BAYESIAN INFERENCE WITH LATENT SPACE SAMPLING

NIKOLAJ TAKATA MÜCKE – NIKOLAJ.MUCKE@CWI.NL

POSTDOC AT CWI

14/11/2024

JOINT WORK WITH:

CORNELIS OOSTERLEE, SANDER BOHTÉ, AND BENJAMIN SANDERSE

WHAT IS THE PROBLEM?

- Bayesian inference is great!
- But it is typically too slow for real-time problems
- … Especially when problems are high-dimensional and non-Gaussian

OVERVIEW

- Static Bayesian inference
  - Markov Chain Monte Carlo
  - Generative Adversarial Networks (GANs)

- Dynamic Bayesian inference (filtering)
  - Particle filter
  - Wasserstein Autoencoders

STATIC BAYESIAN INFERENCE

BAYESIAN INFERENCE

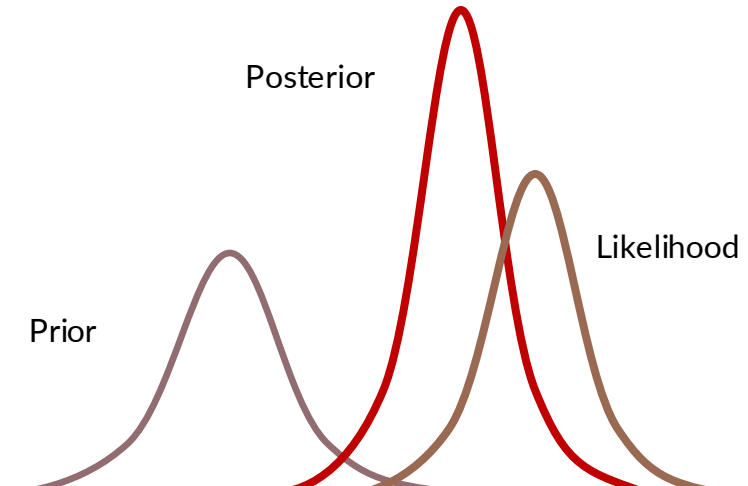
- **Framework** for solving inverse problems in a probabilistic setting

- **Result:** posterior distribution over parameters and/or state

Posterior Likelihood Prior

$$\rho_{u|y}(\mathbf{u}|\mathbf{y}) = \frac{\rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})}{\int_{\mathbb{R}^{N_u}} \rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})\,\mathrm{d}\mathbf{u}}$$

State + parameters Observations Evidence



HOW DO WE COMPUTE THE POSTERIOR?

- Posterior is not analytically available
- We need numerical techniques!
- Sampling?

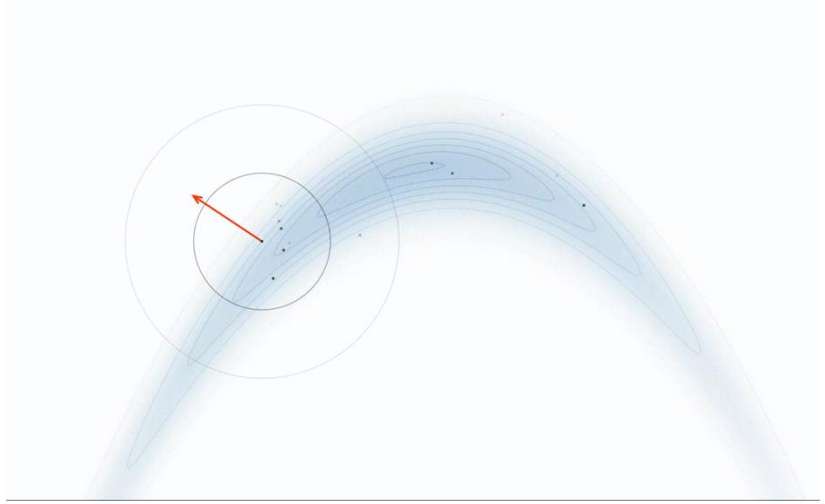
$$\rho_{u|y}(\mathbf{u}|\mathbf{y}) = \frac{\rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})}{\int_{\mathbb{R}^{N_u}} \rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})\,\mathrm{d}\mathbf{u}}$$

https://github.com/chi-feng/mcmc-demo

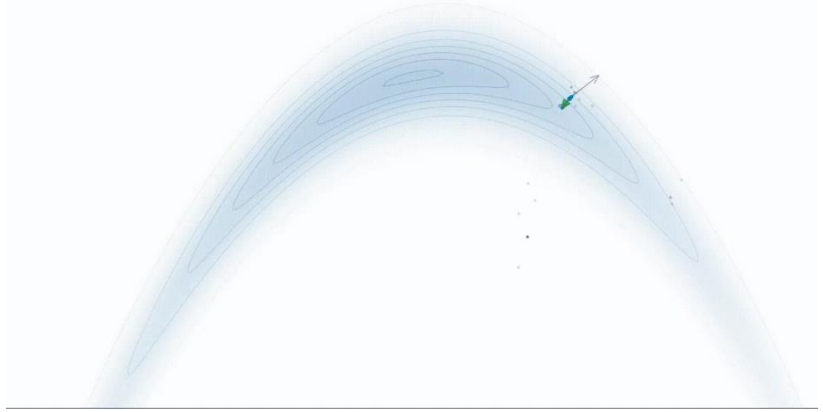
MARKOV CHAIN MONTE CARLO (MCMC)

- Sample from posterior distribution by creating Markov chain
- Accept/reject proposed samples
- Converges weakly towards true distribution

Metropolis-Hastings

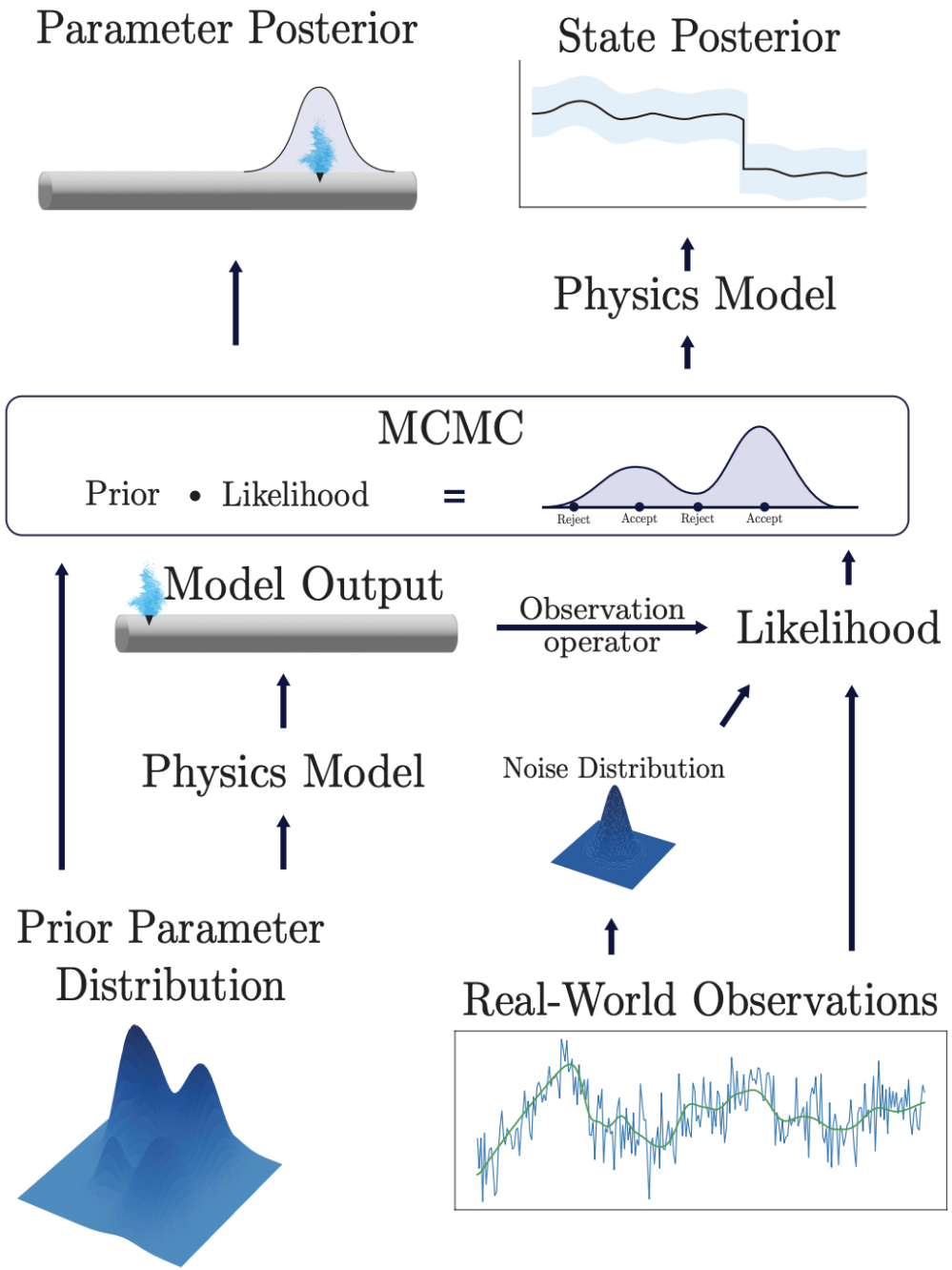


HMC with NUTS



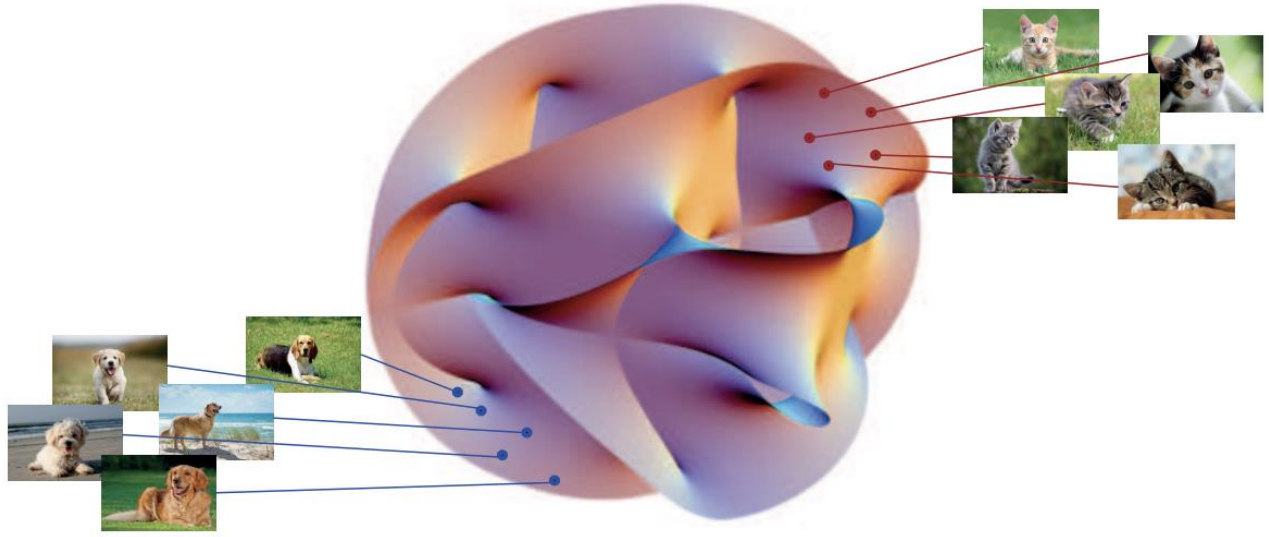
BAYESIAN INFERENCE WITH MCMC

1. Sample from prior distribution — What is a good prior?
2. Evaluate forward model — Expensive!
3. Compute likelihood
4. Accept/reject with MCMC — Requires many samples



GENERATIVE DEEP LEARNING?

- Efficient sampling of unkown distribution
- Dimensionality reduction



Gitta Kutyniok - Solving Mathematical Problems by Deep Learning: Partial Differential Equations

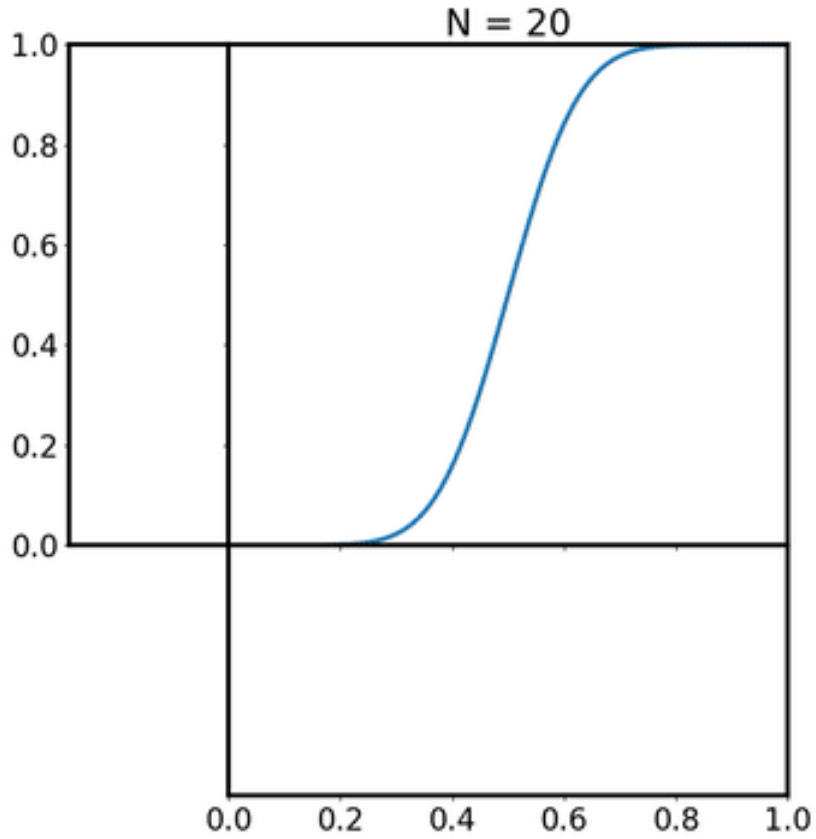
GENERATING DISTRIBUTIONS VIA PUSH-FORWARD MAP

1. Sample from simple distribution

2. Pass sample through function (push-forward map)
   - Generator or decoder

3. Get data sample

Inverse transform sampling

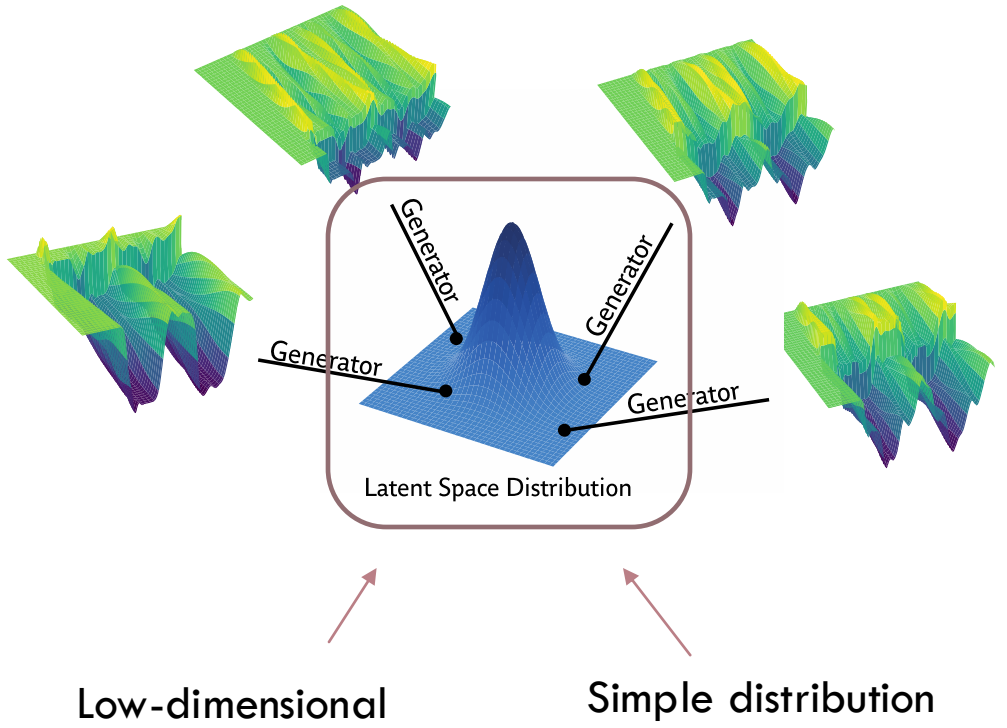


GENERATING DISTRIBUTIONS VIA PUSH-FORWARD MAP

1. Sample from latent distribution

2. Pass sample through function (push-forward map)
   - Generator or decoder

3. Get data sample

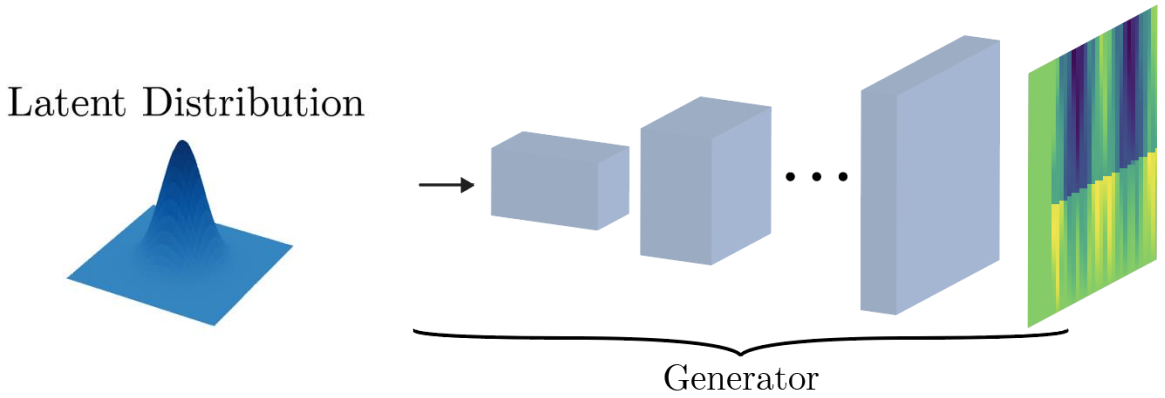


GENERATOR DISTRIBUTION

- Generator maps latent samples to high-fidelity samples: $U = G_\theta(Z)$

- Generator pushes forward the latent distribution:

$$U \sim P_u^g = G_{\theta\#} P_z^g \qquad \rho_u^g = \rho_z^g \circ G^{-1}$$



WASSERSTEIN GANS

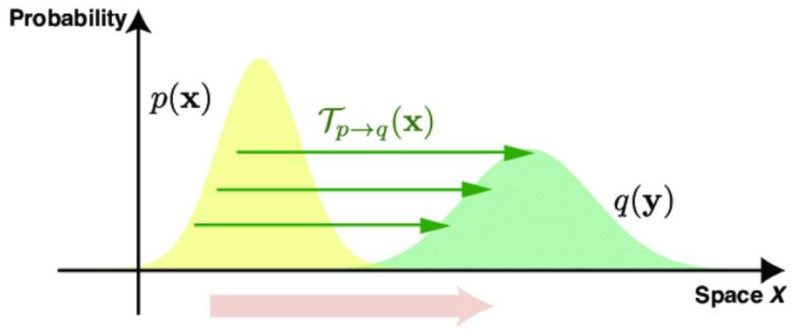
WASSERSTEIN DISTANCE

$$W_1(P_1, P_2) = \inf_{\gamma \in \Gamma(P_1, P_2)} \left| \int_E \int_E d_E(x, y)\gamma(x, y)\mathrm{d}x\mathrm{d}y \right|$$

Kantorovich–Rubinstein duality

$$W_1(P_1, P_2) = \sup_{\mathrm{Lip}(f) \leq 1} \left| \int_E f(x)\rho_1(x)\mathrm{d}x - \int_E f(x)\rho_2(x)\mathrm{d}x \right|$$

"Earth mover distance"



TRAINING OBJECTIVE

Real distribution

$$\min_{P_u^g} W_1(P_u^r, P_u^g) = \sup_{\mathrm{Lip}(f)\leq 1} \left| \int_E f(x)\rho_u^r(x)\mathrm{d}x - \int_E f(x)\rho_u^g(x)\mathrm{d}x \right|$$

Generator distribution

Discriminator

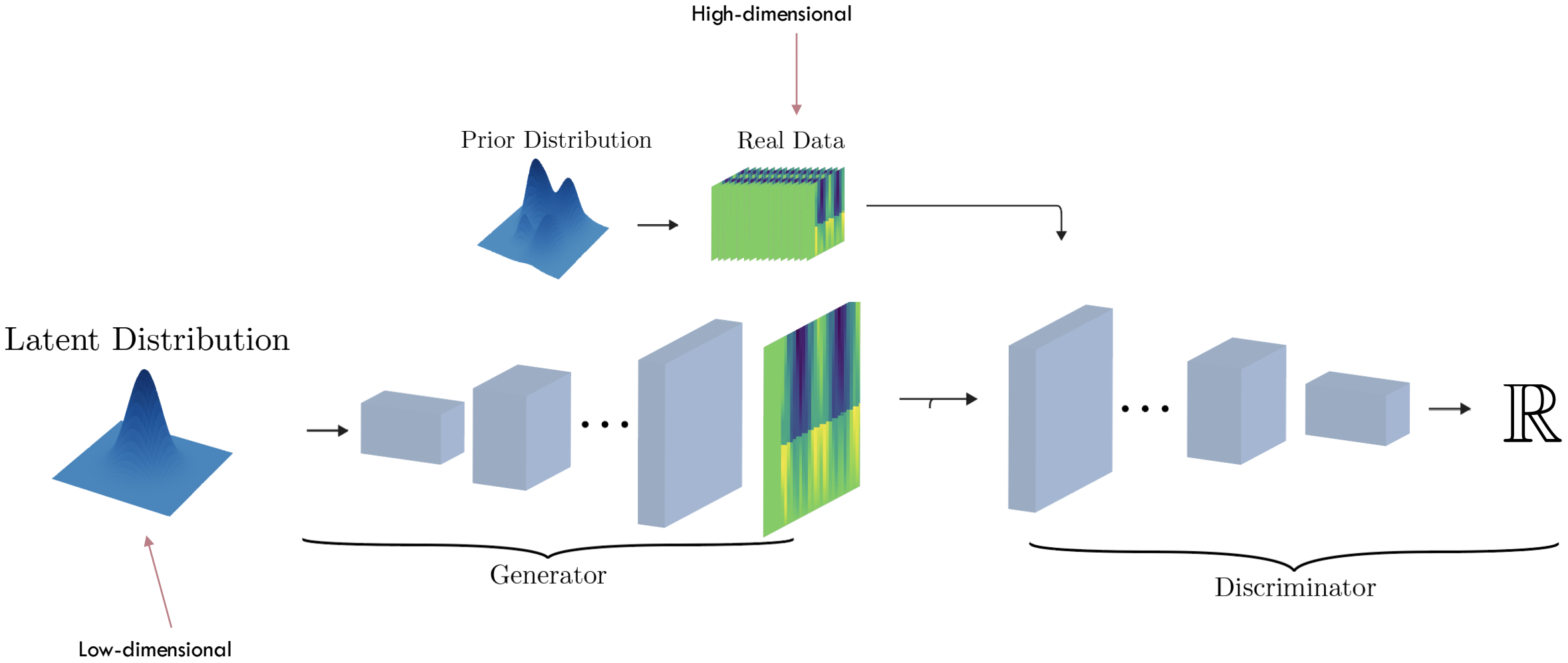
$$\min_{P_u^g} W_1(P_g^r, P_u^g) = \sup_{\mathrm{Lip}(f)\leq 1} \left| \mathbb{E}_{X\sim P_u^r}[f(X)] - \mathbb{E}_{X\sim P_u^g}[f(X)] \right|$$

Generator

$$\min_{\theta} \max_{\omega} L(\theta, \omega) = \left| \mathbb{E}_{X\sim P_u^r}[D_\omega(X)] - \mathbb{E}_{Z\sim P_z}[D_\omega(G(Z))] \right|$$

Latent distribution

GAN LAYOUT


High-dimensional
Prior Distribution
Real Data
Latent Distribution
Generator
Discriminator
$\mathbb{R}$
Low-dimensional

LATENT SPACE POSTERIOR

LATENT SPACE POSTERIOR

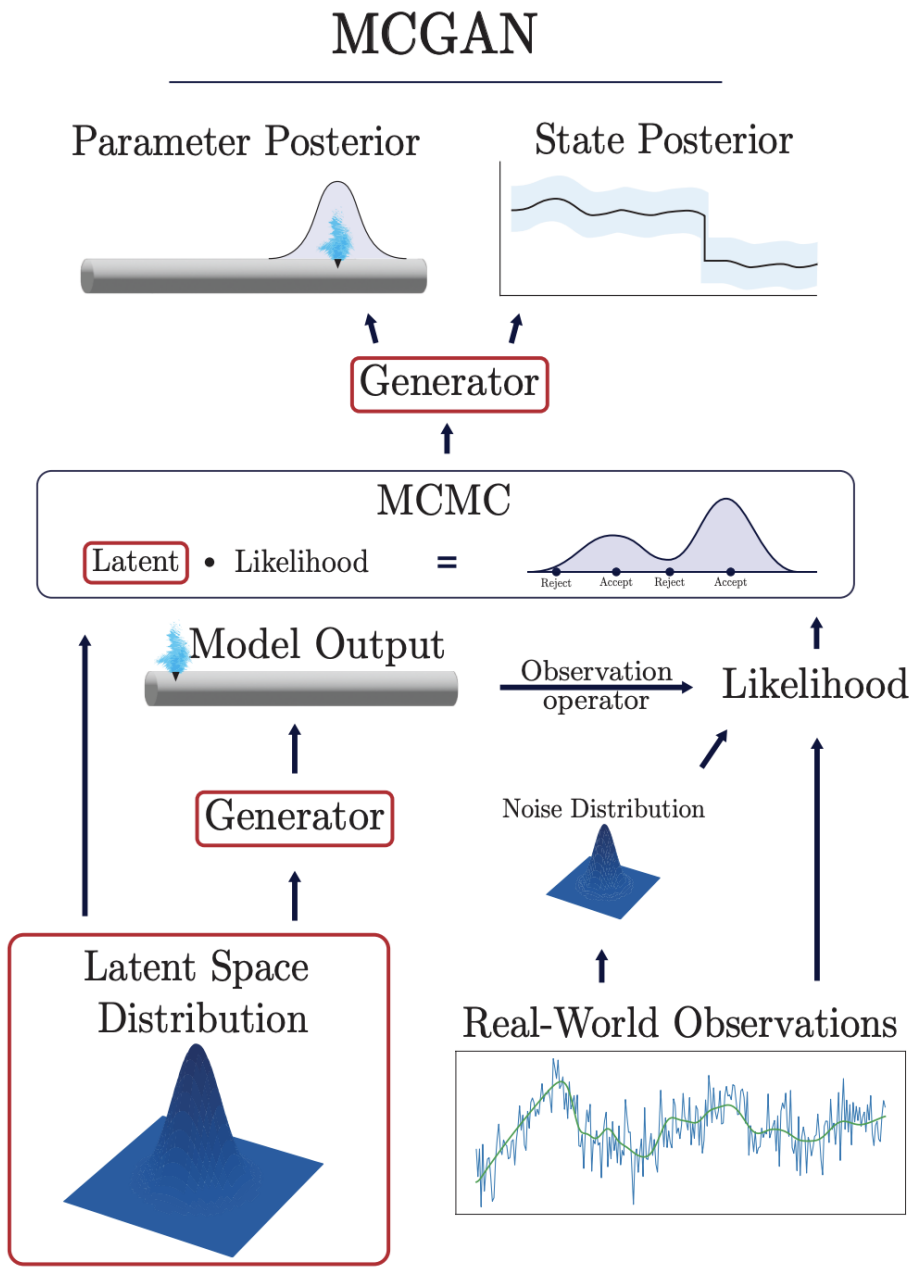
- Train GAN on prior samples of parameters and/or state
- Replace prior distribution with latent distribution
- Compute posterior in latent space!
  - Maximum a-posteriori is now (almost) free
  - Sample Markov chain in latent space

$$\rho_{u|y}(\mathbf{u}|\mathbf{y}) = \frac{\rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})}{\int_{\mathbb{R}^{N_u}} \rho_{y|u}(\mathbf{y}|\mathbf{u})\rho_0(\mathbf{u})\,\mathrm{d}\mathbf{u}}$$

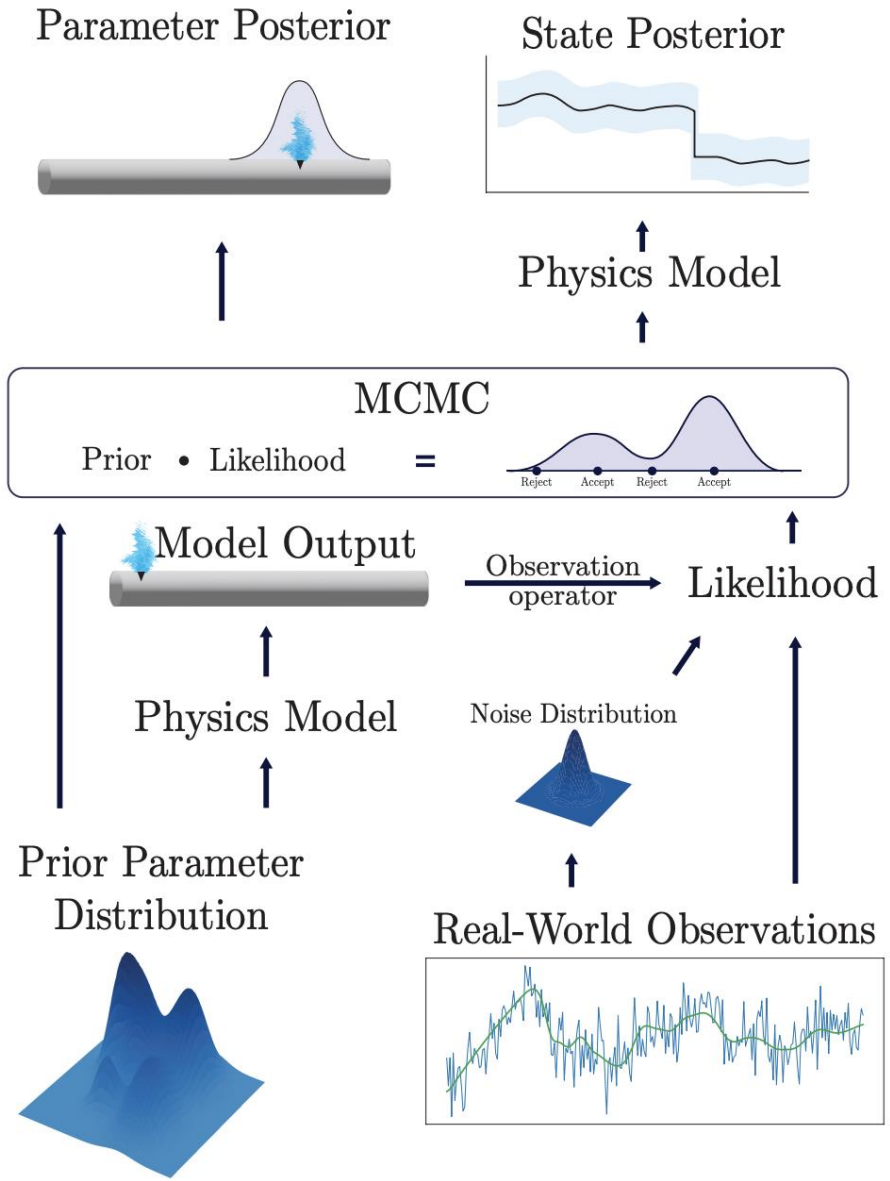
$$\rho^g_{z|y}(\mathbf{z}|\mathbf{y}) = \frac{\rho^g_{y|u}(\mathbf{y}|G_\theta(\mathbf{z}))\rho^g_z(\mathbf{z})}{\int_{\mathbb{R}^{N_z}} \rho^g_{y|u}(\mathbf{y}|G_\theta(\mathbf{z}))\rho^g_z(\mathbf{z})\,d\mathbf{z}}$$

MARKOV CHAIN GAN (MCGAN)

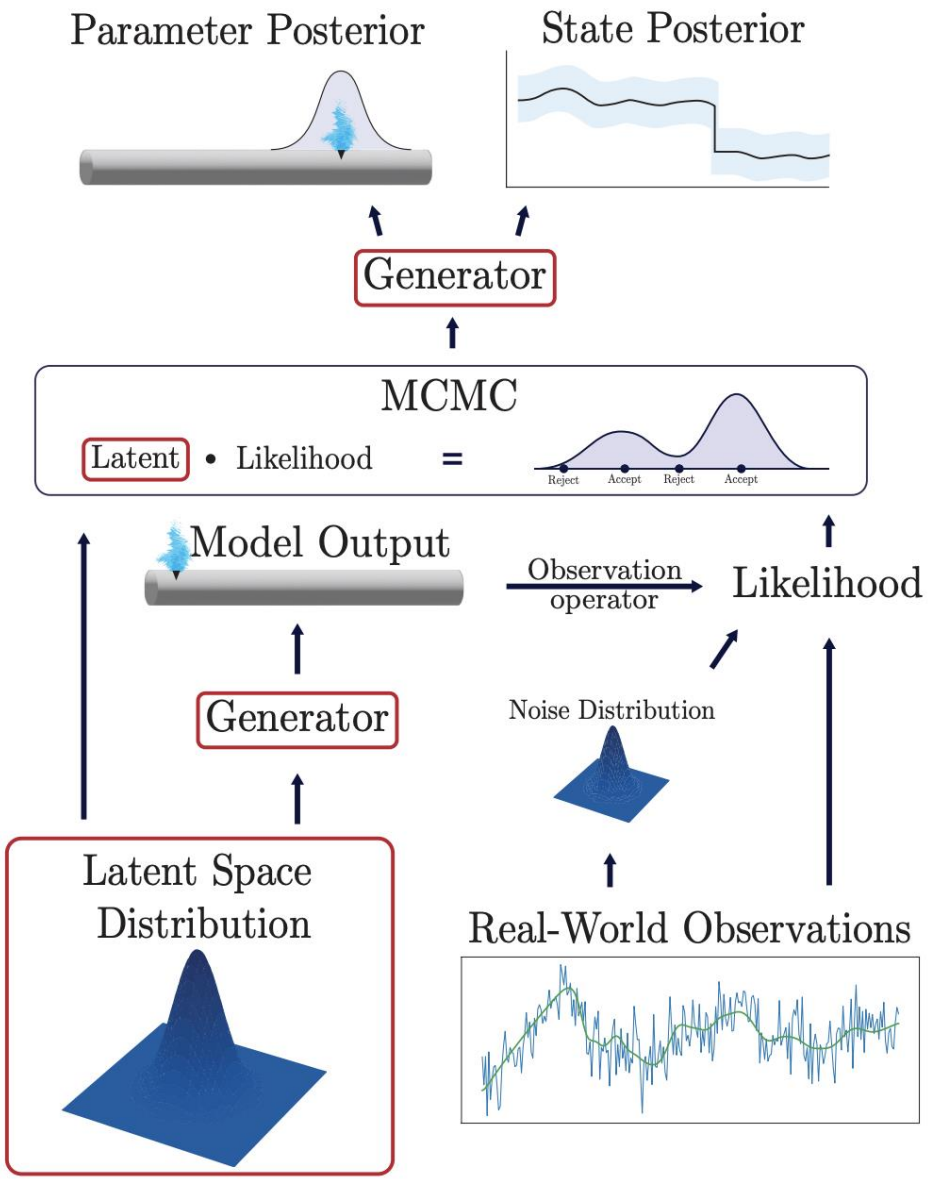
1. Sample from latent distribution
2. Push forward using generator
3. Compute likelihood
4. Accept/reject with MCMC
5. Push forward accepted latent samples using generator



Bayesian Inference with MCMC



MCGAN



LATENT SPACE SAMPLING

- Sampling latent posterior and pushing forward is weakly the same as sampling high-fidelity posterior

$$\mathbb{E}_{U\sim P^{g}_{u|y}}[f(U)] = \mathbb{E}_{Z\sim P^{g}_{z|y}}[f(G(Z))]$$

- Change of variable formula

$$\mathbb{E}_{U\sim P_u}[f(U)] = \int_E f(\mathbf{u})\rho_u(\mathbf{u})\mathrm{d}\mathbf{u} = \int_{G^{-1}(E)} f(G(\mathbf{z}))\rho_z(\mathbf{z})\mathrm{d}\mathbf{z} = \mathbb{E}_{Z\sim P_z}[f(G(Z))]$$

CONVERGENCE OF POSTERIOR

- GAN converges to true prior in Wasserstein 1 metric → Posterior also converges

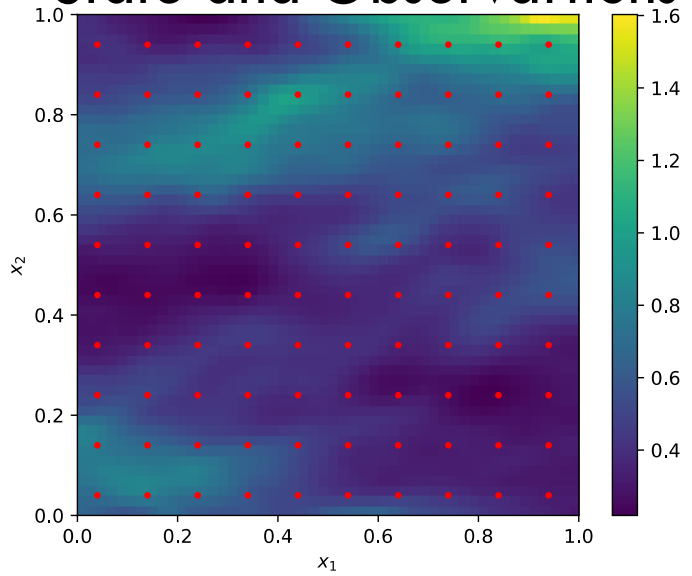
$$W_1(P^r_{u|y}, P^g_{u|y}) \leq C_1 W_1(P^r_0, P^r_0) + C_2 ||l^r(\mathbf{u}) - l^g(\mathbf{u})||_{L^1_{\rho^g_0}} + C_3 ||l^r(\mathbf{u}) - l^g(\mathbf{u})||_{L^2_{\rho^g_0}}$$

$$C_1 = \frac{(1 + D\text{Lip}(\Phi^r))}{Q^r_u(\mathbf{y})}, \quad C_2 = \frac{\max(\Phi^r, \Phi^g)}{Q^r_u(\mathbf{y}) Q^g_u(\mathbf{y})} (1 + D\text{Lip}(\Phi^r)) |P^g_0|_{\mathcal{W}_1}, \quad C_3 = \frac{\max(\Phi^r, \Phi^g)}{Q^g_u(\mathbf{y})} |P^g_0|_{\mathcal{W}_2}$$

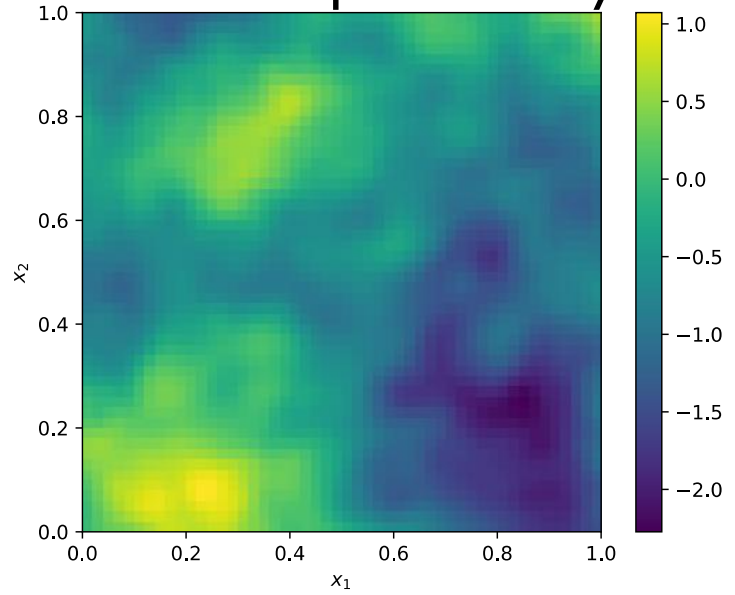
DARCY FLOW

$$
\begin{aligned}
\mathbf{v} + k\nabla p &= 0, && \mathbf{x} \in [0,1]^2, \\
\nabla \cdot \mathbf{v} &= 0, && \mathbf{x} \in [0,1]^2, \\
p &= 1, && \mathbf{x} \in 0 \times [0,1], \\
p &= 0, && \mathbf{x} \in 1 \times [0,1], \\
\mathbf{v} \cdot \mathbf{n} &= 0, && \mathbf{x} \in [0,1] \times \{0,1\}.
\end{aligned}
$$

State and Observartions

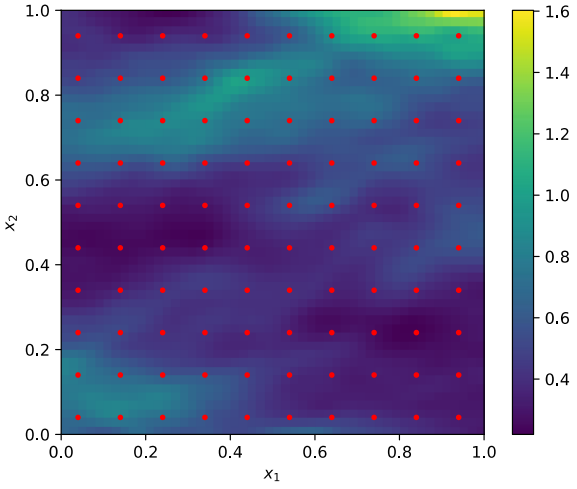


Parameters – permebility field



RECONSTRUCTION

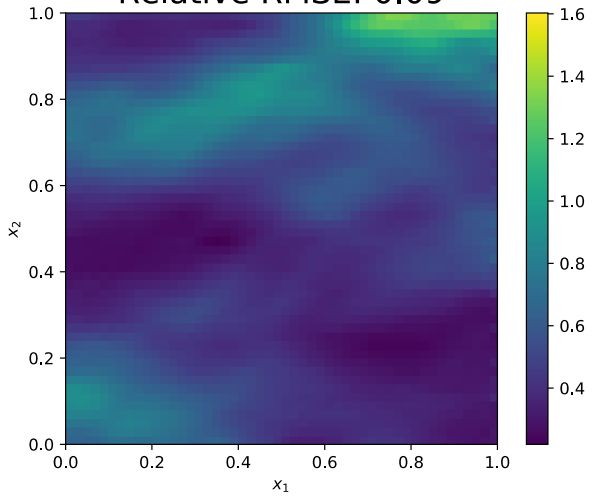
True State



dim (u) = 25,793

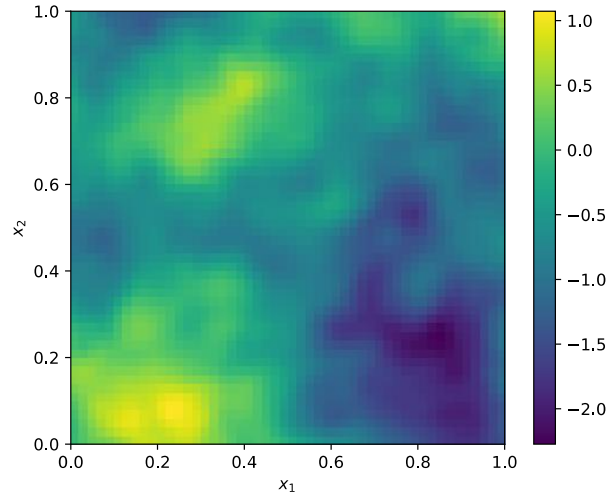
MCGAN

Relative RMSE: 0.09



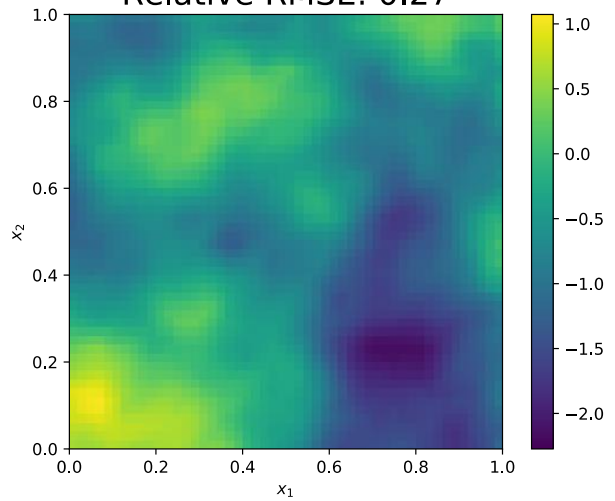
dim (z) = 150

True Parameters



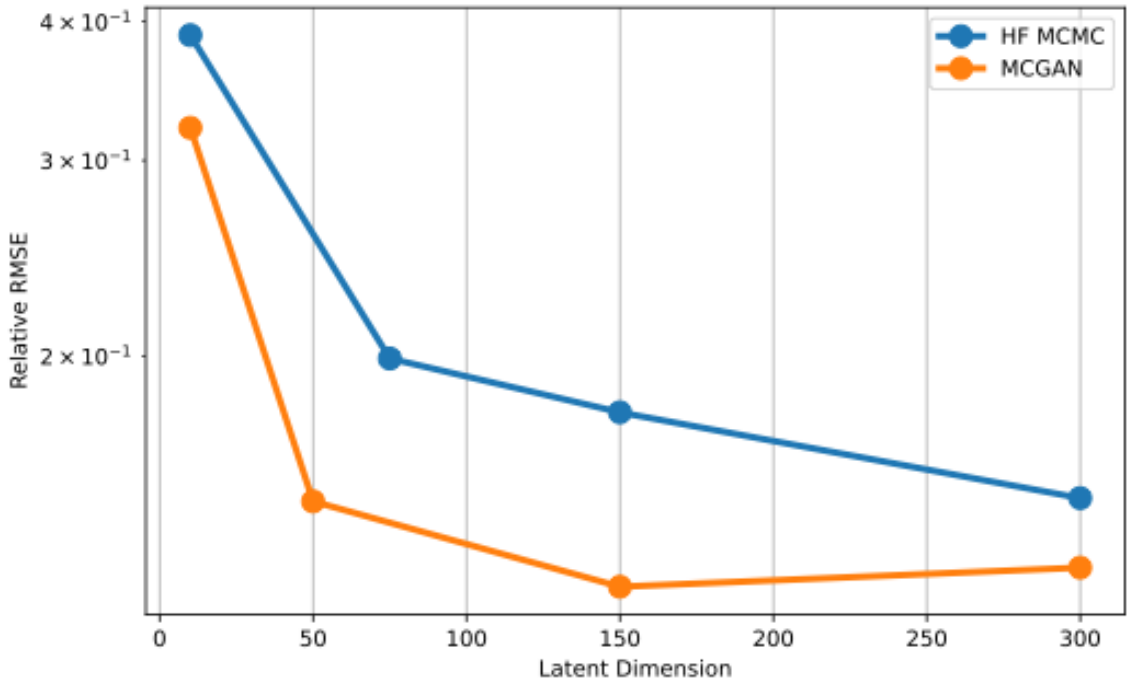
MCGAN Parameters

Relative RMSE: 0.27

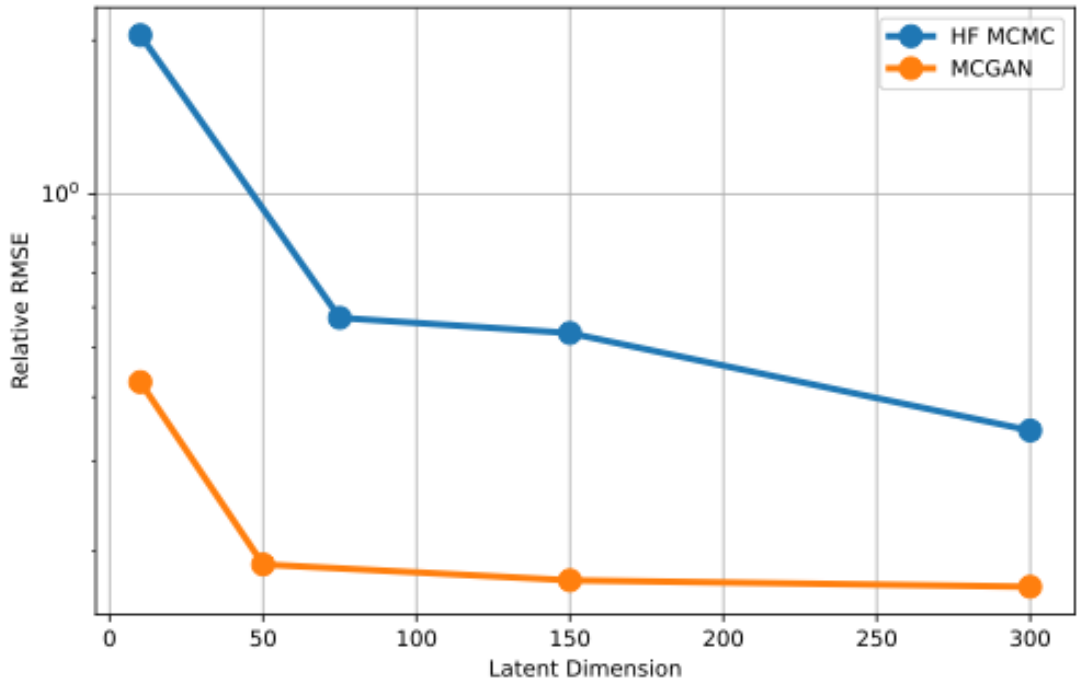


COMPARISON WITH KARHUNEN-LOEVE EXPANSION

State



Parameters

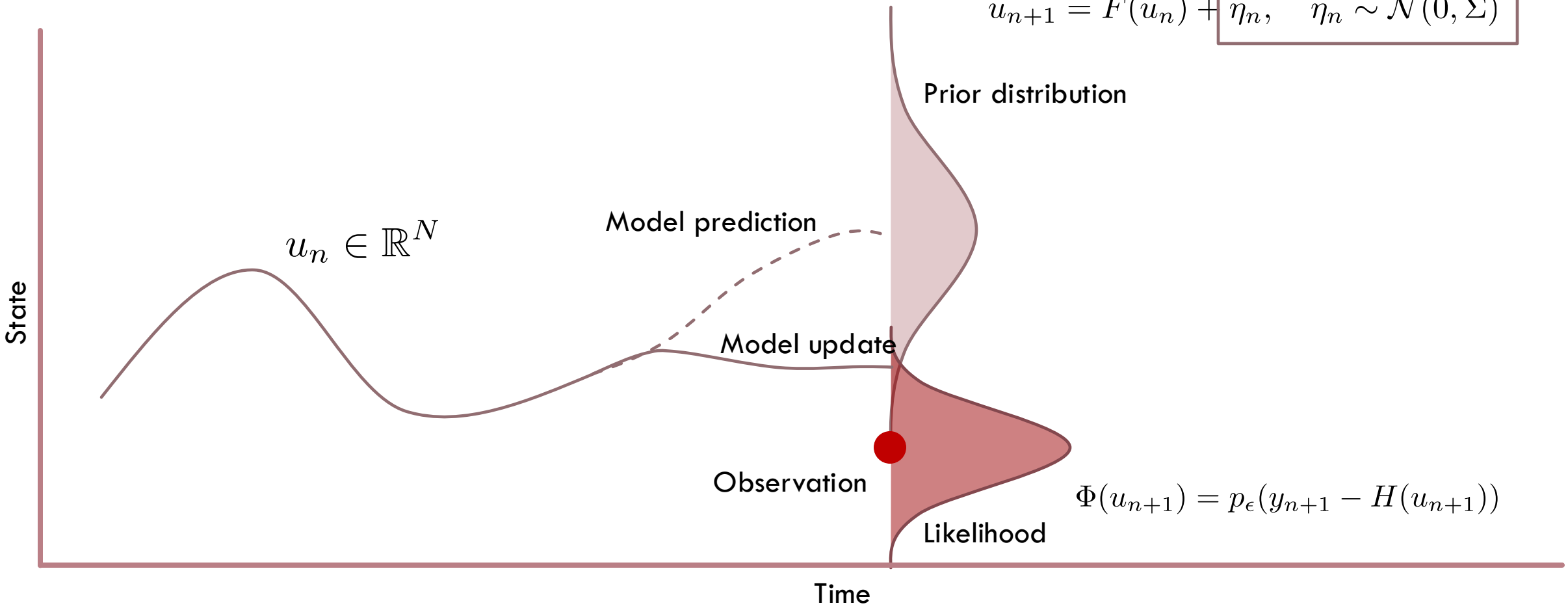


DYNAMIC BAYESIAN INFERENCE

BAYESIAN DATA ASSIMILATION

Model error term

$$u_{n+1} = F(u_n) + \eta_n, \quad \eta_n \sim \mathcal{N}(0, \Sigma)$$

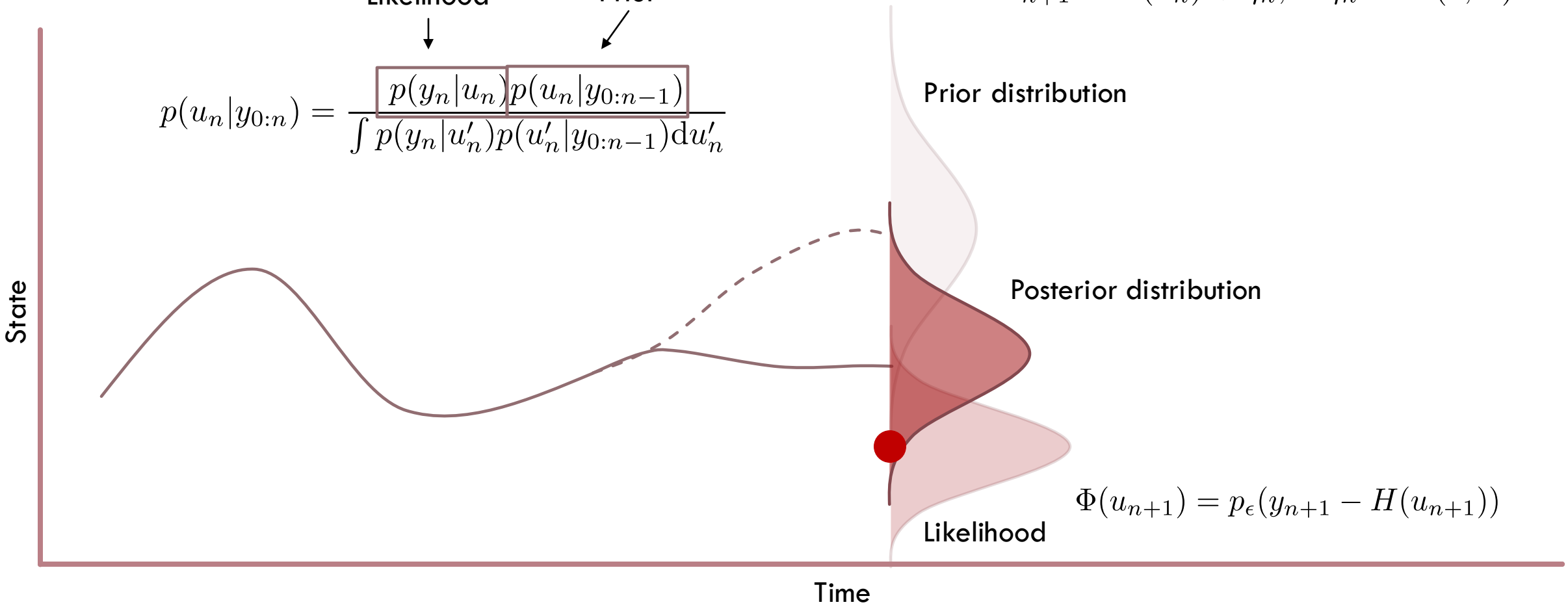


BAYESIAN DATA ASSIMILATION

$$u_{n+1} = F(u_n) + \eta_n, \quad \eta_n \sim \mathcal{N}(0, \Sigma)$$

Likelihood Prior

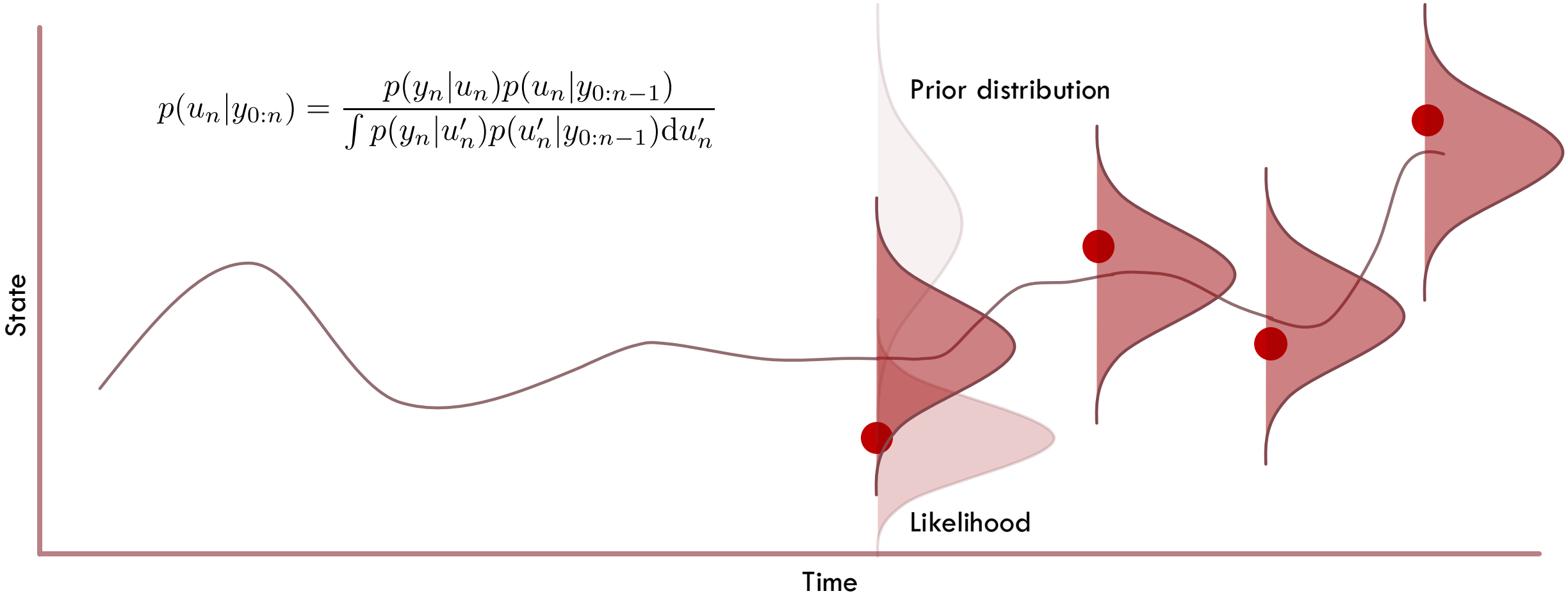
$$p(u_n|y_{0:n}) = \frac{p(y_n|u_n)p(u_n|y_{0:n-1})}{\int p(y_n|u'_n)p(u'_n|y_{0:n-1})\mathrm{d}u'_n}$$



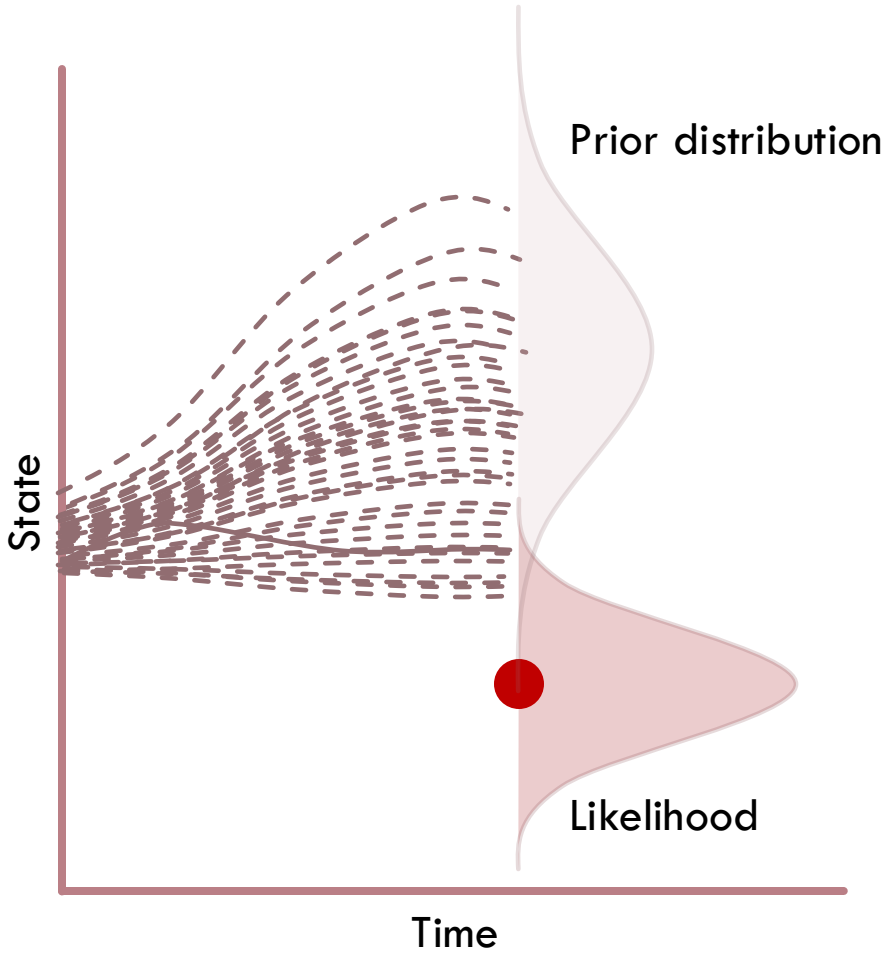
$$\Phi(u_{n+1}) = p_\epsilon(y_{n+1} - H(u_{n+1}))$$

BAYESIAN DATA ASSIMILATION

$$p(u_n|y_{0:n}) = \frac{p(y_n|u_n)p(u_n|y_{0:n-1})}{\int p(y_n|u'_n)p(u'_n|y_{0:n-1})\mathrm{d}u'_n}$$



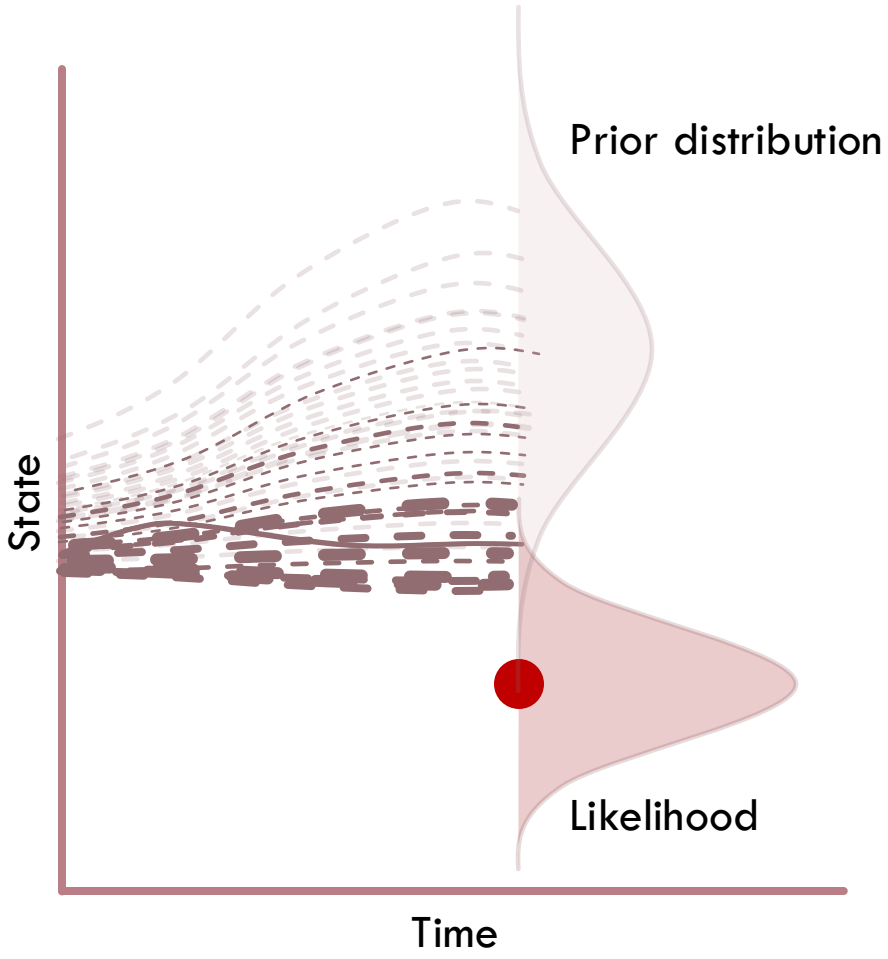
PARTICLE FILTER



1. Compute prior particles

$$\{(w_n^i, u_n^i)\}_{i=0}^{N_p}$$

PARTICLE FILTER



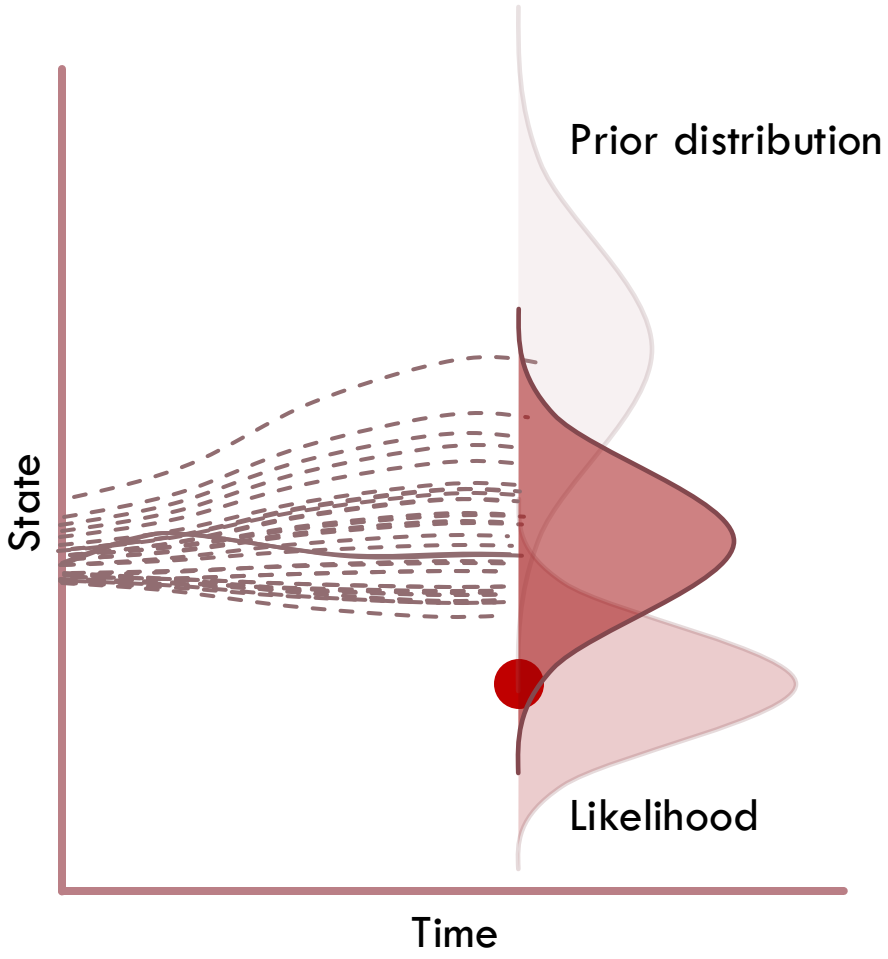
1. Compute prior particles

$$\left\{(w_n^i, u_n^i)\right\}_{i=0}^{N_p}$$

2. Update weights

$$\tilde{w}_{n+1}^i = w(u_n)p(y_{n+1}|u_{n+1}) = w(u_n) \cdot \text{likelihood}$$

PARTICLE FILTER



1. Compute prior particles

$$\left\{(w_n^i, u_n^i)\right\}_{i=0}^{N_p}$$

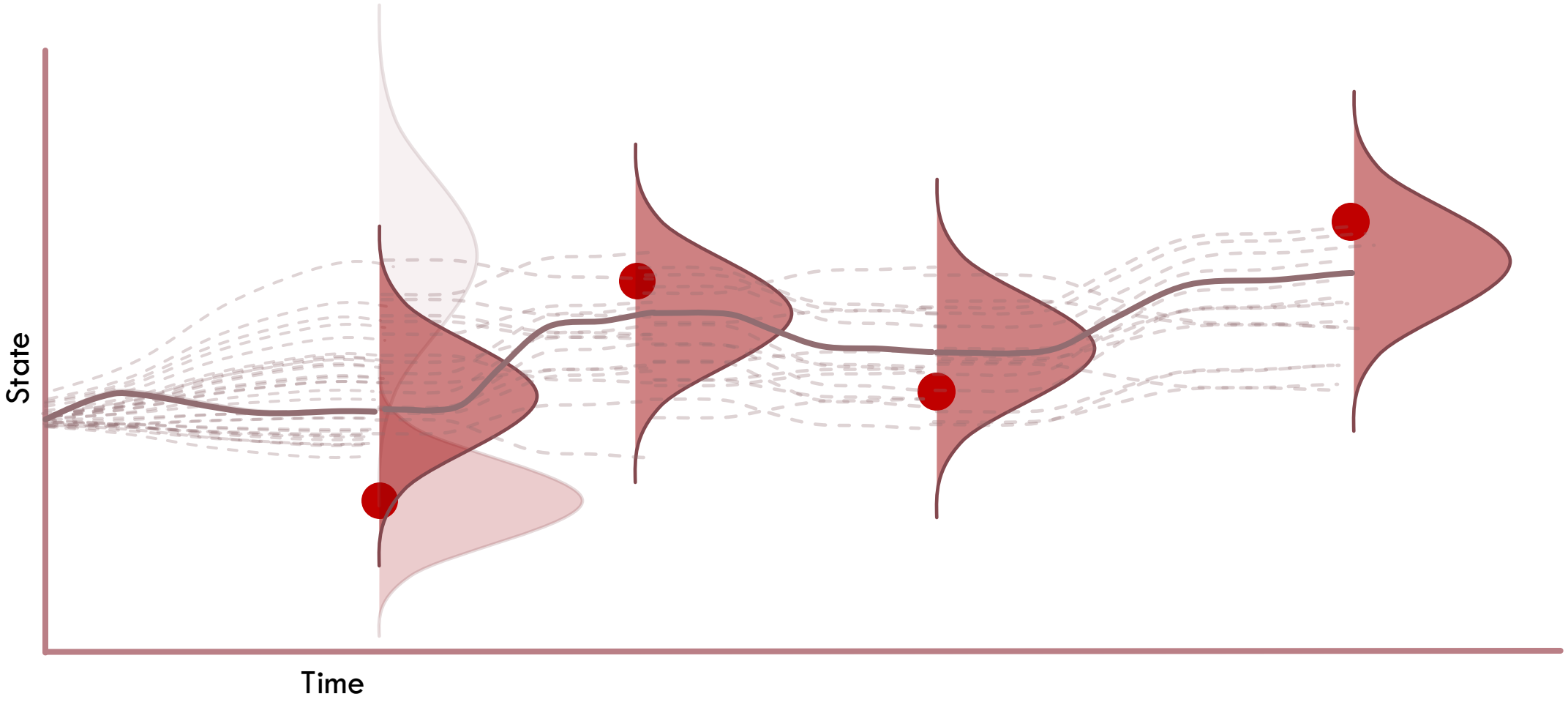
2. Update weights

$$\tilde{w}_{n+1}^i = w(u_n)p(y_{n+1}|u_{n+1}) = w(u_n) \cdot \text{likelihood}$$

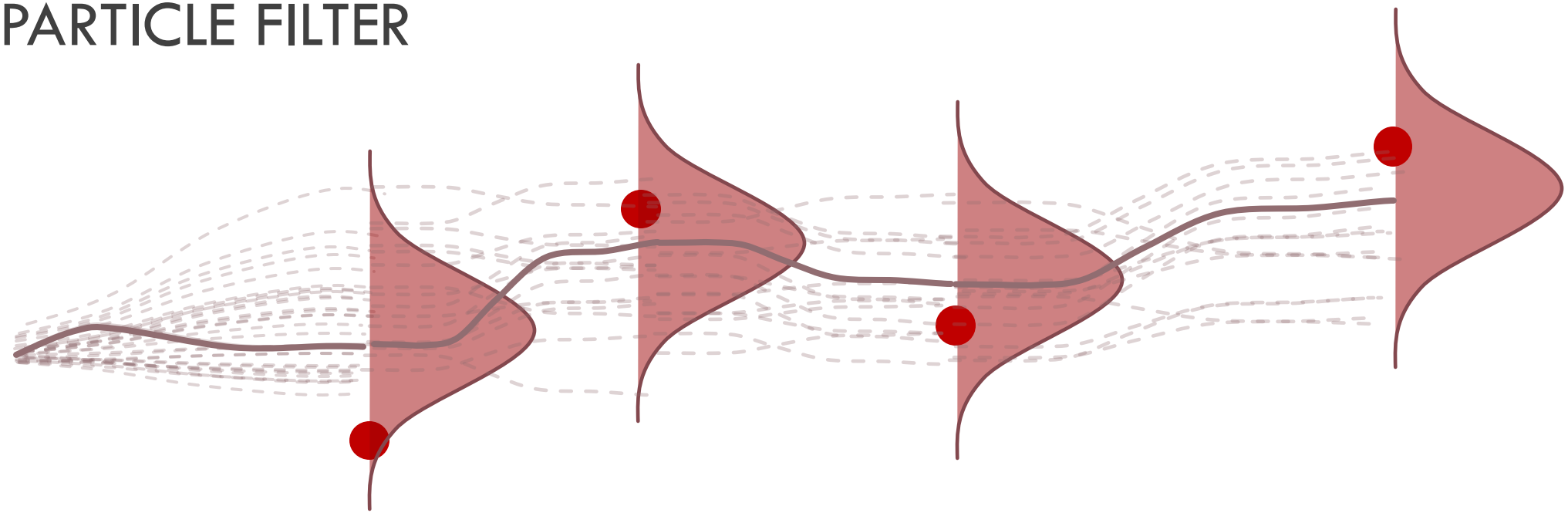
3. Resample according to weights

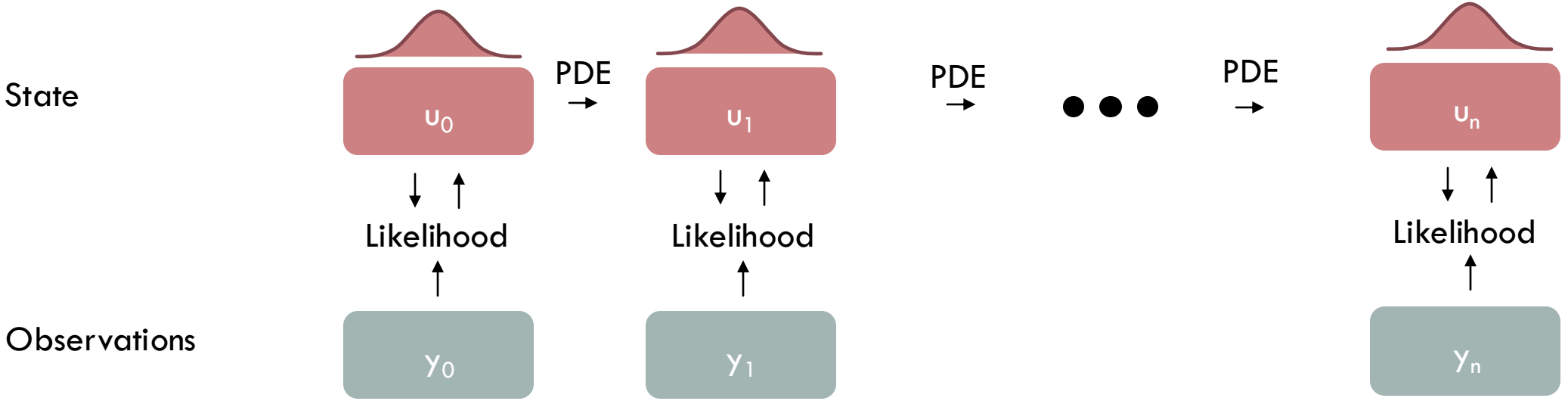
$$p(u_{n+1}|y_{0:n+1}) = \sum_{i=1}^{N_p} w_{n+1}^i \delta\left(u_{n+1} - u_{n+1}^i\right)$$

PARTICLE FILTER

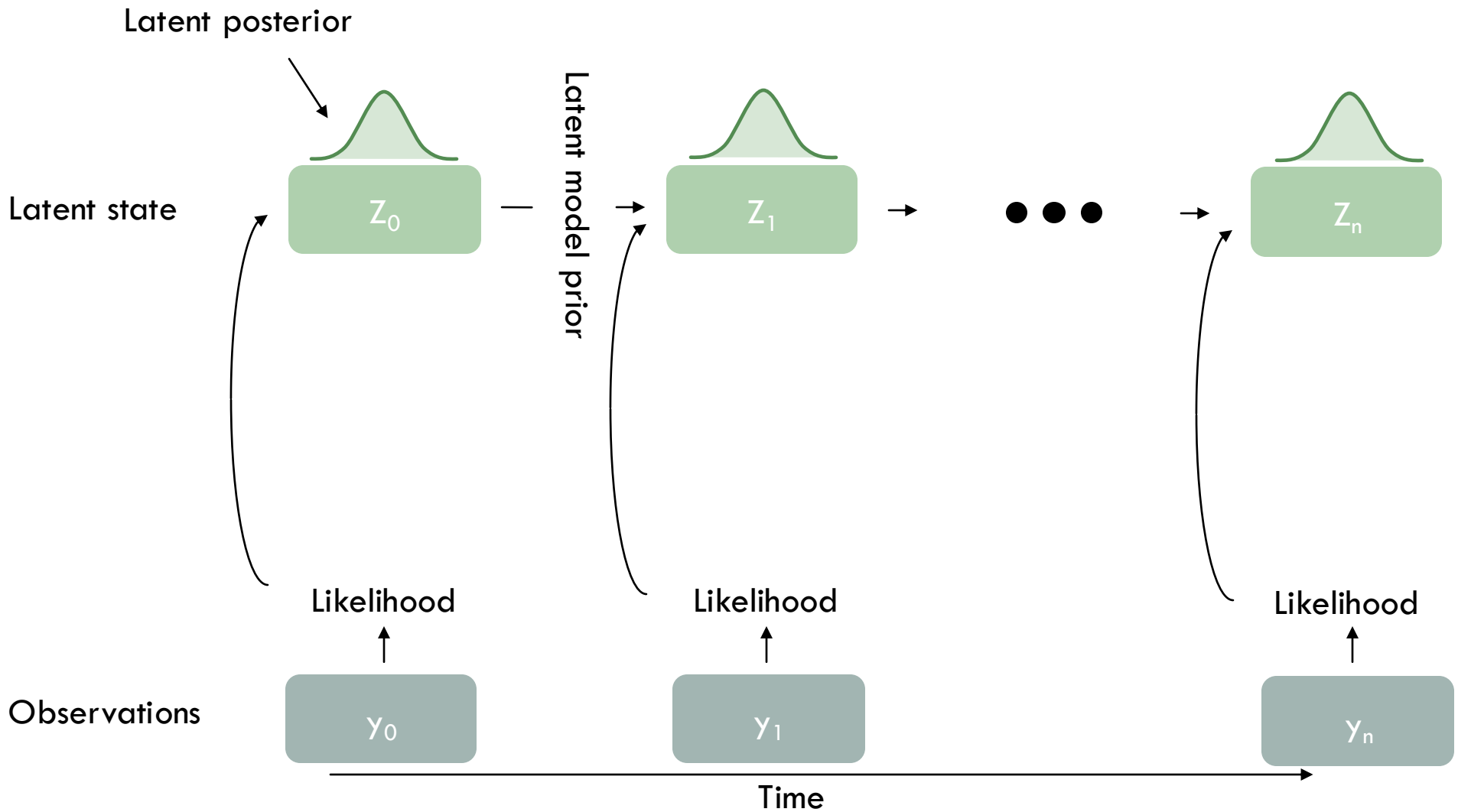


PARTICLE FILTER





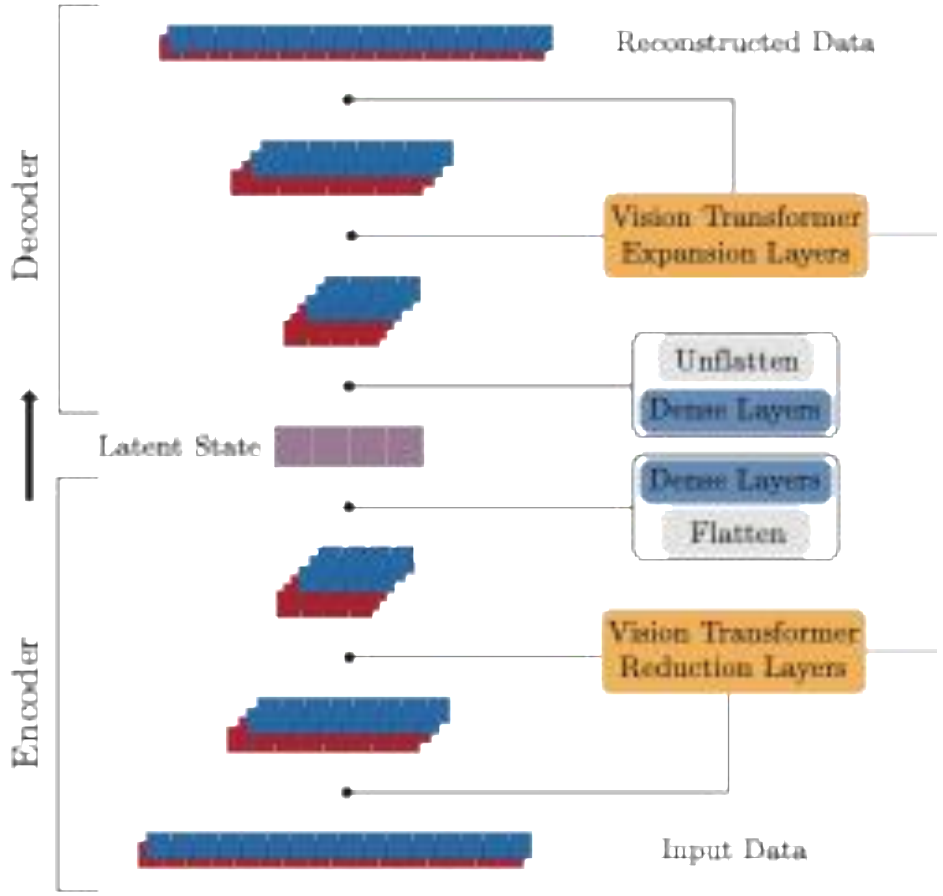
DATA ASSIMILATION IN LATENT SPACE



LATENT SPACE MODEL

AUTOENCODER

- Nonlinear dimensionality reduction
- Encoder – Reduce dimension
- Decoder – Renconstruct data



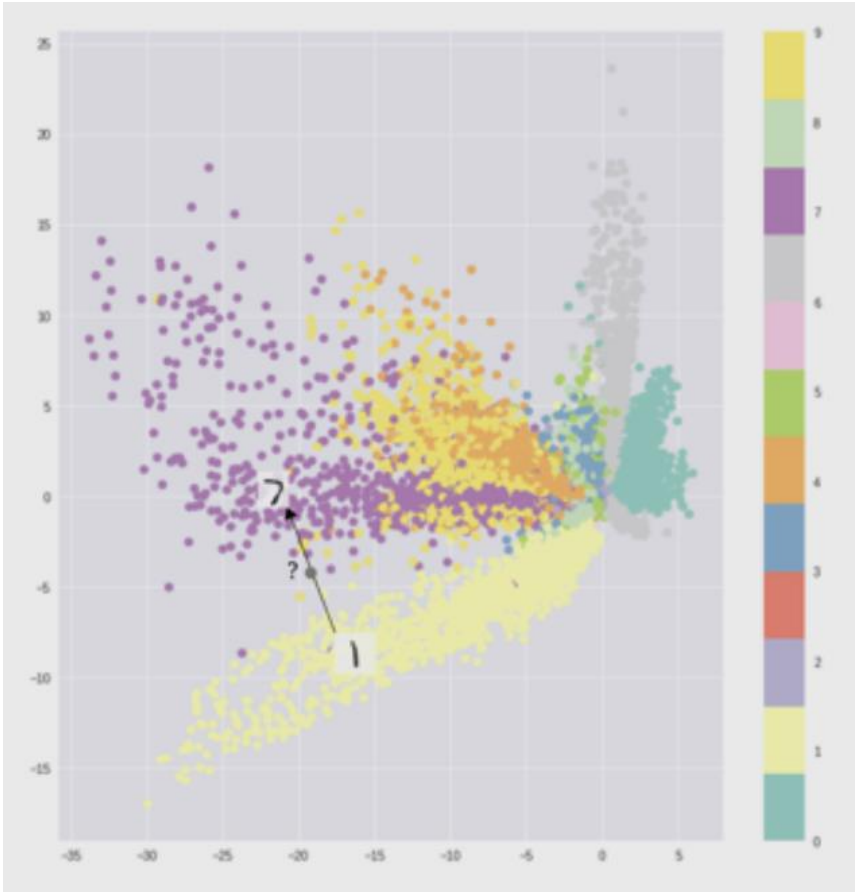
$$L_{\mathrm{AE}}(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}}) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i - \phi_{\mathrm{dec}}(\phi_{\mathrm{enc}}(\boldsymbol{x}_i))\right)^2}_{\text{reconstruction}} + \alpha \underbrace{R(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}})}_{\text{regularization}}.$$

https://www.jeremyjordan.me/variational-autoencoders/

AUTOENCODER

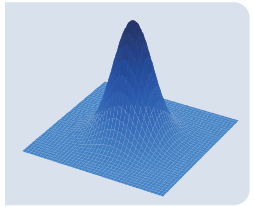
- Nonlinear dimensionality reduction
- Encoder – Reduce dimension
- Decoder – Renconstruct data

2D Latent space

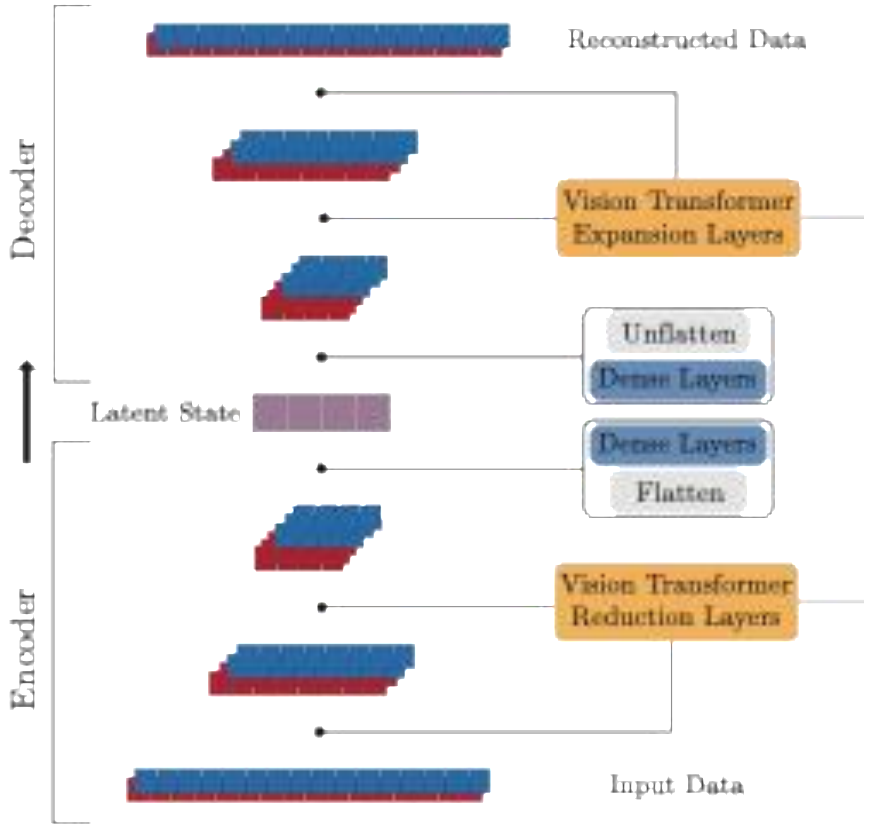


$$L_{\mathrm{AE}}(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}}) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i - \phi_{\mathrm{dec}}(\phi_{\mathrm{enc}}(\boldsymbol{x}_i))\right)^2}_{\text{reconstruction}} + \alpha \underbrace{R(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}})}_{\text{regularization}}.$$

WASSERSTEIN AUTOENCODER



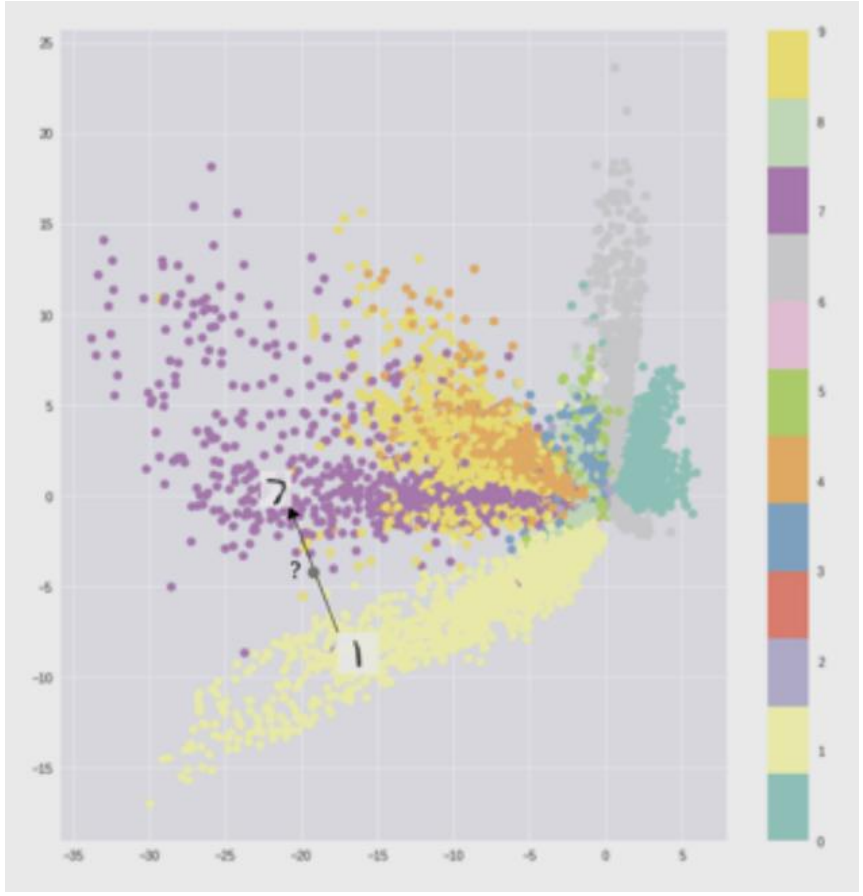
Maximum Mean Discrepancy
Regularization



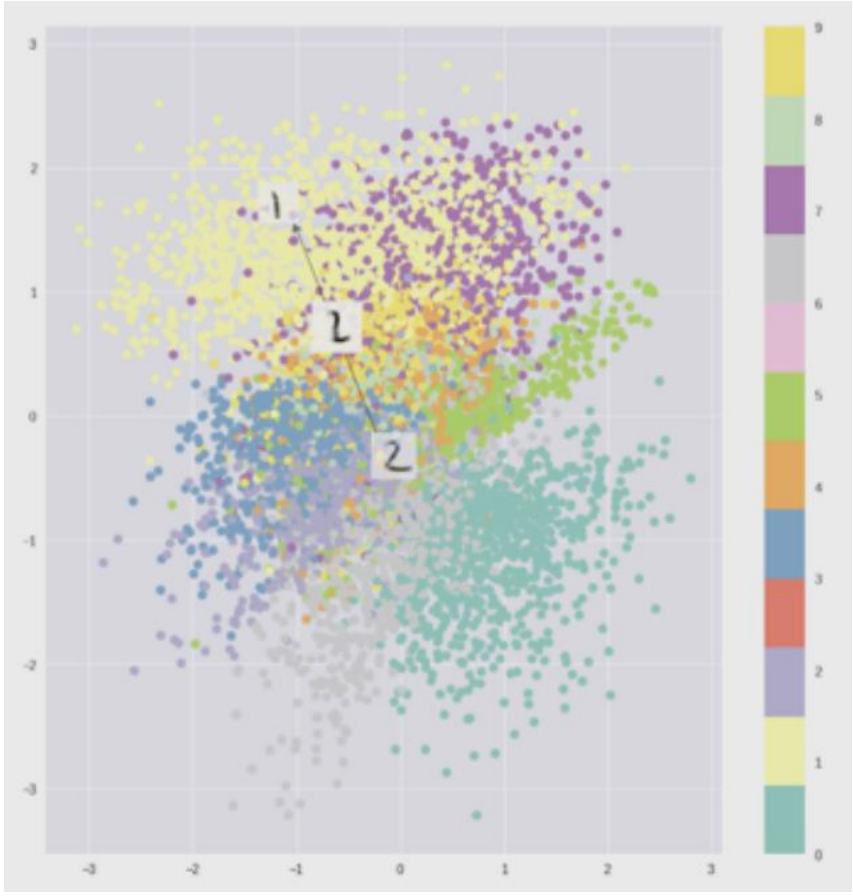
$$L_{\mathrm{WAE}}(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}}) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}_i - \phi_{\mathrm{dec}}(\phi_{\mathrm{enc}}(\boldsymbol{x}_i))\right)^2}_{\text{reconstruction}} + \lambda \underbrace{D(p(\boldsymbol{z}), p_{\mathrm{enc}}(\boldsymbol{z}))}_{\text{divergence}} + \alpha \underbrace{R(\phi_{\mathrm{enc}}, \phi_{\mathrm{dec}})}_{\text{regularization}}.$$

COMPARISON OF LATENT SPACE

No latent space divergence loss



With latent space divergence loss



LEARNING TO TIME-STEP

- Encode high-fidelity training data

$$\{\boldsymbol{q}_0, \ldots, \boldsymbol{q}_{N_t}\} \rightarrow \{\mathrm{Enc}(\boldsymbol{q}_0), \ldots, \mathrm{Enc}(\boldsymbol{q}_{N_t})\} = \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{N_t}\}$$

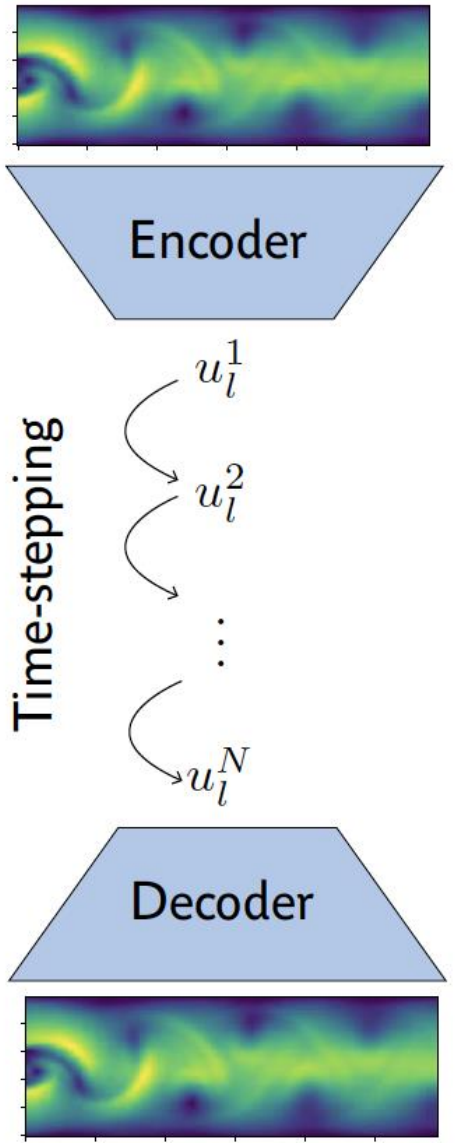
- Minimize time-stepping error

Time-stepping NN

$$\sum_{i=1}^{N_t} ||\boldsymbol{z}_i - f(\boldsymbol{z}_{i-1}, \theta)|$$

- Loop unrolling

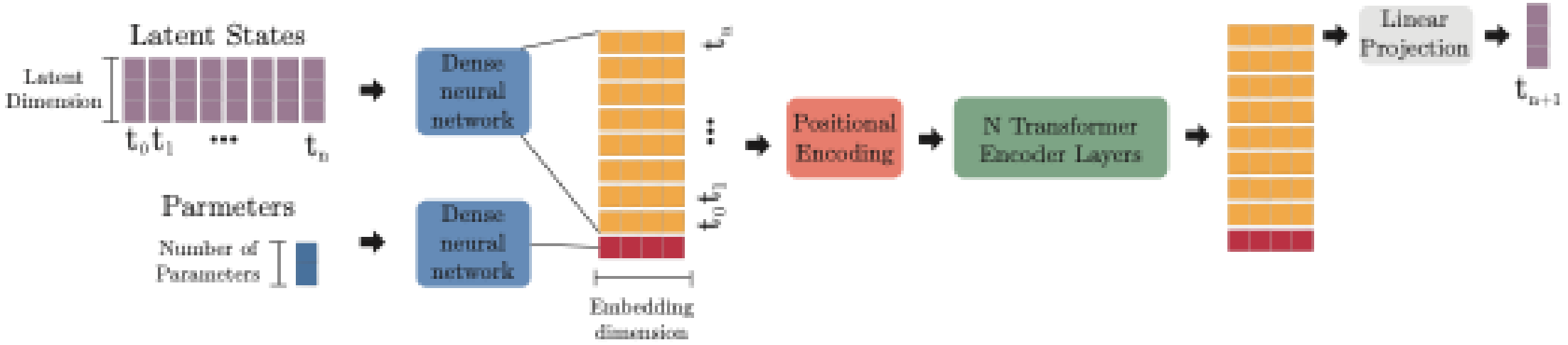
$$\sum_{i=1}^{N_t-k} \sum_{k=0}^{K} ||\boldsymbol{z}_{i+k} - f^k(\boldsymbol{z}_{i-1}, \theta)|$$



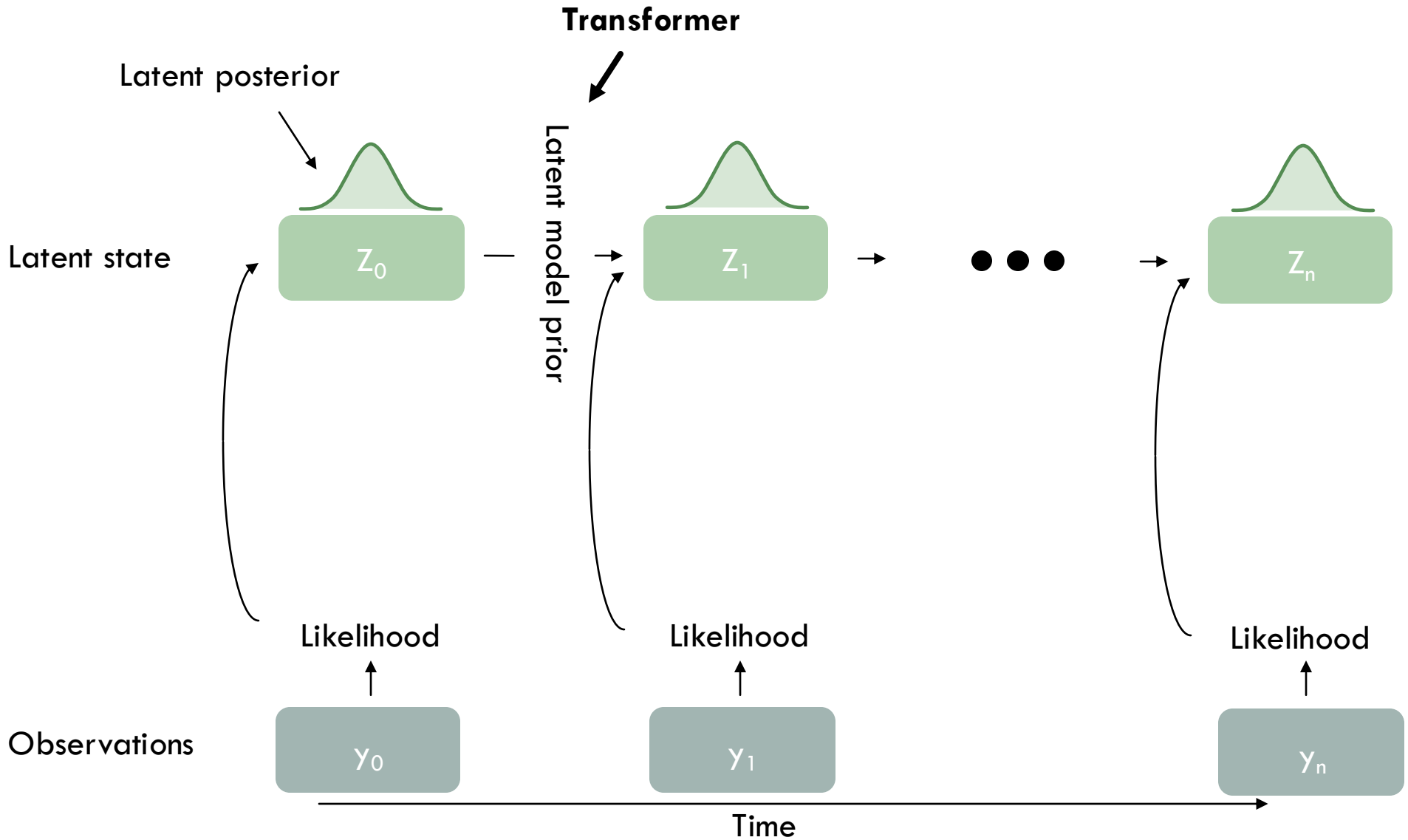
TRANSFORMERS

- They show **state-of-the-art** performance for **time series forecasting**
  - Outperforming RNNs, LSTMs, CCNNs, etc...
- Transformers ”mimic” **multistep methods**
  - “The function space of a self-attention layer with a residual connection contains the space of explicit linear time-integration methods within an arbitrarily small non-zero amount of error.”
  - *Geneva, Nicholas, and Nicholas Zabaras. "Transformers for modeling physical systems." Neural Networks 146 (2022): 272-289.*
- Great at incorporating **parametric dependence through attention**
  - *Han, Xu, et al. "Predicting physics in mesh-reduced space with temporal attention." arXiv preprint arXiv:2201.09113 (2022).*

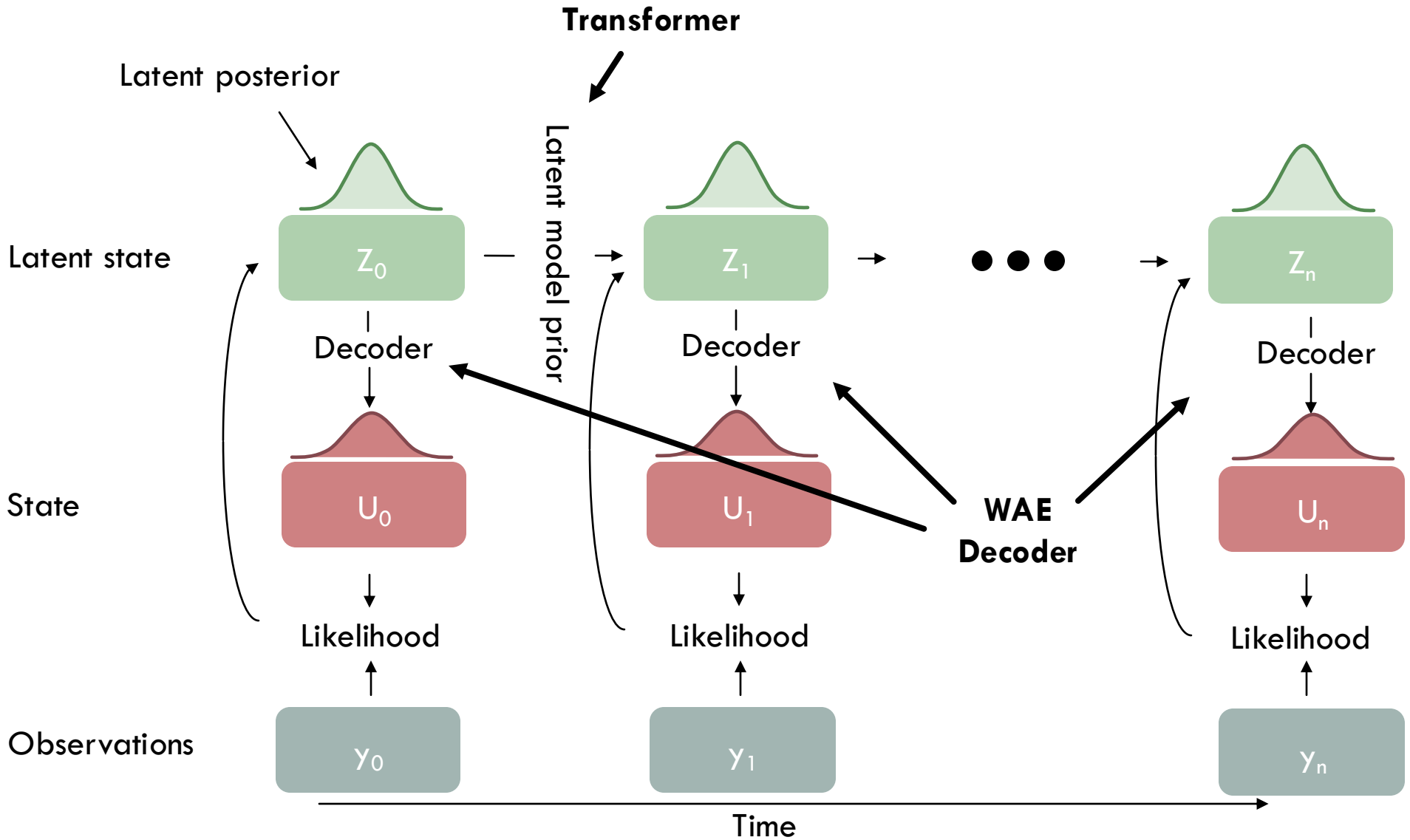
PARAMETERIZED TIME-STEPPING

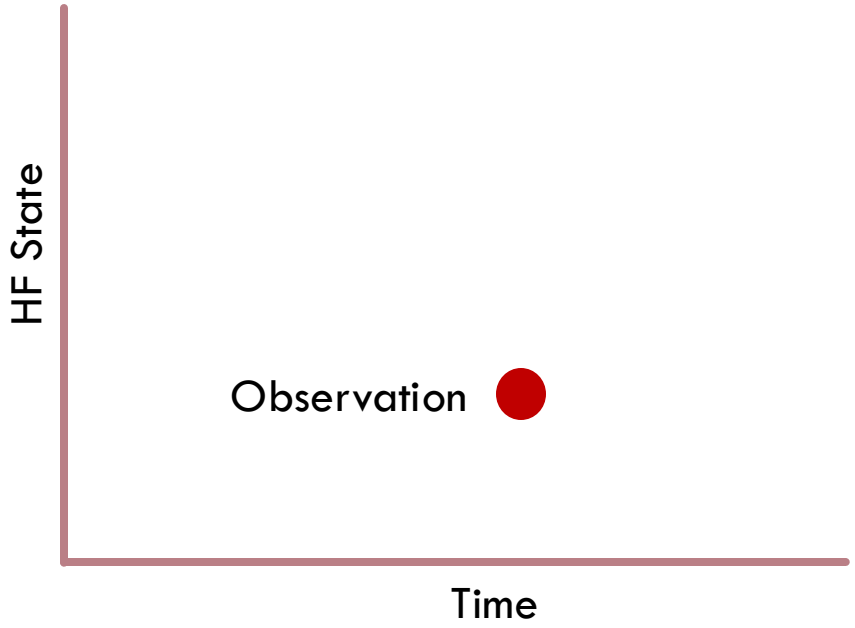


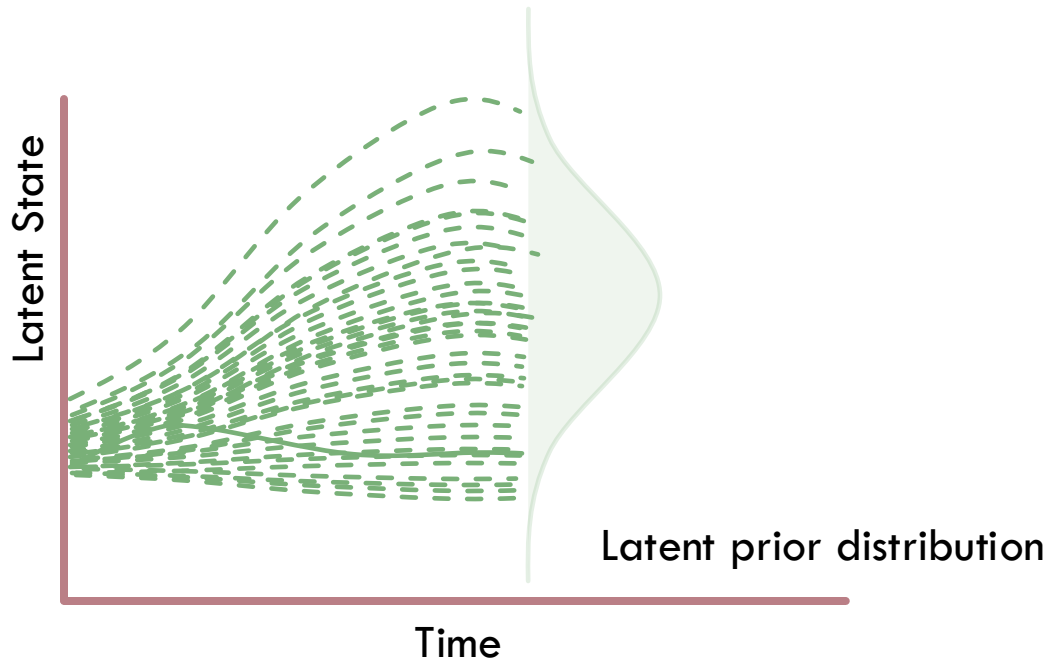
DATA ASSIMILATION IN LATENT SPACE



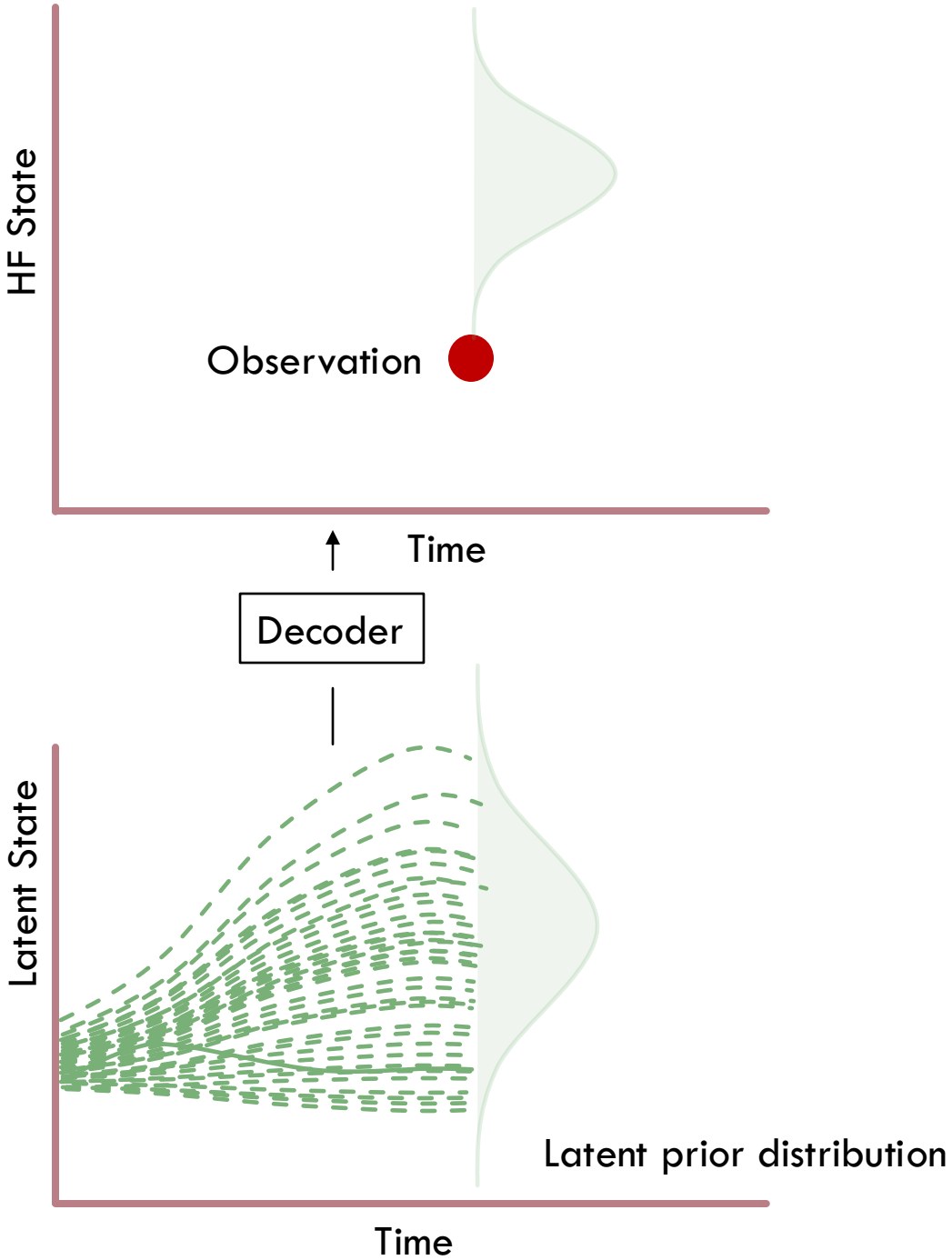
DATA ASSIMILATION IN LATENT SPACE



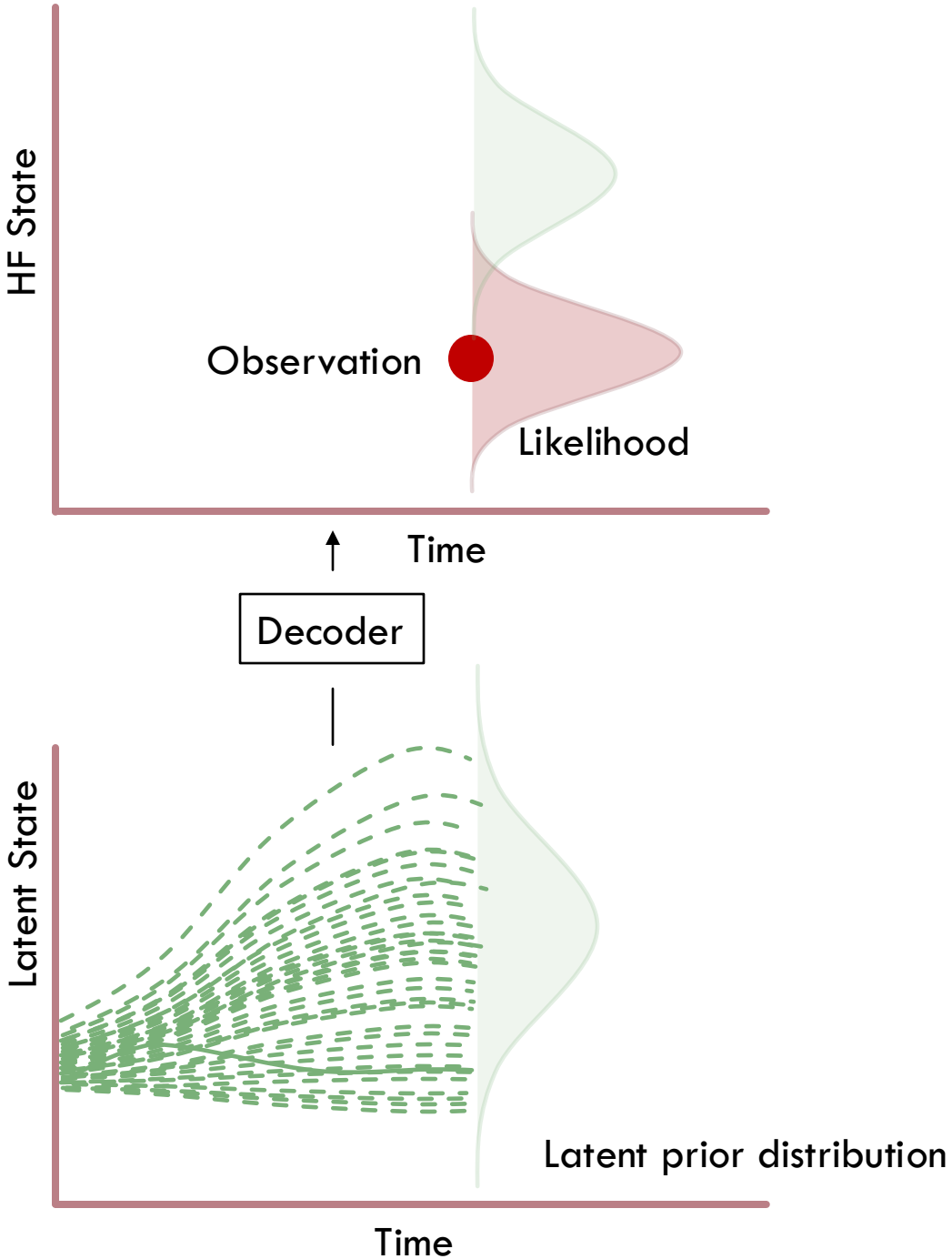




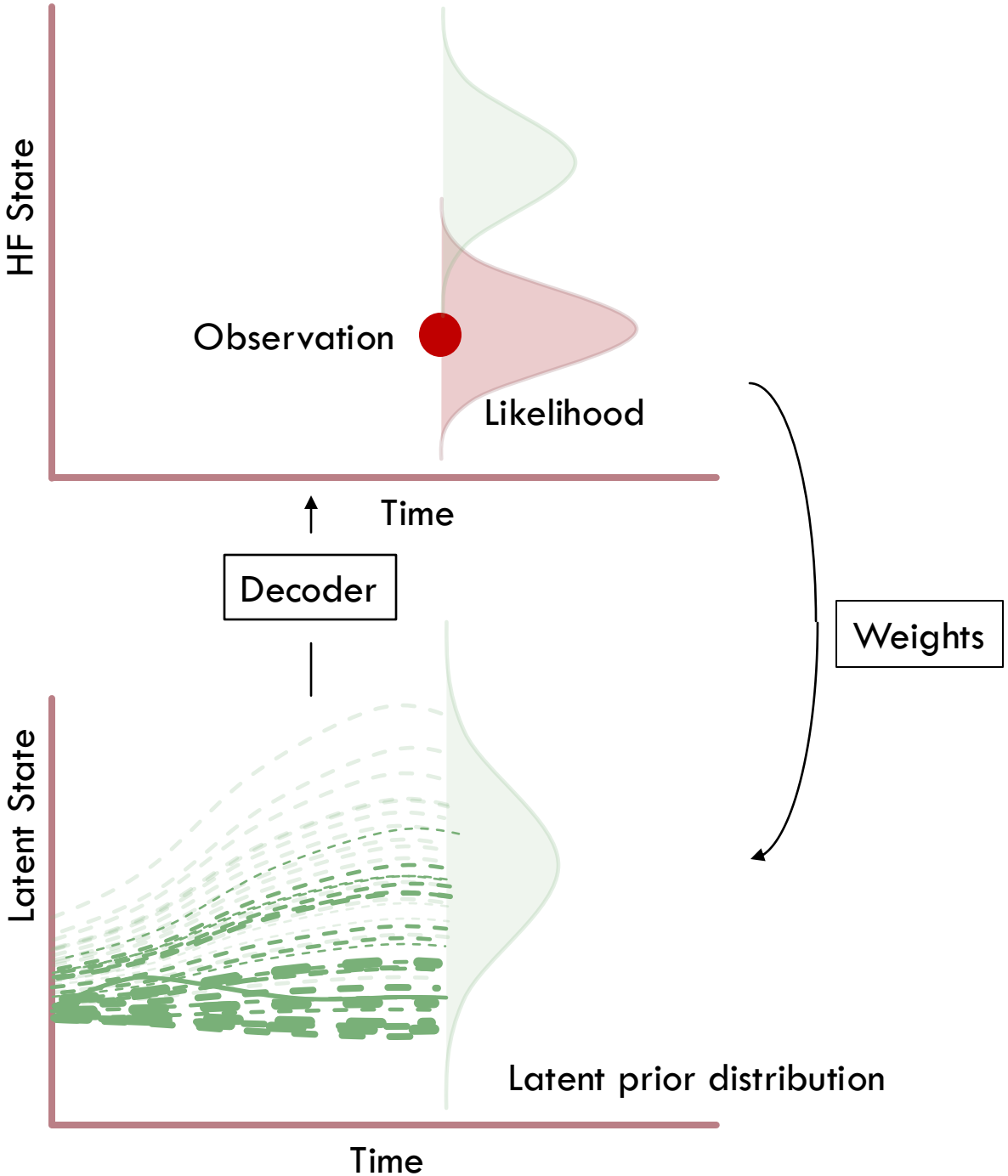
1. Compute latent prior



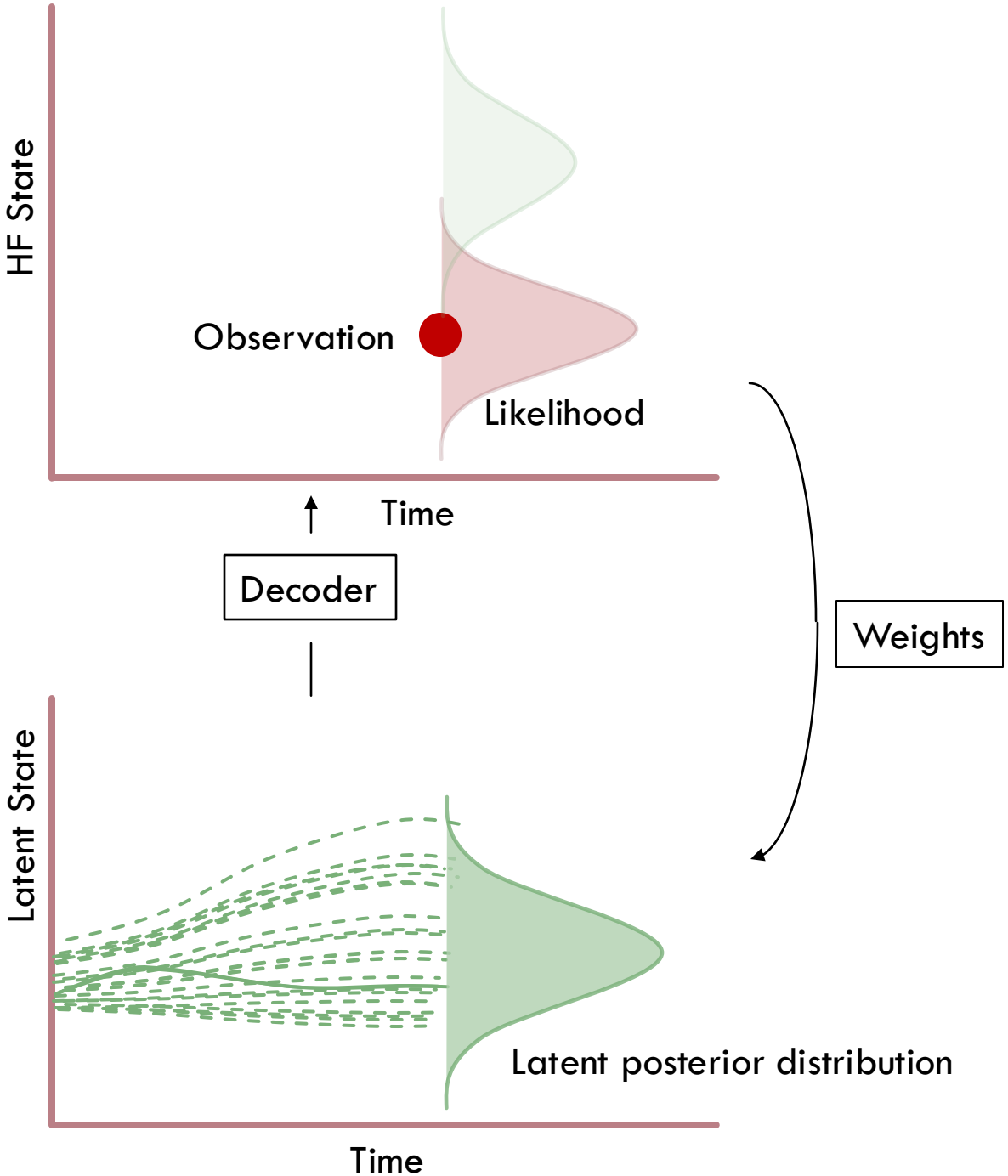
1. Compute latent prior
2. Decode latent samples



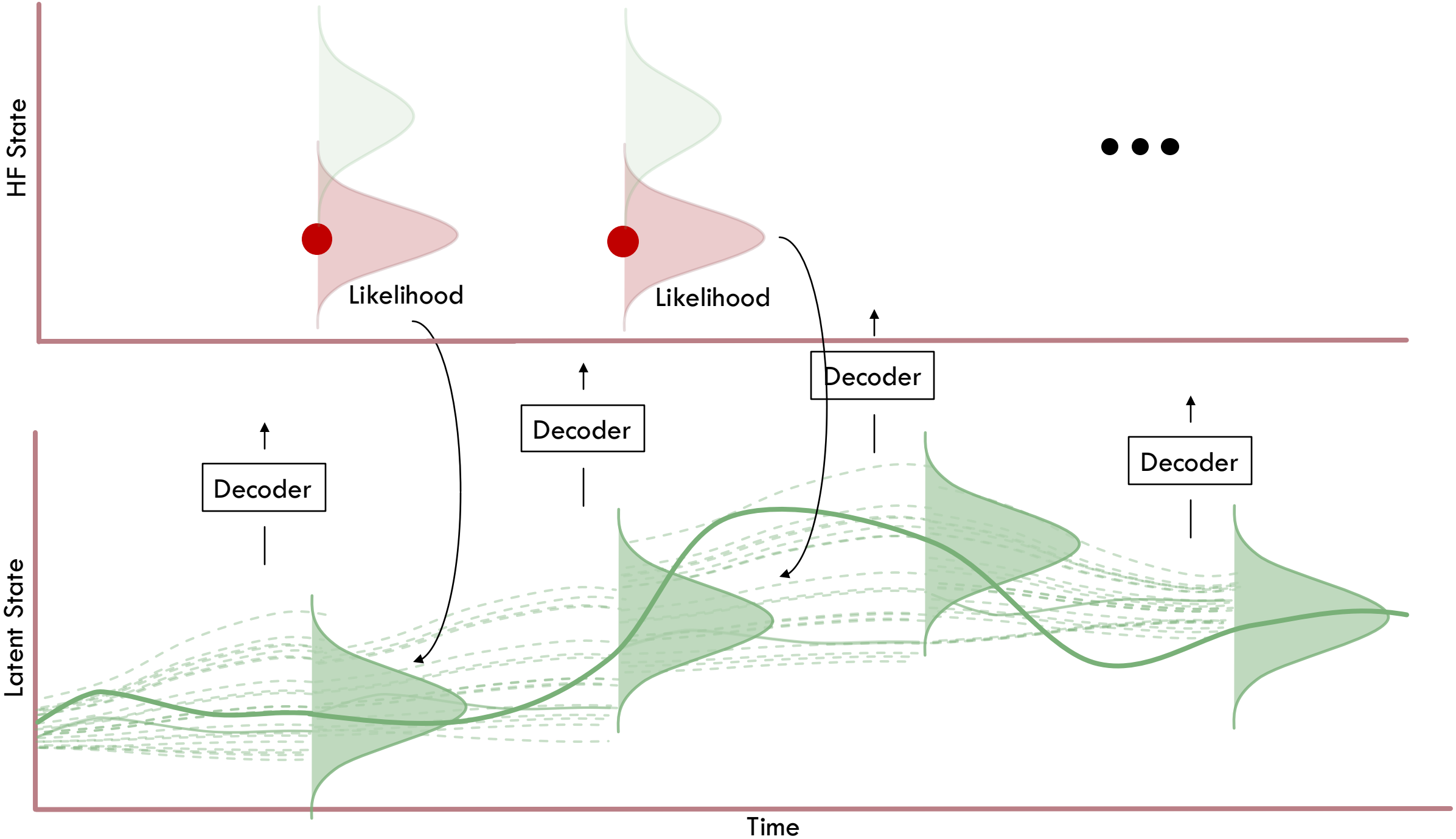
1. Compute latent prior
2. Decode latent samples
3. Compute likelihood



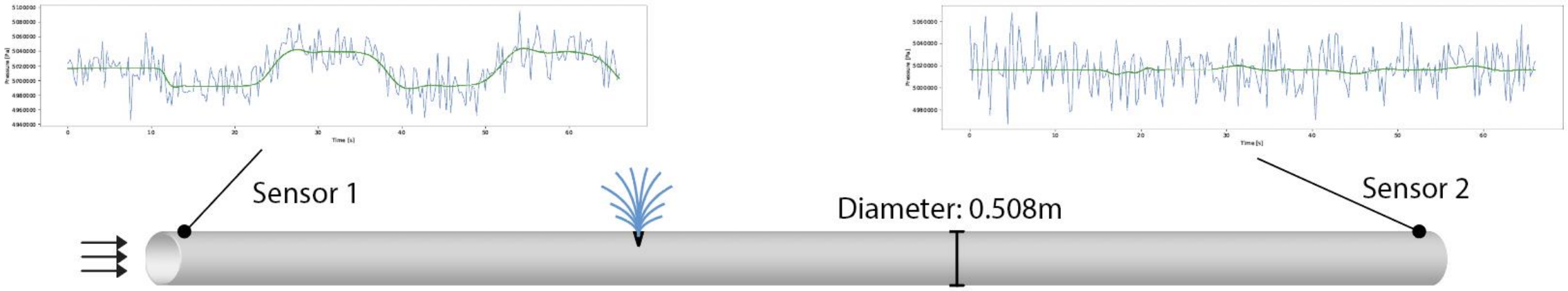
1. Compute latent prior
2. Decode latent samples
3. Compute likelihood
4. Compute weights



1. Compute latent prior
2. Decode latent samples
3. Compute likelihood
4. Compute weights
5. Resample to get latent posterior


HF State
Likelihood
Likelihood
Decoder
Decoder
Decoder
Decoder
Latent State
Time

LEAK LOCALIZATION



PROBLEM SETTING

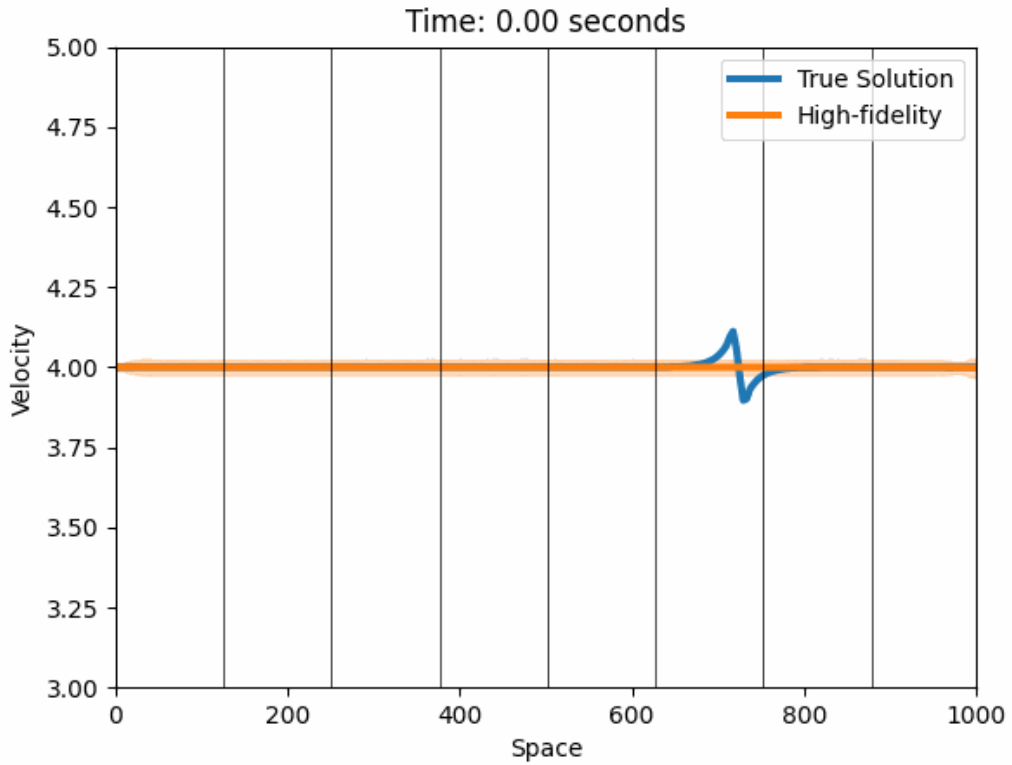
Leak size

Leak location

$$\partial_t q_1 + \partial_x q_2 = C_d \sqrt{\rho(p(\rho) - p_{\text{amb}})}\,\delta(x - x_l) H(t - t_l)$$

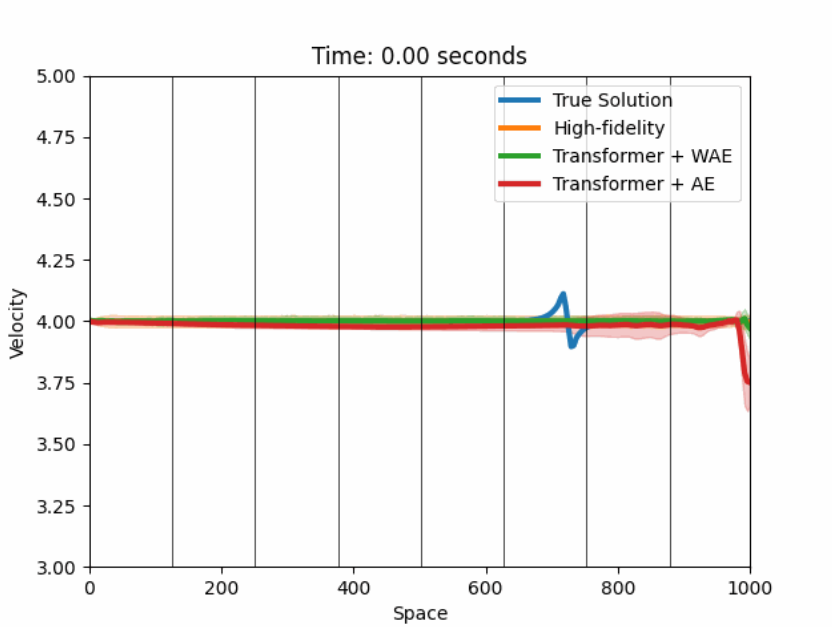
$$\partial_t q_2 + \partial_x \left(\frac{q_2^2}{q_1} + p(\rho) A \right) = -\frac{1}{2d} \frac{q_2^2}{q_1} f_f(q)$$

- Goal:
  - State estimation
  - Leak localization
- Observations:
  - Only velocity
  - 2 seconds (40 time steps)
  - 125m
- High-fidelity solution
  - 10000 particles

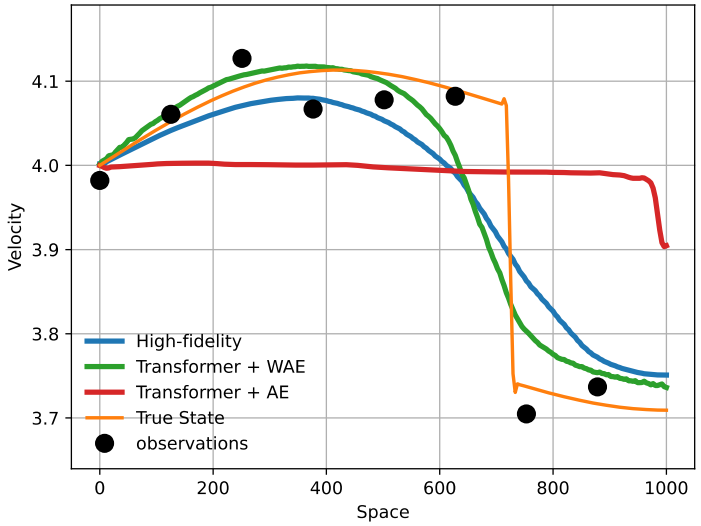


STATE ESTIMATION

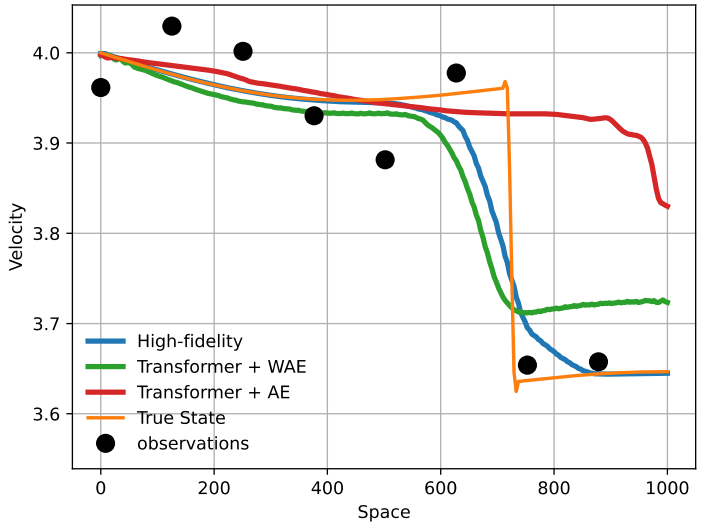
- Latent dimension = 16
- WAE and AE architectures are identical



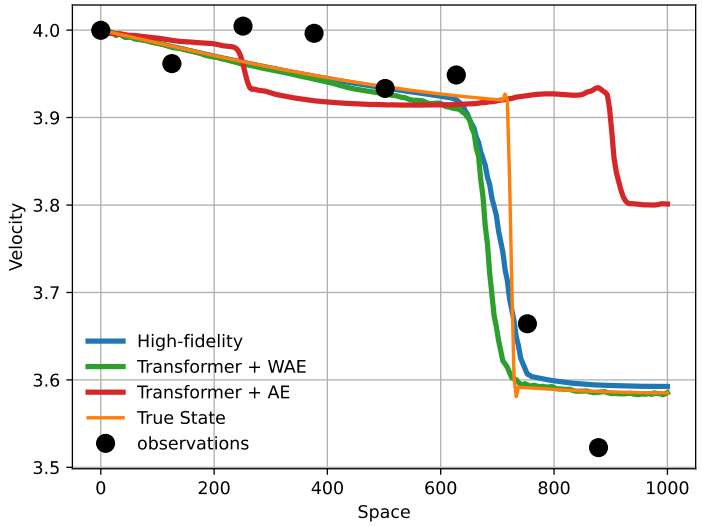
Time: 15 [s]



Time: 30 [s]

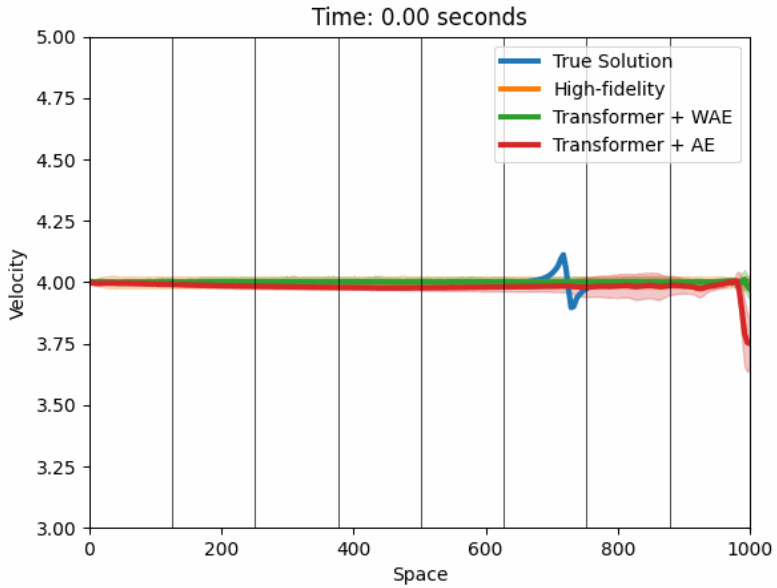


Time: 70 [s]

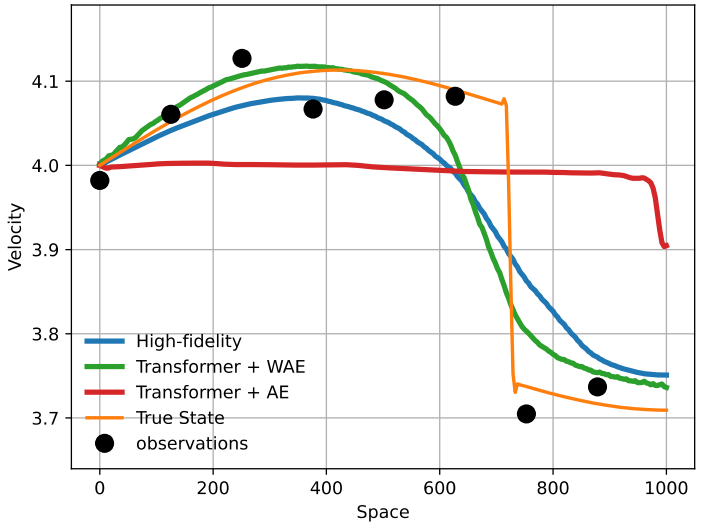


COMPUTATION TIME

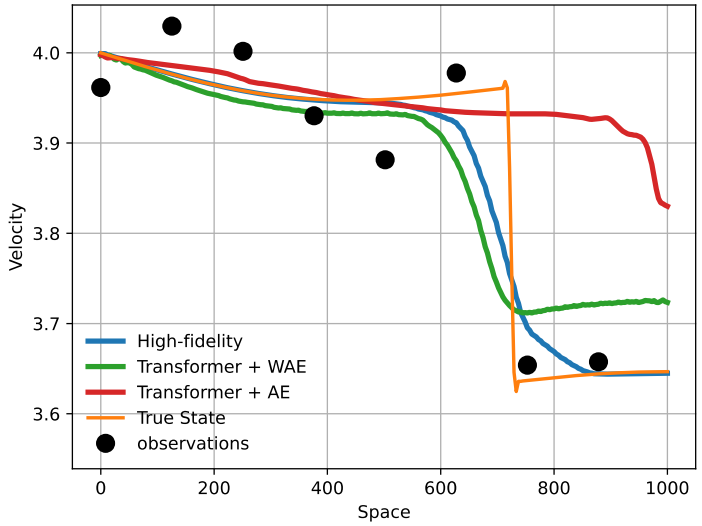
- High-fidelity(30 CPU cores): 18 hours = **64,800** seconds
- Latent:
  - 1 CPU core: **227** seconds
  - 1 GPU: **85** seconds



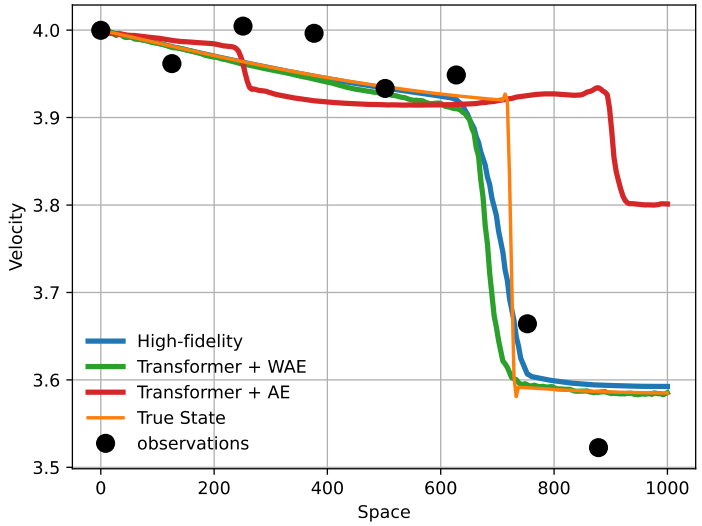
Time: 15 [s]



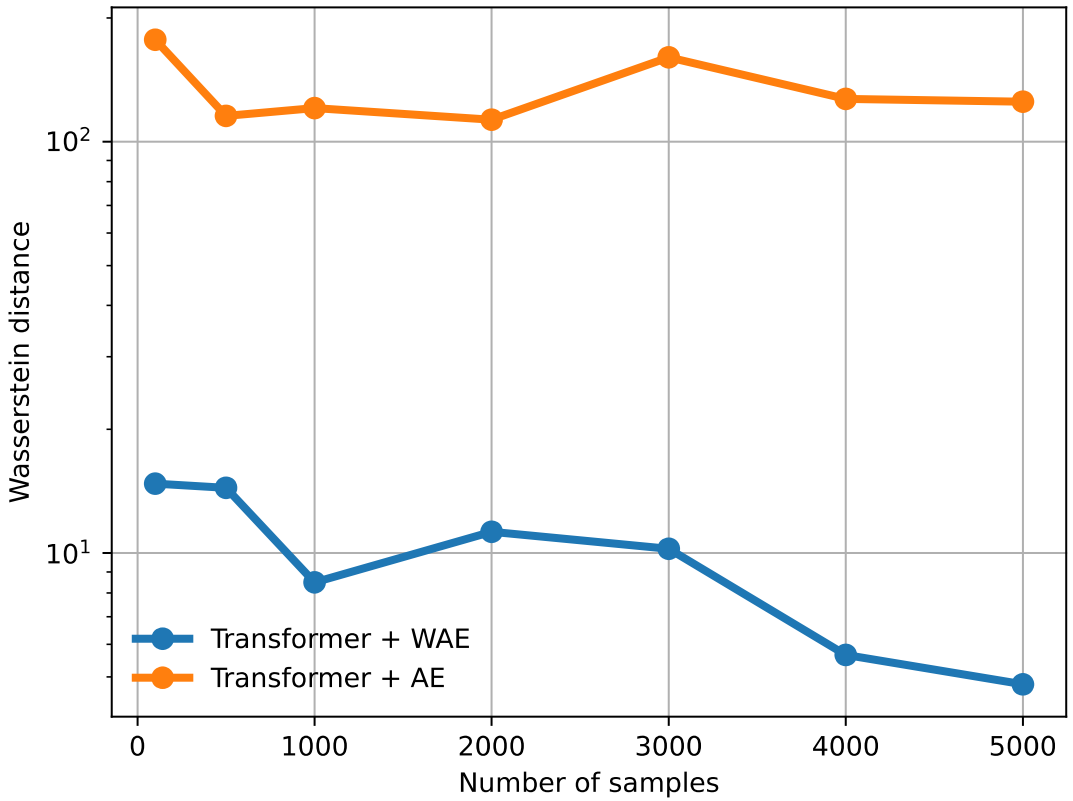
Time: 30 [s]

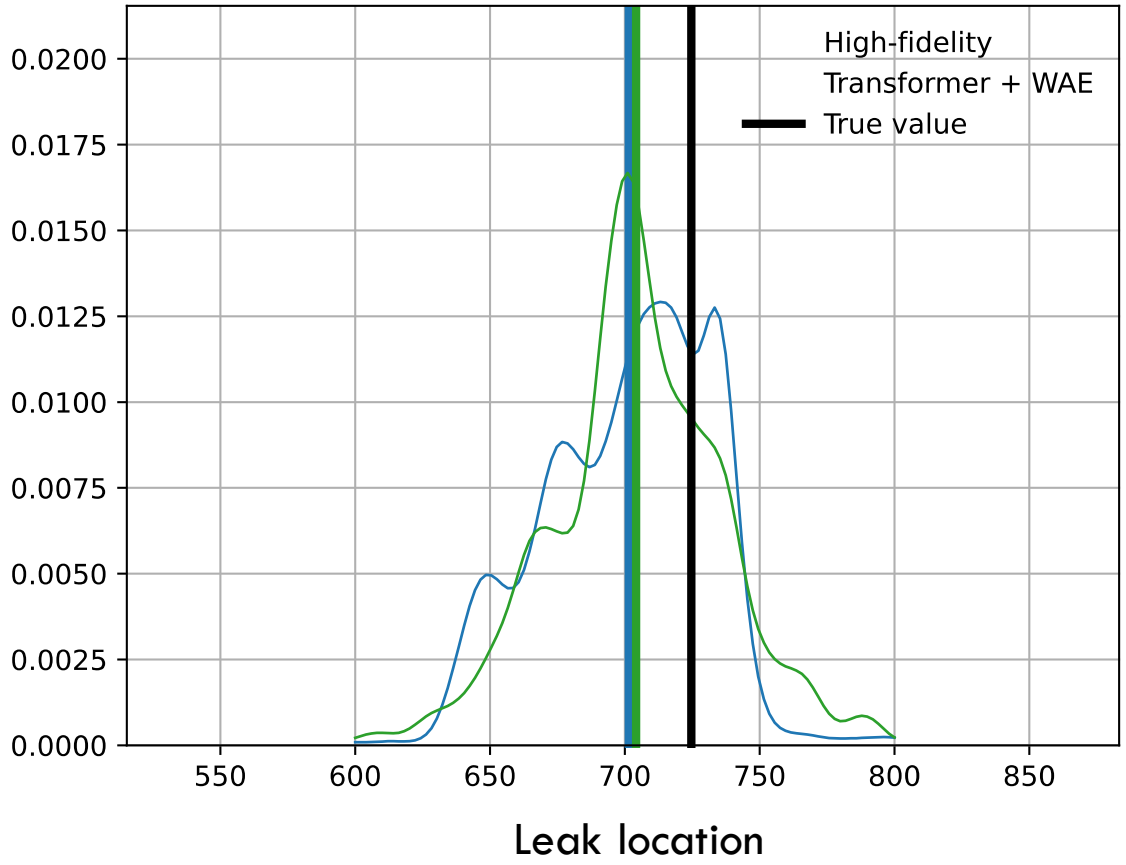


Time: 70 [s]

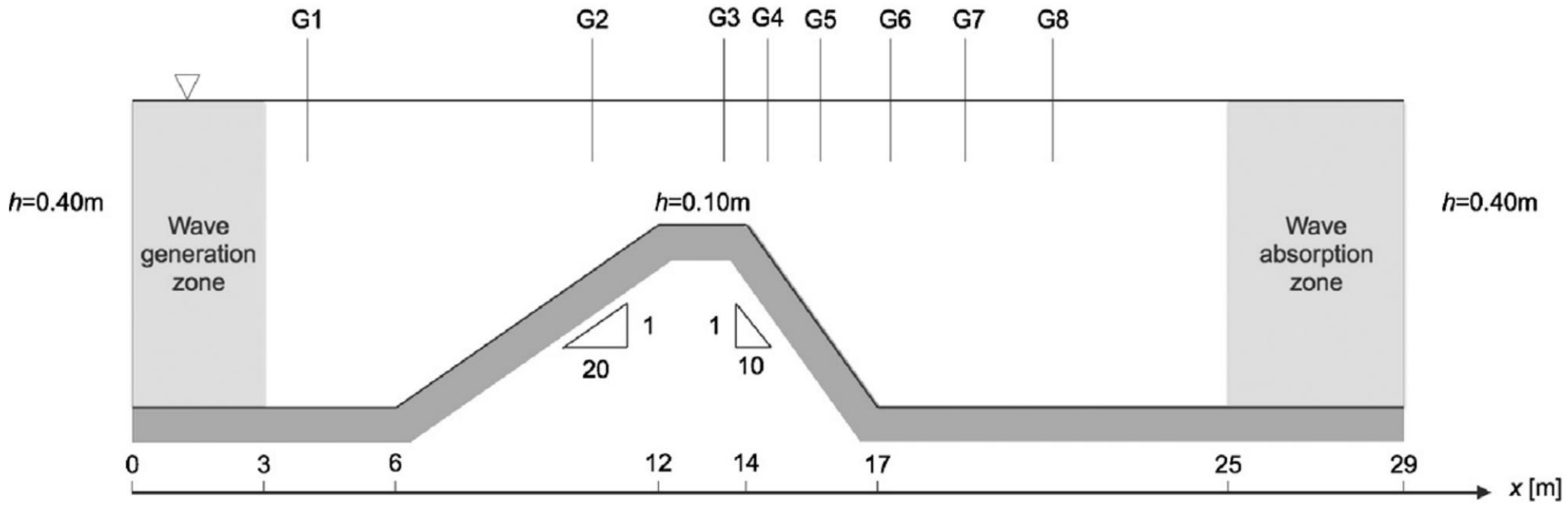


LEAK LOCALIZATION





HARMONIC WAVE GENERATION OVER A SUBMERGED BAR



Engsig-Karup, Allan Peter, Claes Eskilsson, and Daniele Bigoni. "A stabilised nodal spectral element method for fully nonlinear water waves." *Journal of Computational Physics* 318 (2016): 1-21.

Beji, S., and J. A. Battjes. "Experimental investigation of wave propagation over a bar." *Coastal engineering* 19.1-2 (1993): 151-162.

HARMONIC WAVE GENERATION OVER A SUBMERGED BAR

- Fully nonlinear water wave model
- Time horizon: 42s
- Parameter: Bar height
- Surrogate model state:
  - Surface elevation
  - Surface velocity potential

Surface equations

$$\frac{\partial \eta}{\partial t} = -\boldsymbol{\nabla}\eta \cdot \boldsymbol{\nabla}\tilde{\phi} + \tilde{w}(1 + \boldsymbol{\nabla}\eta \cdot \boldsymbol{\nabla}\eta),$$
$$\frac{\partial \tilde{\phi}}{\partial t} = -g\eta - \frac{1}{2}\left(\boldsymbol{\nabla}\tilde{\phi} \cdot \boldsymbol{\nabla}\tilde{\phi} - \tilde{w}^2(1 + \boldsymbol{\nabla}\eta \cdot \boldsymbol{\nabla}\eta)\right).$$

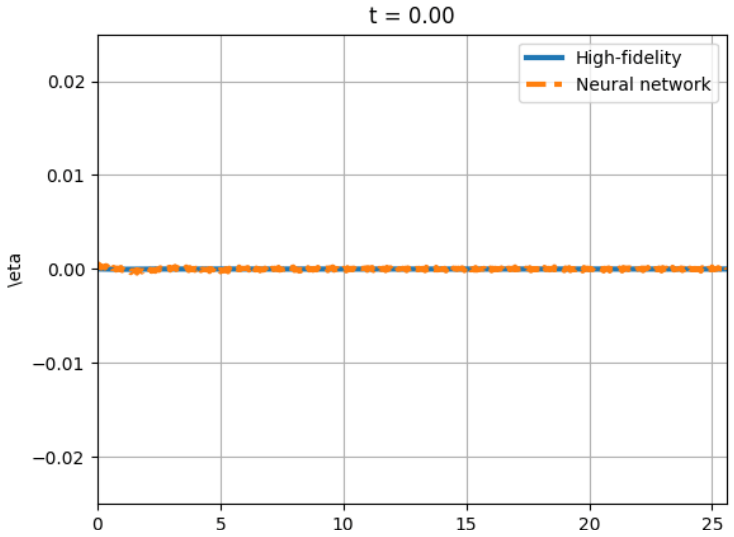
Velocity potential in full domain

$$\nabla^\sigma(K(x;t)\nabla^\sigma\phi) = 0, \quad \text{in} \quad \Omega^c,$$
$$\phi = \tilde{\phi}, \quad z = \eta \quad \text{on} \quad \Gamma^{FS}$$
$$\mathbf{n} \cdot \nabla\phi = 0, \quad z = -h(x,y) \quad \text{on} \quad \Gamma^b.$$

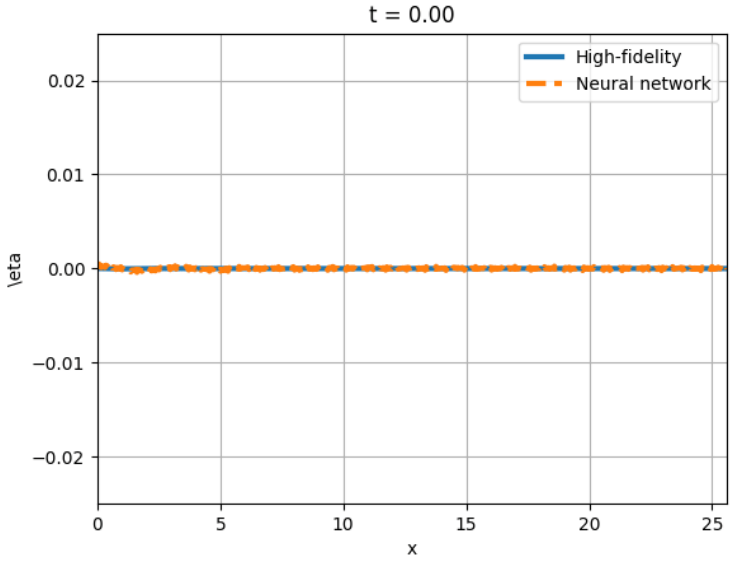
HARMONIC WAVE GENERATION OVER A SUBMERGED BAR

- Fully nonlinear water wave model
- Time horizon: 42s
- Parameter: Bar height
- Surrogate model state:
  - Surface elevation
  - Surface velocity potential

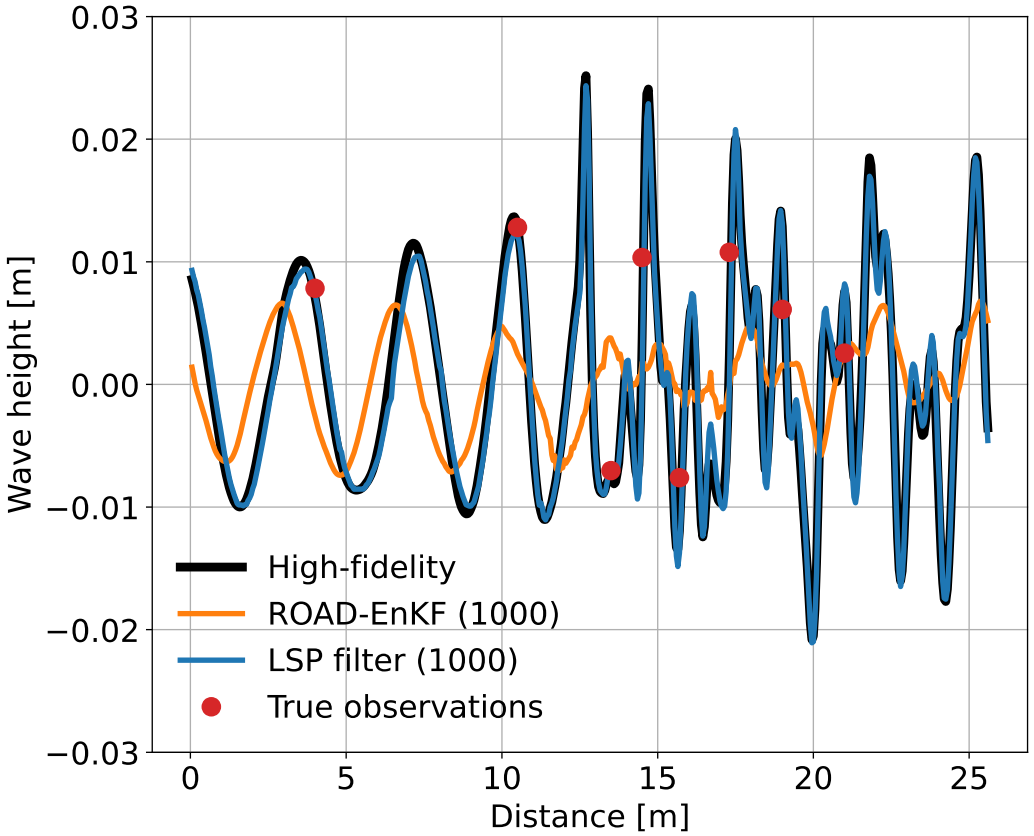
Tall bar

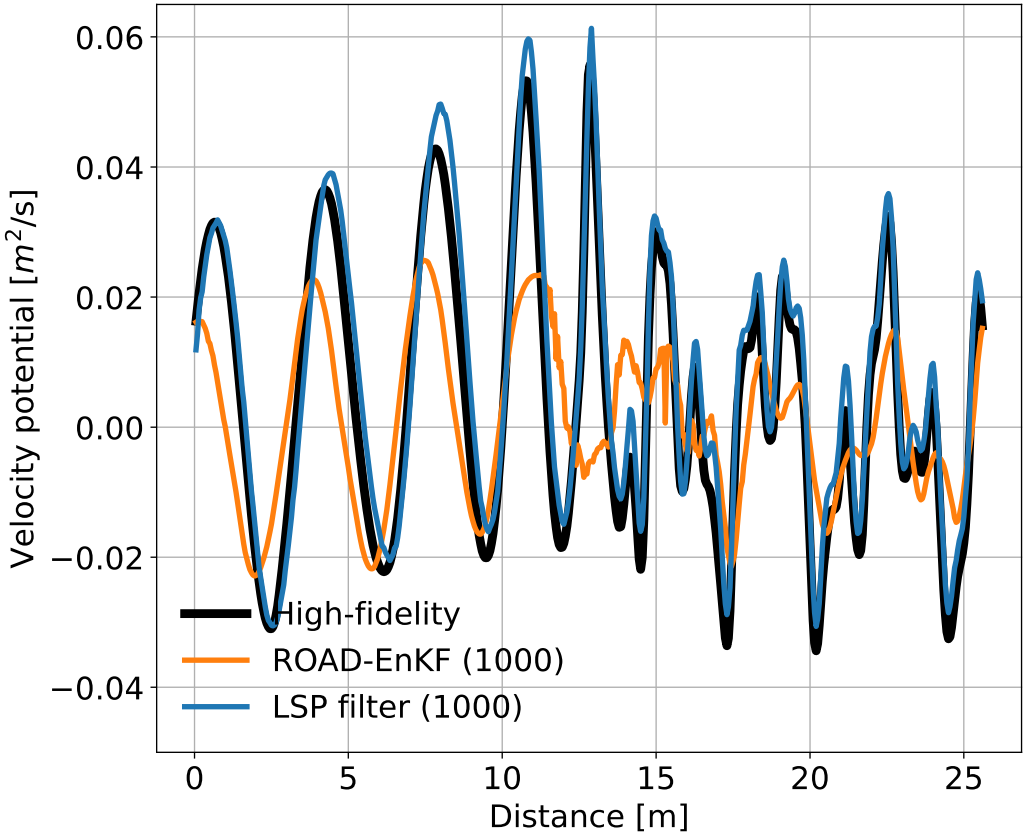


Low bar

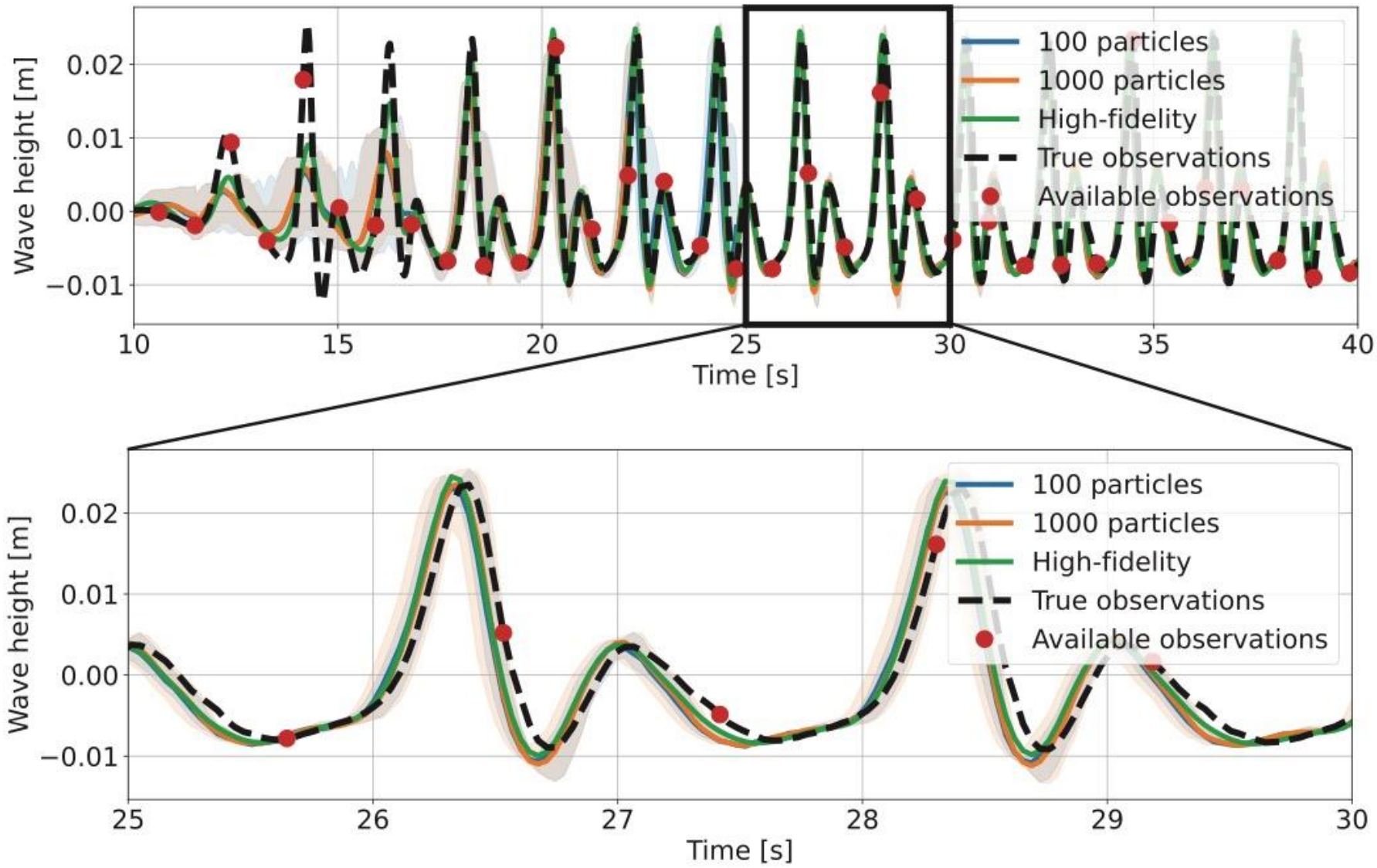


STATE ESTIMATION



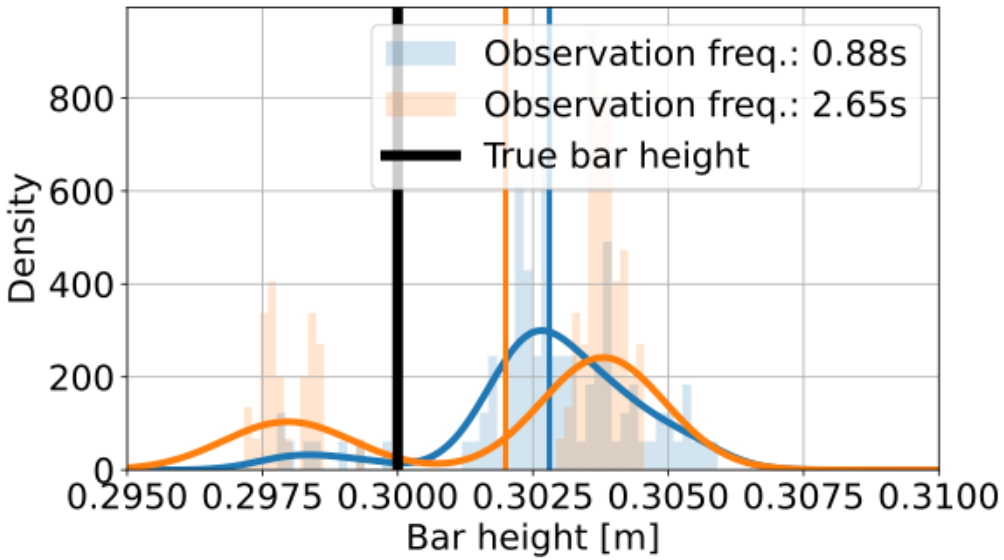


OBSERVATIONS AT SENSOR

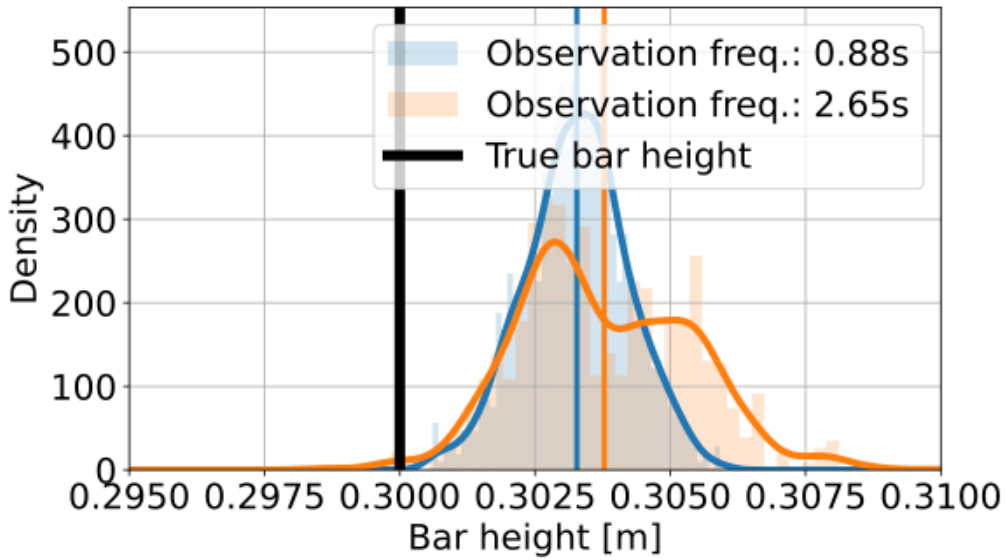


BAR HEIGHT DISTRIBUTION

100 particles



1000 particles



CONCLUSION

- Latent space sampling can speed up Bayesian inference
- Works for static and dynamic problems

WANT TO KNOW MORE?

nikolaj.mucke@cwi.nl

Mücke, N. T., Sanderse, B., Bohté, S. M., & Oosterlee, C. W. (2023). Markov chain generative adversarial neural networks for solving Bayesian inverse problems in physics applications. *Computers & Mathematics with Applications*, *147*, 278-299.

Mücke, N. T., Bohté, S. M., & Oosterlee, C. W. (2024). The deep latent space particle filter for real-time data assimilation with uncertainty quantification. *Scientific Reports*, *14*(1), 19447.