Nonlinear Data Assimilation and Particle Filters

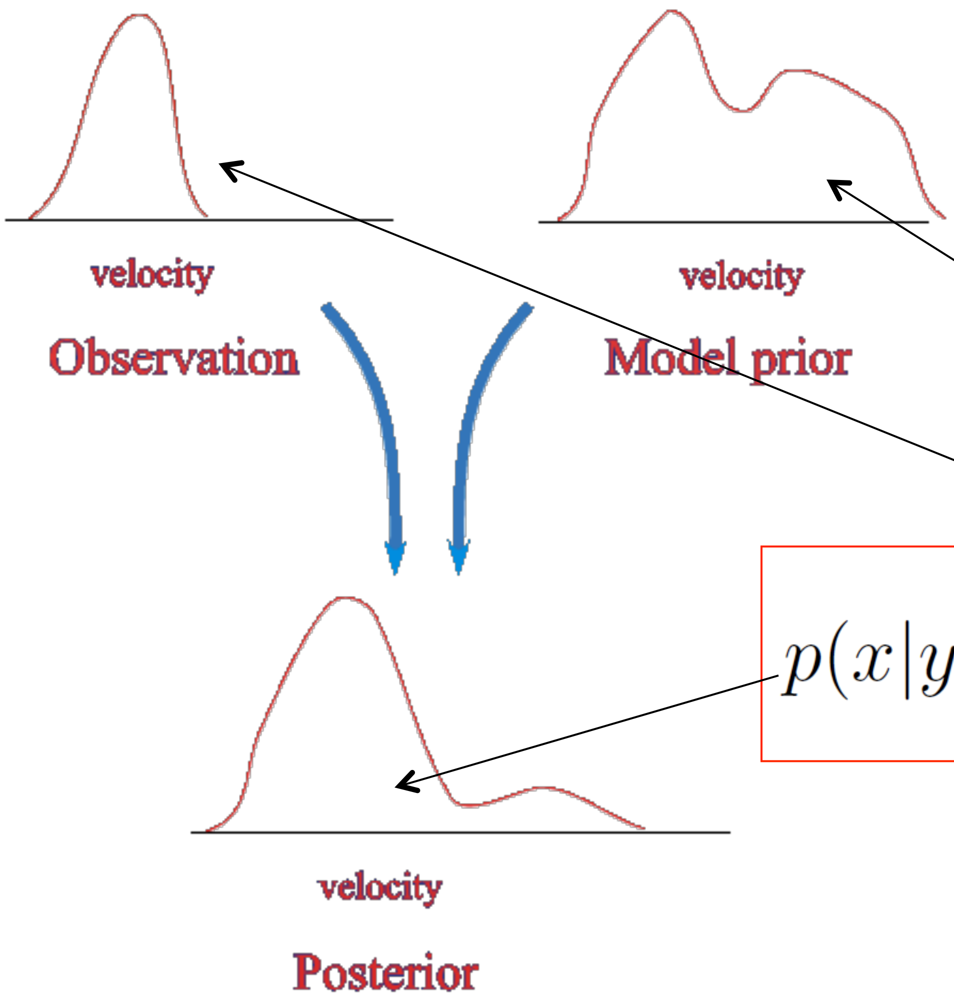
Peter Jan van Leeuwen

Data Assimilation Research Centre

University of Reading

National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL

Data assimilation: general formulation



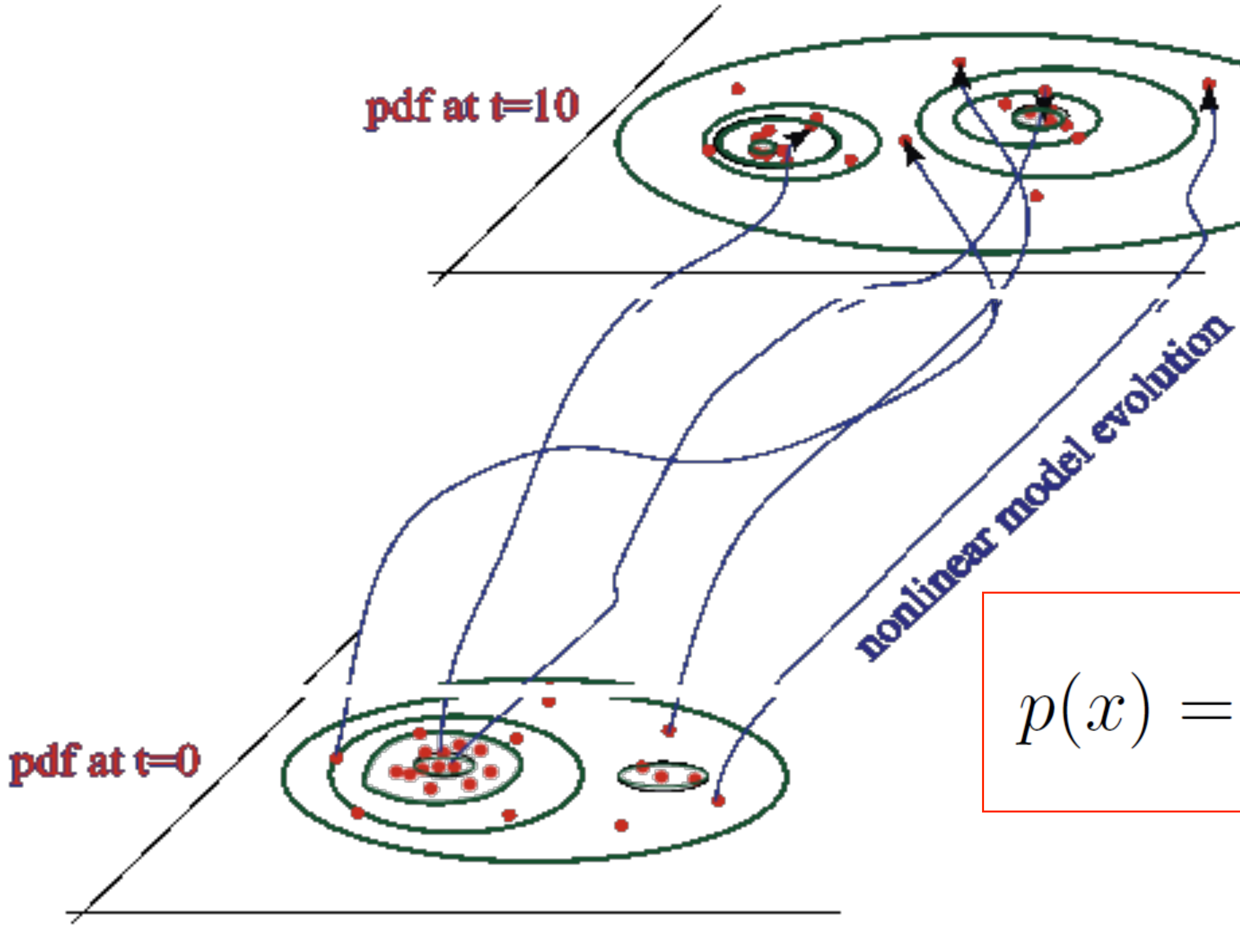
Bayes theorem:

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

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

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

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

$$\Big\downarrow \text{ Use ensemble}$$

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

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

with $\boxed{w_i = \frac{p(y|x_i)}{\sum_j p(y|x_j)}}$ 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 is is given by:

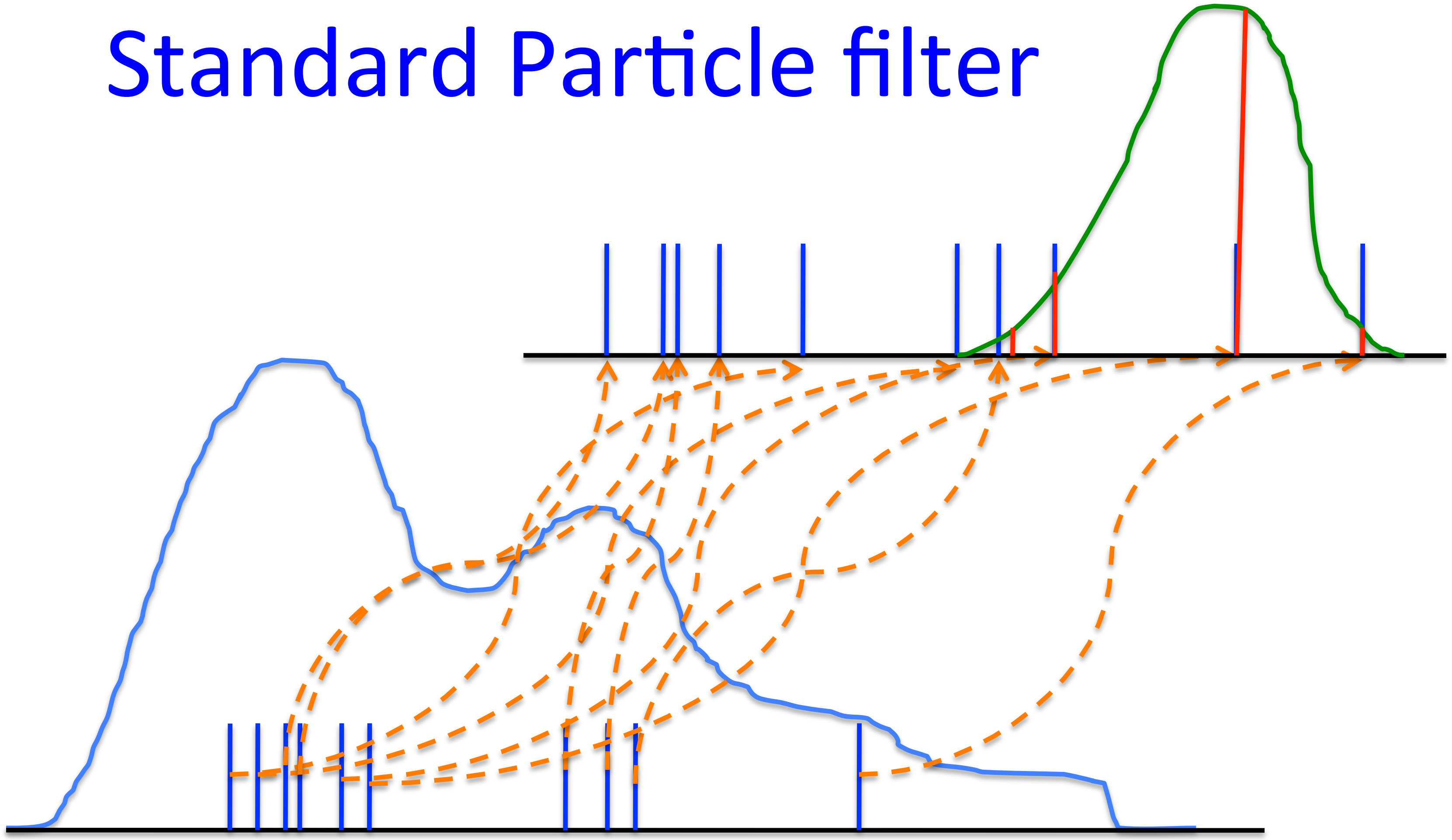
$$\begin{aligned} w_i &\propto p(y|x_i) \\ &\propto \exp\left[-\frac{1}{2}\left(y - H(x_i)\right) R^{-1} \left(y - H(x_i)\right)\right] \end{aligned}$$

- One can just calculate this value
- 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



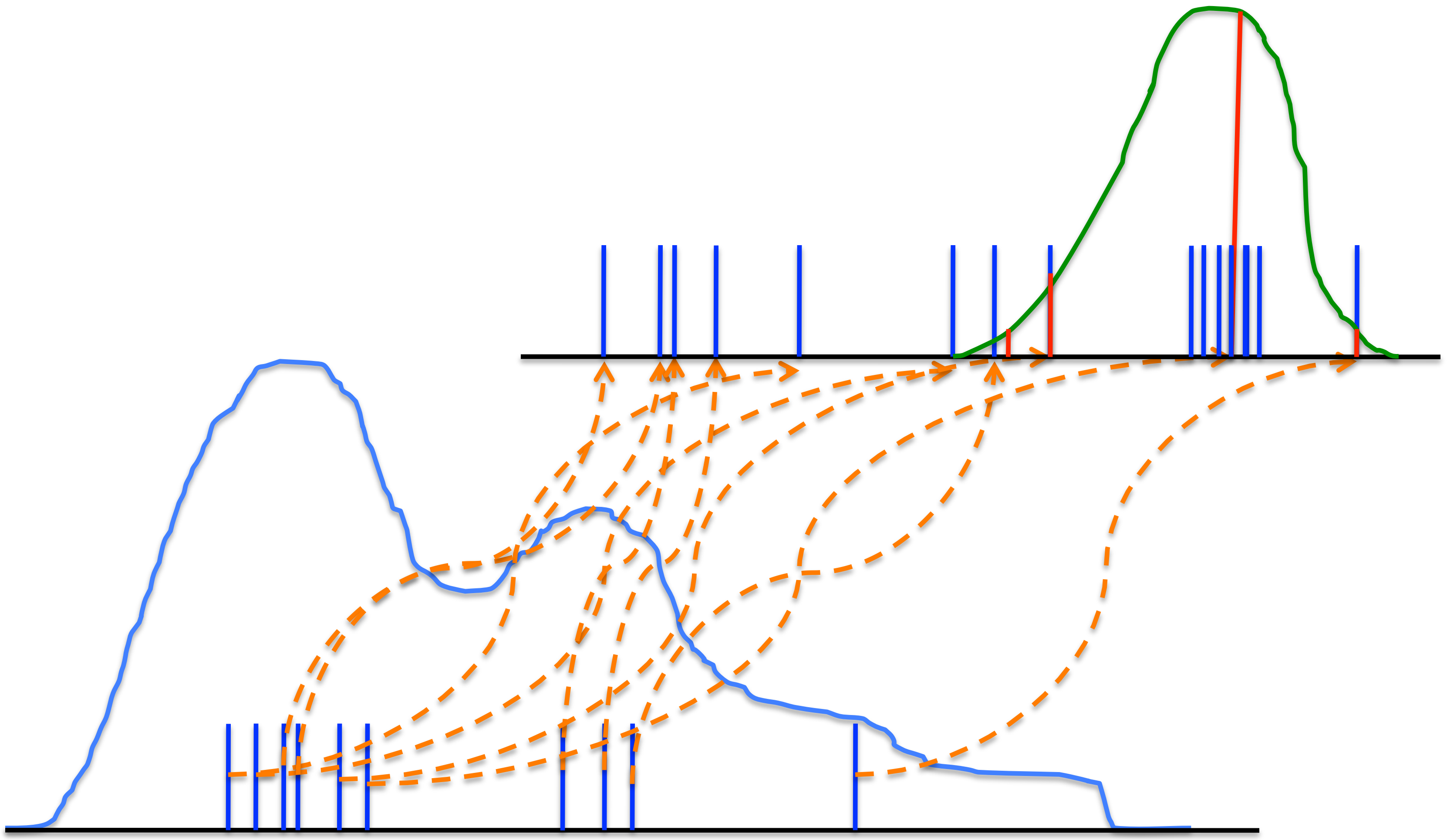
The standard particle filter is degenerate for moderate ensemble size in moderate-dimensional systems.

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:

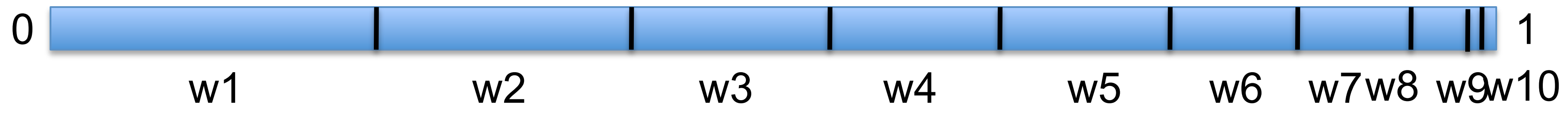
  Resampling: duplicate high-weight particles and abandon low-weight particles

Standard Particle filter

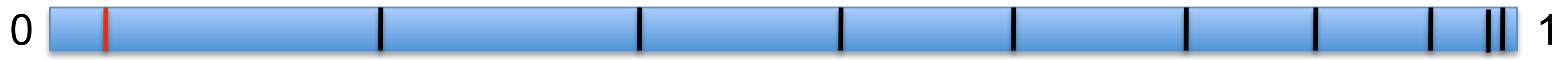


A simple resampling scheme

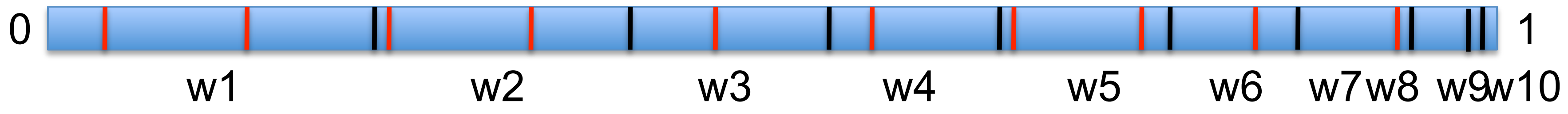
1. Put all weights after each other on the unit interval:



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: its end point is the first member drawn.



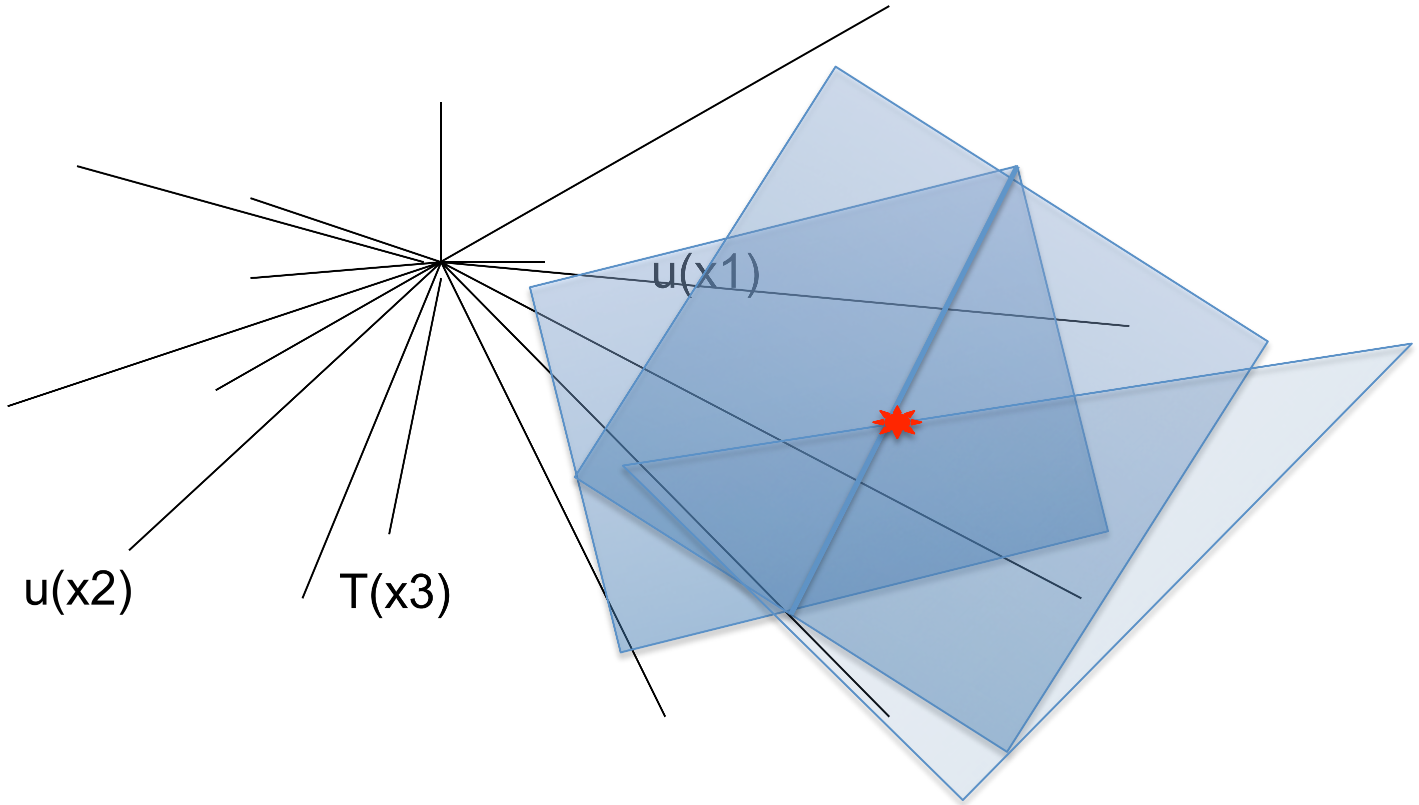
4. Add 1/N to the end point: the new end point is our second member. Repeat this until N new members are obtained.



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

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. Several ad-hoc combinations of Particle Filters and Ensemble Kalman Filters

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.
- So need smooth updates without resampling.
- Example is Ensemble Transform Particle Filter (ETPF, Reich, 2014).

The ETPF

- Find a linear map between prior and posterior ensemble:

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

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

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

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

The ETPF

- Minimisation takes $O(N_e^2 \log N_e)$ 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!

2. Exploring the proposal density freedom

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

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

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

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

We introduced the **transition densities**

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

Meaning of the transition densities

Stochastic model:

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

So, draw a sample from the model error pdf, and use that in the stochastic model equations.
For a Gaussian model error we find:

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

Bayes Theorem and the proposal density

Bayes Theorem now becomes:

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

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

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

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

The magic: the proposal 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 proposal transition density

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

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

Note that 1) the proposal depends on the future observation, and 2) the proposal depends 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 $q(x^n|x^{n-1}_{1:N}, y^n)$, *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:

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

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:

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

Which can be rewritten as:

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

with weights

$$w_i = \underbrace{\frac{p(y^n|x_i^n)}{Np(y^n)}}_{\text{Likelihood weight}} \underbrace{\frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_{1:N}^{n-1}, y^n)}}_{\text{Proposal weight}}$$

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

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

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

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:

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

Example calculation of *p*

- Assume the proposed model is

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

- 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:

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

- So the probability becomes

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

- 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:

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

Example: EnKF as proposal

EnKF update:

$$\boxed{x_i^n = x_i^* + K^e\left(y^n + \epsilon_i - H(x_i^*)\right)}$$

Use model equation:

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

Regroup terms:

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

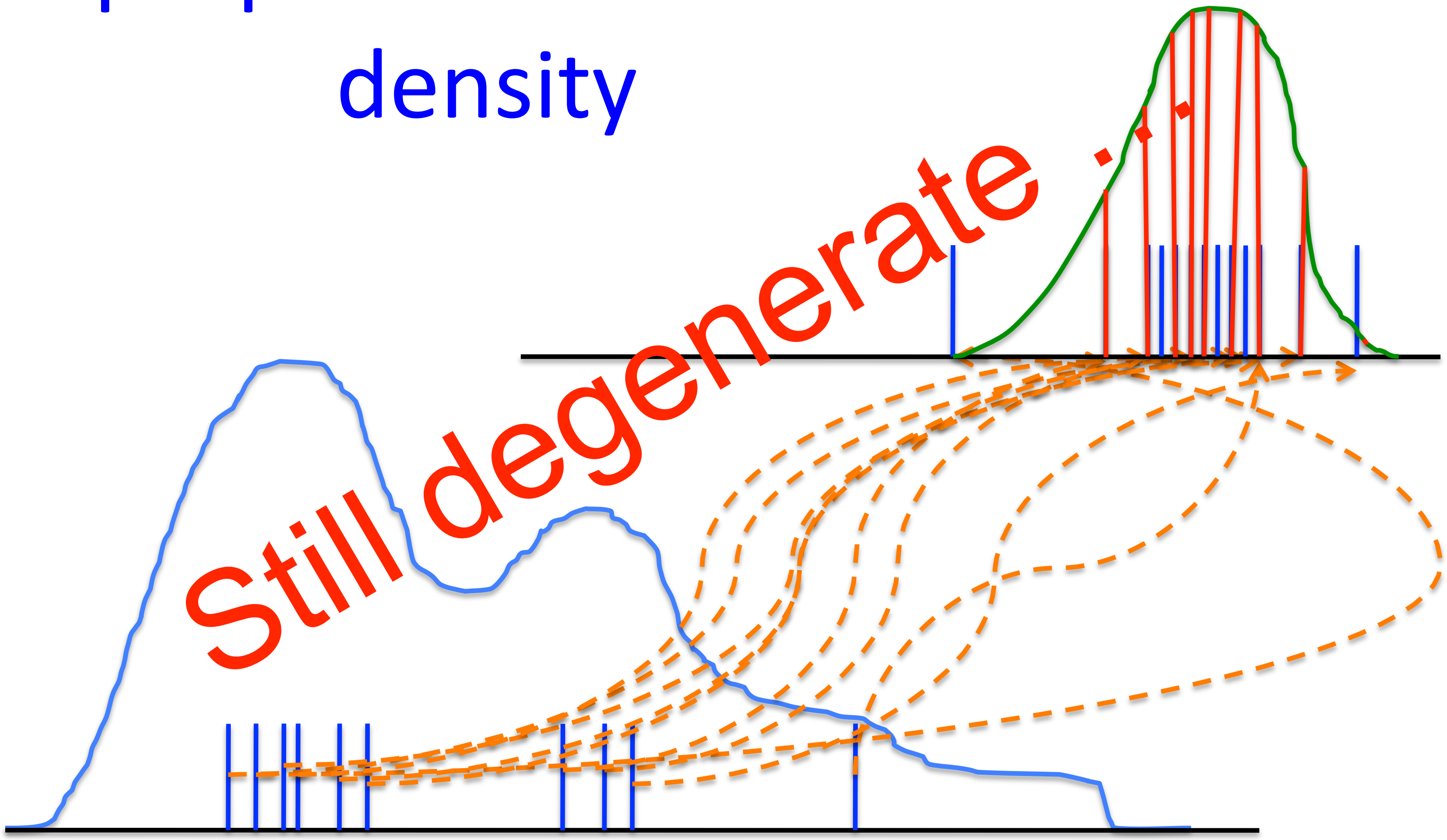
Leading to:

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

Algorithm

- Generate initial set of particles
- Run proposed model conditioned on next observation
- Accumulate proposal density weights *p/q*
- Calculate likelihood weights
- Calculate full weights and resample
- Note, the original model is never used directly.

Particle filter with proposal transition density



Transition densities

- Proposal that depends on previous model state:

$$q(x_i^n|...) = p(x_i^n|x_i^{n-1})$$

- Proposal that depends on observations and previous model state:

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

e.g. optimal proposal density: $q(x_i^n|...) = p(x_i^n|x_i^{n-1}, y^n)$

- Proposal that depends on observations and all we know about previous state:

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

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

Implicit Equal-weight Particle Filter

Define an *implicit map* as follows:

$$x_i^n = x_i^a + \sqrt{\alpha_i} P_i^{1/2} \xi_i$$

x_i^a is the mode of the optimal proposal density,

P_i is the covariance of the optimal proposal density.

1) Draw ξ_i from *N(0,I)*
2) Normalise each element $\xi_i^{(j)} = \xi_i^{(j)} \sqrt{N_x / ||\xi_i||}$
3) Calculate α_i such that pre-weights of all particles equal to target weight (see next slides)
4) Draw *k* from [1,..,N_x] with equal probability
5) Draw $\xi_i^{(k)}$ again from *N(0,1)*
6) Recalculate x_i^n using $\xi_i^{(k)}$ and normalised ξ_i for $i \neq k$.

How to find α_i

Remember the new particle is given by:

$$x_i^n = x_i^a + \sqrt{\alpha_i} P_i^{1/2} \xi_i$$

in which x_i^a, P_i, ξ_i are known, ξ_i normalised . Use this in expression for weights and set all weights equal to a target weight:

$$\boxed{w_i = \frac{p(x_i^n | x_i^{n-1}, y^n) p(y^n | x_i^{n-1})}{q(\xi)} \left|\left| \frac{dx}{d\xi} \right|\right| \cdot w_i^{prev}}$$

and solve for α_i . At this stage all weights are equal by construction!

Solution for α_i

For Gaussian model errors and observation errors and *H* linear we find for α_i:

$$\alpha_i - \log \alpha_i = C - \phi_i$$

with solution

$$\alpha_i = -W_{0,-1}\left[-e^{-(1+C-\phi_i)}\right]$$

in which *W* is the Lambert W function.
We choose one of the two solutions W_0 or W_{-1} with equal probability.
Target weight chosen equal to lowest weight of all particles.

Resulting scheme:

- Effectively we use *N_x-1* random numbers to calculate α_i such that the weights of the particles are equal.
- Then one random number is chosen to ensure a proper map from ξ_i to x_i^n. Weights will vary, but only slightly.
- This last step also ensures that the proposal has full support.
- Scheme can be seen as adaptation of implicit particle filter (optimal proposal) to avoid weight collapse.
- Scheme is biased because target weight is set to lowest weight of all particles at their highest weight positions.
- The latter are equal to solution of weak-constraint 4Dvar without background term.

Experiment

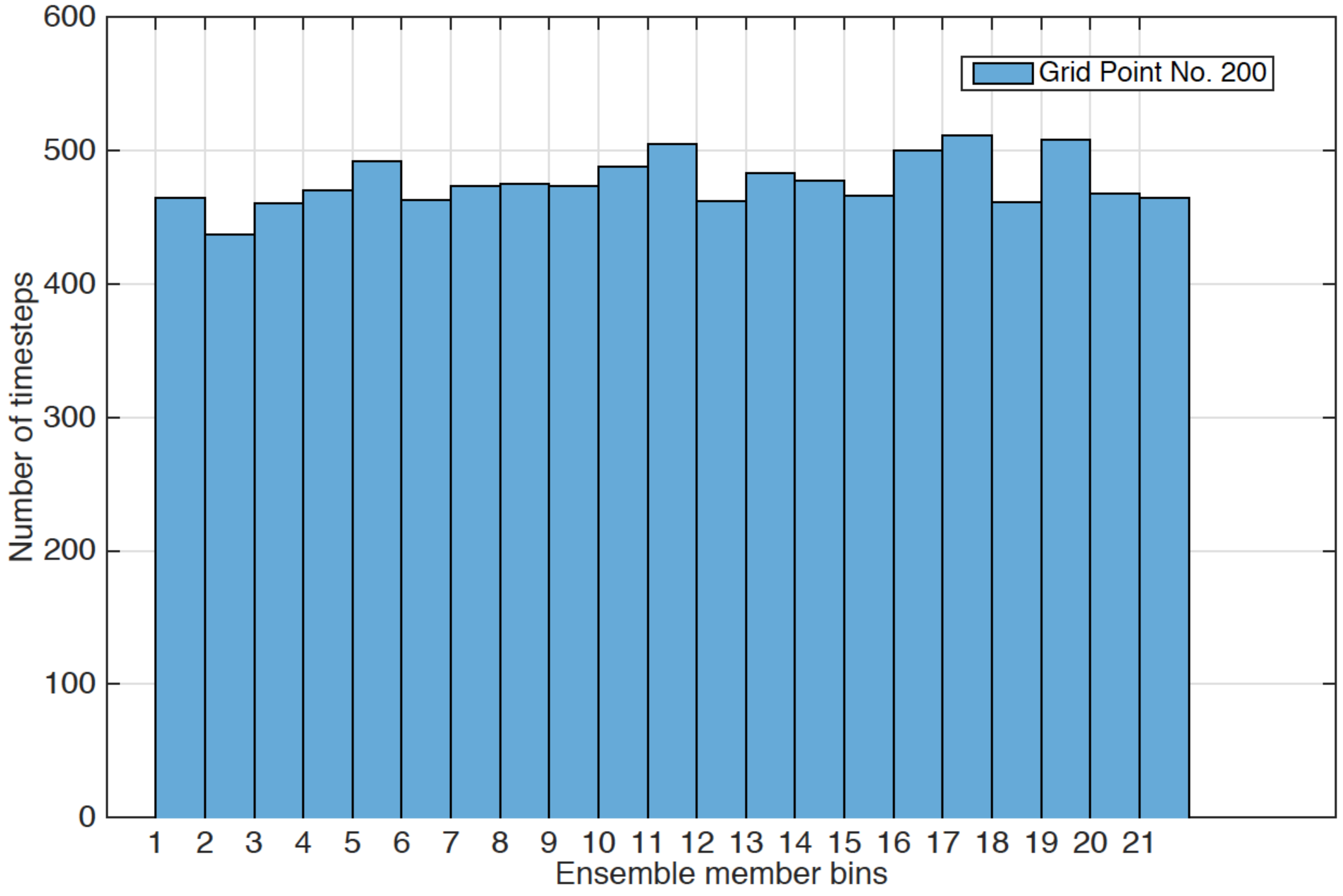
- Linear model of Snyder et al. 2008.
- 1000 dimensional independent Gaussian linear model
- 20 particles
- Observations every time step of whole state

$$x^n = x^{n-1} + \beta^n$$

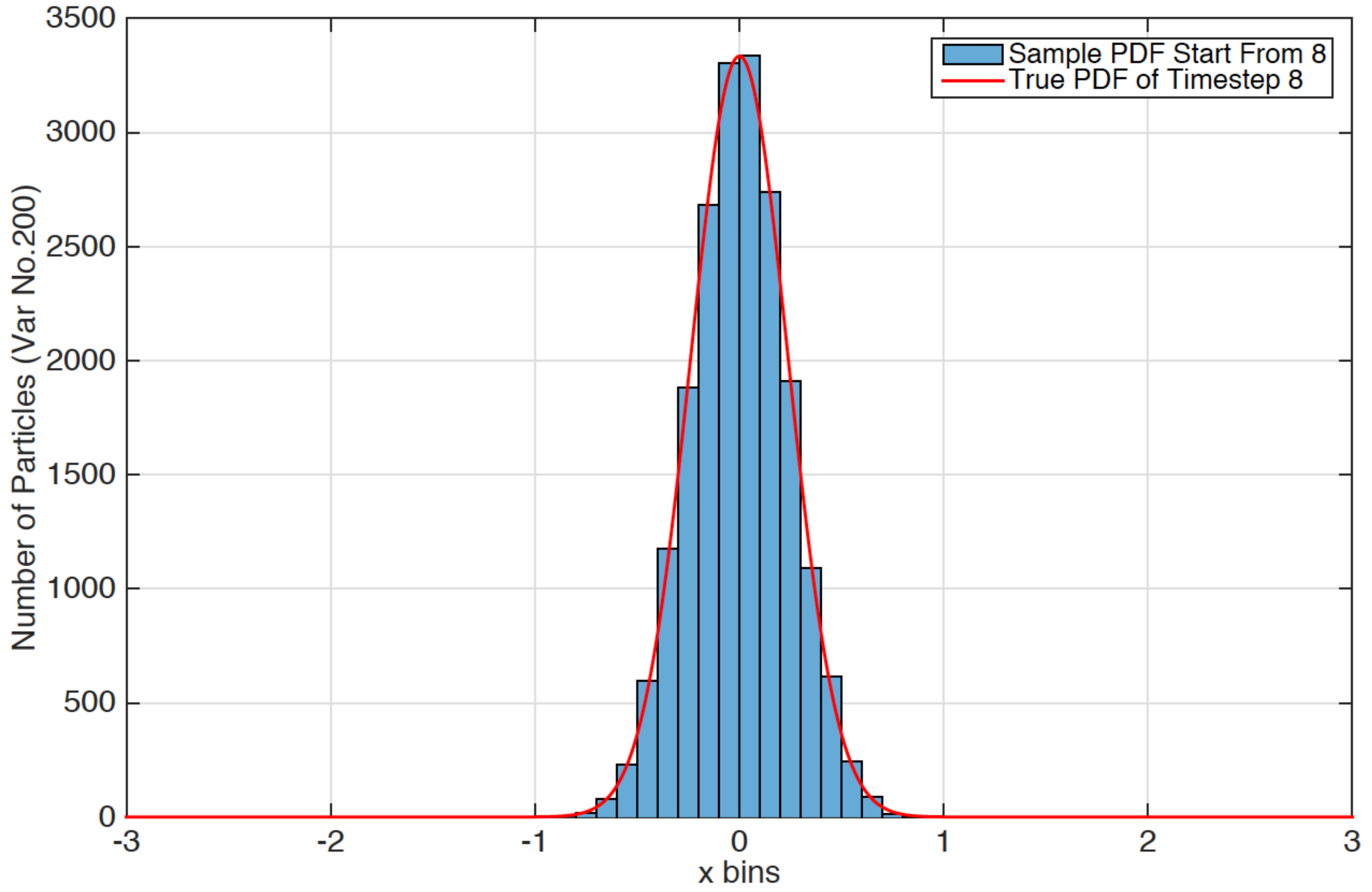
$$\beta^n \sim N(0, I)$$

Linear model: Rank histogram

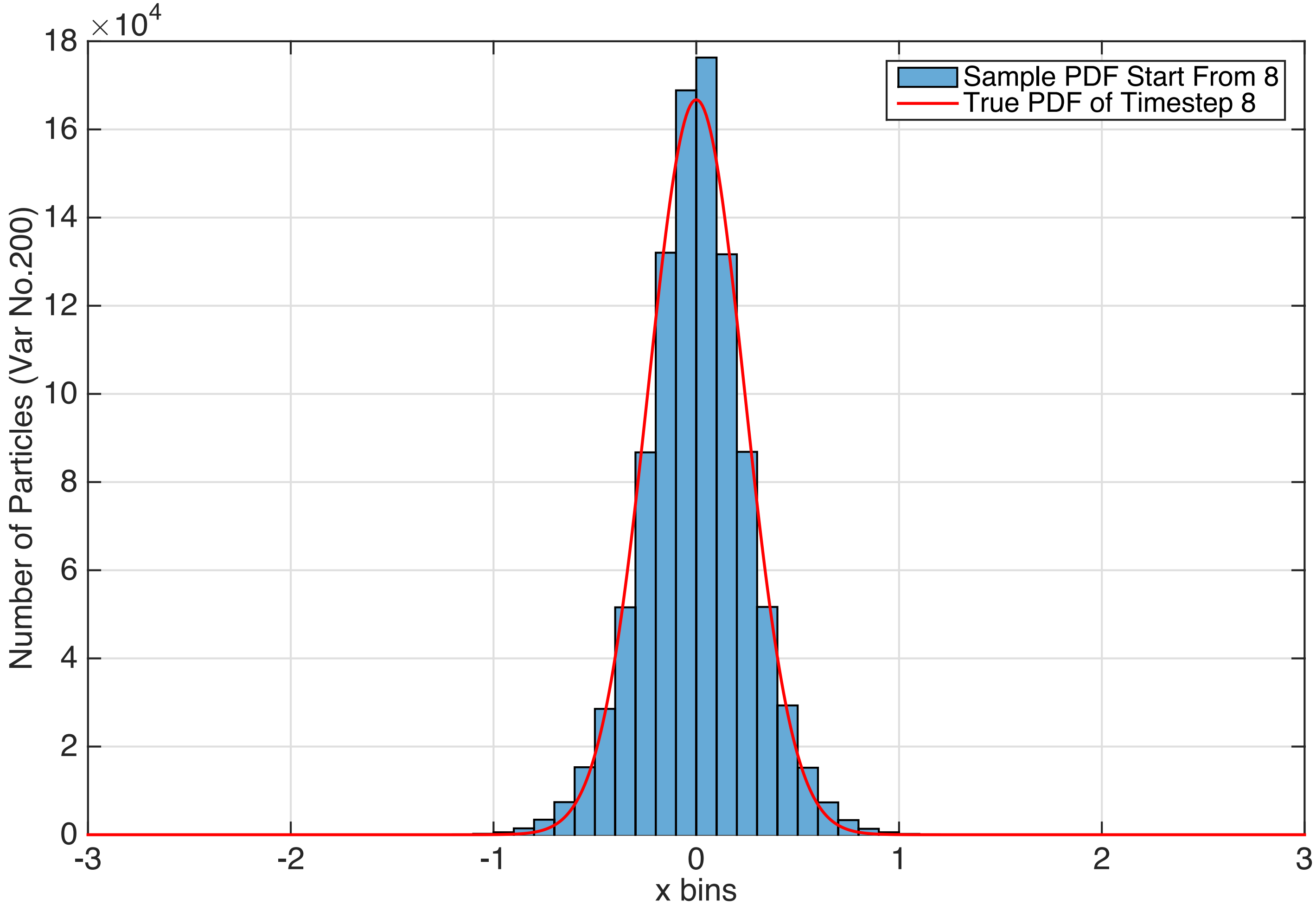
1000 time steps, 20 particles



Normalised pdf 1000 time steps 20 particles



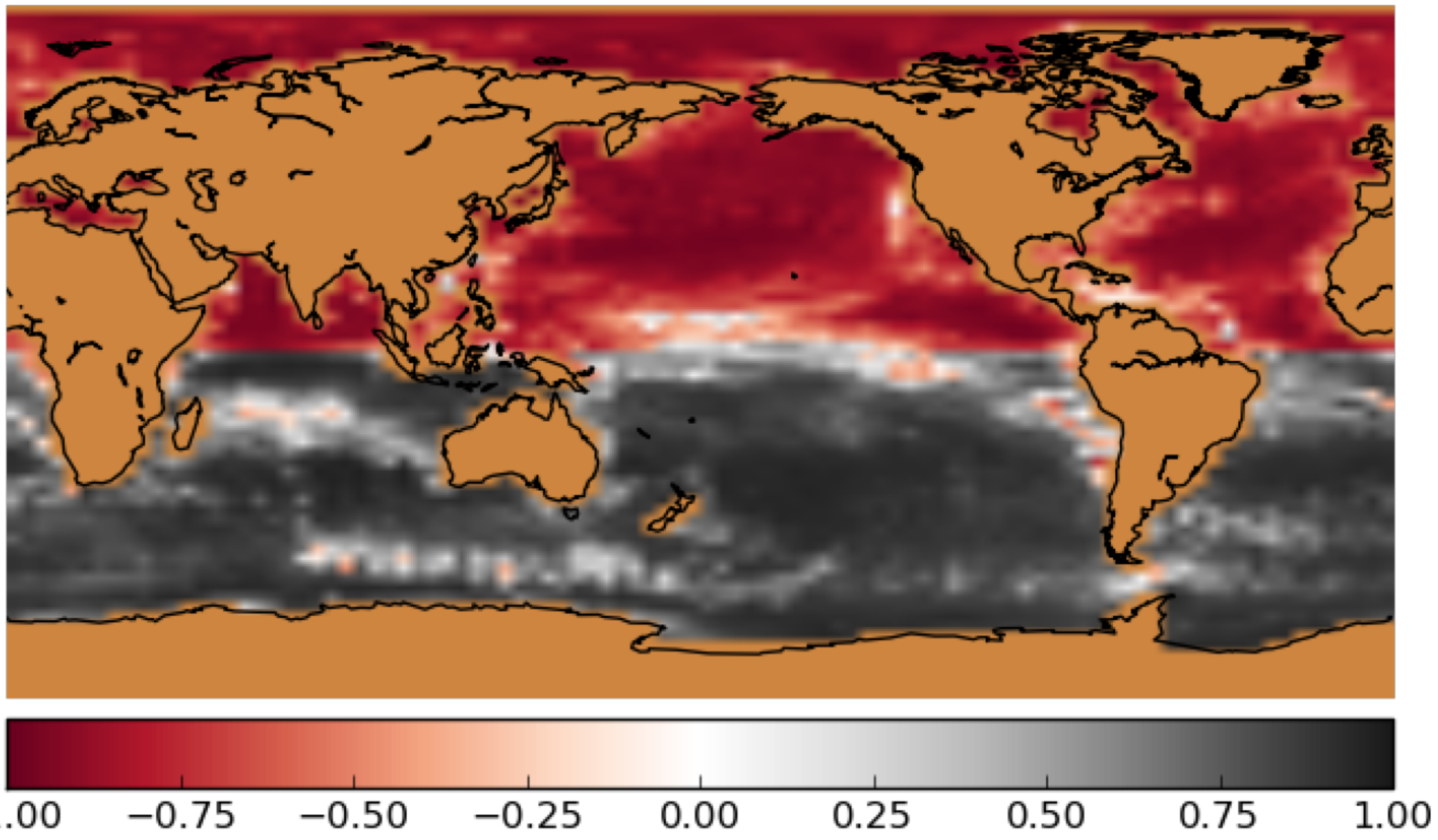
Normalised pdf 1000 time steps 1000 particles -> Convergence!



Climate model HadCM3

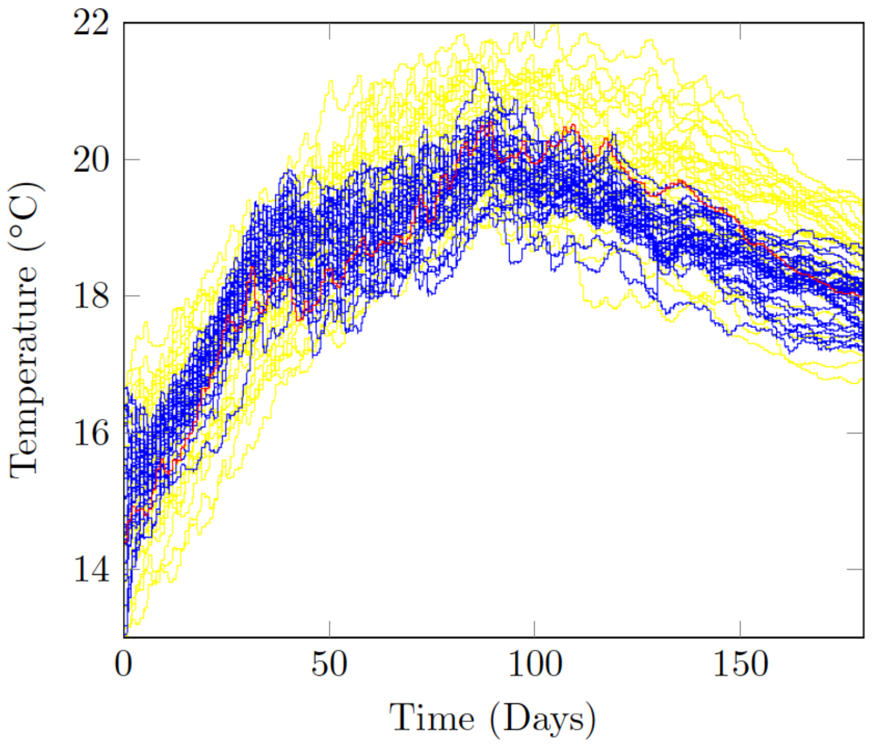
- Identical twin experiment
- 32 particles
- 2.3 million variables
- Daily observations of Sea-Surface Temperature with uncertainty 0.55 K
- Model errors smaller than 0.1 times deterministic model update
- Correlation structure from snapshots of long model run.

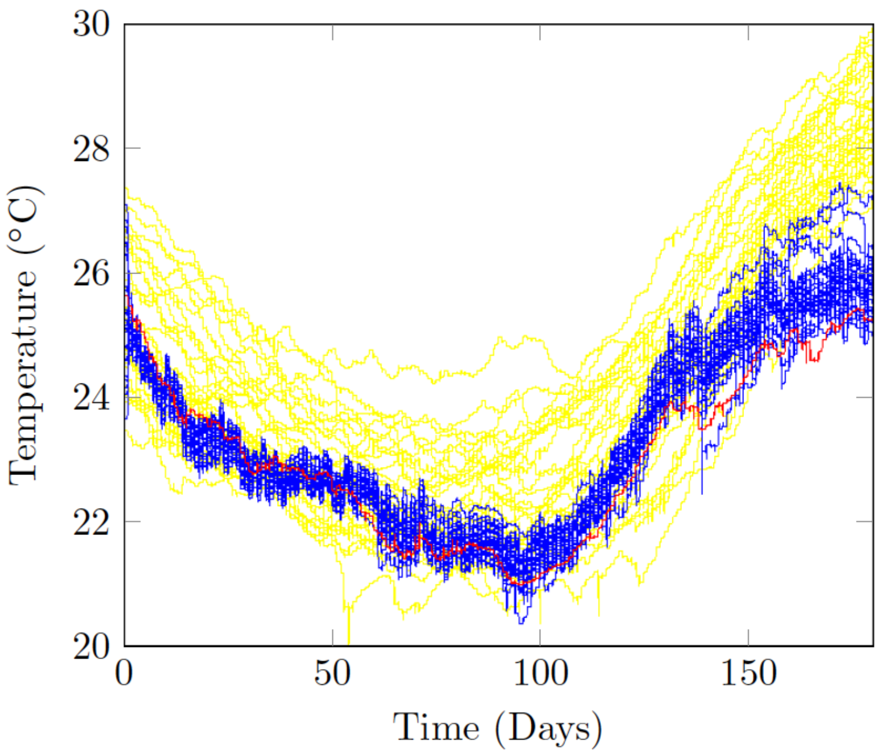
Model error covariance



Correlation atmospheric zonal flow and oceanic meridional flow

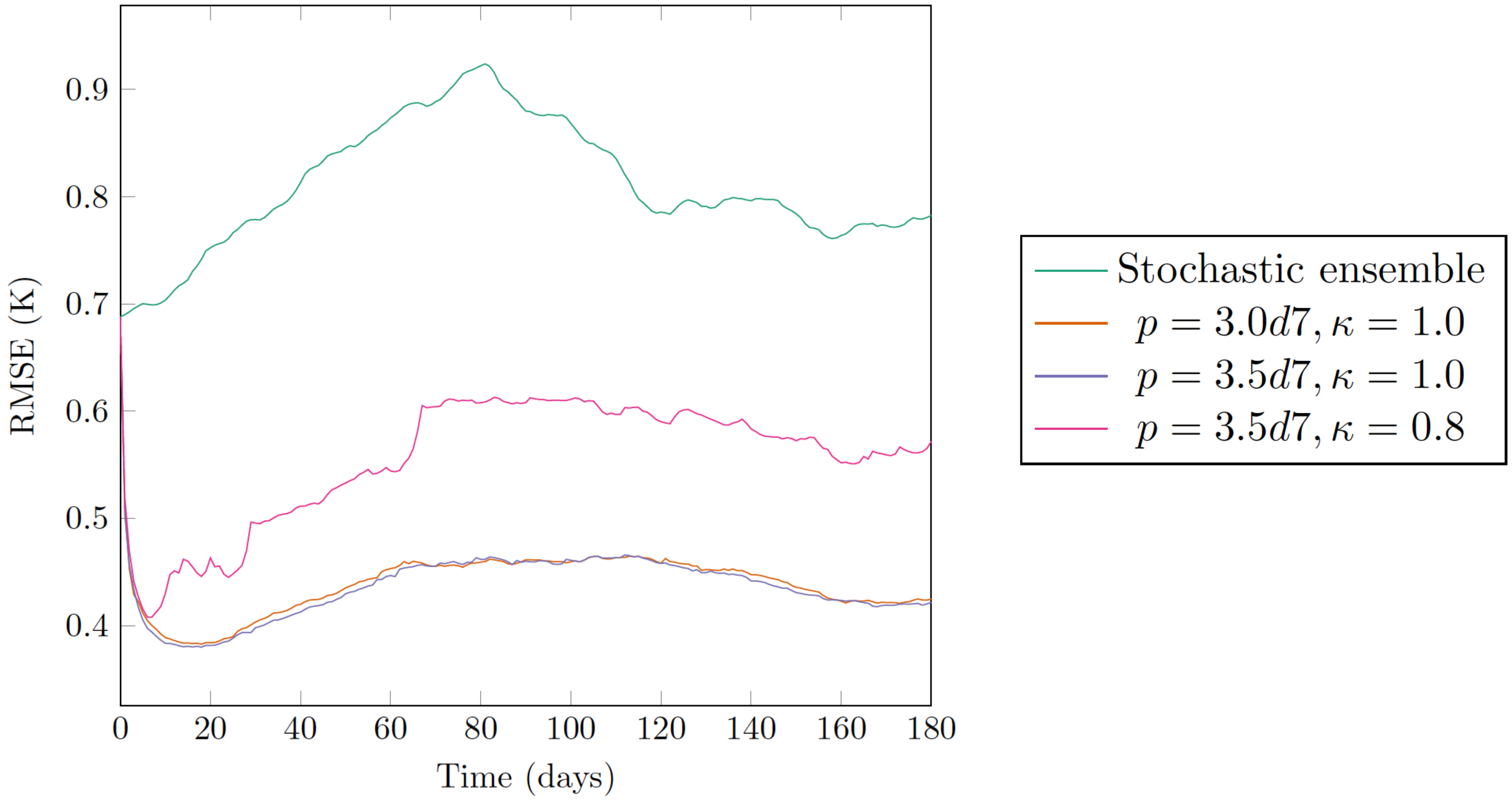
Time evolution of particles



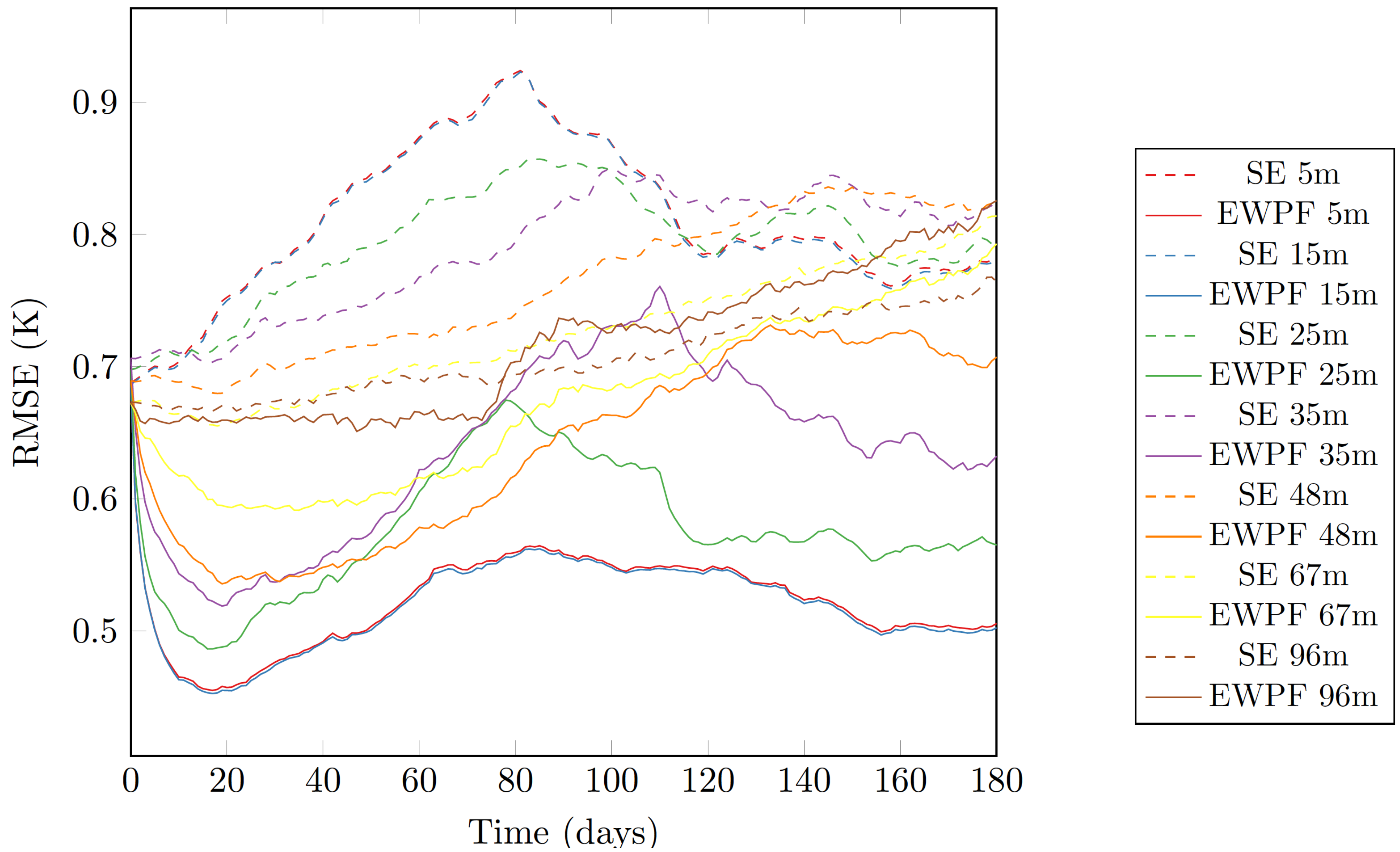


Prior ensemble (yellow), posterior ensemble (blue), truth (red), for SST in two grid points

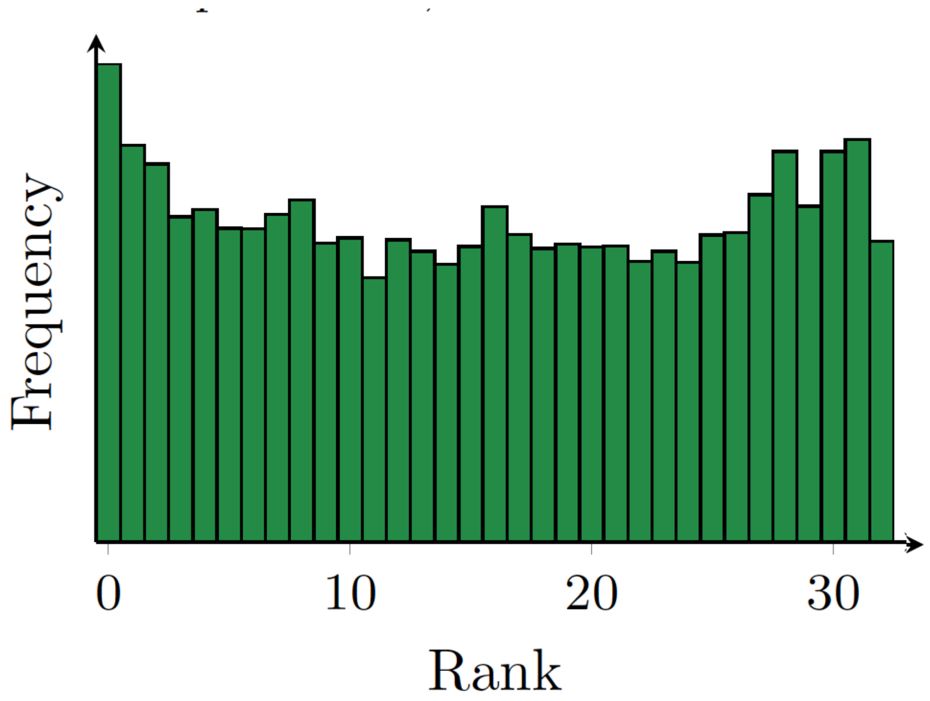
Results: Observed variable SST



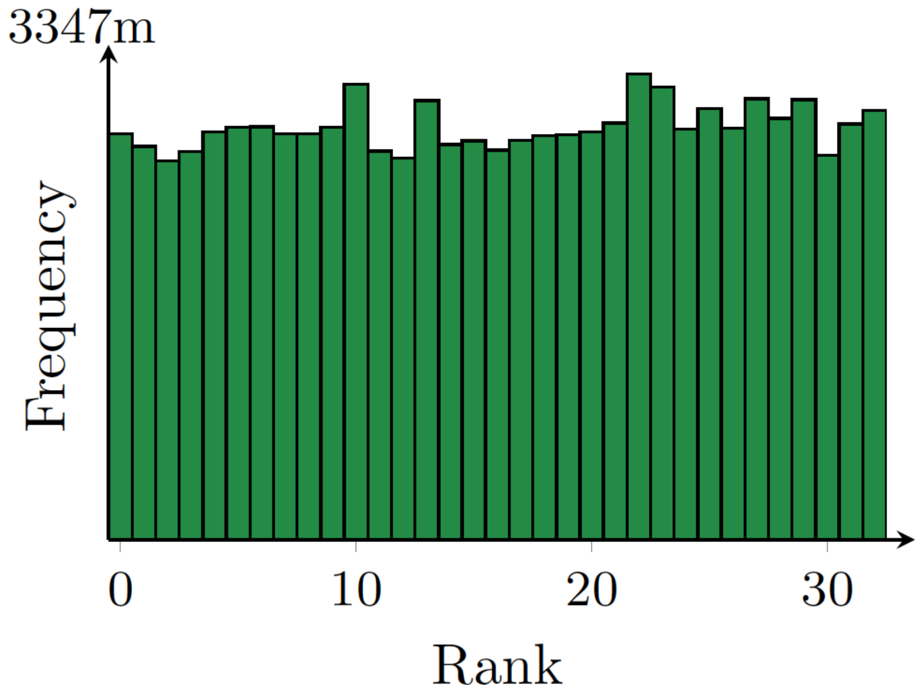
Results: Ocean Temperature



Rank Histograms

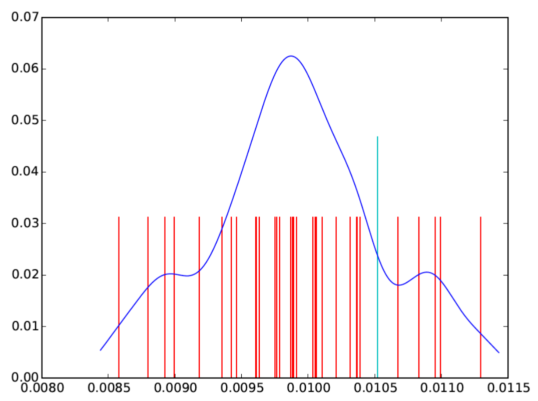


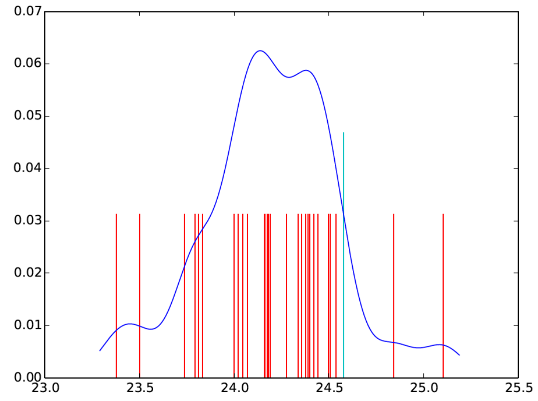
SST (observed)

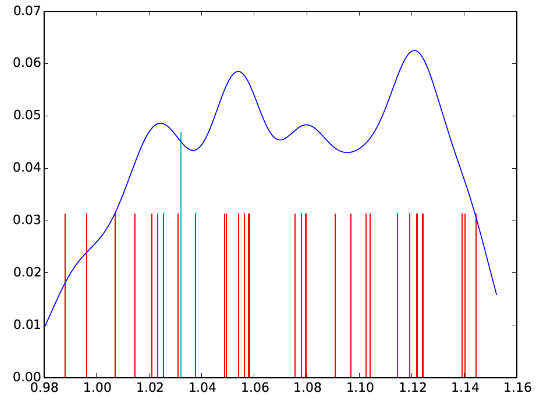


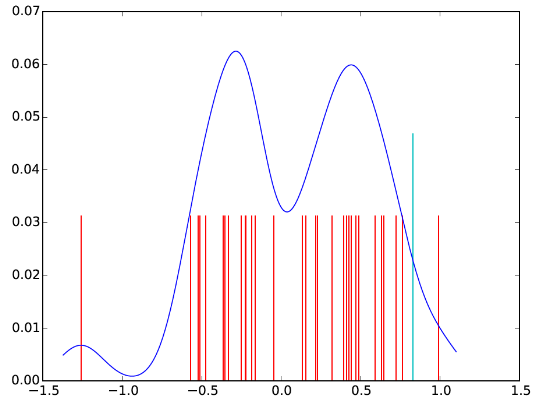
Meridional wind high up in Atmosphere (unobserved)

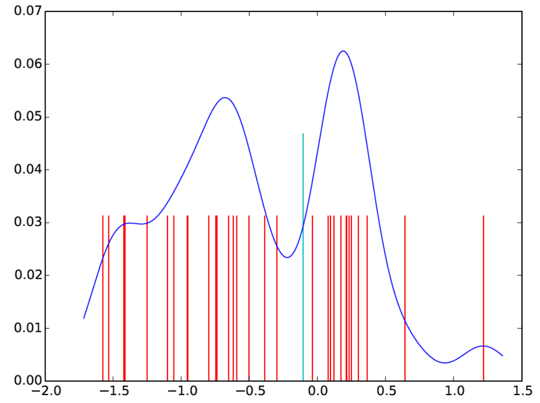
Estimated pdfs

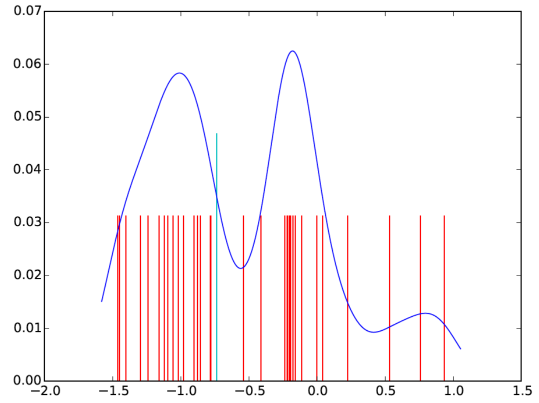

(a) Surface level specific humidity at (20S,270E)


(b) Seawater temperature at (1.875S,213.75E) at a depth of 48m


(c) Seawater temperature at (51.875S,228.75E) at a depth of 3347m


(d) Meridional seawater flow at (15S,100.625E) at 3347m depth on day 75


(e) Meridional seawater flow at (15S,100.625E) at 3347m depth on day 161


(f) Meridional seawater flow at (15S,100.625E) at 3347m depth on day 177

Conclusions

- Large number of 'nonlinear' filters and smoothers available
- Best method will be system dependent
- Fully nonlinear equal-weight particle filters for systems with arbitrary dimensions that converge to the truth posterior pdf do exist.
- Localisation, e.g. Ensemble Transform Particle Filter, needs further exploration.
- Proposal-density freedom needs further exploration.
- Example shown for 1000 dimensional systems, but methods have been applied to 2.3 million dimensional systems too.