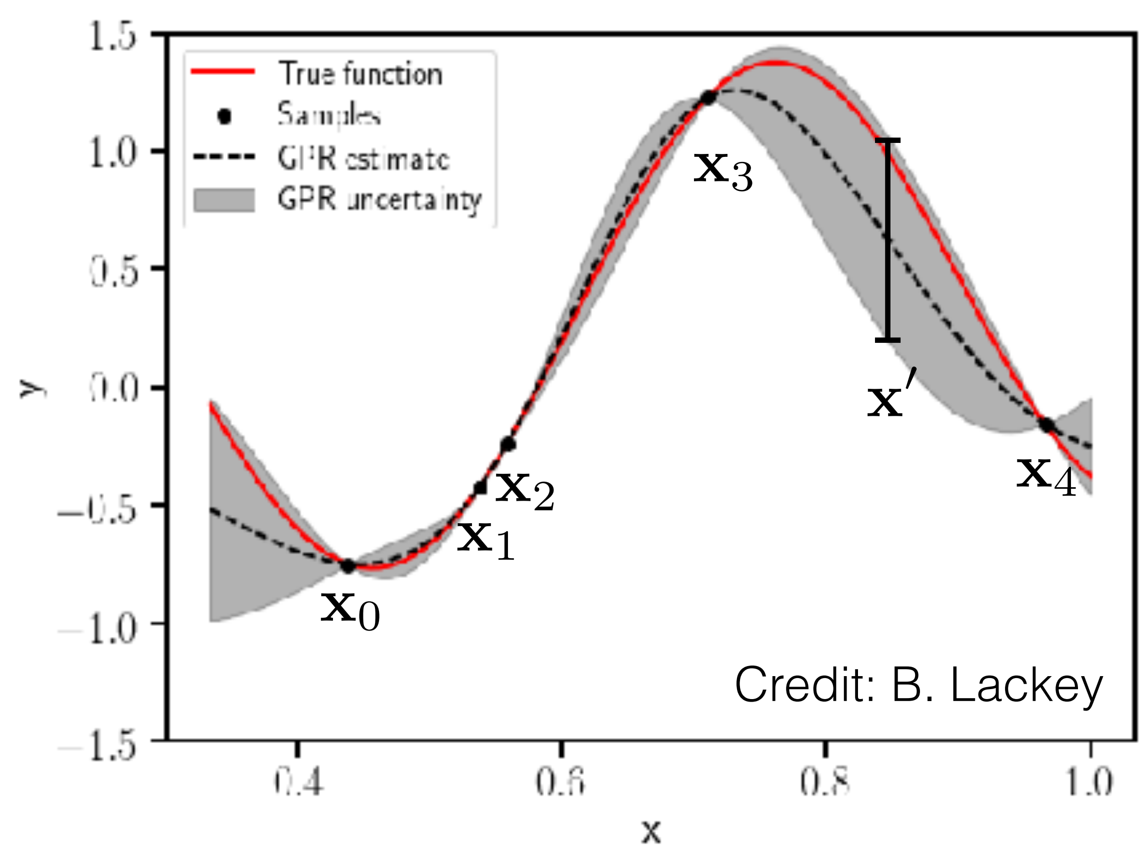
$$\varphi(r) = \sqrt{1 + \left(\frac{r}{\varepsilon}\right)^2}$$

Gaussian process regression (GPR)

- GPR assumes the **values of a function** y_i at the points $\mathbf{x}_i$ are **random variables that follow a multivariate Gaussian**:
  $p(y_i|\mathbf{x}_i) = \mathcal{N}(0, k(\mathbf{x}_i, \mathbf{x}_j))$
- **Covariance function** between two points $k(\mathbf{x}_i, \mathbf{x}_j)$
- Can estimate the functional value y' at $\mathbf{x}'$ given data $(\mathbf{x}_i, y_i)$
- **Mean:** $E[p(y'|\mathbf{x}_i, \mathbf{x}', y_i)]$
- **Uncertainty:** $\mathrm{Var}[p(y'|\mathbf{x}_i, \mathbf{x}', y_i)]$
- **Example kernels:**

$$k_{SE}(r) = \exp\left(\frac{-r^2}{\ell^2}\right)$$

$$k_M(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}r}{\ell}\right)$$



Gaussian process regression (GPR)

- Define a **Gaussian process for zero mean**

$$y(\vec{x}) \sim GP\Big(\mu(\vec{x}) = 0, k(\vec{x}, \vec{x}')\Big)$$

- Can write vector of training and test outputs as

$$\begin{bmatrix} \vec{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

- Find **conditional probability** $p(y_*|\vec{x}_i, \vec{x}_*, \vec{y}, \vec{\theta}) = \mathcal{N}(\bar{y}_*, \mathrm{var}(y_*))$

$$\bar{y}_* = K(X_*, X)(K(X,X))^{-1}_{ij} y_j$$

$$\mathrm{var}(y_*) = K(X_*, X_*) - K(X_*, X_i)(K(X,X))^{-1}_{ij} K(X_*, X_j)$$

$$K(x_i, x_j) = \sigma_f^2 k(x_i, x_j) + \sigma_n^2 \delta_{ij}$$

- **Optimize over hyper-parameters** $\sigma_f, \sigma_n, \{\ell_i\}_{i=1}^n$ by maximizing the *marginal log-likelihood*

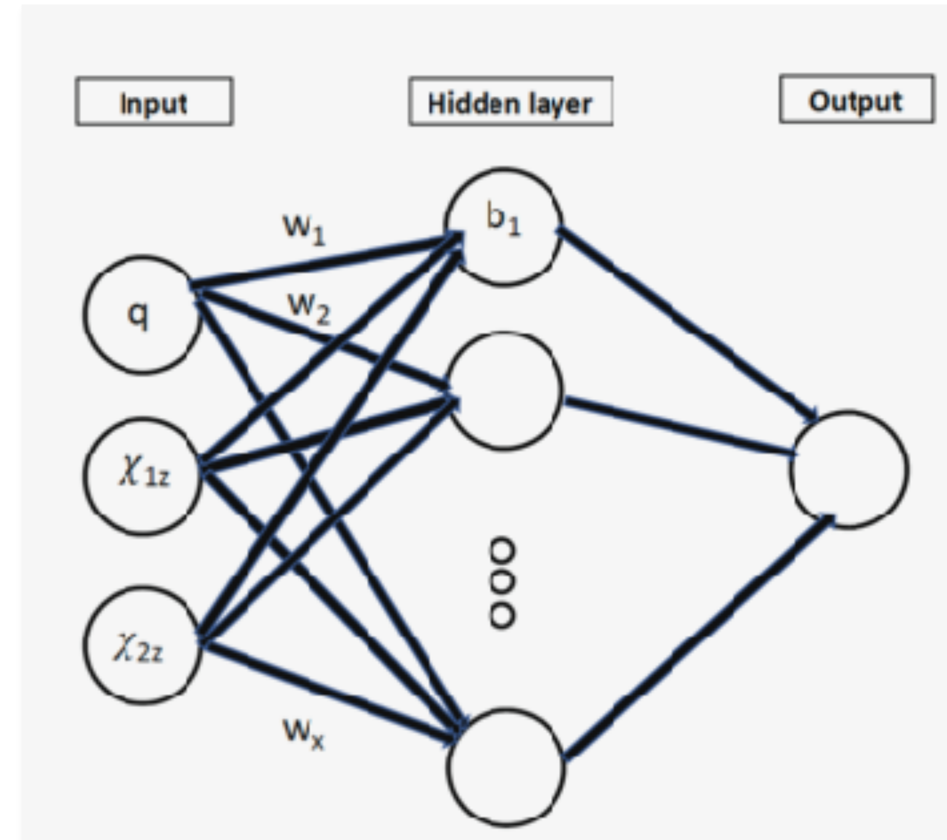
$$\ln p(y_i|\vec{x}_i, \vec{\theta}) = -\frac{1}{2}\Big(y_i (K(X,X))^{-1}_{ij} y_j + \ln|K(X,X)| + N \ln 2\pi\Big)$$

Artificial neural networks (ANNs)

- **ANNs are a machine learning algorithm** used e.g. for pattern recognition, classification.

- We use a **multi-layer-perceptron** with 2 hidden layers between input and output
  - Each layer acts as an *affine transformation* on its inputs

- **Activation functions** are used to introduced nonlinearity
  - We use the Rectified Linear Unit (ReLU) $\sigma(z) = \max(z, 0)$



A single-layer ANN for 3d interpolation

$$x_{i+1,k} = \sigma\left(\sum_j w_{ijk} x_{ij} + b_{ik}\right)$$

activation function

weights

input

bias

i: layer index
j: neuron index

Overview of regression methods

Methods	Advantages	Disadvantages	Training time
Linear (RGI)	standard `scipy`	needs regular grid	$\mathcal{O}(N)$
TPI	robust and high accuracy	needs regular grid	$\mathcal{O}(N^k)$
GMVP	irregular grid fast execution time	complex	#basis function #error tolerance
Polynomial fit	irregular grid simple and fast	Runge's phenomenon only univariate in `scipy`	$\mathcal{O}(N)$ and #polynomial degree
RBF	`scipy` irregular grid	high computational complexity	$\mathcal{O}(N^3)$
GPR	irregular grid can predict uncertainty	depends on the choice of kernel and hyperparameters complex	$\mathcal{O}(N^3)$
ANN	irregular grid flexible architecture choices	complex	#neurons #hidden layers

Setup of this study

Key waveform quantities

- We transform the *inertial* frame waveform modes to the minimally rotating **co-precessing frame** and align the waveforms at a particular time (t = -2000 M)
- We further define a frame that follows the orbital motion and compute the modes in this **co-orbital frame**.
- In this study we don't build a full waveform model, instead we just use **two key quantities** at selected times

$$\phi(t) := \frac{1}{4}\left(\arg\left[\bar{\bar{h}}^{2,-2}_{\mathrm{copr}}(t)\right] - \arg\left[\bar{\bar{h}}^{2,2}_{\mathrm{copr}}(t)\right]\right)$$

$$A(t) := \mathrm{Re}\,\bar{\bar{h}}^{2,2}_{+} = \frac{1}{2}\mathrm{Re}\left(\bar{\bar{h}}^{2,2}_{\mathrm{coorb}}(t) + \bar{\bar{h}}^{2,-2^*}_{\mathrm{coorb}}(t)\right)$$

Key waveform quantities: example

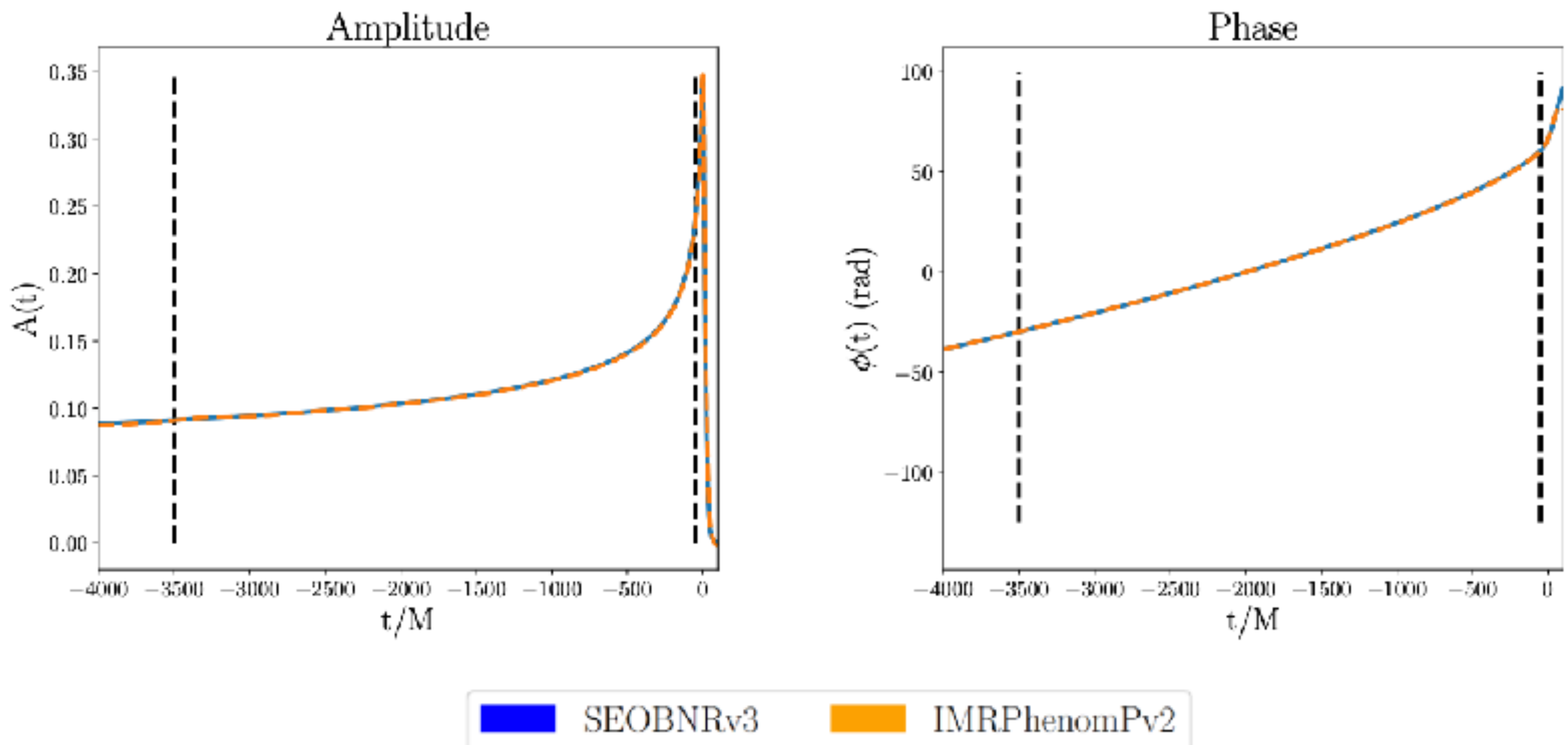


Figure 1. The key quantities of the GW signal of a precessing BBH, here illustrated for a binary with $(q, \chi_{1x}, \chi_{1y}, \chi_{1z}, \chi_{2x}, \chi_{2y}, \chi_{2z}) = (1.99, 0.51, 0.04, 0.03, 0.01, 0.6, 0.1)$. Left: the dimensionless amplitude $A(t)$. Right: the phase $\phi(t)$ (in unit radian). The black dashed lines show the points in time-space, where we perform different interpolation methods (t=-3500M and t=-50M).

Waveforms and data sets

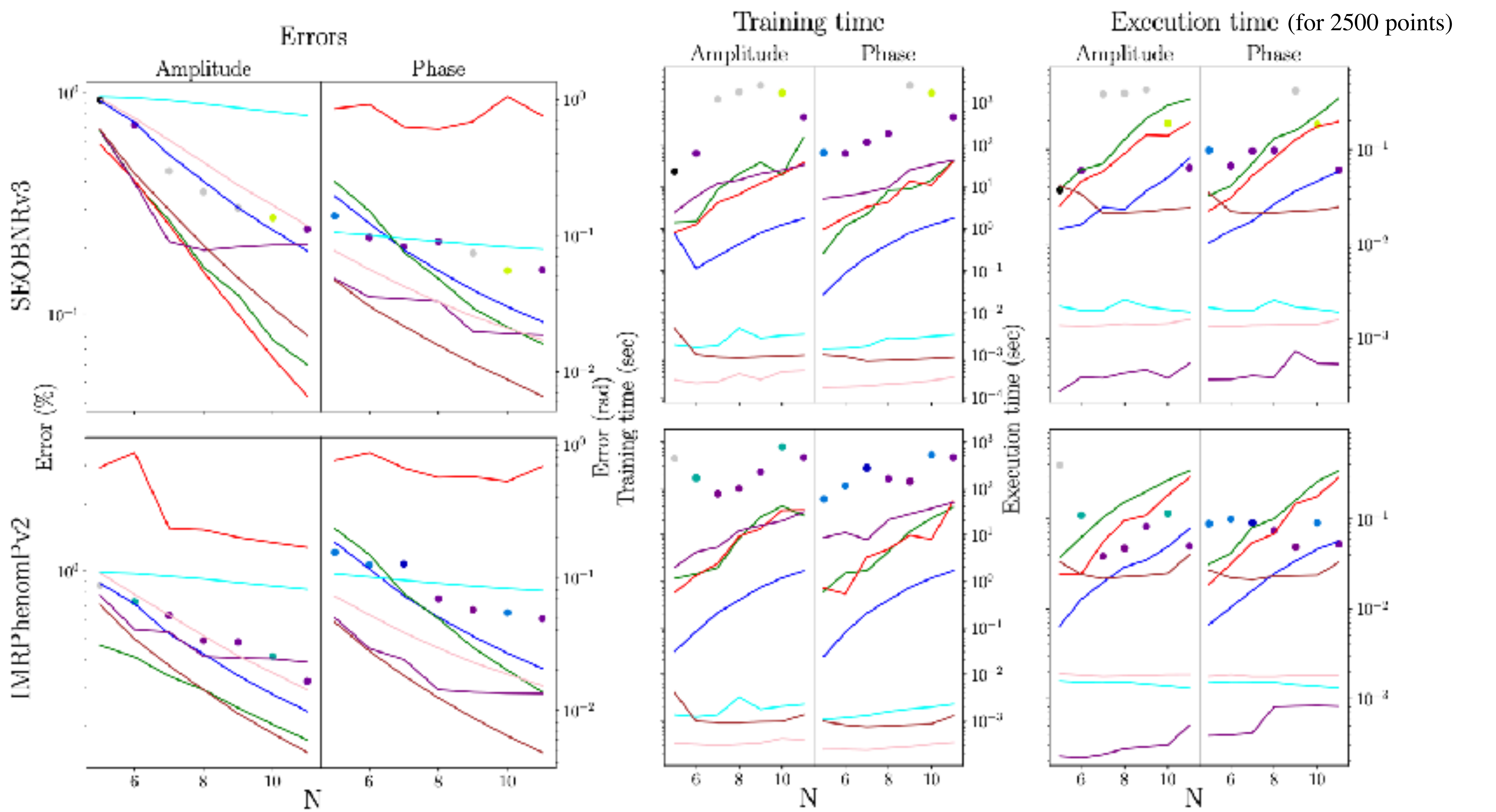
- Our choice of using waveform quantities at fixed times is motivated by the **EIM modeling framework**.

- We use **two inspiral-merger-ringdown waveform models** that describe GWs emitted from precessing black hole binaries:
  - **SEOBNRv3** (time domain waveform)
  - **IMRPhenomPv2** (iFFT of Fourier domain waveform)

- We consider these **data sets**:
  - Dimensionality**:** 3D (q, aligned spin) or 7D (q, generic spins)
  - Early and late times: -3500M or -50M before merger
  - Number of training set points: 3D (5 - 11 per dimension), 7D (total up to 3000); 2500 random test points
  - Physical domain: 3D ($q \leq 10, |\chi_i^z| \leq 1$), 7D ($q \leq 2, |\chi_i| \leq 1/\sqrt{3}$)

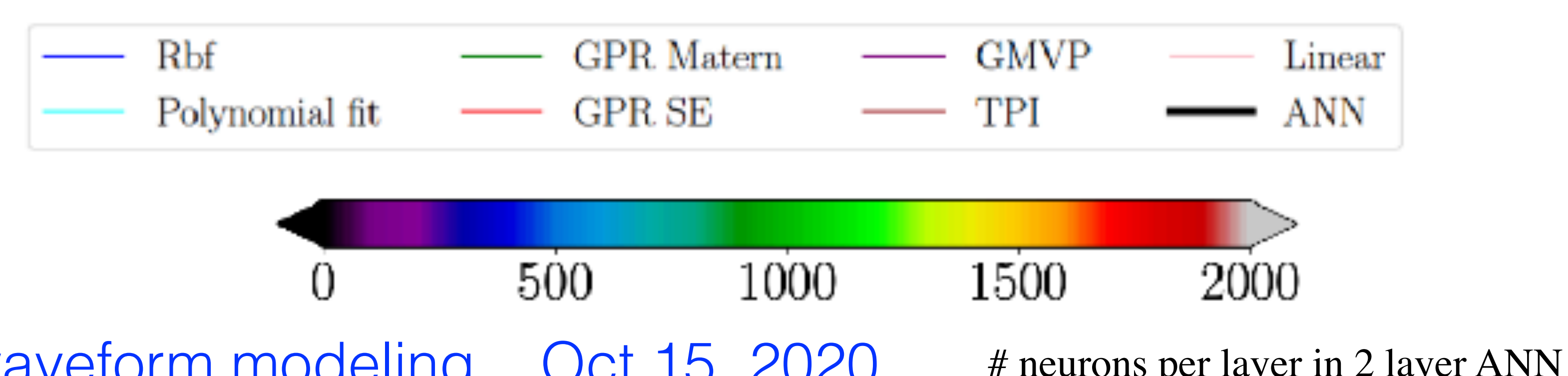
Results

3D interpolation results at t=-3500M



$$\varepsilon_{ae} = \frac{1}{N}\sum_{i}^{N} |\phi^{i}_{\mathrm{pred}}(t) - \phi^{i}_{\mathrm{true}}(t)|$$

$$\varepsilon_{re} = \frac{\sum_i^N |A^i_{\mathrm{pred}}(t) - A^i_{\mathrm{true}}(t)|}{\sum_i^N |A^i_{\mathrm{true}}(t)|} \times 100$$



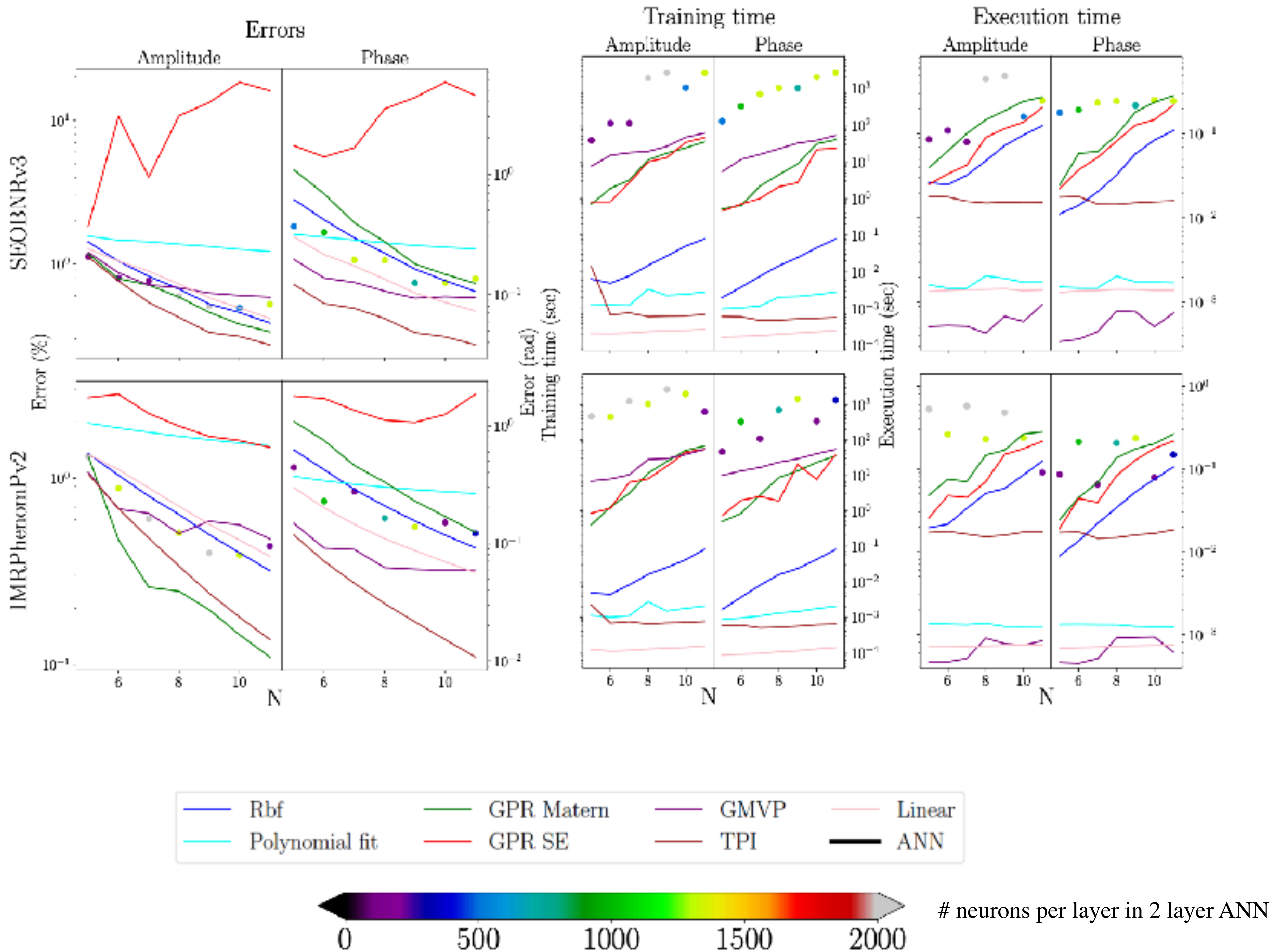
Observations for 3D results

- In general, **errors decrease with the size of the training set.**

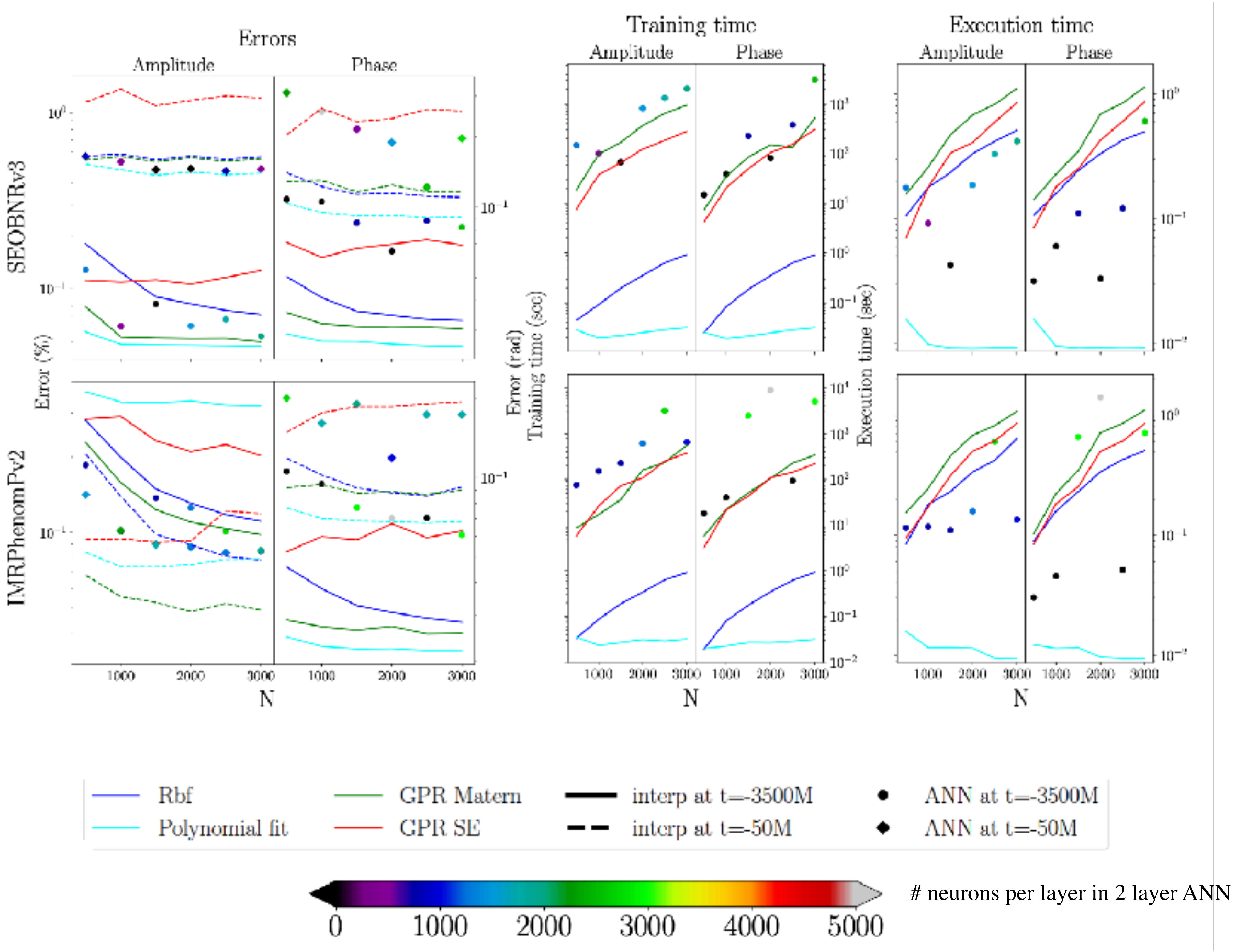
- **Errors are similar between the two waveform models**, but:
  - GPR *amplitude error* for IMRPhenomPv2 is much higher for SE kernel compared to Matérn
  - Noise in the IMRPhenomPv2 data due to the iFFT

- **Comments on the 2-layer ANNs**:
  - We report *minimum error* over # neurons in [2, 2000]
  - Too many neurons lead to *overfitting* (dropout could help) and too few neurons lead to *underfitting*

3D interpolation results at t=-50M



7D interpolation results



Observations for 7D results

- **Amplitude errors near merger / during inspiral:**
  - For SEOBNRv3, errors near merger are higher than in the inspiral.
  - For IMRPhenomPv2, surprisingly, amplitude errors are *smaller* near merger.

- **Phase errors:**
  - They are comparable for SEOBNRv3 and IMRPhenomPv2.

- **Execution time:**
  - for GPR and RBF it depends on the size of the TS,
  - in contrast to ANNs.

- **Comparing errors between 3D and 7D:**
  - For the *same parameter ranges*, errors in 7D can be up to 100 times larger for $A(t)$ and 15 times larger for $\phi(t)$.

Conclusions

- A few **take-aways** from the regression results

- 3D:
  - **TPI** is very accurate and fast in training and evaluation
  - **GMVP** is also doing very well here

- 7D:
  - **Polynomial fits** are fast and accurate
  - **GPRs with a Matérn kernel** have good accuracy (with SE kernel errors are much higher)
  - Note: GMVP not included because implementation was too slow.

- **Methods that have been used build GW surrogate models:**
  - **TPI** (up to 5D)
  - **Greedy polynomial fits** (slightly different from GMVP) in 7D
  - **GPR** (up to 5D)

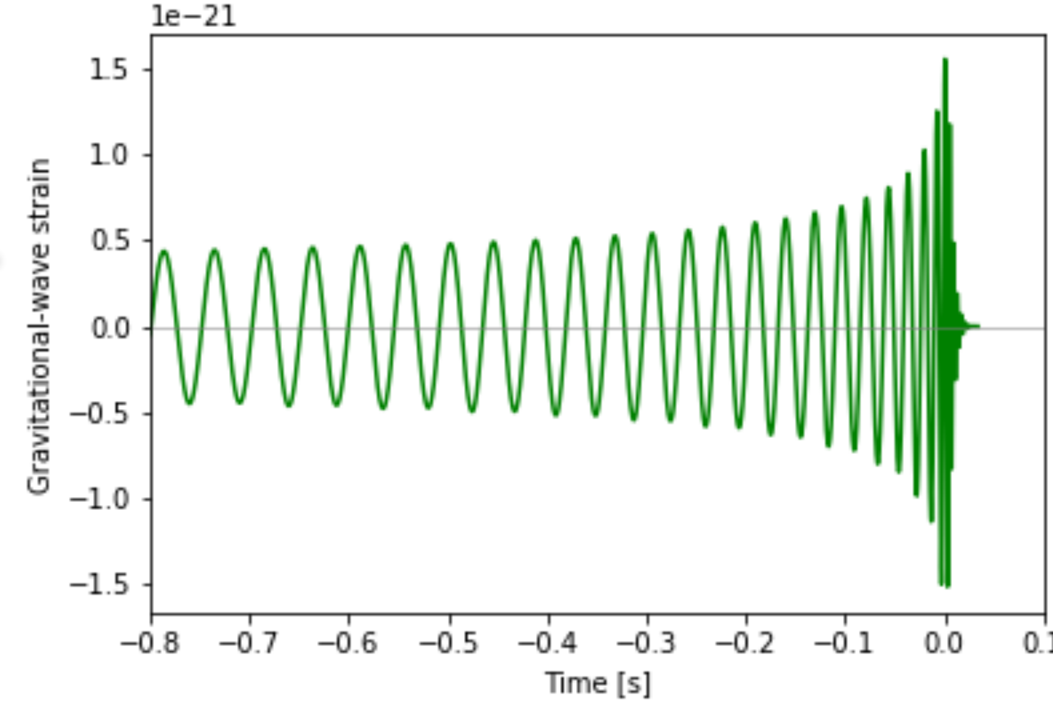
[MP 14, 15, Bohé+…MP 17, Doctor+17 (MP), Lackey+…MP 19, Blackman+15,+17,+17, Varma+19,+19, Cotesta+ 20]

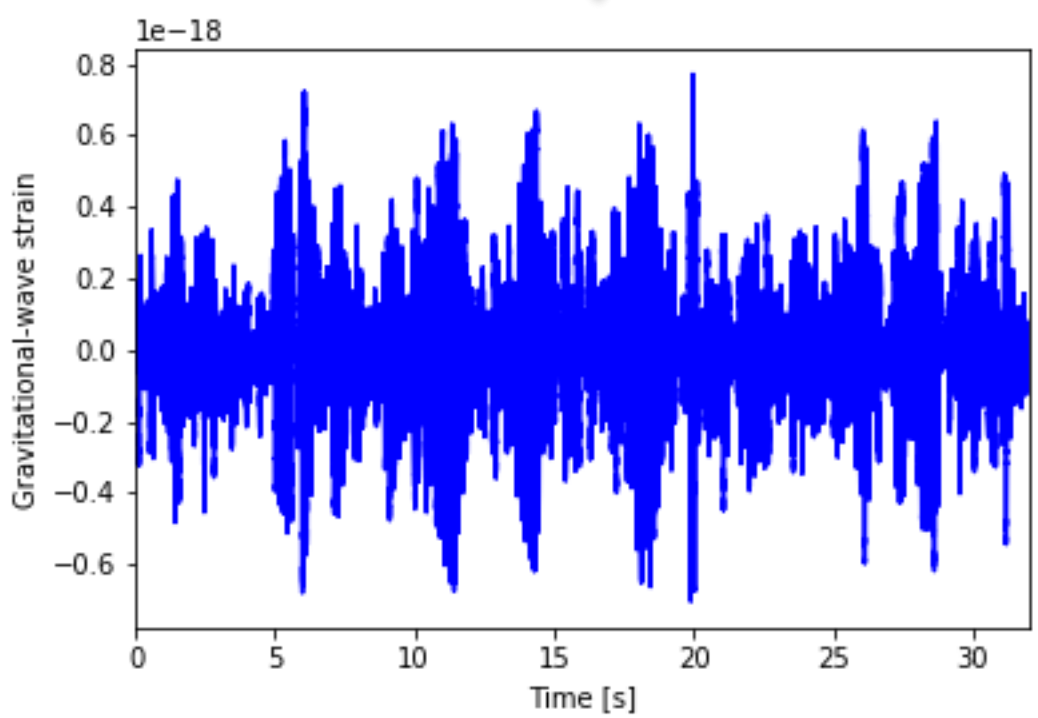
Thank you for your attention!

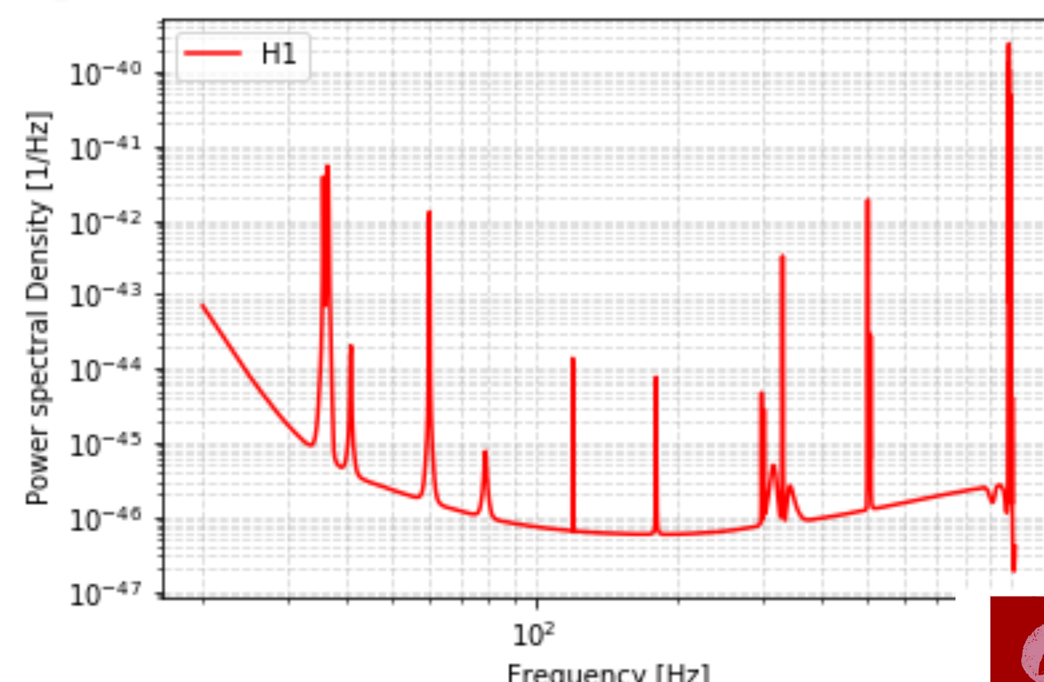
Matched filtering

Inner product

$$\langle d\,|\,h\rangle = 4\,\mathrm{Re}\int_0^\infty \frac{\tilde{d}(f)\tilde{h}^*(f)}{S_n(f)}\mathrm{d}f$$

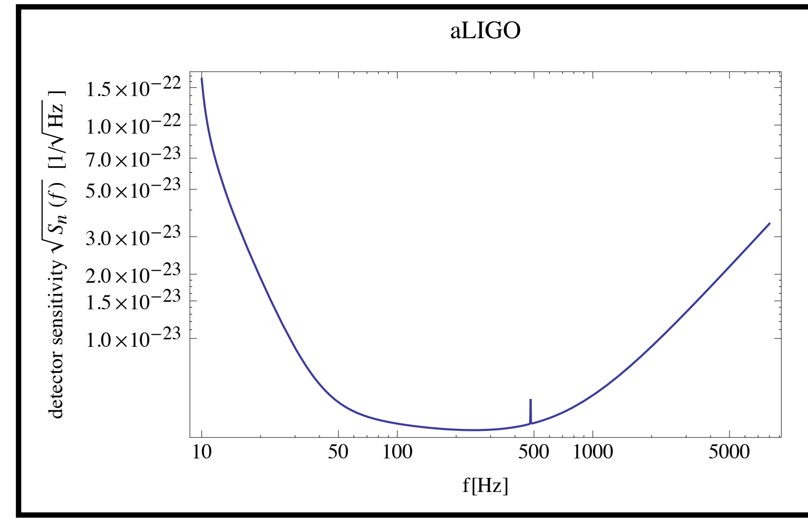






Checking accuracy of waveforms

$$\langle h_1, h_2 \rangle = 4\mathrm{Re}\int_{f_{\min}}^{f_{\max}} \frac{\tilde{h}_1(f)\tilde{h}_2(f)^*}{S_n(f)} df$$



SNR $\boxed{\rho = \|h\|}$

Match (overlap)

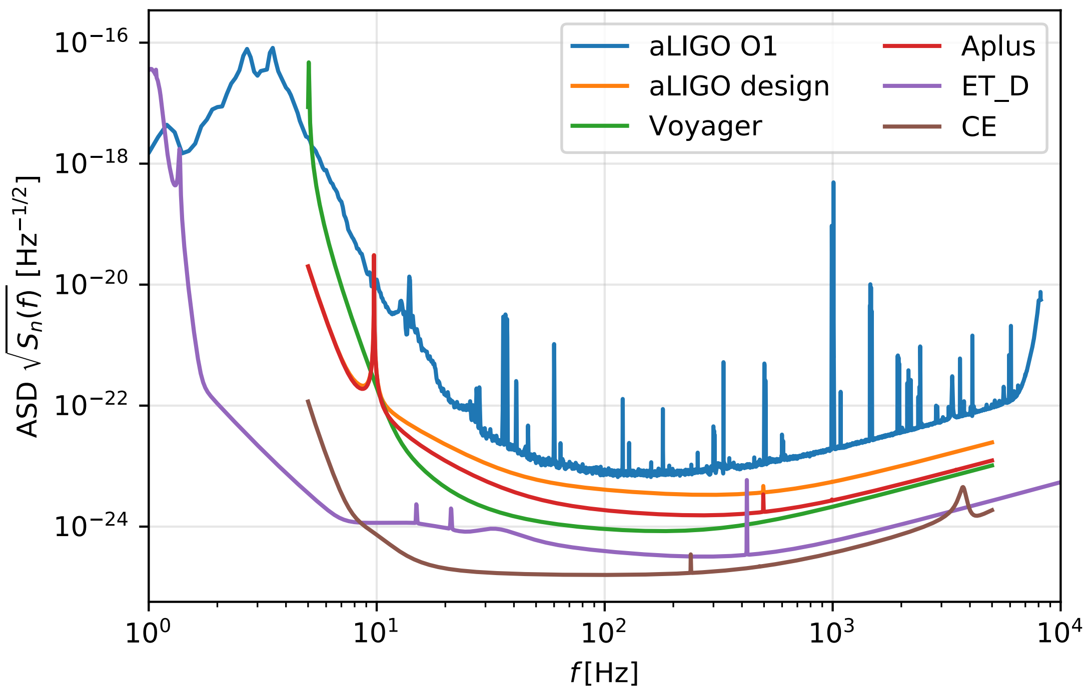
$$\boxed{\mathcal{O}(h_1, h_2) = \frac{4}{\|h_1\|\,\|h_2\|} \max_{t_0} \left| \mathcal{F}^{-1}\left[\frac{\tilde{h}_1(f)\tilde{h}_2(f)^*}{S_n(f)}\right](t_0)\right|}$$

Mismatch *(Faithfulness)* $\boxed{1 - \mathcal{O}(h_1, h_2)}$

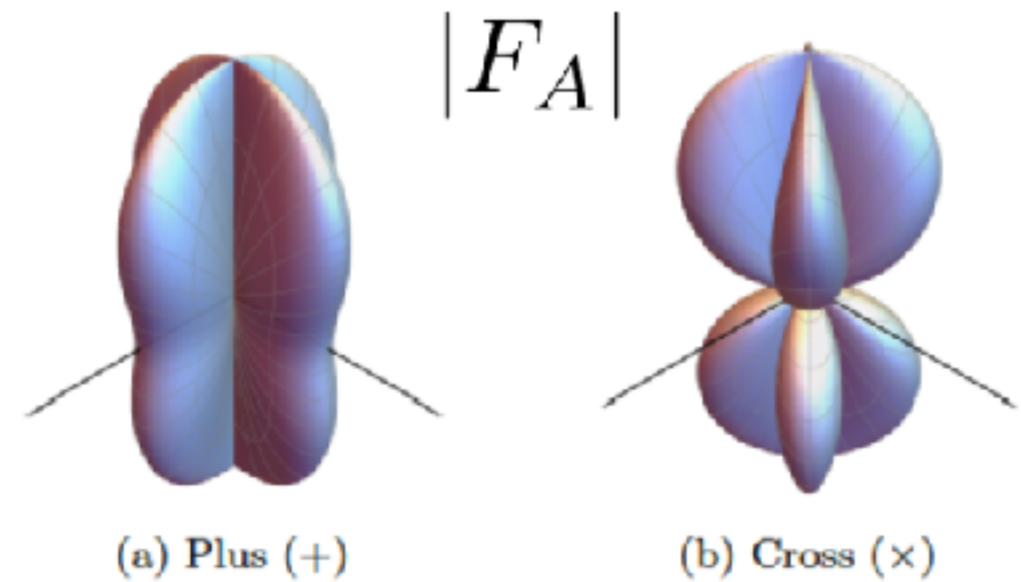
$$\mathcal{F}^{-1}[g(f)] = \int_{-\infty}^{\infty} g(f) e^{-2\pi i f t} df$$

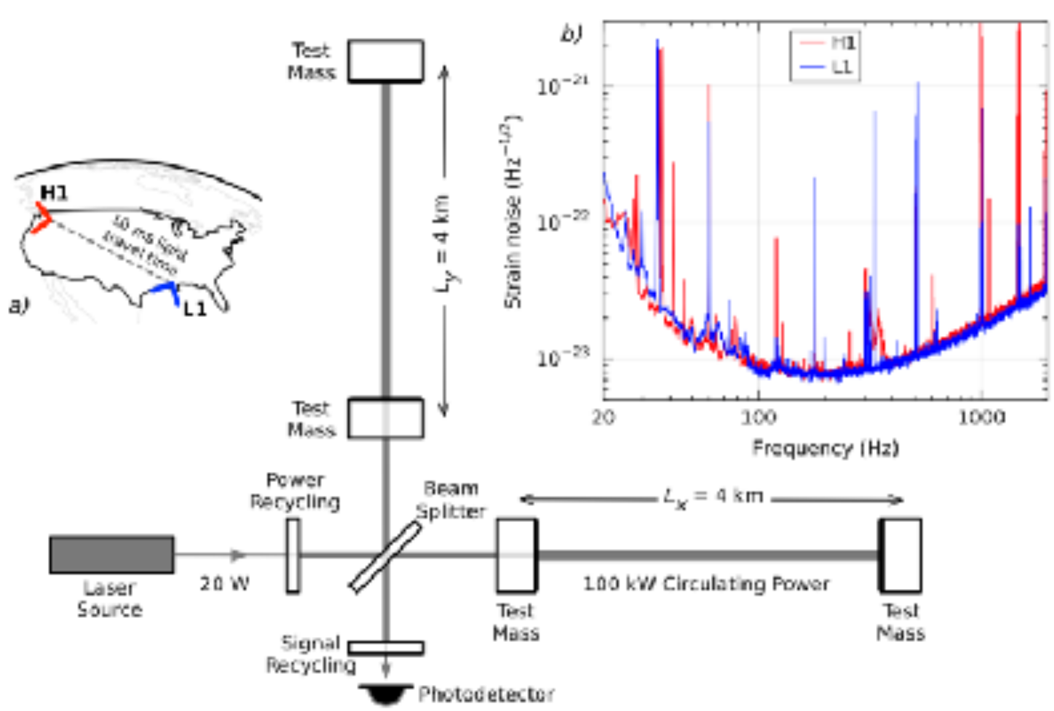
Interferometric GW detectors

Evolution of detector sensitivity



Angular response of IFO





$$h = h_A F^A \qquad F^A(\hat{n},\psi) = D^{ab} e^A_{ab}$$

$$A = +\,,\times\,,\ldots$$

$$D^{ab} = \frac{1}{2}\left(d^a_x d^b_x - d^a_y d^b_y\right)$$

Reduced Basis and Empirical interpolation method

greedy reduced basis

$$\tilde{h}(f;\vec{\lambda}) \approx \sum_{i=1}^{m} c_i(\vec{\lambda}) e_i(f) \qquad c_i(\vec{\lambda}) = \langle \tilde{h}(\cdot;\vec{\lambda}), e_i(\cdot) \rangle$$

- **Empirical interpolation** method finds “good” frequencies:

$$\sum_{i=1}^{m} c_i(\vec{\lambda}) e_i(F_j) \overset{!}{=} \tilde{h}(F_j;\vec{\lambda})$$

EI frequencies

- **EI interpolant**:

$$\tilde{h}(f;\vec{\lambda}) \approx \sum_{i=1}^{m} B_j(f) \tilde{h}(F_j;\vec{\lambda})$$

$$B_j(f) = \sum_{i=1}^{m} e_i(f)(V^{-1})_{ij}$$

$$V_{ij} = e_i(F_j)$$

- **Fit $\tilde{h}(F_i;\vec{\lambda})$ w.r.t. $\vec{\lambda}$ at each** F_i using only data at greedy points

[Field+13, Blackman+15,+17,+17, Varma+19,+19, Gadre+… MP in prep]