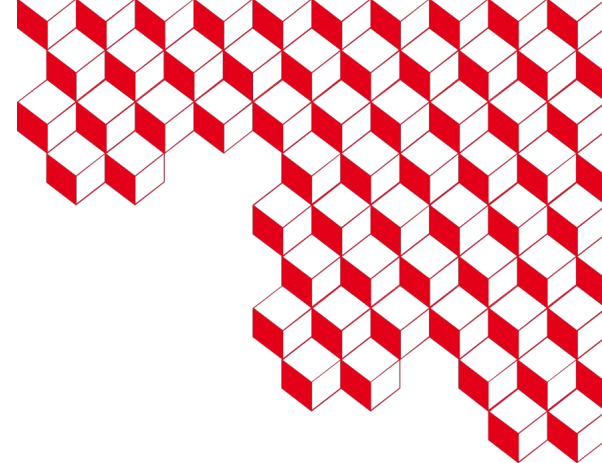




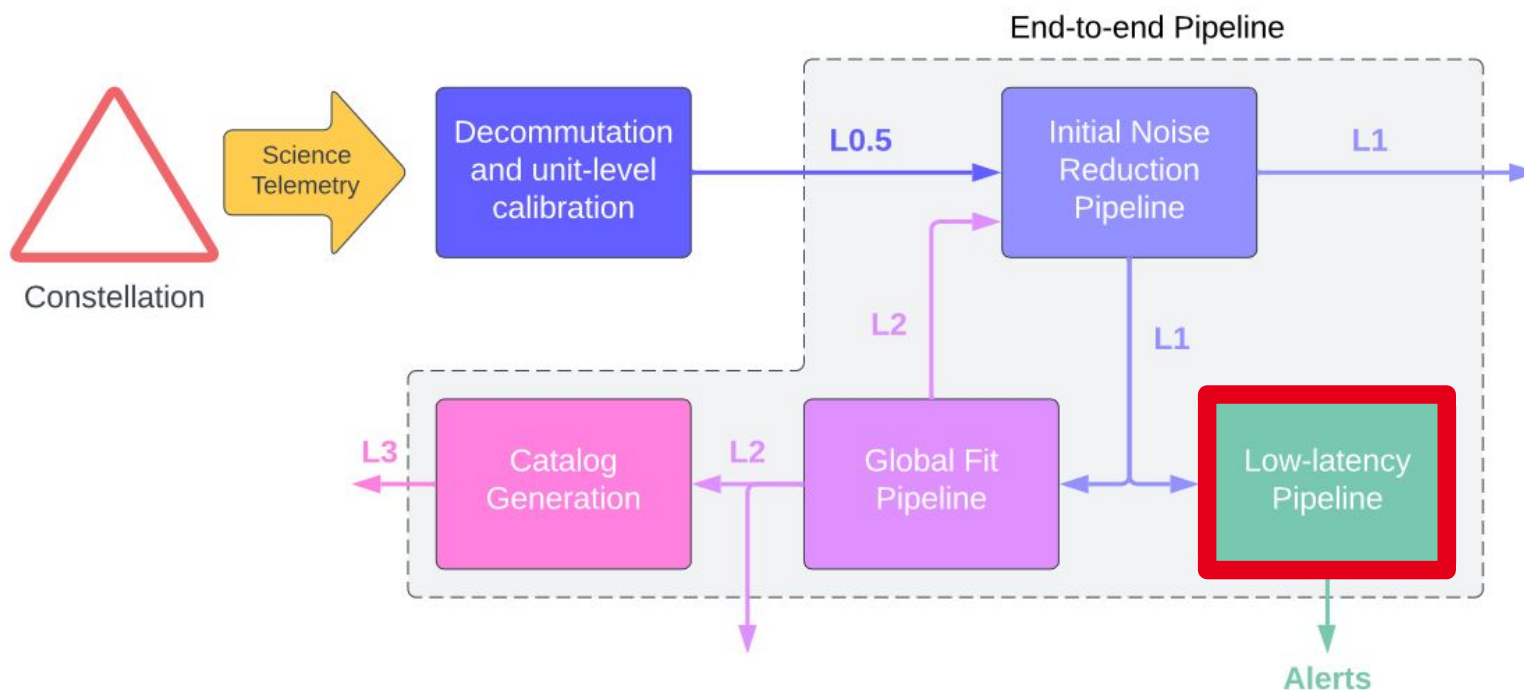
A simulation-based inference pipeline for fast parameter estimation of massive black hole binaries with LISA data

Louis Le Saulnier, Tobías Liaudat, Jean-Baptiste Bayle



Context: LISA Low Latency Alert Pipeline

Massive black hole binaries detections (MBHB) & low latency alert pipeline (LLAP):

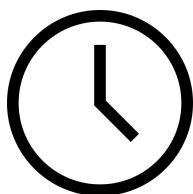


The low latency alert pipeline:

Goal : send live alerts of MBHB events

Alert : tells within 1h after data reception if a MBHB has been detected and gives its parameter estimation (sky location)

Contribution to science objectives : helps the community to observe electromagnetic counterparts, supports multimessenger astronomy



Strong time constraints on our algorithms : **Alerts have to be given before the mergers**

Challenges for the LLAP

The LLAP challenges:

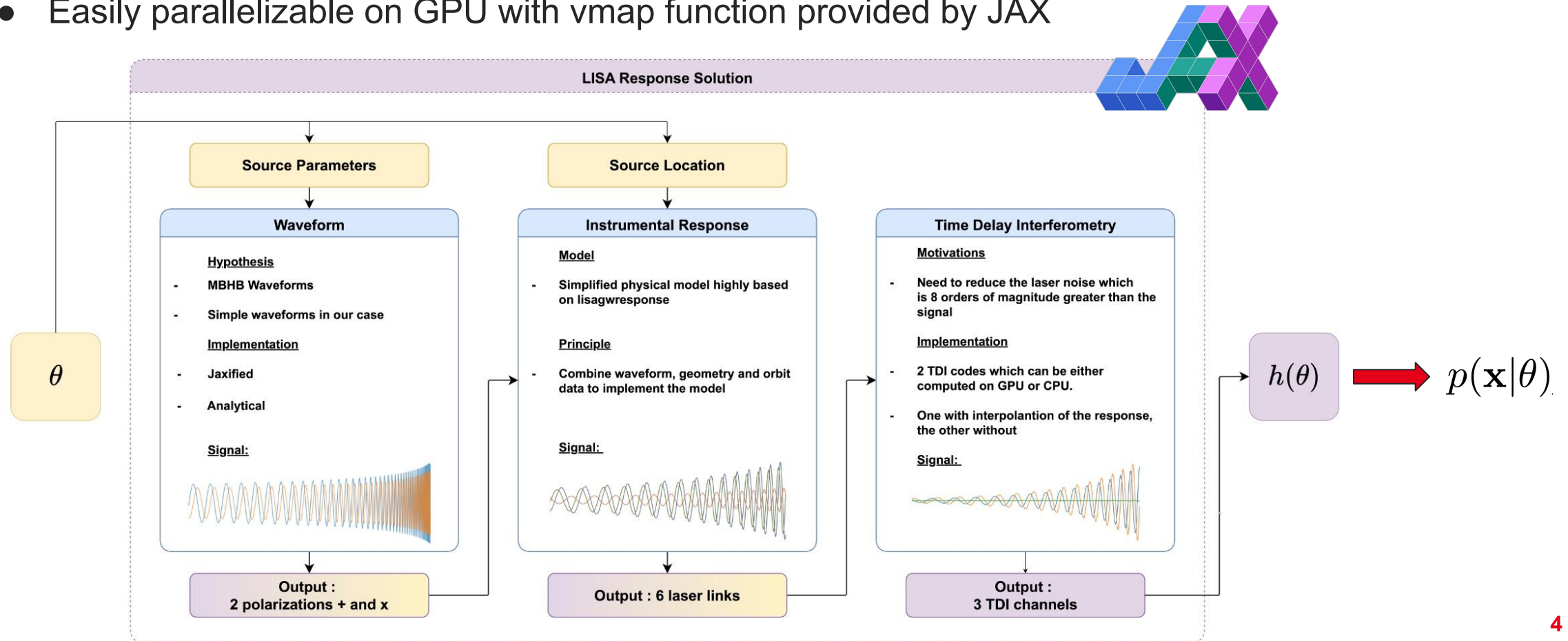
LISA data analysis and especially the low-latency alert pipeline face new challenges compared to ground-based GW detectors.

- **Time constraints** to send real-time alerts with the best possible parameter estimation before the merger of MBHBs.
- **Multimodality** of the posterior. It is degenerated and comprises several modes, making the parameter estimation hard to perform.
- **Gaps** (missing data), **Glitches** (sudden jumps or transients in the data) and **sequential data acquisition** add constraints to the pipeline.
- **Non stationary noise** due to the orbits of the constellation.
- **Long time series** handling in order to get higher signal-to-noise-ratio (SNR).

Accelerated LISA JAX response in time domain

Features:

- Fully differentiable from waveform to likelihood
- Accelerated with JIT
- Easily parallelizable on GPU with vmap function provided by JAX



Tests of standard parameter estimation methods

Definition

- We aim estimate the parameters θ of massive black hole binaries (MBHB), especially the coalescence time and the sky location
- Here we call $\mathbf{X}$ the data, i.e. LISA data observations after pre-treatment (TDI data)

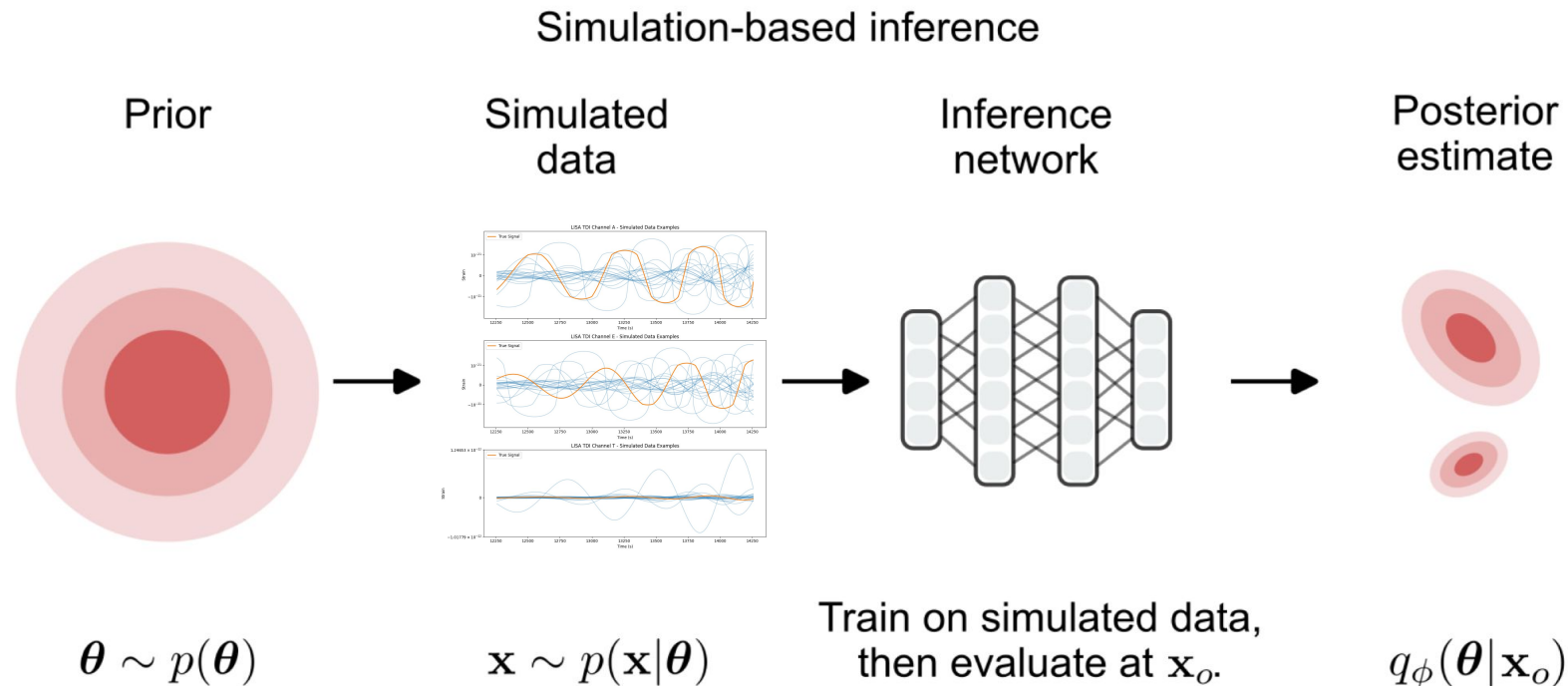
Bayesian Inference

- The goal is to find the posterior distribution of the parameters: $p(\theta|\mathbf{x}) \propto p(\mathbf{x}|\theta)p(\theta)$
 - $p(\theta)$ the prior distribution, mostly uniform laws on the parameters in our case
 - $p(\mathbf{x}|\theta)$ the likelihood function
- Based on Monte Carlo Markov Chain (MCMC) methods
- The posterior distribution gives parameter estimation and error bars
- Has convergence guarantees but **needs to evaluate a consequent amount of likelihoods**

Parameter estimation methods

Simulation based inference

- Training a Neural Posterior Estimator $q_\phi(\theta|\mathbf{x})$ that approximate the posterior distribution over the parameters
- Once the network is trained, the posterior is fully amortized, for any observation $\mathbf{x}_o$ the posterior $q_\phi(\theta|\mathbf{x}_o)$ can be obtained after one forward pass.



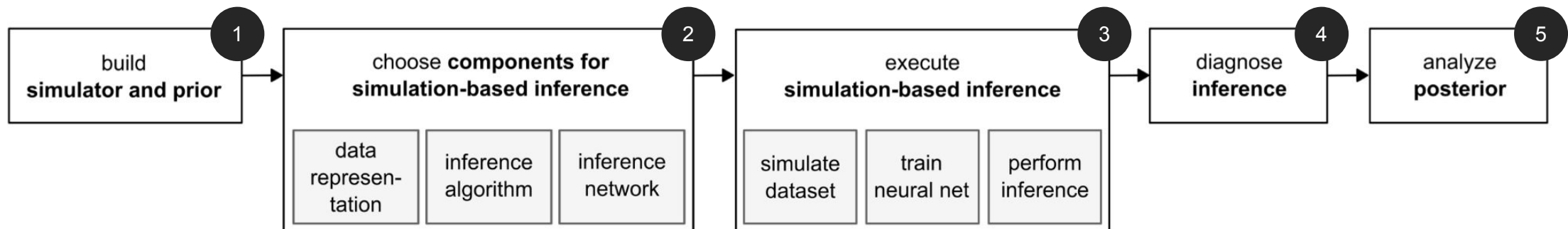
Development of a SBI pipeline

Current goal

Build a full **modular** pipeline that performs posterior estimation using SBI but also deals with the needs of LISA data analysis and especially LLAP.

Four essential steps have to be implemented

- The simulation part with a JAX LISA response and a dataset generator
- Data representation, how we choose summary statistics with LISA data ?
- GPU compatible training script and dealing with large datasets
- Results and diagnostics



Dataset generation

From simulator to datasets:

- In SBI the simulator can be arbitrary, we only need to generate paired training datasets $\mathcal{D} = \{(\mathbf{x}_i, \theta_i)\}_{i=1, \dots, N}$
- Scripts to generate, extend and concatenate datasets if compatibles
- Stored in hdf5 files with all the attributes used to generate
- Fully reproducible data generation with seed management and attribute reading functions



The screenshot shows a JupyterLab interface. On the left, a file browser displays a folder named 'dataset_refactored_priors_precomp...' containing subfolders 'thetas' and 'xs'. On the right, a panel shows the metadata for the file 'dataset_refactored_priors_precomputed_summaries.h5'. The metadata is organized into sections: 'Group' (Path, File path, Raw), 'Attributes' (config, is_summary_dataset, source_dataset), and 'Display'/'Inspect' buttons.

dataset_refactored_priors_precomputed_summaries.h5	
Group	
Path	/
File path	dataset_refactored_priors_precomputed_summaries.h5
Raw	► Inspect
Attributes	
config	<pre>"{"orbit_path": "/Users/l1283482/Documents/Projets/LISA/lisajax_sbi/data/esa-orbits.h5", "output_dir": "/Users/l1283482/Documents/Projets/LISA/lisajax_sbi/generated_datasets", "output_name": "dataset_refactored_priors.h5", "n_ts": 18000, "num_simulations": 10000, "dt": 2.0, "add_noise": true, "rand_seed": 6, "tdi_channel": "AET2", "new_dataset": true, "first_file": "first_file.h5", "params": [{"type": "uniform", "name": "coalescence_time", "low": 50260.0, "high": 64660.0}, {"type": "log_uniform", "name": "total_mass", "low": 1000000.0, "high": 20000000.0}, {"type": "uniform", "name": "reduced_mass", "low": 0.04535147, "high": 0.25}, {"type": "cos_uniform", "name": "inclination", "low": 0.0, "high": 3.141592653589793}, {"type": "log_uniform", "name": "luminosity_distance", "low": 1e+25, "high": 2e+29}, {"type": "uniform", "name": "initial_phase", "low": 0.0, "high": 6.283185307179586}, {"type": "uniform", "name": "lamb", "low": 0.0, "high": 6.283185307179586}, {"type": "sin_uniform", "name": "beta", "low": -1.5707963267948966, "high": 1.5707963267948966}]}</pre>
is_summary_dataset	true
source_dataset	"/Users/l1283482/Documents/Projets/LISA/lisajax_sbi/generated_datasets/dataset_refactored_priors.h5"

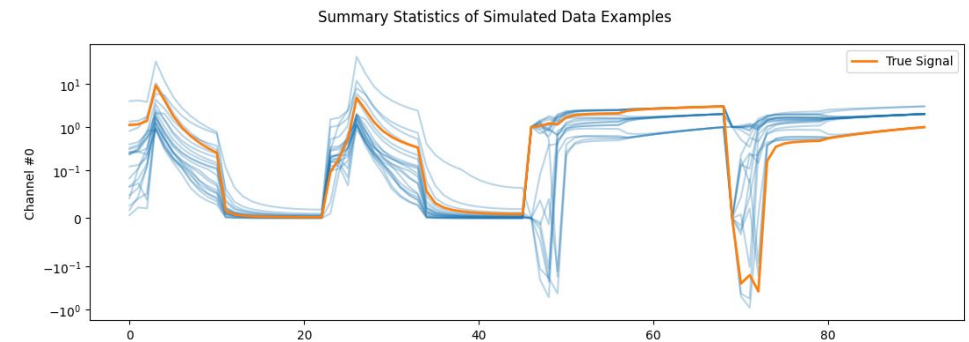
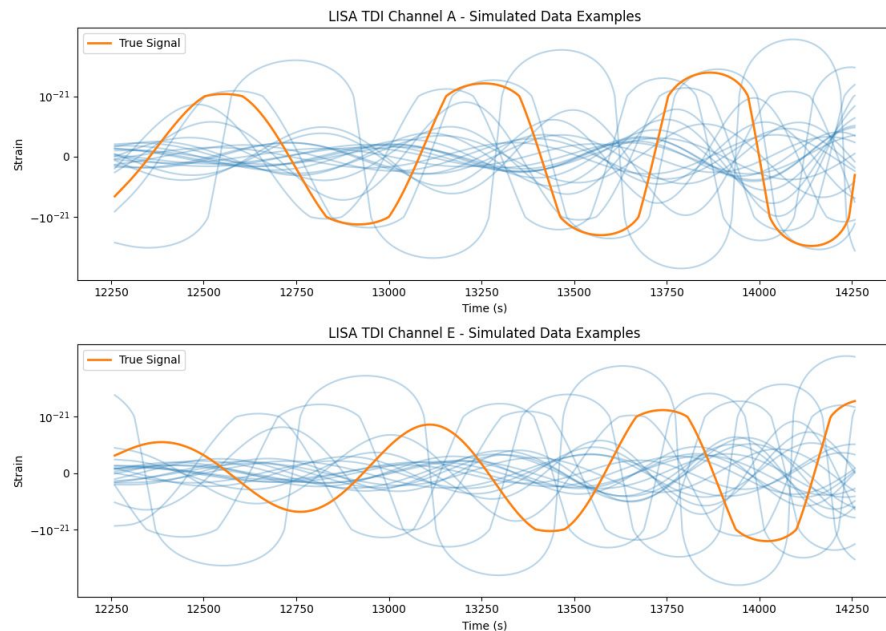
Summary statistics

Motivation:

Neural networks struggle with high dimensional input data. As we are working with time series with more than 1000 datapoints there is a **need to find new data representations of our signals**. It can be:

- Analytical summary statistics (Fourier transform, wavelet transform,...)
- Embedding neural networks (CNN, RNN, transformer,...)

First idea: Concatenating the module and the whitened phase of the Fourier transform of channels A and E

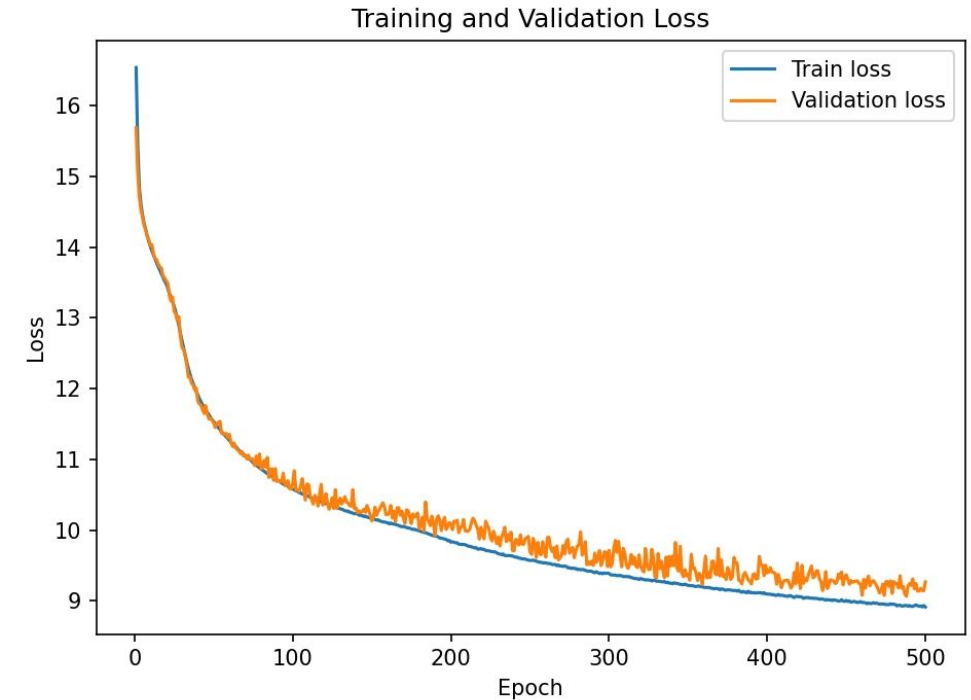


Training of the neural network

First training setup for testing:

Once the simulator and the data representations are chosen, the SBI toolbox provides built in models and tools for training making it easy to compute

- The neural networks used are normalizing flows, which learn how to transform a simple reference distribution into a complex one (posterior here)
 - Density estimator in our work: **Masked Autoregressive Flow** (standard one in SBI)
 - Loss: $\mathcal{L}(\phi) = \frac{1}{N} \sum -\log q_{\phi}(\theta_i | \mathbf{x}_i) + C$
- The training can be performed either on GPU and CPU even with a large amount of data.
- SBI allows to train embedding networks during the training of the density estimator



Loss curves provided by SBI toolbox

Sampling of the posterior

Sampling:

When the model is trained the posterior can be built and then sampled for any data observation $\mathbf{x}_o$.
The posterior is amortized for any observation, meaning **fast parameter estimation**.

Importance sampling:

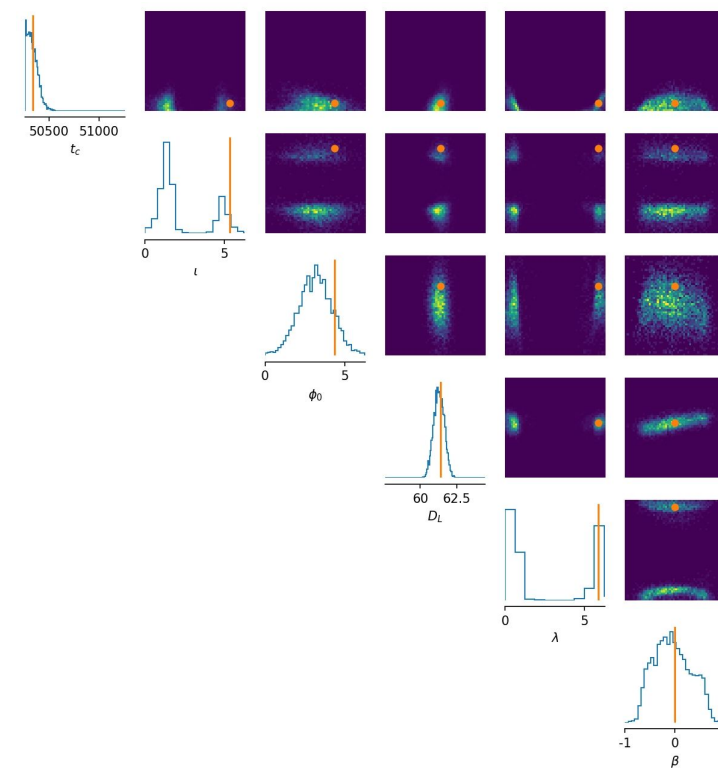
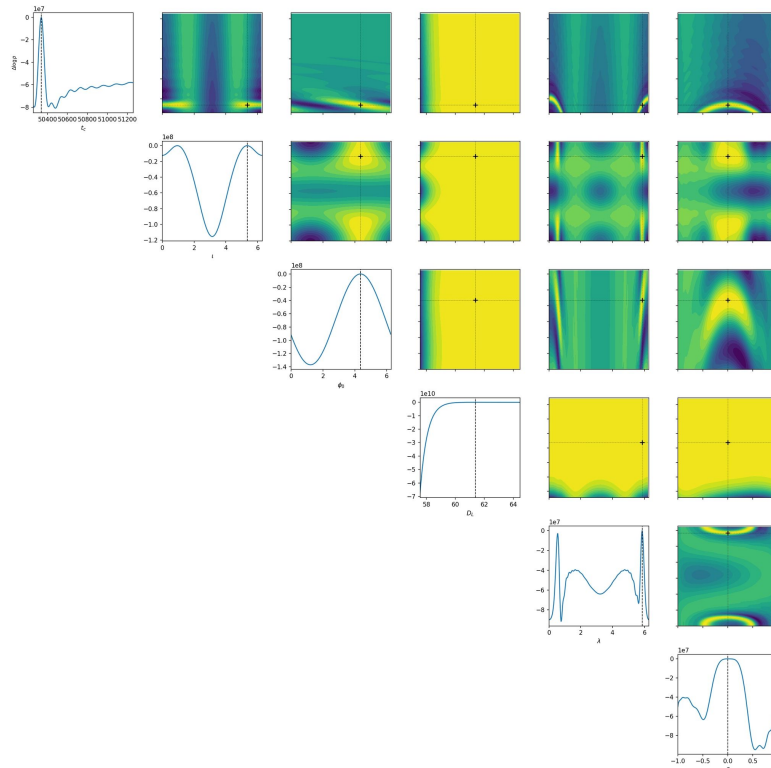
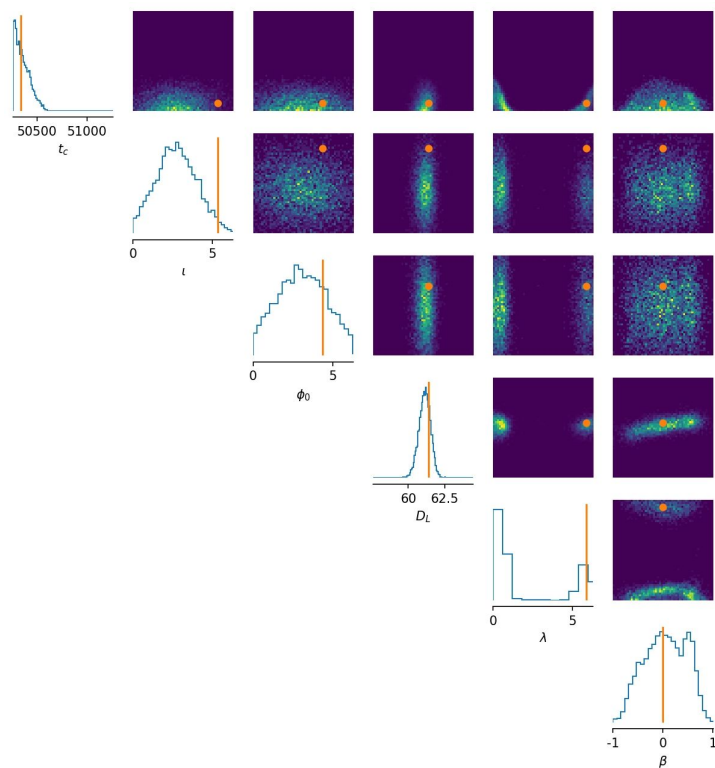
The correction of the posterior can still be improved at this step. It can be refined without modifying the weights of the trained model.

- **Idea:** Compare the “proposal” distribution $q_\phi(\theta|\mathbf{x}_o)$ to the “target” distribution $p(\theta|\mathbf{x})$
- Our fast LISA response enables the computation of the real posterior $p(\theta|\mathbf{x})$ with the likelihood and prior distributions
- Each sample of the “proposal” distribution will be attached to a weight w_i defined as: $w_i = \frac{p(\theta_i|\mathbf{x}_i)}{q_\phi(\theta_i|\mathbf{x}_i)}$
- Small improvement of the method, doesn't change significantly the results

Preliminary sampling results

Cornerplots of posterior samples for the first SBI runs:

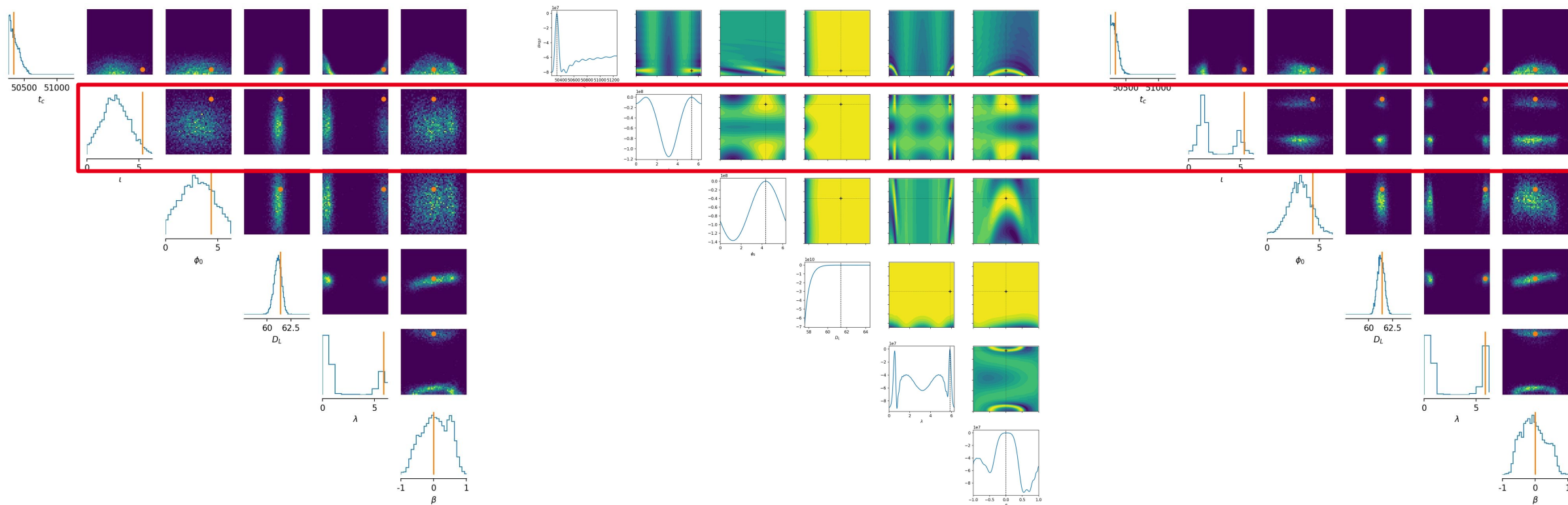
Example of a model trained on 100k simulations of 18000 time samples with a coalescence time window range of 1000 seconds, fixed masses and noise and using a MAF.



Preliminary sampling results

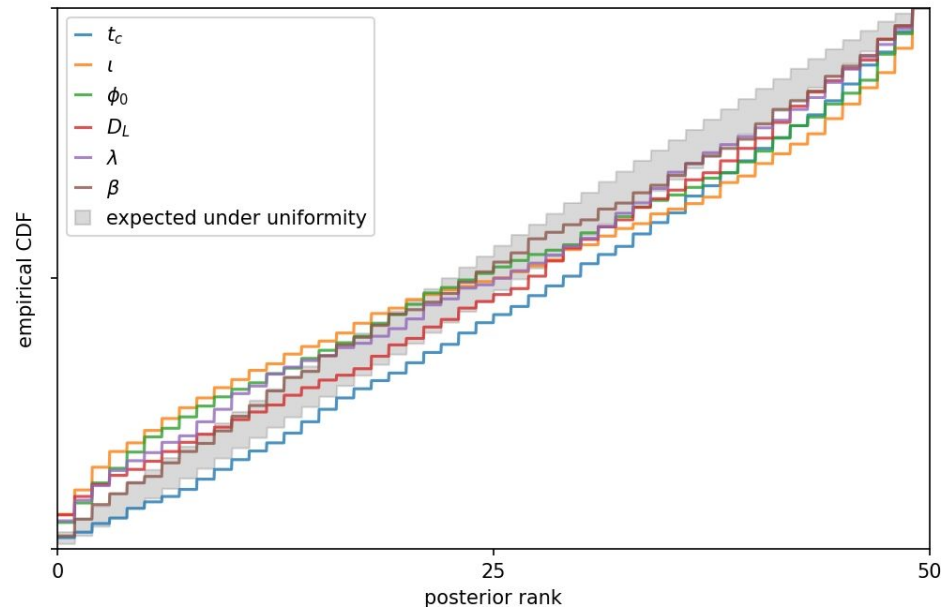
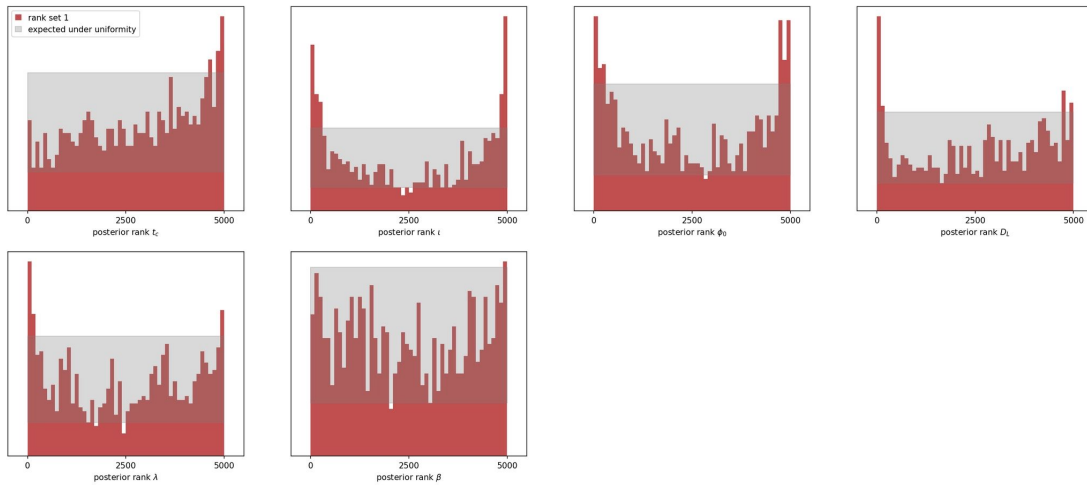
Cornerplots of posterior samples for the first SBI runs:

Example of a model trained on 100k simulations of 18000 time samples with a coalescence time window range of 1000 seconds, fixed masses and noise and using a MAF.

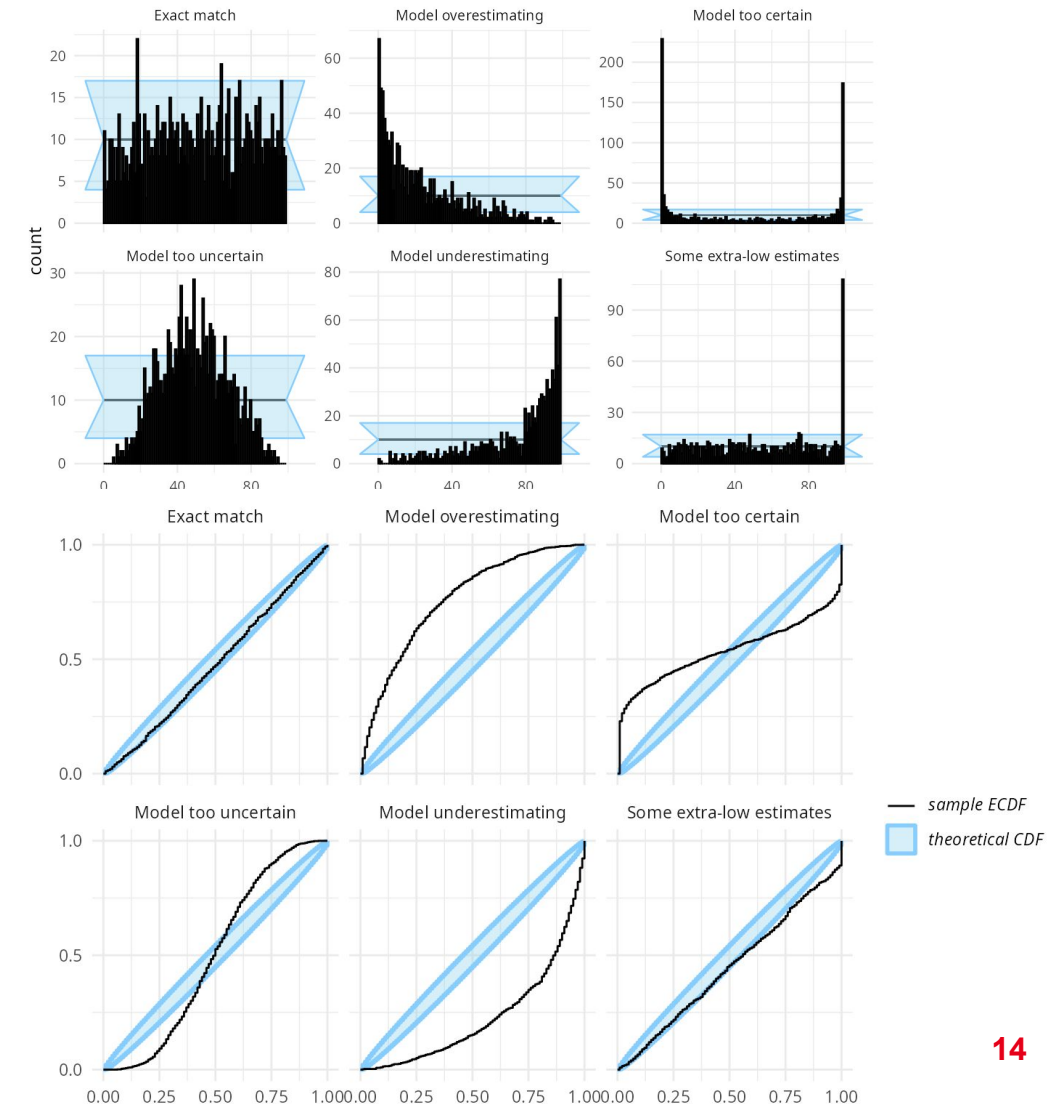


Validation of the training : Simulation based calibration (SBC)

Obtained SBC plots:



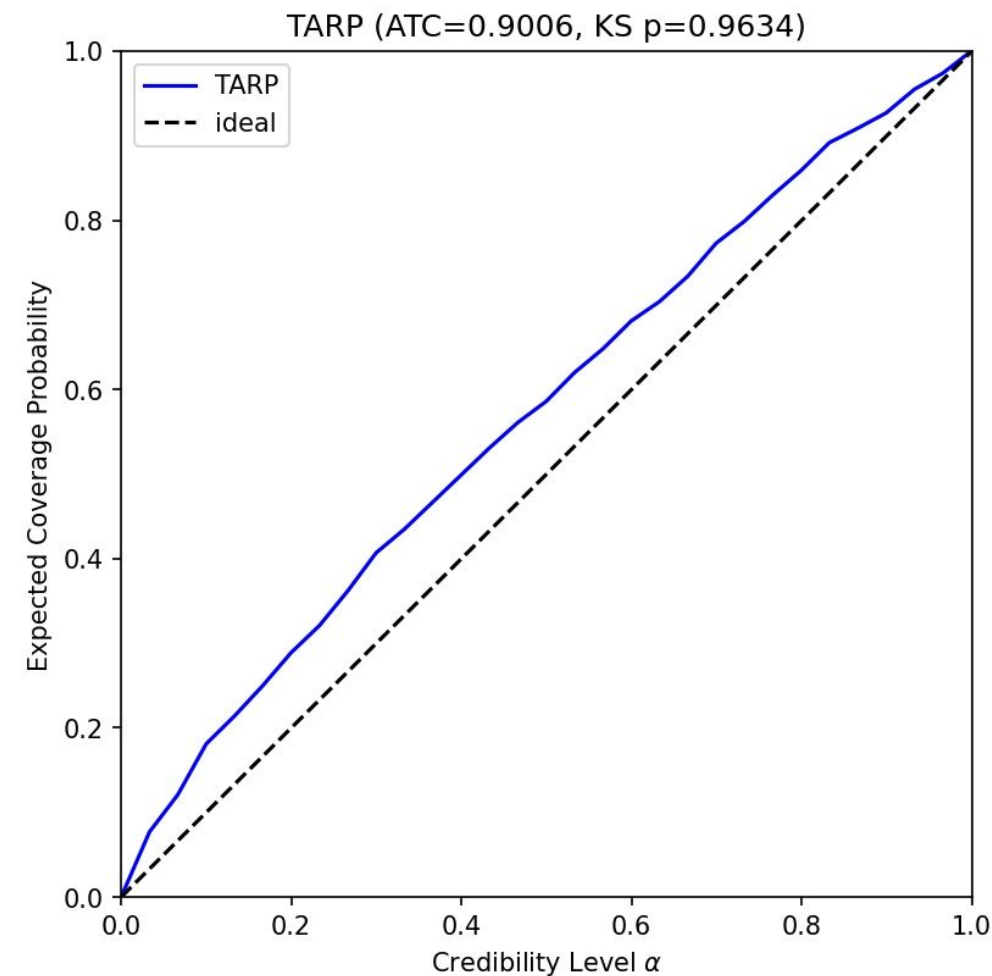
How to analyse SBC plots:



p-p plot calculation with TARP method

TARP method for p-p plots:

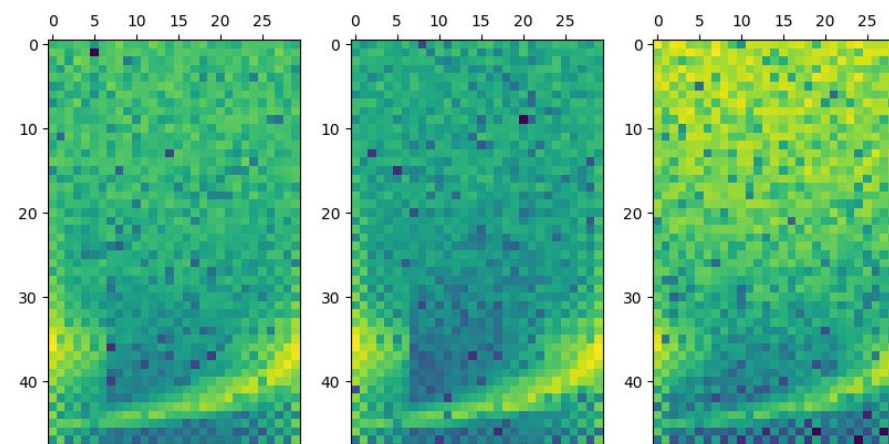
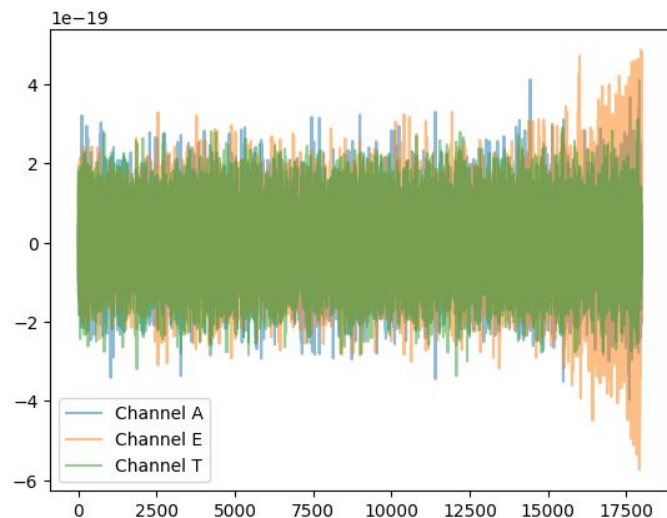
- Efficient method for p-p plot calculation
- Gives an aggregated p-p plot on the parameters
 - Plot the two cumulative distribution functions against each other
 - **If the model is well calibrated, the data will appear to be nearly a straight line**
- TARP gives a necessary and sufficient condition for posterior accuracy
- Detects inaccurate posterior estimators



Improvements: Summary statistics

Recent improvements:

Instead of using raw time series as input of an embedding network, we decided to use **time frequency representations**. In our case we tested **wavelet transforms** to get these ones.



CNN embedding:

The format of the new data representations allows to use a CNN embedding network to extract the deep features of the signals. In our recent works, the embedding reduces considerably the size of the input data of the neural network

Improvements: Flow Matching

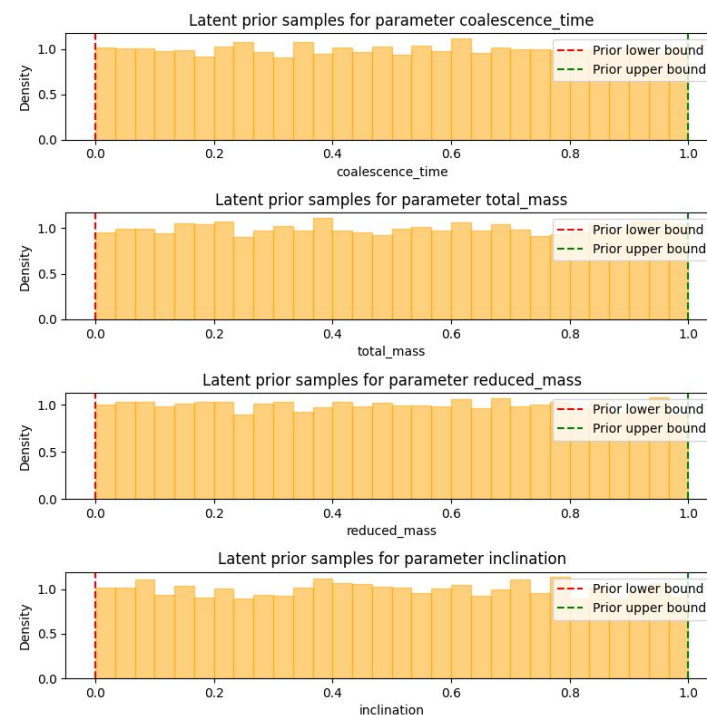
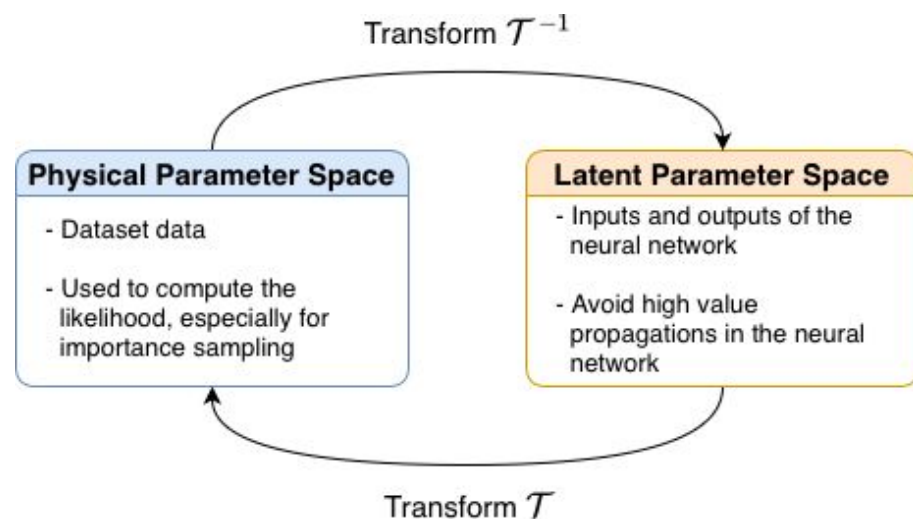
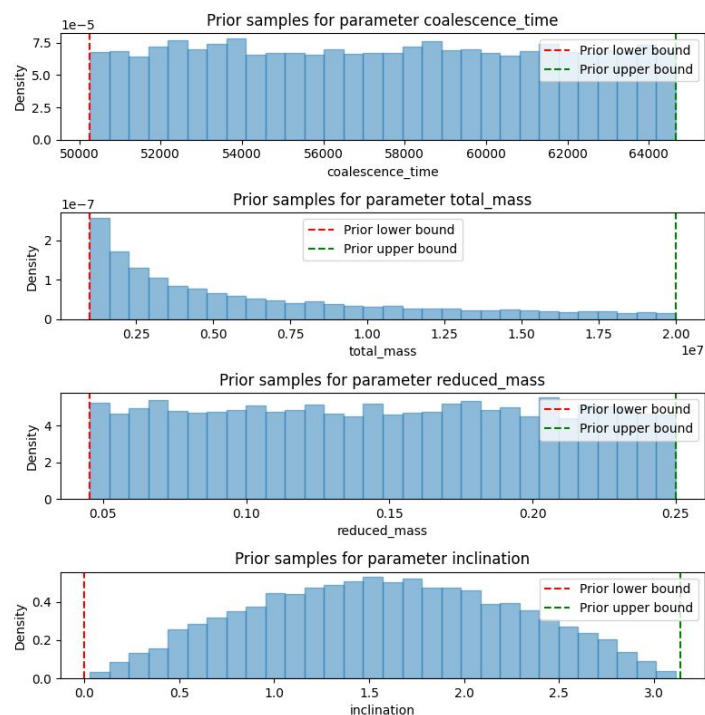
Recent improvements on the model used:

Loss curves provided by our first training of flow matching model

Improvements: Definition of the priors

Prior handling for better control of the model inputs and outputs:

With SBI toolkit, everything regarding the transformation of inputs is done under the hood. For our SBI pipeline **we want to have total control of our input and output data distributions.**



Conclusion and further improvements



Results of the SBI pipeline so far:

- Simulator using an accelerated JAX response
- Dataset generation scripts using the simulator to create paired datasets $\mathcal{D} = \{(\mathbf{x}_i, \theta_i)\}_{i=1, \dots, N}$ according to our needs in data
- Scripts performing training and parallelizing it on GPUs
- Validation script performing importance sampling and calibration plots to diagnose biases in the trained model
- Successful first runs of new model using new data representations and embedding networks
- New priors definition to better control the inputs and outputs of the model

Next steps:

- Connect more realistic waveforms to the JAX response and test the methods on Mojito data. **(almost done!!)**
- Keep tuning the flow matching model and work on summary statistics.
- Investigate SNR based parameter space restrictions **(currently testing the implemented method)**
- **Test on mojito light common dataset**