Counterfactual-informed adaptive MCMC with conditional normalising flows

MaxEnt 2025 conference, Auckland, Aotearoa New Zealand

Florent Leclercq

www.florent-leclercq.eu

Institut d'Astrophysique de Paris
CNRS & Sorbonne Université

ANR-23-CE46-0006: INFOCW

In collaboration with:
Jens Jasche (Stockholm University)

and the Aquila Consortium

www.aquila-consortium.org

15 December 2025

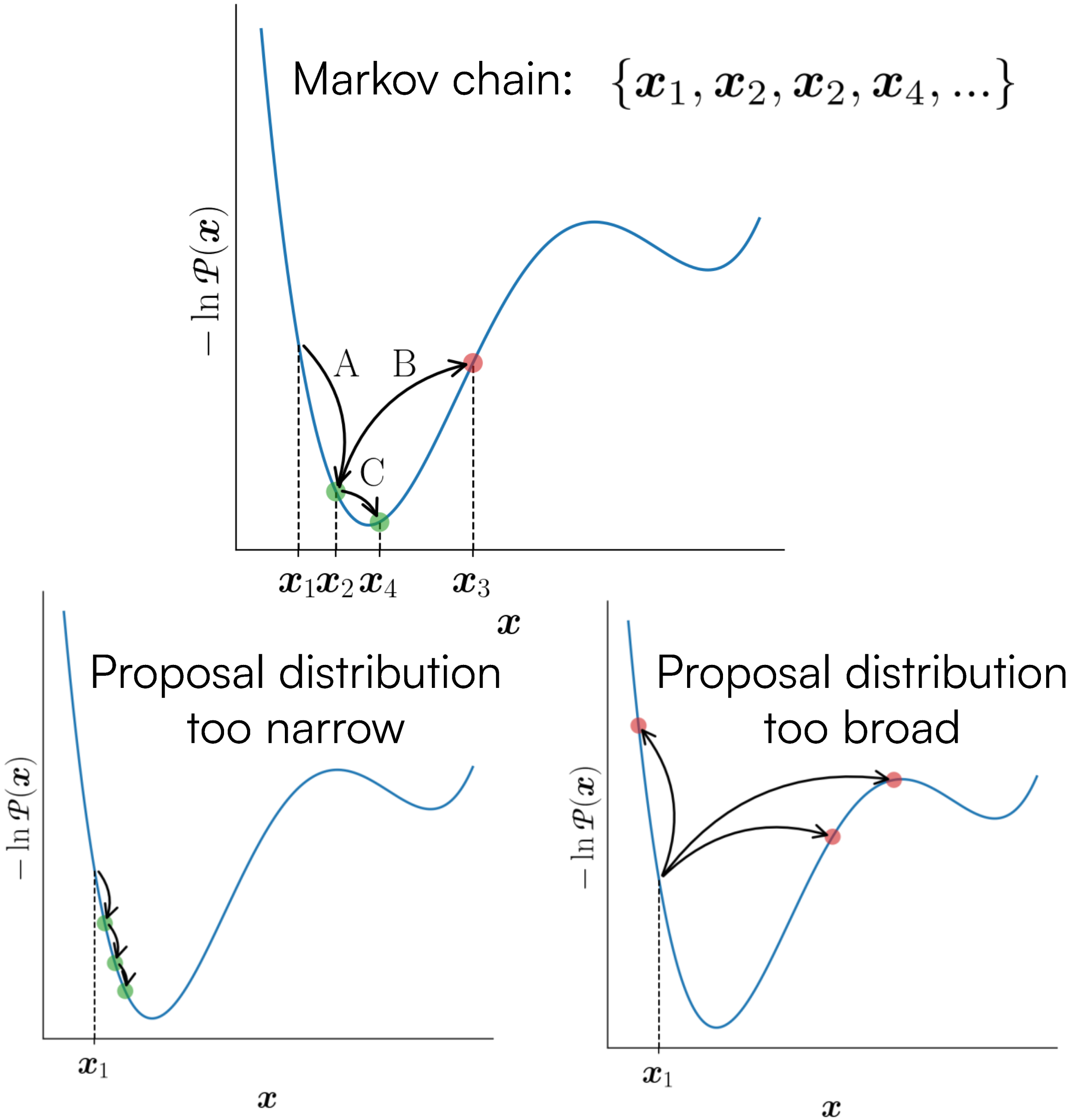
Cathedral Cove, Coromandel Peninsula

Markov Chain Monte Carlo (MCMC) with the Metropolis-Hastings algorithm

- The Metropolis-Hastings algorithm builds Markov chains by a sequence of moves that are either accepted or rejected with probability

$$a = \min\left[1, \frac{\mathcal{P}(\boldsymbol{x}^*|\boldsymbol{d})}{\mathcal{P}(\boldsymbol{x}|\boldsymbol{d})} \frac{\mathcal{Q}(\boldsymbol{x}|\boldsymbol{x}^*)}{\mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x})}\right].$$

- Under general hypotheses, it is possible to prove that the chain has the target distribution as its stationary distribution, i.e. elements of the chain become (asymptotically) samples of $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d})$.
- A good proposal distribution $\mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x})$ creates a distribution that has high acceptance rate and a low correlation length. A frustrating property: the optimal proposal distribution to sample from $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d})$ is... the target distribution $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d})$ itself!
- Is it possible to automatically build a proposal distribution?



A geometric interpretation of the Metropolis-Hastings test

- The MH test is: $\ln u \leq \ln \mathcal{P}(\boldsymbol{x}^*|\boldsymbol{d}) - \ln \mathcal{P}(\boldsymbol{x}|\boldsymbol{d}) + \ln \mathcal{Q}(\boldsymbol{x}|\boldsymbol{x}^*) - \ln \mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x}) \quad \text{for} \quad u \curvearrowleft \mathcal{U}([0,1])$
- Assume:

$$\boldsymbol{d} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{n}, \quad \boldsymbol{n} \curvearrowleft \mathcal{G}(\boldsymbol{0}, \mathbf{N}) \qquad \ln \mathcal{P}(\boldsymbol{x}|\boldsymbol{d}) = -\frac{1}{2}\left[\boldsymbol{d} - \boldsymbol{f}(\boldsymbol{x})\right]^\intercal \mathbf{N}^{-1}\left[\boldsymbol{d} - \boldsymbol{f}(\boldsymbol{x})\right] + \ln \mathcal{P}(\boldsymbol{x}) + \text{const.}$$

↖ Fully general data model (non-linear, non-differentiable...)

- Introduce: $Q(\boldsymbol{x}, \boldsymbol{x}^*) \equiv \ln \mathcal{Q}(\boldsymbol{x}|\boldsymbol{x}^*) - \ln \mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x})$

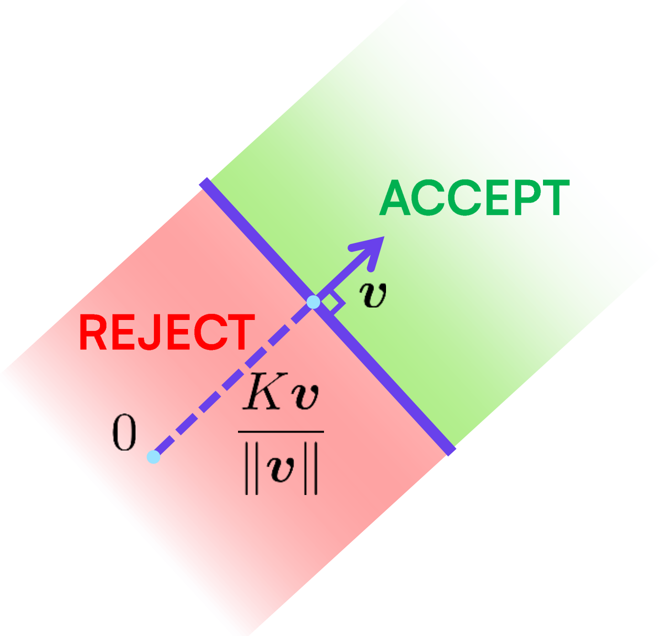
$$L(\boldsymbol{x}, \boldsymbol{x}^*) \equiv -\frac{1}{2}\left[\boldsymbol{f}(\boldsymbol{x})^\dagger \mathbf{N}^{-1} \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}^*)^\dagger \mathbf{N}^{-1} \boldsymbol{f}(\boldsymbol{x}^*)\right]$$

$$P(\boldsymbol{x}, \boldsymbol{x}^*) \equiv \ln \mathcal{P}(\boldsymbol{x}^*) - \ln \mathcal{P}(\boldsymbol{x})$$

$$K(u, \boldsymbol{x}, \boldsymbol{x}^*) \equiv \ln u + Q(\boldsymbol{x}, \boldsymbol{x}^*) + L(\boldsymbol{x}, \boldsymbol{x}^*) + P(\boldsymbol{x}, \boldsymbol{x}^*)$$

$$\boldsymbol{v}(\boldsymbol{x}, \boldsymbol{x}^*) \equiv \boldsymbol{f}(\boldsymbol{x}^*) - \boldsymbol{f}(\boldsymbol{x}) \qquad \langle \boldsymbol{a}|\boldsymbol{b}\rangle \equiv \boldsymbol{a}^\dagger \mathbf{N}^{-1} \boldsymbol{b}$$

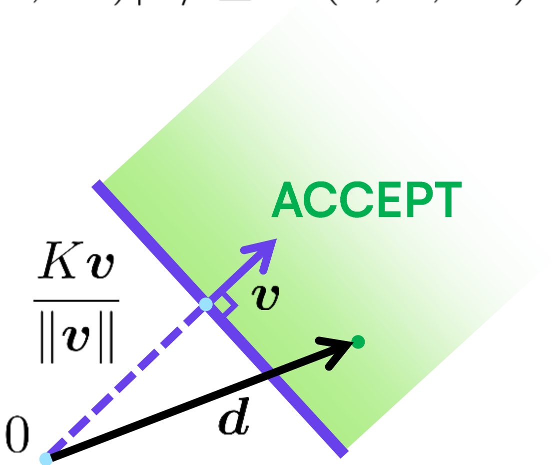
- Then the MH test is equivalent to: $\langle \boldsymbol{v}(\boldsymbol{x}, \boldsymbol{x}^*)|\boldsymbol{d}\rangle \geq K(u, \boldsymbol{x}, \boldsymbol{x}^*)$
- $\langle \boldsymbol{v}|\boldsymbol{d}\rangle = K$ is the equation of a hyperplane in data space.



Counterfactuals in the Metropolis-Hastings test

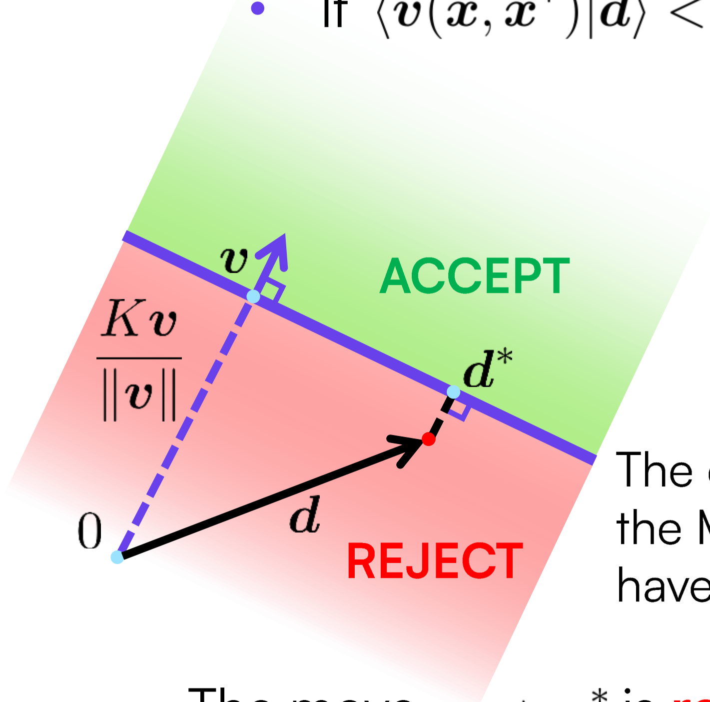
- When running an MCMC based on the MH algorithm, it is possible to build a "replay buffer":
- At each step, draw $\boldsymbol{x}^*$ from $\mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x})$ and u from $\mathcal{U}([0,1])$. Compute $\boldsymbol{v}(\boldsymbol{x},\boldsymbol{x}^*)$ and $K(u,\boldsymbol{x},\boldsymbol{x}^*)$.

  - If $\langle \boldsymbol{v}(\boldsymbol{x},\boldsymbol{x}^*)|\boldsymbol{d}\rangle \geq K(u,\boldsymbol{x},\boldsymbol{x}^*)$:



The move $\boldsymbol{x} \to \boldsymbol{x}^*$ is **accepted** and we record $\{\boldsymbol{x}^*, \boldsymbol{d}\}$ in the replay buffer.

  - If $\langle \boldsymbol{v}(\boldsymbol{x},\boldsymbol{x}^*)|\boldsymbol{d}\rangle < K(u,\boldsymbol{x},\boldsymbol{x}^*)$:

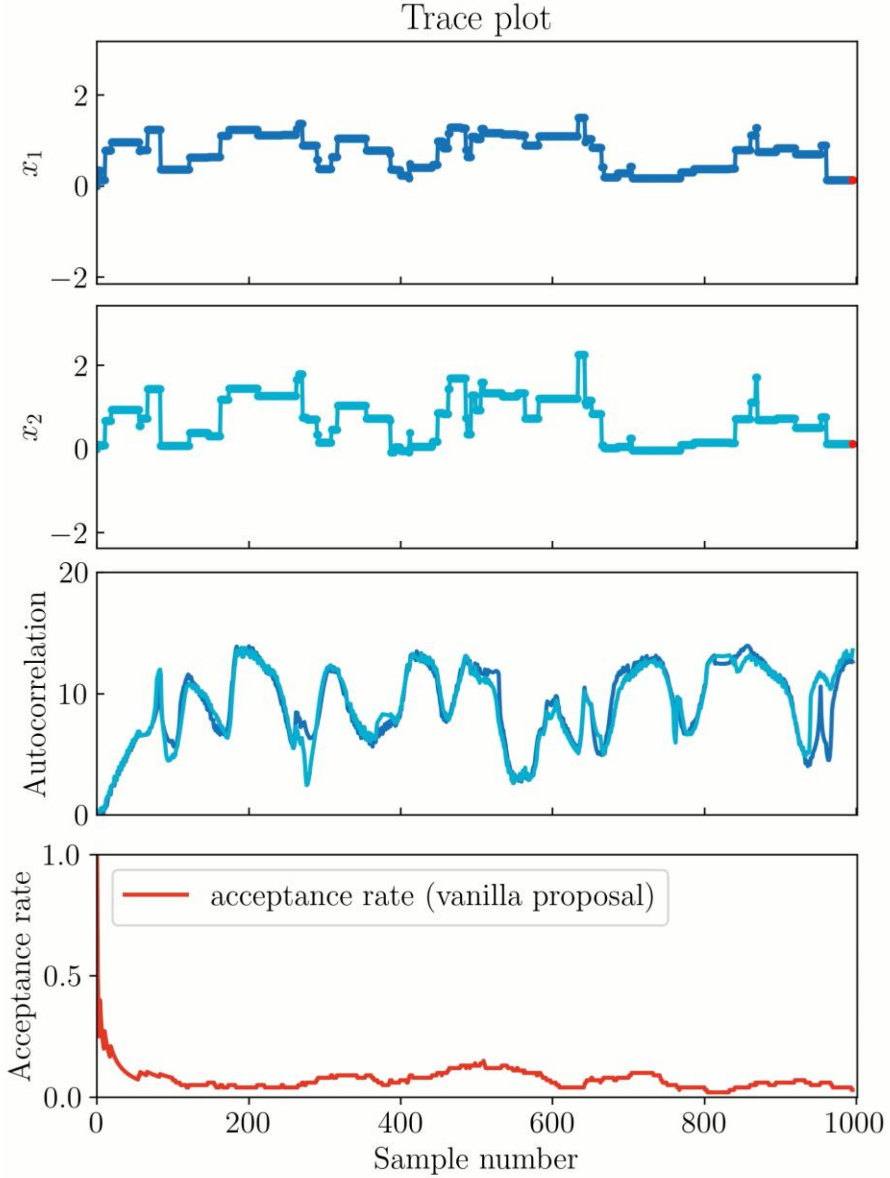


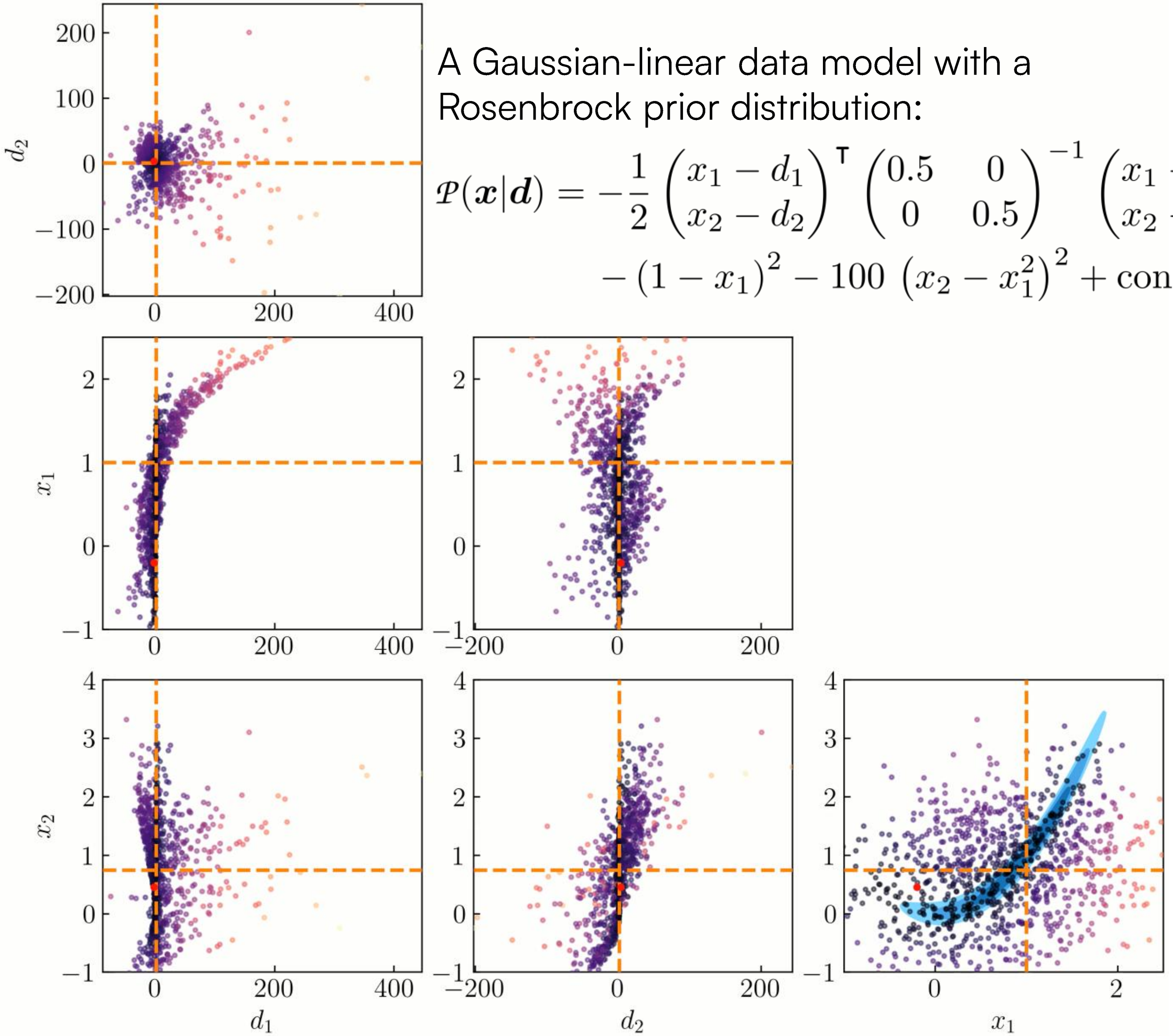
$$\boldsymbol{d}^* \equiv \boldsymbol{d} + \frac{K - \langle \boldsymbol{v}|\boldsymbol{d}\rangle}{\|\boldsymbol{v}\|^2}\,\boldsymbol{v}$$

The closest "alternative data" (in the Mahalanobis sense) that would have led to an acceptance.

The move $\boldsymbol{x} \to \boldsymbol{x}^*$ is **rejected** and we record $\{\boldsymbol{x}^*, \boldsymbol{d}^*\}$ in the replay buffer.

Building the replay buffer for a two-dimensional banana-shaped posterior

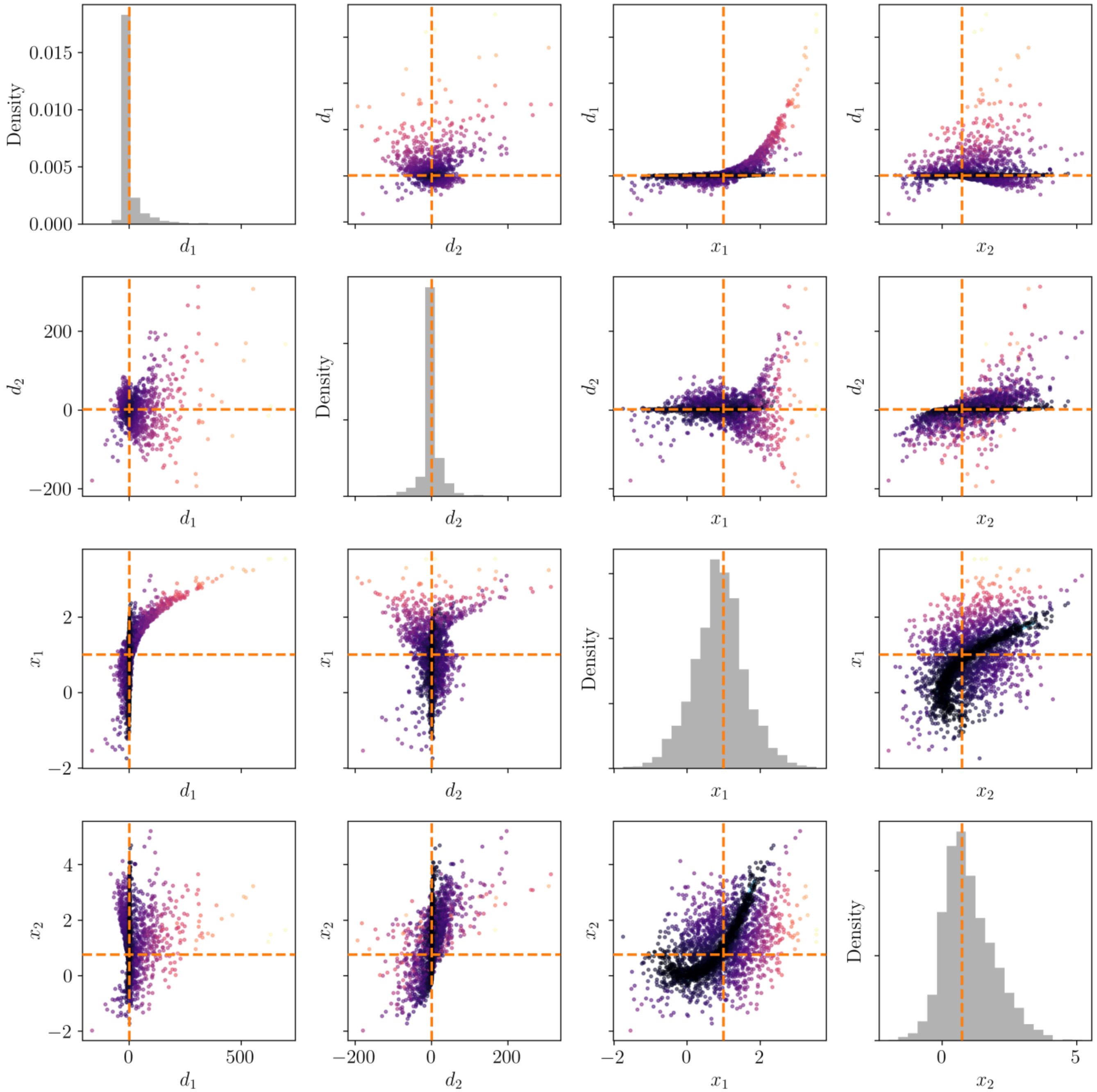




A Gaussian-linear data model with a Rosenbrock prior distribution:

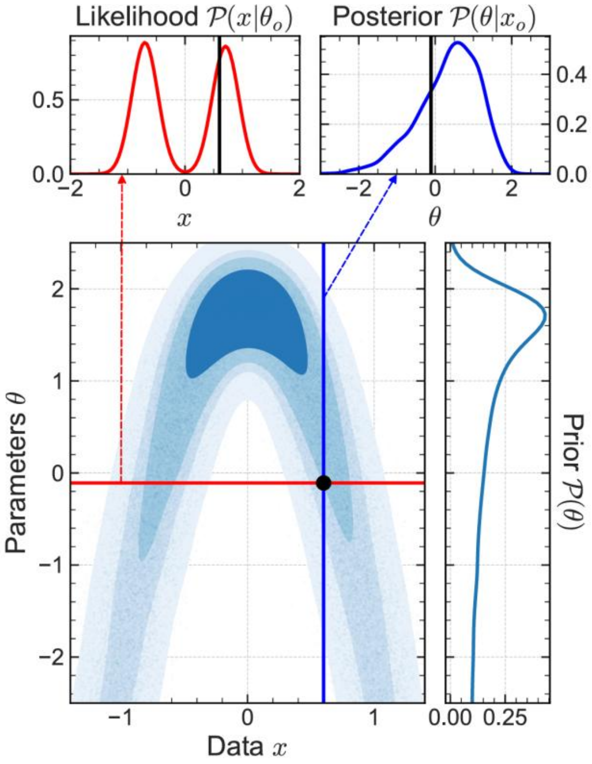
$$\mathcal{P}(\boldsymbol{x}|\boldsymbol{d}) = -\frac{1}{2}\begin{pmatrix} x_1 - d_1 \\ x_2 - d_2 \end{pmatrix}^\intercal \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}^{-1} \begin{pmatrix} x_1 - d_1 \\ x_2 - d_2 \end{pmatrix} - (1 - x_1)^2 - 100\,\left(x_2 - x_1^2\right)^2 + \text{const.}$$

Replay buffer



Fitting a model to pairs of parameters and data

- The replay buffer is reminiscent of simulation-based inference, where the typical problem is fitting a model to pairs $\{\boldsymbol{x}, \boldsymbol{d}\}$ and conditioning on $\boldsymbol{d}_{\mathrm{true}}$ to get the posterior $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d}_{\mathrm{true}})$:



Ho *et al.*, 2402.05137

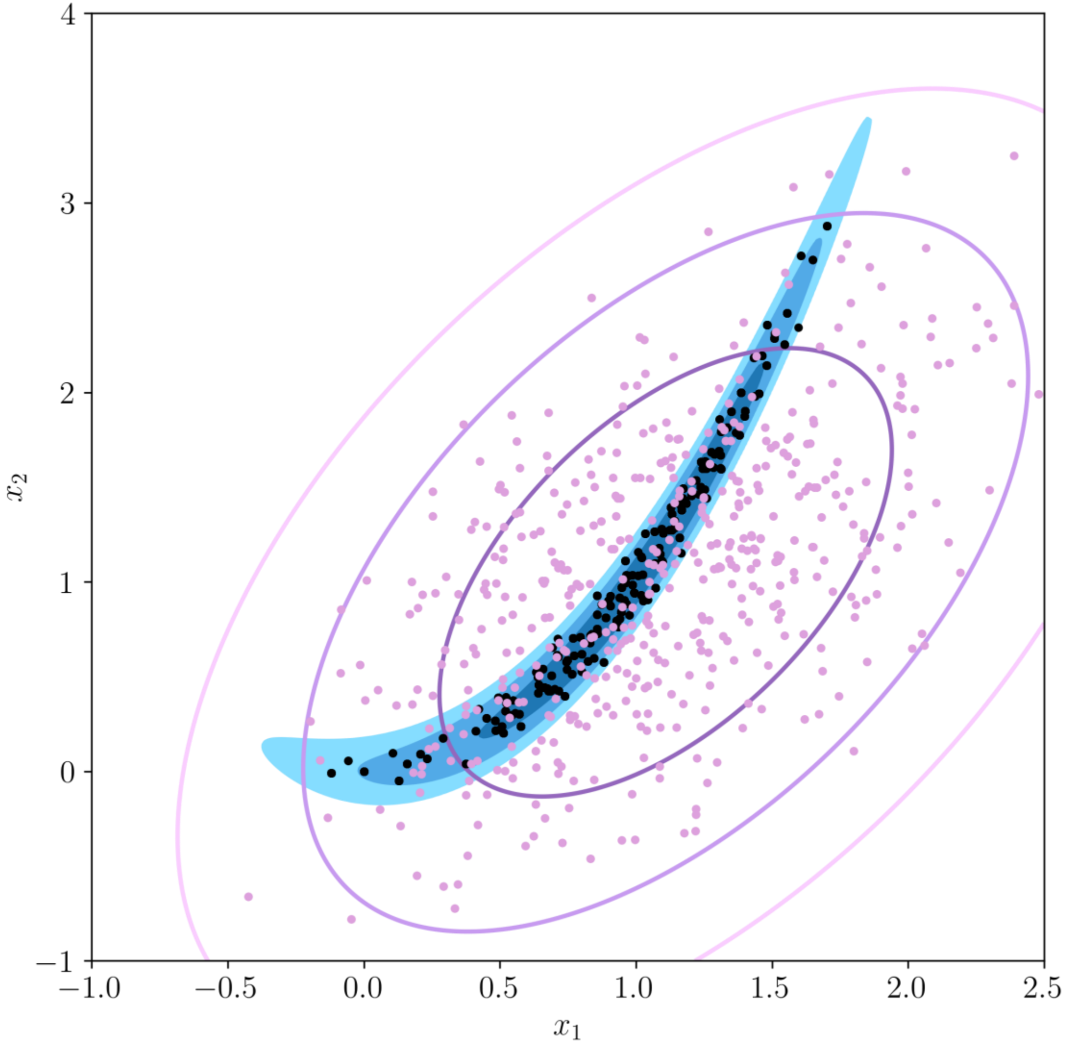
- In our framework, $\{\boldsymbol{x}, \boldsymbol{d}\}$ pairs of the replay buffer are not i.i.d. draws from $\mathcal{P}(\boldsymbol{x}, \boldsymbol{d})$ or any standard joint model.
- The MH sampling rule creates a *selected* and *truncated* joint distribution, where $\boldsymbol{d}$ is only observable at two extreme regions:
  - Exactly at $\boldsymbol{d}_{\mathrm{true}}$, where we have exact samples from the posterior $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d}_{\mathrm{true}})$ (accepted moves).
  - Near $\boldsymbol{d}_{\mathrm{true}}$, on the acceptance boundary in data space (rejected moves).
- When the acceptance rate is tiny, most samples are rejected, so we record lots of $\{\boldsymbol{x}^*, \boldsymbol{d}^*\}$. Notice:
  - $\boldsymbol{d}^*$ carries information about how $\boldsymbol{x}$ relates to data space near the likelihood ridges.
  - Labelled pairs $\{\boldsymbol{x}^*, \boldsymbol{d}^*\}$ still constrain the geometry of the likelihood around $\boldsymbol{d}_{\mathrm{true}}$.
  - Modern conditional models can learn these local structures and predict what the density looks like at the anchor point $\boldsymbol{d}_{\mathrm{true}}$.
- In other words, the model does not need unbiased samples; it only needs structured constraints in the joint. The replay buffer provides rich geometric constraints, despite being biased.

Fitting a conditional Gaussian distribution to the replay buffer

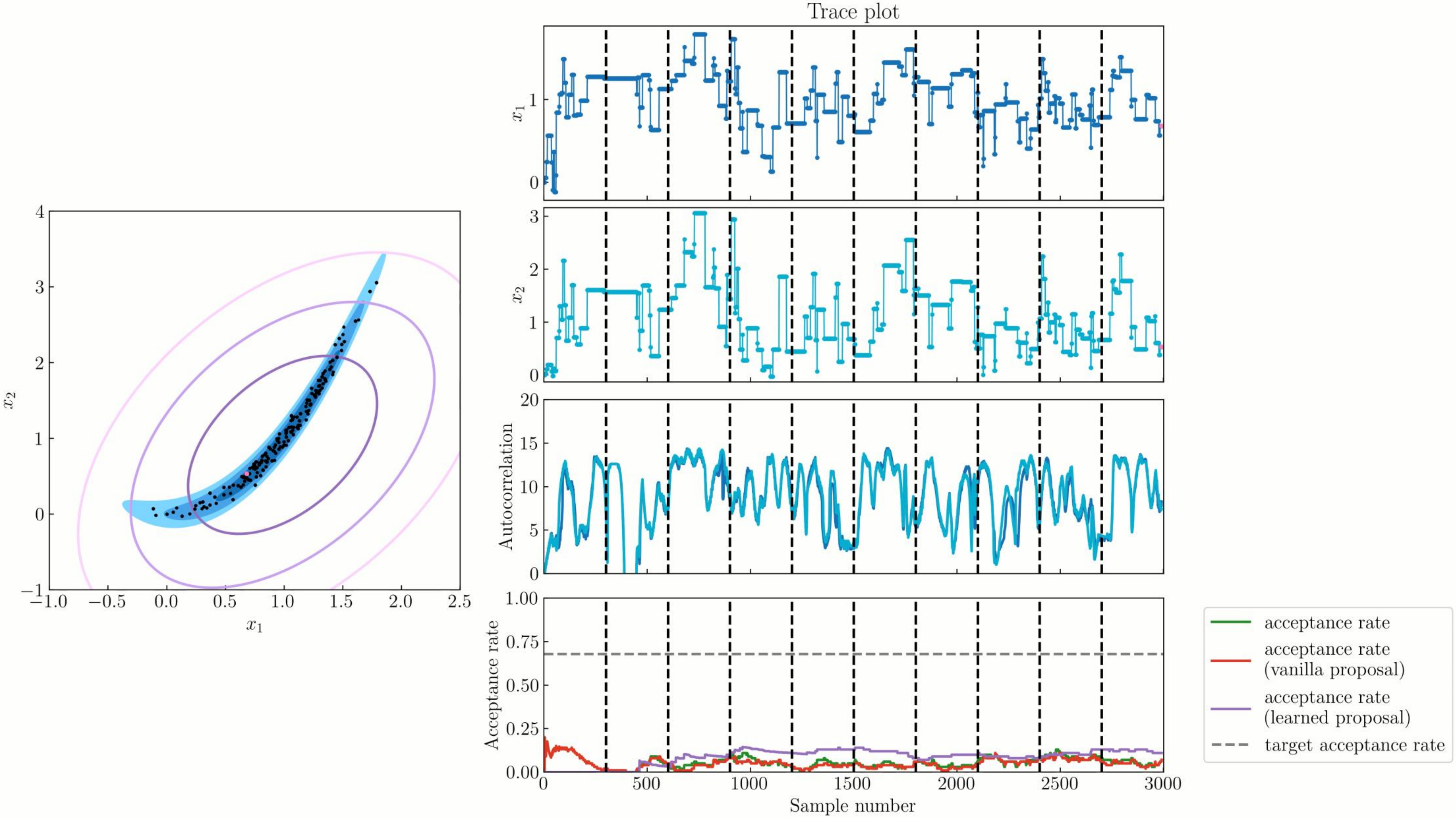
- The easiest option to fit a conditional model to the replay buffer is to assume that pairs $\{\boldsymbol{x}, \boldsymbol{d}\}$ are jointly Gaussian-distributed:

$$\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{d} \end{pmatrix} \sim \mathcal{G}\left[\begin{pmatrix} \boldsymbol{\mu}_{\boldsymbol{x}} \\ \boldsymbol{\mu}_{\boldsymbol{d}} \end{pmatrix}, \begin{pmatrix} \mathbf{C}_{\boldsymbol{xx}} & \mathbf{C}_{\boldsymbol{xd}} \\ \mathbf{C}_{\boldsymbol{dx}} & \mathbf{C}_{\boldsymbol{dd}} \end{pmatrix} \right].$$

- Then, the conditional distribution $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d}_\mathrm{true})$ is Gaussian, with well-known expressions for the mean and covariance matrix:
  - $\boldsymbol{\mu}_{\boldsymbol{x}|\boldsymbol{d}_\mathrm{true}} = \boldsymbol{\mu}_{\boldsymbol{x}} + \mathbf{C}_{\boldsymbol{xd}} \mathbf{C}_{\boldsymbol{dd}}^{-1} (\boldsymbol{d}_\mathrm{true} - \boldsymbol{\mu}_{\boldsymbol{d}})$
  - $\mathbf{C}_{\boldsymbol{x}|\boldsymbol{d}_\mathrm{true}} = \mathbf{C}_{\boldsymbol{xx}} - \mathbf{C}_{\boldsymbol{xd}} \mathbf{C}_{\boldsymbol{dd}}^{-1} \mathbf{C}_{\boldsymbol{dx}}$
- We can use this learned conditional model as the proposal distribution in MCMC.



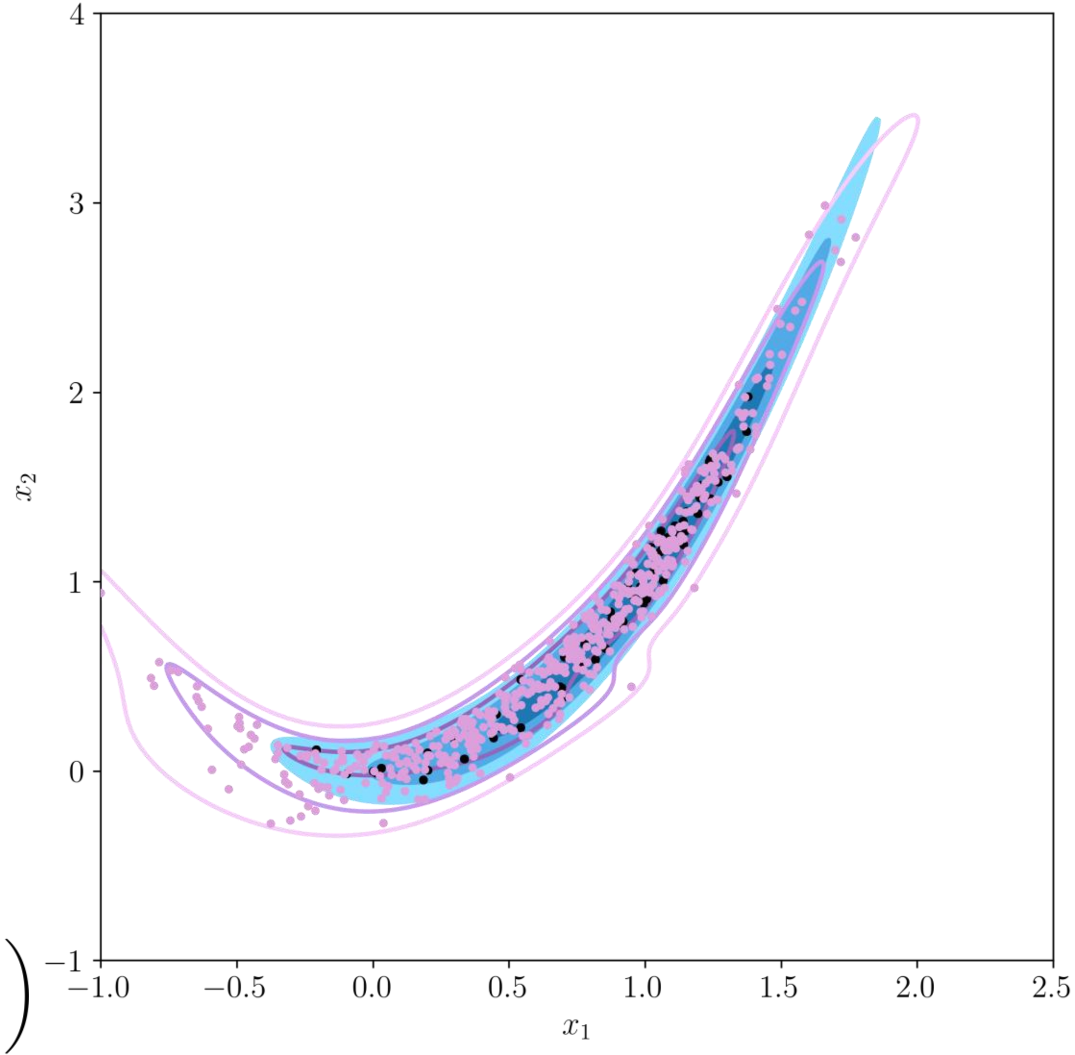
Counterfactual-informed adaptive MCMC with conditional Gaussian distributions



Fitting a conditional normalising flow to the replay buffer

- Conditional normalising flows use machine learning to learn an invertible mapping $\boldsymbol{x} \leftrightarrow \boldsymbol{z}$ conditioned on context ($\boldsymbol{d}_\mathrm{true}$ here), enabling sampling and evaluation of log-densities (both being required for MCMC).
- We use masked autoregressive flows (MAFs) as implemented in the **sbi** package for sequential neural posterior estimation (SNPE).
- In order to focus the training on $\mathcal{P}(\boldsymbol{x}|\boldsymbol{d}_\mathrm{true})$ (the only slice we care about), we use a conditional de-amortisation strategy:
  - We train only on the K-nearest neighbours of $\boldsymbol{d}_\mathrm{true}$ in data space.
  - We introduce weights in the loss function:

$$L(\theta) \propto -\sum_i w_i \log \mathcal{P}_\theta(\boldsymbol{x}_i|\boldsymbol{d}_i), \quad w_i \propto \exp\left(-\frac{1}{2}\frac{||\boldsymbol{d}_i - \boldsymbol{d}_\mathrm{true}||^2}{\sigma^2}\right)$$



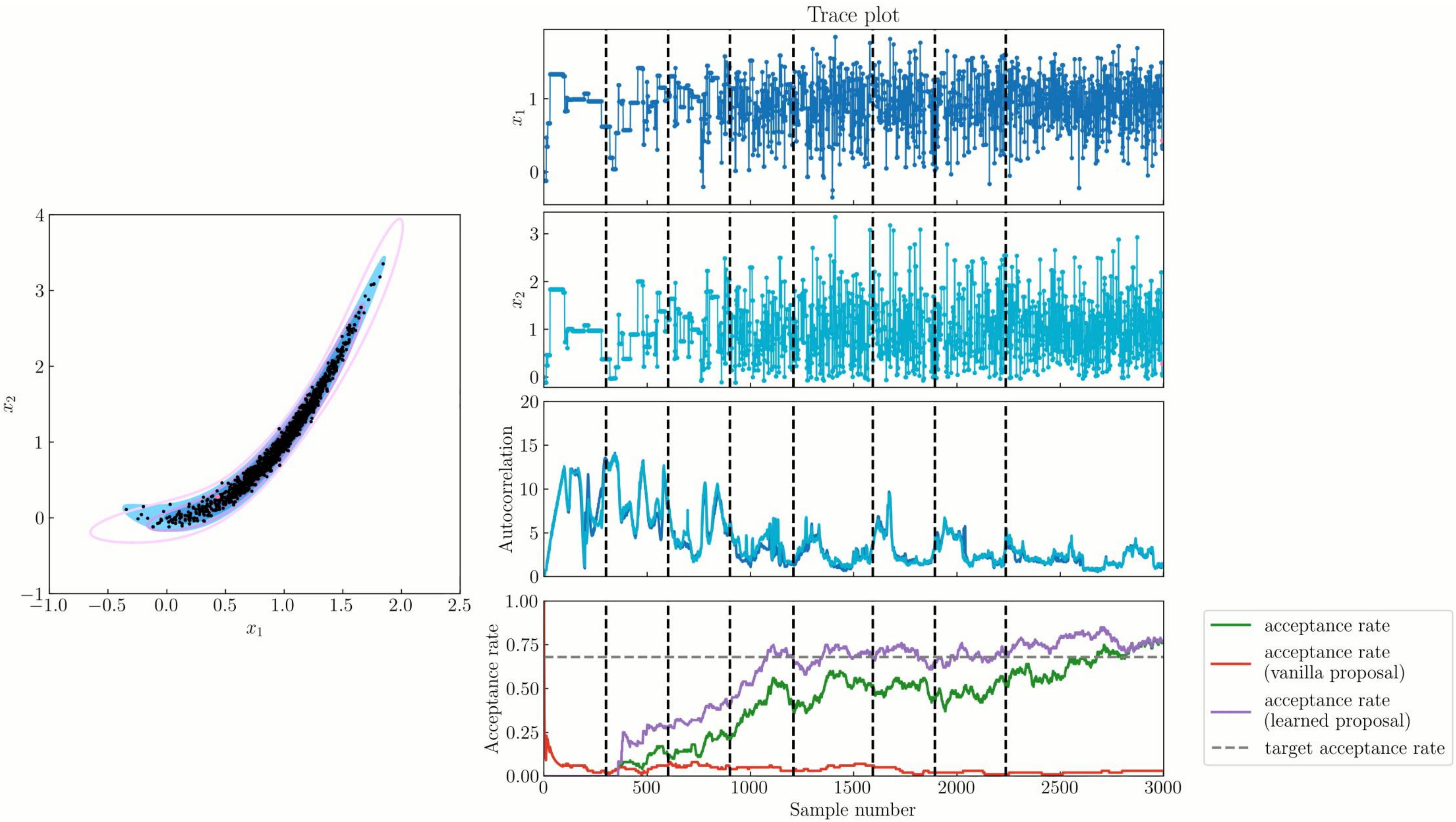
Papamakarios & Murray, 1605.06376; Greenberg *et al.*, 1905.07488; https://sbi-dev.github.io/

Scheduler for an adaptive MCMC

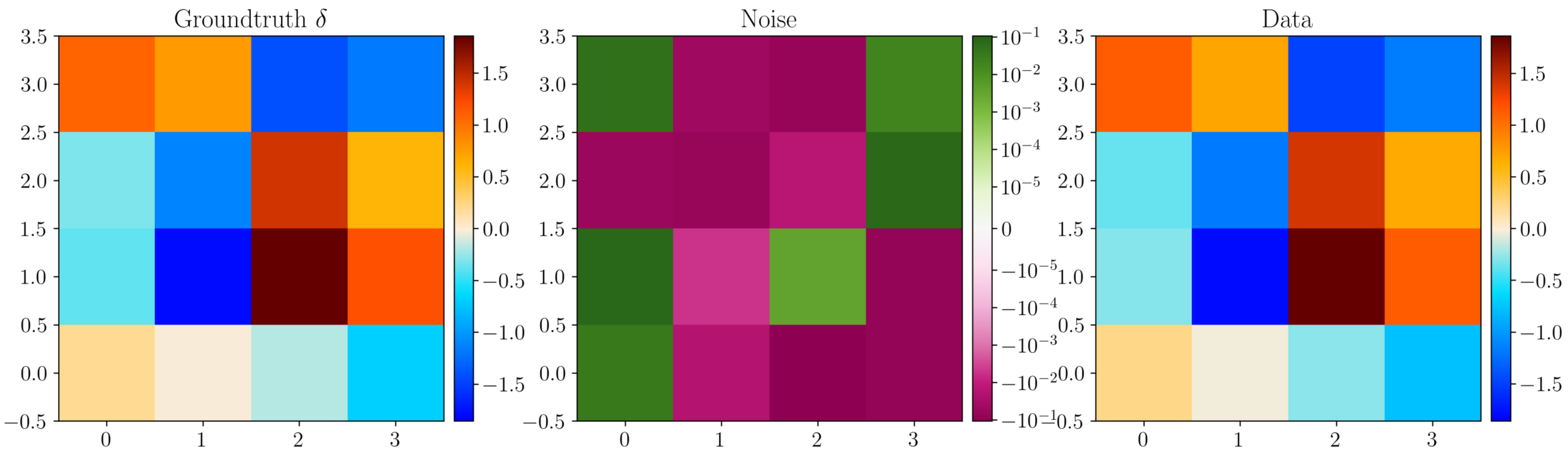
- Introducing weights in the loss function is equivalent to estimating $\mathcal{P}^*_w(\boldsymbol{x}|\boldsymbol{d}) \propto w(\boldsymbol{d})\,\mathcal{P}(\boldsymbol{x}|\boldsymbol{d})$ by minimising
$$D_{\mathrm{KL}}\left[\mathcal{P}^*_w||\mathcal{P}_\theta\right] = \int \mathcal{P}^*_w(\boldsymbol{x}|\boldsymbol{d}) \log \frac{\mathcal{P}^*_w(\boldsymbol{x}|\boldsymbol{d})}{\mathcal{P}_\theta(\boldsymbol{x}|\boldsymbol{d})}\,\mathrm{d}\boldsymbol{x}$$
  - The forward Kullback-Leibler divergence penalises *ignoring* probability mass, so it naturally makes the distribution broad/over-dispersed. This is what we want for a proposal distribution.
- Iterative fits of the replay buffer can be used to produce 'independence' proposal distributions (where $\mathcal{Q}(\boldsymbol{x}^*|\boldsymbol{x})$ does not depend on $\boldsymbol{x}$), in an adaptive MCMC framework.
  - The proposal distribution becomes increasingly effective (yielding high acceptance and low autocorrelation) as sampling continues.
  - We need a scheduler for the adaptation phase, for example:
    - Use the normalising flow proposal distribution with a probability equal to its current acceptance rate (with a window of 100* samples) and a minimum of 10%*, or a vanilla proposal distribution otherwise.
    - Train the normalising flow every 300* samples, lock-in the proposal distribution when the acceptance rate stays above a target (68%*) for 500* samples.
    - When the normalising flow proposal distribution is locked-in, use it with a probability of 99%* (keep the vanilla proposal distribution with probability of 1%* to avoid any unwarranted exclusion of parts of parameter space).

    (*: numbers used for testing, can be adapted to the problem and its dimensionality)

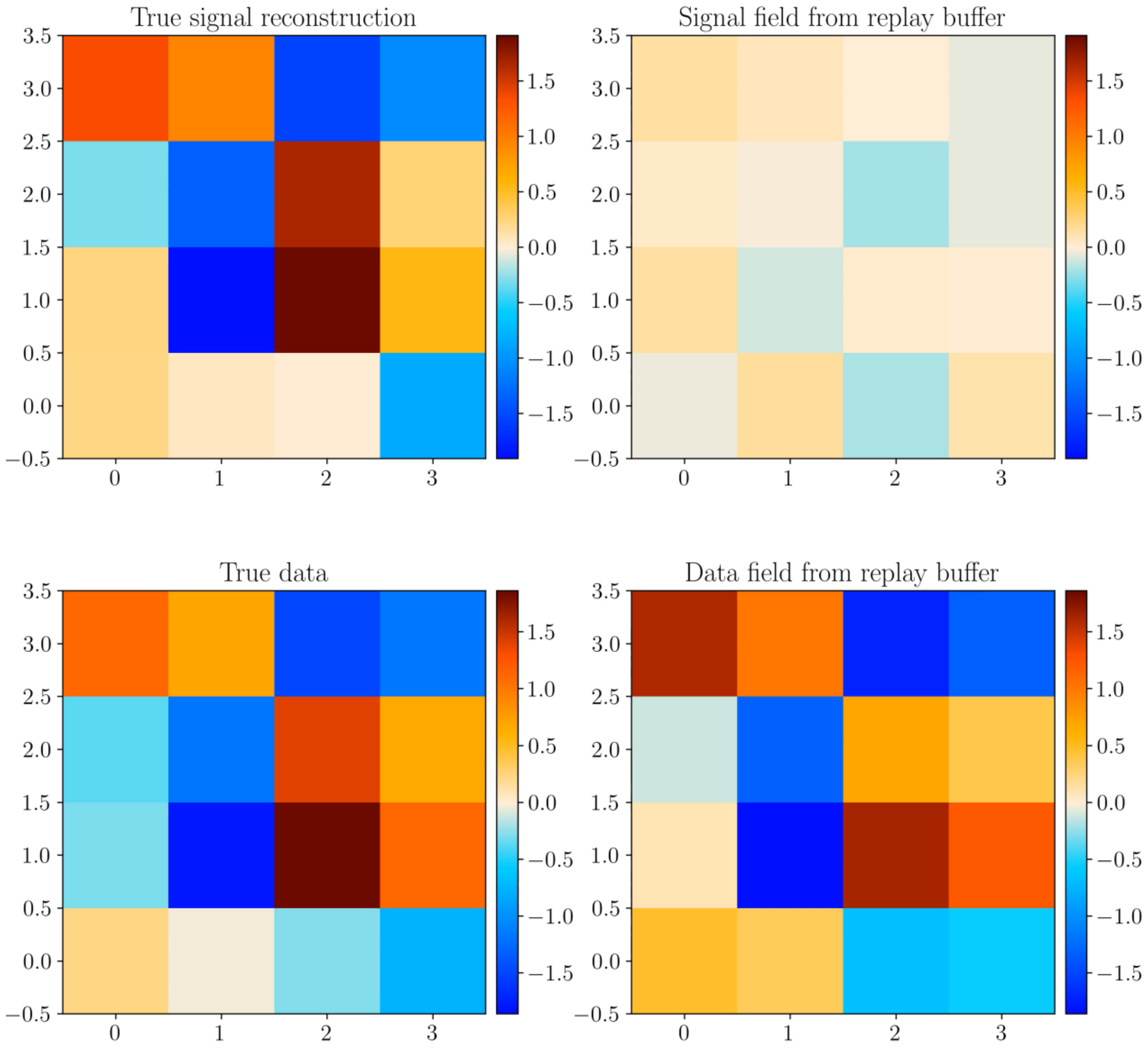
Counterfactual-informed adaptive MCMC with conditional normalising flows



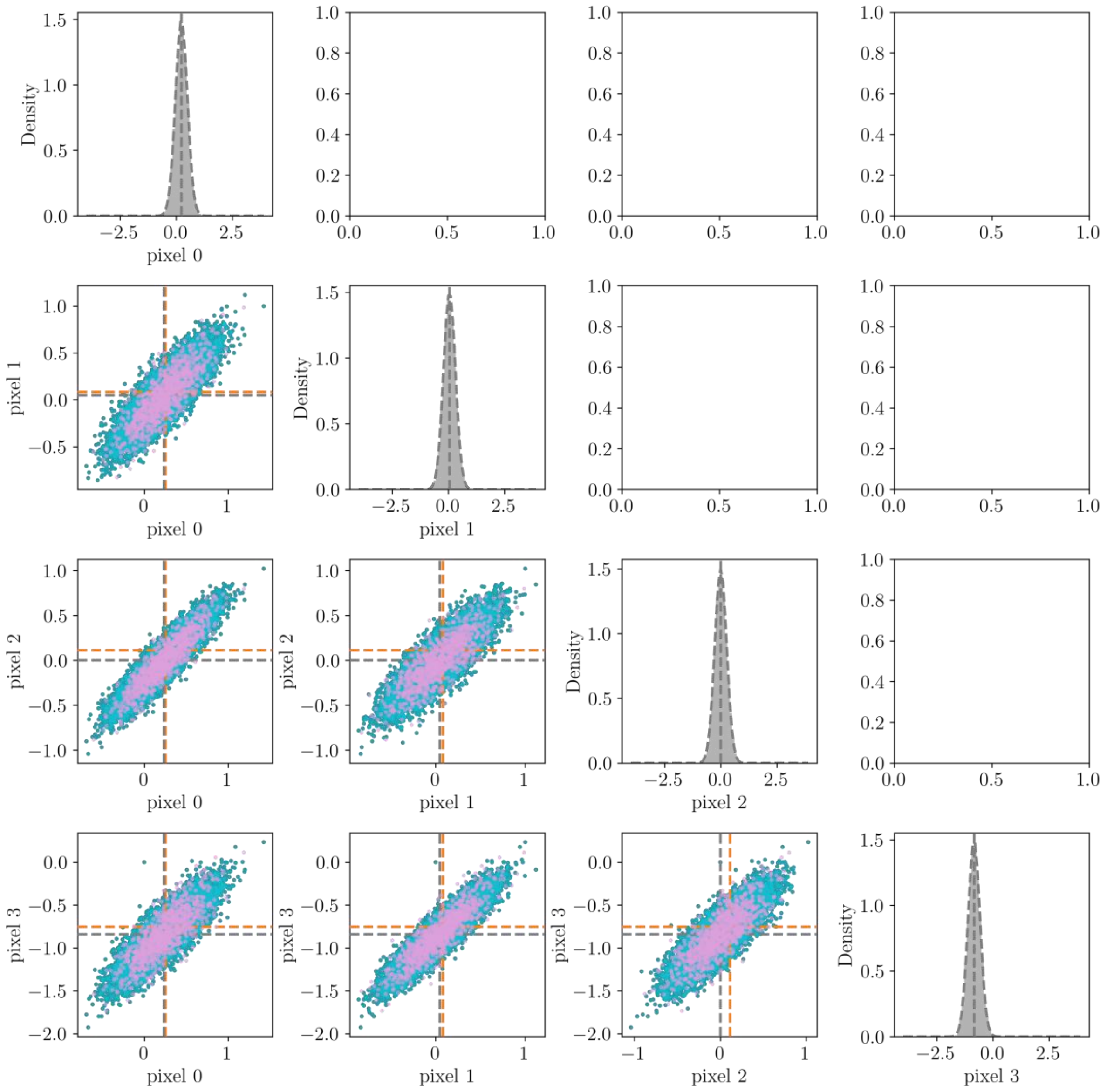
Field-level inference problems in cosmology



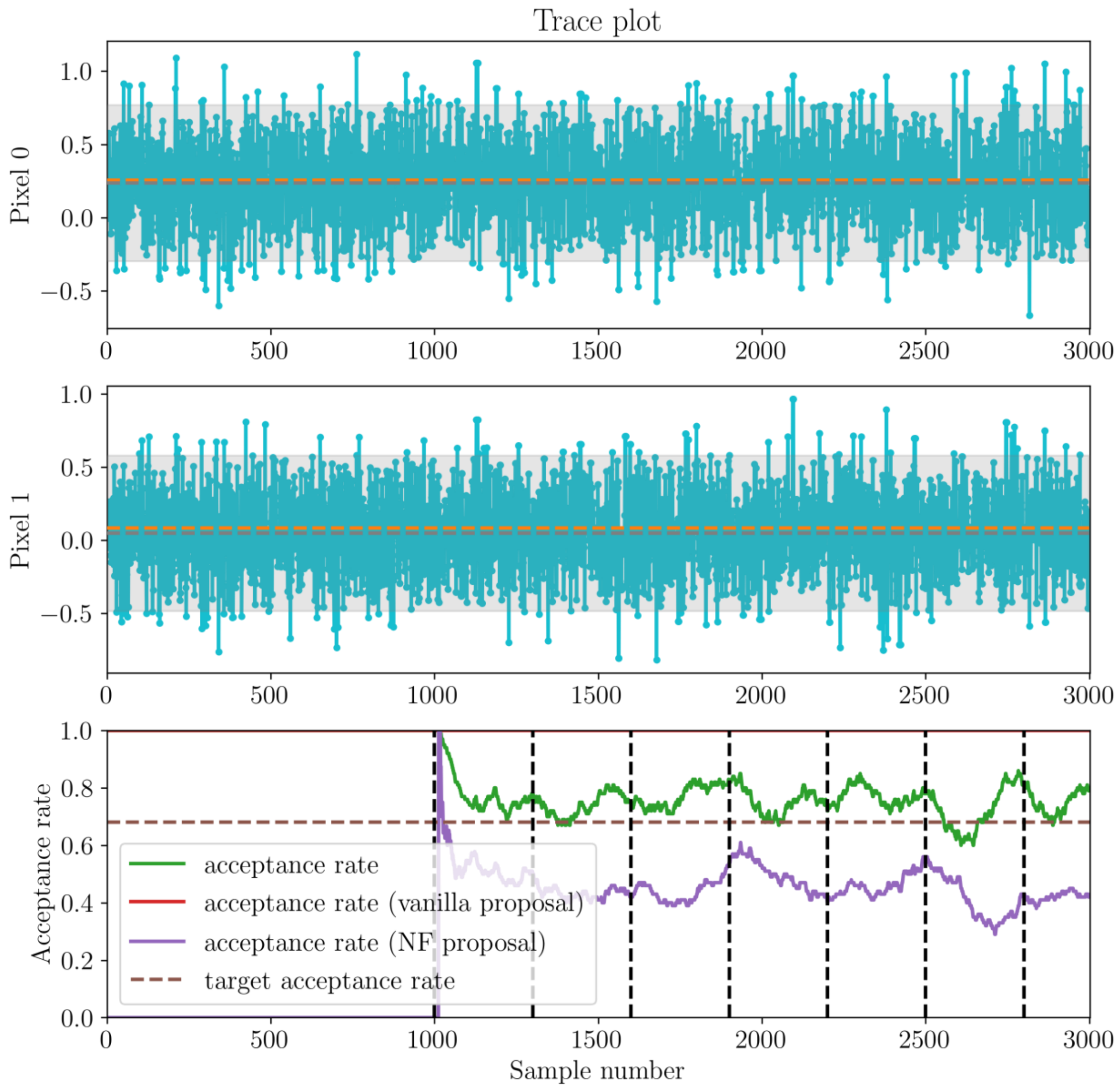
Replay buffer for a field-level inference problem



Fitting a conditional normalising flow to the replay buffer



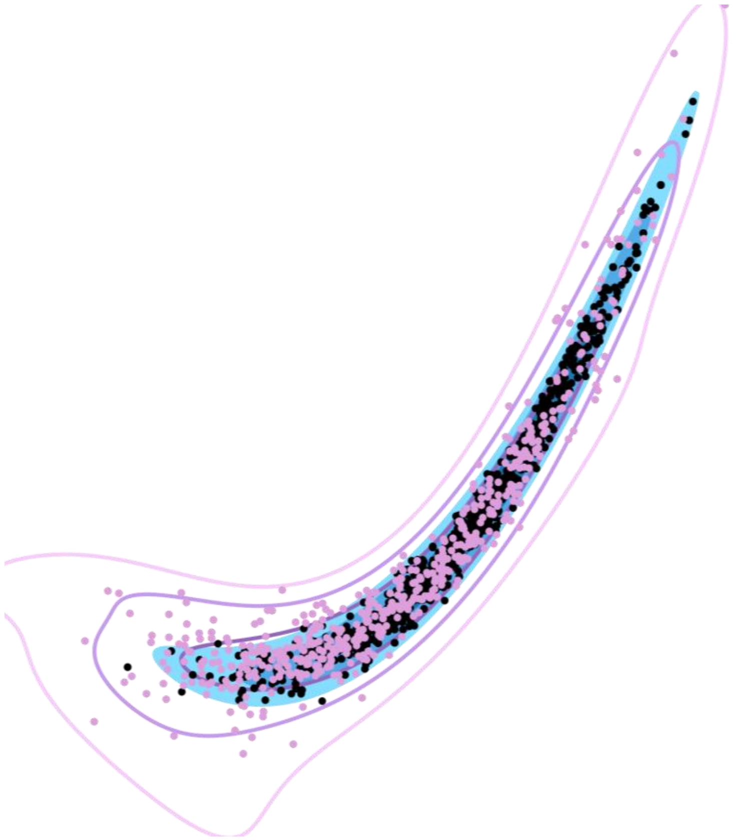
Counterfactual-informed adaptive MCMC with conditional normalising flows



Conclusion and outlook

Counterfactual-informed adaptive MCMC with conditional normalising flows

- We propose:
  - an adaptive MCMC framework, …
  - where an efficient proposal distribution is iteratively learned, …
  - using conditional normalising flows, …
  - trained on a replay buffer that contains both samples at the true data and alternative data (counterfactuals)
- We have successfully tested the framework on low-dimensional problems, the challenge ahead is to scale it.
- Outlook and possible improvements (feedback welcome!):
  - Several MCMC chains running in parallel, gathering their data for a joint replay buffer.
  - Or several MCMC chains, each learning its proposal distribution only from what is produced by other (so that we do not break the Markov property).
  - We (only, but accurately) need to be able to sample and evaluate log density-ratios from the trained model. Can other ML models be used? e.g. conditional score diffusion models?



Acknowledgements, credits, contacts

Slides at:
florent-leclercq.eu/talks.php

Reference:
- Leclercq & Jasche, in prep.

We are currently offering several postdocs in Paris & Stockholm!

Visit us at www.aquila-consortium.org

- The author acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-23-CE46-0006 (project INFOCW).

www.florent-leclercq.eu