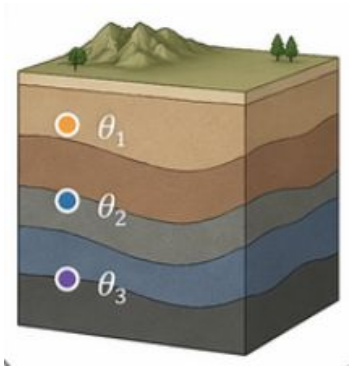
Learning Earth Structure

Generative Posterior Inference for Seismic Inversion under Model Misspecification

DMML Workshop 2026 - 17/18 June 2026

Context and motivation

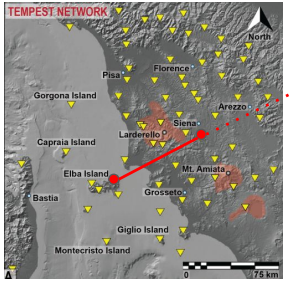
- **Geothermal exploration** requires imaging the subsurface.
- But exploiting it safely requires **knowing the underground structure** (subsurface).
- Classical subsurface imaging remains expensive, especially at the inversion stage.



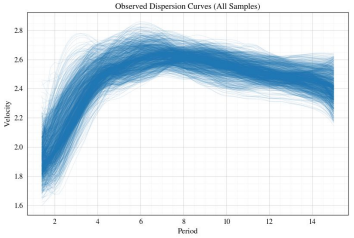
The 2D-1D ANT Pipeline

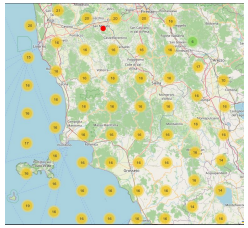
2D-1D Inversion

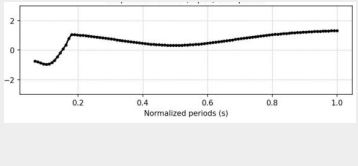
1. **Correlation of ambient noise** : cross-correlation between stations ;
2. **Extraction of dispersion curves** : how groups of frequencies propagate between a pair of stations ;
3. **Reconstruct a 2-D map** : extract local dispersion curves at each point (x,y) ;
4. **<u>1D-inversion</u>** : local curves can be inverted independently to recover 1-D velocity profiles ;

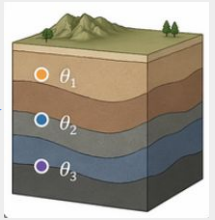


Timeseries









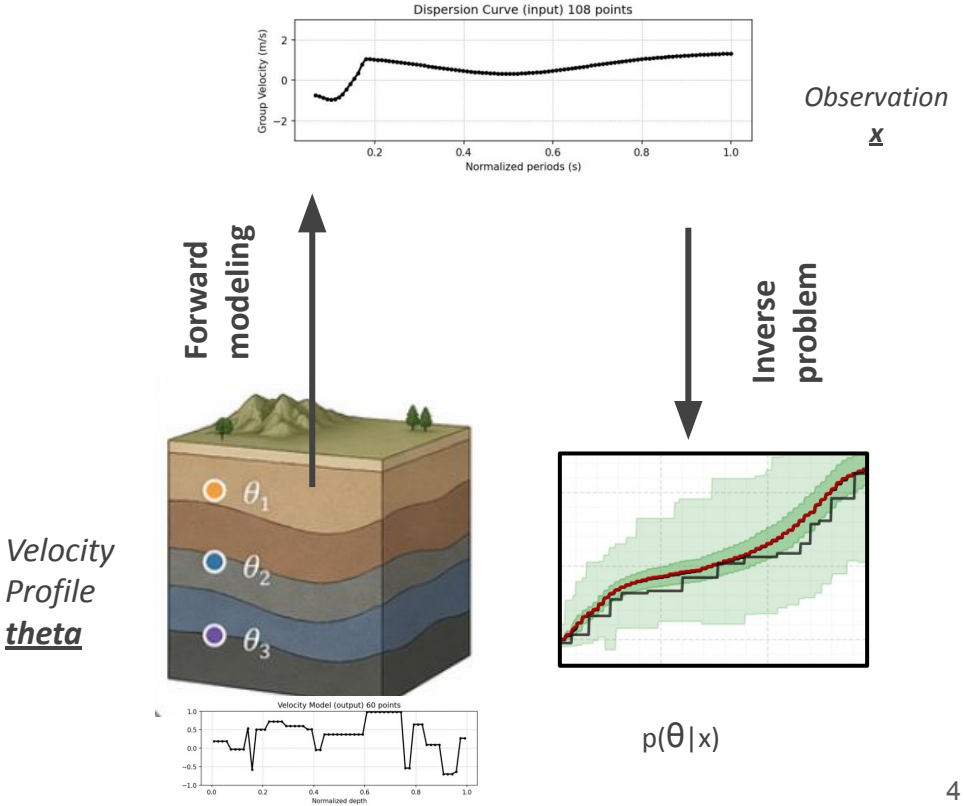
1D inversion of dispersion curves

Forward Modeling

- **Input**: 1D layered model of the subsurface
- **Solves**: dispersion equation derived from elastic wave equations
- **Output**: group velocity curve $v_g(p)$ as a function of *period p*

Inverse Problem

- **Ill-posed** (non-unique solution) and **trans-dimensional**
- **Learn the posterior distribution** $p(\theta|x)$
- **Validation**: do the generated model explain the observation ?



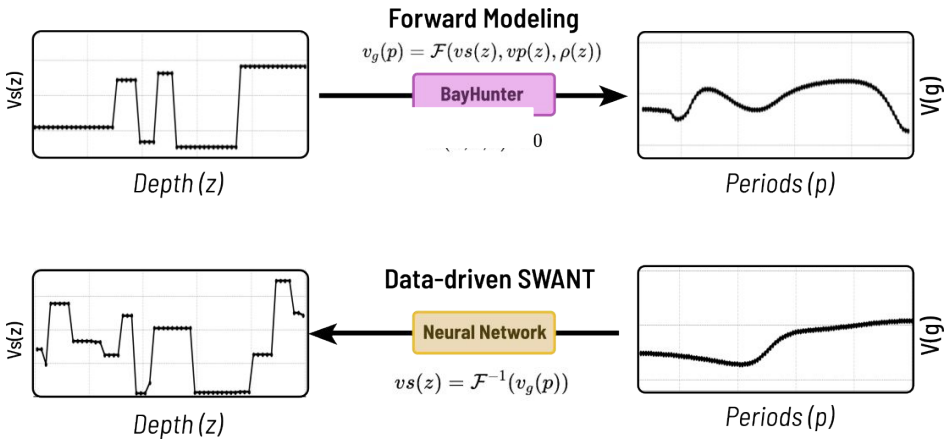
1D inversion of dispersion curves

Forward Modeling

- **Input**: 1D layered model of the subsurface
- **Solves**: dispersion equation derived from elastic wave equations
- **Output**: group velocity curve $v_g(p)$ as a function of *period p*

Inverse Problem

- **Ill-posed** (non-unique solution) and **trans-dimensional**
- **Learn the posterior distribution** p(θ|x)
- **Validation**: do the generated model explain the observation ?



Inversion problem: $v_g(p)$ -> $v_s(z)$

SBI Formulation

Main points

- We want the **conditional posterior** :

$$p(\theta \mid x)$$

- Likelihood-based inference is difficult because:
  - forward model is nonlinear / ill-defined
  - simulator is costly
  - number of layers is unknown
- Instead, we sample training pairs:

$$(\theta_i, x_i) \sim p(\theta)\, p(x \mid \theta)$$

- Then train a neural posterior estimator (NPE)

$$\hat{p}_\phi(\theta \mid x) \approx p(\theta \mid x)$$

- The cost is shifted from per-observation sampling to one offline training stage, enabling fast amortized posterior inference.

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)\, d\theta}$$

RJ-MCMC (Bayhunter)

- **Goal** : infer subsurface structure from observed data (e.g., dispersion curves, travel times, receiver functions).
- **RJ-MCMC** samples both parameter values (V_s) and model dimension (*number of layers*).
- **Within-model move**: perturb parameters with fixed (*k*) (e.g., velocity, depth, thickness).
- **Birth move**: add one component (k to k+1).
- **Death move**: remove one component (k to k-1).
- Move probabilities strongly affect efficiency and mixing.

- Build a Markov chain whose stationary distribution is the posterior.
- **At each iteration** :
  a. propose a candidate model,
  b. accept or reject with a probability rule,
  c. keep the accepted state.
- After burn-in, samples approximate the full posterior uncertainty.
- **Computationally expensive** (many forward solves).
- Start from a prior :
  a. V_s : [0.5, 4.5]
  b. Layers : [2, 20]
  c. V_p/V_s : [1.65, 1.85]

https://github.com/jenndrei/BayHunter

Synthetic Dataset & Prior

Main points

- We generate paired samples (θ_i, x_i) with the forward simulator **surf96**
- Observation:

$$x \in \mathbb{R}^{108}$$

- fundamental-mode Rayleigh-wave **group-velocity curve** sampled from **1 to 30 s.**

$$\theta \in \mathbb{R}^{480}$$

- Fixed-depth **velocity map** $V_s(z)$ on [0, 15] km
    - Total dataset : 10m synthetic pairs

Layered prior

- N_l in [2, 20] layers
- Depth range : 0-15 km
- V_s in [0.5, 4.0] km/s
- Increasing V_s with 10% probability of LVZ (low velocity zone)
- Minimum layer thickness
- V_p/V_s in [1.65, 1.85]
- Key equation : $x = \mathcal{F}(\theta)$

$$x \in \mathbb{R}^{108} \qquad \theta_{\text{layered}} = \{(V_{S,i}, h_i)\}_{i=1}^{N_l} \qquad \theta_{\text{velmap}} \in \mathbb{R}^{480}$$

From sampling to learning

Normalizing Flow (MAF, NSF)

- Learn an invertible conditional transport from Gaussian latent space to parameters.
- Conditional mapping:

$$z \sim \mathcal{N}(0, I) \quad \theta = g_\phi(z, x)$$

- Exact density via change of variables:

$$p_\phi(\theta \mid x) = p_Z(g_\phi^{-1}(\theta, x)) \left| \det \frac{\partial g_\phi^{-1}(\theta, x)}{\partial \theta} \right|$$

- Training

$$\mathcal{L}(\phi) = -\frac{1}{N} \sum_{i=1}^{N} \log p_\phi(\theta_i \mid x_i)$$

Flow Matching (UNet-1D, FUNet-1D)

- Start from Gaussian noise and gradually move toward a target parameter sample.

$$\theta_0 \sim \mathcal{N}(0, I)$$

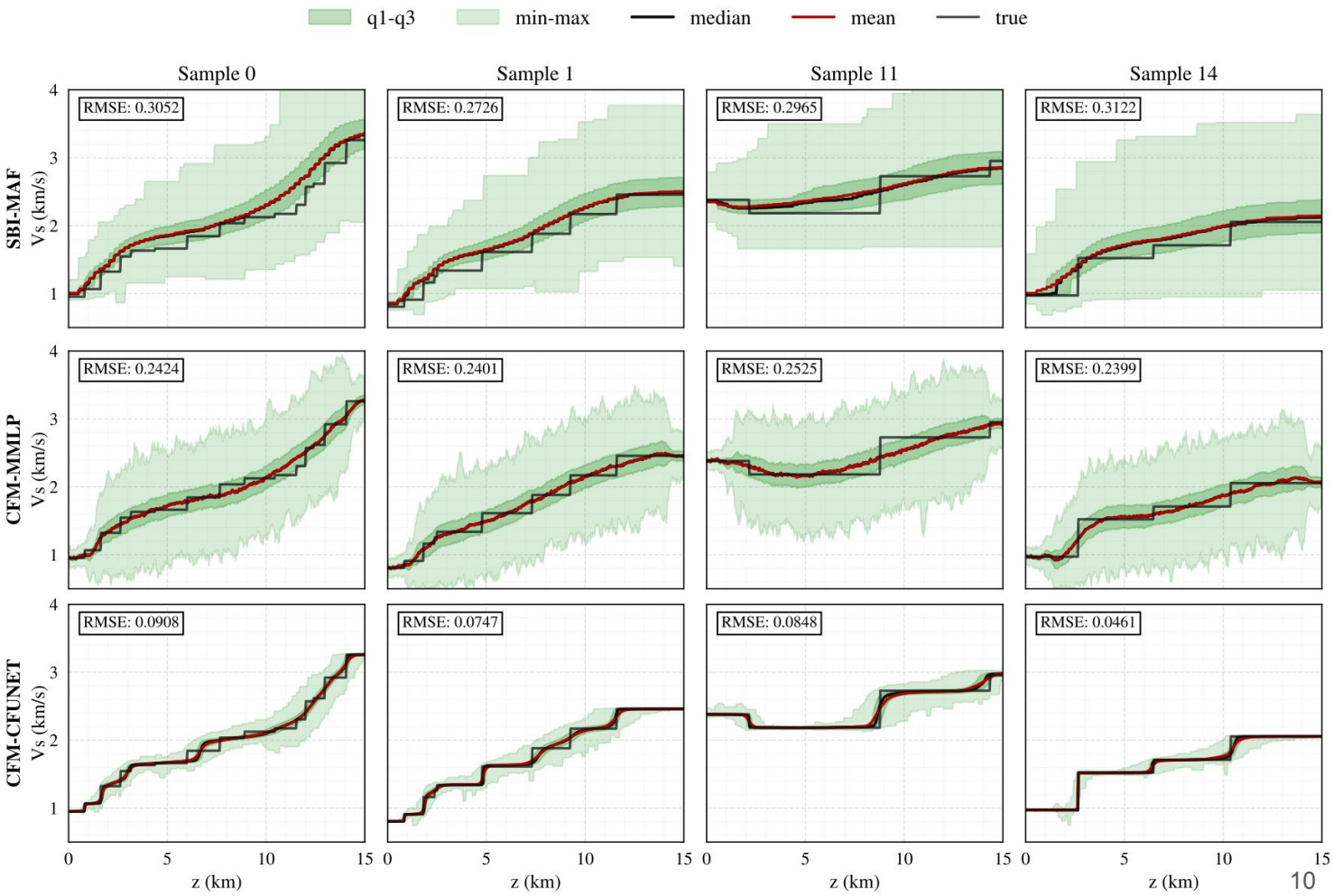
- Learn the vector field

$$\frac{d\theta_t}{dt} = v_\phi(\theta_t, x, t)$$

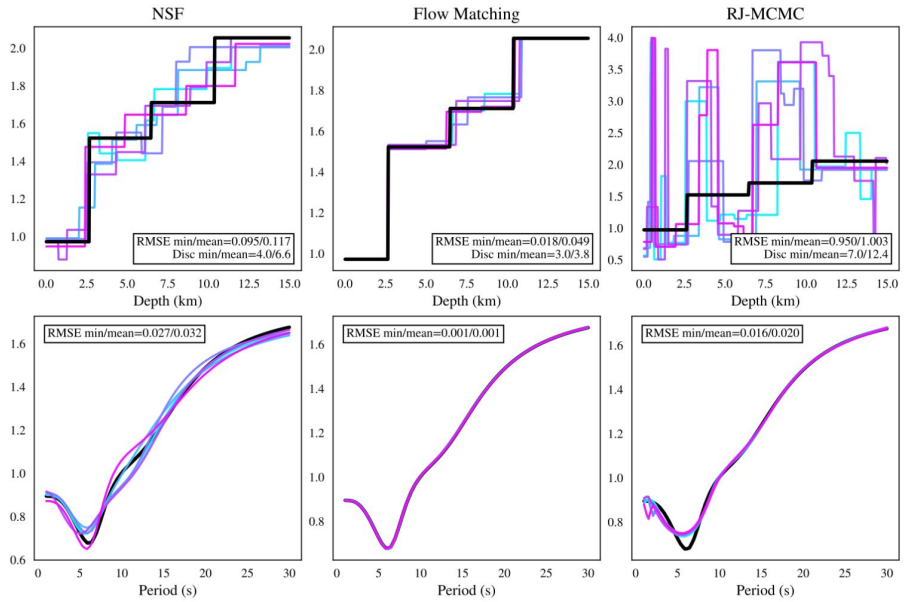
- Train by matching this predicted motion to the true transport direction.

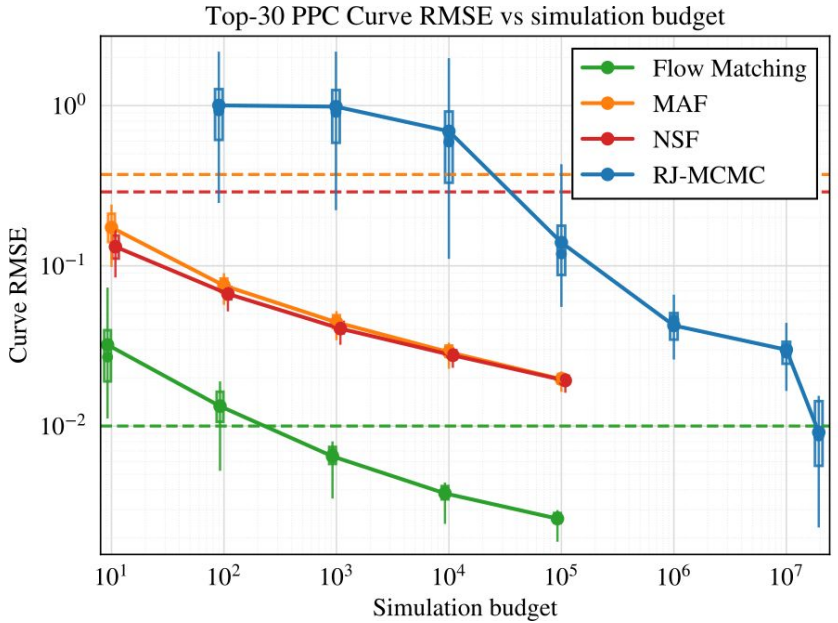
$$\theta_t = (1 - t)\theta_0 + t\theta \qquad \mathcal{L}(\phi) = \mathbb{E}\left[\| v_\phi(\theta_t, x, t) - (\theta - \theta_0) \|^2 \right]$$

Visualization of the posterior

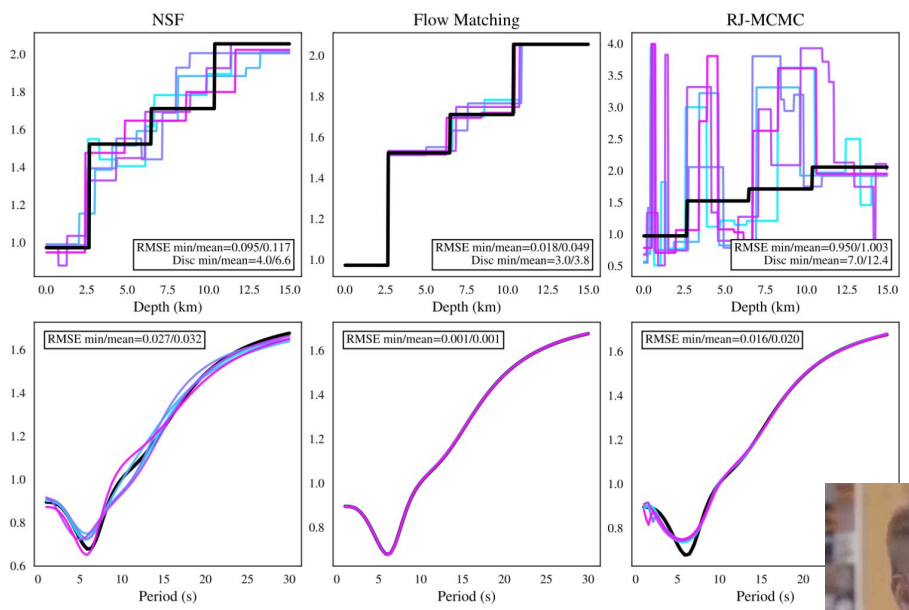


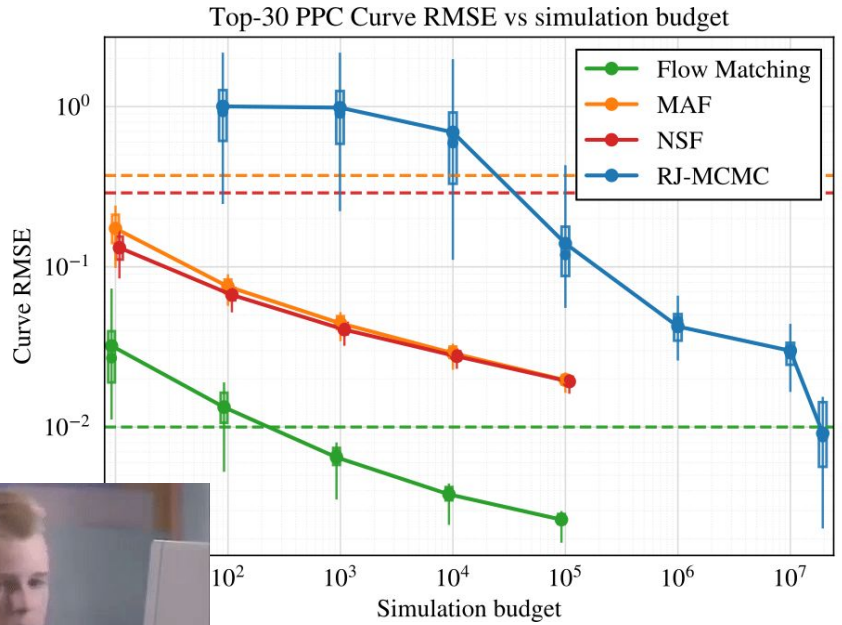
Efficiency and Computational Budget





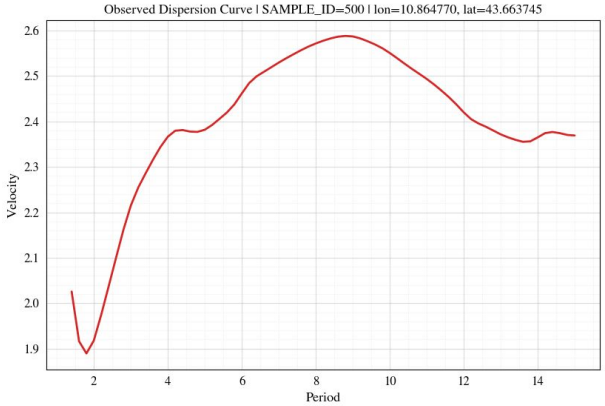
Efficiency and Computational Budget

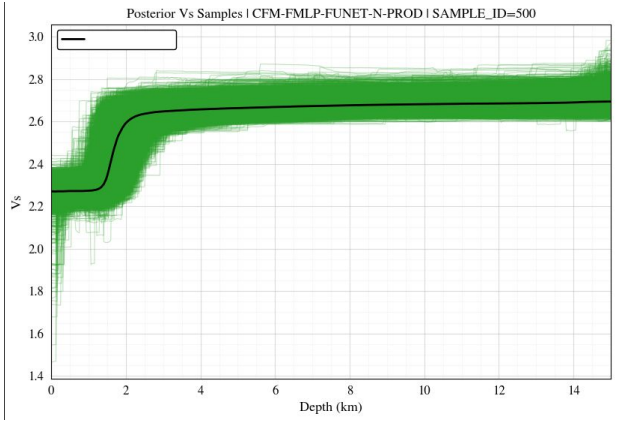


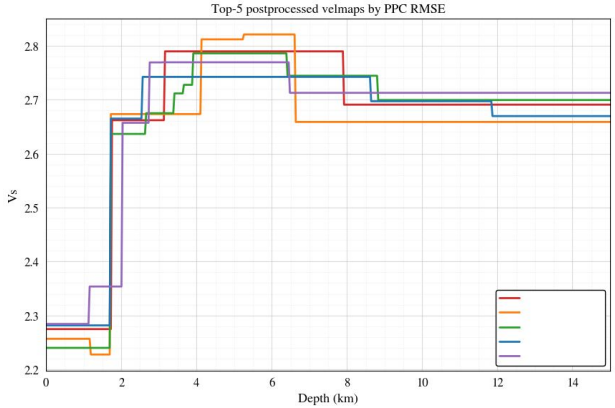


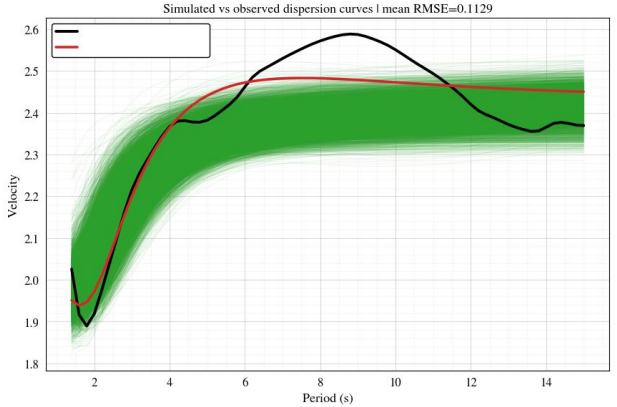
But...


well now I am not
doing it


Observed Dispersion Curve | SAMPLE_ID=500 | lon=10.864770, lat=43.663745
Velocity
Period

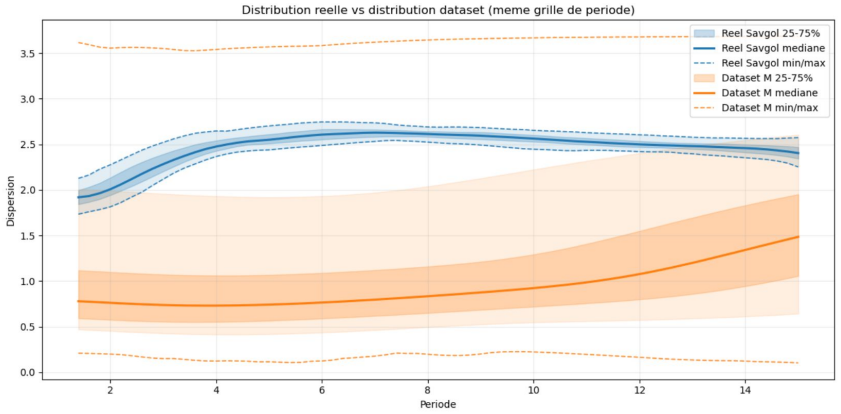

Posterior Vs Samples | CFM-FMLP-FUNET-N-PROD | SAMPLE_ID=500
Vs
Depth (km)

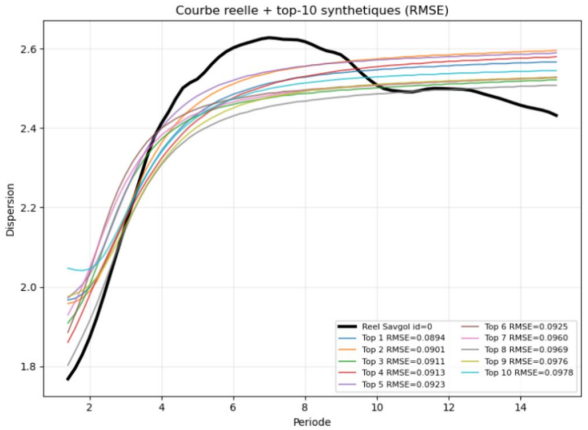

Top-5 postprocessed velmaps by PPC RMSE
Vs
Depth (km)

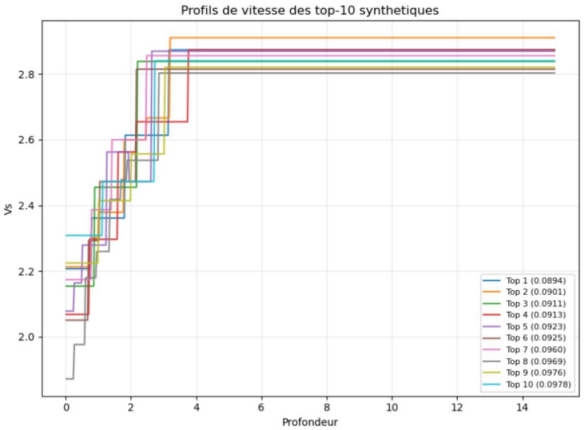

Simulated vs observed dispersion curves | mean RMSE=0.1129
Velocity
Period (s)

Model misspecification

Unrealistic prior







Model Misspecification

A **computer simulator** with parameter $\theta \in \Theta \subset \mathbb{R}^d$ defines implicitly a parametric family of distributions

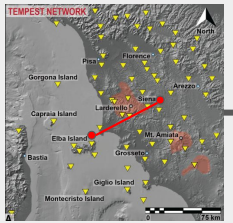
$$\mathbb{P}_\theta := \{p(\cdot \mid \theta); \theta \in \Theta\}.$$

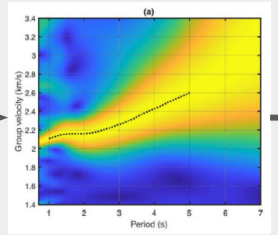
Running the simulator on θ by sampling from the likelihood $x \sim p(x|\theta)$. We have access to one or more observations y_i drawn from an unknown data-generating process p*.

- Well-specified scenario: $p* \in \mathbb{P}_\theta$ with $p^* = p(\cdot \mid \theta^*)$ for some θ*
- Misspecified scenario: $p^* \notin \mathbb{P}_\theta$ then $\mathbb{P}_\theta$ is misspecified with *y*

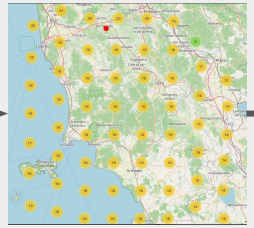
$\mathcal{X} := \left\{ \boldsymbol{x} : \int p(\boldsymbol{x} \mid \theta)\pi(\theta)\,\mathrm{d}\theta > 0 \right\}$ is the **observational support** of the model

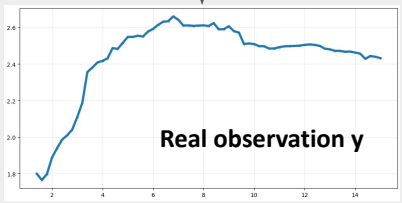
Model Misspecification

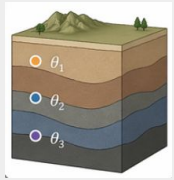


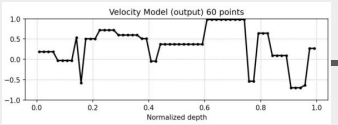


Hand picking

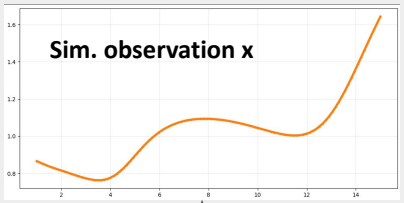




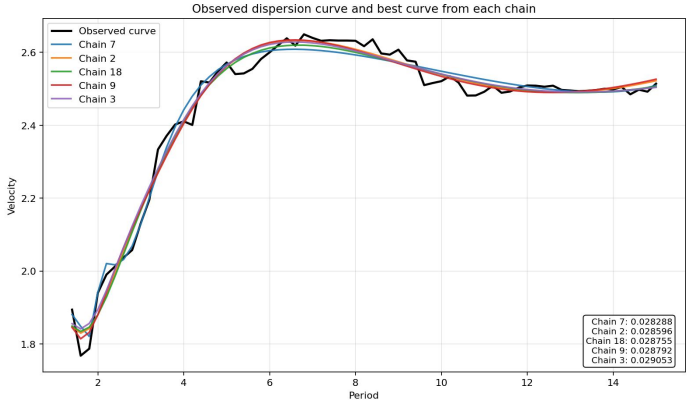


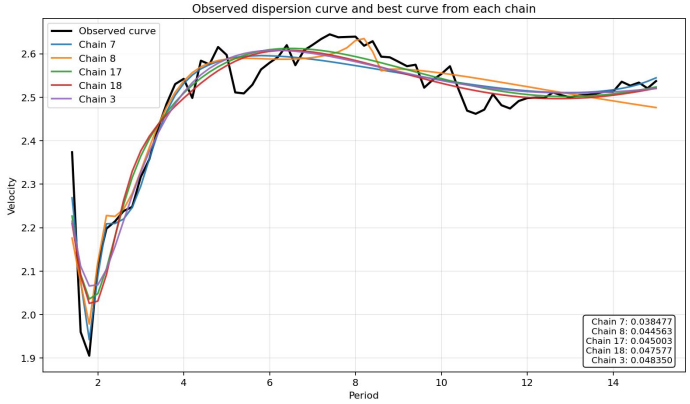


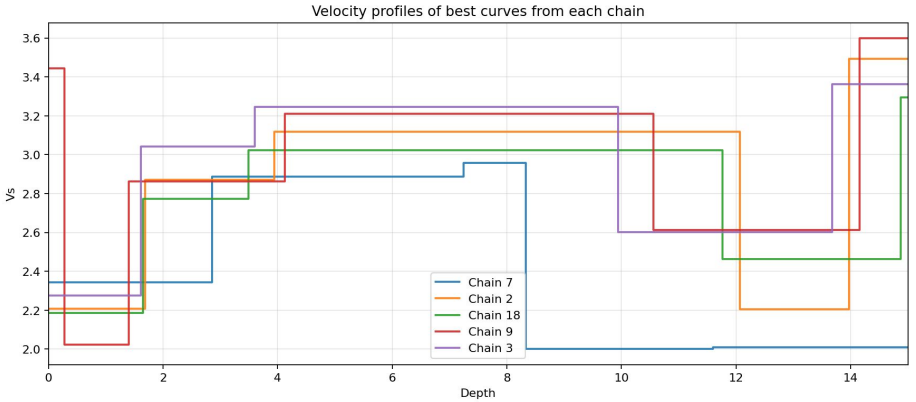
Surfdisp96

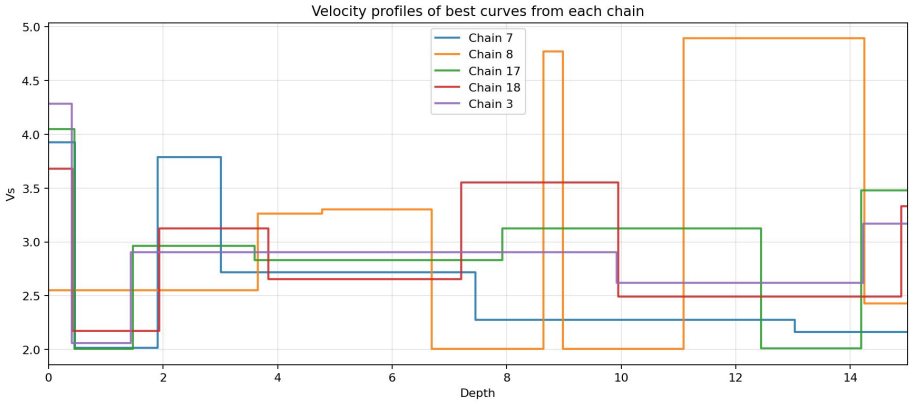


Observational support of Surfdisp96









Robust Neural Posterior Estimation

We can explicitly model discrepancies between observed data and simulations with an **error model** :

$$p(\theta, x, y) = p(y|x)p(x|\theta)p(\theta)$$

θ and **x** are respectively the parameters and output of the simulator and **y** is the observation. **x** is treated as an unobserved latent variable.

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \approx \frac{1}{M}\sum_{m=1}^{M} p(\boldsymbol{\theta}|\tilde{\boldsymbol{x}}_{\boldsymbol{m}}), \quad \tilde{x}_1, ..., \tilde{x}_M, \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}|\boldsymbol{y})$$

If we have access to p(x|y) and p(θ|x), we could sample posterior parameters by first sampling from the posterior over **x** and then from p(θ|x).

D. Ward, P. Cannon, M. Beaumont, M. Fasiolo, and S. M. Schmon, "Robust Neural Posterior Estimation and Statistical Model Criticism." [Online]. Available: https://arxiv.org/abs/2210.06564

Example error model

In practice, the true error model is unknown and should describe our prior belief over the discrepancy between the data generating process and simulations. **Spike and slab** error model,

$$
\begin{aligned}
x &\sim q(x), \\
z_j &\sim \text{Bernoulli}(\rho), \\
y_j \mid x_j, z_j &\sim \begin{cases} N\left(x_j, \sigma^2\right) & \text{if } \; z_j = 0 \\ \text{Cauchy}\left(x_j, \tau\right) & \text{if } \; z_j = 1 \end{cases}
\end{aligned}
$$

A key advantage of this error model : given by the latent variable z. The probability of being in the slab $\Pr(z_j = 1 \mid y)$ can be used as **posterior misspecification probability**.

RNPE Algorithm

Algorithm 1: Robust neural posterior estimation (RNPE)

for i *in* $1 : N$ **do**
1 | Sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$
2 | Simulate $\boldsymbol{x}_i \sim p(\boldsymbol{x} \mid \boldsymbol{\theta})$
end

Generate dataset

3 Train NPE $q(\boldsymbol{\theta} \mid \boldsymbol{x})$ on $\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^{N}$
4 Train $q(\boldsymbol{x})$ on $\{\boldsymbol{x}_i\}_{i=1}^{N}$
5 Sample $\tilde{\boldsymbol{x}}_m \sim \hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_o) \propto p(\boldsymbol{y}_o \mid \boldsymbol{x})q(\boldsymbol{x}),\ m = 1, \ldots, M$ using MCMC
6 Sample $\tilde{\boldsymbol{\theta}}_m \sim q(\boldsymbol{\theta} \mid \tilde{\boldsymbol{x}}_m),\ m = 1, \ldots, M$

Sample denoised x_m with MCMC

return $\{(\tilde{\boldsymbol{\theta}}_m, \tilde{\boldsymbol{x}}_m)\}_{m=1}^{M}$, samples drawn approximately from $p(\boldsymbol{\theta}, \boldsymbol{x} \mid \boldsymbol{y}_o)$

Learning Robust Statistics

- A common practice is to project both the simulated and the observed data onto a low-dimensional space of summary statistics $\mathcal{S}$ via the mapping $\eta : \mathcal{X}^n \to \mathcal{S}$ such that $s_{\text{obs}} = \eta(\mathbf{y}_{1:n})$
- Here η is not known a priori but is learned using a NN, the summary networks η_ψ
- Map each statistic vector s to the posterior $p(\theta|s)$ via inference network h_φ trained on simulated data.
- Perform poorly when the model is misspecified because the observed statistics s_{obs} becomes OOD under misspecification.
- $\eta\#\mathbb{P}_\theta^n$ is the **distribution of simulated** statistics, $\eta\#\mathbb{Q}^n$ is the **distribution of real statistics**. We define :

$$\varepsilon_\eta = \inf_{\theta \in \Theta} \mathcal{D}(\eta\#\mathbb{P}_\theta^n, \eta\#\mathbb{Q}^n)$$

- ε_η is the minimum non-reducible mismatch. $\mathcal{D}$ is distribution distance (MMD).

D. Huang, A. Bharti, A. Souza, L. Acerbi, and S. Kaski, "Learning Robust Statistics for Simulation-based Inference under Model Misspecification." [Online]. Available: https://arxiv.org/abs/2305.15871

Learning Robust Statistics

- ε_η is very slow to compute, but we can compute an upper bound by averaging the distances over some θ sampled from $p(\theta)$.
- To avoid collapse, the loss must be a tradeoff between being informative about θ reducing mismatch,

$$\mathcal{L}_{\mathrm{RS}}(\omega, \psi) = \mathcal{L}(\omega, \psi) + \lambda \mathbb{E}_{p(\theta)}[\mathcal{D}(\eta \# \mathbb{P}_\theta^n, \eta \# \mathbb{Q}^n)]$$

- ω are the **inference model** parameters, ψ the summary network parameters and λ the regularization parameter.
- We don't have access to the complete distribution $\eta \# \mathbb{P}_\theta^n$ and $\eta \# \mathbb{Q}^n$ only to some samples of these ($x_{1:n,i} \sim \mathbb{P}_{\theta i}$ and $y_{1:n,i} \sim \mathbb{Q}$).
- $\lambda \to 0$ converge to classical NPE methods. $\lambda \to \infty$ converges to the prior distribution.
- λ encodes the trade-off between efficiency and robustness.