6. Particle filtering

In this chapter we will discuss particle filters and their use in the geosciences. First the basic idea behind particle filters is presented, followed by why this basic formulation can never work for large-dimensional systems. We then explore methods that do have this potential, with special focus on methods that use the proposal density. The last part of this chapter is devoted to particle filter based methods that solve Bayes theorem approximately, trying to bridge the connection with e.g. the Ensemble Kalman filter.

a. Basic Importance Sampling

The particle filters we will discuss here are based on Importance Sampling. The most straightforward implementation is what is called Basic Importance Sampling here. (In the statistical literature one usually finds Importance Sampling described with a proposal density different from the prior model pdf. However, for pedagogical reasons we present Importance Sampling in the following way.) Basic Importance sampling is straightforward implementation of Bayes Theorem. The idea is to represent the prior pdf by a set of particles x_i, which are delta functions centred around state vectors x_i, and from which all statistical measures of interest can be calculated, like mean, covariance etc. Using the particle description of the prior pdf given by

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

in Bayes:

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\; dx} \tag{2}$$

gives:

$$p(x|y) = \sum_{i=1}^{N} w_i \delta(x - x_i) \tag{3}$$

in which the weights w_i are given by:

$$w_i = \frac{p(y|x_i)}{\sum_{j=1}^{N} p(y|x_j)} \tag{4}$$

The density $p(y|x_i)$ is the probability density of the observations given the model state x_i, which is often taken as a Gaussian:

$$p(y|x_i) = A \exp\left[-\frac{(y - H(x_i))^2}{2\sigma^2}\right] \tag{5}$$

in which $H(x_i)$ is the measurement operator, which is the model equivalent of the observation y, and σ is the standard deviation of the observation error. When more measurements are available, which might have correlated errors, the above should be the joint pdf of all these measurements.

Weighting the particles just means that their relative importance in the probability density changes. For instance, if we want to know the mean of the function $f(x)$ we now have:

$$\overline{f(x)} = \int f(x)p(x)\; dx \approx \sum_{i=1}^{N} w_i f(x_i) \tag{6}$$

Common examples for $f(x)$ are x itself, giving the mean of the pdf, and the squared deviation from the mean, giving the covariance.

Up to now, we haven't specified what x is. If x is a model trajectory over some time window $(0, n\Delta t)$, we have $x = x^{0:n} = (x^0, x^1, ..., x^n)$ over n time steps. Note the change in notation form the last two chapters. There, the superscript denoted the iteration or sample index, in which the sample could be a state vector at a specific time, or a full trajectory over a certain time window. The subscript denoted the component of the state vector (or trajectory). Here the superscript is the time index, and the subscript is the sample, or particle. Bear this in mind! Note also that in this case the 'filter' is actually a smoother, in which a smoother is defined as the posterior pdf in which also future observations have been taken into account.

A practical way to implement this is to calculate this sequentially over time, which is where the name 'filter' comes from. The idea is to write the prior density as

$$p(x^{0:n}) = p(x^n|x^{0:n-1})p(x^{0:n-1}) \tag{7}$$

Using that the state vector evolution is Markov, i.e. to predict the future we only need the present, not the past, we can write:

$$p(x^{0:n}) = p(x^n|x^{n-1})p(x^{n-1}|p(x^{n-2})...p(x^1|x^0)p(x^0) \tag{8}$$

Before we continue it is good to realise what the so-called transition densities $p(x^n|x^{n-1})$ actually mean. Consider a model evolution equation given by:

$$x^n = f(x^{n-1}, \beta^{n-1}) \tag{9}$$

in which $\beta^{n-1}$is a random term or factor in the model equation that describes the error in the model equation. The idea is that the model is not perfect, i.e. any numerical model used in the geosciences that is used to simulate the real world has errors (and these tend to be significant!). These errors are unknown (otherwise we would include them as deterministic terms in the equations) but we assume we are able to say something about their statistics, e.g. their mean, covariance, etc. Typically one assumes the errors in the model equations are Gaussian distributed with zero mean and known covariance, but that is not always the case. To draw from such a transition density $p(x^n|x^{n-1})$ means to draw β^{n-1} from its density and evaluate the model equation given above.

Let us now continue with Importance Sampling. If we also assume that the observations at different times are independent, which is not necessary for the formulation of the theory, but keeps the notation so much simpler, we have for the likelihood:

$$p(y^{1:n}|x^{0:n}) = p(y^n|x^n)...p(y^1|x^1) \tag{10}$$

where we used that y^j is not dependent on x^k with $j \neq k$ when x^j is known. The posterior density can now be written as:

$$\begin{aligned} p(x^{0:n}|y^{1:n}) &= \frac{p(y^{1:n}|x^{0:n})p(x^{0:n})}{p(y^{1:n})} \\ &= \frac{p(y^n|x^n)...p(y^1|x^1)p(x^n|x^{n-1})...p(x^1|x^0)p(x^0)}{p(y^n), ..., p(y^1)} \\ &= \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)}...\frac{p(y^1|x^1)p(x^1|x^0)p(x^0)}{p(y^1)} \end{aligned} \tag{11}$$

Realising that the last ratio in this equation is actually equal to $p(x^{0:1}|y^1)$ we find the following sequential relation:

$$p(x^n|y^n) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)} p(x^{0:n-1}|y^{1:n-1}) \tag{12}$$

This expression allows us to find the full posterior with the following sequential scheme(see figure 1):

1 Sample N particles x_i from the initial model probability density $p(x^0)$, in which the superscript 0 denotes the time index.

2 Integrate all particles forward in time up to the measurement time. In probabilistic language we denote this as: sample from $p(x^n|x_i^{n-1})$ for each i,
i.e. for each particle x_i run the model forward from time $n-1$ to time n using the nonlinear model equations. The stochastic nature of the forward evolution is implemented by sampling from the density that describes the random forcing of the model.

3 Calculate the weights according to (4) and attach these weights to each corresponding particle. Note that the particles are not modified, only their relative weight is changed!

4 Increase n by one and repeat 2 and 3 until all observations have been processed.

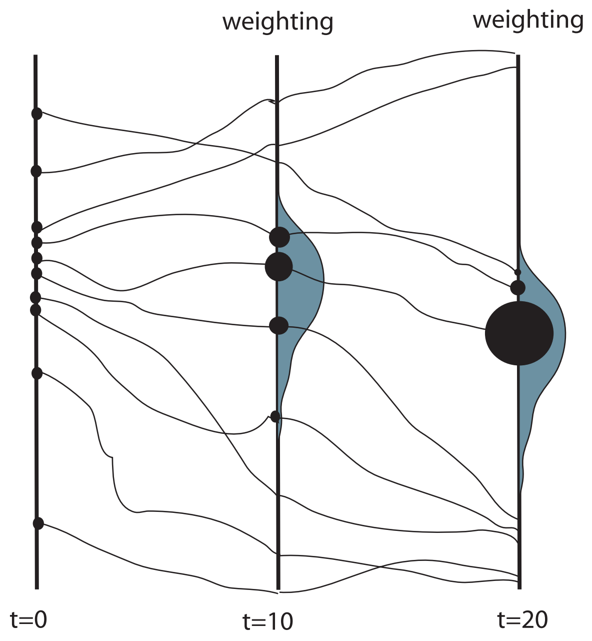

FIG. 1. *The standard particle filter with Importance Sampling. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time 0. At time 10 the likelihood is displayed together with the new weights of each particle. At time 20 only 2 members have weights different from zero: the filter has become degenerate.*

The good thing about importance sampling is that the particles are not modified, so that dynamical balances are not destroyed by the analysis. The bad thing about importance sampling is that the particles are not modified, so that when all particles move away from the observations they are not pulled back to the observations. Only their relative weights are changed.

However, there is more. Even if the particles do follow the observations in time, still the weights will differ more and more. Application to even very low-dimensional systems shows that after a few analysis steps one particle gets all the weight, while all other particles have very low weights (see figure 1, at $t = 20$). That means that the statistical information in the ensemble becomes too low to be meaningful. This is called *filter degeneracy.* It has given importance sampling a low profile until resampling was invented, see later on.

b. Reducing the variance in the weights

Several methods exists to reduce the variance in the weights. We discuss here Sequential Importance Resampling, the Mariginal Particle Filter and hierarchical models, because these are among the few that can be applied directly to large-dimensional problems. The first two methods 'break with the past' in that they get rid of the weights of the particles accumulated during previous assimilation steps. In resampling methods the posterior ensemble is resampled so that the weights become more equal. In the marginal particle filter the past is integrated out. Both methods do not change the position of the particles in state space. In the next section methods are discussed that do change the positions of the prior particles in state space to improve the likelihood of the particles, i.e. to bring them closer to the observations before the weighting with the likelihood is applied. In hierarchical models one tries to break up the full assimilation problem in a sequence of easier to solve smaller assimilation problems, using the concept of conditional probability densities.

1) Resampling

The idea of resampling is simply that particles with very low weights are abandoned, while multiple copies of particles with high weight are kept for the posterior pdf in the sequential implementation. In order to restore the total number of particles N, identical copies of high-weight particles are formed. The higher the weight of a particle the more copies are generated, such that the total number of particles becomes N again. Sequential Importance Re-sampling (SIR) does the above, and makes sure that the weights of all posterior particles are equal again, to $1/N$. Several resampling algorithms exist of which we discuss four. After that we discuss regularisation and the question whether resampling is enough to solve the degeneracy problem.

Sequential Importance Re-sampling is identical to Basic Importance Sampling but for a resampling step after the calculation of the weights. The 'flow chart' reads (see figure 2):

1 Sample N particles x_i from the initial model probability density $p(x^0)$.

2 Integrate all particles forward in time up to the measurement time (so, sample from $p(x^n|x_i^{n-1})$ for each i)

3 Calculate the weights according to (4) and attach these weights to each corresponding particle. Note that the particles are not modified, only their relative weight is changed!

4 Re-sample the particles such that the weights are equal to $1/N$.

5 Repeat 2, 3 and 4 sequentially until all observations have been processed.

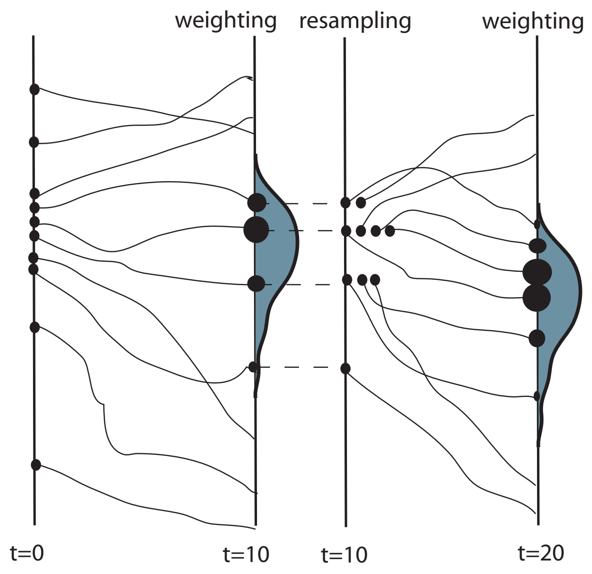

FIG. 2. *The Particle Filter with Resampling, also called Sequential Importance Resampling. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. At time 10 the particles are weighted according to the likelihood, and resampled to obtain an equal-weight ensemble.*

It is good to realise that the resampling step destroys the smoother character of the method. All particles that are not chosen in the resampling scheme are lost, and their evolution is broken. So the smoother estimate is build of of lesser and lesser particles over time, until it consists of only one particle.

The resampling can be performed in many ways, and we discuss the most used.

1) *Probabilistic resampling*

   Most straightforward is to directly sample randomly from the density given by the weights. Since this density is discrete and one-dimensional this is an easy task. However, due to the random character of the sampling, so-called sampling noise is introduced. Note that this method is actually generalised Bernoulli, as discussed in chapter 1.

2) *Residual Sampling*

   To reduce the sampling noise Residual Sampling can be applied. In this re-sampling method all weights are multiplied with the ensemble size N. Then n copies are taken of each particle i in which n is the integer part of Nw_i. After obtaining these copies of all members with $Nw_i \geq 1$, the integer parts of Nw_i are subtracted from Nw_i. The rest of the particles needed to obtain ensemble size N are than drawn randomly from this resulting distribution.

3) *Stochastic Universal Sampling*

   While Residual Sampling reduces the sampling noise, it can bee shown that Stochastic Universal Sampling has lowest sampling noise. In this method all weights are put after each other on a line $[0, 1]$. Then a random number is drawn from a uniform density on $[0, 1/N]$. Then N line pieces starting from the random number, and with interval length $1/N$ are laid on the line $[0, 1]$. A particle is chosen when one of the end points of these line pieces falls in the weight bin of that particle. Clearly, particles with high weights span an interval larger than $1/N$ and will be chosen a number of times, while small weight particles have a negligible change of being chosen.

4) *Monte-Carlo Metropolis-Hastings*

It is unusual to describe it as a re-sampling method, but for the application we have in mind it is. Here we restrict ourselves to a straightforward application to particle filtering. The algorithm works as follows:

1 Sample N particles x_i from the initial model probability density $p(x^0)$.

2 Sample from $p(x^n|x_i^{n-1})$ once for each i (i.e. run the N particles forward in time).

3 Calculate the weights according to (4).

4 We do not automatically accept all particles as members of the new ensemble, but use the Metropolis-Hastings algorithm, as follows.
Choose one particle, e.g. x_1^n as a member of the new ensemble.
Particle 2 is also a member of the new ensemble if either

$$w_2 > w_1 \tag{13}$$

or it is accepted with probability

$$p = \frac{w_2}{w_1} \tag{14}$$

The latter condition is implemented by choosing a random number u from a uniform density on $[0, 1]$, and accepting particle 2 when $u < p$. When it is not accepted we duplicate particle 1.

5 Repeat 4 and 5 using as particle 1 the last chosen particle, and as particle 2 the new particle.

Note that this uses Metropolis-Hastings on the marginal densities (marginal in time) instead of on the full posterior. Also, the samples are not generated using a Markov chain, but the chain is formed by the set of particles.

Regularization

A potential problem with resampling is that several particles will be identical after the resampling step. If the model has no error, these particles will remain identical during forward integration, leading eventually again to filter divergence. Also when the model error is relatively small, or the dynamics of the model is itself degenerate (i.e. a strong low-dimensional attractor is close by), the filter can become degenerate. (On can argue, however, that for realistic atmospheric and oceanic applications model errors are substantial.) To avoid this, some 'jitter' has to be applied to the resampled particles. In the statistical literature this action has be termed *regularization*, and one speaks of regularized particle filters. Several regularization algorithms have been proposed. The central idea is to resample the particles from a continuous representation of the pdf. Typically kernel density smoothers are used, in which the sum of delta functions is replaced by a sum of smooth functions:

$$p(x) = \sum_{i=1}^{N} w_i g_i(x) \tag{15}$$

in which

$$\sum_i w_i = 1 \tag{16}$$

and in which $g_i(x)$ is the continuous pdf centered around particles x_i. Popular choices for $g_i(x)$ are the Gaussian density and the Epanechnikov kernel. The width of these kernels should be as large as possible to avoid degeneracy, but as small as possible to avoid extra statistical noise. One can show from density estimation theory that the optimal choice for the kernel for equal weight particles is the Epanechnikov kernel, given by:

$$g_i(x) = \frac{n+2}{2\gamma}\left(1 - |x - x_i|^2\right) \qquad \text{for} \qquad |x - x_i| < 1 \tag{17}$$

and zero otherwise, with n is the dimension of x and γ is the volume of a unit sphere in R^n.

In practice the sampling from such a mixture density is as follows: first sample from the weight distribution, e.g. via universal sampling explained above, and then sample from the corresponding $g_i(x)$.

Is resampling enough?

We now discuss why resampling the prior pdf is not enough to avoid filter divergence in large-scale applications. The problem is related to the large number of observations, which make the likelihood peak in only a very small portion of the observation space. So, via the likelihood the curse of dimensionality comes back into the problem. The importance of these papers allow us to spend some more time on them. Assuming the observations are identically distributed and independent the weights can be written as:

$$w_i = \frac{p(y|x_i)}{\sum_k p(y|x_k)} = \frac{\Pi_{j=1}^{N_y} p(y_j|x_i)}{\sum_k \Pi_{j=1}^{N_y} p(y_j|x_k)} \tag{18}$$

Now introduce $\phi(x_j) = -\log p(x_j)$ to find:

$$w_i = \frac{\exp\left[-\sum_{j=1}^{N_y} \phi(y_j|x_i)\right]}{\sum_k \exp\left[-\sum_{j=1}^{N_y} \phi(y_j|x_k)\right]} \tag{19}$$

Let us now concentrate on the largest weight. The idea is to show that to keep the largest weight much smaller than 1 the number of particles has to grow exponentially with 'the size of the problem'. We come back to this in a minute. The largest weight can be taken without loss of generality as that of the first member, and by dividing numerator and denominator by the numerator we find for this largest weight:

$$w_1 = \left[1 + \sum_{k=2}^{N} \exp\left(-\sum_{j=1}^{N_y} \phi(y_j|x_k) - \phi(y_j|x_1)\right)\right]^{-1} = \left[1 + \sum_{k=2}^{N} \exp\left(-\tau(S_k - S_1)\right)\right]^{-1} \tag{20}$$

in which τ is the square root of the variance (in particles k) of $\sum_{j=1}^{N_y} \phi(y_j|x_k)$ and S_i the normalized value:

$$S_i = \frac{1}{\tau}\left(\sum_{j=1}^{N_y} p(y_j|x_k) - E\left[\sum_{j=1}^{N_y} p(y_j|x_k)\right]\right) \tag{21}$$

If the observation pdf is gaussian, $\phi(y_j|x_k) = 1/2(y_j - H_j(x_k))^T R^{-1}(y_j - H_j(x_k))$. With these expressions the largest weight can be written as:

$$w_1 = \frac{1}{1+T} \tag{22}$$

in which T is given by:

$$T = \sum_{k=2}^{N} \exp\left[-\tau(S_k - S_1)\right] \tag{23}$$

Filter divergence occurs when T approaches zero, so one has to look at the expectation of T. To find this expectation it is assumed that the S_i are $N(0,1)$ distributed. This is a weak point in the analysis, because a rigorous prove is lacking. However, the end result fits quite well with numerical experiments, so the assumption has serious backup. When the S_i are indeed Gaussian, it is easy to show that as $N \to \infty$ the S_1 tends to $-\sqrt{2\log N}$. After a few manupilations and assumptions the authors arrive at

$$E[T] \approx \frac{\sqrt{2\log N}}{\tau} \tag{24}$$

To prevent the filter from diverging, the number of particles $N >> \exp(\tau^2/2)$. Now recall that τ is the square root of the variance of $\sum_{j=1}^{N_y} \phi(y_j|x_k)$. τ depends on the variability of the prior and on the characteristics of the observations. It can be considered as an effective dimension of the system, but in general it is unclear what the relation to the state dimension is. However, it is reasonable that it will grow with the state dimension, and the ensemble size should grow exponentially with its square. As a rough order of magnitude one might use the number of independent observations for τ^2. So 10 independent observations lead to $N >> 150$, and 50 independent observations need $N >> 10^{11}$. This leads to the conclusion that a particle filter with resampling will diverge for large-scale applications. More is needed, and section c deals with extensions of the simple resampling technique.

2) The Marginal Particle Filter

The Marginal Particle Filter uses the fact that in Importance Sampling the joint-in-time pdf is updated using a new observation, while in the filtering problem we are only interested in the marginal pdf at the analysis time. That Importance Sampling is indeed working on the joint pdf is directly clear when we realize that for each particle all weights from all observations are just multiplied with each other to obtain the weight of the particle at the present time. In the marginal particle filter the past is integrated out. This idea is exploited as follows. The marginal prior pdf can be obtained from the joint pdf by integration:

$$p(x^n|y^{1:n-1}) = \int p(x^n, x^{n-1}|y^{1:n-1})\, dx = \int p(x^n|x^{n-1})p(x^{n-1}|y^{1:n-1})\, dx \tag{25}$$

If we now introduce the particle representation of the pdf at time $n-1$ we find that each particle of the prior pdf at time n can be obtained from

$$x_j \sim \sum_i w_i^{n-1} p(x^n|x_i^{n-1}) \tag{26}$$

in which the $\sim$ denotes that we have to sample from the density at the right-hand-side. The density to sample from is a mixture density. It is hard to sample from the whole density, but relatively easy to draw from the partial densities $p(x^n|x_i^{n-1})$, which is just a model integration including a stochastic term. The way to draw from such a mixture density is to first determine from which partial density we will sample by sampling from the distribution of the weights w_i^{n-1}. This can be done by e.g. universal sampling, as explained before. After that we will draw from the corresponding partial transition density $p(x^n|x_i^{n-1})$.

This is equivalent to the Particle filter with resampling since instead of resampling at n, as with simple resampling, a sample is produced at time $n-1$.

The new ensemble at time n is then confronted with the observations and new weights w^n are obtained from the likelihood, as:

$$w_j^n \propto p(y^n|x_j^n) \tag{27}$$

as usual. It can be shown that this method reduces the noise introduced by the resampling. No applications to geophysical systems have been described yet.

3) Hierarchical models

In some situations it is possible to formulate the full assimilation problem in terms of a sequence of easier-to-handle assimilation problems. The basic idea is the following. Suppose the state vector contains two parts x and z (or x is the vector containing the model variables and z denotes unknown parameters in the model). The pdf of the state vector can be written as:

$$p(x,z) = p(x|z)p(z) \tag{28}$$

If both $p(z)$ and $p(x|z)$ are easy to draw from, this equation allows us to generate samples from $p(x,z)$.

In a particle implementation this decomposition can be used as follows. For example assume that z denote unknown model parameters. Draw N sets of model parameters z_i, and for each parameter set evaluate $p(x|z_i)$, which is in this case a model run with parameters equal to z_i. This then gives us an ensemble of particles drawn from $p(x,z)$.

This is just the way one would in general draw samples from this density. Consider an example in which velocity and pressure observations are available and the model is 'stochastic geostrophic', i.e.

$$\rho f u = \beta_{u1}\frac{\partial p}{\partial x} + \beta_{u2}\frac{\partial p}{\partial y} \tag{29}$$

and a similar equation for v. Clearly, if the field is exactly geostrophic $\beta_{u1} = 0$, and $\beta_{u2} = -1$, but now these two parameters are to be estimated. So in this case we want to estimate:

$$p(\mathbf{u}, p, \beta|y_{\mathbf{u}}, y_p) \sim p(y_{\mathbf{u}}|\mathbf{u})\ p(\mathbf{u}|p,\beta)\ p(\beta)\ p(y_p|p)\ p(p) \tag{30}$$

We assume the densities of the pressure p, and of the β's to be known. So, one starts with drawing samples p_i from $p(p)$, evaluates the likelihoods $p(y_p|p)$, and resamples the p_i to give them equal weight again. Then draw samples for the β's. With these in hand we generate samples of the velocities, using the stochastic geostrophy model. These are then confronted with the observations in $p(y_{\mathbf{u}}|\mathbf{u})$, leading again to a weighted ensemble of velocity, pressure, and parameter values, one for each particle. This ensemble of particles represents the posterior pdf, as we wanted.

Although this sounds simple, it does not work this way in practice since data-reduction techniques like using EOF amplitudes as observations have to be used in practice. Also other smart changes in the formulation of the problem tend to be needed to obtain a stable solution. Furthermore, one must be able to draw samples from the prior densities, in this case pressure and parameters, enforcing relatively simple pdf's for these.

To conclude, hierarchical models provide a systematic way to reduce the complexity of the assimilation problem by highlighting specific dependencies between model variables. This can be very useful, but it is unclear how to exploit this efficiently in high dimensional problems. However, in combination with other techniques it might help solving the particle filter problem, and dismissing particle filters all together is premature.

c. *Improvement of the likelihood*

Related to decreasing the variance of the weights is to make sure that all model integrations end up close to the new observations. First we discuss the application of a proposal density that allows one to sample from a density that is conditioned on the observations, so it is much closer to the observations than the prior density. As an example the EnKF is chosen as the proposal density. Then we discuss the Auxiliary Particle Filter, the Backtracking Particle Filter, and the Guided SIR. In both the Auxiliary Particle Filter and the Guided SIR future observations are used directly to improve the likelihood, while the Backtracking Filter 'tries again'.

1) The proposal density

We are now to discuss a very interesting property of particle filters that has received little attention in the geophysical community. This has to do with the following. Typically, we are interested in evaluating at time n integrals like:

$$\overline{f(x^{0:n})} = \int f(x^{0:n})p(x^{0:n}|y^{1:n})\ dx^{0:n} = \frac{1}{A}\int f(x^{0:n})p(y^{1:n}|x^{0:n})p(x^{0:n})\ dx^{0:n} \tag{31}$$

This integral can be written as:

$$\overline{f(x^{0:n})} = \frac{1}{A}\int f(x^{0:n})p(y^{1:n}|x^{0:n})\frac{p(x^{0:n})}{q(x^{0:n})}q(x^{0:n})\ dx^{0:n} \tag{32}$$

in which $q(x^{0:n})$ is another probability density. Clearly, the support of q has to be equal or larger than that of p to avoid division by zero. The density q is called the *proposal density*, and we could sample from q instead of p to approximate the integral. Using a particle representation of q we obtain for the mean of f in the posterior:

$$\overline{f(x^{0:n})} = \frac{1}{A}\sum_i f(x^{0:n})p(y^{1:n}|x^{0:n})\frac{p(x^{0:n})}{q(x^{0:n})} = \sum_i w_i f(x_i^{0:n}) \tag{33}$$

in which the weights are given by:

$$w_i = \frac{1}{A}p(y^{1:n}|x_i^{0:n})\frac{p(x_i^{0:n})}{q(x_i^{0:n})} \tag{34}$$

in which A is the normalization factor. The interest comes from the fact that we can try to put information of the observations y^n in the density q such that the particles $x_i^{0:n}$, which are sampled from

q, are close to these observations. This opens a whole new area of possible improvements of efficiency of particle filters. We thus have:

$$\overline{f(x^{0:n})} = \sum_i f(x_i^{0:n}) p(y^{1:n}|x_i^{0:n}) \frac{p(x_i^{0:n})}{q(x_i^{0:n}|y^{1:n})} = \sum_i w_i f(x_i^{0:n}) \tag{35}$$

with

$$w_i = \frac{1}{A} p(y^{1:n}|x_i^{0:n}) \frac{p(x_i^{0:n})}{q(x_i^{0:n}|y^{1:n})} \tag{36}$$

To actually use these expressions we would like the method sequetial. This can be done by exploring the Markov property of the evolution, so we use:

$$p(x^{0:n}) = p(x^n|x^{n-1})...p(x^1|x^0)p(x^0) \tag{37}$$

Furthermore, we assume that the observations at different times are independent. (There is no actual need to make this assumption, but it does simplify the expressions considerably.) We thus have, as in Importance Sampling:

$$p(y^{1:n}|x^{0:n}) = p(y^n|x^{0:n})...p(y^1|x^{0:n}) = p(y^n|x^n)...p(y^1|x^1) \tag{38}$$

Finally, we want to explore a Markov property in the proposal density q too to be able to generate a sequential algorithm. A straightforward implementation of this Markov property would be

$$q(x^{0:n}|y^{1:n}) = q(x^n|x^{n-1}, y^{1:n}) \; ... \; q(x^1|x_i^0, y^{1:n}) q(x^0|y^{1:n}) \tag{39}$$

The problem with this form is the dependence of each individual density on all future observations. We do want to explore future observations in the proposal density, as mentioned before, but perhaps not all of them. Also, the algorithm is not sequential in the observations, as can be seen from the following:

$$q(x^{0:n}|y^{1:n}) = q(x^n|x^{n-1}, y^{1:n}) q(x^{0:n-1}|y^{1:n}) \tag{40}$$

A simple way out is to allow only a given set of future observations to be involved, so one or perhaps two. Note that this does not affect the dependence of the posterior pdf on all future observations. As an example, let us use the observations one step into the future, so:

$$q(x^{0:n}|y^{1:n}) = q(x^n|x^{n-1}, y^{1:n}) q(x^{0:n-1}|y^{1:n-1}) \tag{41}$$

leading to

$$q(x^{0:n}|y^{1:n}) = q(x^n|x^{n-1}, y^{1:n}) q(x^{n-1}|x^{n-2} y^{1:n-1}) \; ... \; q(x^1|x^0, y^1) q(x^0) \tag{42}$$

Using this in (35) gives:

$$\overline{f(x^{0:n})} = \sum_i f(x_i^{0:n}) p(y^n|x_i^n)...p(y^1|x_i^1) \frac{p(x_i^n|x_i^{n-1})...p(x_i^1|x_i^0)p(x_i^0)}{q(x_i^n|x_i^{n-1}, y^{1:n})...q(x_i^1|x_i^0, y^1)q(x_i^0)} = \sum_i w_i f(x_i^{0:n}) \tag{43}$$

with weights:

$$w_i = p(y^n|x_i^n)...p(y^1|x_i^1) \frac{p(x_i^n|x_i^{n-1})...p(x_i^1|x_i^0)p(x_i^0)}{q(x_i^n|x_i^{n-1}, y^{1:n})...q(x_i^1|x_i^0, y^1)q(x_i^0)} \tag{44}$$

This equation can be made more transparent by introducing a time index on the weights :

$$w_i^j = p(y^j|x_i^j)\frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^{1:j})} \tag{45}$$

so that

$$w_i = \Pi_{j=0}^n w_i^j \tag{46}$$

where the sequential nature of the algorithm becomes transparent again. It consists of drawing a sample from $q(x^0)$, drawing a sample from $q(x^1|x_i^0, y^1)$ for each i which is propagating the ensemble forward to time 1. The weights for each member i are then calculated as w_i^1. Then a draw is made from $q(x^2|x_i^1, y^{1:2})$ for each i (model integration), followed by calculation of w_i^2 for each member i, etc.

Example: the EnKF as proposal

As an example we will explore this technique with the Gaussian of the EnKF as the proposal density. First we have to evaluate the prior transition density. Since we know the starting point of the simulation, x_i^{n-1}, and its end point, the posterior EnKF sample x_i^n, and we know the model equation, written formally as:

$$x_i^n = f(x_i^{n-1}) + \epsilon_i^n \tag{47}$$

we can determine ϵ_i^n from this equation directly. We also know the distribution from which this ϵ_i^n is supposed to be drawn, let us say a Gaussian with zero mean and covariance Q_m. We then find for the transition density:

$$p(x_i^n|x_i^{n-1}) \propto \exp\left[-1/2\left(x_i^n - f(x_i^{n-1})\right) Q_m^{-1}\left(x_i^n - f(x_i^{n-1})\right)\right] \tag{48}$$

This will give us a number for each $[x_i^{n-1},\ x_i^n]$ combination.

Let us now calculate the proposal density $q(x_i^n|x_i^{n-1}, y^n)$. This depends on the ensemble Kalman filter used. For the Ensemble Kalman filter with perturbed observations the situation is as follows. Each particle in the updated ensemble is connected to those before analysis as:

$$x_i^n = x_i^{n,old} + K^e\left(y + \delta_i - H(x_i^{n,old})\right) \tag{49}$$

in which δ_i is the random error that has to be added to the observations in this variant of the ensemble Kalman filter. K^e is the ensemble Klman gain, i.e. the Kalman gain using the prior error covariance calculated from the prior ensemble. The particle prior to the analysis comes from that of the previous time step through the stochastic model:

$$x_i^{n,old} = f(x^{n-1}) + \epsilon_i^n \tag{50}$$

Combining these two gives:

$$x_i^n = f(x_i^{n-1}) + \epsilon_i^n + K^e\left(y + \delta_i - H(x_i^{n-1}) - H(\epsilon_i^n))\right) \tag{51}$$

or

$$x_i^n = f(x_i^{n-1}) + K^e\left(y - H(f(x_i^{n-1}))\right) + (1 - K^e H)\epsilon_i^n + K^e\delta_i \tag{52}$$

assuming that H is a linear operator. The right-hand side of this equation has a deterministic and a stochastic part. The stochastic part provides the transition density going from x_i^{n-1} to x_i^n. Assuming both model and observation errors to be Gaussian distributed and independent we find for this transition density:

$$q(x_i^n|x_i^{n-1}y^n) \propto \exp\left[-1/2\left(x_i^n-\mu_i^n\right)^T \Sigma_i^{-1}\left(x_i^n-\mu_i^n\right)\right] \tag{53}$$

in which μ_i^n is the deterministic 'evolution' of x, given by:

$$\mu_i^n = f(x_i^{n-1}) + K^e\left(y - H(x_i^{n-1})\right) \tag{54}$$

and the covariance Σ_i is given by:

$$\Sigma_i = (1-K^eH)Q_m(1-K^eH)^T + K^eRK^{eT} \tag{55}$$

where we assumed that the model and observation errors are uncorrelated. It should be realized that x_i^n does depend on all $x_j^{n,old}$ via the Kalman gain, that involves the error covariance P^e. Hence we have calculated $q(x_i^n|P^e, x_i^{n-1}, y^n)$ instead of $q(x_i^n|x_i^{n-1}, y^n)$, in which P^e depends on all other particles. The reason why we ignore the dependence on P^e is that in case of an infinitely large ensemble P^e would be a variable that depends only on the system, not on specific realizations of that system. This is different from the terms related to x_i^n, that will depend on the specific realization for ϵ_i^n even when the ensemble size is 'infinite'. (Hence another approximation related to the finite size of the ensemble comes into play here and at this moment it is unclear how large this approximation error is.)

The calculation of $p(x^n|x^{n-1})$ and $q(x_i^n|x_i^{n-1}y^n)$ look like very expensive operations. By realizing that Q_m and R can be obtained from the ensemble of particles, computationally efficient schemes can easily be derived.

We can now determine the full new weights. Since the normalization factors for the transition and the posterior densities are the same for all particles the weights are easily calculated. The procedure now is as follows:

1 Run the ensemble up to the observation time

2 Perform a (local) EnKF analysis of the particles

3 Calculate the relative weights $w_i^* = p(x_i^n|x_i^{n-1})/q(x_i^n|x_i^{n-1}y^n)$

4 Calculate the relative weights $w_i = p(y^n|x_i^n)$

5 Calculate the full relative weights as $w_i = w_i * w_i^*$ and normalize them.

6 Resample

It is good to realize that the EnKF step is only used to draw the particles close to the observations. This means that when the weights are still varying too much, one can do the EnKF step with much lower observational errors. This might look like overfitting but it is not since the only thing we do in probabilistic sense is to generate particles to those positions in state space where the likelihood is large.

Example: the Marginal Particle filter with proposal

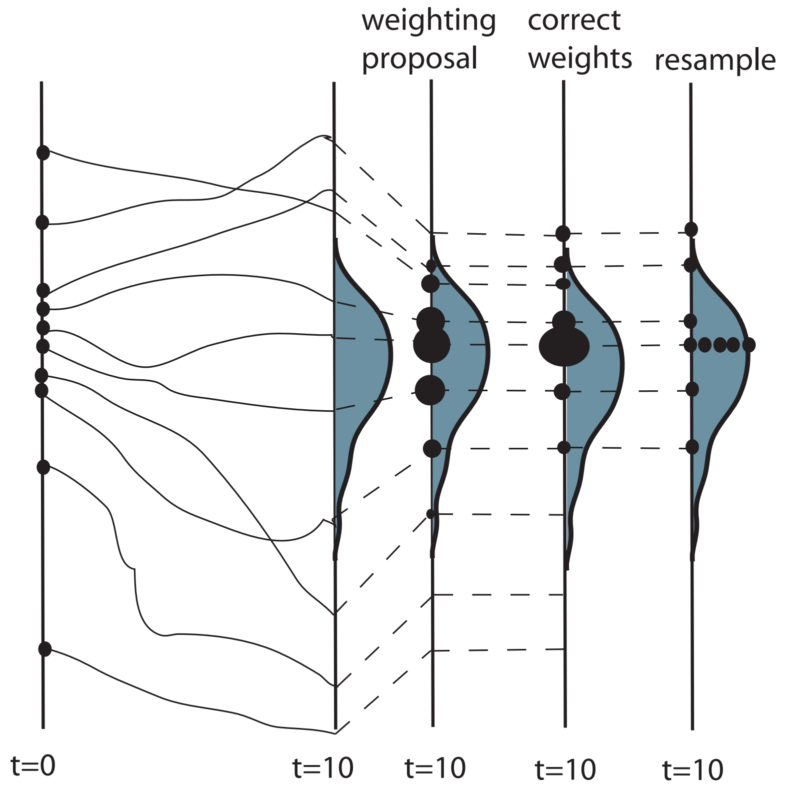

FIG. 3. *The particle filter with proposal density. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. At time 10 the particles are brought closer to the observations by using e.g. the EnKF. Then they are weighted with the likelihood and these weights are corrected for the artificial EnKF step.*

As another example the Marginal Particle Filter with proposal density is discussed briefly. The Marginal Particle Filter can easily be expanded by utilizing the proposal density as :

$$x_j \sim \sum_i w_i^{n-1} q(x^n | x_i^{n-1} y^n) \tag{56}$$

and forming weights at time n as:

$$w_j^n \approx p(y^n | x_j^n) \frac{\sum_i w_i^{n-1} p(x^n | x_i^{n-1})}{\sum_i w_i^{n-1} q(x^n | x_i^{n-1} y^n)} \tag{57}$$

2) Auxiliary Particle Filter

In the auxilary particle filter the ensemble at time $n-1$ is weighted with information of the likelihood at time n. In this method one generates a representation of each particle at the time of the new observation, e.g. by integrating each particle from time $n-1$ to time n using zero model noise. (Depending on the complexity of the stochastic model integrator this can save considerable time.) Then the particles are weighted with the observations, and a new resampled ensemble is integrated from $n-1$ to arrive closer to the observations. A flow chart reads (see figure 4):

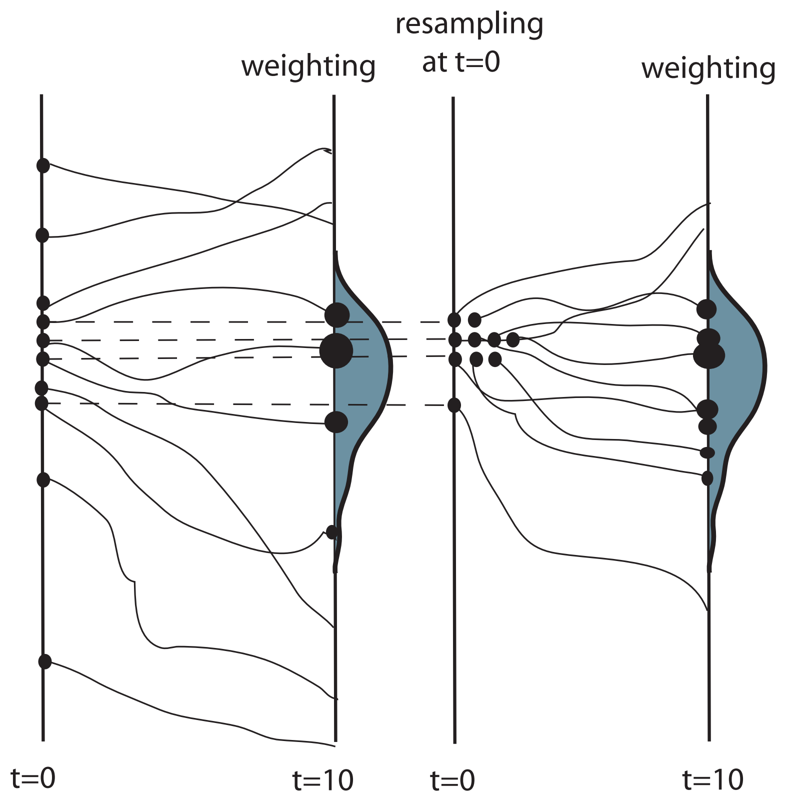

Fig. 4. *The Auxiliary Particle Filter. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. At time 10 the particles are weighted according to the likelihood. These weights are used at time 0 to rerun the ensemble up to time 10.*

1 Integrate each particle from $n-1$ to n with simplified dynamics (e.g. without model noise), producing the a representation of the proposal density $q(\psi^n)$ at the measurement time.

2 Weight each particle with the new observations as

$$w_i \propto q(y^n|x_i^n) \tag{58}$$

These weights are called the 'first-stage weights' or the 'simulation weights'.

3 Resample the particles at time $n-1$ with these weights, and use this ensemble as a representation of the proposal density by integrating it forward to n with the full stochastic model.

4 Re-weight the members with weights

$$w_i = \frac{1}{A}\frac{p(y^n|x_i^n)}{q(y^n|x_i^n)} \tag{59}$$

in which A is the normalization factor. A resampling step can be done, but is not really necessary because the actual resampling is done at step 3.

It should be noted that 2N integrations have to be performed with this method, one ensemble integration to find the proposal, and one for the actual pdf. If adding the stochastic noise is not expensive step 1 can be done with the stochastic model, which comes down to doing Sequential Importance Resampling twice. One can imagine to do it even more times, zooming in into the likelihood, but at a cost of performing more and more integrations of the model. Figure 4 displays how the method works.

The name 'auxiliary' comes from the introduction of a member index in the original formulation, but that is not needed explicitly. Implicitly, the member index keeps track of the relation between the first-stage weights and the particle sample at $n-1$, so that a formal treatment of the characteristics of the method can be investigated.

3) The Backtracking Particle Filter

One possibility to solve the problem of filter divergence (i.e. when all members obtain very low weights, such that none of the particles follows the observations) is to go back to the time when the particle filter did work properly, and try again. This 'trying again' can be done in several ways, and we describe four methods to do so.

The first idea is to duplicate each particle at time $n-b$ when the filter performed fine, and propagate $2N$ particles up to time $n+1$, one time step after the divergence. Then the filter proceeds with the original N particles until a new divergence is detected.

Another possibility, called cloud expansion, is to go back to $n-b$ and just rerun the N particles with different realizations of the model noise, or, if that doesn't work, add N particles each with zero mean Gaussian random noise centered around the original N particles. These $2N$ particles are then used until time $n+1$, after which one proceeds again with N particles.

A more serious change is directed doubling, in which one goes back to $n-b$, and uses the ensemble at that time before resampling. Take the M particles with the largest weights (typically, $M = N/10$) and add new particles along the line between each of these particles and the observations. (With more than one observation at a time this becomes more complicated, but one can probably generate efficient schemes that do this in an approximate way.) To preserve the mean of the ensemble, add the same number of particles on the same line at the other side of each of the M original particles. The number of extra particles is such that the total size of the new ensemble becomes $2N$. This ensemble is propagated up to time $n+1$, after which the size is brought back to N. A potential problem with this method in high-dimensional systems with implicit balances is that it is unclear how to generate these new particles because they will not all be close to an original particle.

Finally, one could use doubling with perturbed observations, in which one goes back to $n-b$, perturb the observations $2N$ times, and calculate $2N$ particles using Bayes. This step reminds us of the perturbed-observation implementation of the EnKF. Then these $2N$ particles are used up to time $n+1$, after which one returns back to N partilces.

A possible disadvantage is that the filters typically need $2N$ model integrations, similar to the auxilary particle filter.

4) The Guided SIR

In this method we use the fact that observations are available not at every time step, but at say every L time steps. This means that the model is stepping forward L time steps before new observations are available. This is the common situation in geophysical systems. The idea is to weight and resample the ensemble of particles using future observations at the present time. This will guide the particles into the direction of these future observations. To avoid drawing the particles to the observations too strongly too early, one increases the error covariance of the observations by a factor > 1, typically a factor 10-100 is used. A possible implementation is as follows (see figure 5):

1 Run the particles from the previous analysis time $n - L$ to some predefined time $n - L + k$ in between measurement times, in which $k < L$.

2 Calculate the weights using the observations of the next measurement time with the observational error covariance multiplied by e.g. $100(L - k)/L$, as

$$w_i \propto p(y^n | x_i^{n-L+k}) \qquad \text{with} \qquad \sum_i w_i = 1 \tag{60}$$

3 Resample the particles according to these weights. So we have made sure that the particles are on the good track towards the next observations.

4 Re-weight this new ensemble to compensate for the extra weights that have been introduced in step 3, *but do not resample*, as:

$$\alpha_i \propto \left[p(y^n | x_i^{n-L+k})\right]^{-1} \qquad \text{with} \qquad \sum_i \alpha_i = 1 \tag{61}$$

So now each particle i has weight α_i, but is still on the good track towards the observations.

5 Propagate this ensemble forward in time to the next predefined time $n - L + 2k$, again before the actual measurement time (so also $2k < L$).

6 Repeat 2, using as weights

$$w_i \propto \alpha_i p(y^n | x_i^{n-L+2k}) \qquad \text{with} \qquad \sum_i w_i = 1 \tag{62}$$

with the observational error covariance multiplied by e.g. $100(L - 2k)/L$.

7 Repeat 3,4, 5, 6 until the ensemble reaches the true measurement time.

8 Calculate the weights at the measurement time, as

$$w_i \propto p(y^n | x_i^n) \qquad \text{with} \qquad \sum_i w_i = 1 \tag{63}$$

9 Resample the particles according to these weights, i.e. without corrections, since they are not needed at this step.

Note that all we did in this procedure is to guide the particles towards the observations via resampling at intermediate steps (steps 3). In order to avoid a bias we compensated for the extra weights introduced by the resampling (steps 4).

A few comments:

1 One could do a resampling at step 4 (Arnaud Doucet, personal comunication)

2 One can choose k differently in each cycle of step 7.

3 One can use modifications of the true observations at time n. For instance, when a daily cycle is present and observations are only once a day, one could introduce a daily variation in the pseudo observations too. Note that any pseudo observation can be used, as long as it brings the ensemble closer to the true observations at time n, and the weights are adjusted accordingly.

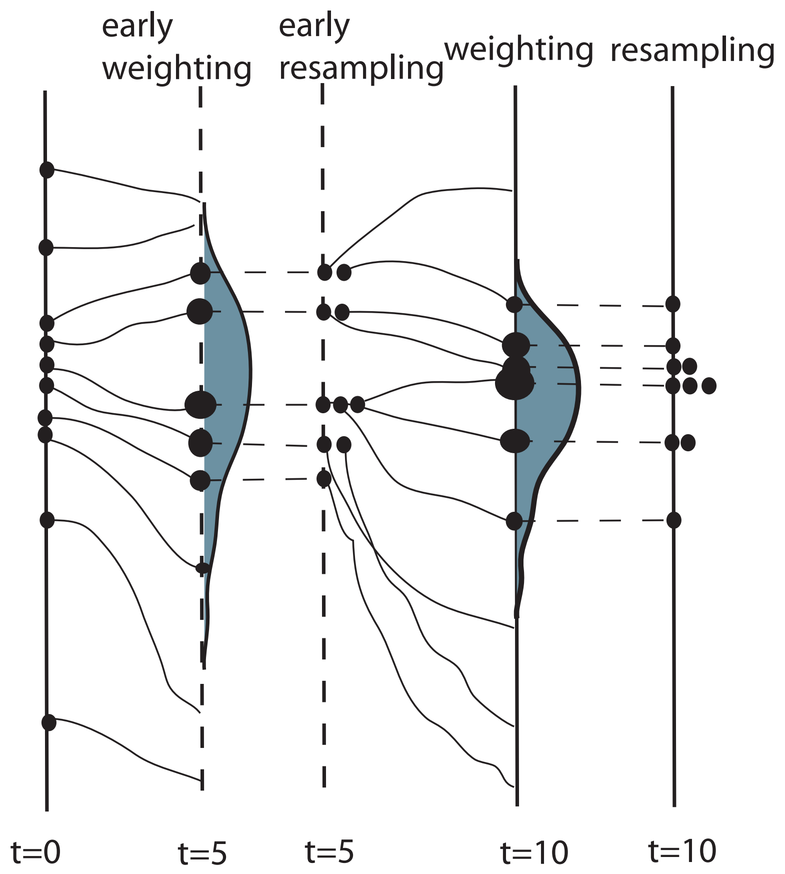

FIG. 5. *The Guided Particle Filter. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. Pseudo observations are introduced at time 5. These are equal to those at time 10, but with larger observational errors, so wider likelihood. The particles are weighted with that likelihood and resampled. At time 10 the particles are weighted according to the true likelihood, and these weights are corrected for the artificial weights from the pseudo observations at time 5.*

d. *Approximations*

Even with all ideas from the statistical literature presented above, it is difficult to avoid filter degeneracy in geophysical problems with state vectors of size 1 million or larger. To this end, several approximation to especially the Sequential Importance Resampling filter have been proposed. We discuss here the Merging Particle Filter, Localization, as used in e.g. the Ensemble Kalman Filter, Kernel dressing, in which eacch particle is 'dressed' with a - usually Gaussian - pdf, and the Maximum Entropy Particle Filter.

Also, a consideral number of papers have appeared recently that approach the problem from the EnKF side, that is, one tries to introduce non-Gaussian elements in the EnKF. Unfortunately, the adaptations to the EnKF or its square-root variants are rather ad hoc. Nonetheless, interesting results have been achieved that are of interest in trying to solve the full nonlinear problems. For instance, one changes the shape of the prior pdf from a Gauss to a pdf with much broader tail. The standard Kalman-filter update equations do not hold, but instead a 3DVar formulation is used that can handle non-Gaussian priors.

1) The Merging Particle Filter

In this method linear combinations of the particles are taken at the measurement time to reduce the variance in the weights. These linear combinations are taken such that the mean and the covariance of the ensemble are kept, at the costs of a less correct description of the higher-order moments. In this way the algorithm is still variance-minimizing, unlike e.g. the Ensemble Kalman Filter.

The procedure is as follows (see figure 6):

1) Sample N particles x_i from the initial model probability density $p(x^{n-1})$ (or have them from a previous assimilation step.

2) Integrate all particles forward in time up to the measurement time (so, sample from $p(x^n|x_i^{n-1})$ for each i)

3) Calculate the weights according to (4) and attach these weights to each particle. Note that the particles are not modified, only their relative weight is changed!

4) Re-sample the particles such that the weights are equal to $1/N$. So far, the algorithm is identical to the SIR.

5) Repeat 4) $M-1$ times, with $M>2$ to form M ensembles of size N.

6) Merge M particles:

$$x_i^n = \sum_{j=1}^{M} \alpha_j x_{j,i}^n \tag{64}$$

in which $x_{j,i}^n$ denotes particle $i \in [1,2,..,N]$ in ensemble $j \in [1,2,..,M]$. To ensure the correct mean and covariance of the ensemble choose

$$\sum_{j=1}^{M} \alpha_j = 1 \quad \text{and} \quad \sum_{j=1}^{M} \alpha_j^2 = 1 \tag{65}$$

as is easily verified.

7 Back to step 2.

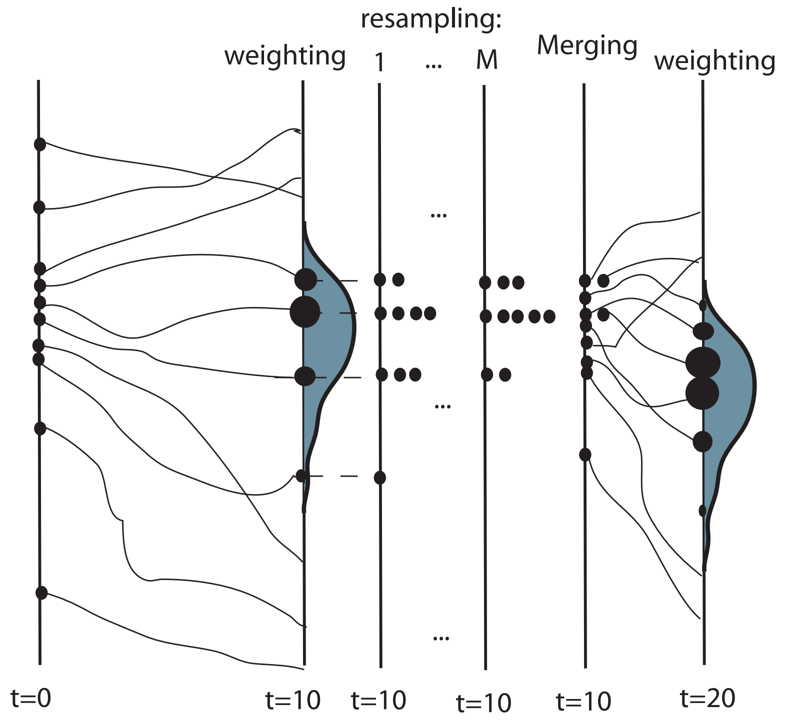

FIG. 6. *Merging Particle Filter. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. The particles are resampled according to the true likelihood M times at time t=10. The new particles are averages over M of these resampled particles. Note that the new particles are not identical copies of the old ones, so 'new blood' enters the ensemble.*

It has been argued that for strongly non-Gaussian pdf's the merging particle filter will perform less because only the mean and the covariance of the ensemble are optimized, at the cost of higher-order moments. The difference with the EnKF lies in the fact that in the EnKF approximations are made *before* the analysis step. In the merging particle filter approximations are made after the analysis step, and only the mean and the covariance remain unchanged (up to sample noise). Hence the merging particle filter is truly variance minimizing, while the EnKF and its variants are not.

Finally, note that the positions of the particles in state space have changed, unlike a particle filter with direct resampling. This is both good and bad. Good because the spread in the ensemble is enhanced, bad because nonlinear balances in each particle are destroyed, or at least strongly perturbed.

2) Gaussian Resampling

It has been proposed to use Gaussian resampling from the posterior pdf as obtained from the particle filter. It is interesting to compare this idea to the Merging Particle Filter. In the merging particle filter the $\alpha's$ have to be determined analytically. In Gaussian resampling new particles have to be drawn from the covariance of the weighted ensemble of particles. Clearly, the latter is much more expensive. Gaussian Resampling is related to the Ensemble Transform Kalman Filter (ETKF), in which the new ensemble is generated in a similar way. Also, Gaussian Resampling in combination with weights from the particle filter is superior to EnKF-like methods because the posterior is truly variance minimizing, while the EnKF-like methods are not. On the other hand, the particles are changing position in state space, but not towards the observations. In EnKF-like methods all particles tend to be drawn towards the observations. (Note that this is not necessarily true for multi-modal likelihoods.)

3) Weights from reduced-dimension models

To avoid large variance in the weights one can reduce the dimension of the assimilation problem by performing the importance-sampling step in a reduced space. For instance, one can use the dominant EOF's of the model system and perform the importance sampling on the EOF amplitudes of the particles, given the EOF amplitudes of the observations. The procedure is to first determine the dominant EOF's of the system, e.g. from a long model run, or from a long observational data set, or from the ensemble of particles at the observation time. Then run the particles up to observation time, determine the EOF amplitudes of both particles and observations, do the importance sampling on the EOF amplitudes, and use these weights as the weights of the full model particles.

More realistic model reduction might be possible using dynamical modes from dynamical system theory. This is one of the areas where data assimilation and dynamical system theory meet directly, and this author expects fruitful collaboration.

4) Localization

It is common practice that also the Ensemble Kalman Filter (EnKF) shows degeneracy when the state space size is of order 1 million or more. A remedy used in that method is to update locally, i.e. to update the state vector at a certain grid point only the observations in the immediate neighborhood are included in the analysis step. This operation is called localization, and is common practice nowadays in meteorology and oceanography, see the module on Operational Data Assimilation.

The rationale for this procedure comes from the idea that an observation at one location is not expected to have anything to do with the state of the system at the other side of the globe. This neglected correlation can be wrong for e.g. very long-scale phenomena, like long-wavelength waves, and one can imagine to do localization in e.g. spectral space, or in the space spanned by EOF's.

One advantage of localizations are that spurious long-range correlations can be suppressed. Due to the relatively small ensemble size we can typically afford, it is hard to obtain zero correlation when the actual correlation is zero. So, correlations close to zero are suspicious, and one can argue to ignore them altogether. Another advantage is that since the update of the ensemble is performed over only part of the full model domain, the size of the state vector is strongly reduced, or the effective ensemble size is greatly increased. Finally, in a direct application of the EnKF without localization a new ensemble which consists of ensemble members that are (almost) a linear combination of the old members, and the space spanned by the ensemble remains the same. Using localization gives 'new blood' in the ensemble because new members consist of different linear combinations of old members in different parts of the domain.

These advantages motivate the search for a localization procedure in particle filtering. One can calculate the weights locally, i.e. make the weights space dependent and use only observations close to the spacial point. So, it is easy to localize the Importance Sampling filter. This is less so for more sophisticated filters that employ resampling. In the resampling step low-weight particles are abandoned and high-weight particles are duplicated. However, with local weights, different particles are selected in different parts of the domain. The problem now is that we have to have continuous (in space) model fields to propagate forward in time with the model equations. Just constructing a new particle that consists of one particle in one part of the model domain and another particle in another domain will lead to problems at the boundary between these two. This is less of a problem in the EnKF because the error covariances used there tend to smooth the solution in space. In e.g. the SIR this smoothing is not present.

The above problem is not present in parameter estimation. In most model systems parameters can be discontinuous without causing serious problems. When model fields are estimated the smoothness issue is very serious, Several methods to avoid this problem come to mind, which we discuss briefly below.

Spatial interpolation

The problem with spatial interpolation is that we have a-priori no idea which particles one has to put together in different parts of the domain to generate one new particle over the full domain, so that potentially very large gradients can occur, and interpolation will still yield 'wild' model fields. So, unless we find efficient ways to generate new global particles which are as smooth as possible given the favorite particles in different parts of the domain this does not work.

Using the pdf

One can try to use the joint pdf between neighboring points to generate a transition from one particle to the other as smooth as possible. The smoothing procedure is as follows:

1) Start from a point somewhere in the domain and choose the particle with highest weight. Decrease that weight by $1/N$ (because the particle is chosen deterministically).

2) Walk to a next grid point and use the same particle if the weight is still high (i.e. $w_i > 1/N$). If the weight is low $w_i < 1/N$ choose another particle there which has high weight, and is close to the original particle at the previous grid point.

3) Repeat 2) until all grid points have been visited and a particle is attached to each grid point. This is the first new member.

4) Repeat 1), 2) and 3) until N particles have been created.

Experience learns that this procedure works nicely for the first few new particles, but generates less smooth particles towards the end, because less particles can be chosen from at a breakpoint where w_i becomes smaller than 1.

Using an approximate smooth solution

As mentioned above, one needs guidance on how to choose the different particles in different parts of the domain to generate smooth global particles. One could use the another approximate solution that is smooth to provide this guidance, for example the EnKF solution. In that case step 2) above is replaced by

2') Walk to a next grid point and use the same particle if the weight is still high (i.e. $w_i > 1/N$). If the weight is low $w_i < 1/N$ choose another particle there which has high weight, and is close to one chosen EnKF member.

One could choose this EnKF member have highest global weight for the first global particle, and move to the EnKF member lower in weight rank for the next global particle etc. The advantage of this procedure is that when the likelihood of all members is very low in a certain part of the domain one can fall back on the EnKF solution, which is likely to be closer to the observations there.

Discontinuous forcing

Instead of trying to fix discontinuous state vectors one can try to put the discontinuity in the forcing. Because the forcing is integrated at each time step the influence on the model fields themselves is rather smooth. This could be implemented as follows:

1) Integrate the particles forward from time $n-1$ to the new observations at time n. Make sure that the random forcing used in each particle is either stored or repeatable.

2) Do local weighting, and attach these weights to the random forcing field used between time $n-1$ and time n.

3) Resample the used forcing fields, and use these to integrate the particles again from time $n-1$ to time n.

This method assumes that the random forcing fields are causing the large weight variance, and not the particles at $n-1$. The latter is changed by e.g. the Auxiliary Particle Filter. The method has in common with that method that is needs $2N$ model integrations. In fact, one can imagine a combination of these two. An advantage is that a smooth transition in time is present at measurement times, so the method can be considered a smoother in that aspect.

5) Kernel dressing

Another approximation is to 'dress' each particle with a continuous pdf to approximate the true continuous pdf using a standard kernel technique. Usually a Gaussian pdf is chosen since that has only two parameters to be estimated. In the Gaussian case the mean of the Gaussian is the particle state, and the covariance can be taken as a factor smaller than one times the covariance of the full ensemble. This results in a so-called Gaussian mixture model.

The idea is to represent the prior pdf as:

$$p(x) = \sum_{i=1}^{N} \frac{1}{N} N(x_i, \Sigma_i) \tag{66}$$

The $N(x_i, \Sigma)$ stands for the Gaussian (Normal) density with as mean particle i, x_i, and covariance Σ_i. Efficient scheme's to determine these covariances are difficult to find. In one solution all Σ_i are takes as a factor α times the full covariance of the prior ensemble. The factor α is taken as a tuning parameter that takes care of the fact that the spread of the Gaussian pdf around each particle is smaller than that of the full ensemble, while at the same time it should prevent ensemble collapse by the filter being overconfident, showing up as a too small prior error covariance.

Using Bayes, and assuming Gaussian observational errors, leads to a posterior pdf which is again a sum of Gaussian pdf's. It is easy to show that the posterior pdf can be written as

$$p(x|y) = \sum_{i} c_i N(\nu_i, \Sigma_{new}) \tag{67}$$

where ν_i is the mean of the new Gaussian pdf i, Σ_{new} is the new posterior covariance (identical for each Gaussian), and c_i is a normalization constant that comes from the product of each prior Gaussian with the likelihood. c_i is given by:

$$c_i = (\frac{1}{2\pi)^{d/2}|\alpha\Sigma + R|} \exp\left[-\frac{1}{2}(y - H(\psi)_i)^T(\alpha\Sigma + R)^{-1}(y - H(\psi_i))\right] \tag{68}$$

in which R is the error covariance of the observations. The relative weight of each Gaussian in the posterior pdf is given by this c_i as:

$$w_i = \frac{c_i}{\sum_j c_j} \tag{69}$$

Interestingly, these weights remind us of simple importance sampling (4), but now the prior error covariance (multiplied by a factor α) is added to the error covariance of the observations. This means that the likelihood has been broadened by the extra prior error covariance, and filter degeneracy is less likely to occur. The procedure is depicted in figure 7.

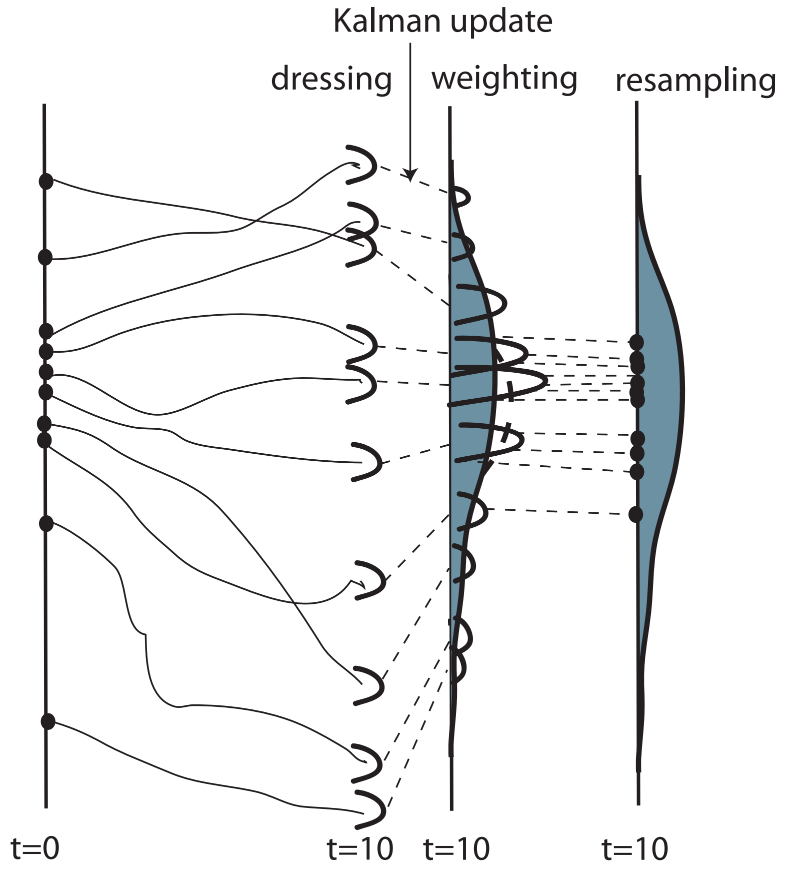

FIG. 7. *Kernel dressing. At analysis time each particle is 'dressed' with a Kernel. e.g. a Gausssian. Then each kernel is updated using the Kalman filter update. Next step is weighting of the kernel with the likelihood of the Kalman posterior. Finally, samples are drawn from this new posterior density. Note that the new particles are not identical copies of the old particles, bur drawn from the continuous posterior pdf.*

To understand better what happens consider the simple scalar case in which the prior is distributed as $N(\mu, P)$ and the likelihood is $N(y, R)$. Bayes tells us that the posterior is obtained by multiplying these two. Concentrating on the exponents we have:

$$\frac{(x-\mu)^2}{P} + \frac{(y-x)^2}{R} = \tag{70}$$

$$\left(\frac{1}{P} + \frac{1}{R}\right) x^2 - 2\left(\frac{\mu}{P} + \frac{y}{R}\right) x + \frac{\mu^2}{P} + \frac{y^2}{R} = ... = \tag{71}$$

$$\frac{(x-\mu_n)^2}{P_n} + \frac{(y-\mu)^2}{R_n} \tag{72}$$

in which

$$\mu_n = \frac{R\mu}{P+R} + \frac{Py}{P+R} \tag{73}$$

and

$$\frac{1}{P_n} = \frac{1}{P} + \frac{1}{R} \tag{74}$$

and finally

$$R_n = P + R \tag{75}$$

Hence, as function of x the pdf has smaller variance, while as function of the observations y the pdf is wider. So the posterior pdf around each particle is narrower than the prior pdf around that particle, and the likelihood is wider. Furthermore, the posterior mean has moved towards the observation. Note that, since the prior variance is smaller than the variance of the full ensemble, the prior is not affected that much by the observation. That is taken care of by the relative weights of the individual Gaussians in the mixture. In a true particle filter the prior pdf is a sum of delta functions, so the prior variance of each particle P is infinitely small. In that case is the posterior pdf remains a delta function, the mean does not move, and the likelihood does not change. Herewith we can understand the smooth transition from continuous representations of the pdf to particle representations.

The new ensemble that represents the posterior pdf can be obtained by first drawing randomly from the distribution of w_i, i.e. choose a specific Gaussian, and than draw a new particle randomly from that specific Gaussian, the standard technique for mixture densities.

A few problems exists when trying to apply this method to large-scale problems. First, manipulating the large error covariances becomes problematic, and second, sampling from a large-dimensional pdf, even if it is a Gaussian, is hard.

One can combine this approach with localization to try to solve these problems. Furthermore, they estimate local (in state space) error covariances around each prior particle, instead of using the same fraction of the full error covariance. This can be implemented by using the following steps:

1) First choose randomly L particles (typically 10-40) from the full N (typically 100) ensemble to represent the centers of the Gaussians in the Gaussian mixture.

2) Choose the M nearest particles (typically 25) to each center to determine the local (in state space) error covariance for that Gaussian. One can use the Euclidian distance, but other choices can be made.

3) Calculate the weights w_i according to (69).

4) Choose randomly one of the Gaussians from the distribution of w_i.

5) From that Gaussian choose one of the M members with probability $1/M$, call that member ψ_I and update each of the M members similarly to the update used in the EnKF:

$$x_i^{new} = x_i + K(y + \epsilon_i - H(x_I)) \tag{76}$$

in which ϵ_i is chosen from $N(0, R)$.

6 Repeat 4 and 5 until N new members have been chosen.

With this approach, the computational efficiency is similar to that of the EnKF. Localization can now be implemented by using the above procedure at each grid point, using only observations close to that grid point by putting correlations over large spatial differences equal to zero. A problem with this approach is, however, that since different Gaussians are chosen at different grid points the new global particles will not be smooth. Note that we have encountered the similar problem with the resampling scheme with localization. One could for instance solve this problem as follows. First, each particle is spitted in two parts, one inside $x^{(i)}$ and one outside $x^{(o)}$ the local neighborhood used in the localization. The posterior density for state vector x can be written as:

$$p(x|y) = p(x^{(i)}|x^{(o)}, y)p(x^{(o)}|y) \tag{77}$$

It has been proposed to use the nonlocal EnKF particles to sample $p(x^{(o)}|y)$, and take a linear combination of that particle with the local particle obtained with the local Gaussian mixture model inside the local neighborhood. In this way a hybrid method is obtained that to some extend preserves non-Gausianity locally.

6) The Maximum Entropy Particle Filter

In the Maximum Entropy Particle Filter one tries to incorporate the pdf of the model without observations in the particle filter. In this way the degrees of freedom in the data-assimilation problem can greatly be reduced, and the potential of filter divergence might reduce too. The basic idea is that the pdf without observations is given by a background pdf q. One can take for q a Gaussian mixture, but there is no need to do so. The background pdf is used to describe the 'model climatology'. So, instead of Gaussian mixture models that dress each particle with a separate pdf, one tries to describe the model climatology by this background pdf.

Also in maximum entropy filters an ensemble of particles is run up to the next observation. In the EnKF and ESRKF the assumption is that the prior is a Gaussian, and the analysis is performed as if the pdf is Gaussian. In the maximum entropy filter the assumption is that the prior pdf is as close as possible to q, but at the same time it enforces that the mean and the covariance of the pdf in observation space is obtained directly from the ensemble. Actually, it is easy to show that when q is a simple Gaussian, the maximum entropy filter is just an ensemble Kalman filter. To determine the prior pdf from the ensemble we do the following. The closeness to q is expressed as the relative entropy:

$$E(p, q) = \int p \log\left(\frac{p}{q}\right)\, dx \tag{78}$$

Further we use the constraints that p is a pdf that integrates to 1:

$$\int p(x)\ dx = 1 \tag{79}$$

that the mean of the observation operator η is found from the ensemble:

$$\int H(x)p(x)\ dx = \frac{1}{N}\sum_i H(x_i) = \eta \tag{80}$$

and that the covariance *function* of the observation operator Σ is found from the ensemble:

$$\int H(x)H^T(x)p(x)\ dx = \frac{1}{N-1}\sum_i H(x_i)H^T(x_i) = \Sigma \tag{81}$$

The constraints are build into the minimalization through Lagrange multipliers θ, λ, and Λ, so we minimize

$$\tilde{E}(p,q) = \int p \log\left(\frac{p}{q}\right)\ dx + \theta\left(1 - \int p(x)\ dx\right) + \tag{82}$$

$$+ \lambda\left(\eta - \int H(x)p(x)\ dx\right) + \Lambda^T\left(\Sigma - \int H(x)H^T(x)p(x)\ dx\right) \tag{83}$$

with respect to p, θ, λ, and Λ. This leads to:

$$\int \left(\log\frac{p}{q} + 1 + \theta + \lambda H(x) + \frac{1}{2}H(x)\Lambda H^T(x)\right)\delta p\ dx = 0 \tag{84}$$

so that

$$p(x,\lambda,\Lambda) = \frac{1}{Z(\lambda,\Lambda)}\exp\left[\lambda H(x) + \frac{1}{2}H(x)\Lambda H^T(x)\right] q(x) \tag{85}$$

in which Z is the normalization factor. The Lagrange multipliers are found from maximizing the expected value of p, or rather $\log p$, so

$$max\left[\eta\lambda + \frac{1}{2}\Sigma^T\Lambda - \log Z\right] \tag{86}$$

Now that we have found the prior pdf, the analysis is discussed. The idea is to use Bayes theorem to update λ and Λ, which, for Gaussian error statistics for the observations and the maximum entropy density (85), leads us to:

$$\lambda^{new} = \lambda + R^{-1}d \tag{87}$$

and

$$\Lambda^{new} = \Lambda - R^{-1} \tag{88}$$

The result is a maximum entropy density of the same form as (85) but with the new parameters λ^{new} and Λ^{new}, and new new normalization factor, as can be verified easily. The last step is to sample from this new pdf. This is not so easy, but feasible ways to do this have been presented.

Another variant is the Mean-Field Filter, in which only the mean of the ensemble is updated. The algorithm closely follows that of the Maximum Entropy Particle Filter and is not discussed here. Obviously, the update step is much cheaper, but less accurate.

Interesting is that when the background probability is a single Gaussian, and the mean and the covariance of the ensemble are taken as constraints in the prior maximum entropy pdf, the method is identical to the EnKF. In this sense it can be considered as a non-Gaussian extension to that method.

The methods were tested on the double-well and the Lorenz 63 problems. With a well-chosen background pdf q, both methods outperformed the EnKF and the SIR for ensemble sizes 10-100. The conclusion is that when knowledge of this background pdf is available use it, and the Maximum Entropy Particle Filter is one of the methods to do so.