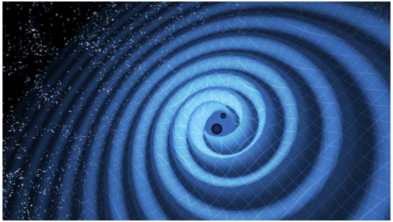
Reconstructing gravitational wave polarizations with bivariate signal processing and plug-and-play method

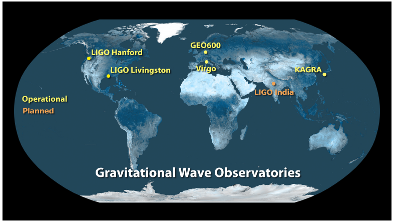
Pierre Palud[1], Eric Chassande-Mottin[1],
Y. Y. Pilavci[2], P. Chainais[2], P.-A. Thouvenin[2]

[1] APC, gravitation team, Paris
[2] Centrale Lille Institut, CRIStAL, Sigma team
ANR RICOCHET – https://ricochet-anr.github.io

Workshop on Bayesian neural networks for comsology & time domain astrophysics
– 21th May 2025 –

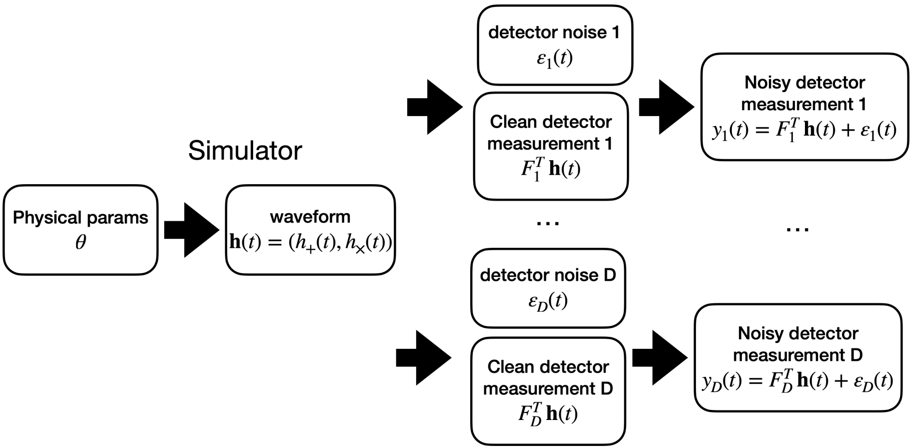
Gravitational waves





- space-time deformation propagating as a wave
- due to large mass acceleration
- predicted by Einstein from general relativity
- first observed in 2015 by LIGO-Virgo interferometers
- today: hundreds of observed events
- most common source = **merger of stellar black hole binaries**
- **bivariate signal**: contains two "polarizations" $h_+(t)$ and $h_\times(t)$

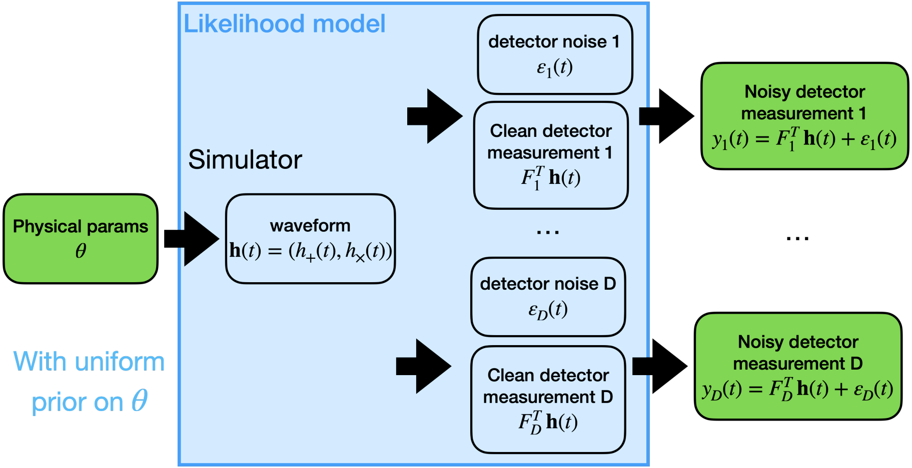
Data analysis focus in the LVK collaboration



Data analysis focus in the LVK collaboration

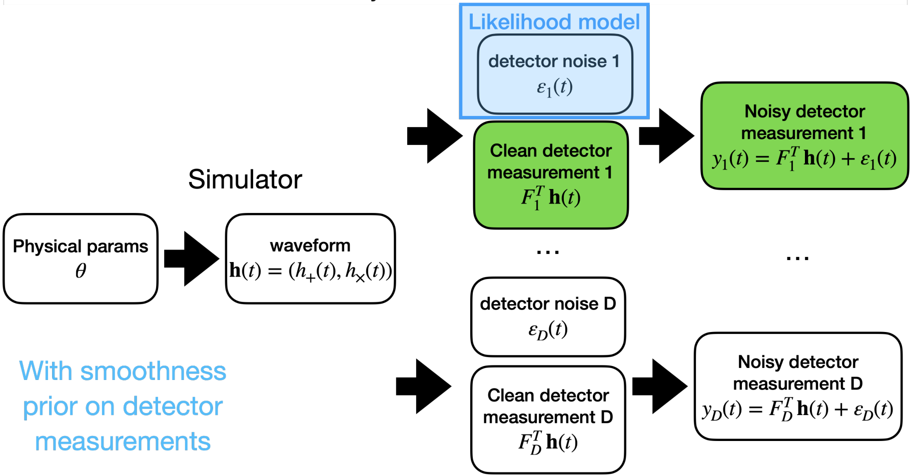
Most common case in LVK collab.: infer θ from the $\{y_d(t)\}$

MCMC (Bilby), Neural param estimation (Dingo), etc.



Other common case in LVK collab.: denoise detector measurements

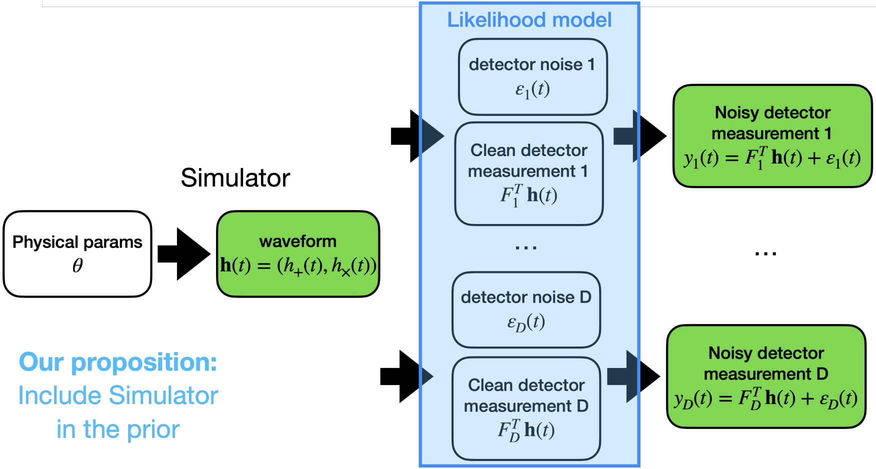
Bayeswave, AWARE



With smoothness prior on detector measurements

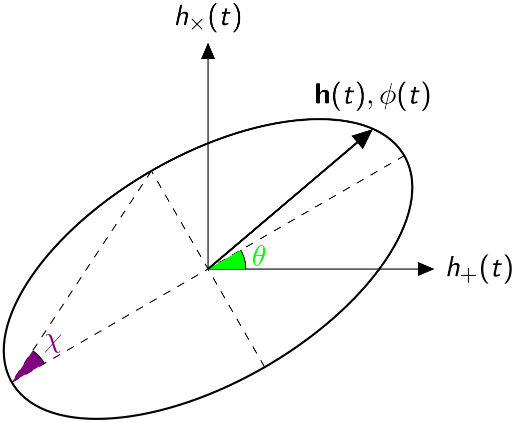
Data analysis focus in the LVK collaboration

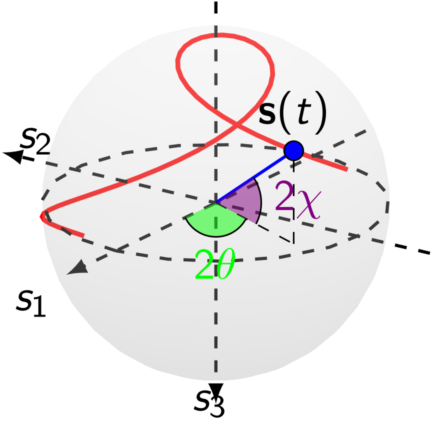
Our case: recover the waveform from all detector measurements



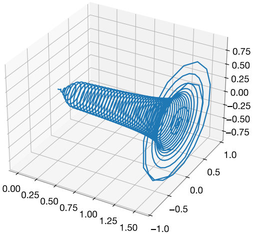
Our proposition:
Include Simulator
in the prior

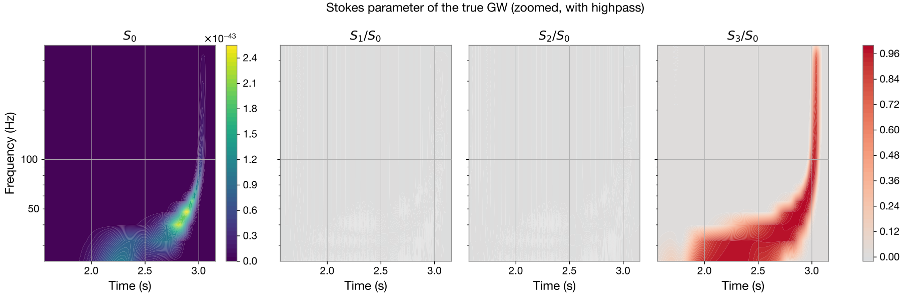
instantaneous polarization: ellipse parameters & Stokes parameters

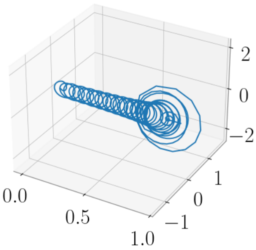


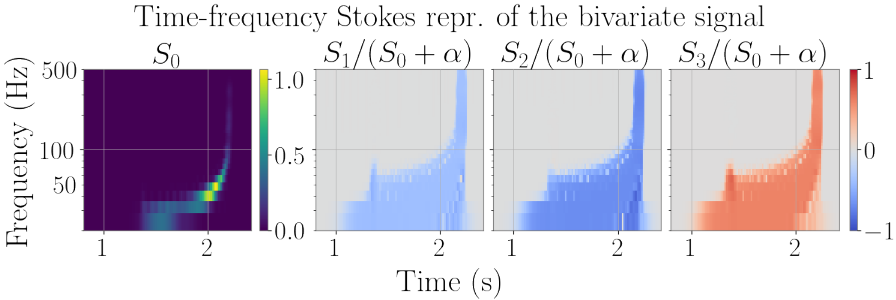


Bivariate signal processing: polarization description

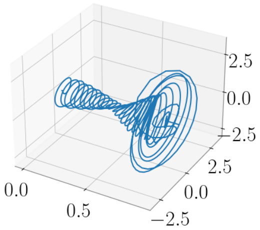


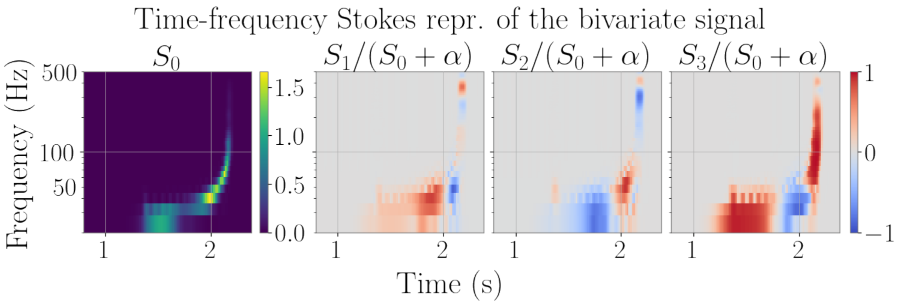



2
0
−2
1
0
−1
0.0
0.5
1.0


Time-frequency Stokes repr. of the bivariate signal
S_0
$S_1/(S_0+\alpha)$
$S_2/(S_0+\alpha)$
$S_3/(S_0+\alpha)$
Frequency (Hz)
500
100
50
1.0
0.5
0.0
1
0
−1
1
2
Time (s)

Bivariate signal processing: polarization description





Recovering the GW: set up

- ★ GW $\{\mathbf{h}(t_i) = (h_+(t_i), h_\times(t_i))\} \in \mathbb{R}^{2\times N}$ $\rightarrow$ TF repr. $\mathcal{H} \in \mathbb{C}^{2\times N_f \times N_t}$
- ★ obs. $\{\mathbf{y}(t_i)\} \in \mathbb{R}^{D\times N}$ (D detectors) $\rightarrow$ TF repr. $\mathcal{Y} \in \mathbb{C}^{D\times N_f \times N_t}$
- ★ noise n_d, usually Gaussian with one PSD per detector d
- ★ source loc.: $(\delta, \phi) \in \mathbb{S}^2$, causes a prop. time delay τ_d (neglected here)
- ★ projection of GW onto detectors: encoded in $\mathbf{F} \in \mathbb{R}^{D\times 2}$ matrix

$$y_d(t_i) = F_d(\delta,\phi)^T \mathbf{h}(t_i) + n_d(t_i) \quad \text{or} \quad \widetilde{\mathcal{Y}} = \widetilde{\mathbf{F}}\mathcal{H} + \widetilde{N},\ \widetilde{N} \sim \mathcal{N}(0, \sigma^2 \mathcal{I}) \ \text{(whitened)}$$

Recovering the GW: set up

- ★ GW $\{\mathbf{h}(t_i) = (h_+(t_i), h_\times(t_i))\} \in \mathbb{R}^{2\times N}$ $\rightarrow$ TF repr. $\mathcal{H} \in \mathbb{C}^{2\times N_f \times N_t}$
- ★ obs. $\{\mathbf{y}(t_i)\} \in \mathbb{R}^{D\times N}$ (D detectors) $\rightarrow$ TF repr. $\mathcal{Y} \in \mathbb{C}^{D\times N_f \times N_t}$
- ★ noise n_d, usually Gaussian with one PSD per detector d
- ★ source loc.: $(\delta, \phi) \in \mathbb{S}^2$, causes a prop. time delay τ_d (neglected here)
- ★ projection of GW onto detectors: encoded in $\mathbf{F} \in \mathbb{R}^{D\times 2}$ matrix

$$y_d(t_i) = F_d(\delta,\phi)^T \mathbf{h}(t_i) + n_d(t_i) \quad \text{or} \quad \widetilde{\mathcal{Y}} = \widetilde{\mathbf{F}}\mathcal{H} + \widetilde{N},\ \widetilde{N} \sim \mathcal{N}(0, \sigma^2 \mathcal{I}) \ \text{(whitened)}$$

Goal: Reconstruct $\mathcal{H}$ from $\mathcal{Y}$ – assume (δ, ϕ) known here (perform joint estim. in real life)

Metrics to evaluate the reconstruction: **speed**, **R-SNR** & **mismatch**

$$\text{R-SNR} = -10\log_{10}\left(\frac{\|\widehat{h} - h^*\|_F^2}{\|h^*\|_F^2}\right) \ \text{and}\ \varepsilon(\widehat{h}, h^*) = \min_{\tau\in\mathbb{R}}\left[1 - \frac{|\langle \widehat{h}_\tau, h^*\rangle|}{\|\widehat{h}\|\|h^*\|}\right]$$

Recovering the GW: an ill-posed inverse problem

Maximum of likelihood estimator (MLE) (in time-freq. domain)

$$\widehat{\mathcal{H}}^{(\mathrm{MLE})} = \arg\min -\log \pi(\mathcal{Y}|\mathcal{H})$$

$$= \arg\min \sum_{d} \sum_{\omega_i} \sum_{t_j} \left| \widetilde{\mathcal{Y}}_{d,\omega_i,t_j} - \widetilde{F}_d(\omega_i)^T \mathcal{H}_{\cdot,\omega_i,t_j} \right|^2$$

$$\Longrightarrow \widehat{H}^{(\mathrm{MLE})}_{\cdot,\omega_i,t_j} = \widetilde{\mathbf{F}}(\omega_i)^\dagger \, \widetilde{\mathcal{Y}}_{\cdot,\omega_i,t_j},$$ closed-form solution, fast to evaluate ✓

Recovering the GW: an ill-posed inverse problem

Maximum of likelihood estimator (MLE) (in time-freq. domain)

$$\widehat{\mathcal{H}}^{(\mathrm{MLE})} = \arg\min -\log \pi(\mathcal{Y}|\mathcal{H})$$
$$= \arg\min \sum_{d} \sum_{\omega_i} \sum_{t_j} \left| \widetilde{\mathcal{Y}}_{d,\omega_i,t_j} - \widetilde{F}_d(\omega_i)^T \mathcal{H}_{\cdot,\omega_i,t_j} \right|^2$$

$\Longrightarrow \widehat{H}^{(\mathrm{MLE})}_{\cdot,\omega_i,t_j} = \widetilde{\mathbf{F}}(\omega_i)^\dagger \, \widetilde{\mathcal{Y}}_{\cdot,\omega_i,t_j}$, closed-form solution, fast to evaluate ✓

ill-posed problem: MLE leads to poor reconstructions due to noise

improving reconstructions: requires the choice of a prior distri. $\pi(\mathcal{H})$

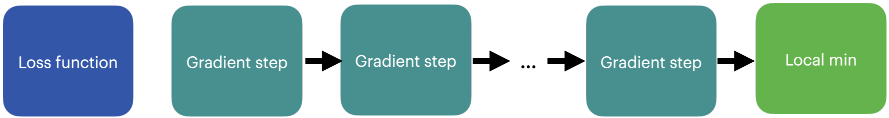
challenge: hard to encode relevant features & need a fast optim. algo.

Our approach

use plug-and-play method: learn prior $\pi(\mathcal{H})$ from a dataset of simulations

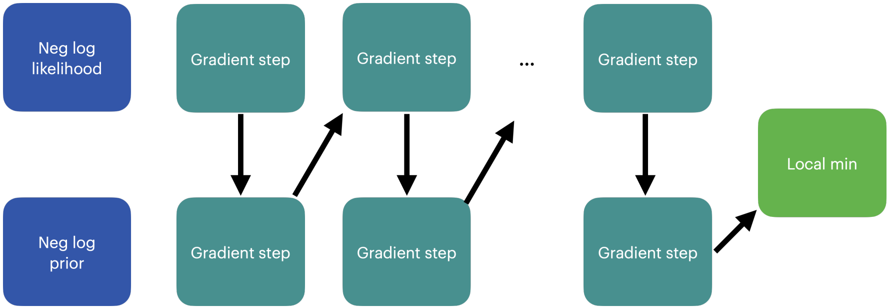
The plug-and-play (PnP) approach: general idea

Minimizing the loss function $-\log \pi(\mathcal{H}|\mathcal{Y}) = -\log \pi(\mathcal{Y}|\mathcal{H}) - \log \pi(\mathcal{H})$



The plug-and-play (PnP) approach: general idea

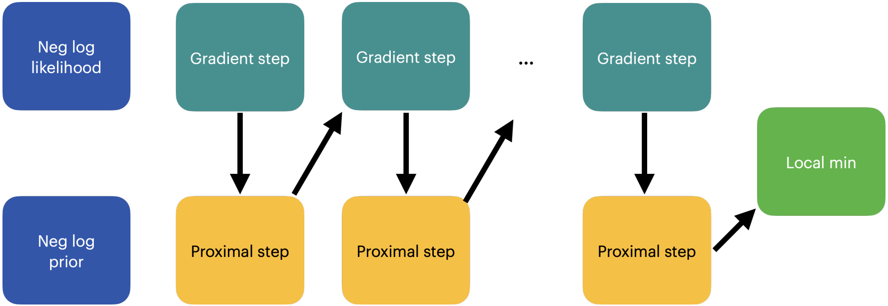
Minimizing the loss function $-\log \pi(\mathcal{H}|\mathcal{Y}) = -\log \pi(\mathcal{Y}|\mathcal{H}) - \log \pi(\mathcal{H})$



★ one can alternate grad steps wrt likelihood and prior

The plug-and-play (PnP) approach: general idea

Minimizing the loss function $-\log \pi(\mathcal{H}|\mathcal{Y}) = -\log \pi(\mathcal{Y}|\mathcal{H}) - \log \pi(\mathcal{H})$



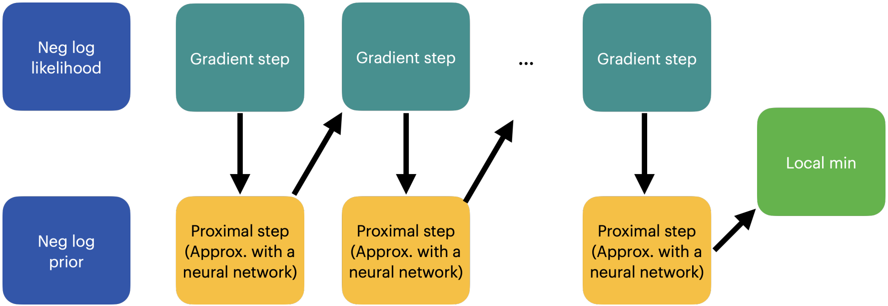
proximal operator: used for minimization of non-differentiable functions

$$\text{prox}_{-\log \pi}(\mathcal{H}) = \arg\min_{\mathcal{W}} \left[\|\mathcal{H} - \mathcal{W}\|_F^2 - \log \pi(\mathcal{W}) \right]$$

at each iteration k, $\mathcal{H}^{(k+1)} = \text{prox}_{-\log \pi} \left(\mathcal{H}^{(k)} + \varepsilon \nabla \log \pi \left(\mathcal{Y} | \mathcal{H}^{(k)} \right) \right)$

The plug-and-play (PnP) approach: general idea

Minimizing the loss function $-\log \pi(\mathcal{H}|\mathcal{Y}) = -\log \pi(\mathcal{Y}|\mathcal{H}) - \log \pi(\mathcal{H})$



replace $\text{prox}_{-\log \pi}$ with a denoiser D_σ trained from samples of $\pi(\mathcal{H})$

$$\min_\theta \mathbb{E}_{\mathcal{H}\sim\pi(\cdot),\ \sigma\sim\text{Unif}(0,\sigma_{\max}),\ \xi\sim\mathcal{N}(0,\sigma^2\mathcal{I})} \left[\| D_\sigma(\mathcal{H}+\xi) - \mathcal{H}\|_F^2\right]$$

at each iteration k, $\quad \mathcal{H}^{(k+1)} = D_\sigma\left(\mathcal{H}^{(k)} + \varepsilon \nabla \log \pi\left(\mathcal{Y}|\mathcal{H}^{(k)}\right)\right)$

The plug-and-play approach: training a signal denoiser

generated train set (2048 elements), test set (128) with

- ★ $m_1, m_2 \sim \text{Unif}(15, 30)\ \text{M}_\odot$
- ★ $d_L \sim \text{Unif}(100, 700)$ Mpc
- ★ $\cos(\text{inclination}) \sim \text{Unif}(0, 1)$
- ★ $\chi_1, \chi_2 \sim \text{Unif}(B(0, 1))$ – with $B(0, 1)$ the 3-ball around 0 of radius 1

each GW $\mathcal{H}_k$ transformed with a time-frequency transform (Gabor)

PyTorch only handles real values: time-freq representation stored as

$$\mathcal{H} = [\text{Re}(\mathcal{H}_+), \text{Im}(\mathcal{H}_+), \text{Re}(\mathcal{H}_\times), \text{Im}(\mathcal{H}_\times)]$$

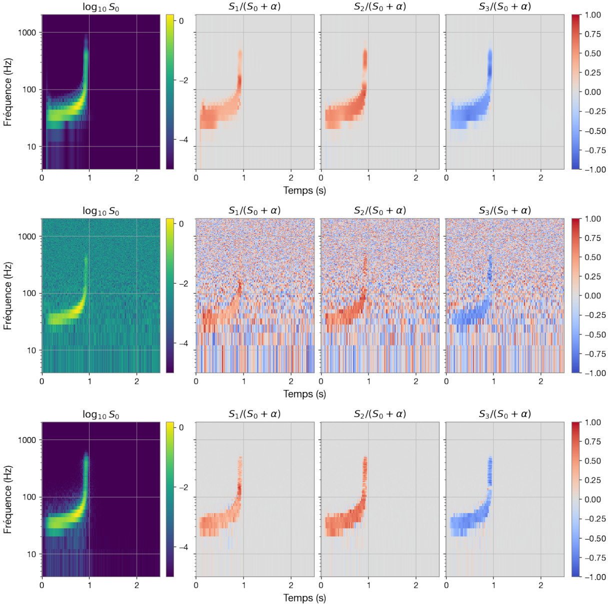
- ★ use an adapted DRUNet architecture (SOTA in image processing)
- ★ trained on a laptop with GPU (Mac's "metal performance shader")
- ★ trained for 24h

Example of application of the denoiser: easy

clean signal
$\mathcal{H}$

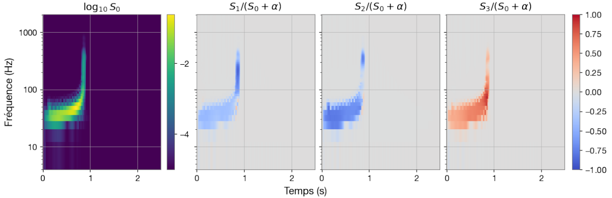
signal with Gaussian noise
$\mathcal{H} + \xi,\ \xi \sim \mathcal{N}(0, \sigma^2)$

denoised signal
$D_\sigma(\mathcal{H} + \xi)$

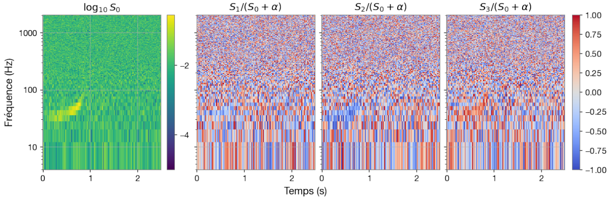


Example of application of the denoiser: harder

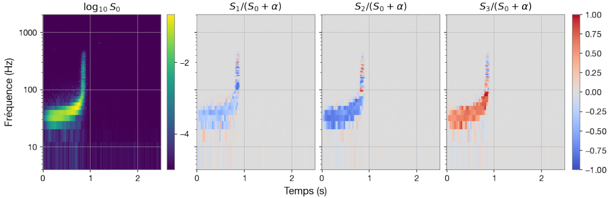
clean signal
$\mathcal{H}$



signal with Gaussian noise
$\mathcal{H} + \xi$, $\xi \sim \mathcal{N}(0, \sigma^2)$



denoised signal
$D_\sigma(\mathcal{H} + \xi)$

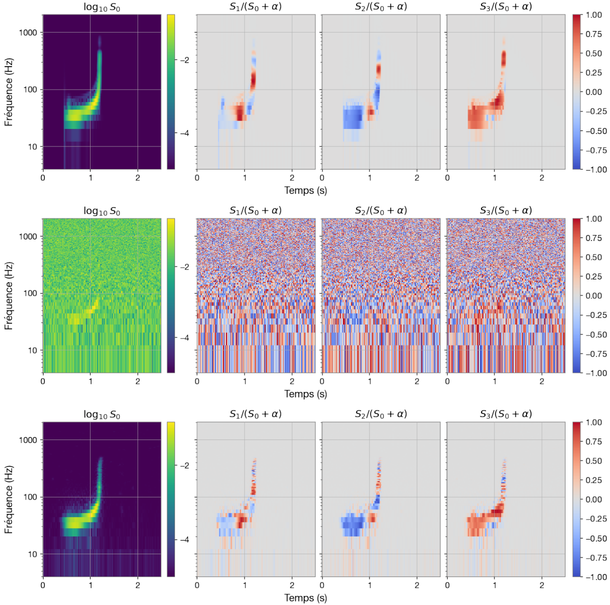


Example of application of the denoiser: much harder

clean signal
$\mathcal{H}$

signal with Gaussian noise
$\mathcal{H} + \xi,\ \xi \sim \mathcal{N}(0, \sigma^2)$

denoised signal
$D_\sigma(\mathcal{H} + \xi)$

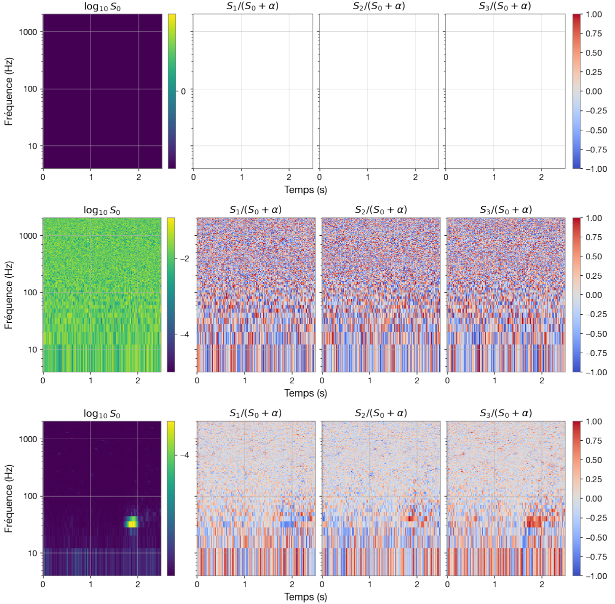


Example of application of the denoiser: empty

clean signal
$\mathcal{H}$

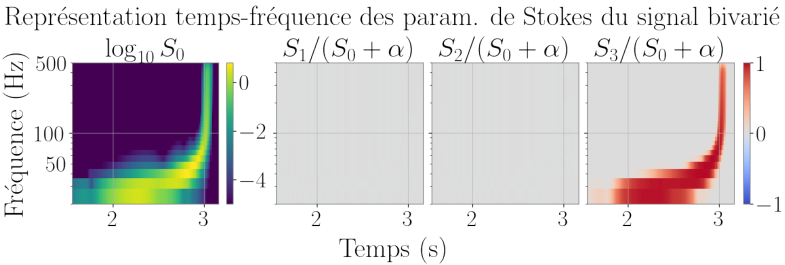
signal with Gaussian noise
$\mathcal{H} + \xi$, $\xi \sim \mathcal{N}(0, \sigma^2)$

denoised signal
$D_\sigma(\mathcal{H} + \xi)$

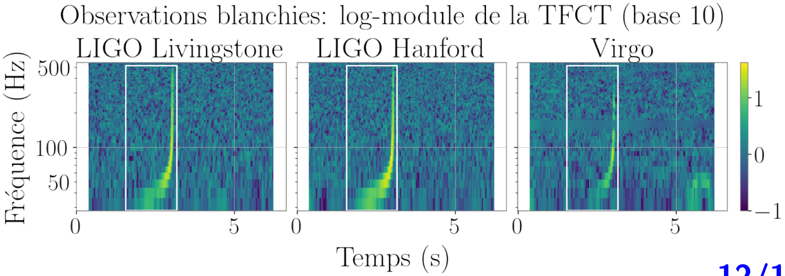


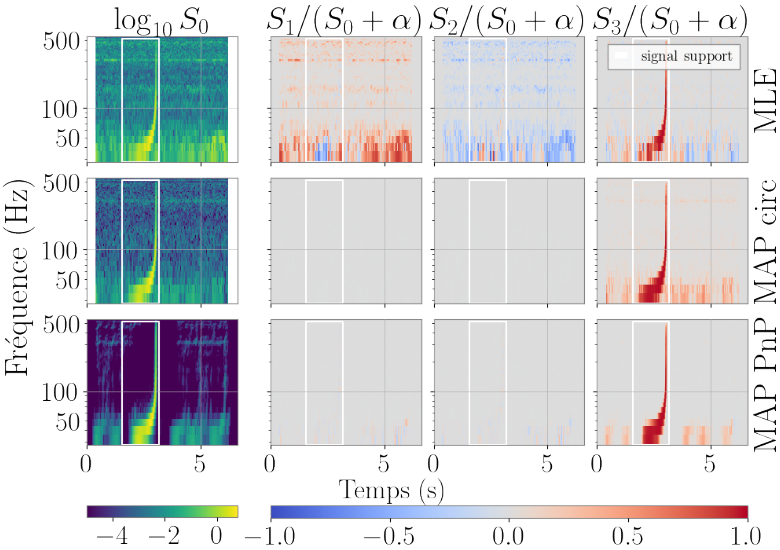
Application on synthetic data: observation

clean signal
$\mathcal{H}$



noisy observations
$\mathcal{Y}$ (modulus)





$\log_{10} S_0$
$S_1/(S_0+\alpha)$
$S_2/(S_0+\alpha)$
$S_3/(S_0+\alpha)$
signal support
MLE
MAP circ
MAP PnP
Fréquence (Hz)
500
100
50
0
5
Temps (s)
−4
−2
0
−1.0
−0.5
0.0
0.5
1.0

Application on synthetic data: quantitatively

method	mismatch		R-SNR (dB)		runtime (s)
	full	support	full	support	
MLE	0.328	0.123	-0.3	11.3	0.001
MAP circ	0.063	0.019	8.7	28.3	0.002
MAP PnP	**0.024**	**0.014**	**13.2**	**31.2**	2.7

Conclusion

Summary

✓ PnP allow for the accurate reconstruction of h_+ and $h_\times$ from LIGO and Virgo obs data, driven by physical models

✓ Well adapted to compact binary mergers where large sets of waveform models are available or easily computed

✓ **critically, PnP only learns prior**, thus can be used with any likelihood model: Versatile!
$\rightarrow$ can be applied readily for any noise PSD, projection matrix $\mathbf{F}$ or detector network!

Next steps

★ joint inference of $\mathcal{H}$ with sky location

★ quantify uncertainties using MCMC

★ apply to real data