Applications of normalising flows in gravitational wave astronomy

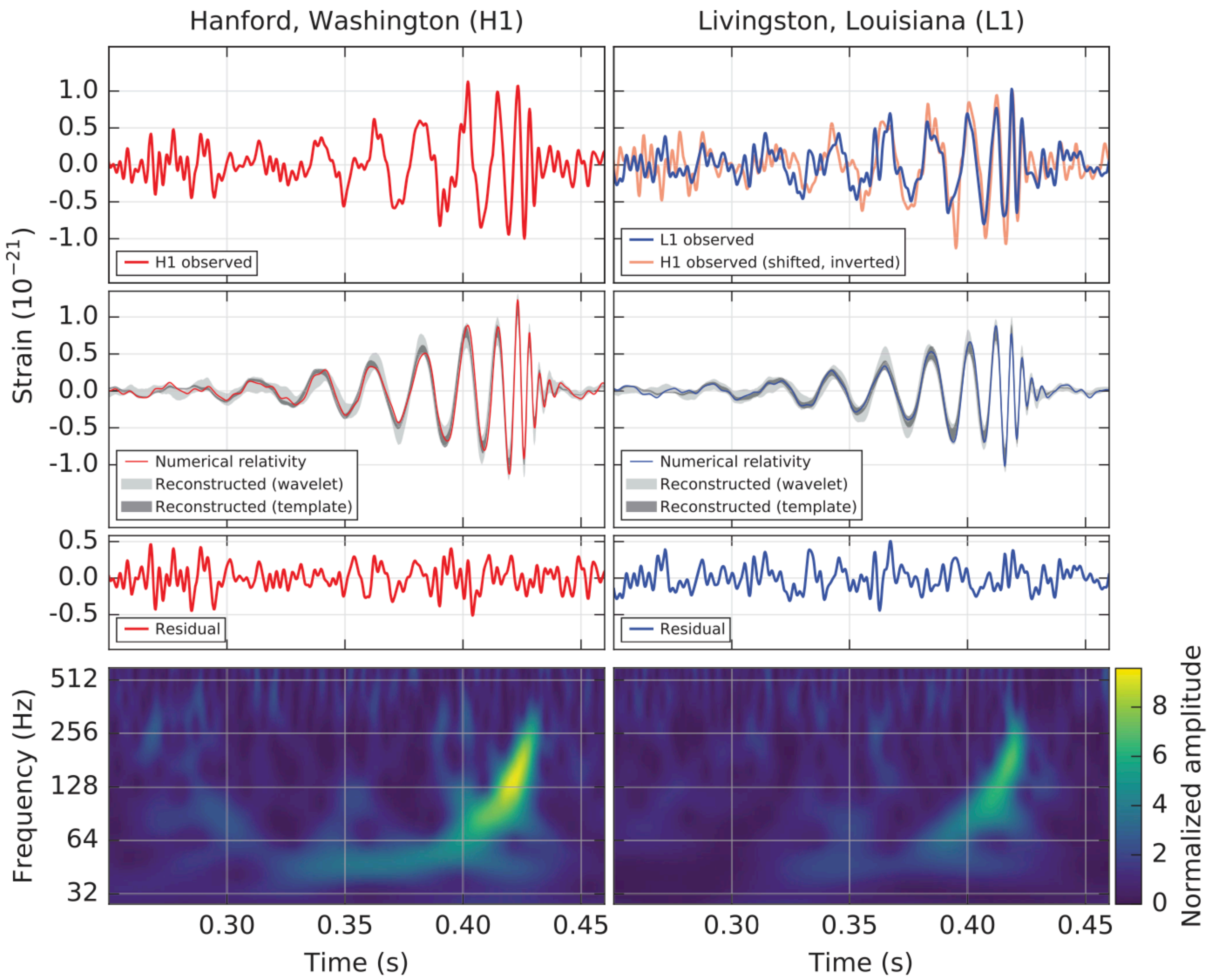
Konstantin Leyde

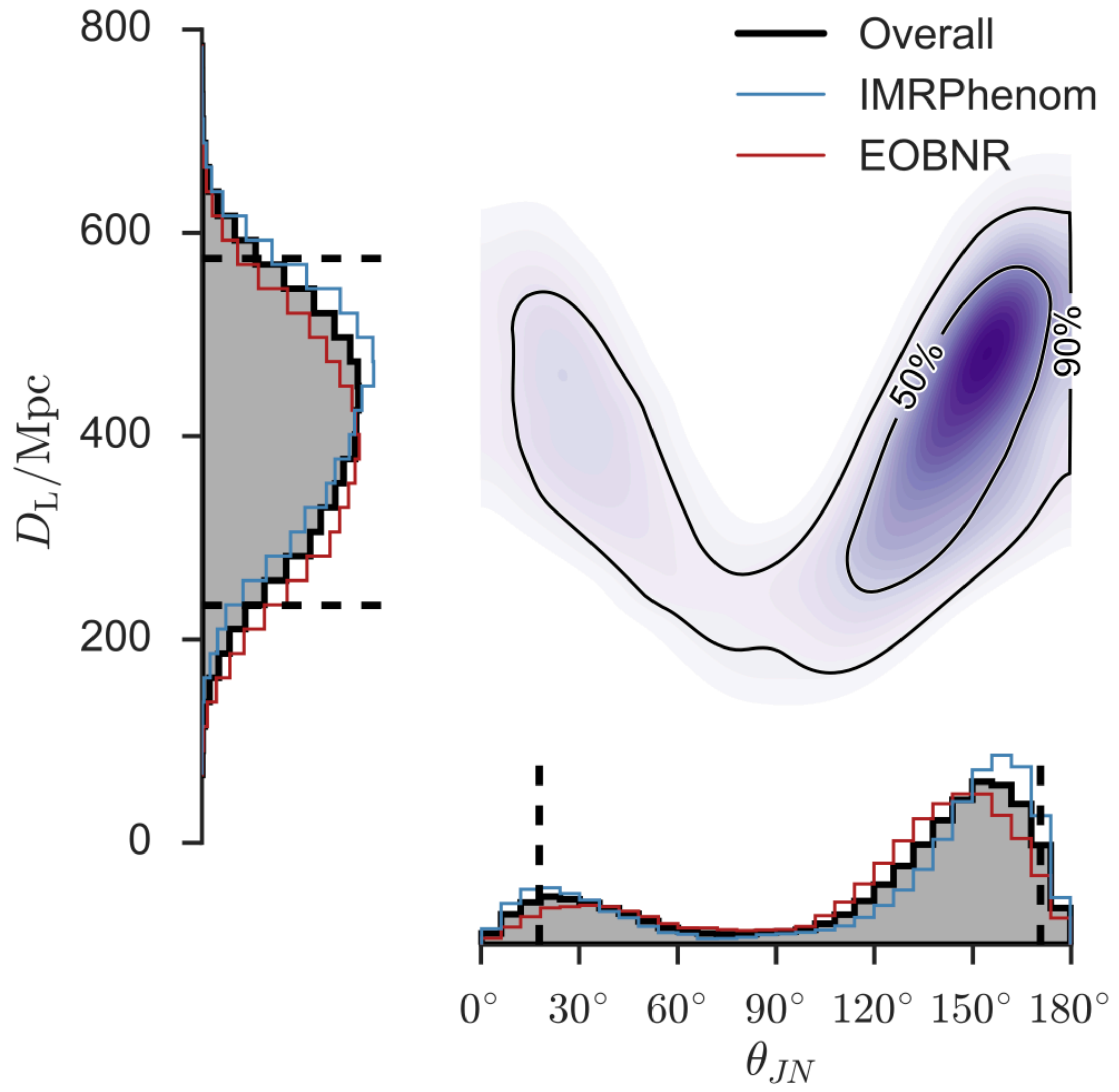
Université Paris Cité

"Méthodes d'analyse des données" du GdR Ondes Gravitationnelles, 15.11.2022

General problem

Data → Posterior distribution of parameters describing the data



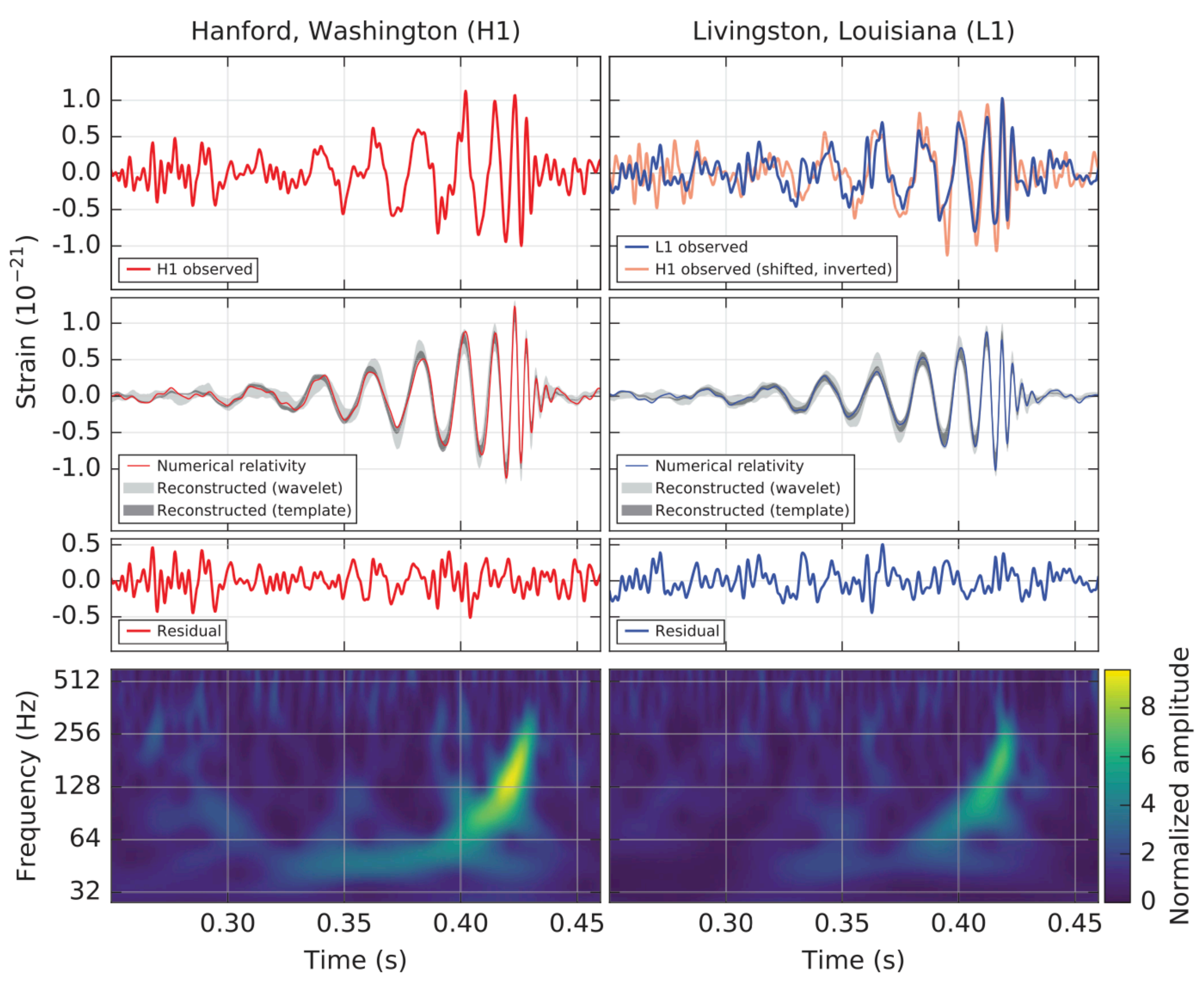


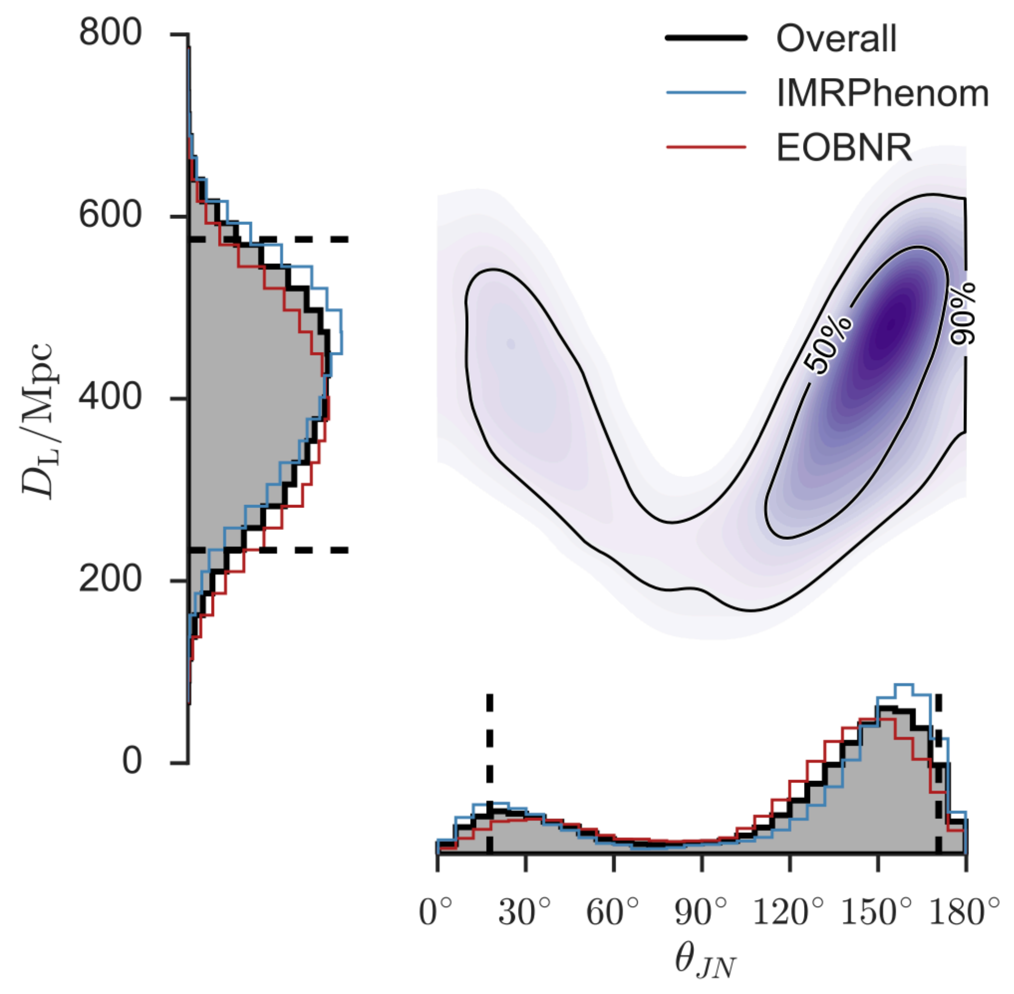
PhysRevLett.116.061102

1602.03840

Why new methods?

- Computational speed
- Need for explicit formulation of the noise model (difficult for non-Gaussian errors)
- Amortisation of computational cost over large number of events





Abstract method

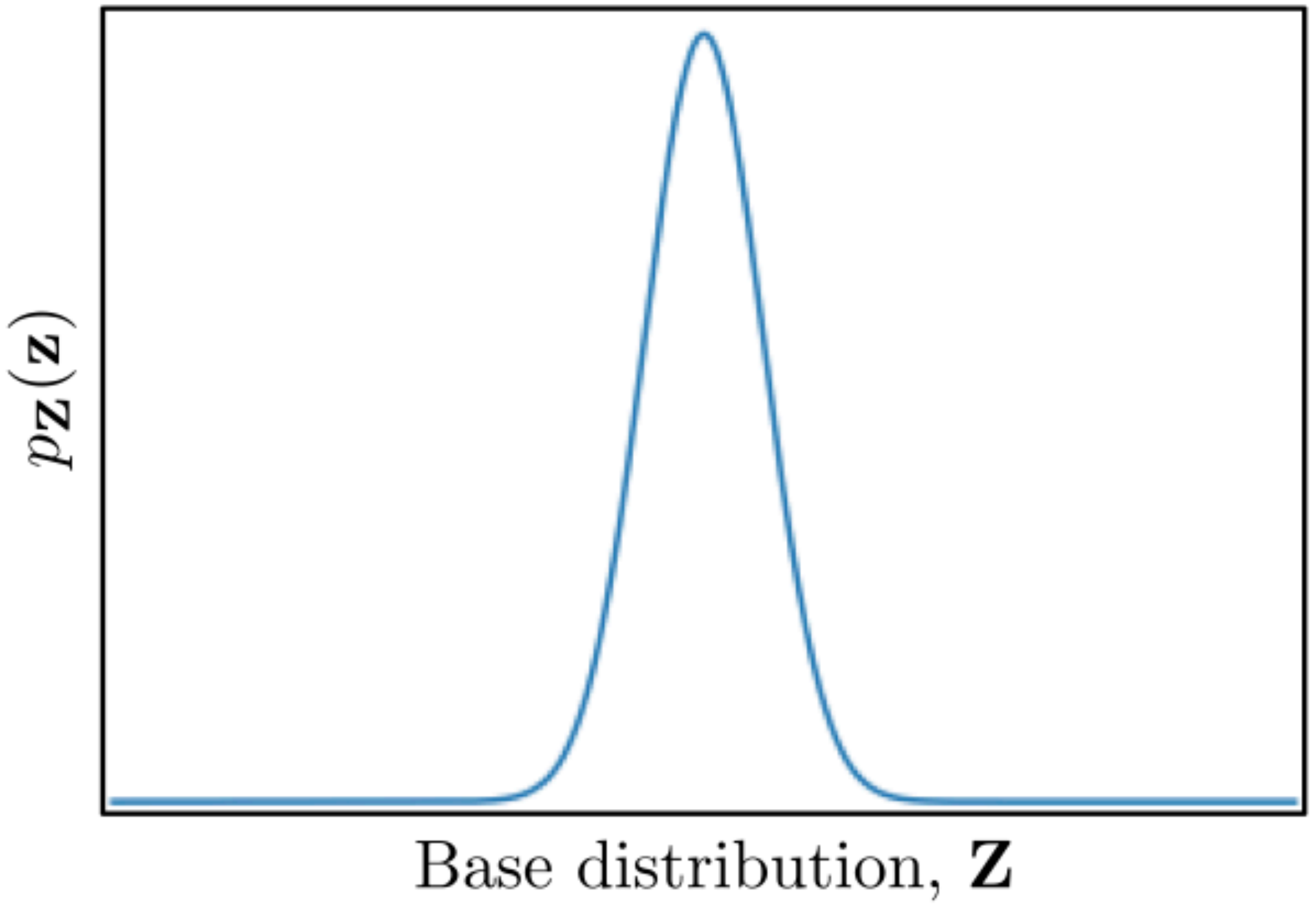
- Need for a function $\mathscr{F}$ that maps

$$\mathscr{F} : \overset{\text{Data}}{d} \to q(Y|d) \quad \text{Posterior distribution of parameters } Y$$

- A normalising flow is one possible realisation of this function
  - Very versatile
  - Fast to compute
  - Fast to sample from

1 dimensional normalising flow

$$q(Y|d) = \mathcal{N}(0,1)^D\left(g^{-1}(Y)\right)\left|\det J_{g^{-1}}\right|$$

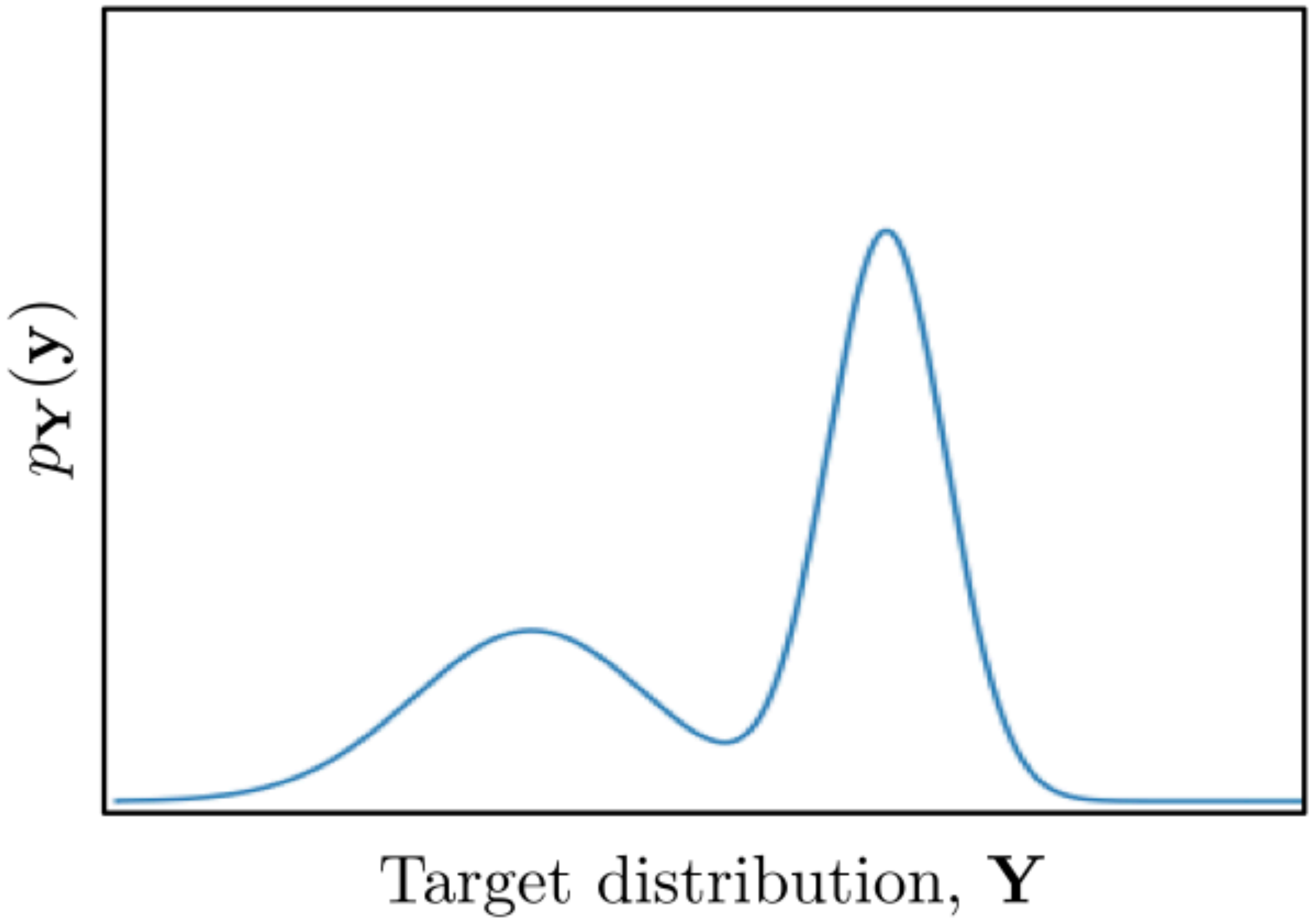


“Generative direction” →

$Y := g(Z, d)$

Mapping g depends on data

“Normalising direction” ←

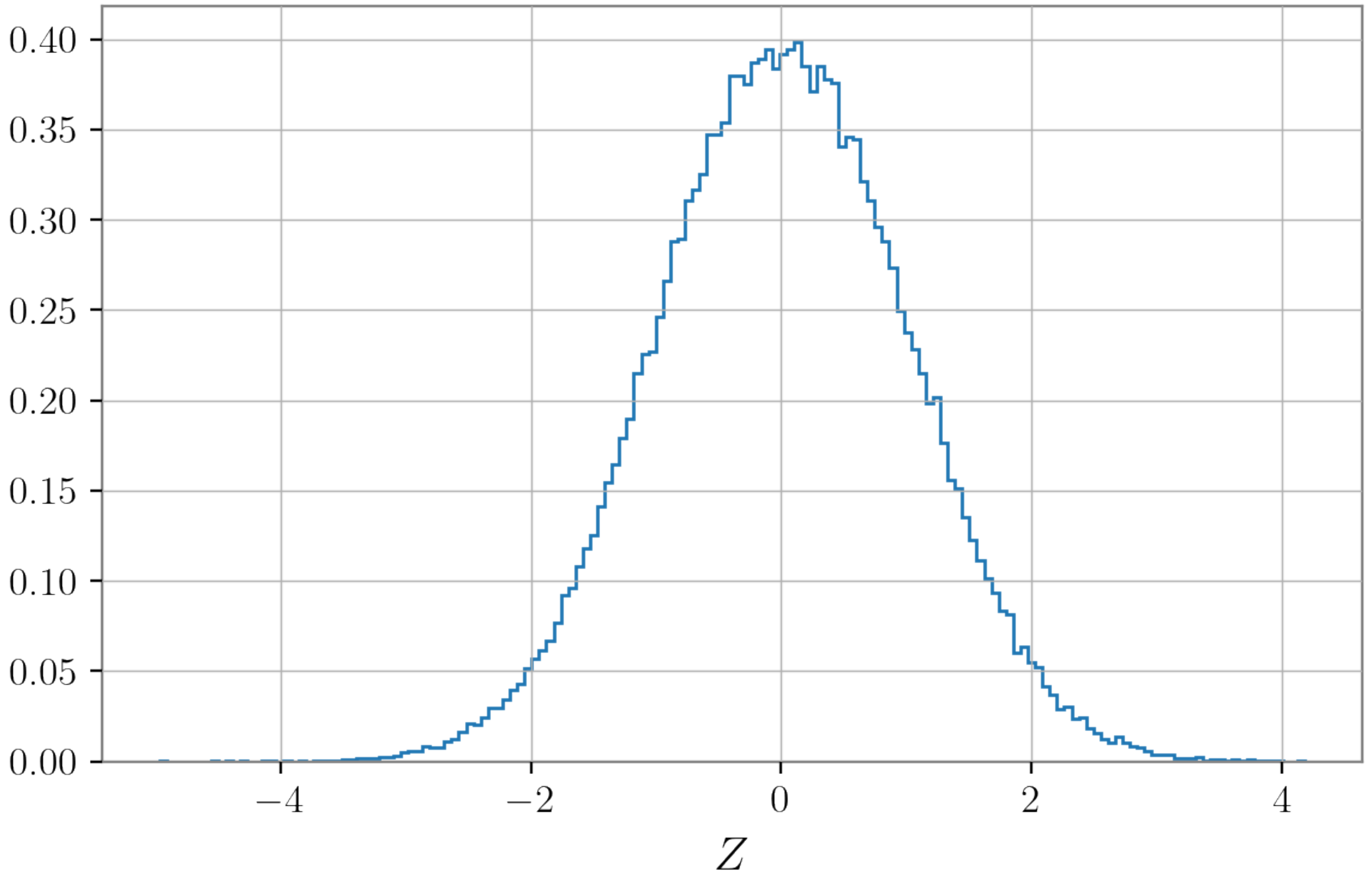


Find a g such that the normal distribution is mapped to the posterior

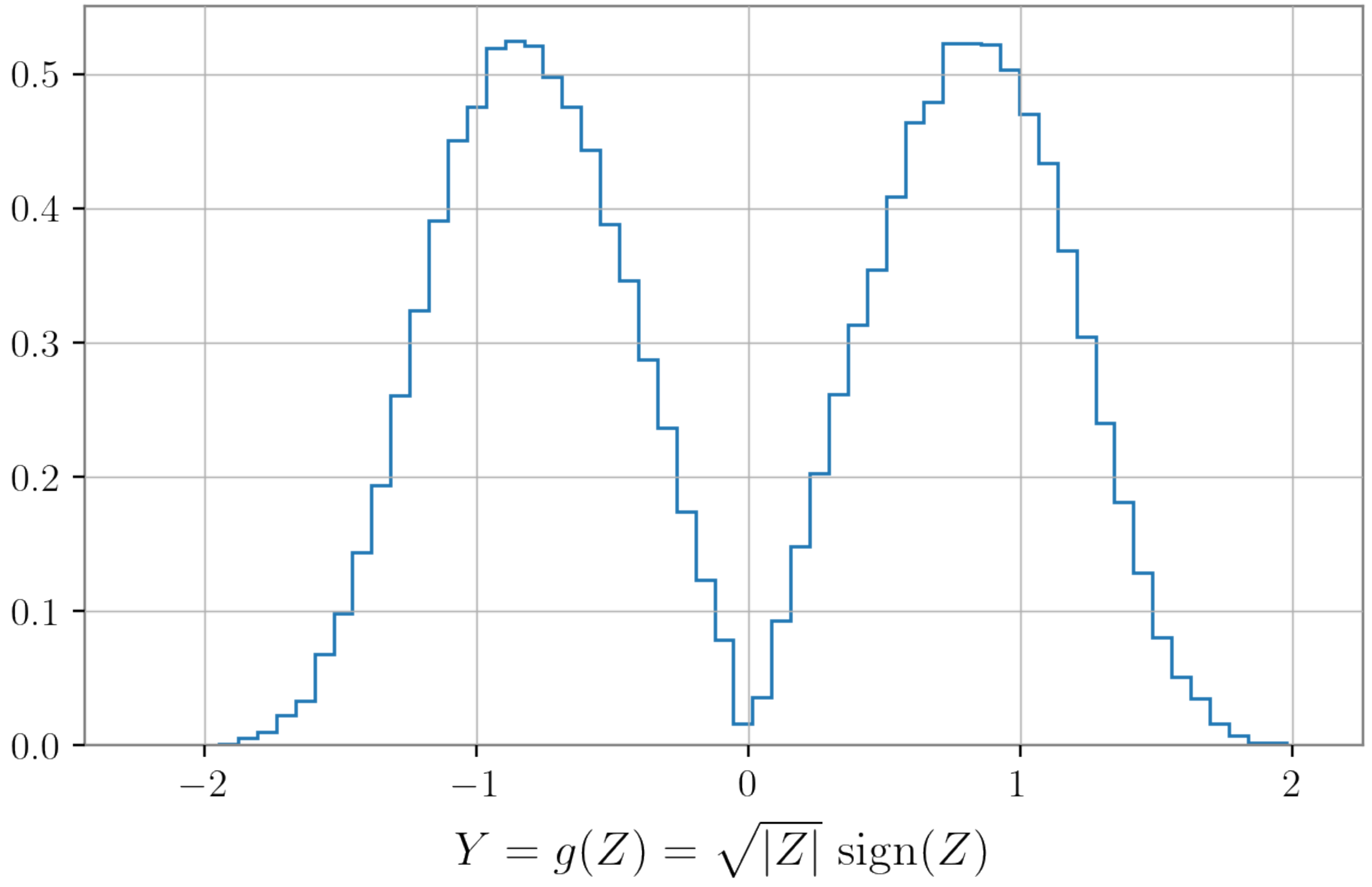
1908.09257

1 dimensional normalising flow: Example

$$q(Y\,|\,d) = \mathcal{N}(0,1)^D\left(g^{-1}(Y)\right)\left|\det J_{g^{-1}}\right|$$

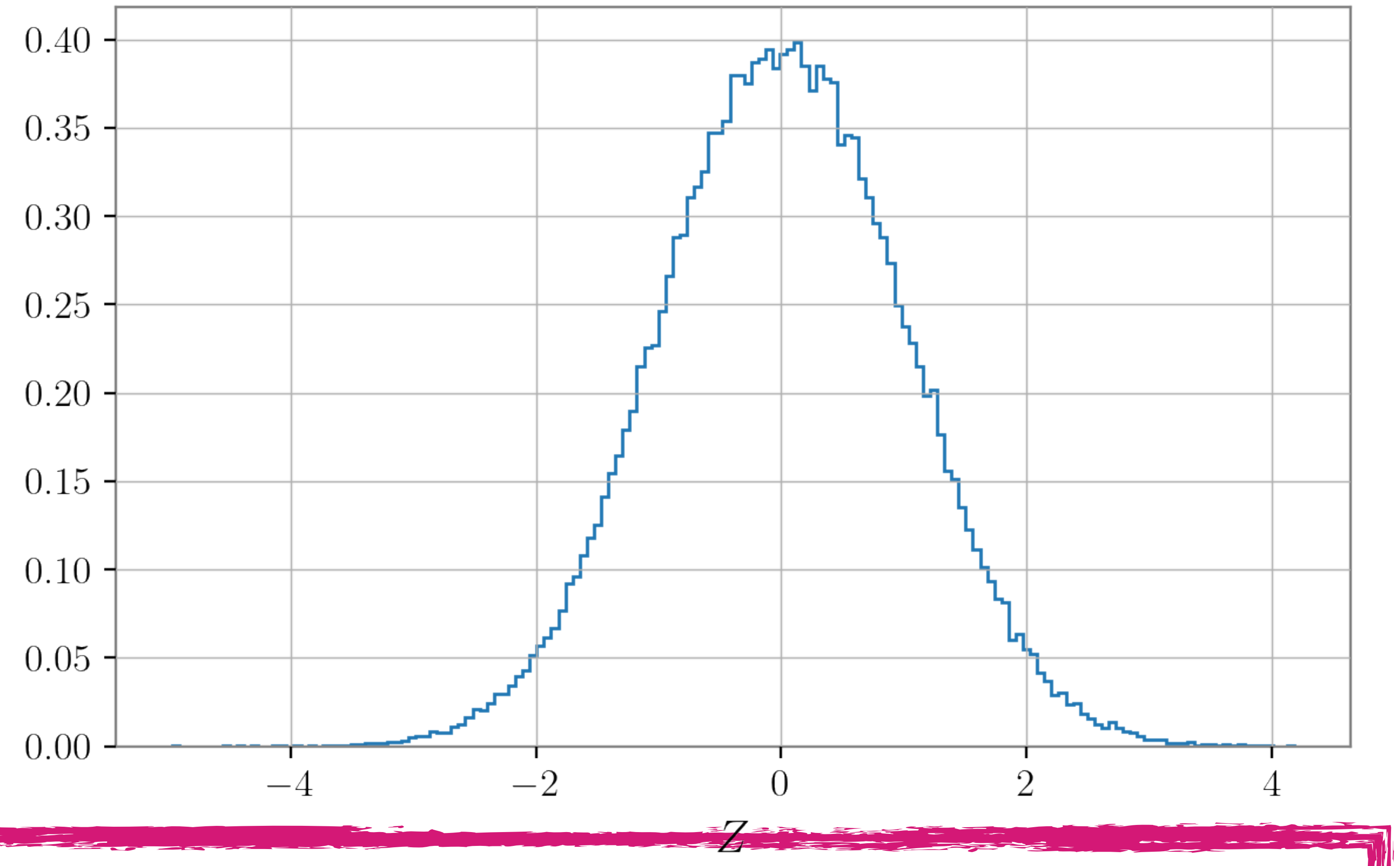


Example mapping g

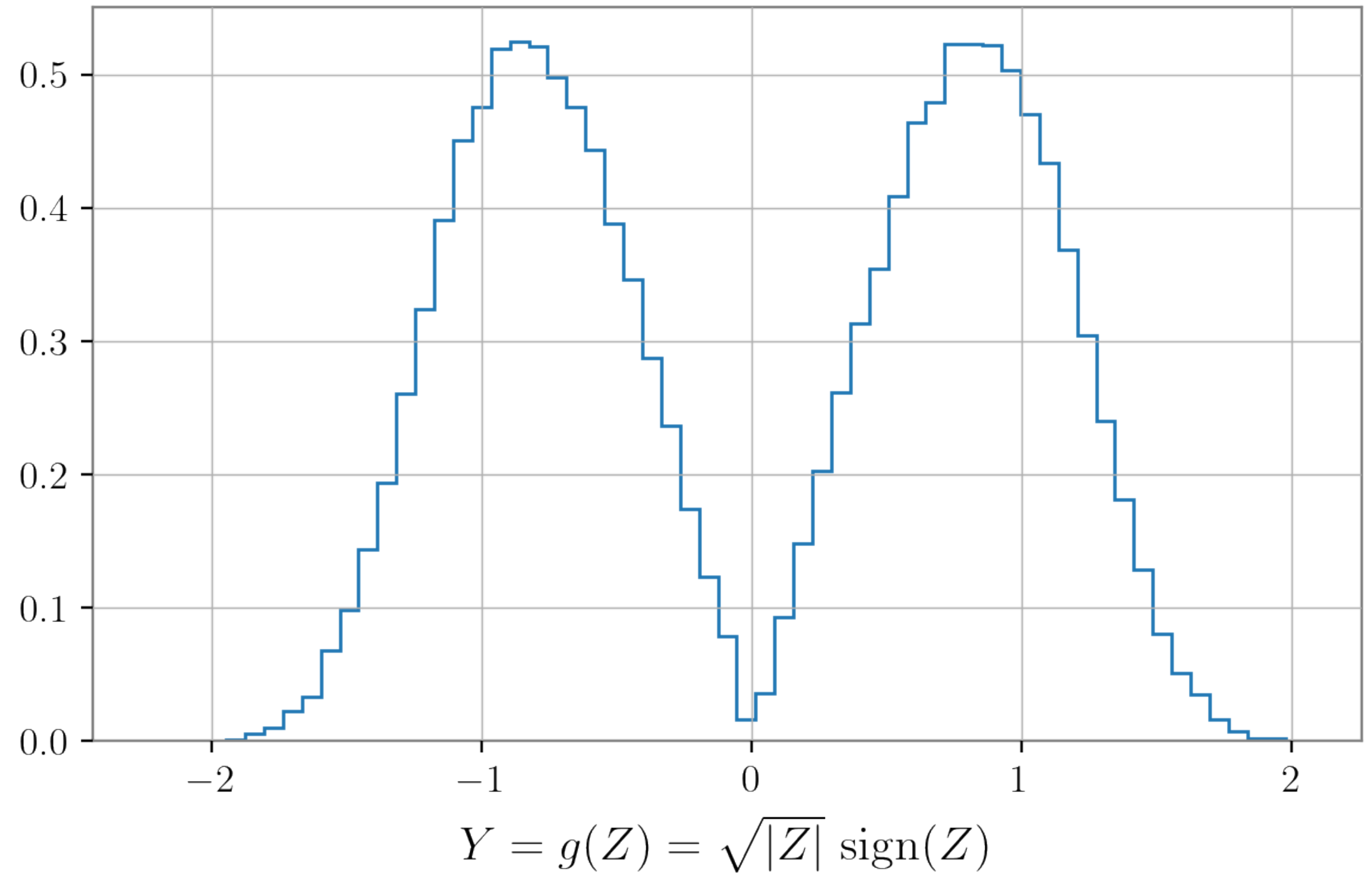


$$Y := g(Z,d) = g(Z) = \sqrt{|Z|}\,\mathrm{sign}(Z)$$

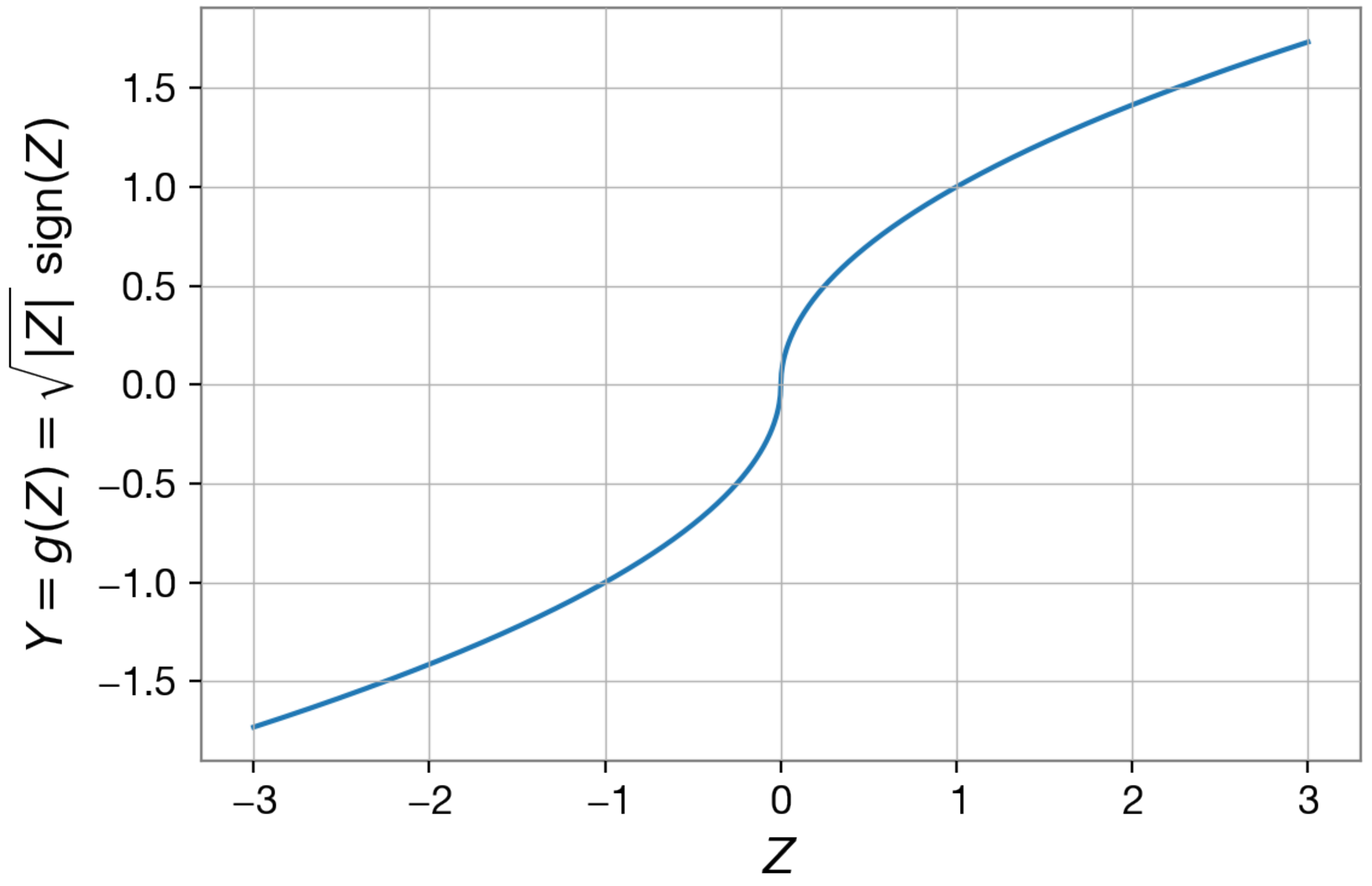
1 dimensional normalising flow: Example

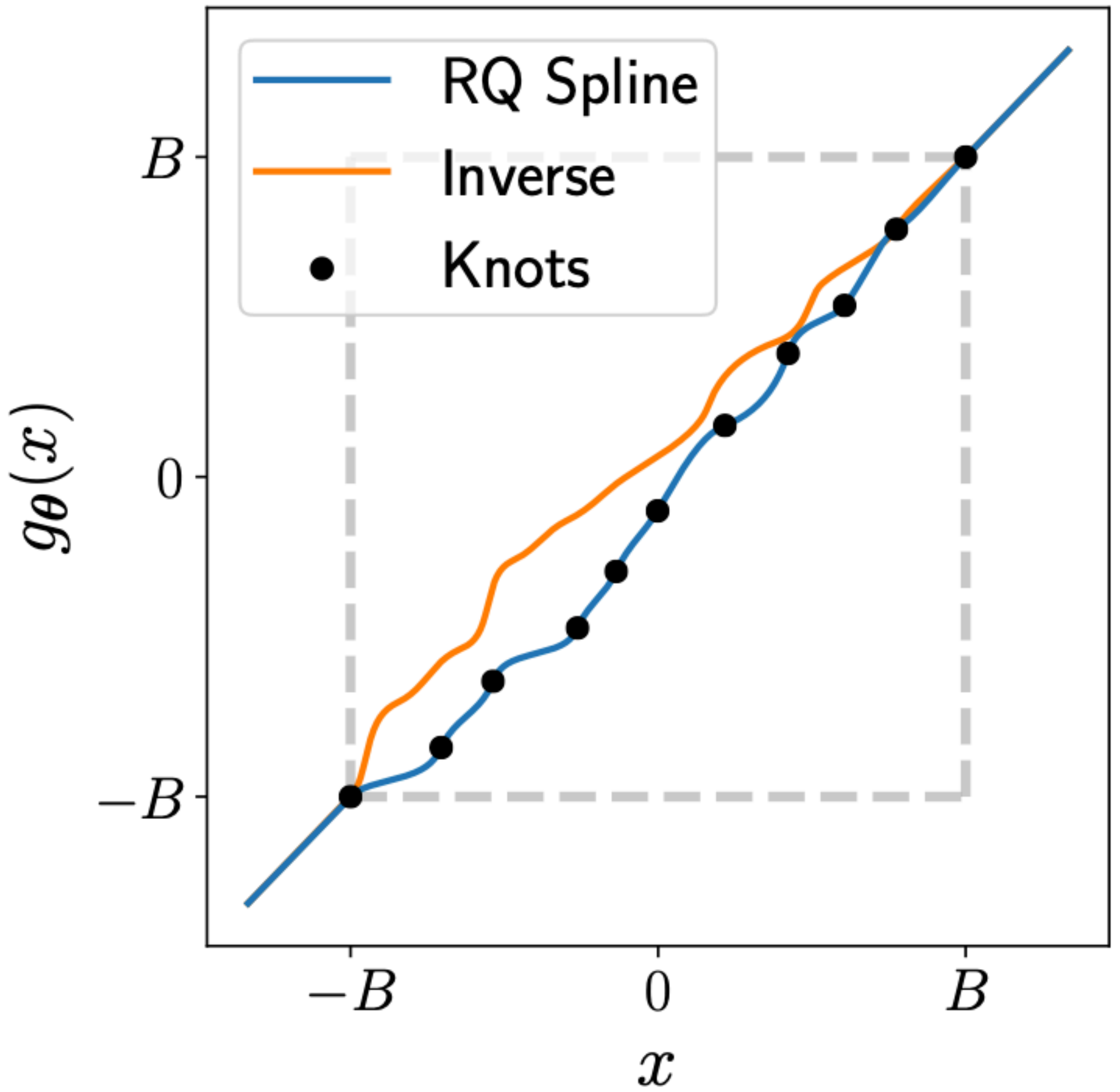


Example mapping g



$$Y := g(Z, d) = g(Z) = \sqrt{|Z|}\,\mathrm{sign}(Z)$$



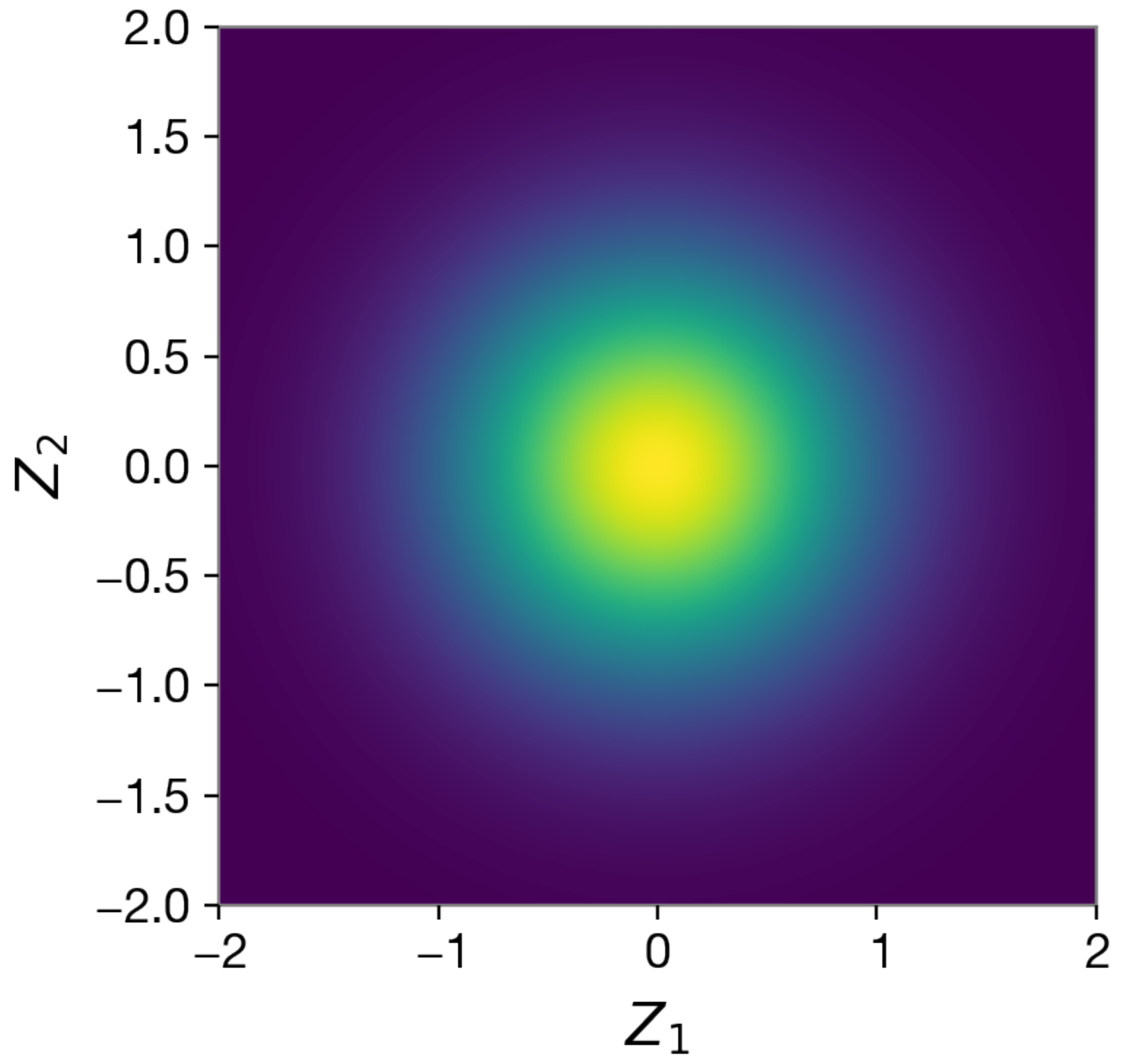


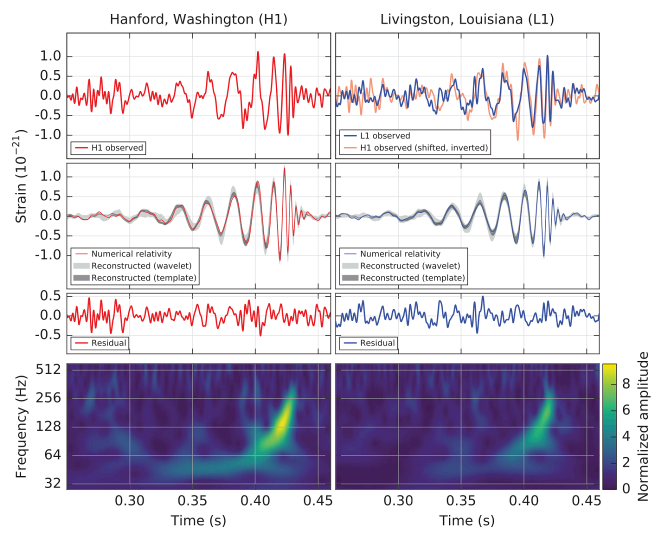
Spline points can approximate many functions

1906.04032

Mapping in higher dimensions

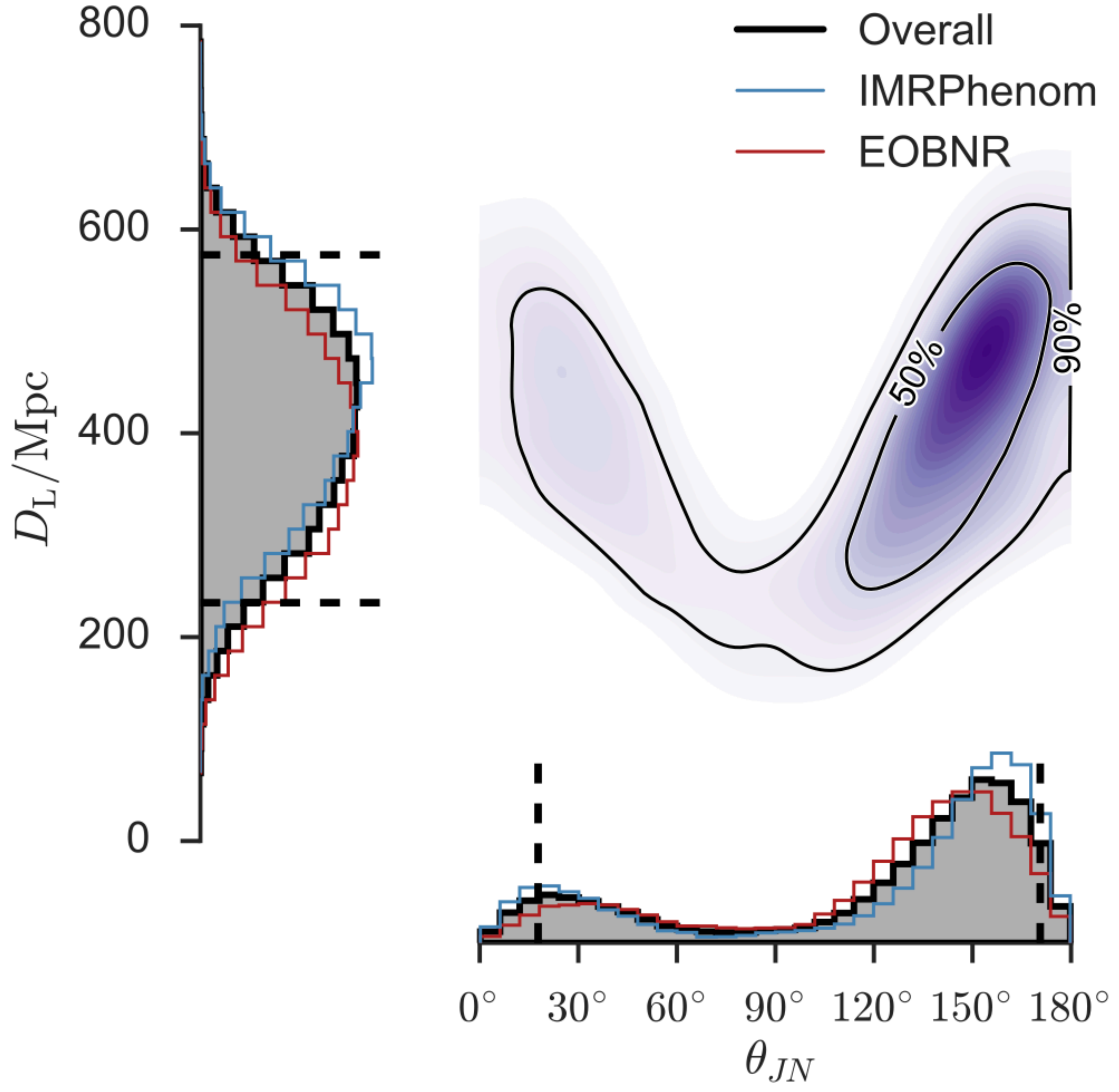
$$q(Y\,|\,d) = \mathcal{N}(0,1)^D\left(\vec{g}^{-1}(\vec{Y})\right)\left|\det J_{\vec{g}^{-1}}\right|$$





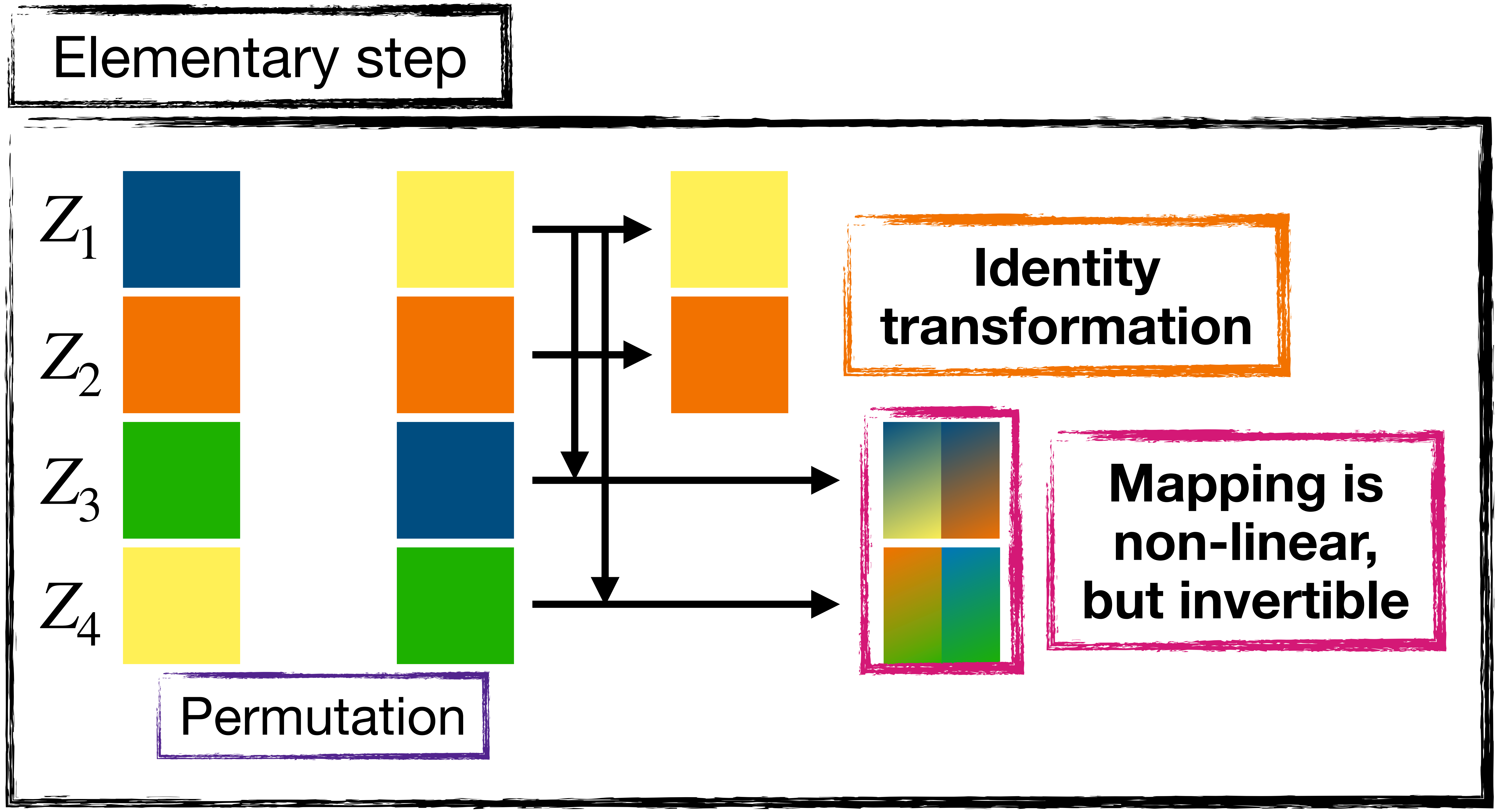
Mapping g depends on data

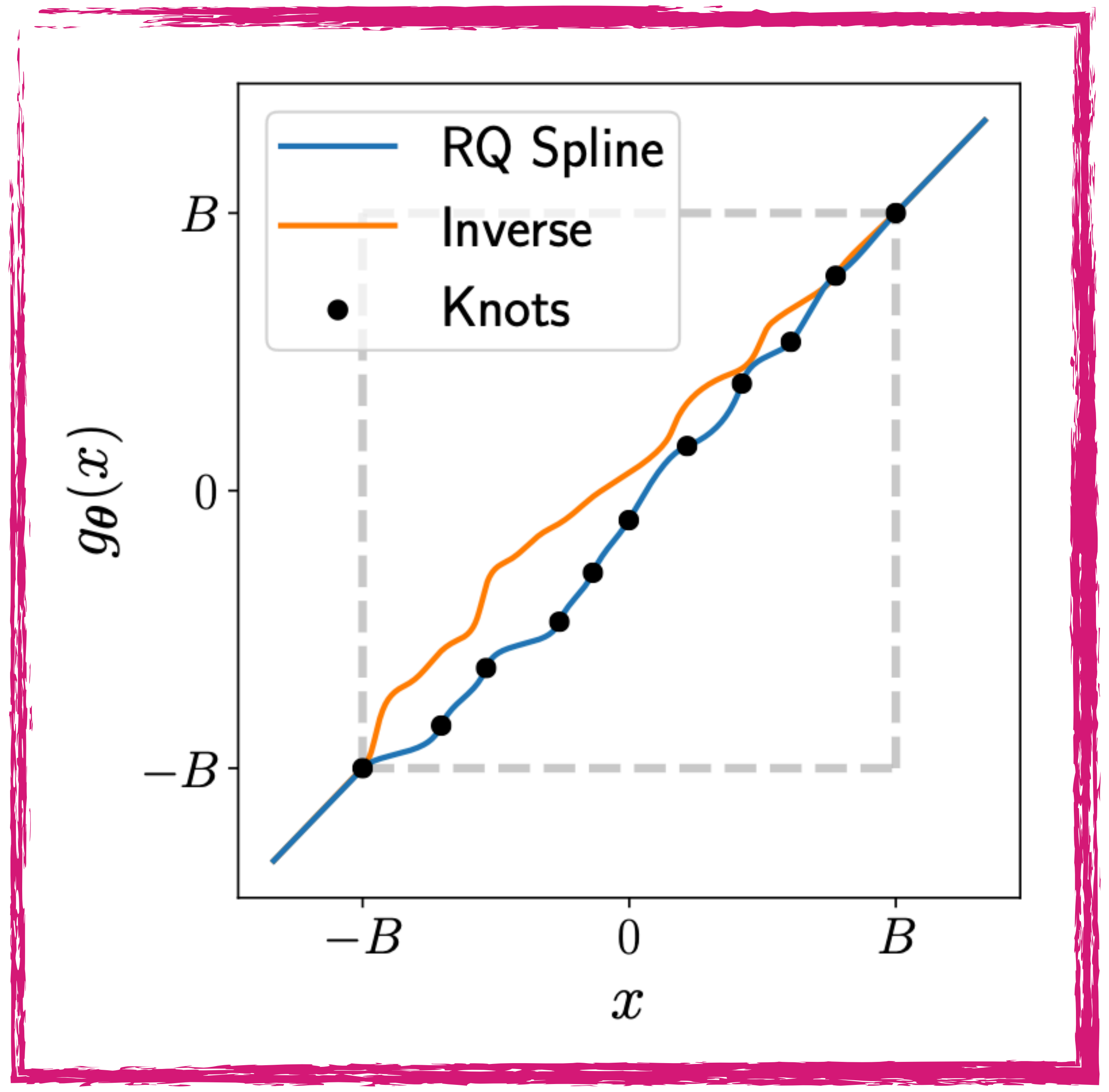
$$\vec{Y} := \vec{g}(\vec{Z}, d)$$



1908.09257

Elementary step of the normalising flow

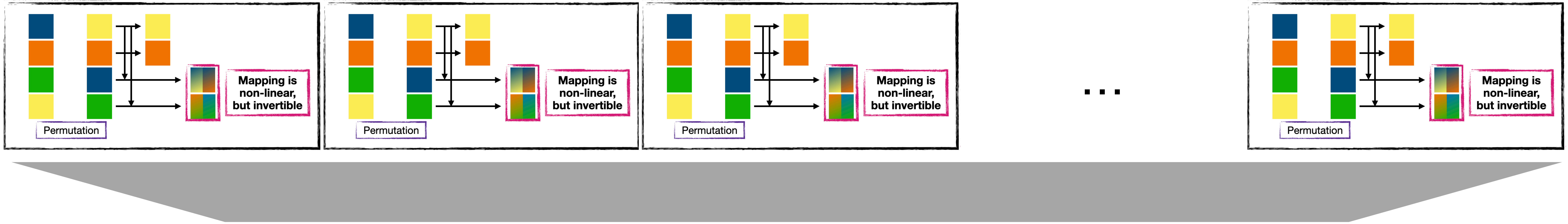


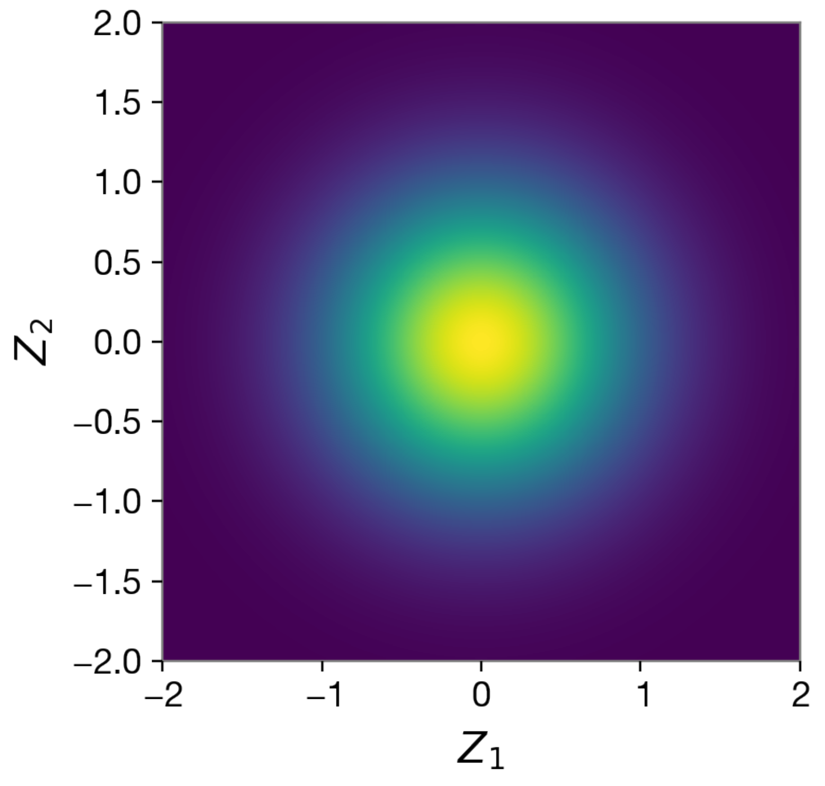


1906.04032

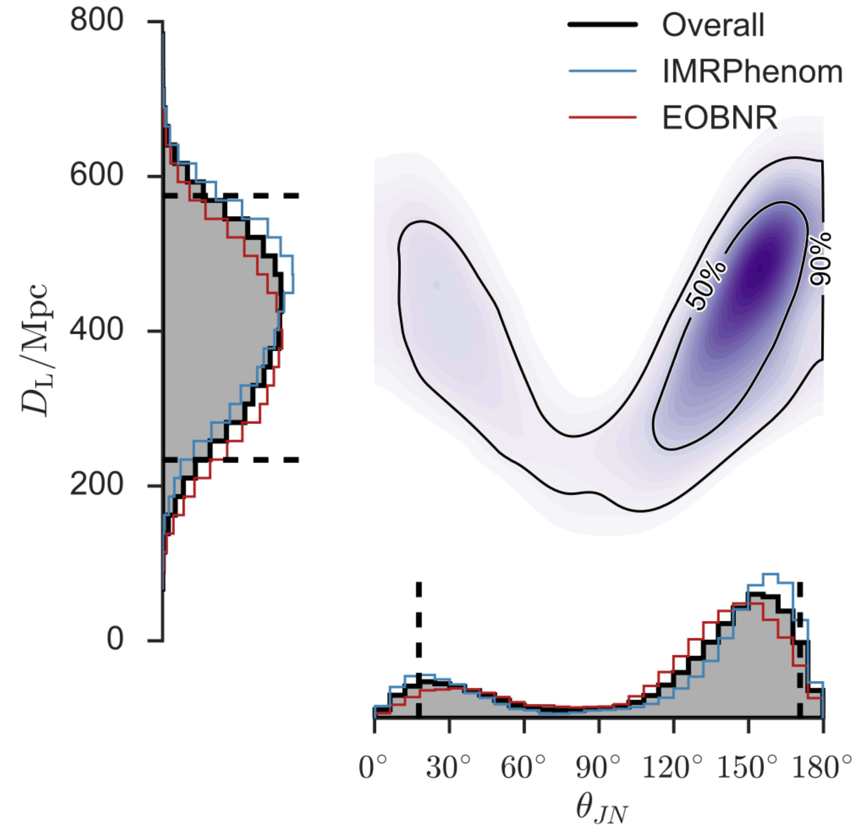
Normalising flow: Architecture

Many $\mathcal{O}(10)$
elementary steps



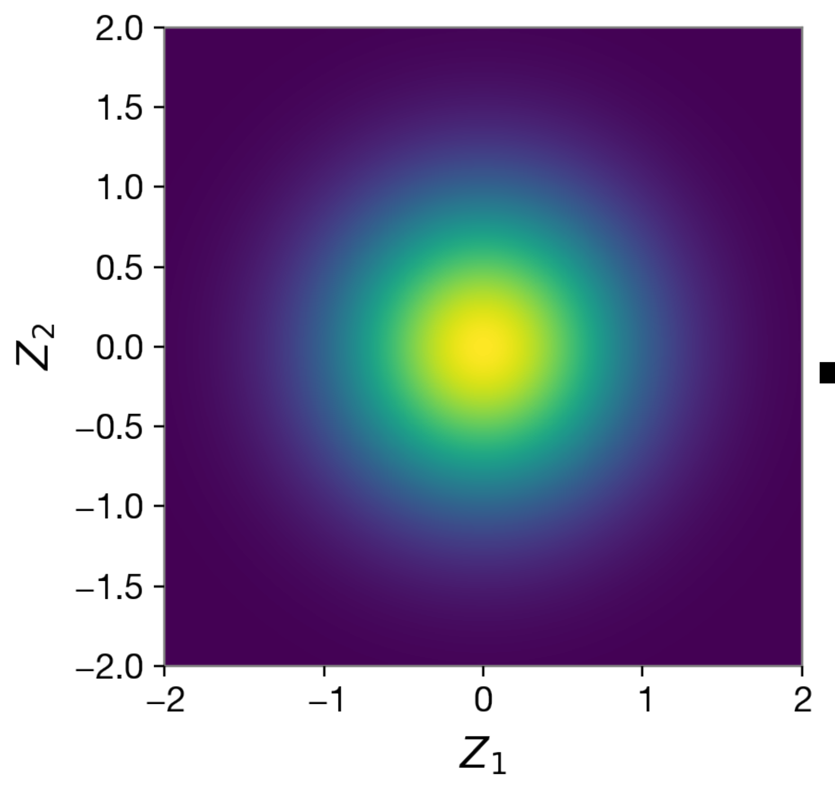


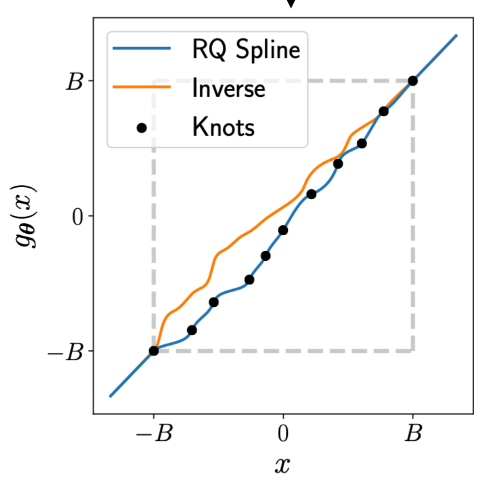
Normalising flow

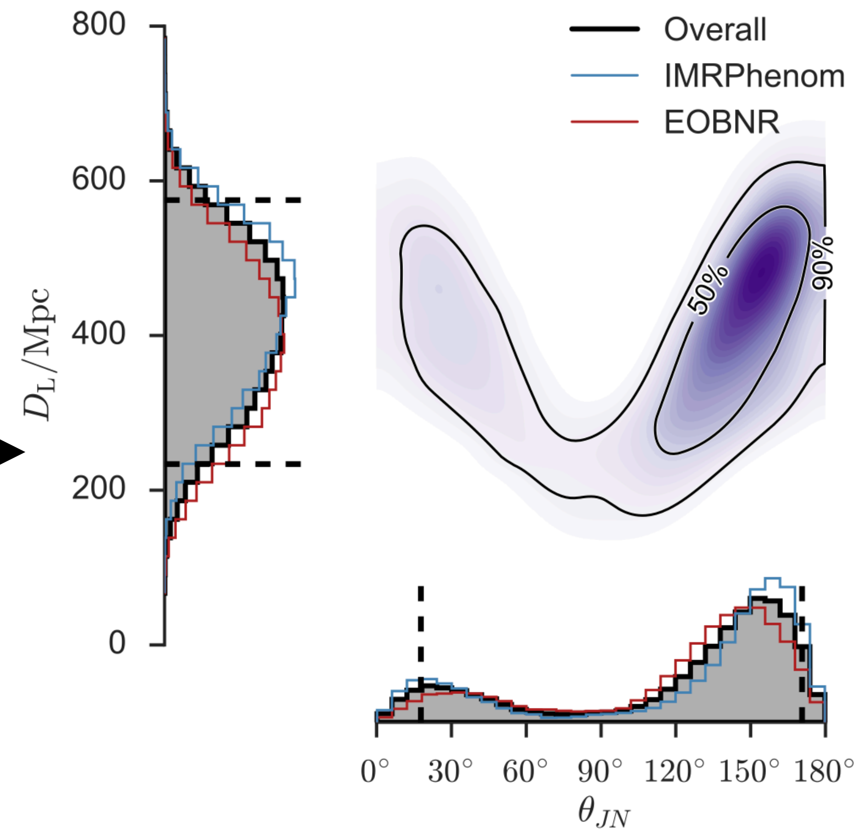


Training the flow: Fixing the parameters of the neural network

Data







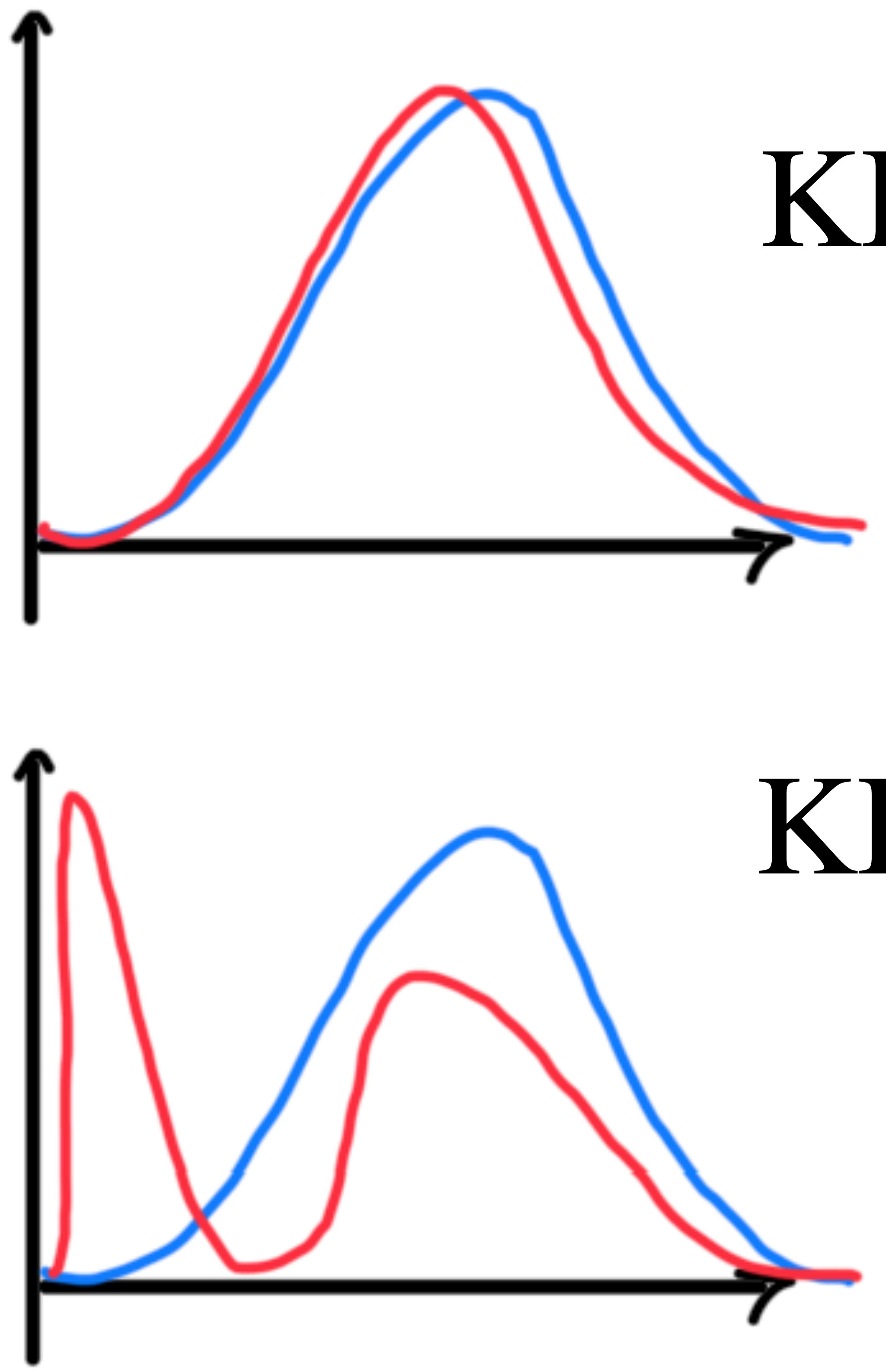
Normalising flow

Neural network parameters describe the spline(s)

Training the flow: Fixing the parameters of the neural network

- Spline points are parametrised by the neural network parameters → How to fix them?
- Define the "difference" between two distributions
- Define a loss according to this difference
- Tune the network parameters $\mathcal{O}(10^{6-9})$ s.t. this loss is minimised (very difficult task, since the problem has very high dimension)

Kullback-Leibler divergence



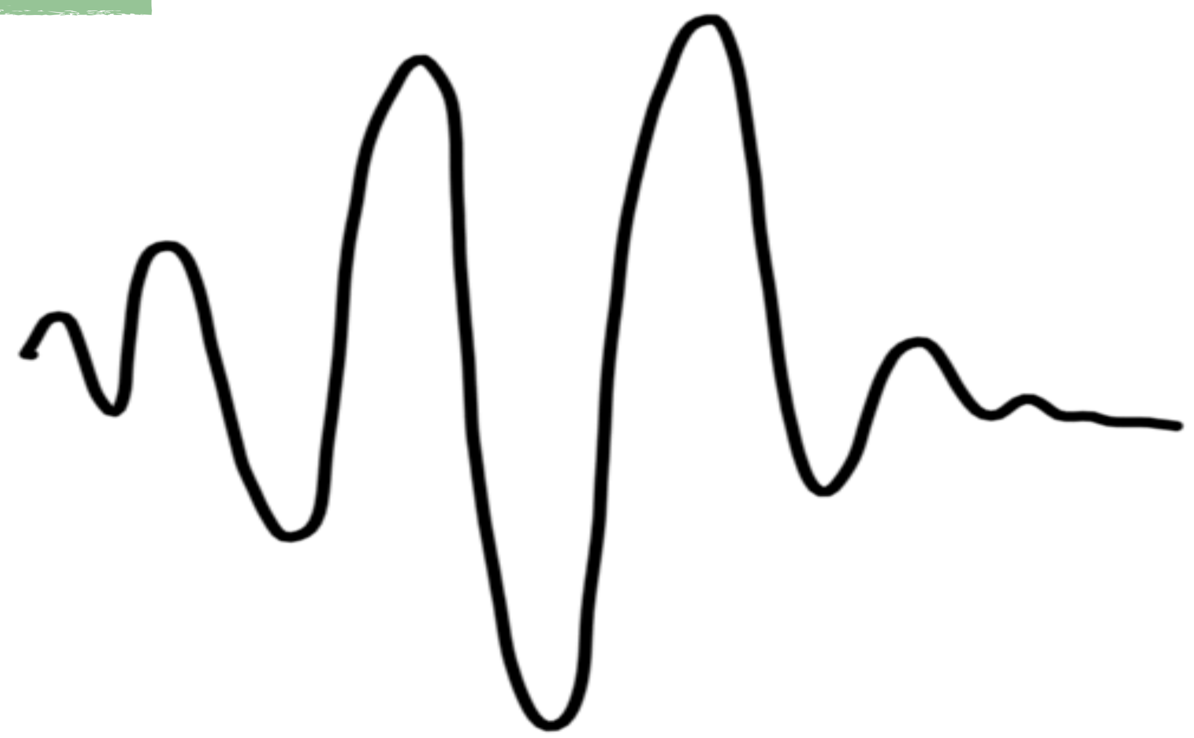
$$\mathrm{KL}(p|q) \approx 0$$

$$\mathrm{KL}(p|q) > 0$$

Application of a normalising flow: DINGO

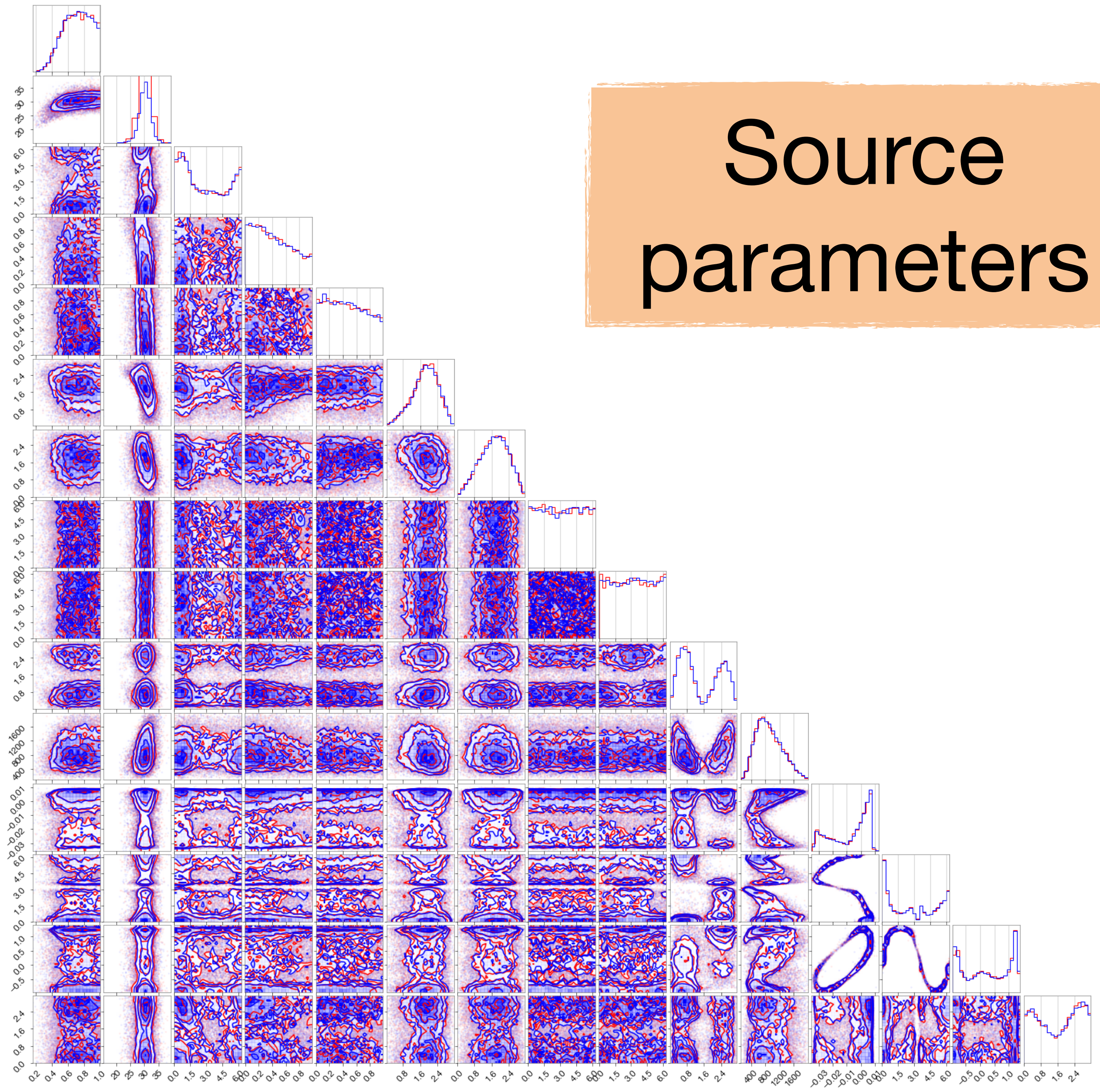
- Demonstrated to allow for fast and efficient sampling in case of individual event parameters *(Green et al. 2002.07656, Green et al. 2008.03312, Dax et al. 2106.12594, Dax et al. 2111.13139)*
- Distribution learned $p(\theta \,|\, \text{strain})$

Strain



DINGO

= Deep INference for Gw Observations



Source parameters

Training DINGO

- Generating the training data to extremize the given loss function, s.t. $q(\theta \,|\, \text{strain})$ converges towards the true posterior distribution

Expectation value over

Event parameters such as masses, distance

Noise realisations

Hidden in here are the network parameters

$$\text{Loss} = \mathbb{E}_{p(\theta)} \mathbb{E}_{p(d|\theta)} \left[-\log q(\theta \,|\, d) \right]$$

- Modify the network parameters such that loss is minimised

Training DINGO

- Generating the training data to extremize some given loss function, s.t. $q(\theta \mid \text{strain})$ converges towards the true posterior distribution

$$\text{Loss} = \mathbb{E}_{p(\theta)}\mathbb{E}_{p(d|\theta)}\left[-\log q(\theta \mid d)\right]$$

Bayes' theorem

$$= \mathbb{E}_{p(d)}\mathbb{E}_{p(\theta|d)}\left[-\log q(\theta \mid d)\right]$$

$$= \mathbb{E}_{p(d)}\left[\mathrm{KL}(p \| q)\right] + \text{constant}$$

Kullback-Leibler divergence

Measures the difference between 2 distributions

Likelihood-free inference

Expectation value over

Event parameters such as masses, distance

Noise realisations

$$\text{Loss} = \mathbb{E}_{p(\theta)}\mathbb{E}_{p(d|\theta)}\left[-\log q(\theta \,|\, d)\right]$$

- No need for evaluating a likelihood or producing posterior samples!
- Go beyond the approximation of Gaussian stationary noise

Only simulation of datasets is necessary (likelihood-free inference)

Training DINGO

- Generating the training data to extremize some given loss function, s.t. $q(\theta\,|\,\text{strain})$ converges towards the true posterior distribution

Expectation value over

Event parameters such as masses, distance

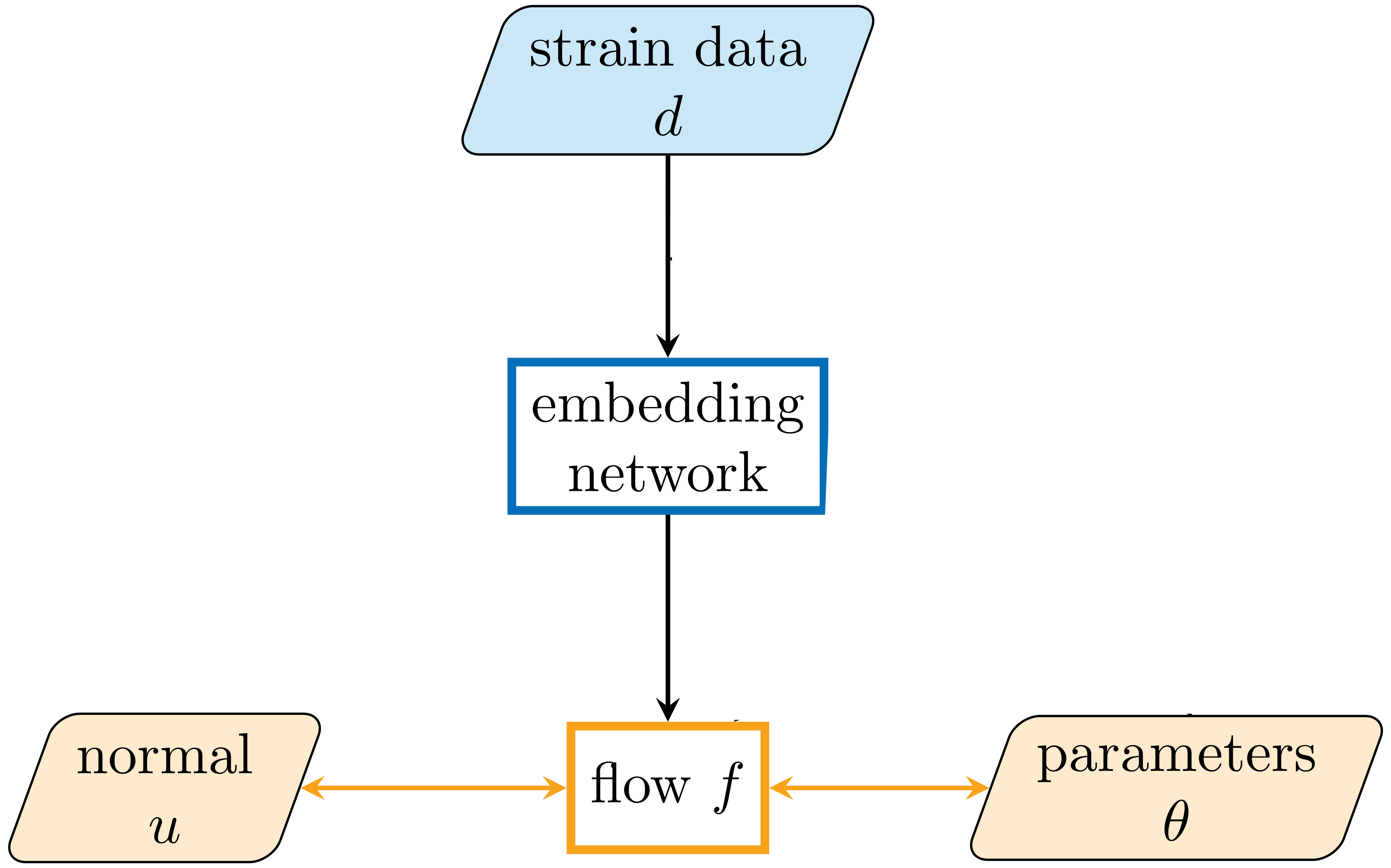
Noise realisations

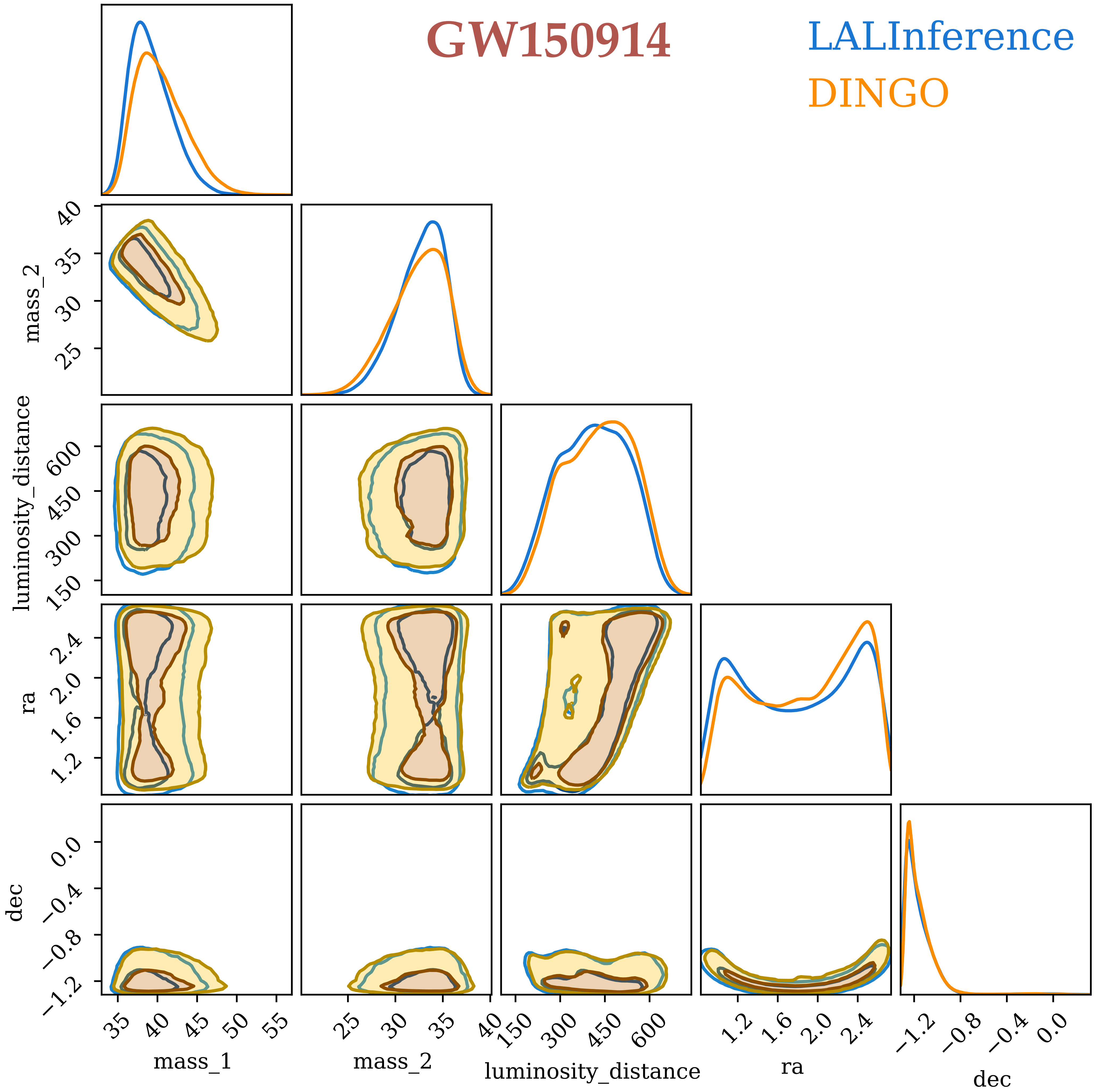
$$\text{Loss} = \mathbb{E}_{p(\theta)}\mathbb{E}_{p(d|\theta)}\left[-\log q(\theta\,|\,d)\right]$$

- Training time ~2 weeks
- 1.31×10^8 learnable parameters for 2 detectors (1.42×10^8 for 3)

DINGO: First results

- 15 dimensional parameter space
- Trained from $\mathcal{O}(10^6)$ samples
- Posterior samples from a fixed PSD in seconds!

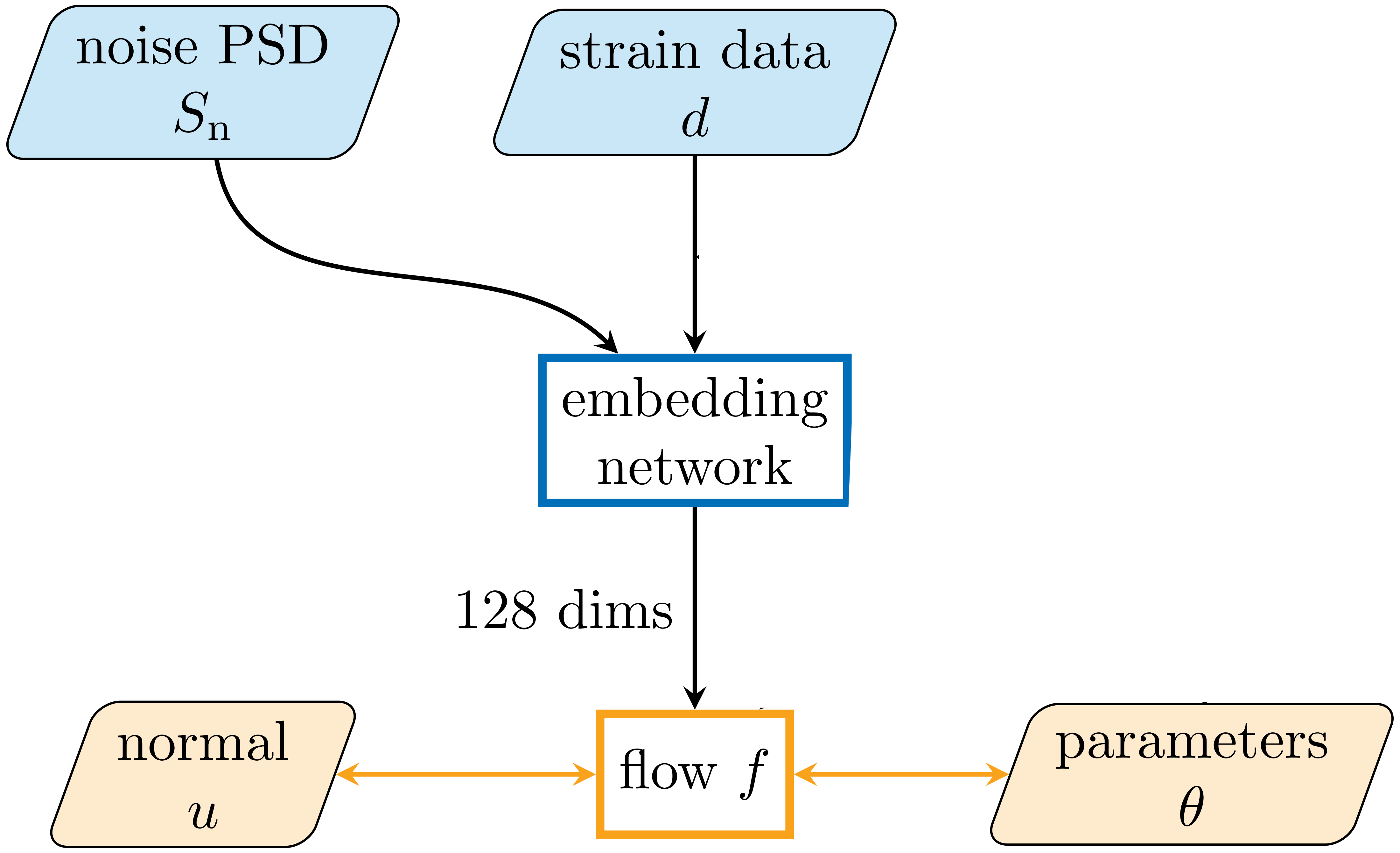




Noise curve as input

Dax et al. (PRL 2021)

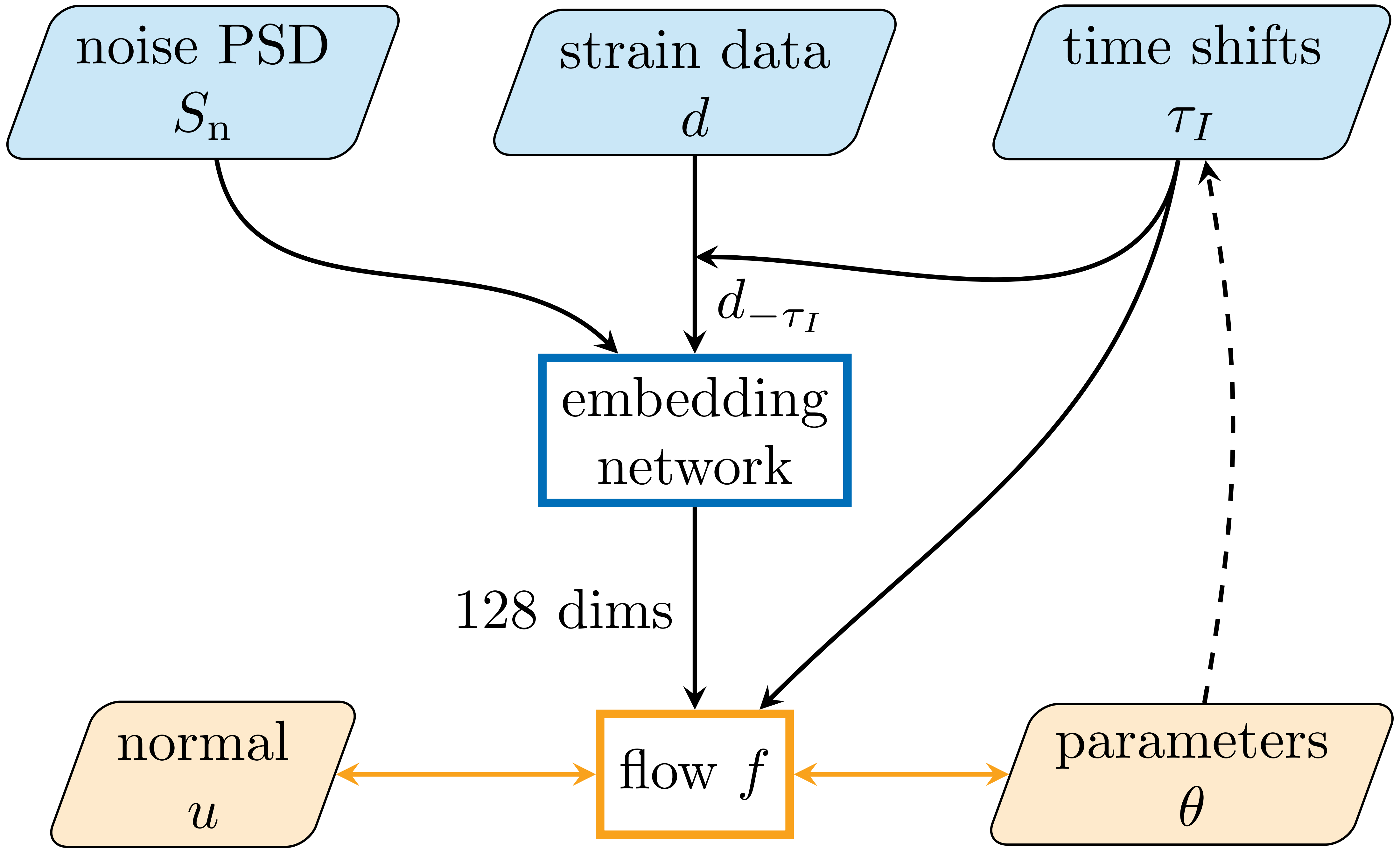
- Motivation: The PSD is time-dependent
- Training on PSD calculated from real data
- PSD is part of input data (loss is modified accordingly)



Standardising the time of arrival

Dax et al. (PRL 2021)

- Standardising the time of arrival makes training easier (dimension of data is reduced)
- Recursive process, shown to converge within $\mathcal{O}(10)$ iterations



DINGO: Results

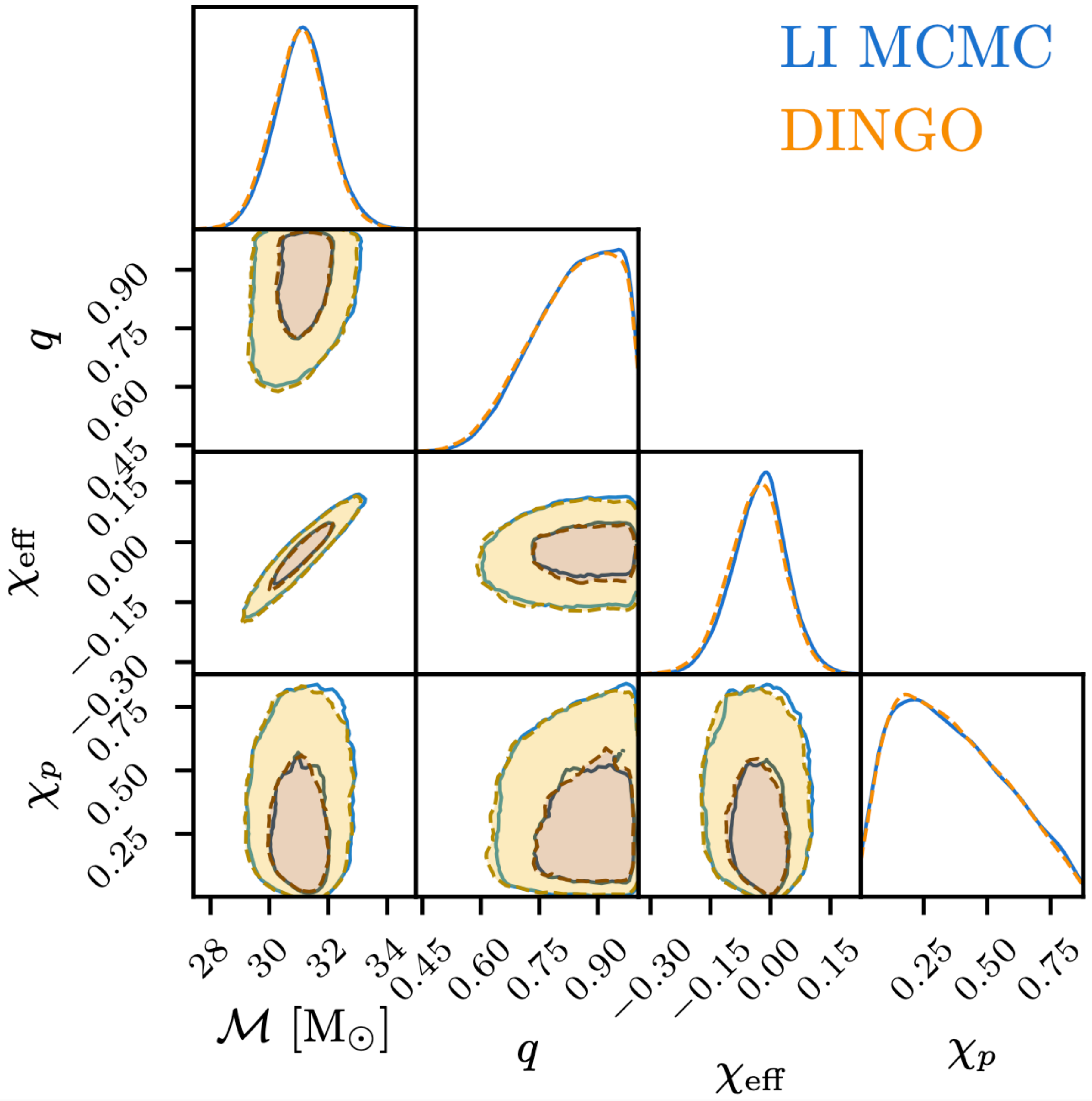
2106.12594

	m_1	m_2	ϕ_0	d_L	a_1	a_2	θ_1	θ_2	ϕ_{12}	ϕ_{JL}	θ_{JN}	ψ	α	δ
GW150914	0.8	1.1	0.2	0.8	0.2	0.3	0.5	0.5	0.1	0.3	0.8	0.2	0.7	1.4
GW151012	2.7	1.6	0.1	0.9	0.4	0.2	0.5	0.5	0.1	0.1	0.6	0.1	1.4	0.5
GW170104	6.4	2.6	0.2	0.4	0.7	0.1	0.7	0.4	0.1	0.1	0.3	0.3	0.8	0.6
GW170729	0.9	1.5	0.4	6.3	0.2	0.2	1.0	0.8	0.2	0.3	3.4	0.3	1.2	1.2
GW170809	0.5	0.8	0.1	0.5	0.2	0.1	0.4	0.4	0.1	0.5	1.4	0.2	2.2	5.5
GW170814	1.2	1.3	0.2	1.5	0.2	0.2	0.4	0.3	0.2	1.4	1.4	1.2	2.5	2.0
GW170818	1.6	1.3	0.2	1.1	1.0	0.2	1.9	0.5	0.1	2.4	1.8	0.4	3.8	2.4
GW170823	0.5	0.6	0.1	0.9	0.2	0.2	0.4	0.2	0.2	0.2	0.5	0.2	0.4	0.4

JS divergence [$\times 10^{-3}$ nat] (colour scale 0–20)

Expect: 0.7

Generating samples is easy (and very fast)

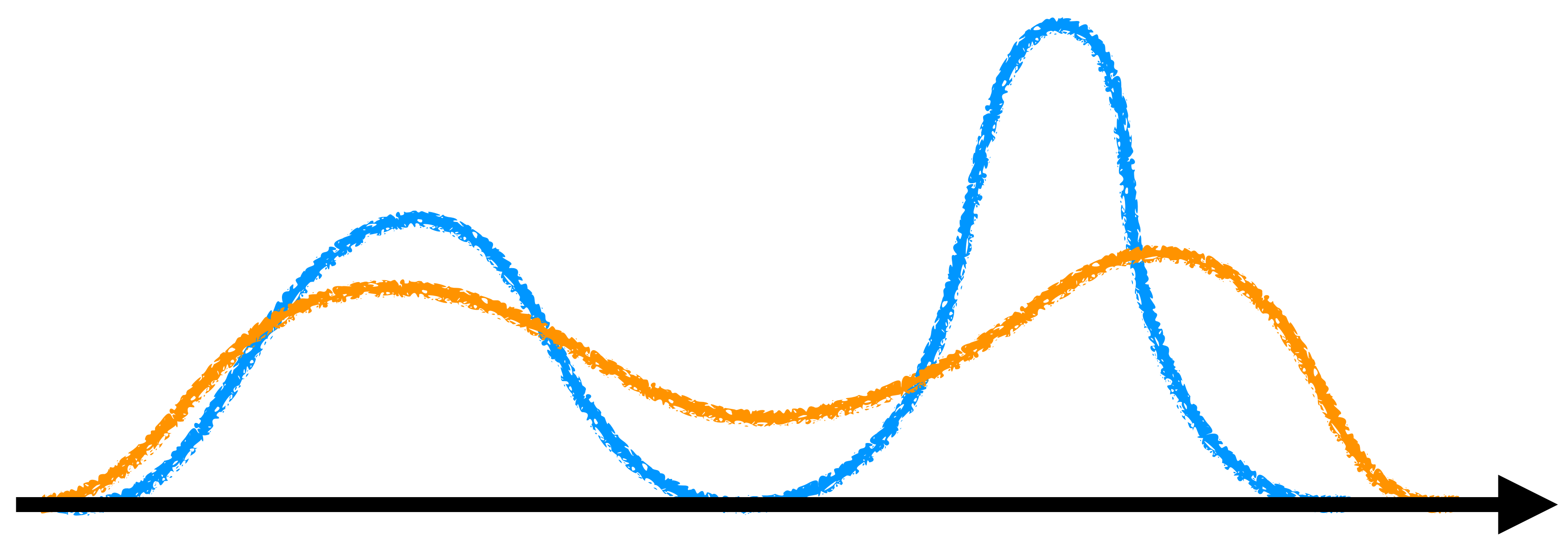


GW150914

Sanity check

2210.05686

- The exact likelihood of each sample produced from the flow is known →
  - Importance sampling: Recover exact results with an explicit likelihood by taking the DINGO posterior as a proposal distribution



Thank you!