Formulating the digital twin problem as a two-step Bayesian Inference problem

Peter Jan van Leeuwen

Department of Atmospheric Science,

Colorado State University

Interests and work over the years

- Physics, astrophysics, physical chemistry, theoretical physics, coastal waves, physical oceanography, data assimilation, fully nonlinear causal discovery, information propagation in complex systems, cloud microphysics, cloud aggregation, air-sea interaction.

- Discovered a new ocean current, the South Indian Ocean Counter Current.

- Involved in Ensemble Kalman Filter and Smoother, (Local) Particle Filters, and Particle Flow Filters and Smoothers. From theory to very high-dimensional applications.

- All kinds of practical data-assimilation stuff: estimating model errors, observation errors, representation errors, and estimating missing physics.

- Some other stuff…

Digital Twins

The four ingredients of a digital twin, fed by observations from the real-world system and models and resources.
Arrows denote Information Flow (blue), Causal Flow (purple), and Workflow (red).

Notation

x_t	State vector, state of the system
y_t	Observations
α_t	Controls
c_t	Costs
z_t	Target variable (subset of x_t)

Formulation of the digital twin problem I

Our goal is to find the controls that lead to a target evolution of the system, given observations of the system. Since 'everything' is uncertain, our goal will be a pdf:

$$p(\alpha_{t:\tau}|y_{0:t}, z_{t:\tau}, c_{0:\tau}) = \int p(\alpha_{t:\tau}, x_{t:\tau}|y_t, z_{t:\tau}, c_{0:\tau})\, dx_{t:\tau}$$

Since this parameter estimation problem is highly nonlinear, we often reformulate it as finding

$$p(\alpha_{t:\tau}, x_{t:\tau}|y_t, z_{t:\tau}, c_{0:\tau})$$

and then marginalize over $x_{t:\tau}$. (We typically also would like to know $x_{0:\tau}$ to better understand what happened.)

We then use Bayes' Theorem to find:

$$p(\alpha_{t:\tau}, x_{t:\tau}|y_t, z_{t:\tau}, c_{0:\tau}) = \frac{p(y_t|x_t)}{p(y_t)} \frac{p(z_{t:\tau}|x_{t:\tau}, \alpha_{t:\tau})}{p(z_{t:\tau})} \frac{p(c_{t:\tau}|\alpha_{t:\tau}, c_{0:t})}{p(c_{t:\tau}|c_{0:t})} p(x_{t:\tau}|\alpha_{t:\tau}) p(\alpha_{t:\tau})$$

Formulation of the digital twin problem II

We found:

$$p(\alpha_{t:\tau}, x_{t:\tau}|y_t, z_{t:\tau}, c_{0:\tau}) = \frac{p(y_t|x_t)}{p(y_t)} \frac{p(z_{t:\tau}|x_{t:\tau}, \alpha_{t:\tau})}{p(z_{t:\tau})} \frac{p(c_{t:\tau}|\alpha_{t:\tau}, c_{0:t})}{p(c_{t:\tau}|c_{0:t})} p(x_{t:\tau}|\alpha_{t:\tau}) p(\alpha_{t:\tau})$$

which means that the state evolution $x_{t:\tau}$ and the controls $\alpha_{t:\tau}$ are 'driven' by the observations of the system, y_t , the target evolution, $z_{t:\tau}$, and the costs, $c_{0:\tau}$ via the likelihoods $p(y_t|x_t)$ and $p(z_{t:\tau}|x_{t:\tau}, \alpha_{t:\tau})$ and $p(c_{t:\tau}|\alpha_{t:\tau}, c_{0:t})$.

Their relation is via

$$y_t = H^y(x_t) + \epsilon_t^y \qquad z_{t:\tau} = H^z(x_{t:\tau}, \alpha_{t:\tau}) + \epsilon_{t:\tau}^z \qquad c_{t:\tau} = H^c(\alpha_t, c_{0:t-1}) + \epsilon_t^c$$

$H^y(x_t)$ is given, while the difficult part is to find $H^z(x_{t:\tau}, \alpha_{t:\tau})$ and especially $H^c(\alpha_t, c_{0:t-1})$. The latter includes factors such as the value of human lives versus the cost of property.

The $\epsilon_{t:\tau}^z$ and ϵ_t^c are tolerances, i.e., how flexible we are on target and cost.

Formulation of the digital twin problem III

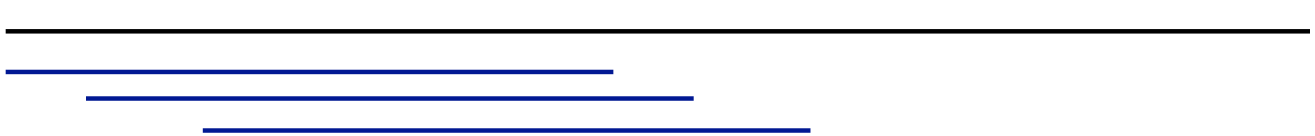
The main point is that, formulated this way, i.e., including full uncertainty, there is no clear separation between the data-assimilation problem and the ‘optimization problem’.

The optimization problem is written as an estimation problem, allowing direct access to powerful methods from data assimilation.

Practical ways to use the formulation

Several practical methods exist to make the problem tractable:

- Restrict the temporal dependence to a fixed time window, i.e. replace time τ with $t + t_w$
- We can then formulate this as a moving window problem, resulting in so-called continuous data assimilation:



Use ML surrogates

or use Gaussian time correlations (to avoid adjoints)

Continuous nonlinear data assimilation I

Explore that observations far in the future do not constrain the solution now in any significant way. Assume this time horizon is t_w.
We can write in general, for any *x* and *z*:

$$p(x_{0:t_w}|z_{0:t_w}) = p(x_{1:t_w}|z_{0:t_w}, x_0)p(x_0|z_{0:t_w})$$

and use the time horizon idea:

$$p(x_0|z_{0:t_w}) = p(x_0|z_{0:t_w-1})$$

The particles at time 0 will not change with future target states for $t \geq t_w$, so they are fixed when the latest *z* arrive, and we can write:

$$p(x_0|z_{0:t_w-1}) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_0 - x_0^i)$$

Continuous nonlinear data assimilation II

This allows us to integrate out x^0:

$$p(x_{1:t_w}|z_{1:t_w}) = \int p(x_{1:t_w}|z_{1:t_w}, x_0)p(x_0|z_{1:t_w})\, dx_0 = \frac{1}{N}\sum_{i=1}^{N} p(x_{1:t_w}|z_{1:t_w}, x_0^i)$$

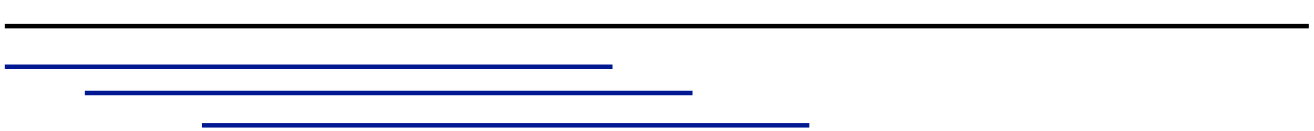
$$= \frac{1}{N}\sum_{i=1}^{N} \frac{p(z_{1:t_w}|x_{1:t_w}, x_0^i)}{p(z_{1:t_w})} p(x_{1:t_w}|x_0^i) = \prod_{t=2}^{t_w} \frac{p(z_k|x_{1:t_w}, x_0^i)}{p(z_{1:t_w})} p(x_t|x_{t-1})p(x_1|x_0^i)$$

The beauty of this equation is that it **only contains transition densities and likelihoods** at the different time steps. Transition densities correspond to model errors.
We 'know' these analytically, so the gradient of the posterior can be calculated.

Hence, we do not need to specify $p(x_0)$, we just continue from the particles at the start of the window.

Practical ways to use the formulation

Use continuous data assimilation



We cannot use the same observations twice, ***only new observations at end of the window!***

This will result in:
- Continuous DA, no shocks at the start of a window
- No Gaussian priors necessary, prior only from transition densities!
- Fully nonlinear methodology
- Allows for long-window smoother
- Each new window uses the solution of the previous window as prior, and only assimilates new observations at the end of the window, hence fast convergence.

Langevin-based Monte-Carlo

The Langevin equation is known to generate samples from pdf *p(x)*, starting from samples from any pdf.

It can be generalized as

$$dx = (D + Q) \cdot \nabla \log p(x) dt + \nabla \cdot (D + Q) dt + \sqrt{2} D^{1/2} dW$$

For any positive semi-definite matrix D and antisymmetric matrix Q.

This methodology is used in generative diffusion ML models as follows:

Choose $D = \sigma^2(t) I$ and $Q = 0$ and use a standard Gaussian as target pdf, such that

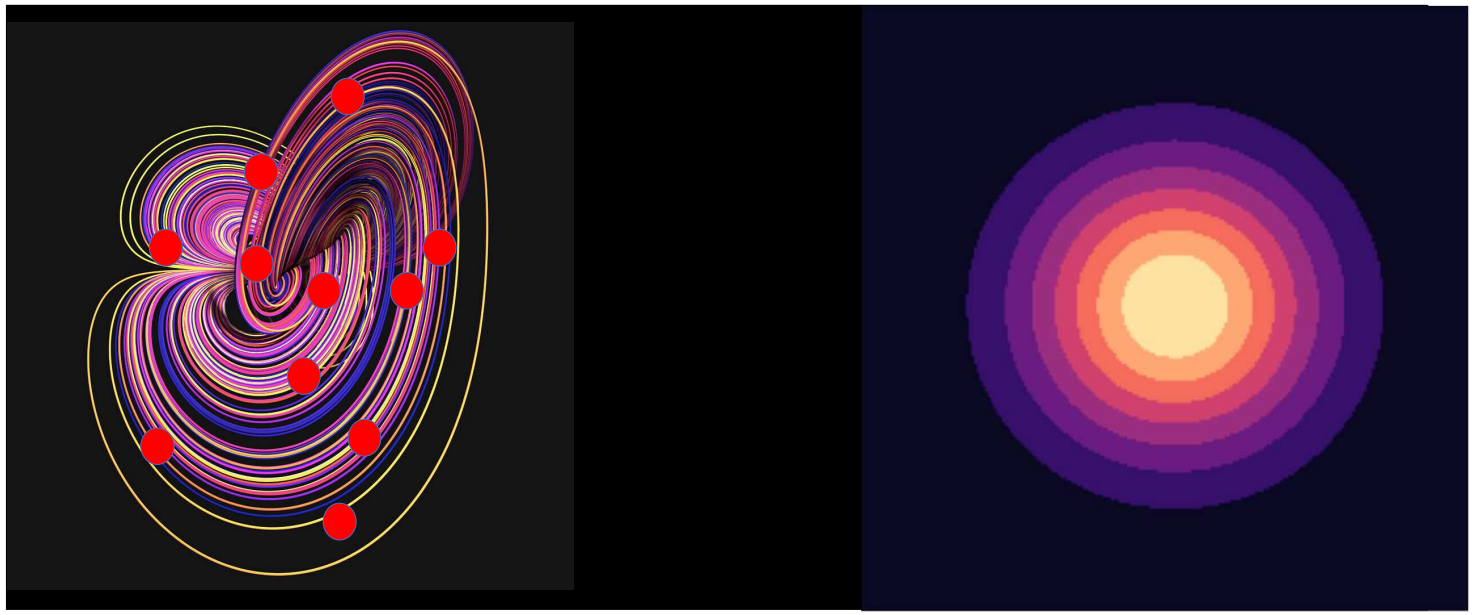
$$\nabla \log p_G(x) = -x$$

Hence, simply add random white noise and this forcing at every timestep or iteration.

Machine learning: Diffusion-based generative methods

Step 1: Transform prior ensemble members to samples from standard Gaussian

Prior pdf
$p(x)$



Multivariate standard Gaussian

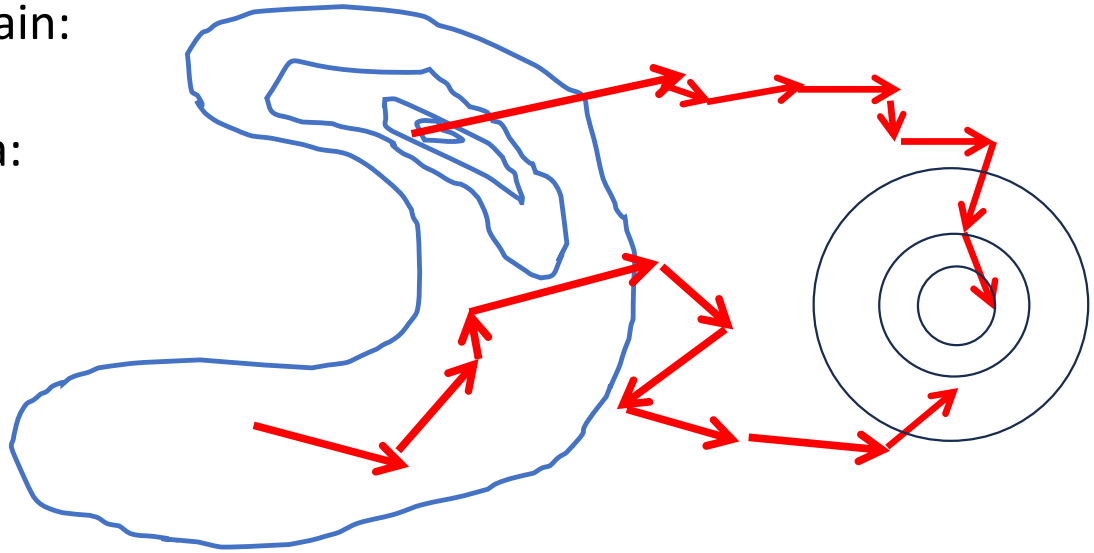
The transformation is done by adding Gaussian noise to the samples in an iterative way, and add a forcing towards the standard Gaussian.

Langevin dynamics to move prior samples to Gaussian samples

For each particle we will have a Markov chain:

The MC evolves forward in pseudo time via:

$$dx^t = -bx^t dt + \sigma dW$$

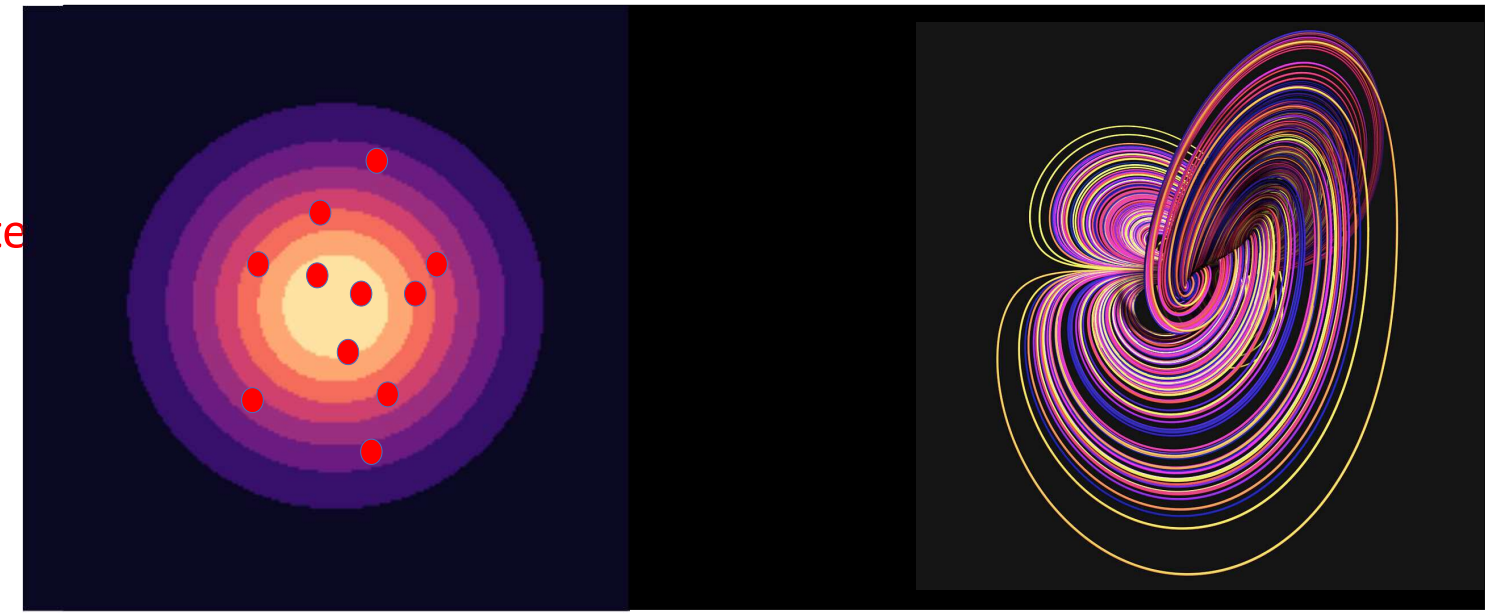


Since the SPDE is linear **the transition densities** $p(x^t|x^{t-1})$ **are all Gaussian!**

Machine learning: Diffusion-based generative modeling

Step 2: Transform standard Gaussian samples to prior samples

Multivariate standard Gaussian



Prior pdf $p(x)$

This transformation is done by adding noise and **adding a forcing towards the prior.**
Learn this forcing towards the prior **or use analytical form**.

ML or use analytical form…

For each particle we will have a Markov chain:

The MC evolves backward in pseudo time via:

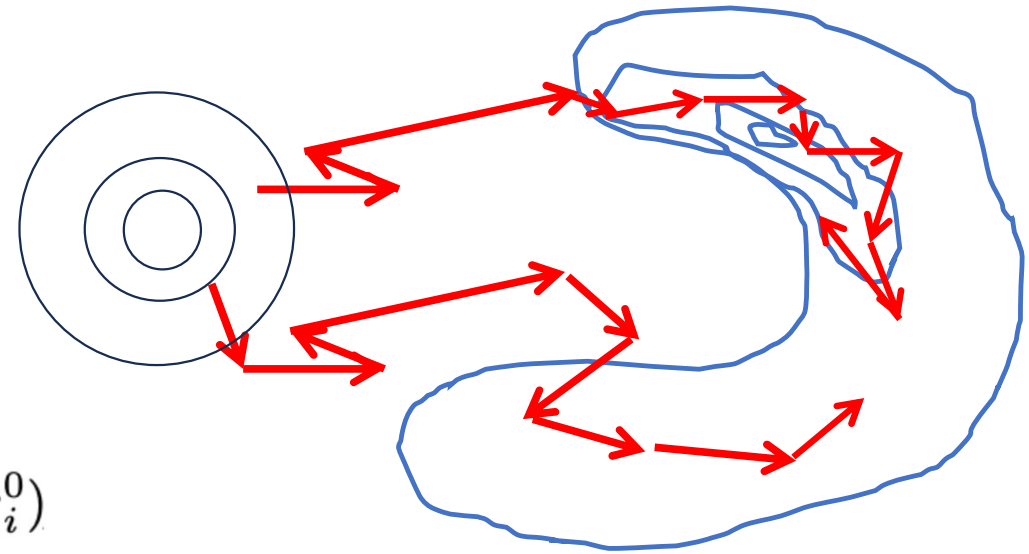
$$dx^t = bx^t dt - \sigma^2 \nabla \log p^t(x^t) dt + \sigma dW$$

in which:

$$p^t(x^t) = \int p(x^t|x^0) p(x^0)\, dx^0 = \sum_{i=1}^{N} p(x^t|x_i^0)$$

and each $p(x^t|x_i^0)$ is Gaussian!
Hence, no need to learn it, fast to calculate directly.

But what about data assimilation?



Diffusion-based data assimilation

Add likelihood forcing $\nabla \log p(y|x)$ to the evolution equation, which will lead to posterior?

$$dx^t = bx^t dt - \sigma^2 \nabla \log p^t(x^t) dt - \sigma^2 h^t \nabla \log p(y|x^t) dt + \sigma dW$$

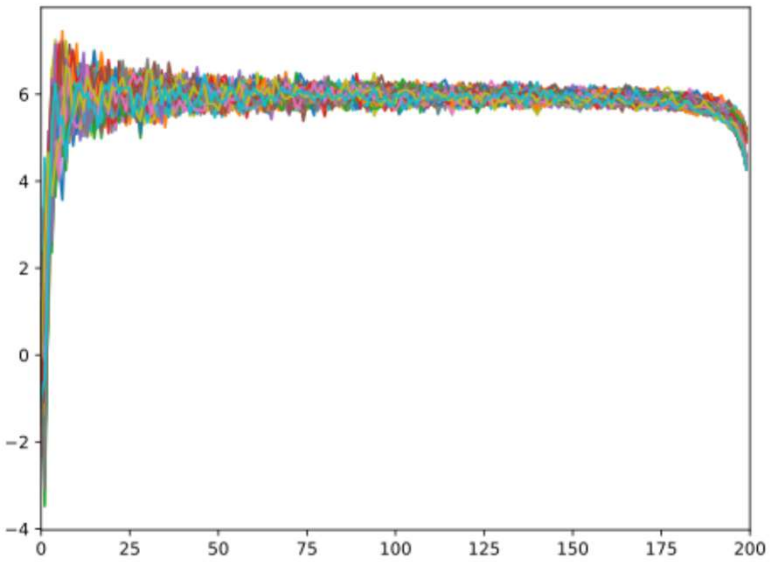
Prior pdf
$p(x)$

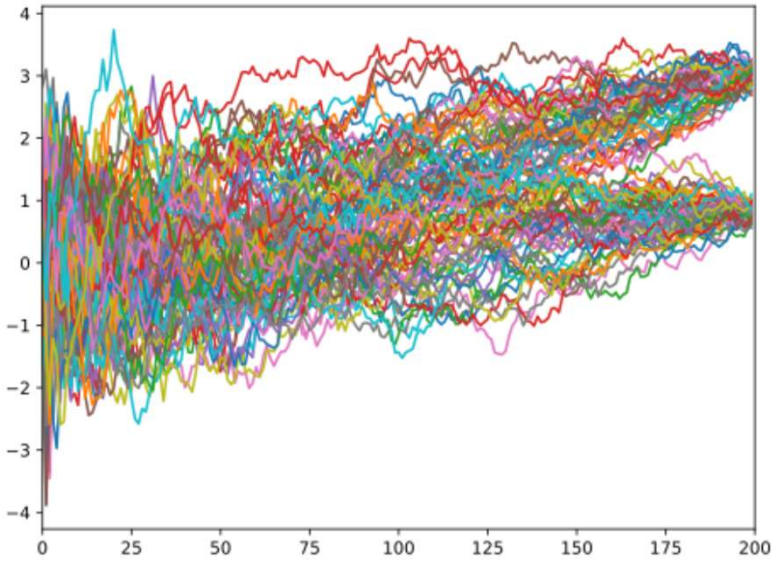
Posterior pdf
$p(x|y)$

However, there is no principled way to weight the prior and likelihood forcings, yet…

Example: backward flow with extra likelihood term

Prior: 2-dimensional bimodal Gaussian with means (3,3) and (0.8,0.8), standard deviations 0.1.
Observe first component, y=6, standard deviation 0.1. Use 100 particles.





Backward flow from standard Gaussian to 'posterior pdf'. Two issues:

- likelihood and prior forcing not 'well balanced', and
- no communication between state components

Indeed, this is not correct!

Physical nudging **in real time**

Consider the problem **in real time** of 'nudging' particles from a start time towards a set of observation at observation time. (This is also called synchronization.)

If the model equation is given by:

$$dx_t = M(x_t)dt + \sqrt{2}D^{1/2}dW_t$$

it is modified to

$$dx_t = M(x_t)dt + \sqrt{2}D^{1/2}dW_t + \Gamma_t(y_{t_{obs}} - H(x_t))$$

and the goal is to find Γ_t such that the model smoothly converges towards the observation, and ends up at the observation within the observation uncertainty.

We can make this more formal!

Physical nudging

Consider the pdf $$P(x_t) = \frac{p(y_1|x_t)p(x_t|x_0)}{p(y_1|x_0)}$$

which is equal to the prior pdf at t=0: $$P(x_0) = \frac{p(y_1|x_0)p(x_0|x_0)}{p(y_1|x_0)} = \delta(x - x_0)$$

and equal to the posterior pdf at t=1: $$P(x_1) = \frac{p(y_1|x_1)p(x_1|x_0)}{p(y_1|x_0)} = p(x_1|y_1, x_0)$$

Assume *x* evolves under $p(x_t|x_0)$ as

$$dx_t = M(x_t)dt + \sqrt{2}D^{1/2}dW_t$$

Then we can find the evolution equation for $P(x_t)$ as follows.

Physical nudging

Remember $$P(x_t) = \frac{p(y_1|x_t)p(x_t|x_0)}{p(y_1|x_0)}$$

Using the forward and backward Kolmogorov equations on $p(x_t|x_0)$ and $p(y_1|x_t)$, find:

$$\frac{\partial P(x_t)}{\partial t} = -\nabla_x \cdot \left[M(x_t)P(x_t) + D \cdot \nabla_x \log p(y_1|x_t)P(x_t)\right] + \frac{1}{2}\nabla_x \cdot [D \cdot \nabla P(x_t)]$$

with corresponding SDE

$$dx_t = M(x_t)dt + D \cdot \nabla_x \log p(y_1|x_t)dt + \sqrt{2}D^{1/2}dW_t$$

This SDE will bring a prior particle starting at x_0 to a posterior particle at time 1.

The likelihood term

The likelihood follows the Kolmogorov backward equation:

$$\frac{\partial p(y_1|x_t)}{\partial t} = -M(x_t) \cdot \nabla_x p(y_1|x_t) - \frac{1}{2}\nabla_x \nabla_x p(y_1|x_t)$$

with final condition $p(y_1|x_1)$ at $t = t_1 = t_{obs}$

When *M* is nonlinear, this equation has to be solved numerically.
Instead, we can use the small time approximation (Conti, PJvL, Anderson 2025), which, for Gaussian observation errors, leads to:

$$p(y_1|x_t) = c \exp\left[-\frac{1}{2}(y_1 - H(x_t) - \mathbf{H}M(x_t)(t_1 - t))\mathbf{A}^{-1}(y_1 - H(x_t) - \mathbf{H}M(x_t)(t_1 - t))\right)$$

in which $\mathbf{A} = \mathbf{H}\mathbf{D}\mathbf{H}^T(t_1 - t) + \mathbf{R}$

Stochastic physical nudging equation

This leads to:

$$dx_t = M(x_t)dt + K_D\left(y_1 - \tilde{H}(x_t)\right)dt + \sqrt{2}D^{1/2}dW_t$$

in which

$$K_D = DH^T\left(HDH^T + R\right)^{-1} \qquad \text{and} \qquad \tilde{H}(x_t) = H(x_t) + HM(x_t)(t_1 - t)$$

This can be compared to the traditional nudging equation:

$$dx_t = M(x_t)dt + \Gamma_t(y_{t_1} - H(x_t)) + \sqrt{2}D^{1/2}dW_t$$

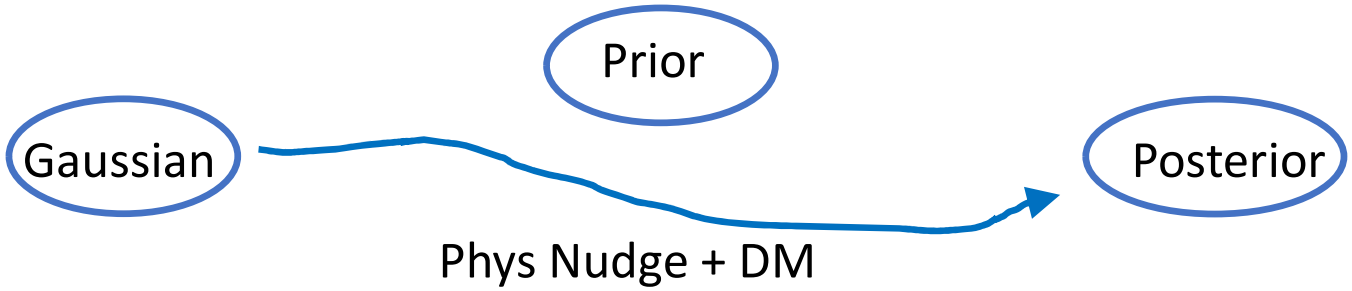
showing that the traditional method is a special case of the physical nudging scheme.

Physical nudging into diffusion method

We use physical nudging in pseudo time instead of real time.
To obtain samples from the posterior pdf, we need to start from the prior pdf. However, diffusions start from the standard Gaussian.

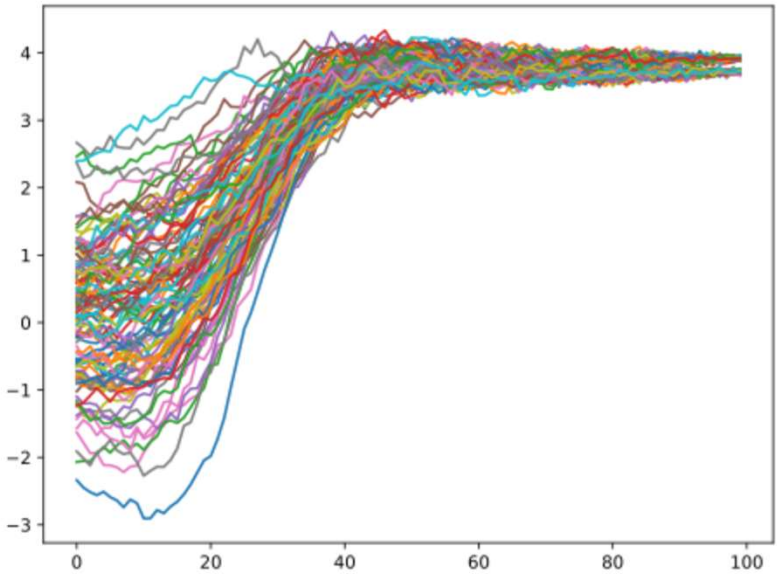
In principle, we can do:

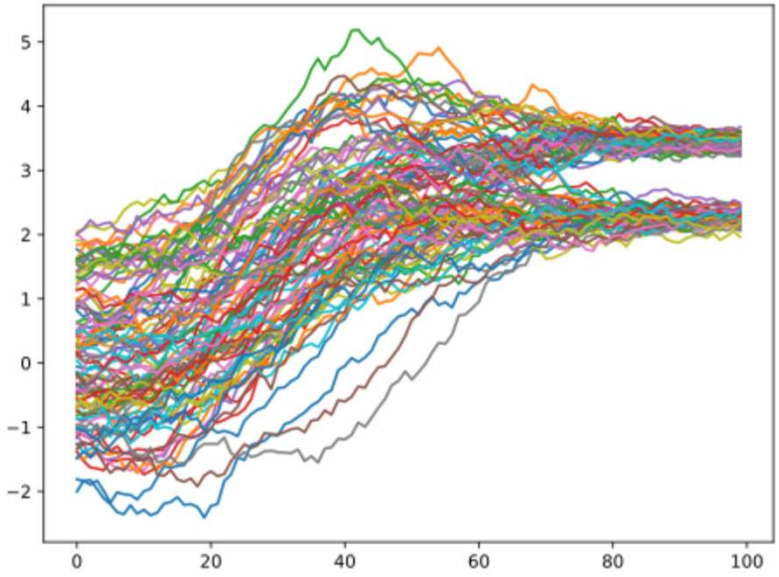
but not very efficient…
However, we can combine them into one:



Example: backward flow with extra likelihood term

Prior: 2-dimensional bimodal Gaussian with means (3,3) and (0.8,0.8), standard deviations 0.1.
Observe first component, y=4, standard deviation 0.1. Use 100 particles.





The correct answer!

(Note we used time stepping ideas from stochastic interpolants.)

Conclusions and Take-home message

- **The Digital Twin problem can be reformulated as a Bayesian Inference problem**
- **This shows how to include uncertainty in a principled way, and approximations become well defined.**
- **Many possibilities for ML, we discussed generative diffusion methods**
- **Diffusion generative methods** use ad-hoc infusion of likelihood, existing methods do not generalize.
- **Combining diffusive generative methods with physical nudging** might solve this.

Stochastic particle flow

We can generalize the Langevin dynamics:

$$dz = f(z)ds + Qd\beta$$

Now define $z = (x_1^T, ..., x_N^T)^T$ and **consider the Langevin dynamics over the configuration space spanned by the N particles.** (Gallego and Insua 2020; Leviyev et al., 2022)

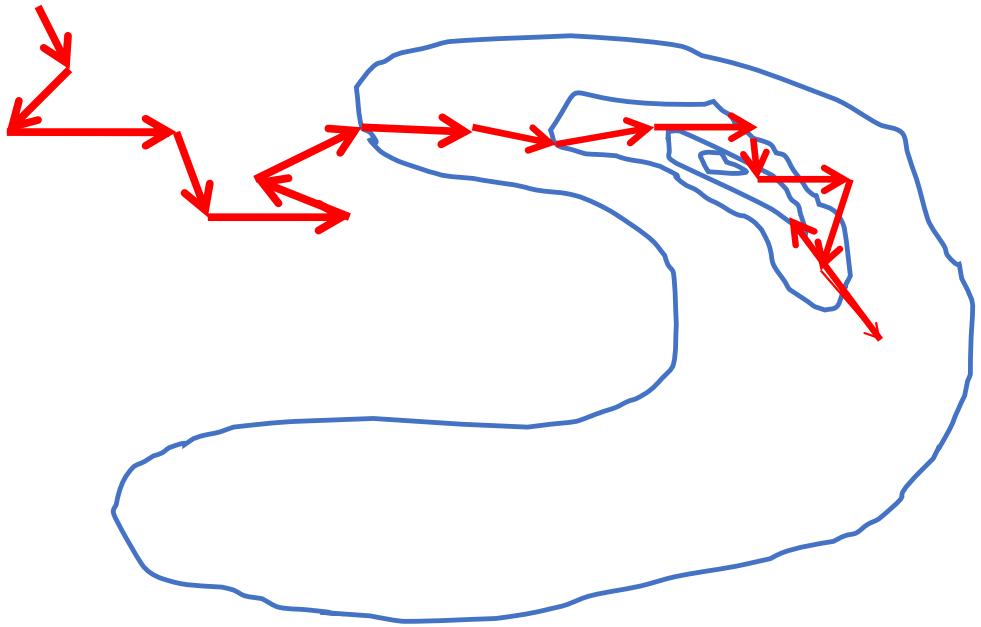
If one chooses

$$f(z) = D\,\nabla \log p(z|y) + \nabla \cdot D \qquad \text{and} \qquad Q = \sqrt{2}\,D^{1/2}$$

then this MC generates samples from $p(z|y) = \prod_{i=1}^{N} p(x_i|y)$

Using Markov Chain to sample from the posterior

Note we have one chain that contains all particles!



Stochastic Particle Flow Filter

If we choose $D = D_B C_K$ we find the Stochastic Particle Flow Filter:

$$\Delta x_i = \frac{1}{N}\sum_{j=1}^{N} K_{ij} B \nabla_{x_j} \log p(x_j|y)\Delta s + B\nabla_{x_j} K_{ij}\Delta s + \sqrt{2\Delta s}\beta_i$$

attracting term repulsive term stochastic term

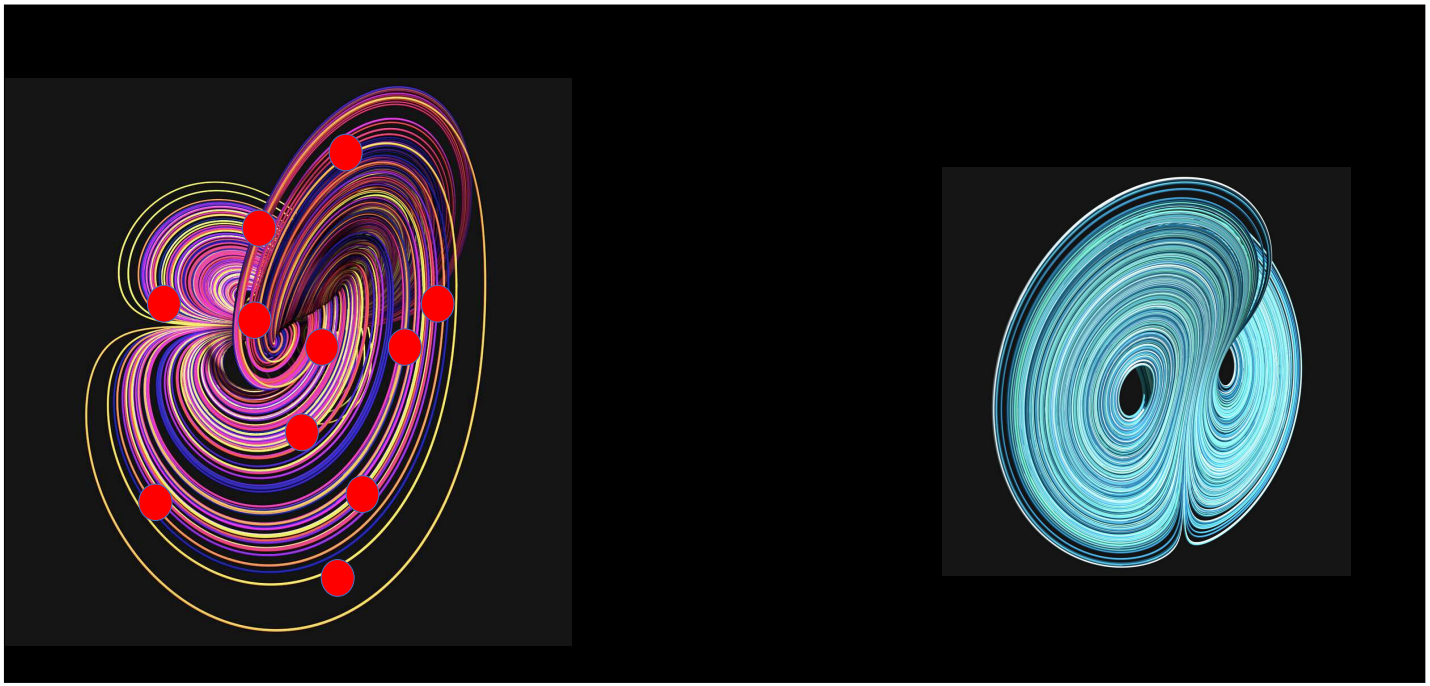
This flow leads to an unbiased posterior independent of ensemble size!

Nonlinear data assimilation: Particle flow Filters

Geophysical data assimilation is a nonlinear problem.

Prior pdf

$p(x)$



Posterior pdf

$p(x|y)$

The prior and posterior can be for a model state (filter) or a model trajectory (smoother) or a set of parameters, or a combination of these.

Stochastic Particle Flow Filter

The Stochastic Particle Flow Filter reads:

$$\Delta x_i = \frac{1}{N}\sum_{j=1}^{N} K_{ij} B \nabla_{x_j} \log p(x_j|y)\Delta s + B\nabla_{x_j} K_{ij}\Delta s + \sqrt{2\Delta s}\beta_i$$

This flow leads to an unbiased posterior independent of ensemble size!

However $\nabla \log p(x|y) = \nabla \log p(y|x) + \nabla \log p(x)$, and we know the likelihood, but the prior is only known via a few samples.
Stochastic PFF practice uses Gaussian pdf for $p(x)$. This can be bad…

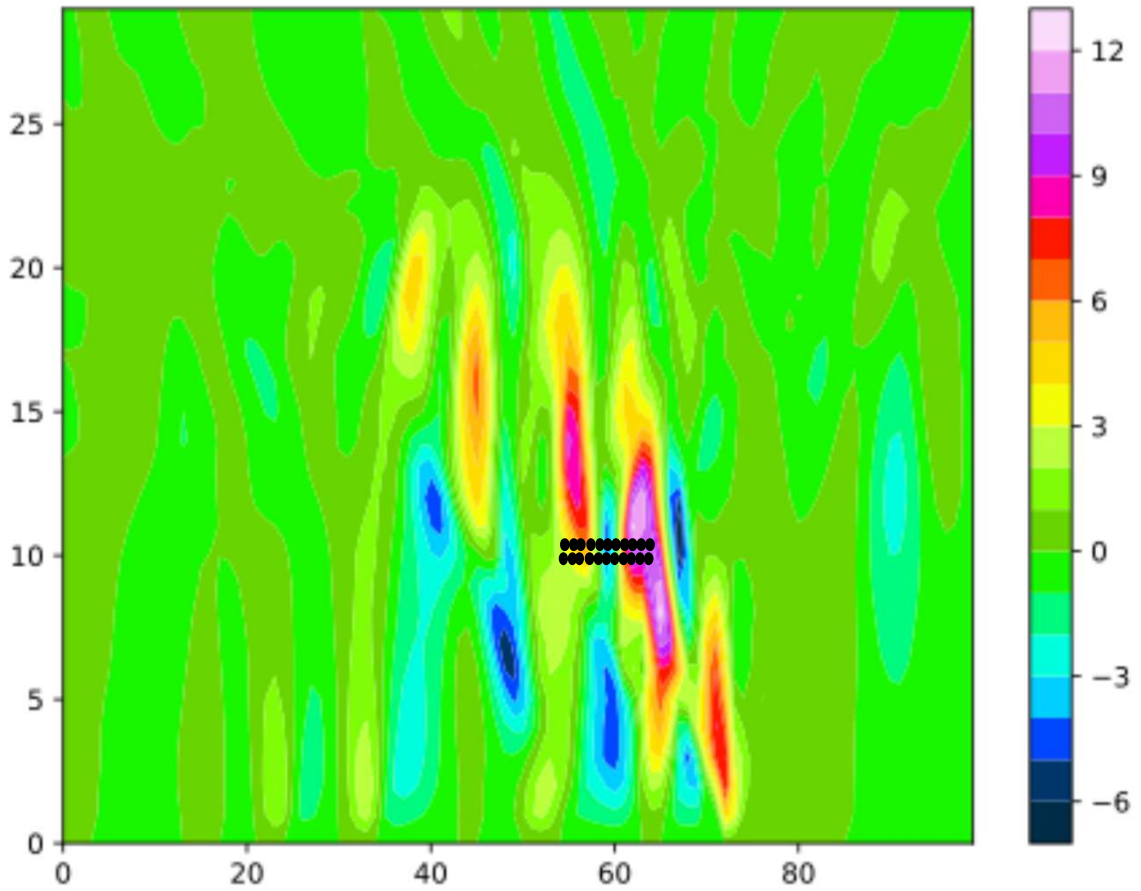
Diffusion PFF: use diffusion generative model to approximate $\nabla \log p(x)$.

Example: Squall line

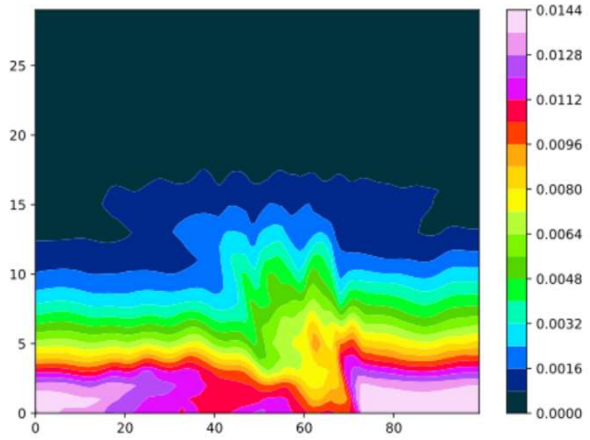
Application on squall-line case, using Cloud Model 1 (CM1)

- dx = 1 km, dz = 500 m
- Observe w at 20 grid points
- 50 particles
- Obs error standard deviation 0.2
- Extremely nonlinear!

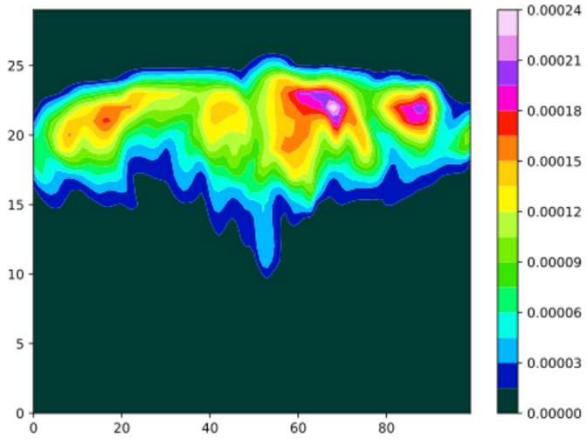
Vertical velocity.
Black dots are observation locations.



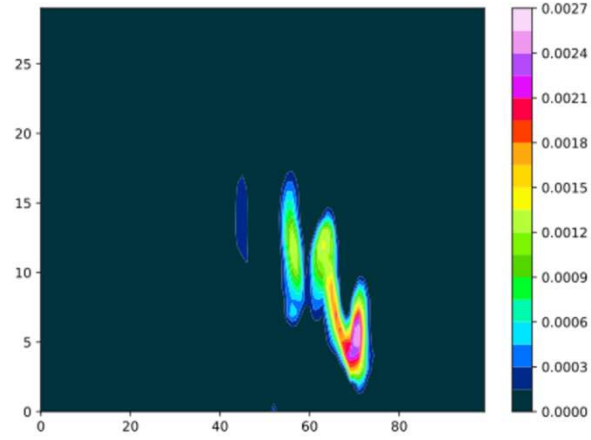
Application on squall-line case, using Cloud Model 1 (CM1)



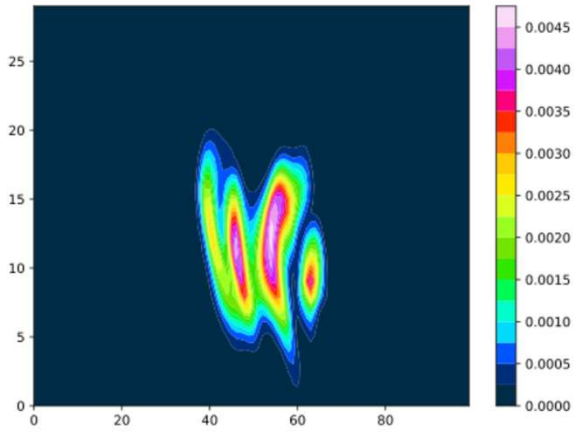
Water vapor and Graupel

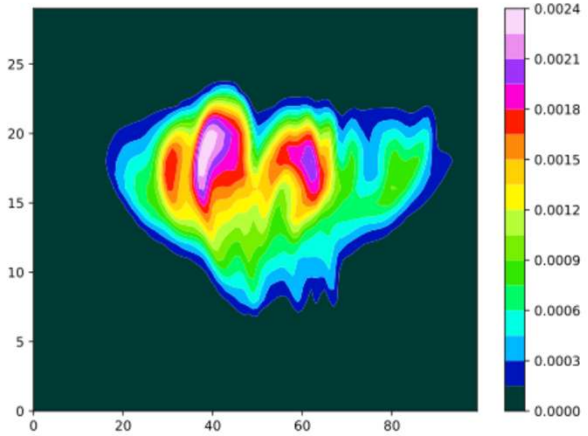


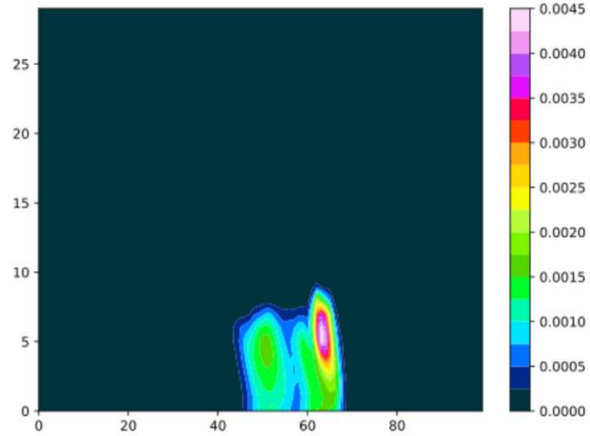
ice and snow



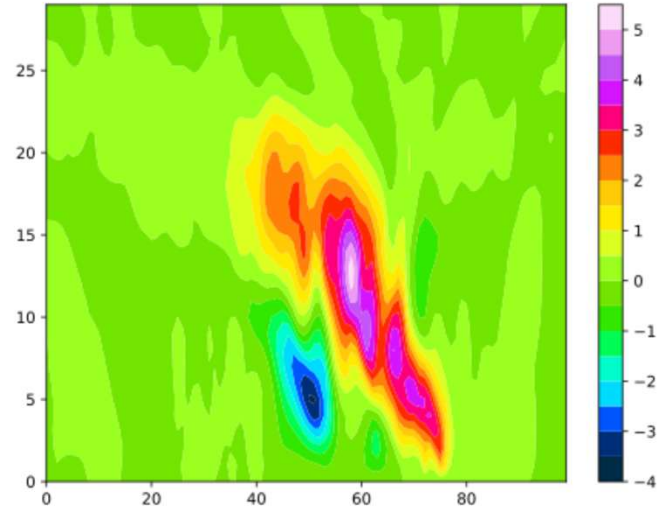
cloud liquid water and rain



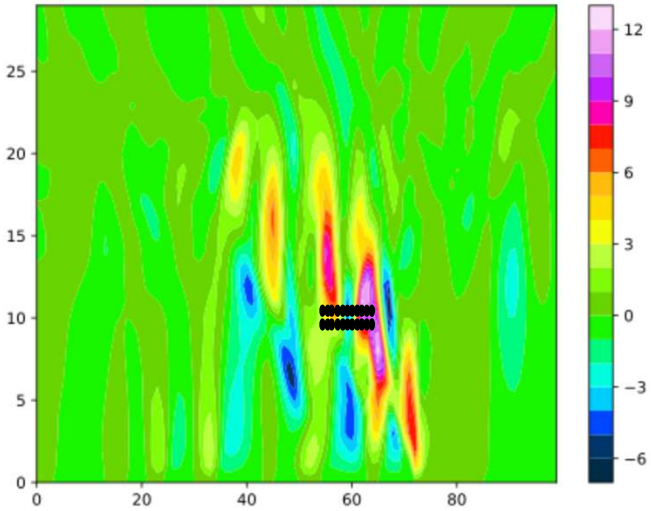




Results on vertical velocity

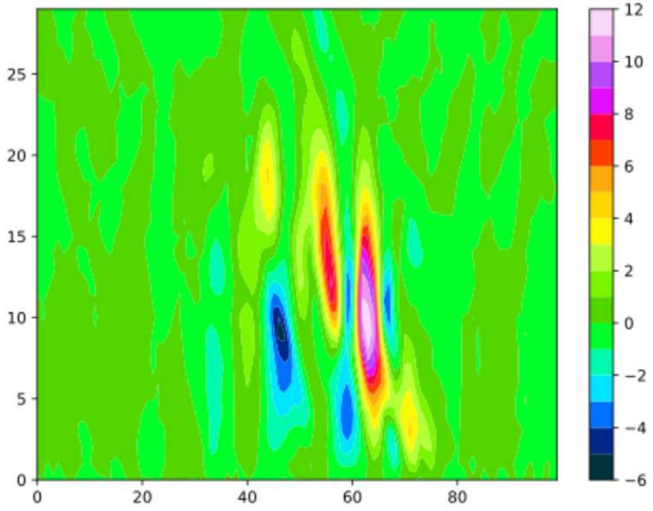


Prior mean



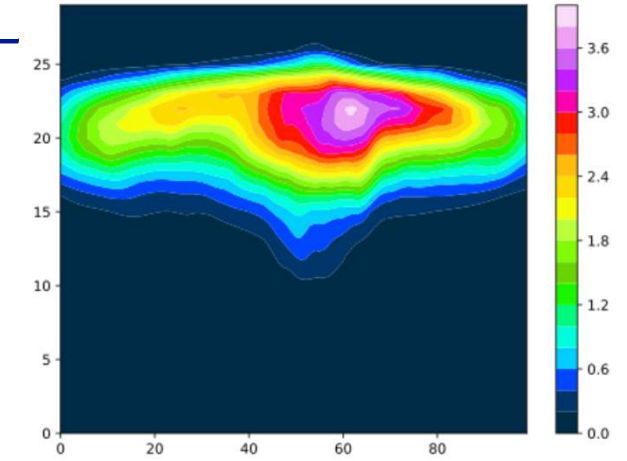
True state

Black dots are observation locations

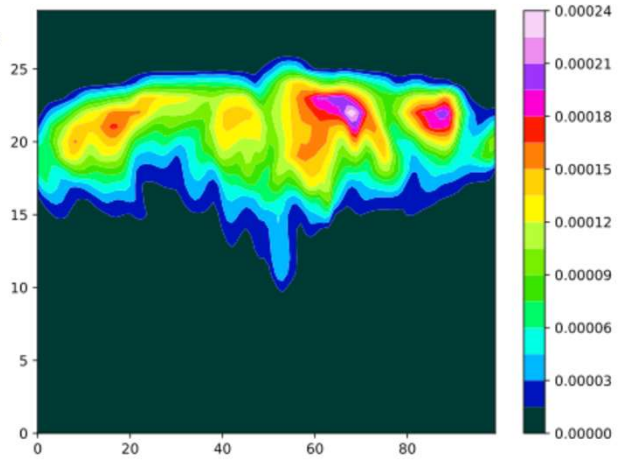


posterior mean

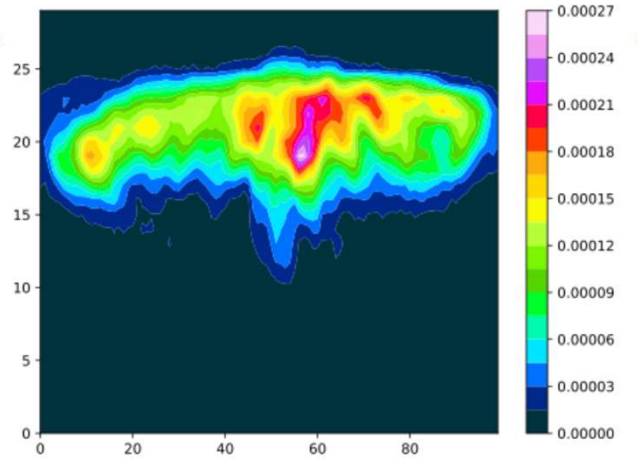
Results on ice and snow content



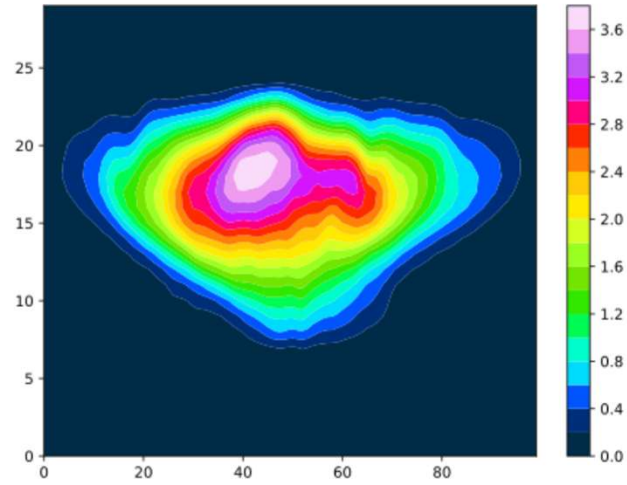
Prior mean

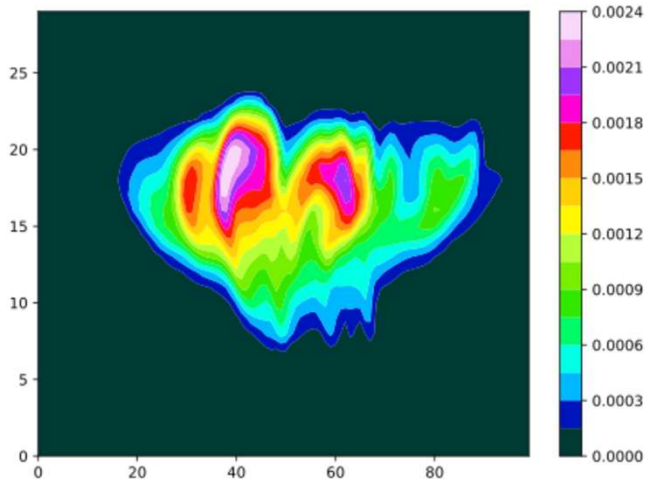


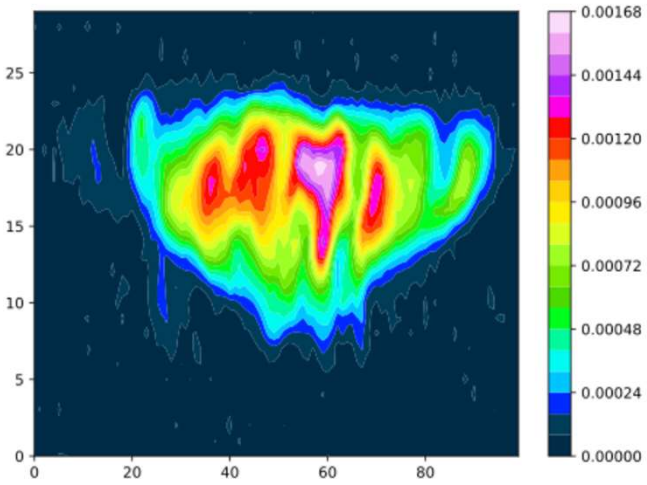
True state



posterior state







Stochastic Particle Flow Filter

The Stochastic Particle Flow Filter reads:

$$\Delta x_i = \frac{1}{N}\sum_{j=1}^{N} K_{ij} B \nabla_{x_j} \log p(x_j|y)\Delta s + B\nabla_{x_j} K_{ij}\Delta s + \sqrt{2\Delta s}\beta_i$$

in which $\beta_i = \left(L_B N L_k\right)_i$ with Cholesky factors *L,* and *N* is a matrix with standard Gaussian noise values, and kernel

$$K_{ij} = K(\mathbf{x}_s^i, \mathbf{x}_s^j) = \exp\left(-\frac{1}{2}(\mathbf{x}_s^i - \mathbf{x}_s^j)^T(\alpha\mathbf{B})^{-1}(\mathbf{x}_s^i - \mathbf{x}_s^j)\right)$$

and

$$k = \frac{1}{N_p}\begin{bmatrix} K(\mathbf{x}_s^1, \mathbf{x}_s^1) & K(\mathbf{x}_s^1, \mathbf{x}_s^2) & \cdots & K(\mathbf{x}_s^1, \mathbf{x}_s^{N_p}) \\ K(\mathbf{x}_s^2, \mathbf{x}_s^1) & K(\mathbf{x}_s^2, \mathbf{x}_s^2) & \cdots & K(\mathbf{x}_s^2, \mathbf{x}_s^{N_p}) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_s^{N_p}, \mathbf{x}_s^1) & K(\mathbf{x}_s^{N_p}, \mathbf{x}_s^2) & \cdots & K(\mathbf{x}_s^{N_p}, \mathbf{x}_s^{N_p}) \end{bmatrix}$$