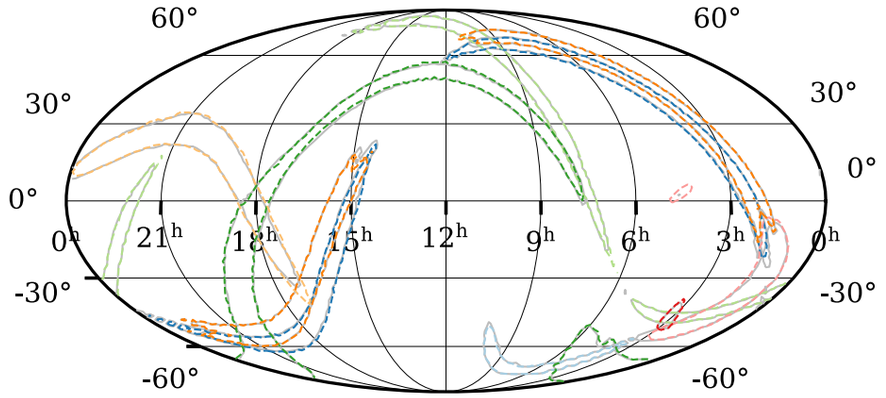


Amortized simulation-based inference for compact binaries

Stephen Green
Albert Einstein Institute Potsdam

based on 2106.12594
(w/ **Dax**, Gair, Macke, Buonanno, and Schölkopf)

14th Edoardo Amaldi Conference on Gravitational Waves
July 21, 2021

Bayesian inference for compact binaries

Posterior

$$p(\theta|s) = \frac{p(s|\theta)p(\theta)}{p(s)}$$

- **Likelihood** based on assumption of stationary Gaussian detector noise,

$$p(s\,|\,\theta) \propto \exp\left(-\frac{1}{2}\sum_I \left(s_I - h_I(\theta)\,|\,s_I - h_I(\theta)\right)\right)$$

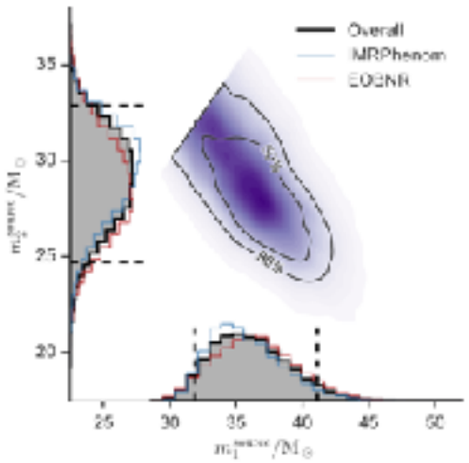
$$(a\,|\,b) = 2\int_0^\infty \mathrm{d}f\, \frac{\hat{a}(f)\hat{b}(f)^* + \hat{a}(f)^*\hat{b}(f)}{S_n(f)}$$

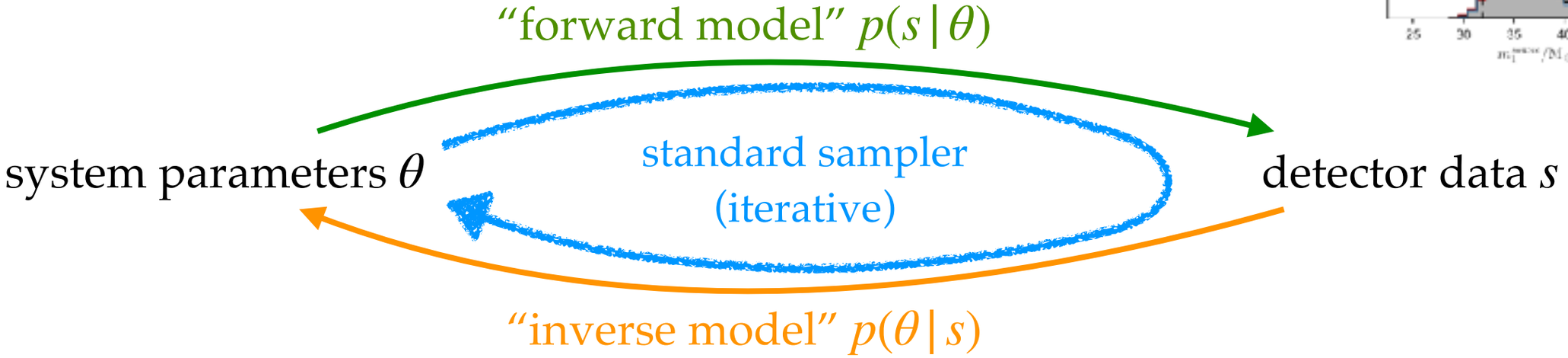
- **Prior** based on beliefs about system before looking at data,

  e.g., uniform in m_1, m_2 over some range,
  uniformly distributed in sky

Bayesian inference for compact binaries

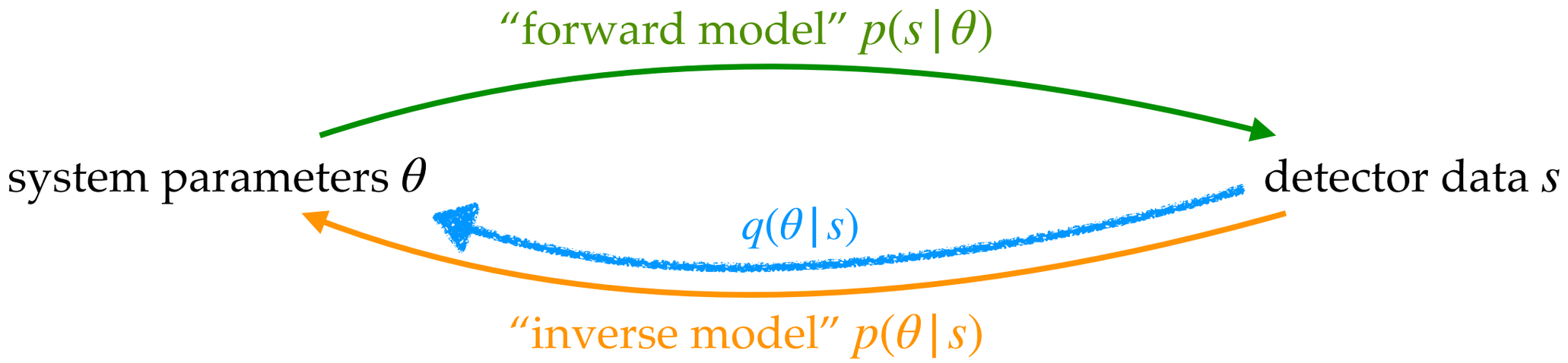
- **Inference:** draw samples $\theta \sim p(\theta|s)$





- **Drawbacks** of standard approach:
  - Sampling time ~ waveform generation time $\implies$ slow
  - Requires ability to evaluate likelihood

Neural posterior estimation



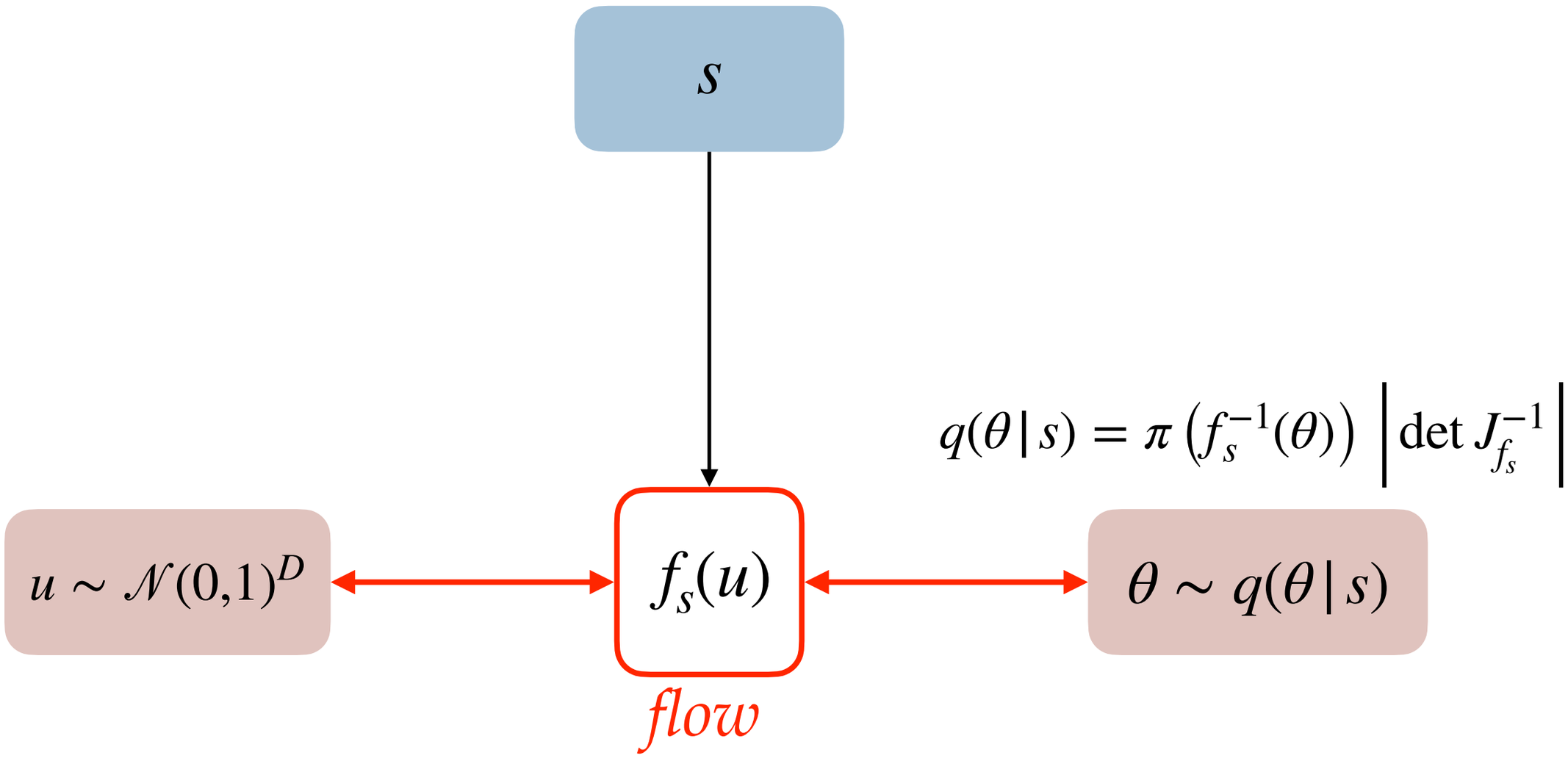
- **Neural Posterior Estimation:**
  - Construct an inverse model $q(\theta\,|\,s)$ using a neural network called a "normalizing flow".
  - Train $q(\theta\,|\,s) \to p(\theta\,|\,s)$ using simulated data, $(\theta^{(i)}, s^{(i)}) \sim p(\theta, s)$

$$\begin{cases} \theta^{(i)} \sim p(\theta), \\ s^{(i)} \sim p(s|\theta^{(i)}) \end{cases}$$

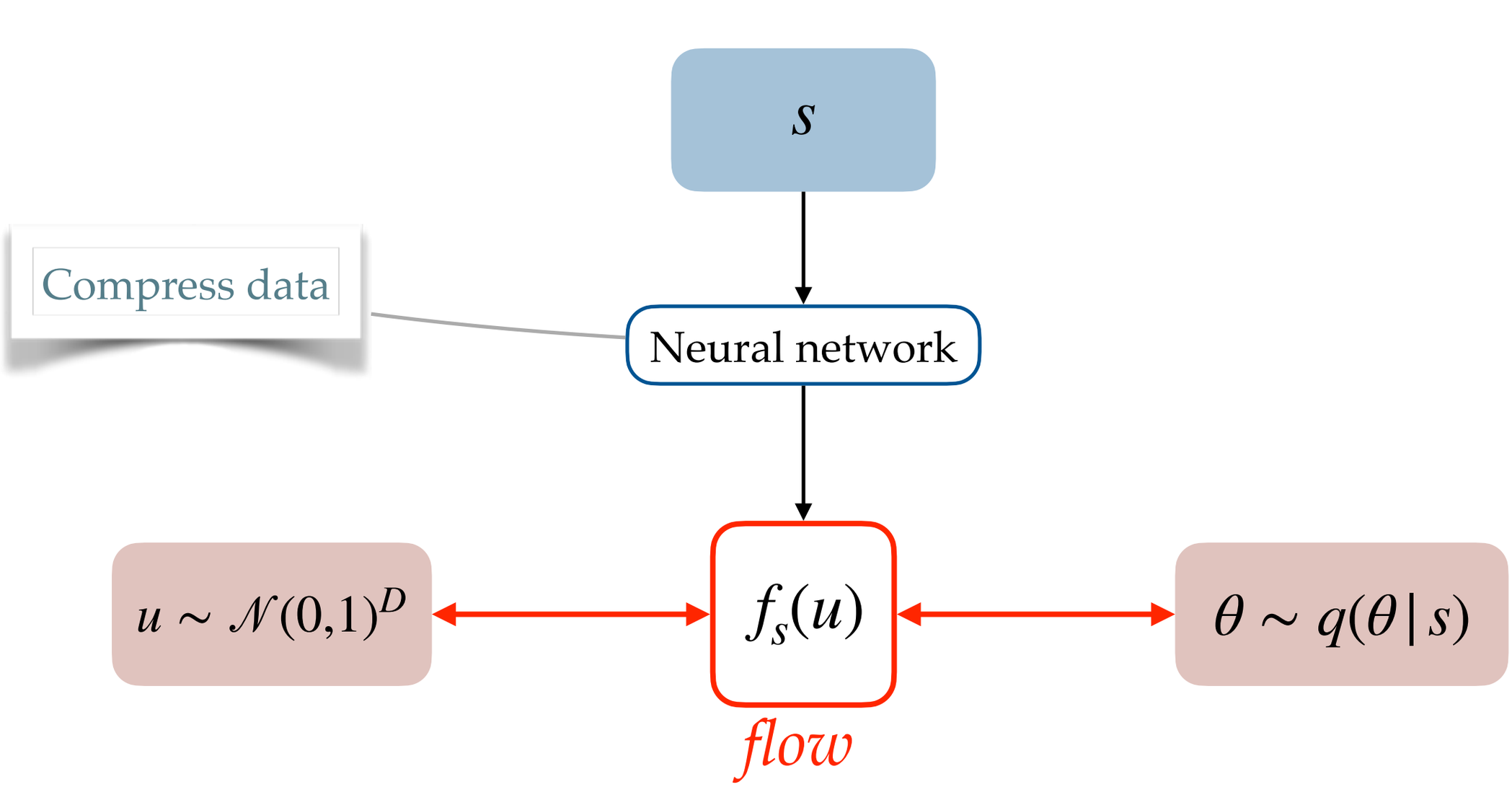
"A surrogate for the posterior"

Normalizing flow

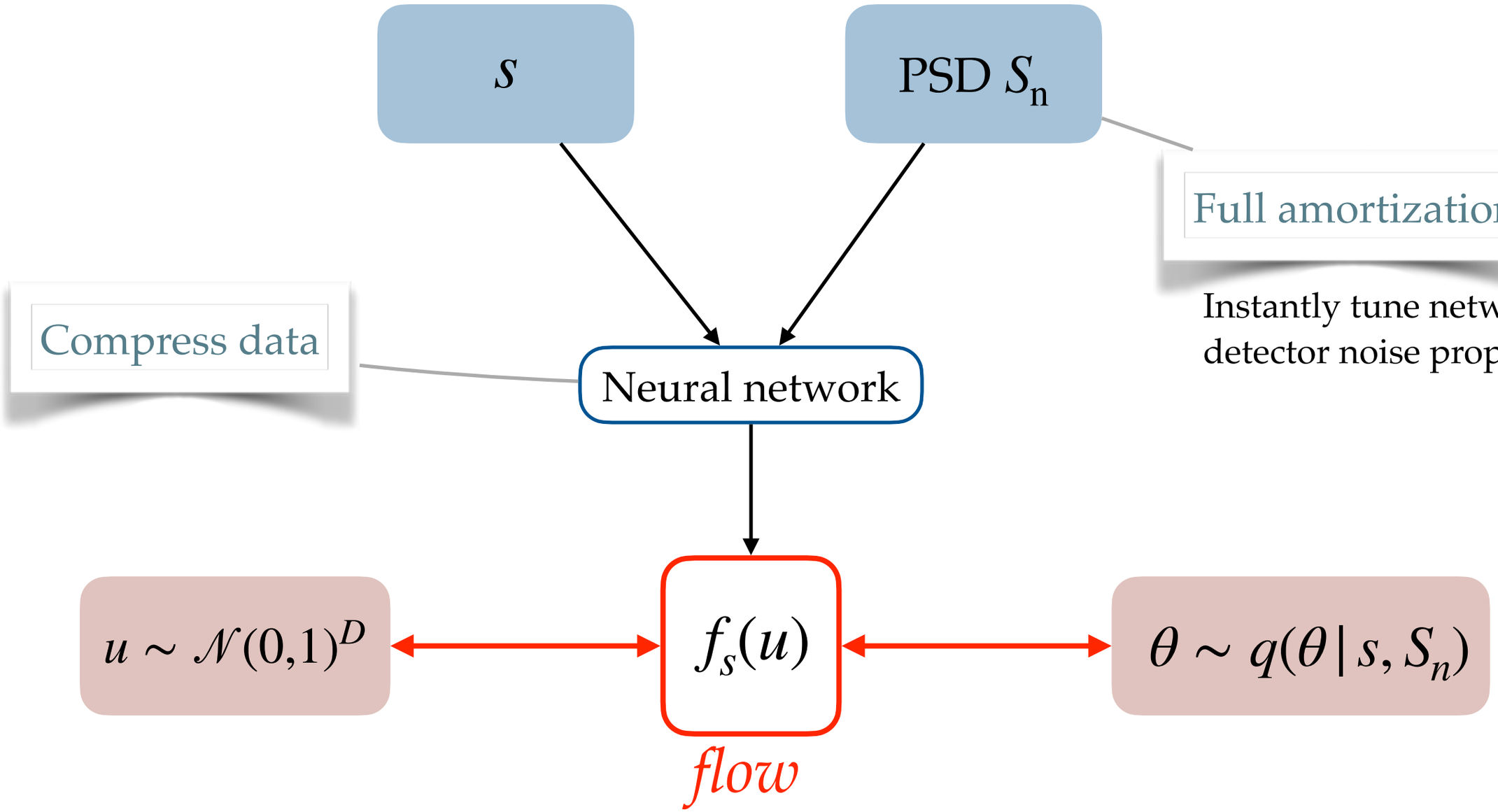
Express posterior in terms of a mapping from normally-distributed variables.



Normalizing flow



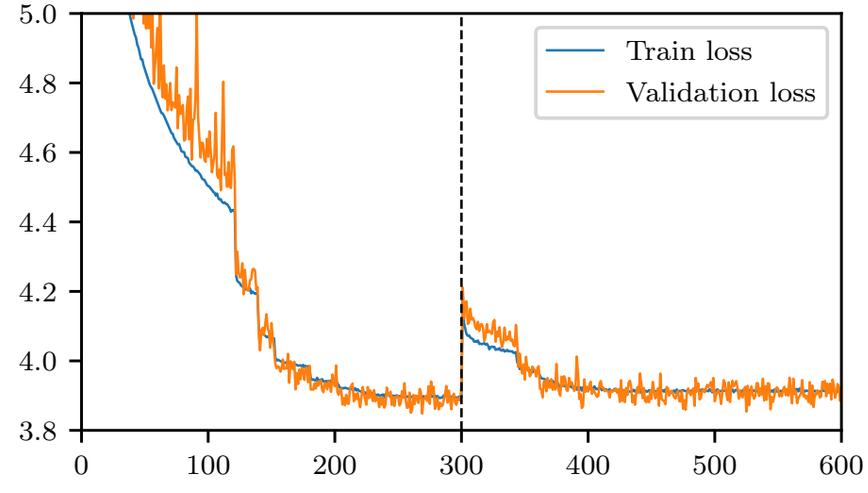
Normalizing flow



Big neural networks: ≈ 350 layers and 150 million parameters

Training

- 5×10^6 training waveforms
  - Model: IMRPhenomPv2
  - $T = 8$ s, $f_{\text{min}} = 20$ Hz, $f_{\text{max}} = 1024$ Hz
  - 15D parameter space
  - $m_1, m_2 \in [10,80]$ $M_\odot$



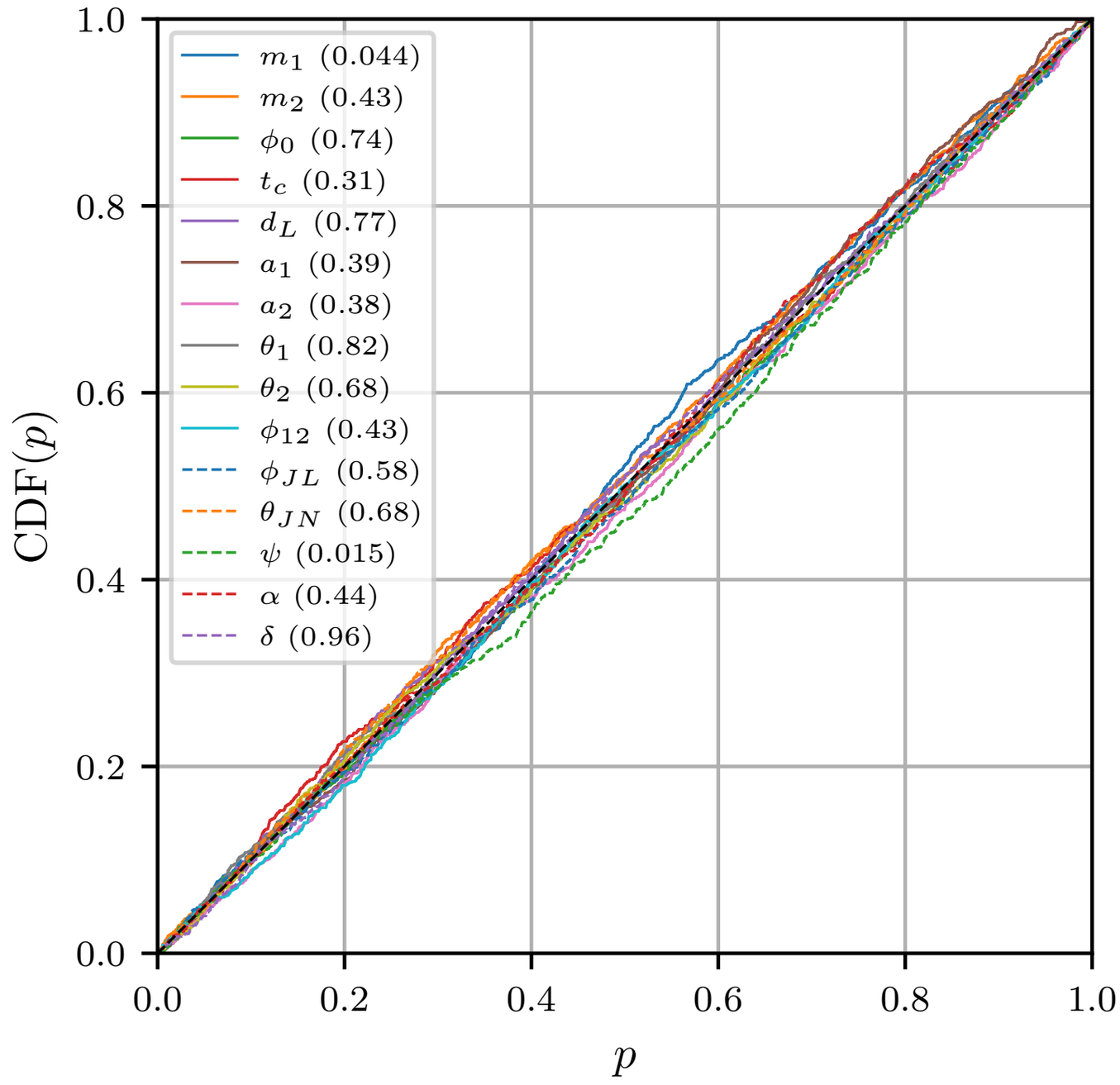
- + stationary Gaussian noise realizations consistent with given PSD

- Train several neural networks based on different noise level / number of detectors/ distance range:

Observing run	Detectors	Distance range [Mpc]
O1	HL	[100, 2000]
O2	HL	[100, 2000]
		[100, 6000]
	HLV	[100, 1000]

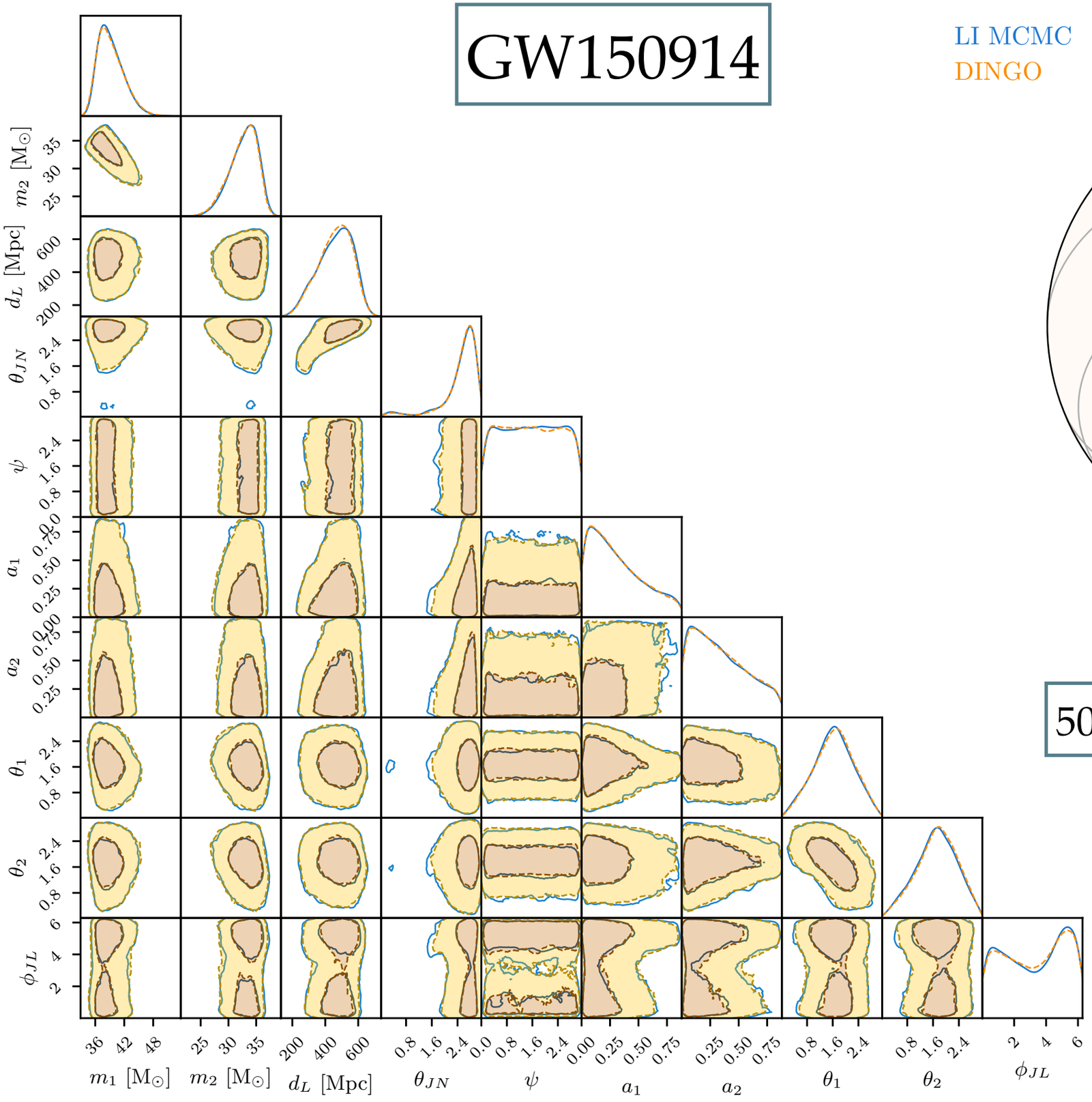
PP plot

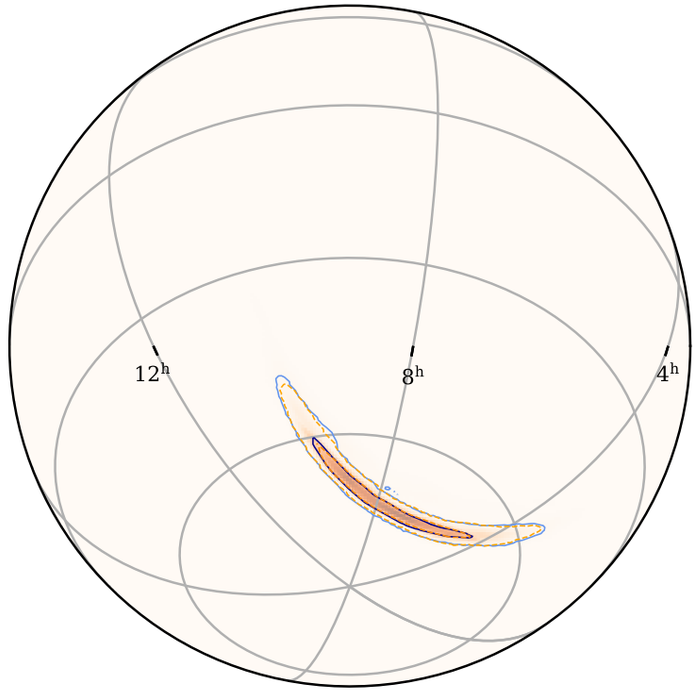
- 1000 injections
- Combined p-value of 0.46



GW150914

LI MCMC
DINGO



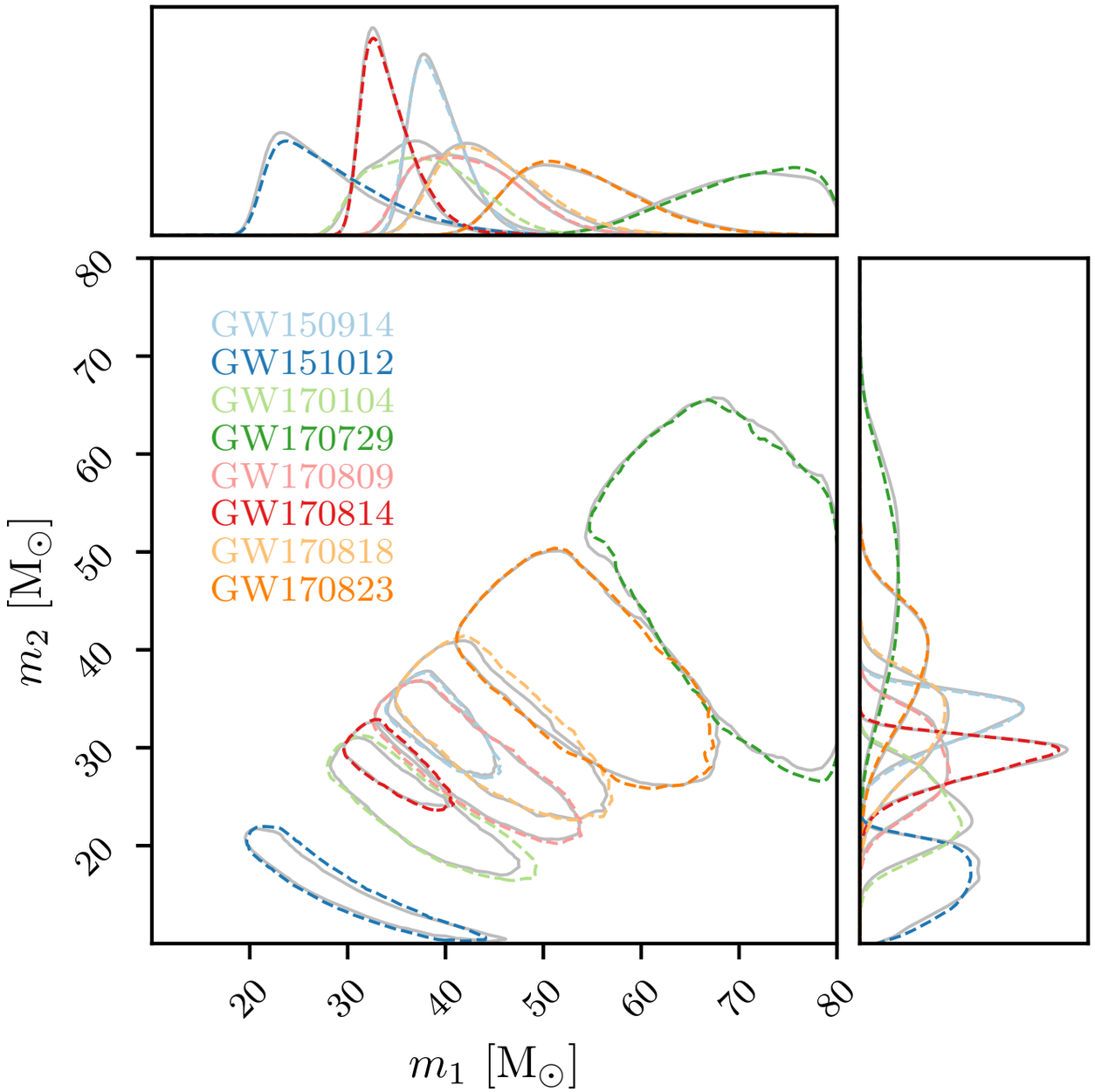


50,000 samples / minute

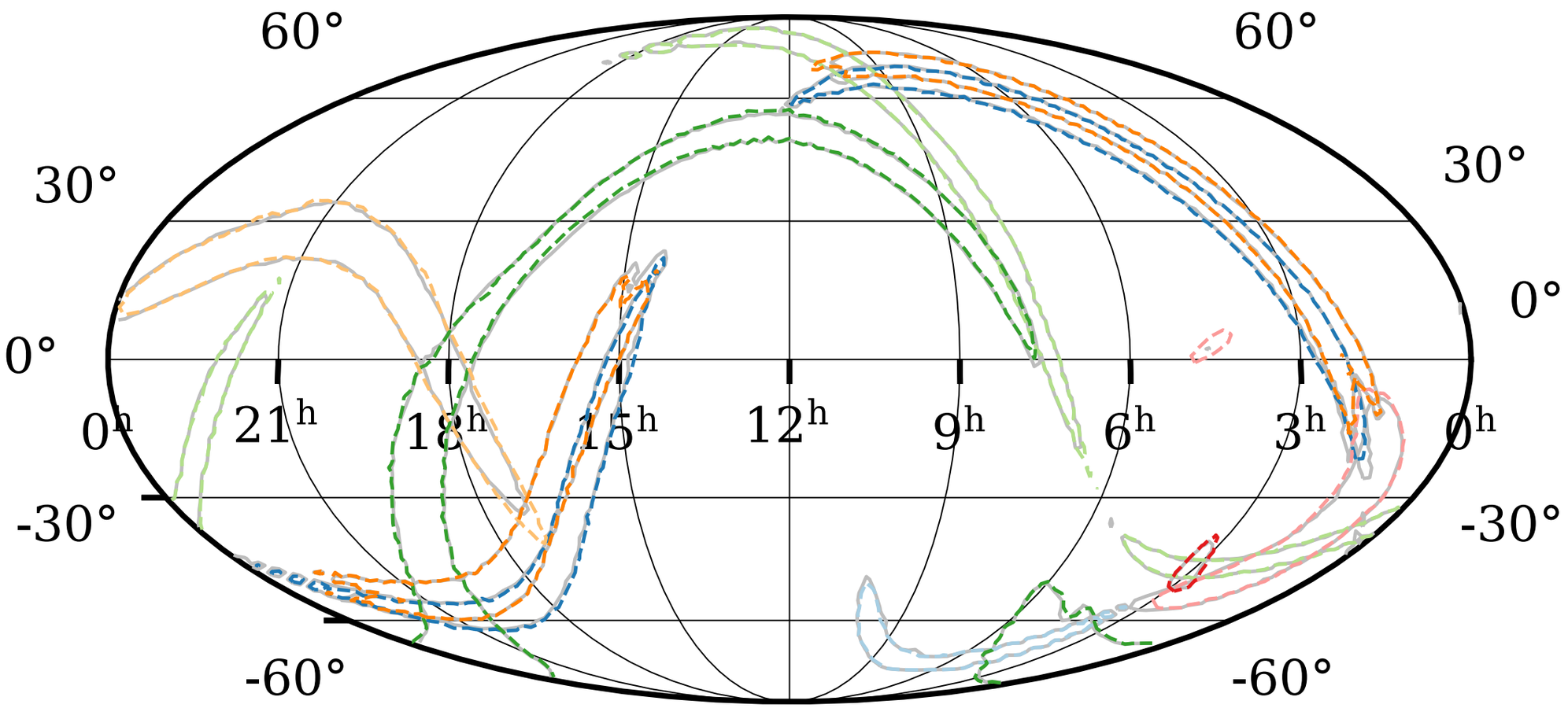
GWTC-1

- Analyzed all events consistent with prior ($m_1, m_2 \geq 10\ \mathrm{M}_\odot$)

- LALInference MCMC in gray

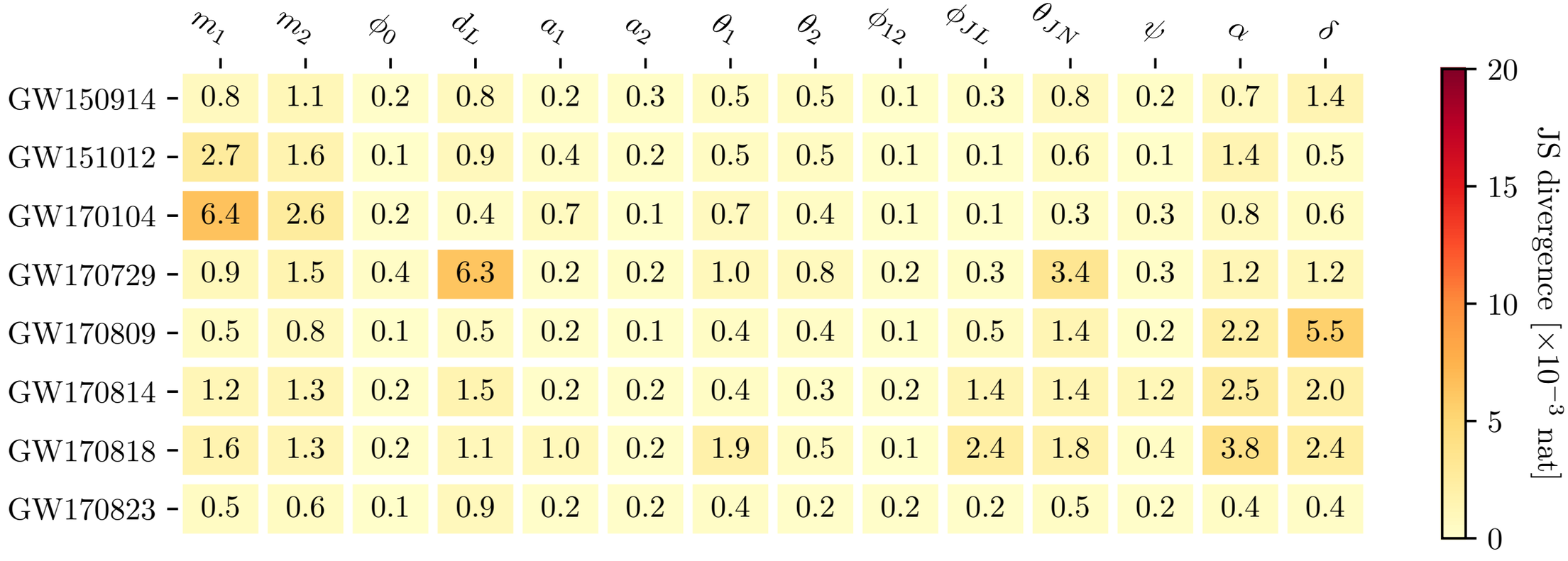


GWTC-1



JS divergence vs LALInference

- Measures a “distance” between probability distributions
- JSD $< 2 \times 10^{-3}$ nat “indistinguishable” (i.e., difference between LI runs)



Conclusions and roadmap

- Reduced analysis times for generic BBH events from ≈ day to a minute, at with results nearly indistinguishable from standard codes.

- Method accounts for noise nonstationarity from event to event.

- Can use this to analyze data in real time:

    1. Now writing a user-friendly extensible code (“Dingo”)

    2. Train with more sophisticated waveform models (e.g., SEOBNRv4PHM)

    3. Extend to binary neutron stars (requires improved data compression)

- Long term: Realistic (nonstationary / non-Gaussian) noise, LISA data analysis, populations…

Thank You