Nonlinear Data Assimilation and Particle Filters

Peter Jan van Leeuwen

Data Assimilation Research Centre

University of Reading

National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL

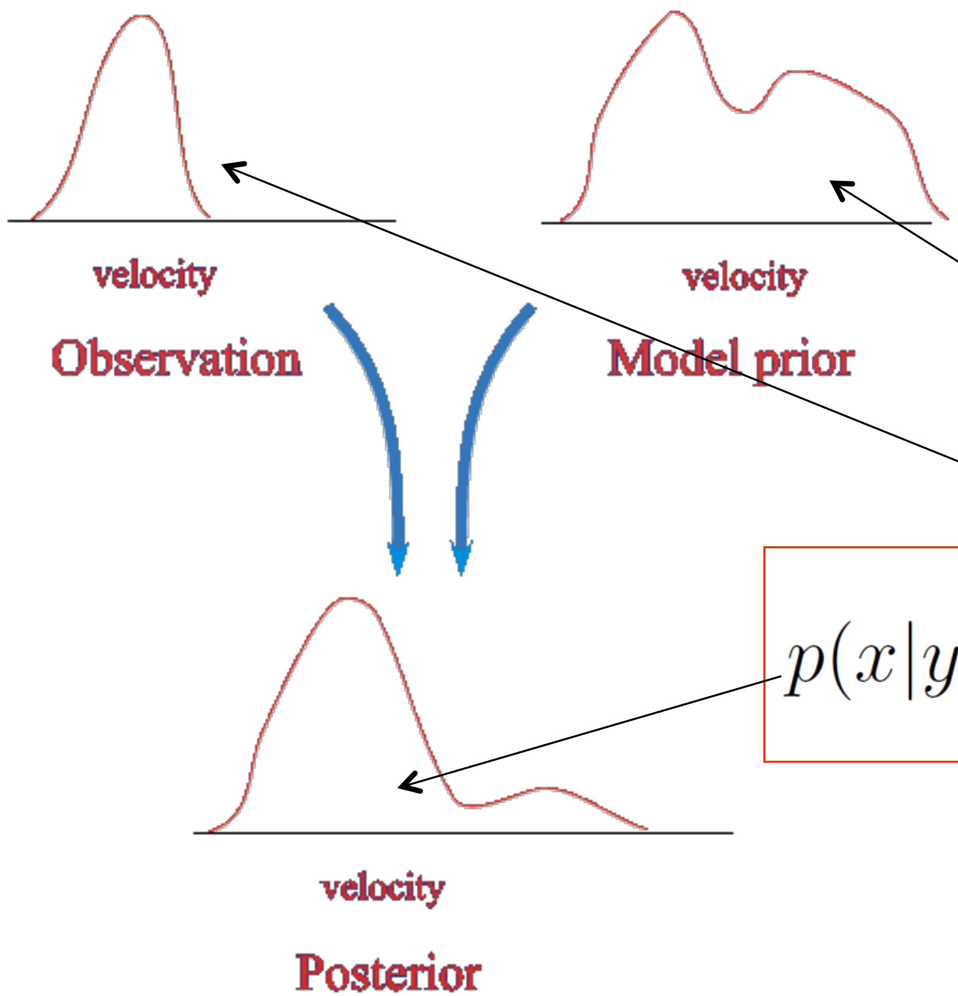
Big Data

- How big is the nonlinear data-assimilation problem?
- Assume we need 10 frequency bins for each variable to build the joint pdf of all variables.
- Let's assume we have a modest model with a million variables.
- Then we need to store $10^{1,000,000}$ numbers.
- The total number of atoms in the universe is estimated to be about 10^{80}.
- So the data-assimilation problem is larger than the universe…

Present-day methods

- Find mode of posterior (very efficient methods for high-dimensional weakly nonlinear problems, e.g. 4DVar).
- Note that first guess is typically quite good, so linearisation makes sense.
- Gaussian assumptions on prior and likelihood, Ensemble Kalman Filters.
- Hybrids between the two.
- But e.g. high-resolution NWP is highly nonlinear…

Data assimilation: general formulation



Bayes theorem:

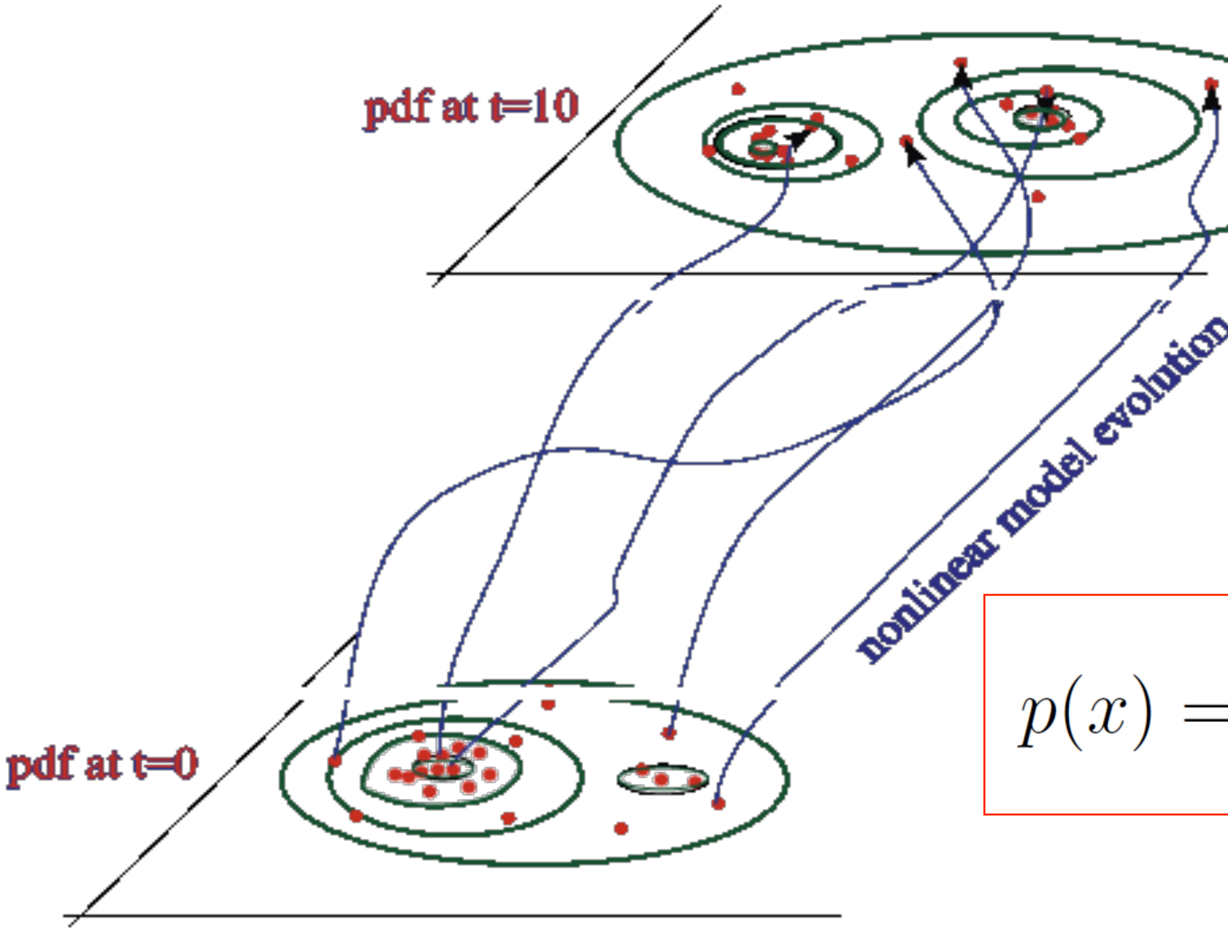
$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\ dx}$$

Solution is pdf!

NO INVERSION !!!

Motivation ensemble methods:

'Efficient' propagation of pdf in time

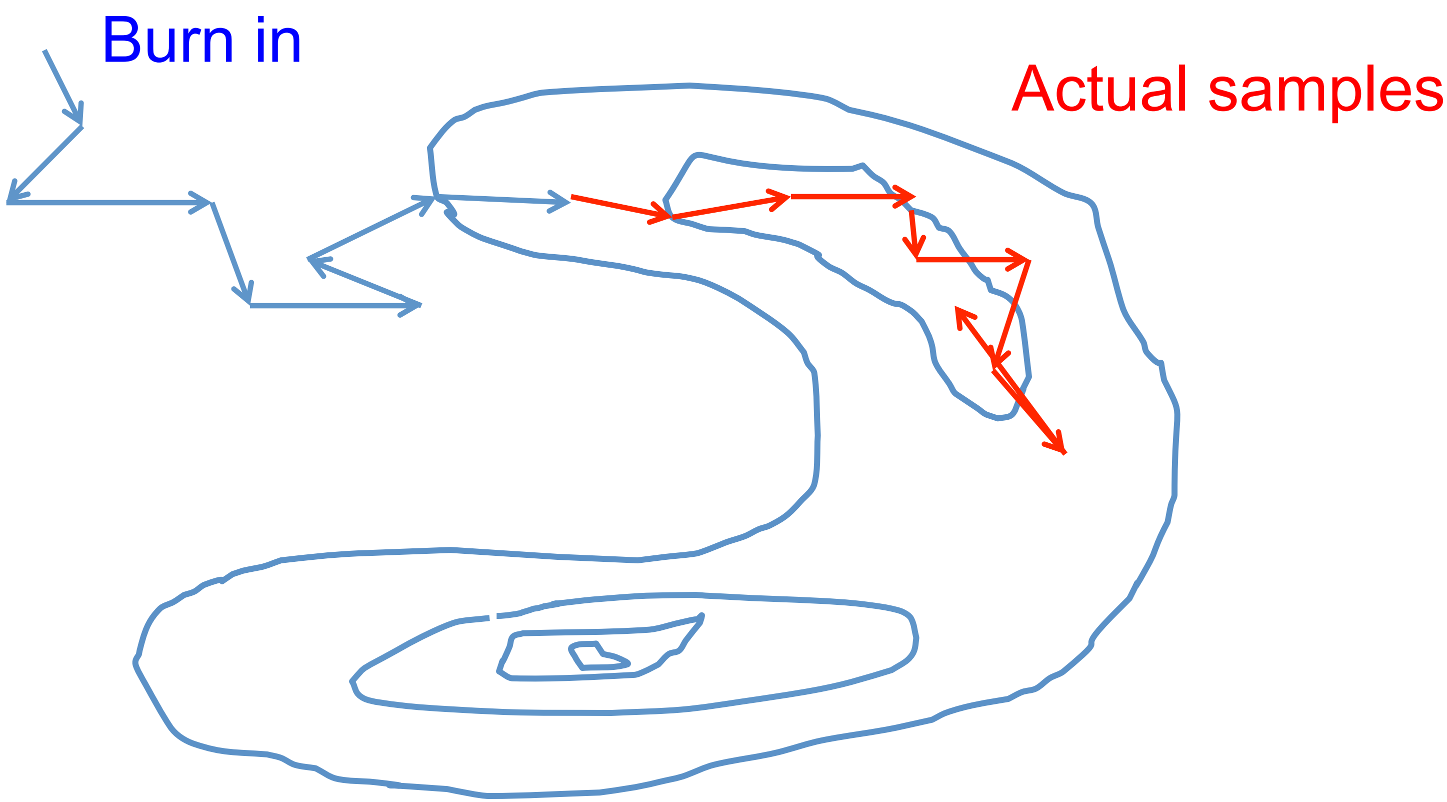


$$p(x) = \sum_{i=1}^{N} \frac{1}{N} \delta(x - x_i)$$

Non-linear Data Assimilation

- Metropolis-Hastings Start from one sample, generate a new one and decide on acceptance (better, or by chance), etc. Slow convergence, but new smarter algorithms are being devised.
- Langevin sampling Idem, but always accept, each sample expensive. Slow convergence, but smarter algorithms are being devised.
- Hamiltonian Monte-Carlo idem, but almost always accept, each sample expensive, faster convergence

All these methods use Markov Chains to sample from the posterior



Non-linear Data Assimilation

- Particle Filters/Smoothers Generate samples in parallel sequential over time and weight them according how good they are. Importance sampling. Can be made very efficient.
- Combinations of MH and PF Expensive but good for e.g. parameter estimation.

The Particle filter

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\ dx}$$

↓ Use ensemble

$$p(x) = \sum_{i=1}^{N} \frac{1}{N}\delta(x - x_i)$$

$$p(x|y) = \sum_{i=1}^{N} w_i \delta(x - x_i)$$

with

$$w_i = \frac{p(y|x_i)}{\sum_j p(y|x_j)}$$

the weights.

What are these weights?

- The weight w_i is the normalised value of the pdf of the observations given model state x_i.
- For Gaussian distributed variables it is given by:

$$
\begin{aligned}
w_i &\propto p(y|x_i) \\
&\propto \exp\left[-\frac{1}{2}\left(y - H(x_i)\right)^T R^{-1}\left(y - H(x_i)\right)\right]
\end{aligned}
$$

- That is all !!!

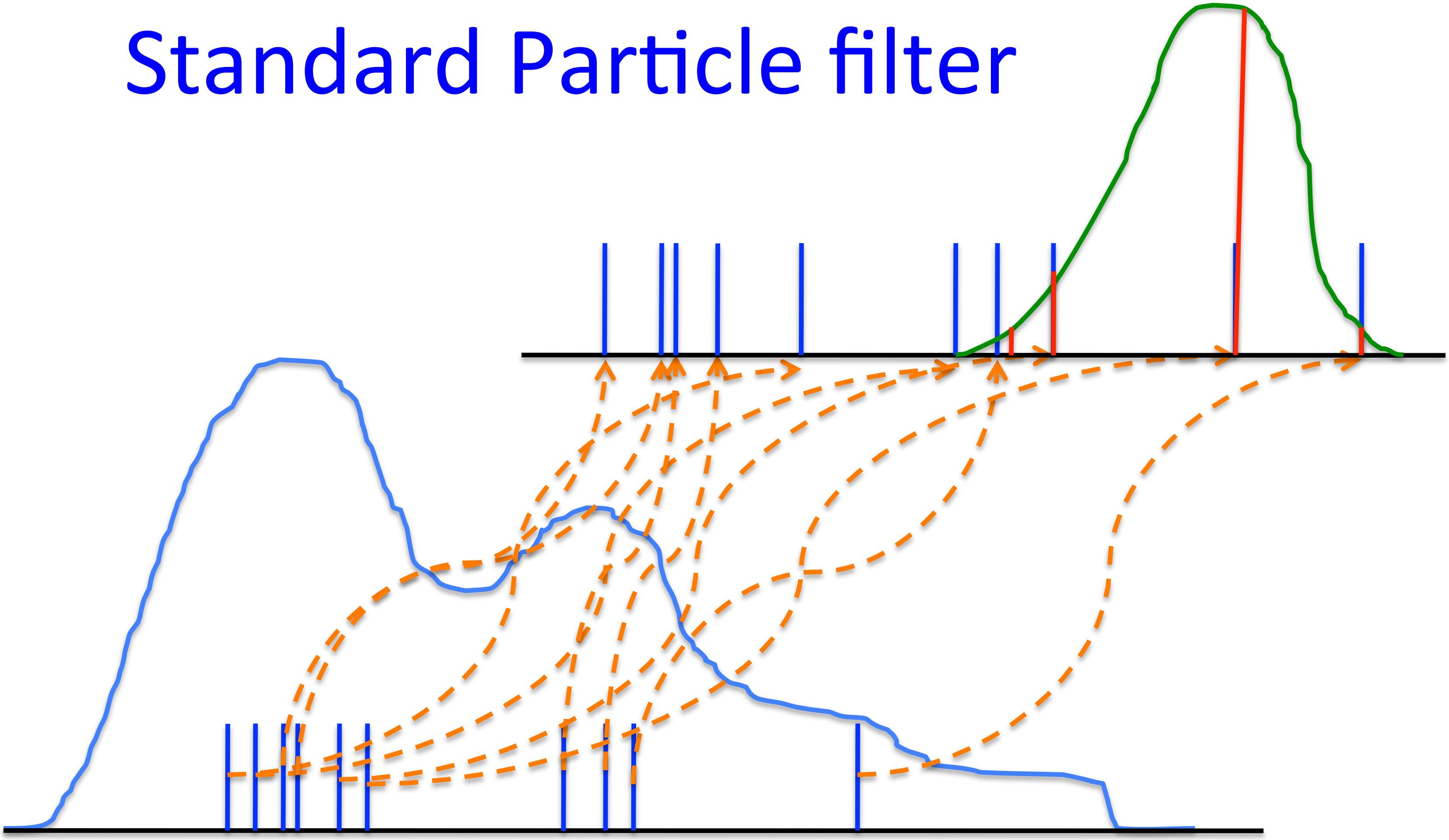
No explicit need for state covariances

- 3DVar and 4DVar need a good error covariance of the prior state estimate: complicated

- The performance of Ensemble Kalman filters relies on the quality of the sample covariance, forcing artificial inflation and localisation.

- Particle filter doesn't have this problem, but…

Standard Particle filter



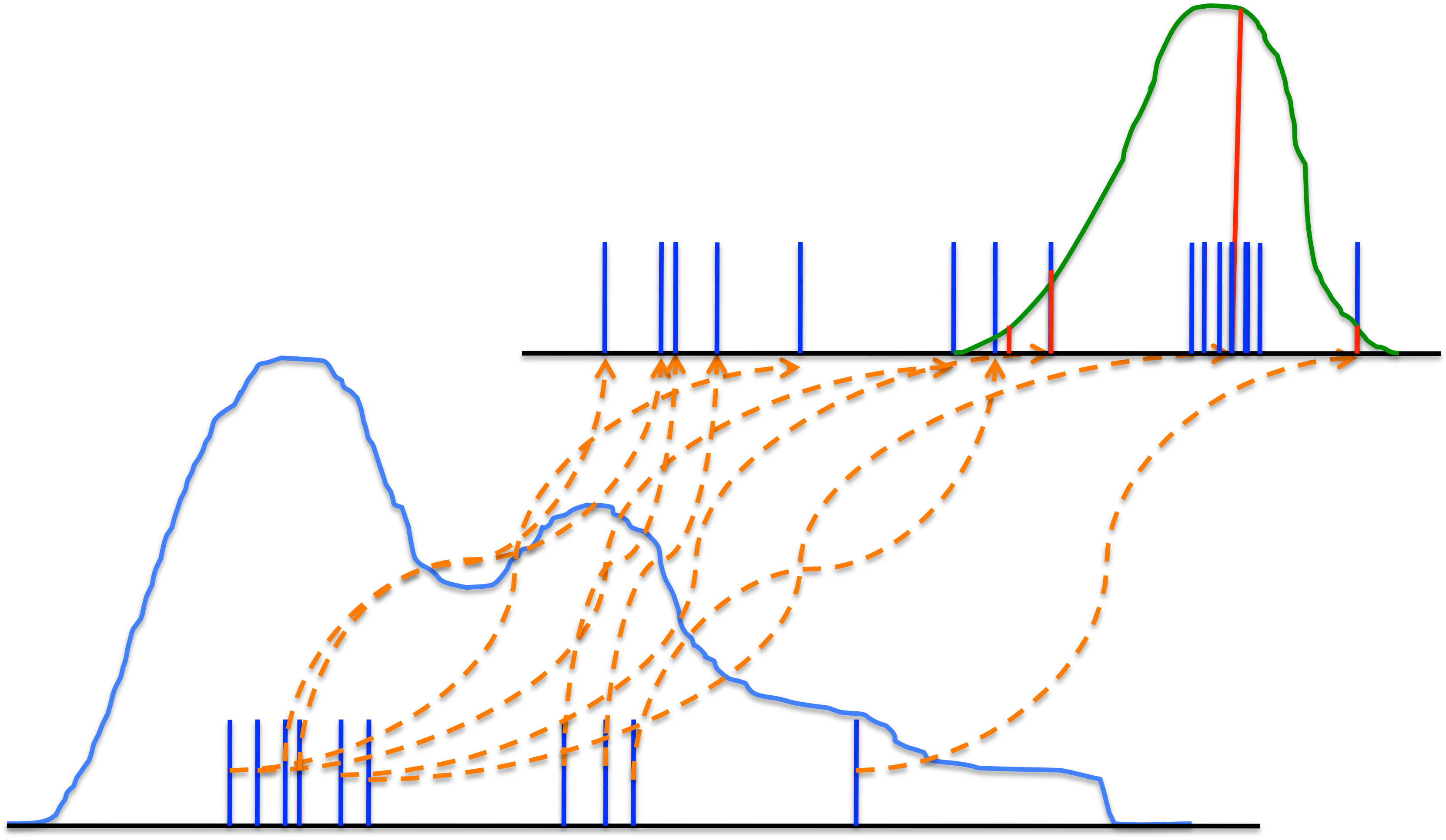
The standard particle filter is degenerate for moderate ensemble size in moderate-dimensional systems.

Particle Filter degeneracy: resampling

- With each new set of observations the old weights are multiplied with the new weights.
- Very soon only one particle has all the weight…
- Solution:

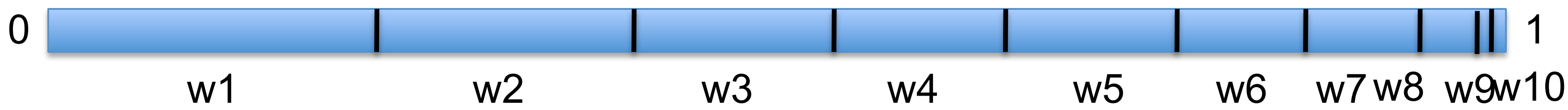
  Resampling: duplicate high-weight particles and abandon low-weight particles

Standard Particle filter

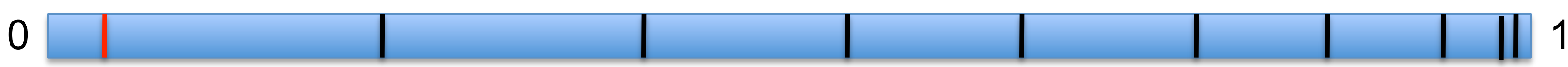


A simple resampling scheme

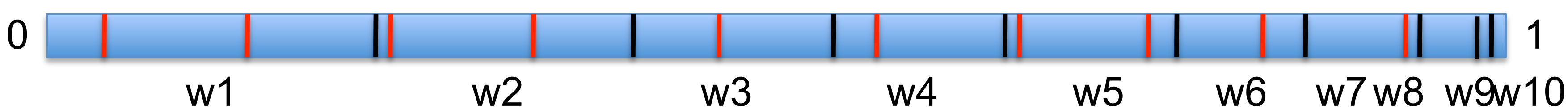
1. Put all weights after each other on the unit interval:



2. Draw a random number from the uniform distribution over [0,1/N], in this case with 10 members over [0,1/10].
3. Put that number on the unit interval: this points to the first member



4. Add 1/N to the end point: the new end point is our second member. Repeat this until N new members are obtained.



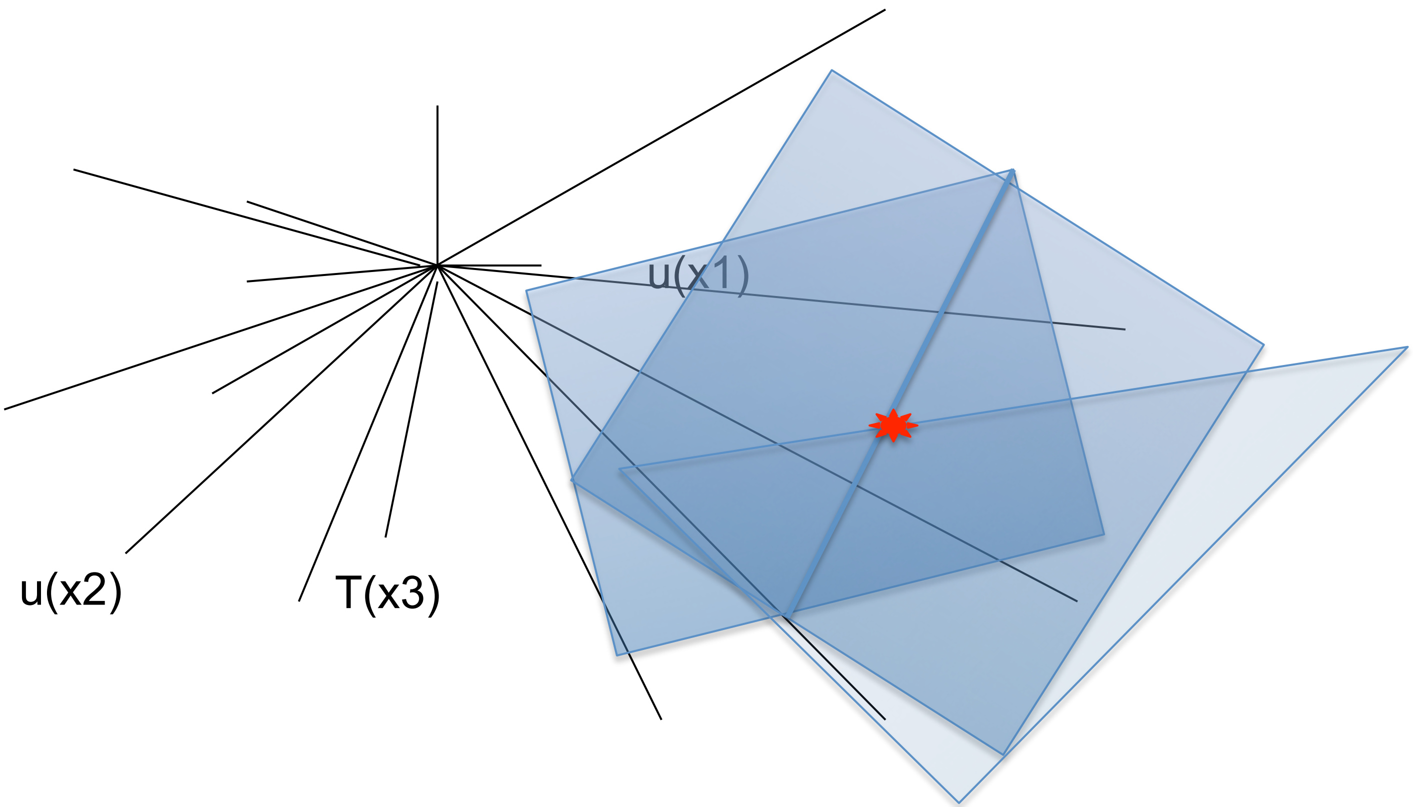
5. In our example we choose m1 2 times, m2 2 times, m3, m4, m5 2 times, m6 and m7.

Resampling is not enough…

- When the umber of observations is large the particle filter with resampling is still degenerate…
- Why?

A closer look at the weights I

Probability space in large-dimensional systems is 'empty': the curse of dimensionality



A closer look at the weights II

Assume particle 1 is at 0.1 standard deviations *s* of M independent observations.
Assume particle 2 is at 0.2 *s* of the M observations.

The weight of particle 1 will be

$$w_1 \propto \exp\left[-\frac{1}{2}\ (y - H(x_i))\, R^{-1}\, (y - H(x_i))\right] = exp(-0.005M)$$

and particle 2 gives

$$w_2 \propto \exp\left[-\frac{1}{2}\ (y - H(x_i))\, R^{-1}\, (y - H(x_i))\right] = exp(-0.02M)$$

A closer look at the weights III

The ratio of the weights is

$$\frac{w_2}{w_1} = exp(-0.015M)$$

Take M=1000 to find

$$\boxed{\frac{w_2}{w_1} = exp(-15) \approx 3\ 10^{-7}}$$

Conclusion: the number of independent observations is responsible for the degeneracy in particle filters.

How to make particle filters useful?

1. Introduce localisation to reduce the number of observations.

2. Use proposal-density freedom.

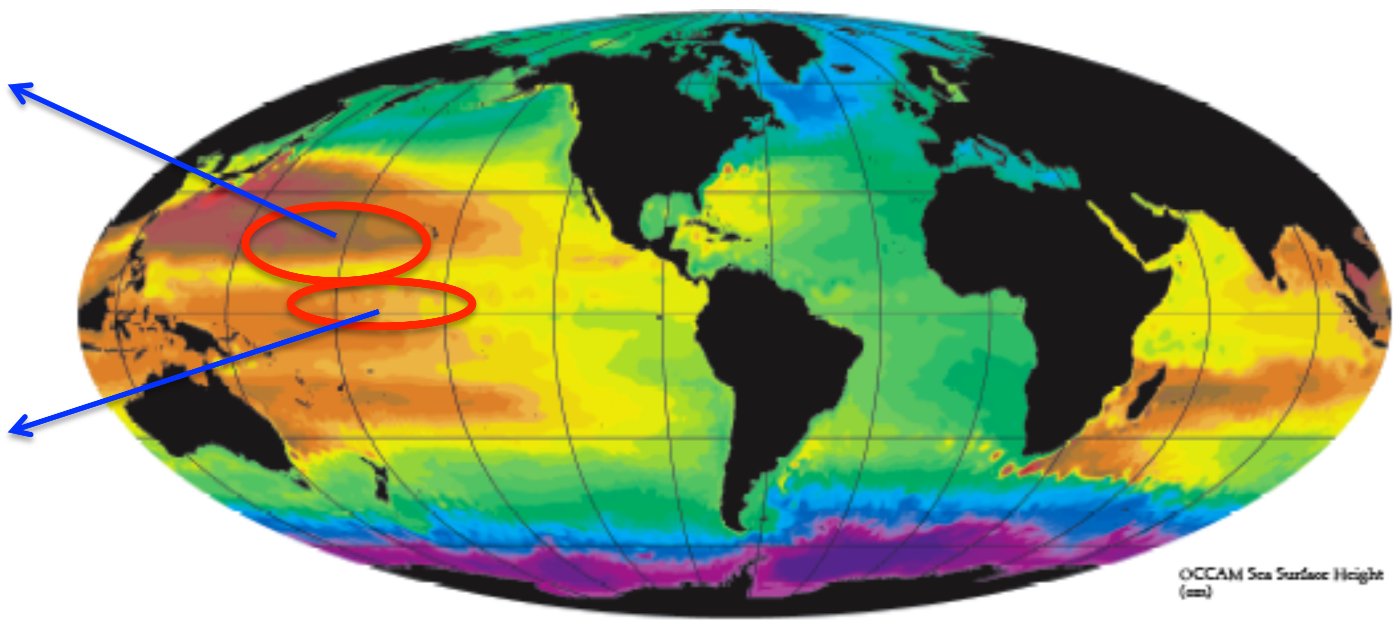
3. Transportation Particle Filters

4. Several ad-hoc combinations of Particle Filters and Ensemble Kalman Filters (not discussed here).

1. Localisation in particle filters

- Easy to make weights spatially varying, similar to observation-space localisation in ETKF.
- Main issue is at the resampling step: how to combine particles from different areas in the domain.

w_1 is high
w_5 is low



w_1 is low
w_5 is high

- Examples are the Localized Particle Filter (Poterjoy, 2016) and the Ensemble Transform Particle Filter (ETPF, Reich, 2014).

Localized Particle Filter (Poterjoy, 2016)

- For each observation *k* do:

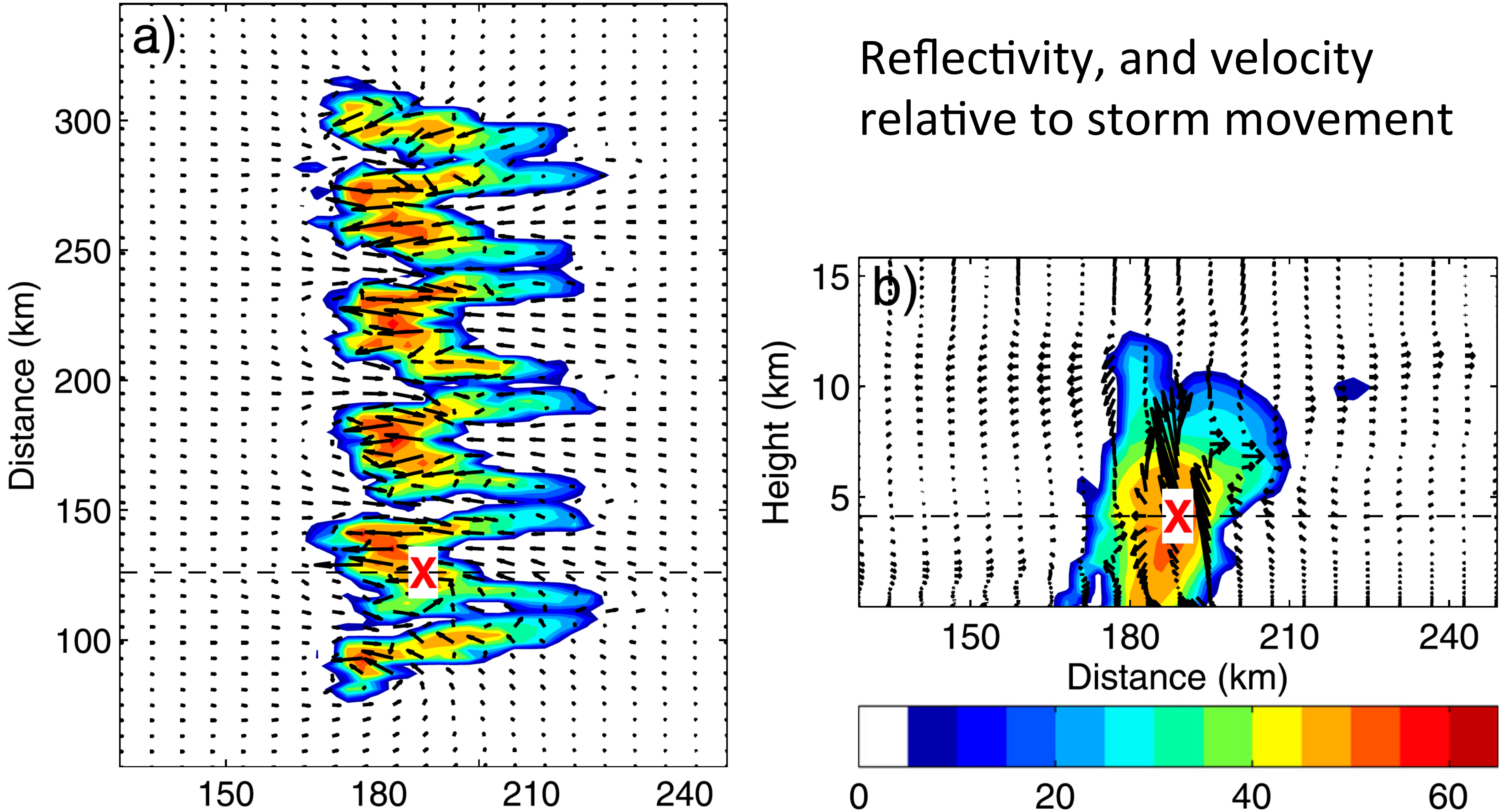
1. Calculate weights $w_i \propto (1 - w_{min})\, p(y_k | x_i^{(k-1)}) + w_{min}$
2. Resample particles globally
3. For each grid point *j* do:
   1. Calculate localized weights:
      $$w_{i,j}^{(k)} \propto w_{i,j}^{(k-1)} (1 - w_{min}(k,j))\, p(y_k | x_i^{(k-1)}) + w_{min}(k,j)$$
   2. Calculate weighted mean x_m and variance
   3. Calculate new particles at grid point j
      $$x_{i,j}^{(k)} = x_m + r_1 \left(x_{i,j}^{(k)}(global) - x_m \right) + r_2 \left(x_{i,j}^{(k-1)}(local) - x_m \right)$$

- Pdf mapping for higher order moments

Example convective storm

(Poterjoy, Sobash, Anderson, MWR, 2017)

Reflectivity, and velocity relative to storm movement



Horizontal cross section

Vertical cross section

Data assimilation set up

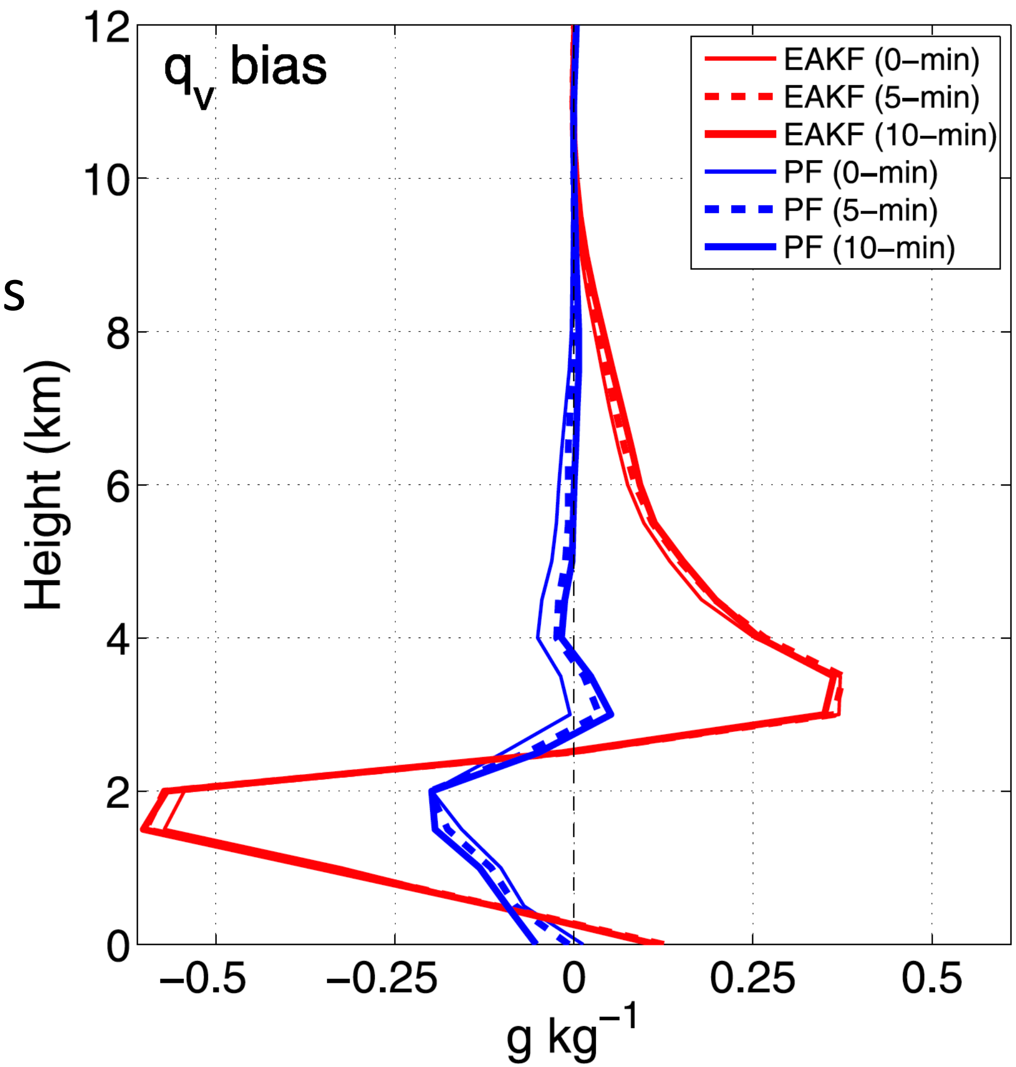
Observations:
Reflectivity and radial velocity from radar in centre of domain with 14 scan elevations between 0.5 and 19.5 degrees, every 5 min. Observation errors 2m/s and 2dBZ, assumed independent.

DA methods:
Ensemble Adjustment Kalman Filter (EAKF), 100 members,
localisation radius 11.46km, adaptive multiplicative inflation.
Localized Particle Filter (PF), 100 members,
localisation radius 11.46 km, additive inflation 0.25 m/s and 0.25 K

DA experiment:
Experiments run 3 hours, after formation of squall line.

Horizontally averaged q_v bias

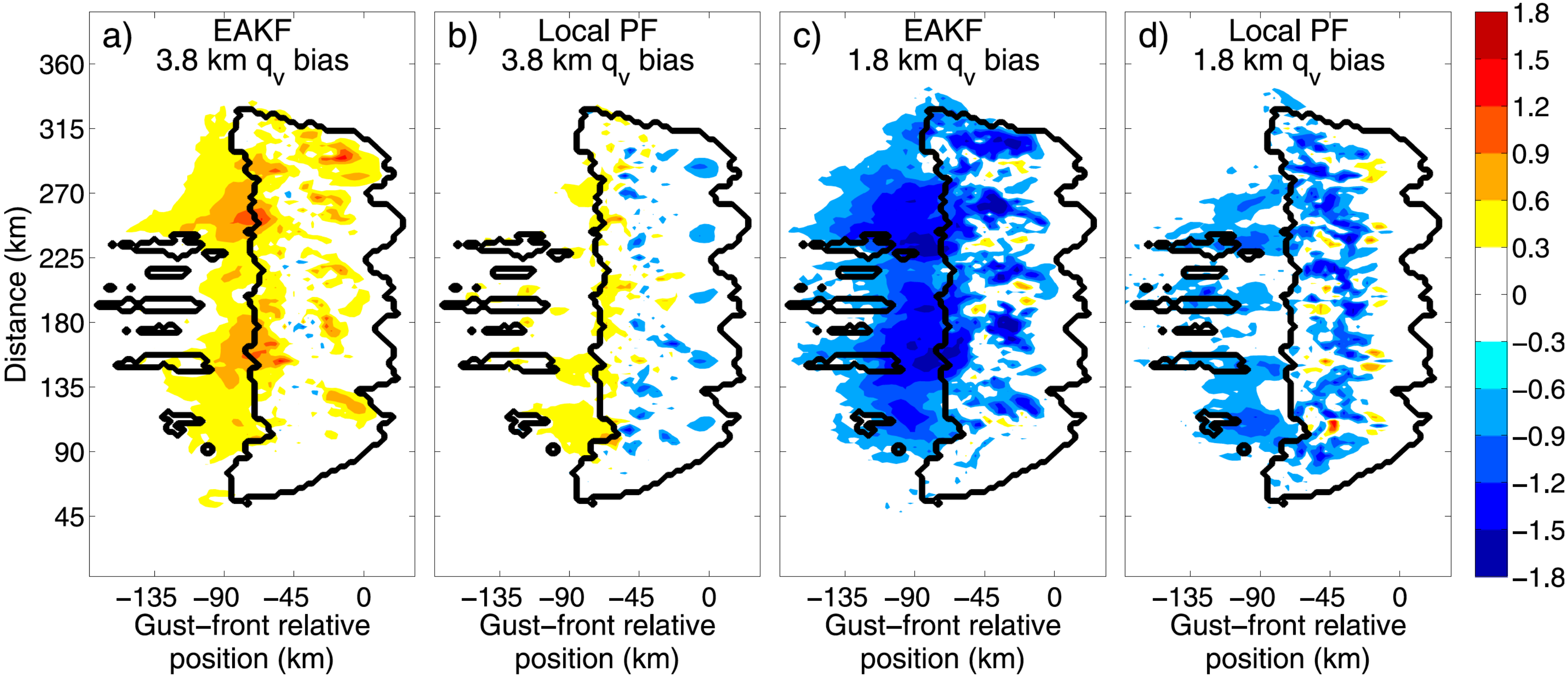
q_v bias at analysis (thin), after 5 min (dashed), and after 10 min (thick)forecasts for EAKF and LPF.



Note that

1) LPF has much less bias
2) Bias is constant over forecast window.

Time averaged q_v bias



3.8 km high, freezing level

1.8 km high

A major issue…

With localisation we can reduce the number of independent observations that each grid point sees.
However, a rough estimate tells us that the standard deviation of the weights is

$$\sigma_{w_i} \approx \exp[N_y]$$

Hence a modest 10 observations will give a typical difference in the weights of about 22026,
so the filter will be degenerate even with localisation.
(This is a prediction…)

Ensemble Transform Particle Filter ETPF

- Find a linear map between prior and posterior ensemble:

$$x_j^a = \frac{1}{N}\sum_{i=1}^{N} x_i^f t_{ij} + \xi_j$$

with $\sum_{i=1}^{N} t_{ij} = \frac{1}{N}$ and $\sum_{j=1}^{N} t_{ij} = w_i$

- Infinite number of solutions for t_{ij} . ETPF uses optimal transportation by minimising

$$J(t) = \sum_{i=1}^{N} t_{ij} ||x_i^f - x_j^f||$$

The ETPF

- Minimisation takes $O(N^2 \log N)$ operations. Minimisation performed at every gridpoint, like the ETKF, so expensive algorithm.

- Possibility to reduce this to larger areas.

- The random perturbation acts as inflation.

- Localisation has same problem as in ETKF that large-scale balances are broken.

- Needs further exploration!

3. Exploring the proposal density freedom

Model: $$x^n = f(x^{n-1}) + \beta^n$$

with stochastic model error $$\beta^n \sim N(0, Q)$$

Observations: $$y^n = H(x^n_{true}) + \epsilon^n_{true}$$

with for Gaussian obs errors $$\epsilon^n_{true} \sim N(0, R)$$

The transition density

The joint-in-time prior pdf can be written as:

$$p(x^n, x^{n-1}) = p(x^n|x^{n-1})p(x^{n-1})$$

So the marginal prior pdf at time *n* becomes:

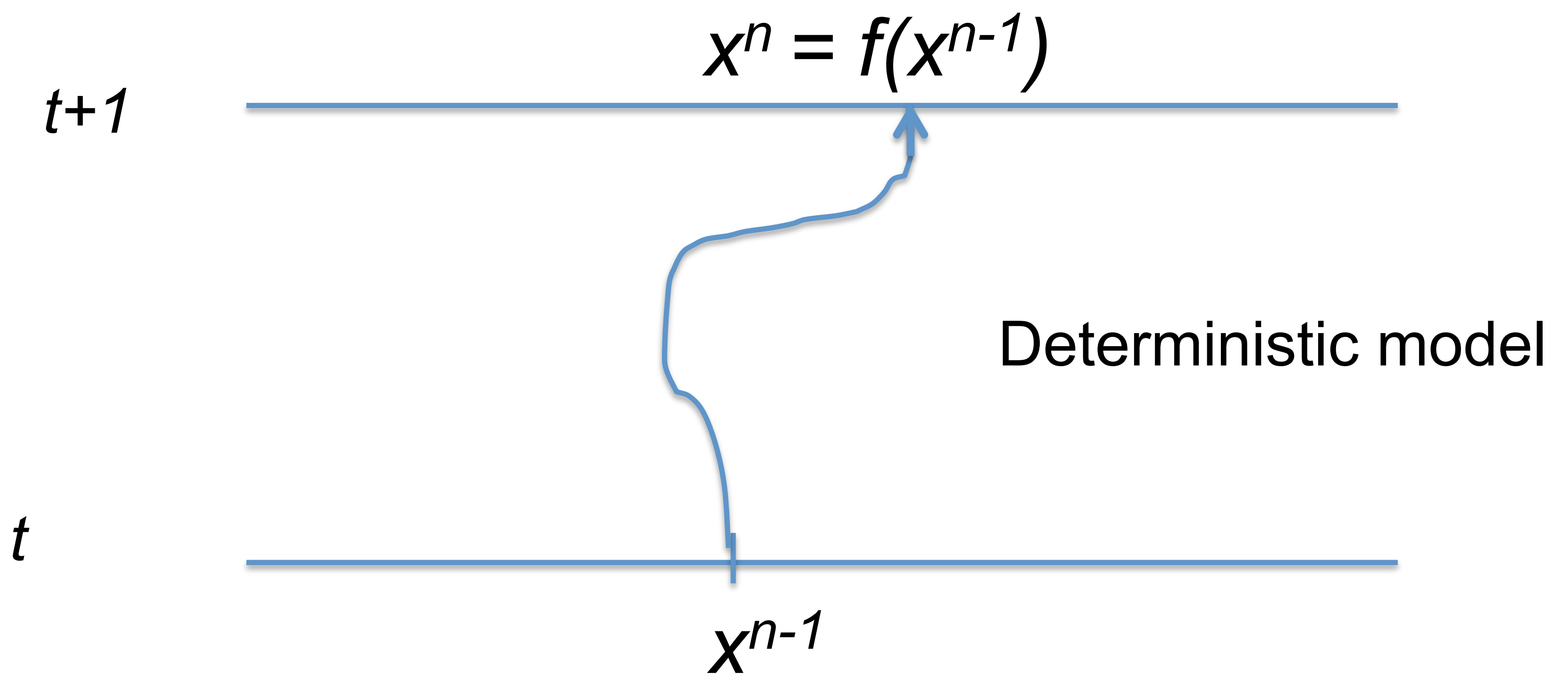
$$p(x^n) = \int p(x^n|x^{n-1})p(x^{n-1})\; dx^{n-1}$$

We introduced the **transition densities**

$$p(x^n|x^{n-1})$$

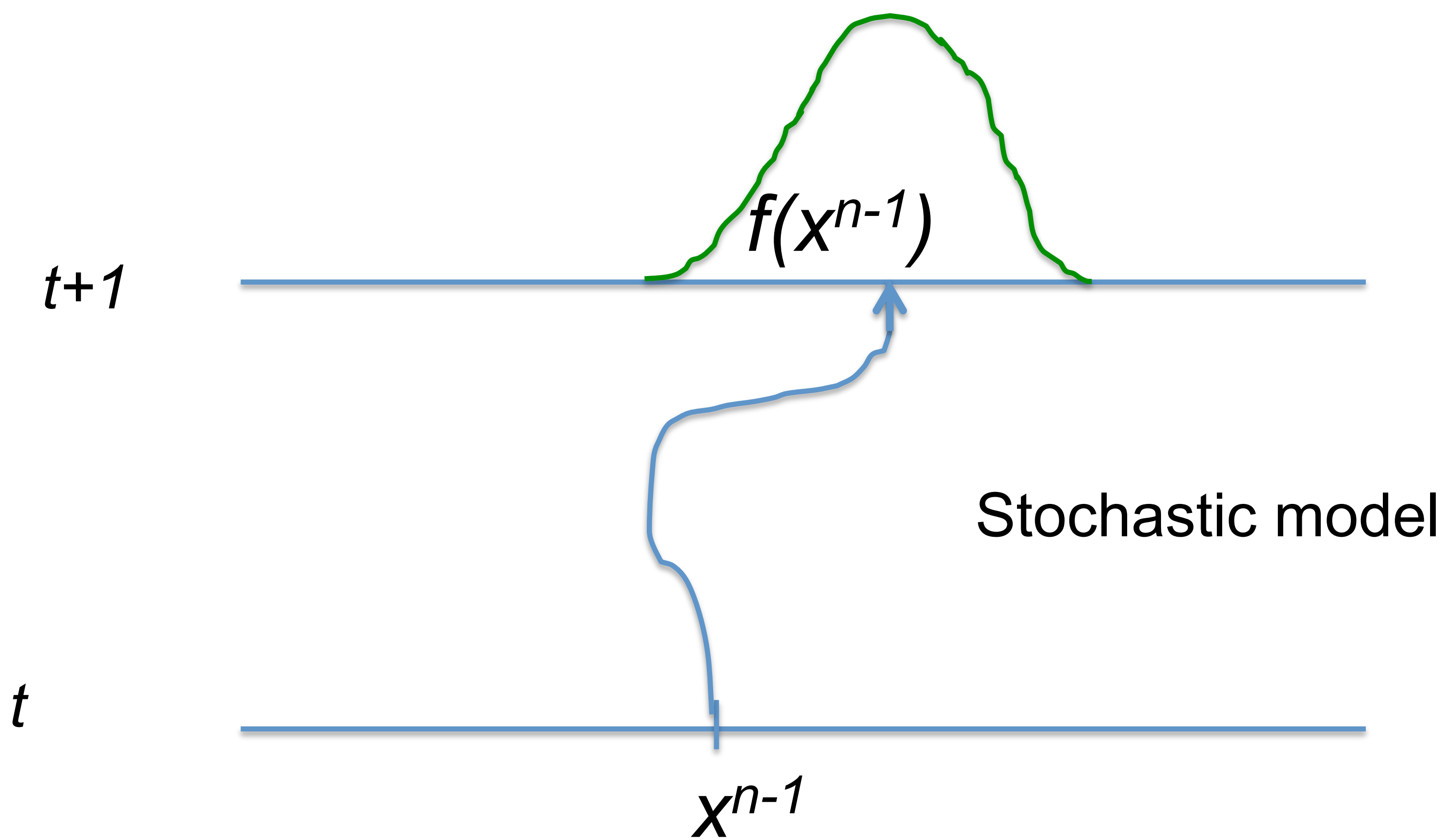
Transition densities $p(x^n|x^{n-1})$

$$p(x^n|x^{n-1}) = \delta(x^n - f(x^{n-1}))$$



Transition densities $p(x^n|x^{n-1})$

$$p(x^n|x^{n-1}) = N\left(f(x^{n-1}), Q\right)$$



Bayes Theorem and the proposal density

Bayes Theorem now becomes:

$$
\begin{aligned}
p(x^n|y^n) &= \frac{p(y^n|x^n)p(x^n)}{p(y)} \\
&= \frac{p(y^n|x^n)}{p(y)} \int p(x^n|x^{n-1})p(x^{n-1})\, dx^{n-1}
\end{aligned}
$$

We have a set of particles at time *n-1* so we can write

$$
p(x^{n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(x^{n-1} - x_i^{n-1})
$$

and use this in the equation above to perform the integral:

The transition density

Performing the integral over the sum of delta functions gives:

$$p(x^n|y^n) = \frac{p(y^n|x^n)}{p(y^n)} \frac{1}{N} \sum_{i=1}^{N} p(x^n|x_i^{n-1})$$

The posterior is now given as a sum of transition densities. In the standard particle filter we use these to draw particles at time *n*, which, remember, is running the stochastic model from time *n-1* to time *n*. We know that is degenerate.

So we introduce another transition density, the proposal.

The magic: proposal transition density

Multiply numerator and denominator with a proposal density *q*:

$$p(x^n|y^n) = \frac{p(y^n|x^n)}{p(y^n)} \frac{1}{N} \sum_{i=1}^{N} \frac{p(x^n|x_i^{n-1})}{q(x^n|x_{1:N}^{n-1}, y^n)} q(x^n|x_{1:N}^{n-1}, y^n)$$

Note that
1) the proposal depends on the future observation, and
2) the proposal can depend on all previous particles, not just one.

1) Ensures that the particles end up close to the observations because they know where the observations are.
2) Allows for an equal-weight filter, as the performance bounds suggested by Snyder, Bickel, and Bengtsson do not apply.

What does this all mean?

- The standard Particle Filter propagates the original model by drawing from $p(x^n|x^{n-1})$.
- Now we draw from $q(x^n|x_{1:N}^{n-1}, y^n)$, *so we propagate the state using a different model.*
- This model can be anything, e.g.

$$x^n = g(x^{n-1}, y^n) + \hat{\beta}^n$$

Examples of proposal transition densities

The proposal transition density is related to a proposed model.

For instance, add a relaxation term and change random forcing:

$$x^n = f(x^{n-1}) + \hat{\beta}^{n-1} + K\left(y^n - H(x^{n-1})\right)$$

Or, run a 4D-Var on each particle (implicit particle filter).
This is a special 4D-Var:
- initial condition is fixed
- model error essential
- needs extra random forcing

Or use the EnKF as proposal density.

How are the weights affected?

Draw samples from the proposal transition density *q*, to find:

$$p(x^n|y^n) = \frac{p(y^n|x_i^n)}{p(y^n)} \frac{1}{N} \sum_{i=1}^{N} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_{1:N}^{n-1}, y^n)} \delta(x^n - x_i^n)$$

which can be rewritten as: $$p(x^n|y^n) = \sum_{i=1}^{N} w_i \delta(x^n - x_i^n)$$

with weights $$w_i = \frac{p(y^n|x_i^n)}{Np(y^n)} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_{1:N}^{n-1}, y^n)}$$

Likelihood weight

Proposal weight

Algorithm

1. Generate initial set of particles

2. Run proposed model conditioned on next observation

3. Accumulate proposal density weights *p/q*

4. Calculate likelihood weights

5. Calculate full weights and resample

Note, the original model is never used directly.

How to calculate *p/q* in the weights?

Let's assume that the original model has Gaussian distributed model errors:

$$p(x^n|x^{n-1}) = N\left(f(x^{n-1}), Q\right)$$

To calculate the value of this term realise it is the probability of moving from x_i^{n-1} *to* x_i^n. Since x_i^n and x_i^{n-1} are known from the proposed model we can calculate directly:

$$p(x_i^n|x_i^{n-1}) \propto \exp\left[-\frac{1}{2}\left(x_i^n - f(x_i^{n-1})\right)^T Q^{-1}\left(x_i^n - f(x_i^{n-1})\right)\right]$$

Example calculation of *p*

- Assume the proposed model is

$$x^n = f(x^{n-1}) + \hat{\beta}^n + K\left(y^n - H(x^{n-1})\right)$$

- Then we find

$$p(x_i^n|x_i^{n-1}) \propto$$
$$\propto \exp\left[-\frac{1}{2}\left(K(y^n - H(x_i^{n-1})) + \beta_i^n\right)^T Q^{-1}\left(K(y^n - H(x_i^{n-1})) + \beta_i^n\right)\right]$$

- We know all the terms, so this can be calculated

And *q* …

- The deterministic part of the proposed model is:

$$x^n = f(x^{n-1}) + K\left(y^n - H(x^{n-1})\right)$$

- So the probability becomes

$$q(x^n | x^{n-1}_{1:N}, y^n) \propto \exp\left[-\frac{1}{2}\hat{\beta}_i^{n\ T}\hat{Q}^{-1}\hat{\beta}_i^n\right]$$

- We did draw the stochastic terms, so we know what they are, so this term can be calculated too.

The weights

- We can calculate *p/q* and we can calculate the likelihood so we can calculate the weights:

$$w_i = \frac{p(y^n|x_i^n)}{Np(y^n)} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_{1:N}^{n-1}, y^n)}$$

Example: EnKF as proposal

Model forecast to observation time: $x_i^* = f(x_i^{n-1}) + \beta_i^n$

EnKF update: $x_i^n = x_i^* + K(y^n - H(x_i^*) - \epsilon_i)$

Use model equation:

$$x_i^n = f(x_i^{n-1}) + \beta_i^n + K(y^n - H\left(f(x_i^{n-1}) + \beta_i^n\right) - \epsilon_i)$$

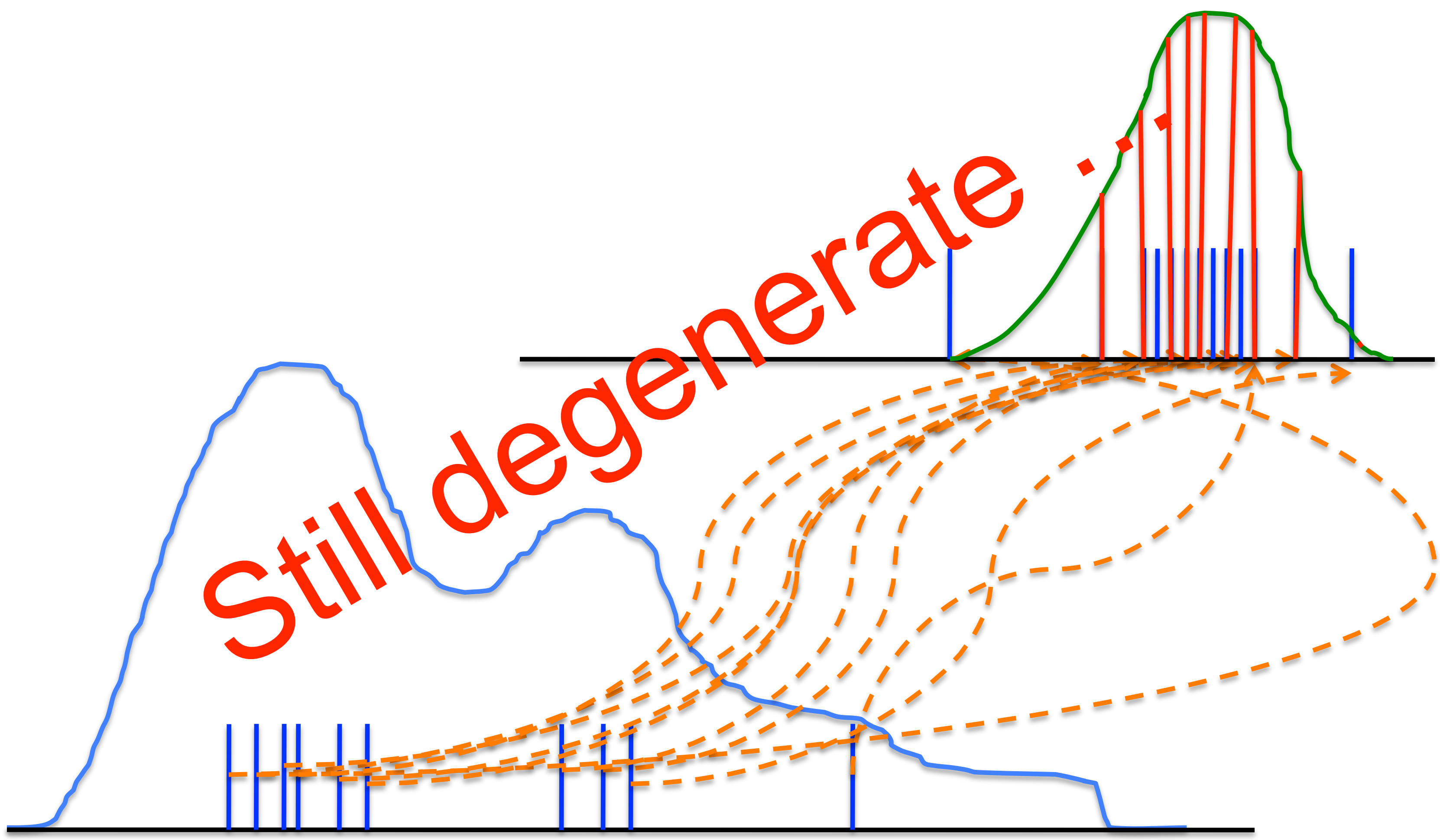
Regroup terms:

$$x_i^n = f(x_i^{n-1}) + K\left(y^n - H\left(f(x_i^{n-1})\right)\right) + (1 - KH)\beta_i^n - K\epsilon_i)$$

Leading to:

$$x_i^n = g(x_i^{n-1}, y^n) + \hat{\beta}_i^n$$

Particle filter with 'usual' proposal transition density



Proposal density freedom

Given particles at time *n-1* the posterior pdf can be written:

$$p(x^n|y^n) = \frac{1}{N}\sum_{i=1}^{N} \frac{p(y^n|x_i^{n-1})}{p(y^n)} \frac{p(x^n|x_i^{n-1},y^n)}{q(x^n|y^n,..)} q(x^n|y^n,..)$$

Consider the pair of random variables (I, X^n) and

$$W = w_i(x^n) = \frac{1}{N} \frac{p(y^n|x_i^{n-1})}{p(y^n)} \frac{p(x^n|x_i^{n-1},y^n)}{q(x^n|y^n,..)}$$

The variance in the weights can be written:

$$Var(W) = Var_I(E_X(W|I)) + E_I(Var_X(W|I))$$

Optimal proposal density

A standard choice is to assume

$$q(x^n|y^n,..) = q(x^n|x_i^{n-1},y^n)$$

One also chooses $$Prob(I=i) = \frac{1}{N}$$

Minimal variance in the weights is achieved by the *optimal proposal*:

$$q(x^n|x_i^{n-1},y^n) = p(x^n|x_i^{n-1},y^n)$$

The variance of the weights is

$$Var(W) = \frac{1}{N^3}\sum_{i=1}^{N}\left(\frac{p(y^n|x_i^{n-1})^2}{p(y^n)^2} - 1\right)$$

Degenerate for large number of independent observatoins.

Better than optimal: example 1

Again write posterior as:

$$p(x^n|y^n) = \frac{1}{N}\sum_{i=1}^{N} \frac{p(y^n|x_i^{n-1})}{p(y^n)} p(x^n|x_i^{n-1}, y^n)$$

See posterior expression as *mixture density* and draw from complete mixture: each particle has same weight by construction.

So we choose

$$Prob(I = i) = \frac{1}{N}\frac{p(y^n|x_i^{n-1})}{p(y^n)}$$

and $$x_i^n \sim p(x^n|x_i^{n-1}, y^n)$$

We now find $Var(W) = 0$, so 'optimal proposal density' not optimal! But when number of independent observations is large we sample from just one mixture density…

An even better proposal:

More general proposals are possible, specifically multi-step proposals:

$$p(x^n|y^n) = \frac{1}{N}\sum_{i=1}^{N} \frac{p(y^n|x_i^{n-1})}{p(y^n)} \frac{p(x^n|x_i^{n-1},y^n)}{q(x^n x^*|x_{i;1:N}^{n-1},y^n)} q(x^n x^*|x_{i;1:N}^{n-1},y^n)$$

where we just multiplied and divided by a proposal *q(…)* which can depend on *all* previous particles, and with

$$q(x^n x^*|x_{i;1:N}^{n-1},y^n) = q(x^n|x^*,x_{1:N}^{n-1})q(x^*|x_i^{n-1},y^n)$$

This leads to a whole class of particle filters not hampered by classical proofs of degeneracy.

Example

The following particle filter results in equal weights but is also efficient for small ensemble sizes.

1. For each *i* draw $x_i^* \sim p(x^n | x_i^{n-1}, y^n)$
2. For each *i* draw $\xi_i \sim N(0, P)$ with $P^{-1} = Q^{-1} + H^T R^{-1} H$
3. For each *i* write $x_i^n = x_i^* + \alpha_i P^{1/2} \xi_i$

And solve for α_i in

$$w_i(\alpha_i) = \frac{p(y|x_i^n) p(x_i^n | x_i^{n-1})}{q(x_i^n x_i^* | x_{i;1:N}^{n-1}, y^n)} = w_{target}$$

Variance of the weights in these filters

Instead of seeing the weights as a function of the index I and the position of the particle in state space x_i^n, so

$$W(I, X^n)$$

These filters try to find the position of the particles in state space x_i^n given that the weight is equal to the target weight w_{target}, so:

$$X^n(I, w_{target})$$

Hence we have turned the problem around, ensuring equal weights!

Note that the full mathematical justification for this is still missing.

Experiments on Lorenz 1963 model

- 10,000 independent Lorenz 1963 models
- 30,000 variables, 10,000 σ parameters
- 10 particles
- Observations:
  every 20 time steps,
  first two variables
- Observation errors Gaussian
- SIR needs 500,000 particles
  for an effective ensemble
  size of about 300 on just one
  of the L63 models…

Sequential parameter estimation

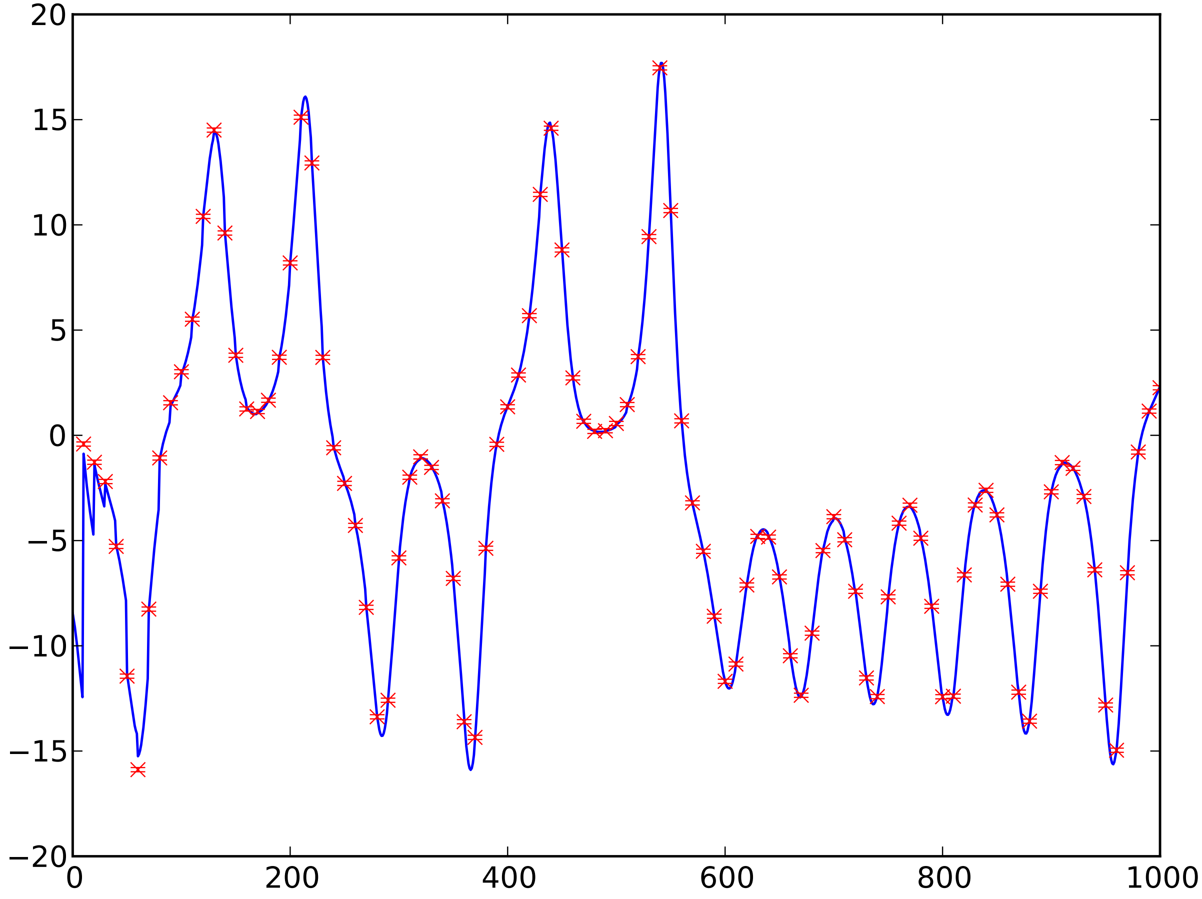
- SPDE $$x^n = f(x^{n-1}, \theta) + \beta^n$$

- Unknown parameter
$$x^n = f(x^{n-1}, \theta_0) + \frac{\partial f}{\partial \theta}(\theta - \theta_0) + \beta^n$$

- Model as $$\theta^n = \theta^{n-1} + \eta^n$$

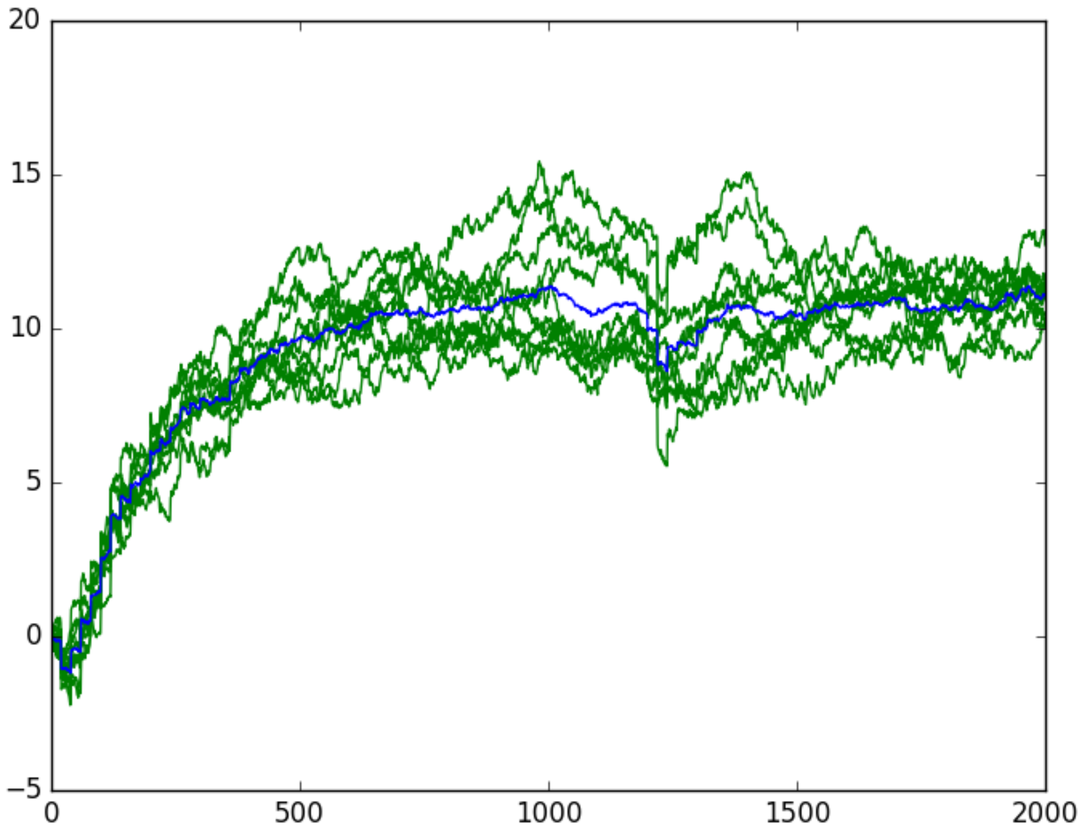
hence model error
$$Q_{xx} = Q_\beta + \frac{\partial f}{\partial \theta} Q_\eta \frac{\partial f}{\partial \theta}^T$$
$$Q_{x\theta} = \frac{\partial f}{\partial \theta} Q_\eta$$
$$Q_{\theta\theta} = Q_\eta$$

40,000 dimensional system (30,000 variables, 10,000 parameters).



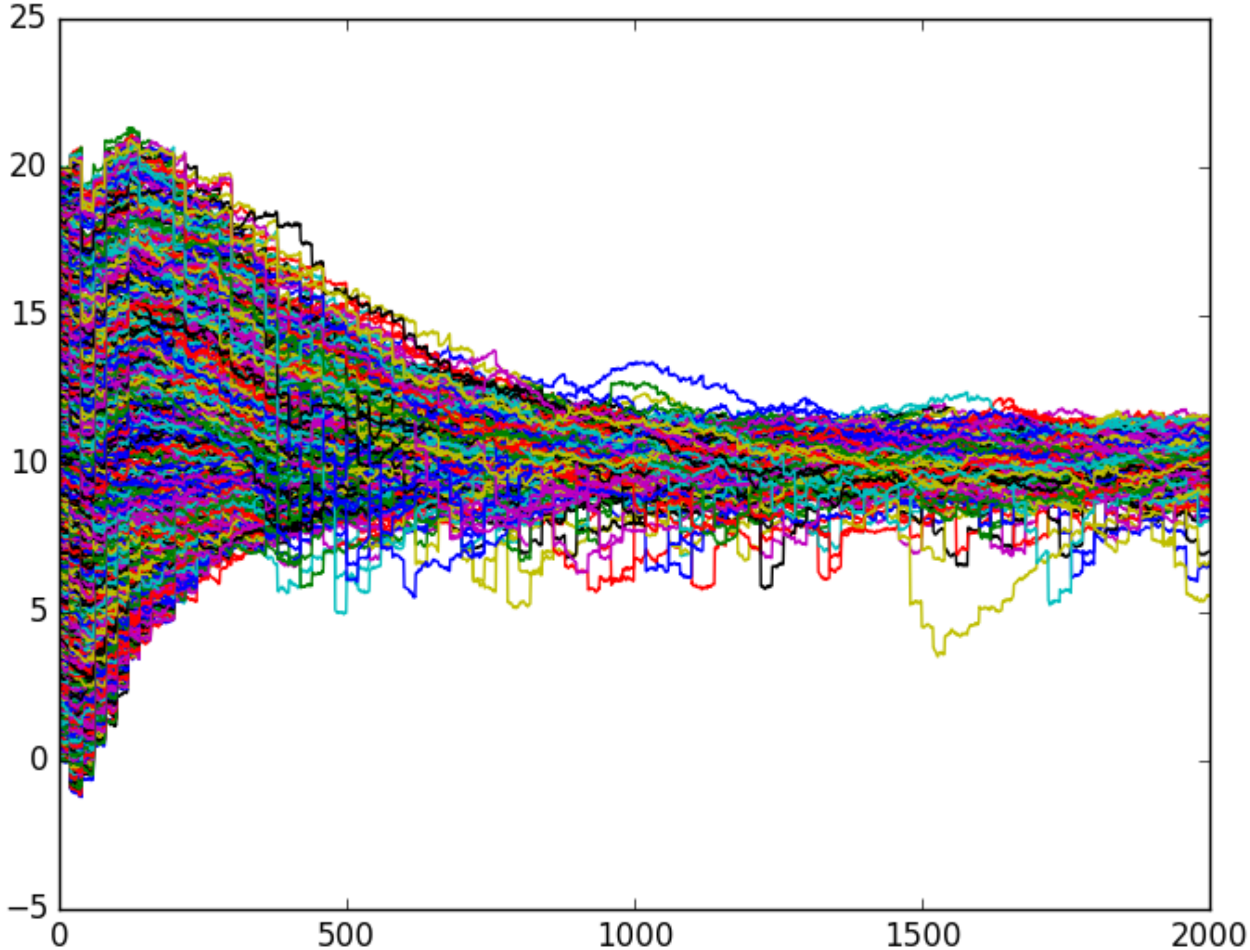
Time evolution mean of first variable system 1, starting 10 lower than true value.

40,000 dimensional system (30,000 variables, 10,000 parameters).



Time evolution mean of parameter system 1, starting 10 lower than true value.

Parameter mean values (dim=10,000)



Time evolution mean values parameter all 10,000 systems

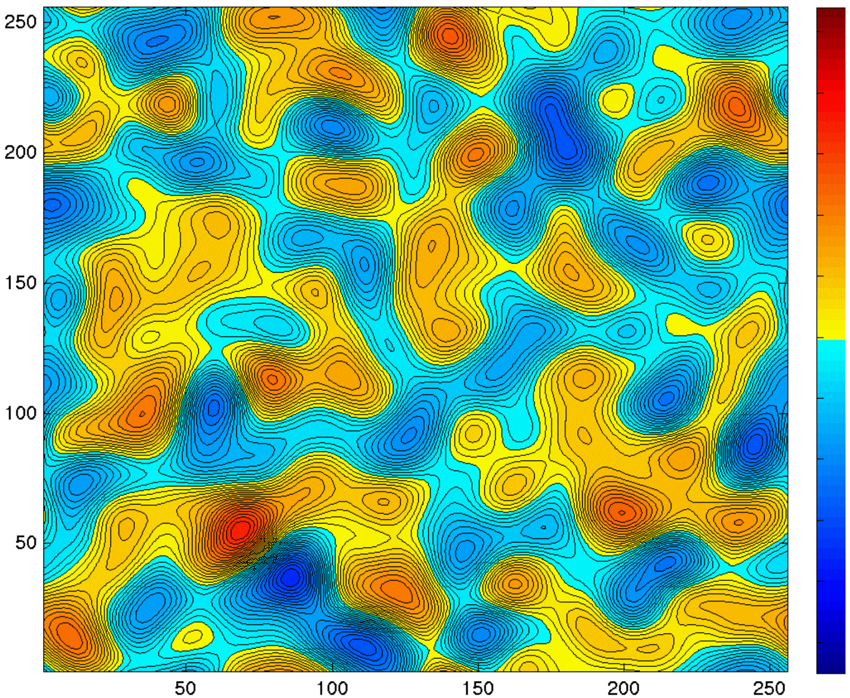
Application: the barotropic vorticity equation

- Stochastic barotropic vorticity equation:

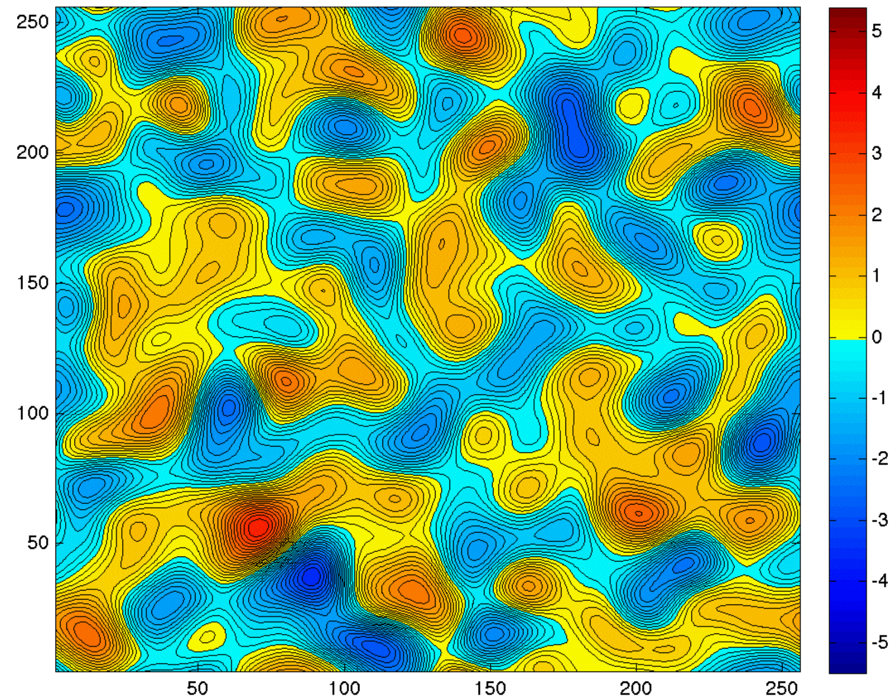
$$\frac{\partial q}{\partial t} + u \cdot \nabla q = F$$

- 256 by 256 grid - 65,536 variables
- Double periodic boundary conditions
- Semi-Langrangian time stepping scheme
- Twin experiments
- Observations every 50 time steps – decorrelation time of 42
- 32 particles
- Nudging plus equivalent-weights scheme

¼ Observations over half of state

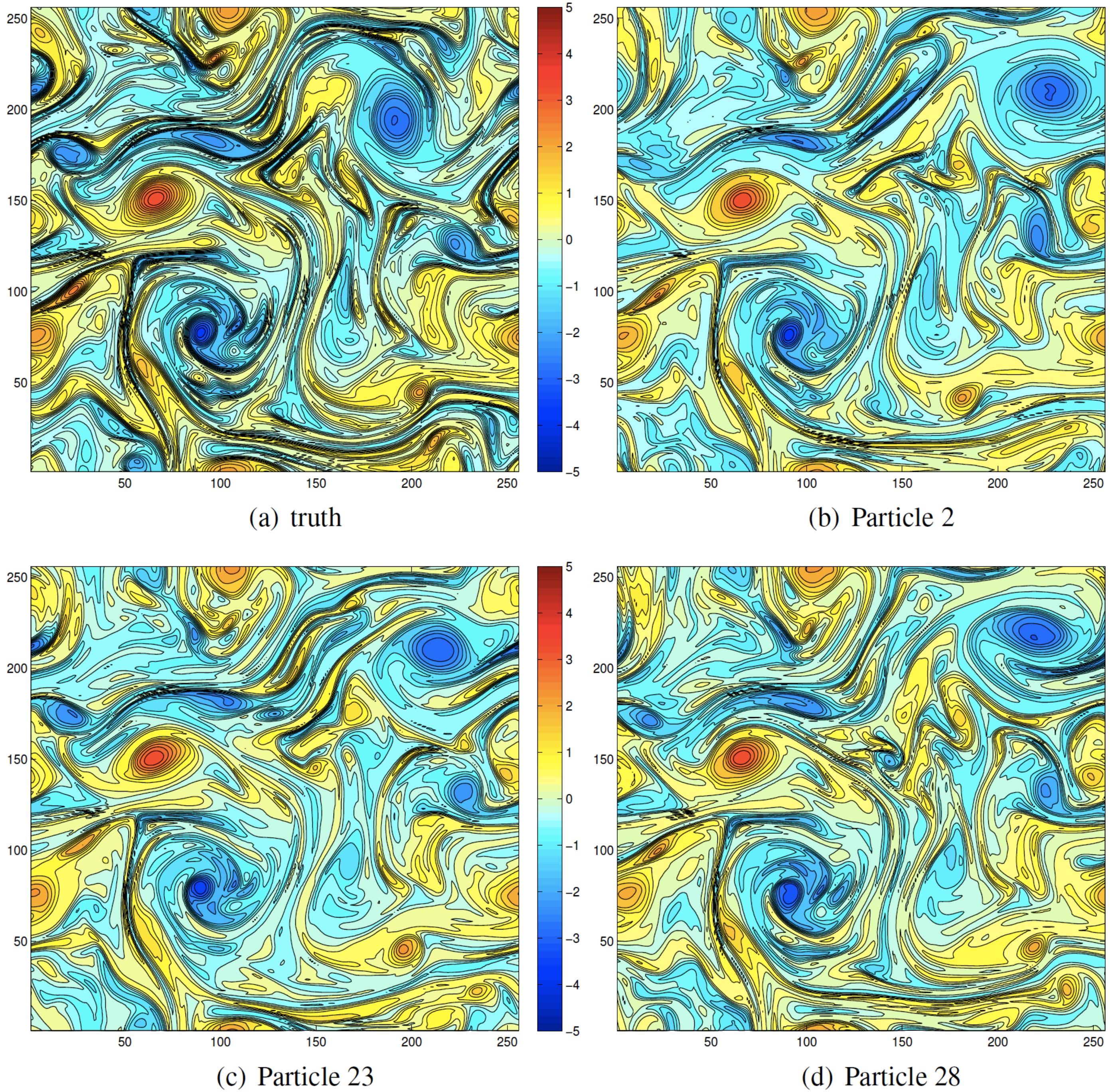


Truth

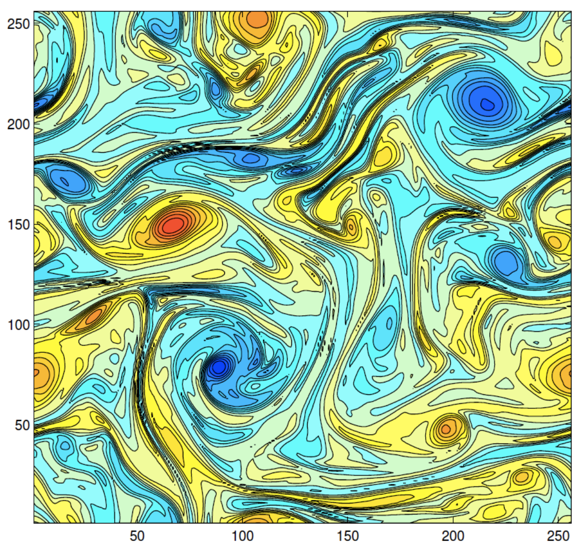


Mean of particle filter ensemble

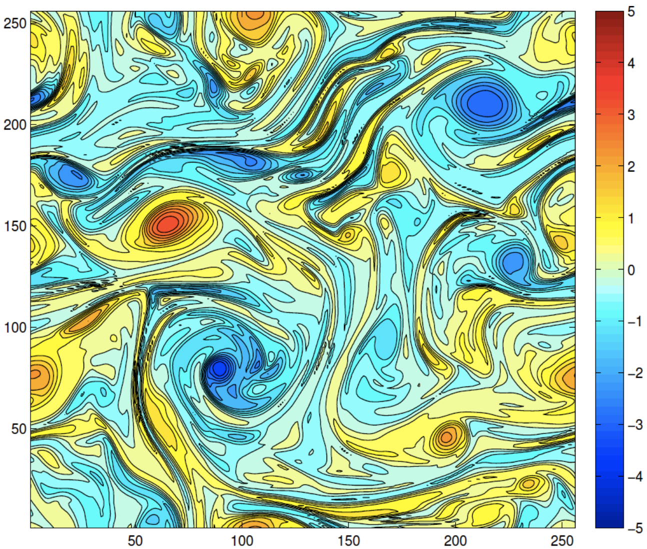
Individual particles are not too smooth.



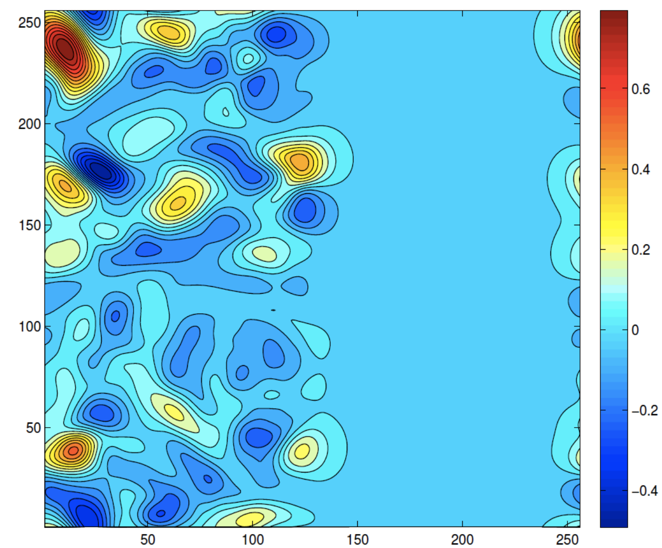
The update of the unobserved part



Particle 23 before update

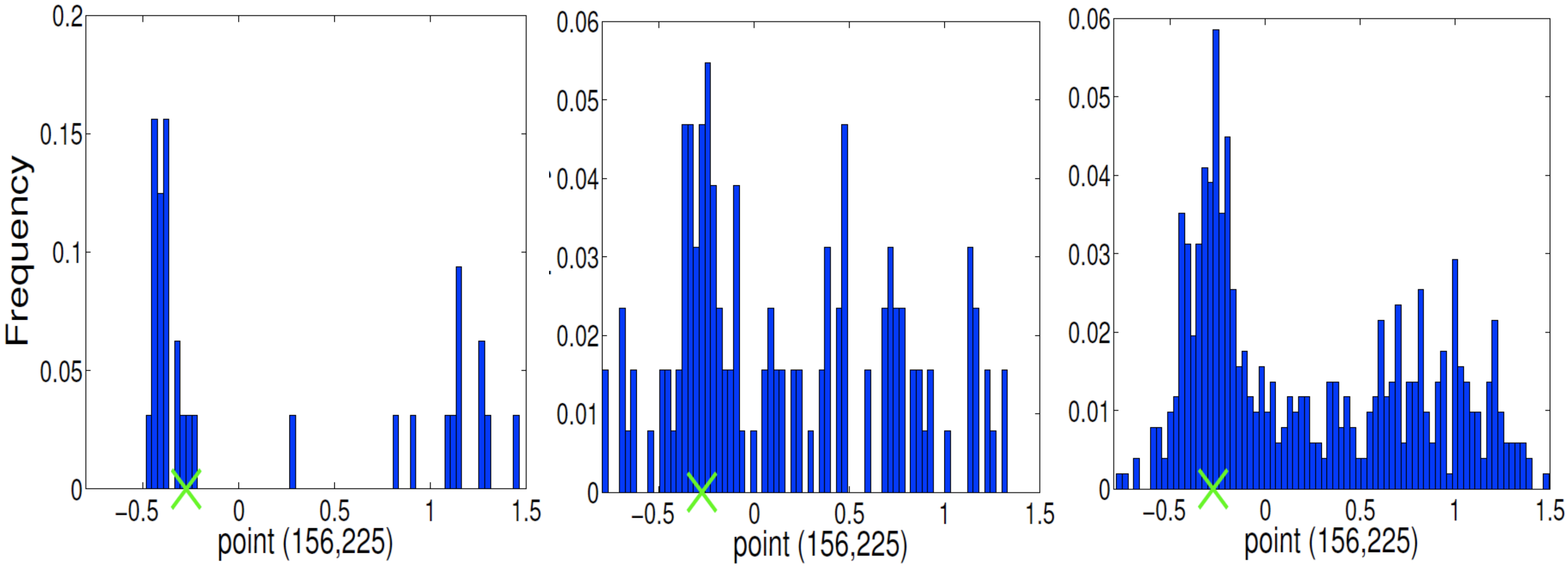


Particle 23 after update



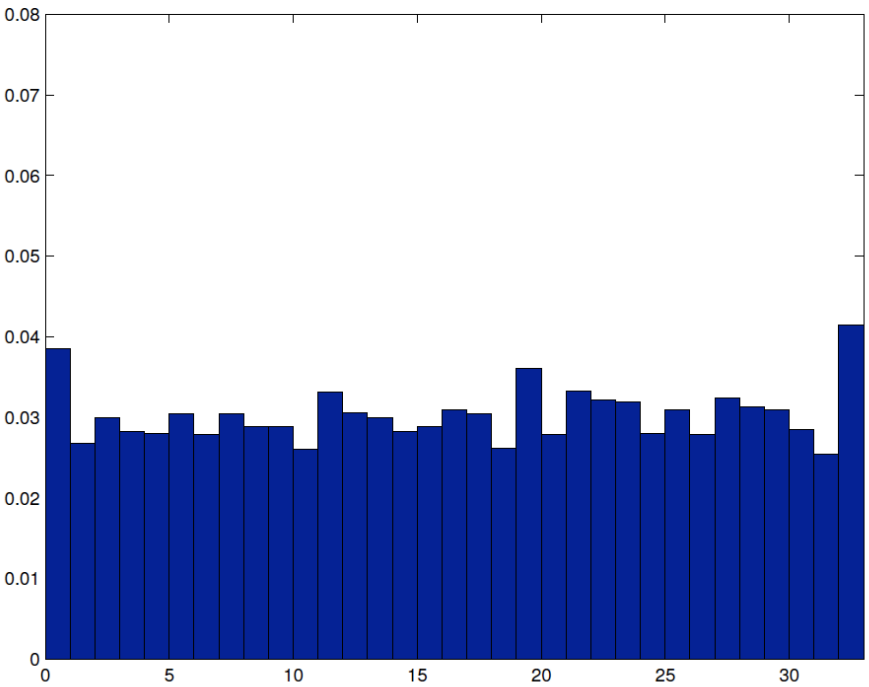
Difference

Convergence of the pdf’s

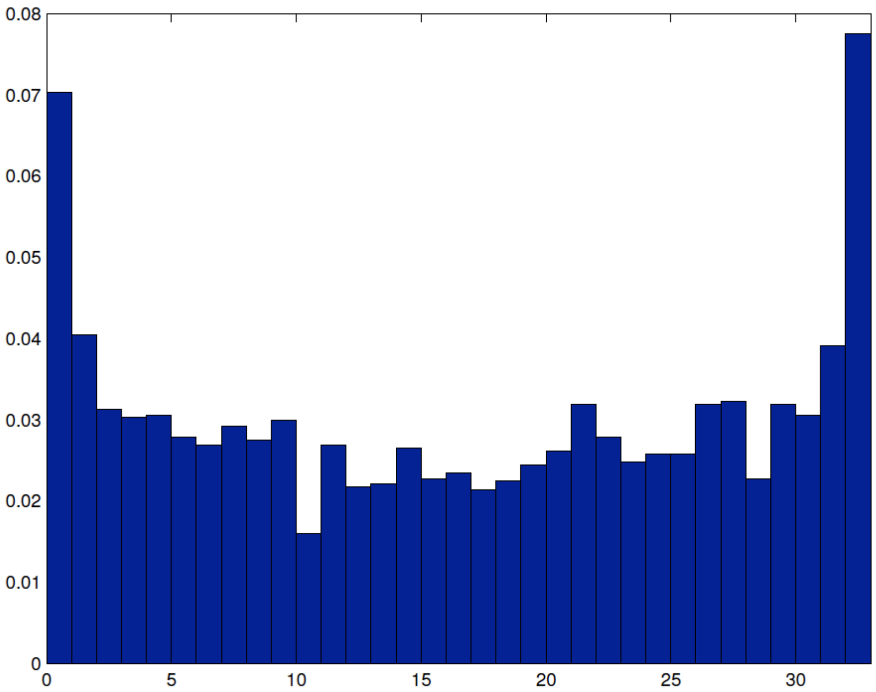


32 particles　　128 particles　　512 particles

Rank histograms



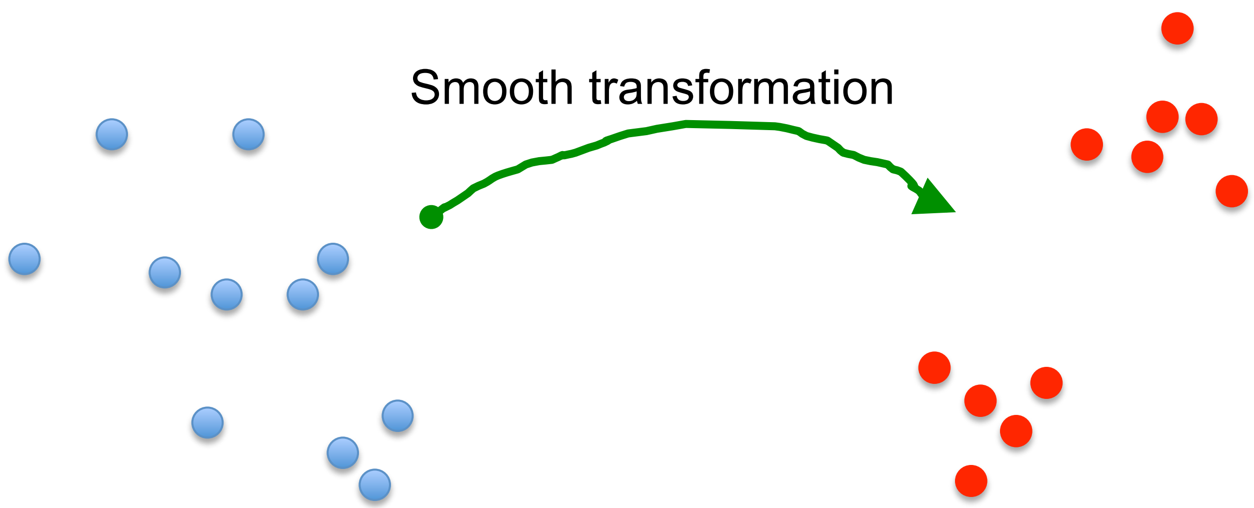
Full state observed



¼ of half state observed

2. Optimal transportation

The prior particles are a sample of the prior pdf, and we want to *transform* that sample into a sample from the posterior pdf:



Estimate of the posterior

In sequential Bayesian Inference we do not know what the posterior looks like. Here we explore the model transition density to find

$$p(x^n|y^n) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(y^n|x_i^{n-1})}{p(y^n)}p(x^n|x_i^{n-1},y^n)$$

This estimate will not be very accurate when

- the number of particles at previous time is small,
- the number of observations is large, so the likelihood is highly peaked in state space.

We will discuss this later.

Minimise Relative Entropy (or Kullback-Leibner Divergence)

Use smooth iterative transport map $z = T(x)$ that minimises K-L divergence:

$$KL = \int q(x^n|y^n) \log\left(\frac{q(x^n|y^n)}{p(x^n|y^n)}\right) dx^n$$

We use

$$T(x) = x - \epsilon\phi(x)$$

leading to the iterative scheme.

$$x_i^{(j)} = x_i^{(j-1)} - \epsilon\nabla_{\phi(x)} KL\left(x_{1:N}^{(j-1)}\right)$$

How should we choose $\phi(x)$?

Use reproducing kernels as basis

Embed $\phi(x)$ in the reproducing Kernel basis:

$$\phi(x) = \langle K(x,.), \phi(.) \rangle_{\mathcal{F}}$$

Leading to

$$\nabla_{\phi(x)} KL(x) = -E_{x' \sim q}\left[K(x',x)\nabla_x \log p(x'|y) + \nabla_x K(x',x)\right]$$

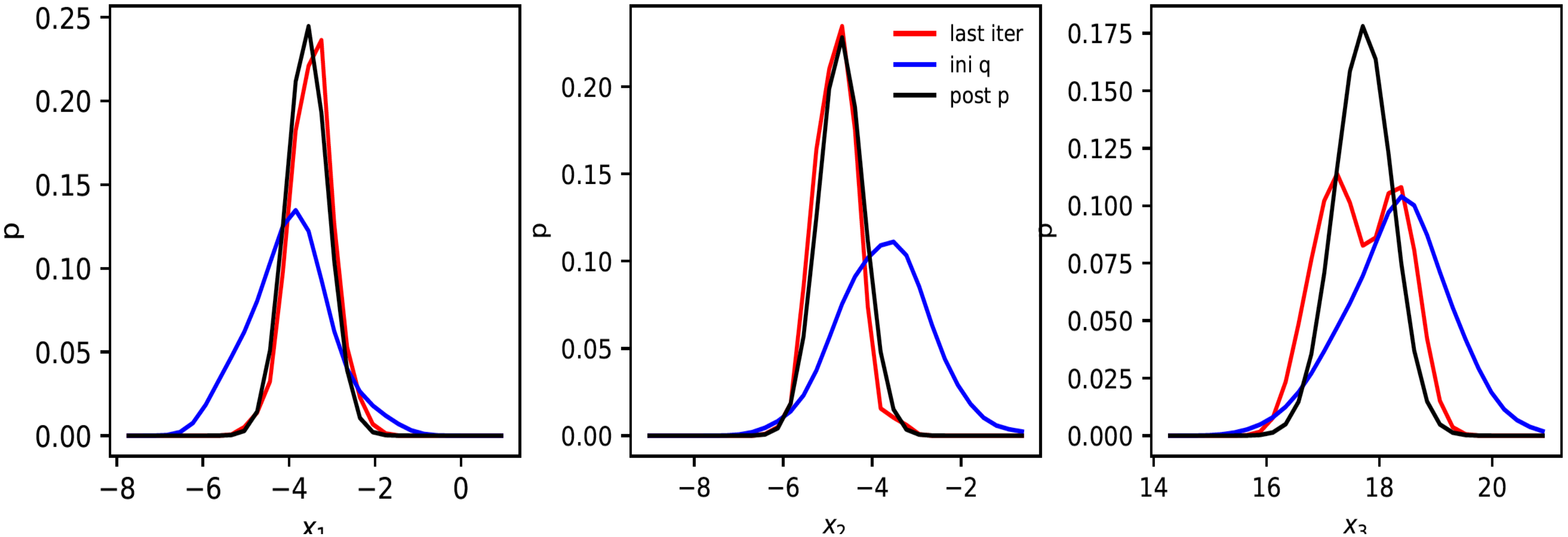
So we can now use

$$x_i^{(j)} = x_i^{(j-1)} - \epsilon \nabla_{\phi(x)} KL\left(x_i^{(j-1)}\right)$$

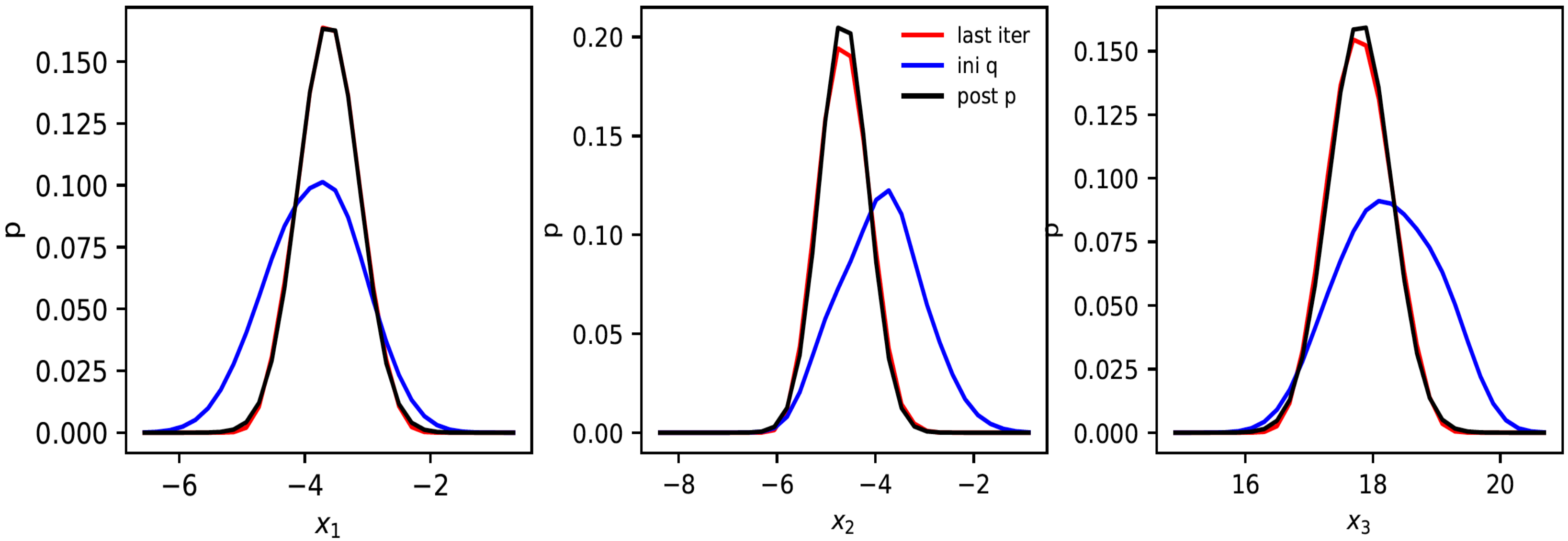
Weights play no role here, and unbiased if $p(x^n|y^n)$ unbiased.

Results on L63 for marginal pdfs 1D

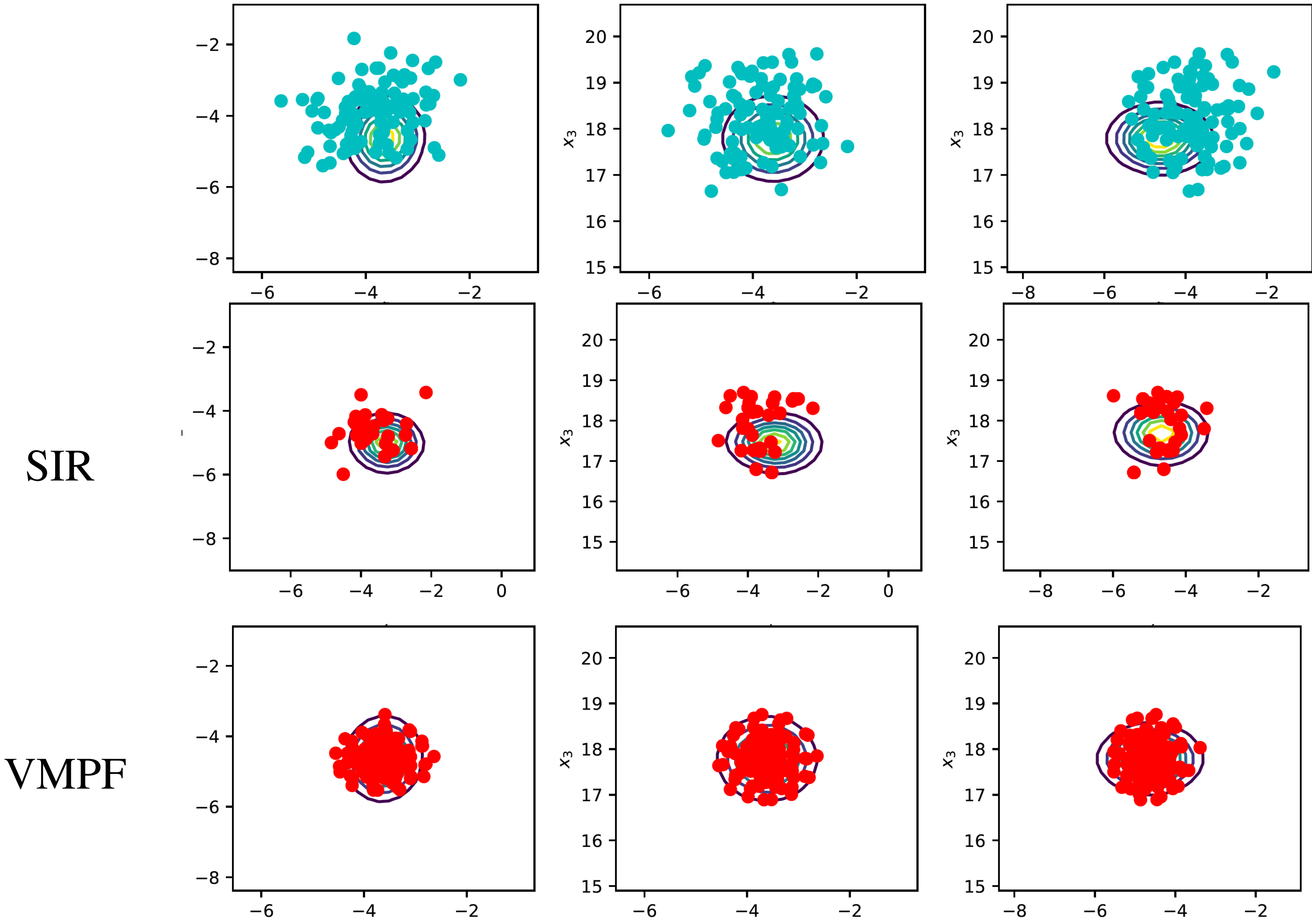
SIR



VMPF



Results on marginal pdfs 2D



And in high dimensions?

In sequential Bayesian Inference we do not know what the posterior looks like. Here we explore the model transition density to find

$$p(x^n|y^n) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(y^n|x_i^{n-1})}{p(y^n)}p(x^n|x_i^{n-1},y^n)$$

New element is that we can use localisation to obtain a smooth but more accurate estimate of the posterior, without sampling from this posterior! Localisation will:

- Increase the effective number of particles
- Leads to better weight balance for the mixture coefficients (Note that localisation scale not dictated by physics, so can be smaller, so less observations in each area, so less degenerate!)

Model equation and Kolmogorov equation

If the model equation is

$$\frac{\partial x}{\partial t} = f(x)$$

Then the pdf of *x* evolves as:

$$\frac{\partial p(x)}{\partial t} = -\nabla_x \left(f(x) p(x) \right)$$

This is the Kolmogorov equation (Fokker-Plank equation).

Turn this around: if I know the evolution equation for the pdf I can find the evolution equation for the state, so for the particles!

Transport of pdf

Bayes theorem reads:

$$p(x|y) = L(y|x)p(x)$$

with

$$L(y|x) = \frac{p(y|x)}{p(y)}$$

Define a sequence of pdfs that smoothly transform from prior to posterior:

$$\pi(x,\tau) = L(y|x)^{\gamma(\tau)}p(x)$$

with $\gamma(0) = 0 \quad \gamma(1) = 1$,so $\quad \pi(x,0) = p(x), \quad \pi(x,1) = p(x|y)$

Take time derivative to artificial time τ :

$$\frac{\partial \pi}{\partial \tau} = L^\gamma p \log L \frac{\partial \gamma}{\partial \tau} = \pi \log L \frac{\partial \gamma}{\partial \tau}$$

Finding f for the particles…

So we find the evolution equation for the pdf

$$\frac{\partial \pi}{\partial \tau} = \pi \log L \frac{\partial \gamma}{\partial \tau}$$

Remember that we want to write it as:

$$\frac{\partial \pi}{\partial \tau} = -\nabla (f \pi)$$

This leads to an equation for the vector function *f* in terms of the scalar function $\gamma(\tau)$ as follows:

$$\nabla f + f \left(\gamma \nabla \log L + \nabla \log p \right) = -\log L \frac{\partial \gamma}{\partial \tau}$$

This equation can be solved for the 1D problem and needs smart people for the high-dimensional problem…

Conclusions

- Data assimilation is based on a solid mathematical framework: Bayes Theorem.
- Large number of filters and smoothers can be derived from that.
- Best method will be system dependent
- Fully nonlinear equal-weight particle filters for systems with arbitrary dimensions do exist.
- Localisation needs further exploration.
- Proposal-density freedom needs further exploration.
- Transportation Particle Filters need further exploration…
- But first local Particle Filter has been implemented by DWD for weather prediction!