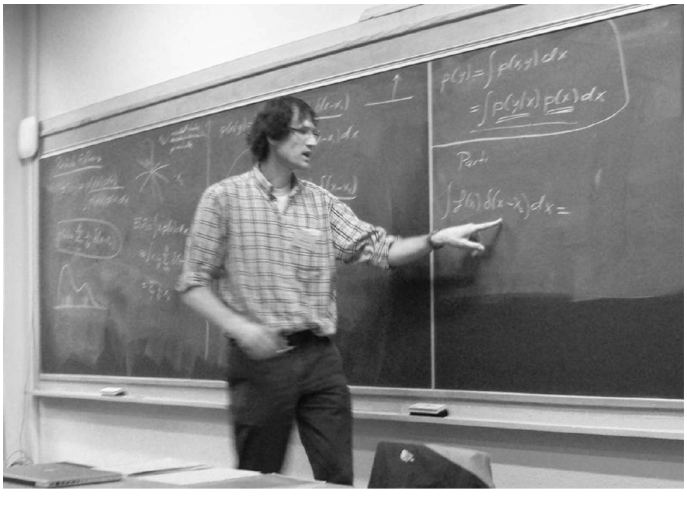
13
Particle filters for the geosciences

P. J. van Leeuwen

Data Assimilation Research Centre
Department of Meteorology
University of Reading
Earley Gate
P.O. Box 243
Reading RG6 6BB
United Kingdom



Advanced Data Assimilation for Geosciences. First Edition.
Edited by É. Blayo, M. Bocquet, E. Cosme, and L. F. Cugliandolo.

Chapter Contents

13.1 Introduction

In this chapter, we will discuss particle filters and their use in the geosciences. A general review on the application and usefulness of particle filters in geosciences in given in van Leeuwen (2009), and a general overview of particle filtering is given in the excellent book by Doucet et al. (2001). There, it was shown that although interesting progress had been made until 2009, no solution for the degeneracy problem or the curse of dimensionality had been found. However, a lot of progress has been made during the last couple of years.

First, the basic idea behind particle filters is presented, followed by an explanation of why this basic formulation can never work for large-dimensional systems. We discuss resampling as a way to increase the efficiency of particle filters. Then we discuss proposal densities, which form the major part of this chapter. We show that they give us an enormous amount of freedom to build particle filters for very high-dimensional systems, and present an example of a successful approach that works in systems of any dimension by construction. This is the first example of what is undoubtedly an enormous growth in useful methods for extremely high-dimensional systems encountered in the geosciences. To keep the text fluent, I have kept the literature references to a minimum; a more comprehensive, but slightly outdated, literature list for this rapidly emerging field can be found in van Leeuwen (2009).

We discuss the basic particle filter as an importance sampler, and show why straightforward implementation will lead to so-called degeneracy, in which the effective ensemble size reduces to a very small number of particles and the method fails. At the same time, the strength of particle filters will be investigated, namely that particle filters are completely nonlinear they have no problems with model balances after updates (at least not the simple versions), and their performance is not reliant on a good representation of the error covariance of the model state. The latter has been overlooked in the past, but is actually a major selling point.

As real-world examples show us again and again, the data assimilation problem is typically a nonlinear one, especially in the geosciences. The models we use for simulation are almost never linear, and the observation operator that relates model states to observations is quite often nonlinear too. While linearizations have been shown to be very useful to tackle real-world problems, there are several problems that are so nonlinear that these linearizations are just not enough.

As has been discussed in Chapter 3 of this volume, Kalman filters either assume that the update is a linear combination between observations and prior estimate, the best linear unbiased estimate (BLUE), or assume that both the prior and the likelihood are Gaussian-distributed in model state. Of course, when the system is weakly nonlinear, the Kalman filter can be used quite efficiently, and even iterations of the Kalman-filter update can be performed. But when the system if highly nonlinear, these iterations are unlikely to converge, and, if they do, it is unclear to what. Also, the interpretation of the ensemble as a measure for the posterior covariance becomes questionable. It is important to realize that the (ensemble) Kalman filter is not variance-minimizing for a non-Gaussian posterior pdf!

Variational methods such as 4D-Var and the representer method look for the maximum of the posterior probability density function (pdf), or to the minimum of minus the logarithm of this pdf, which amounts to the same state. When the system is linear or Gaussian, it is easy to prove that there is indeed one maximum. Also, for a weakly nonlinear system, variational methods are very useful, and the variational problem can be solved by iteration, sometimes called 'incremental 4D-Var'. However, when the problem is highly nonlinear, it can be expected that the posterior pdf has several local maxima, and the variational methods will converge to one of them. This is not necessarily the global maximum. Another issue is of course the lack of covariance information. Even if the inverse of the Hessian, the local curvature of the pdf at the maximum, can be calculated, it does not represent the covariance of the full posterior pdf.

Nonlinear data assimilation is a whole new ball game, especially when the posterior pdf is multimodal. What does the 'best estimate' mean? Is it the mean of the posterior pdf? Well, definitely not when the posterior is bimodal and the two modes have equal probability mass and are of equal shape. In that case, the mean will fall between the two peaks. Is the global maximum the best estimate? If the posterior pdf has multiple maxima of equal size, the answer is no. Also, when the maximum is related to a relatively small probability mass, it is also not that useful. It becomes clear that the notion of 'best estimate' depends very strongly on the application, and it is perhaps not a very useful concept in nonlinear data assimilation.

The solution to the data assimilation problem is not a best estimate, but the posterior pdf itself. That is exactly what Bayes' theorem tells us—given the prior pdf and the likelihood, we can calculate the posterior pdf, and that is the answer. And the calculation is extremely simple, just a multiplication. So, this little excursion into nonlinear data assimilation learns us that data assimilation is *not an inverse problem*, but a multiplication problem. That is the starting point for this chapter on particle filters.

13.2 A simple particle filter based on 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, as we will show below.

13.2.1 Basic importance sampling

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, such as mean and covariance. If one represents the prior pdf by a number of particles, or ensemble members, as in the ensemble Kalman Filter, then

$$p(x) = \sum_{i=1}^{N} \frac{1}{N} \delta(x - x_i), \tag{13.1}$$

and we use this in Bayes' theorem:

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\,\mathrm{d}x}. \tag{13.2}$$

We find

$$p(x|y) = \sum_{i=1}^{N} w_i \delta(x - x_i), \tag{13.3}$$

in which the weights w_i are given by

$$w_i = \frac{p(y|x_i)}{\sum_{j=1}^{N} p(y|x_j)}. \tag{13.4}$$

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:

$$p(y|x_i) = A \exp\left\{ -\frac{[y - H(x_i)]^2}{2\sigma^2} \right\}, \tag{13.5}$$

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

$$\overline{f(x)} = \int f(x)p(x)\,\mathrm{d}x \approx \sum_{i=1}^{N} w_i f(x_i). \tag{13.6}$$

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 have not specified what x is. It can be a state vector x^n at a certain time n, or x can be a model trajectory over some time window $(0, n\Delta t)$, so $x = x^{0:n} = (x^0, x^1, \ldots, x^n)$ over n time steps. Here the superscript is the time index, and the subscript is the sample, or particle.

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

$$p(x^{0:n}) = p(x^n|x^{0:n-1})p(x^{0:n-1}). \tag{13.7}$$

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

$$p(x^{0:n}) = p(x^n|x^{n-1})p(x^{n-1}|p(x^{n-2}) \ldots p(x^1|x^0)p(x^0). \tag{13.8}$$

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

$$x^n = f(x^{n-1}) + \beta^n, \tag{13.9}$$

in which β^n 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, (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 from its density and evaluate the model equation given above. In fact, for normally, or Gaussian, distributed model errors β^n with mean zero and covariance Q, we can write

$$p(x^n|x^{n-1}) = N(f(x^{n-1}), Q). \tag{13.10}$$

Note that we assume that the model errors are additive in this chapter. Multiplicative model errors in which the size of the random forcing is dependent on the state x can be accounted for too, but we use additive model errors here for simplicity.

Let us now continue with importance sampling. If we also assume that the observations at different times, conditional on the states at those 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

$$p(y^{1:n}|x^{0:n}) = p(y^n|x^n)\dots p(y^1|x^1), \tag{13.11}$$

where we have 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

$$\begin{aligned} p(x^{0:n}|y^{1:n}) &= \frac{p(y^{1:n}|x^{0:n})p(x^{0:n})}{p(y^{1:n})} \\ &= \frac{p(y^n|x^n)\dots p(y^1|x^1)p(x^n|x^{n-1})\dots p(x^1|x^0)p(x^0)}{p(y^n)\dots p(y^1)} \\ &= \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)} \dots \frac{p(y^1|x^1)p(x^1|x^0)p(x^0)}{p(y^1)}. \end{aligned} \tag{13.12}$$

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

$$p(x^{0:n}|y^{0:n}) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)} p(x^{0:n-1}|y^{1:n-1}). \tag{13.13}$$

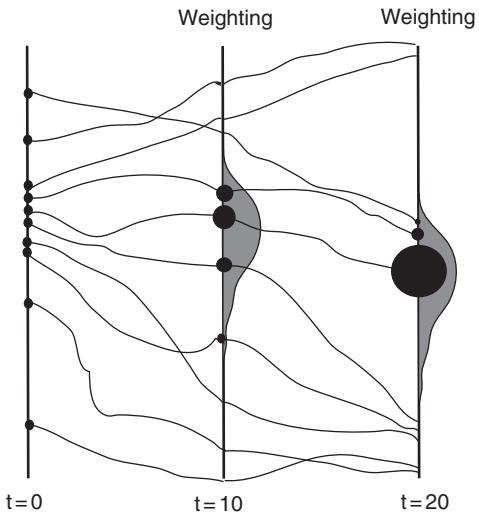


Fig. 13.1 The standard particle filter with importance sampling. The model variable runs along the vertical axis; the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time 0. At time 10, the likelihood is displayed together with the new weights of each particle. At time 20, only two members have weights different from zero: the filter has become degenerate. (From van Leeuwen (2010).)

This expression allows us to find the full posterior with the following sequential scheme (see Fig. 13.1):

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 (13.4), normalize them so their sum is equal to 1, 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 setps 2 and 3 until all observations have been processed.

13.2.2 Why particle filters are so attractive

Despite the problems just discussed, the advantages of particle fitter compared with traditional methods should not be underestimated. First of all, they do solve the

complete nonlinear data assimilation problem (see the discussion at the beginning of this chapter).

Furthermore, 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.

And finally it should be stressed how simple this scheme is compared with traditional methods such as 3D- or 4D-Var and (ensemble) Kalman filters. The success of these scheme depends heavily on the accuracy and error covariances of the model state vector. In 3D- and 4D-Var, this leads to complicated covariance structures to ensure balances etc. In ensemble Kalman filters, artificial tricks such as covariance inflation and localization are needed to get good results in high-dimensional systems. Particle filters do not have these difficulties.

However, there is (of course) a drawback. Even if the particles manage to follow the observations in time, 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 Fig. 13.1 at $t = 20$). That means that the statistical information in the ensemble becomes too low to be meaningful. This is called *filter degeneracy*. It gave importance sampling a low profile until resampling was invented, see Section 13.3.

13.3 Reducing the variance in the weights

Several methods exist to reduce the variance in the weights, and we discuss sequential importance resampling here. See van Leeuwen (2009) for other methods. In resampling methods, the posterior ensemble is resampled so that the weights become more equal (Gordon et al., 1993). In Section 13.4, 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.

13.3.1 Resampling

The idea of resampling is simply that particles with very low weights are abandoned, while multiple copies of particles with high weights 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 resampling does the above, and makes sure that the weights of all posterior particles are equal again, to $1/N$.

Sequential importance resampling is identical to basic importance sampling but for a resampling step after the calculation of the weights. The 'flow chart' reads as follows (see Fig. 13.2):

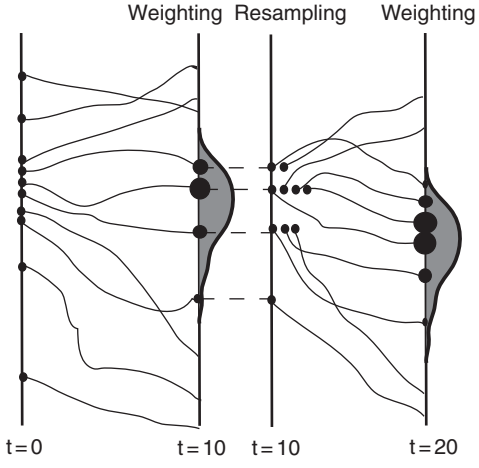


Fig. 13.2 The particle filter with resampling, also called sequential importance resampling. The model variable runs along the vertical axis; the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time 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. (From van Leeuwen (2010).)

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 (13.4) and attach these weights to each corresponding particle. Note that the particles are not modified—only their relative weight is changed!
4. Resample the particles such that the weights are equal to $1/N$.
5. Repeat steps 2, 3, and 4 sequentially until all observations have been processed.

It is good to realize 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 fewer and fewer particles over time, until it consists of only one particle, loosing again all statistical meaning.

The resampling can be performed in many ways, and we discuss the most commenly 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, because of the random character of the sampling, so-called sampling noise is introduced. Note that this method is actually generalized Bernoulli, for those versed in sampling techniques.
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 then 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 the unit interval $[0, 1]$. Then a random number is drawn from a uniform density on $[0, 1/N]$, and 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 endpoints 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.

13.3.2 Is resampling enough?

Snyder et al. (2008) prove that resampling will not be enough to avoid filter collapse. 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. The conclusion is that more than simple resampling is needed to solve the degeneracy problem.

13.4 The proposal density

In this section, we will concentrate on recent developments in using the so-called proposal transition density in solving the degeneracy problem. Related to decreasing the variance of the weights is to make sure that all model integrations end up close to the new observations, or, more precisely, ensuring that all posterior particles have similar weights.

First, we discuss what a proposal density is in particle filtering, and how it can be useful. This is then illustrated with using an ensemble Kalman filter as proposal density. This is followed by a discussion of more traditional methods such as the auxiliary particle filter, the backtracking particle filter, and guided sequential importance sampling.

Next, we discuss methods that change the model equations by bringing information on where the future observations are directly into the model equations. We start with the so-called optimal proposal density and show that that idea does not work in high-dimensional spaces with large numbers of independent observations. The optimal proposal density is a one-time-step scheme, assuming observations every time step. The so-called implicit particle filter extends this to multiple time steps between observations. It is shown that the implicit particle filter can be interpreted as a weak-constraint 4D-Var on each particle, with fixed initial condition. We will show that when the number of independent observations is large, this filter will also be problematic. This is followed by a method that will not be degenerate by construction—the equivalent-weights particle filter. Its working is illustrated on a 65 000-dimensional barotropic vorticity model of atmospheric or oceanic flow, hinting that particle filters

are now mature enough to explore in, for example, operational numerical weather prediction settings.

We are now to discuss a very interesting property of particle filters that has received little attention in the geophysical community. We start from Bayes' theorem:

$$p(x^{0:n}|y^{0:n}) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)} p(x^{0:n-1}|y^{1:n-1}). \tag{13.14}$$

To simplify the analysis, and since we concentrate on a filter here, let us first integrate out the past, to get

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int p(x^n|x^{n-1})p(x^{n-1}|y^{1:n-1}) \, \mathrm{d}x^{n-1}. \tag{13.15}$$

This expression does not change when we multiply and divide by a so-called proposal transition density $q(x^n|x^{n-1}, y^n)$, so

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int \frac{p(x^n|x^{n-1})}{q(x^n|x^{n-1}, y^n)} q(x^n|x^{n-1}, y^n) p(x^{n-1}|y^{1:n-1}) \, \mathrm{d}x^{n-1}. \tag{13.16}$$

As long as the support of $q(x^n|x^{n-1}, y^n)$ is equal to or larger than that of $p(x^n|x^{n-1})$ we can always do this. This last condition makes sure we do not divide by zero. Let us now assume that we have an equal-weight ensemble of particles from the previous analysis at time $n-1$, so

$$p(x^{n-1}|y^{1:n-1}) = \sum_{i=1}^{N} \frac{1}{N} \delta(x^{n-1} - x_i^{n-1}). \tag{13.17}$$

Using this in (13.16) gives

$$p(x^n|y^{0:n}) = \sum_{i=1}^{N} \frac{1}{N} \frac{p(y^n|x^n)}{p(y^n)} \frac{p(x^n|x_i^{n-1})}{q(x^n|x_i^{n-1}, y^n)} q(x^n|x_i^{n-1}, y^n). \tag{13.18}$$

As a last step, we run the particles from time $n-1$ to n, i.e. we sample from the transition density. However, instead of drawing from $p(x^n|x_i^{n-1})$, so running the original model, we sample from $q(x^n|x_i^{n-1}, y^n)$, so from a modified model. Let us write this modified model as

$$x^n = g(x^{n-1}, y^n) + \hat{\beta}^n, \tag{13.19}$$

so that we can write for the transition density, assuming $\hat{\beta}^n$ is Gaussian-distributed with covariance $\hat{Q}$:

$$q(x^n|x^{n-1}, y^n) = N(g(x^{n-1}, y^n), \hat{Q}). \tag{13.20}$$

Drawing from this density leads to

$$p(x^n|y^{0:n}) = \sum_{i=1}^{N} \frac{1}{N} \frac{p(y^n|x_i^n)}{p(y^n)} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^n)} \delta(x^n - x_i^n), \tag{13.21}$$

so the posterior pdf at time n can be written as

$$p(x^n|y^{1:n}) = \sum_{i=1}^{N} w_i \delta(x^n - x_i^n), \tag{13.22}$$

with weights w_i given by

$$w_i = \frac{1}{N} \frac{p(y^n|x_i^n)}{p(y^n)} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^n)}. \tag{13.23}$$

We recognize the first factor in this expression as the likelihood, and the second as as a factor related to using the proposal transition density instead of the original transition density to propagate from time $n-1$ to n, so it is related to the use of the proposed model instead of the original model. Note that because the factor $1/N$ and $p(y^n)$ are the same for each particle and we are only interested in relative weights, we will drop them from now on, so

$$w_i = p(y^n|x_i^n) \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^n)}. \tag{13.24}$$

Finally, let us formulate an expression for the weights when multiple model time steps are present between observation times. Assume the model needs m time steps between observations. This means that the ensemble at time $n-m$ is an equal-weight ensemble, so

$$p(x^{n-m}|y^{1:n-m}) = \sum_{i=1}^{N} \frac{1}{N} \delta(x^{n-m} - x_i^{n-m}). \tag{13.25}$$

We will explore the possibility of a proposal density at each model time step, so for the original model we write

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int \prod_{j=n-m+1}^{n} p(x^j|x^{j-1}) p(x^{n-m}|y^{1:n-m}) \, \mathrm{d}x^{n-m:n-1}, \tag{13.26}$$

and, introducing a proposal transition density at each time step, we find:

$$\begin{aligned} &p(x^n|y^{0:n}) \\ &\quad = \frac{p(y^n|x^n)}{p(y^n)} \int \prod_{j=n-m+1}^{n} \frac{p(x^j|x^{j-1})}{q(x^j|x^{j-1}, y^n)} q(x^j|x^{j-1}, y^n) p(x^{n-m}|y^{1:n-m}) \, \mathrm{d}x^{n-m:n-1}. \end{aligned} \tag{13.27}$$

Using the expression for $p(x^{n-m}|y^{1:n-m})$ from (13.25) and choosing randomly from the transition proposal density $q(x^j|x^{j-1}, y^n)$ at each time step leads to

$$w_i = p(y^n|x_i^n) \prod_{j=n-m+1}^{n} \frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^n)}. \tag{13.28}$$

13.4.1 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 endpoint, the posterior EnKF sample x_i^n, and we know the model equation, written formally as:

$$x_i^n = f(x_i^{n-1}) + \beta_i^n, \tag{13.29}$$

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

$$p(x_i^n|x_i^{n-1}) \propto \exp\left\{-\left(\frac{1}{2}\right)\left[x_i^n - f\left(x_i^{n-1}\right)\right] Q^{-1}\left[x_i^n - f\left(x_i^{n-1}\right)\right]\right\}. \tag{13.30}$$

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

$$x_i^n = x_i^{n,old} + K^e\Big[y + \epsilon_i - H\Big(x_i^{n,\mathrm{old}}\Big)\Big], \tag{13.31}$$

in which ϵ_i is the random error drawn from $N(0, R)$ that has to be added to the observations in this variant of the ensemble Kalman filter. K^e is the ensemble Kalman 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:

$$x_i^{n,old} = f(x^{n-1}) + \beta_i^n. \tag{13.32}$$

Combining these two gives

$$x_i^n = f\left(x_i^{n-1}\right) + \beta_i^n + K^e\left[y + \epsilon_i - H\left(f\left(x_i^{n-1}\right)\right) - H(\beta_i^n))\right], \tag{13.33}$$

or

$$x_i^n = f(x_i^{n-1}) + K^e\left[y - H\left(f(x_i^{n-1})\right)\right] + (1 - K^e H)\,\beta_i^n + K^e\epsilon_i, \tag{13.34}$$

assuming that H is a linear operator. The right-hand side of this equation has a deterministic part 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

$$q(x_i^n|x_i^{n-1}y^n) \propto \exp\left[-\frac{1}{2}\left(x_i^n - \mu_i^n\right)^{\mathrm{T}} \Sigma_i^{-1}\left(x_i^n - \mu_i^n\right)\right], \tag{13.35}$$

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

$$\mu_i^n = f(x_i^{n-1}) + K^e\left[y - H(x_i^{n-1})\right] \tag{13.36}$$

and the covariance Σ_i is given by

$$\Sigma_i = (1 - K^e H)Q(1 - K^e H)^{\mathrm{T}} + K^e R K^{e\mathrm{T}}, \tag{13.37}$$

where we have assumed that the model and observation errors are uncorrelated. It should be realized that x_i^n does depend on all $x_j^{n,\mathrm{old}}$ via the Kalman gain, which 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 the 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, which will depend on the specific realization for β_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 calculations 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 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 (see Fig. 13.3):

1. Run the ensemble up to the observation time.
2. Perform a (local) EnKF analysis of the particles.
3. Calculate the proposal weights $w_i^* = p(x_i^n|x_i^{n-1})/q(x_i^n|x_i^{n-1}y^n)$.
4. Calculate the likelihood 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 smaller 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.

Finally, other variants of the EnKF, such as the adjusted and the transform variants can be used too, as detailed in van Leeuwen (2009). The efficiency of using the EnKF as proposal is under debate at the moment. The conclusions so far seem to be that using the EnKF as proposal in high-dimensional systems does not work. What has

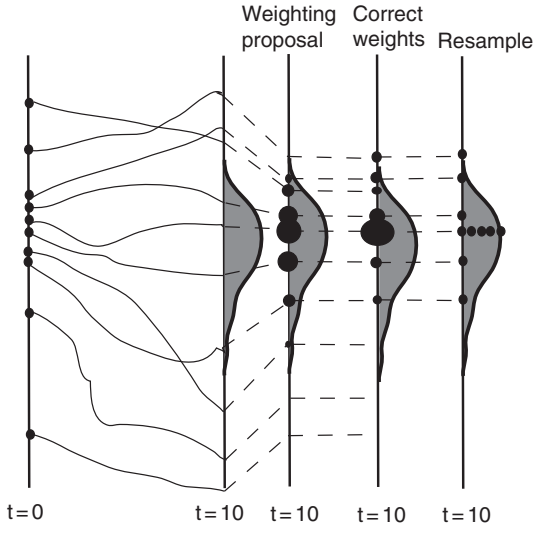


Fig. 13.3 The particle filter with proposal density. The model variable runs along the vertical axis; the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. At time 10, the particles are brought closer to the observations by using, for example, the EnKF. Then they are weighted with the likelihood and these weights are corrected for the artificial EnKF step. (From van Leeuwen (2010).)

not been tested, however, is to use EnKF proposals with smaller observation matrix R, and more possibilities are still open, like using localisation (see later on in this chapter).

13.4.2 The auxiliary particle filter

In the auxiliary particle filter, the ensemble at time $n-1$ is weighted with information of the likelihood at time n (see Pitt and Shephard, 1999). In this method, one generates a representation of each particle at the time of the new observation, for example 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 Fig. 13.4) as follows:

1. Integrate each particle from $n-1$ to n with simplified dynamics (e.g. without model noise), producing a representation of the proposal density $q(x^n|x_i^{n-1}, y^n)$.
2. Weight each particle with the new observations as

$$\beta_i \propto p(y^n|x_i^n)w_i^{n-1}. \tag{13.38}$$

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

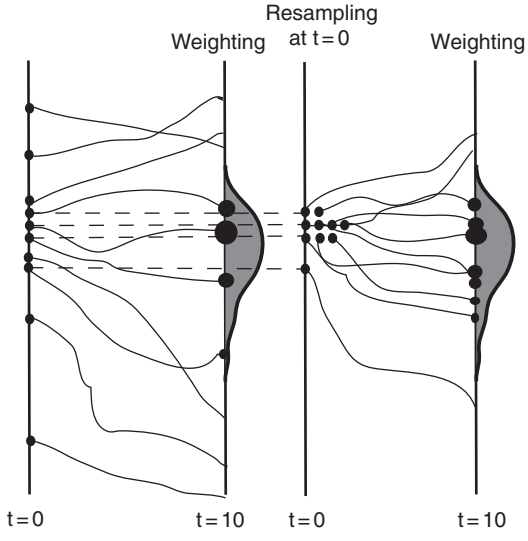


Fig. 13.4 The auxiliary particle filter. The model variable runs along the vertical axis; the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. 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. (From van Leeuwen (2010).)

3. Resample the particles i at time $n-1$ with these weights, and use this resampled particles j_i as a representation of the proposal density by integrating each forward to n with the full stochastic model, so choosing from $q(x^n|x_{j_i}^{n-1}, y^n)$. Note that j_i connects the original particle i with its new position in state space, that of particle j.
4. Re-weight the members with weights

$$w_i^n = \frac{1}{A} p(y^n|x_i^n) \frac{p(x_i^n|x_{j_i}^{n-1})}{q(x_i^n|x_{j_i}^{n-1}, y^n)\beta_{j_i}}, \tag{13.39}$$

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.

The name 'auxiliary' comes from the introduction of the member index j_i in the formulation. This member index keeps track of the relation between the first-stage weights and the particle sample at $n-1$.

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. However, one could also use a simplified model for the first set of integrations. A geophysical example would be

to use a quasi-geostrophic model for the first set of integrations and the full model for the second. One can imagine doing it even more times, zooming in into the likelihood, but at a cost of performing more and more integrations of the model. Figure 13.4 displays how the method works.

13.4.3 Including future observations in the model equations

So far, we have discussed proposal density applications in which the model equations were not changed directly. Of course, in, for example, the auxiliary particle filter, one could use a different model for the first set of integrations to obtain the first-stage weights, but the future observations were not used directly in the model equations. However, much more efficient schemes can be derived that change the model equations such that each particle is pulled towards the future observations at each time step. By keeping track of the weights associated with this, it can be assured that the correct problem is solved and the particles are random samples from the posterior pdf.

As mentioned before, the idea of the proposal transition density is that we draw samples from that density instead of from the original model. Furthermore, these samples can be dependent on the future observations. To see how this works, let us write the stochastic model equation as

$$x_i^n = f(x_i^{n-1}) + \beta_i^n. \tag{13.40}$$

First, we have to understand how this equation is related to the transition density $p(x_i^n|x_i^{n-1})$. The probability to end up in x_i^n starting from x_i^{n-1} is related to β_i^n. For instance, if $\beta_i^n = 0$, so there is no model error and, a perfect model, then this probability is 1 if the x_i^n, x_i^{n-1} pair fulfils the perfect model equations and zero otherwise. So, in this case, $p(x_i^n|x_i^{n-1})$ would be a delta function centred on $f(x_i^{n-1})$. However, the more realistic case is that the model error is non-zero. The transition density will now depend on the distribution of the stochastic random forcing. Assuming Gaussian random forcing with zero mean and covariance Q, so $\beta_i^n \sim N(0, Q)$, we find

$$p(x_i^n|x_i^{n-1}) \propto N(f(x_i^{n-1}), Q). \tag{13.41}$$

As mentioned above, we will not use the normal model equation for each particle, but a modified model equation, one that 'knows' about future observations and actually draws the model to those observations. Perhaps the simplest example is to add a term that relaxes the model to the future observation, such as

$$x_i^n = f(x_i^{n-1}) + \beta_i^n + K^n\big[y^{n+m} - H(x_i^{n-1})\big], \tag{13.42}$$

in which $n+m$ is the next observation time. Note that the observation operator H does not contain any model integrations, it is just the evaluation of x_i^{n-1} in observation space. The reason is simple, we do not have x_i^{n+m} yet. Clearly, each particle i will now be pulled towards the future observations, with relaxation strength related to the matrix K^n. In principle, we are free to choose K^n, but it is reasonable to assume that it is related to the error covariance of the future observation R and that of the model equations Q. We will show possible forms in the examples discussed later.

With the simple relaxation, or other techniques, we have ensured that all particles end up closer to the observations. But we cannot just alter the model equations, we have to compensate for this trick. This is why the proposal density turns up in the weights. Each time step, the weight of each particle changes with

$$w_i^n = \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^n)} \tag{13.43}$$

between observation times. This can be calculated in the following way. Using the modified model equations, we know x_i^{n-1} for each particle, that was our starting point, and also x_i^n. So, assuming the model errors are Gaussian-distributed, this would become

$$p(x_i^n|x_i^{n-1}) \propto \exp\left\{-\frac{1}{2}\left[x_i^n - f(x_i^{n-1})\right]^{\mathrm{T}} Q^{-1}\left[x_i^n - f(x_i^{n-1})\right]\right\}. \tag{13.44}$$

The proportionality constant is not of interest since it is the same for each particle and drops out when the relative weights of the particles are calculated. Note that we have all the ingredients to calculate this and that $p(x_i^n|x_i^{n-1})$ is just a number.

For the proposal transition density we use the same argument, to find:

$$\begin{aligned} q(x_i^n|x_i^{n-1}y^n) \propto \exp\Bigg\{&-\frac{1}{2}\left(x_i^n - f(x_i^{n-1}) - K^n\left[y^n - H\left(x^{n-1}\right)\right]^{\mathrm{T}}\right) \\ &\times Q^{-1}\Big(x_i^n - f(x_i^{n-1}) - K^n\left[y^n - H(x^{n-1})\right]\Big)\Bigg\} \\ = \exp&\left(-\frac{1}{2}\beta_i^{n\,\mathrm{T}} Q^{-1}\beta_i^n\right). \end{aligned} \tag{13.45}$$

Again, since we did choose β to propagate the model state forward in time, we can calculate this, and it is just a number. In this way, any modified equation can be used, and we know, at least in principle, how to calculate the appropriate weights.

13.4.4 The optimal proposal density

The so-called optimal proposal density is described in the literature, (see e.g. Doucet et al. 2001). It is argued that taking $q(x^n|x^{n-1}, y^n) = p(x^n|x^{n-1}, y^n)$ results in optimal weights. However, it is easy to show that this is not the case. Assume observations every time step and a resampling scheme at every time step, so that a equal-weighted ensemble of particles is present at time $n-1$. Furthermore, assume that model errors are Gaussian-distributed according to $N(0, Q)$ and observation errors are Gaussian-distributed according to $N(0, R)$. First, using the definition of conditional densities, we can write:

$$p(x^n|x^{n-1}, y^n) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n|x^{n-1})}, \tag{13.46}$$

where we have used $p(y^n|x^n, x^{n-1}) = p(y^n|x^n)$. Using this proposal density gives posterior weights

$$\begin{aligned} w_i &= p(y^n|x_i^n)\frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^n)} \\ &= p(y^n|x_i^n)\frac{p(x_i^n|x_i^{n-1})}{p(x_i^n|x_i^{n-1}, y^n)} \\ &= p(y^n|x_i^{n-1}). \end{aligned} \tag{13.47}$$

The latter can be expanded as

$$w_i = \int p(y^n, x^n|x^{n-1})\,\mathrm{d}x^n = \int p(y^n|x^n)p(x^n|x^{n-1})\,\mathrm{d}x^n, \tag{13.48}$$

in which we have again used $p(y^n|x^n, x^{n-1}) = p(y^n|x^n)$. Using the Gaussian assumptions mentioned above (note that the state is never assumed to be Gaussian), we can perform the integration to obtain

$$w_i \propto \exp\left\{-\frac{1}{2}\left[y^n - Hf\big(x_i^{n-1}\big)\right]^{\mathrm{T}}\left(HQH^{\mathrm{T}} + R\right)^{-1}\left[y^n - Hf\big(x_i^{n-1}\big)\right]\right\}. \tag{13.49}$$

Note that we have just calculated the probability density of $p(y^n|x_i^{n-1})$.

To estimate the order of magnitude of the first two moments of the distribution of $y^n - Hf(x_i^{n-1})$, it is expanded to $y^n - Hx_t^n + H\big[x_t^n - f(x_i^{n-1})\big]$ in which x_t^n is the true state at time n. If we now use $x_t^n = f(x_t^{n-1}) + \beta_t^n$, this can be expanded further as $y^n - Hx_t^n + H\big[f(x_t^{n-1}) - f(x_i^{n-1})\big] + H\beta_t^n$. To proceed, we make the following restrictive assumptions that will nevertheless allow us to obtain useful order-of-magnitude estimates. Let us assume that both the observation errors R and the observed model errors HQH^{T} are uncorrelated, with variances V_y and V_β, respectively, to find:

$$-\log(w_i) = \frac{1}{2(V_\beta + V_y)}\sum_{j=1}^{M}\left\{y_j^n - H_j x_t^n + H_j\beta_t^n + H_j\left[f\big(x_t^{n-1}\big) - f\big(x_i^{n-1}\big)\right]\right\}^2. \tag{13.50}$$

The variance of w_i arises from varying the ensemble index i. Clearly, the first three terms are given, and we introduce the constant $\gamma_j = y_j^n - H_j x_t^n + H_j\beta_t^n$. To proceed with our order-of-magnitude estimate, we assume that the model can be linearized as $F(x_i^{n-1}) \approx Ax_i^{n-1}$, leading to

$$-\log\, w_i = \frac{1}{2(V_\beta + V_y)}\sum_{j=1}^{M}\left[\gamma_j + H_j A\left(x_t^{n-1} - x_i^{n-1}\right)\right]^2. \tag{13.51}$$

The next step in our order-of-magnitude estimate is to assume $x_t^{n-1} - x_i^{n-1}$ to be Gaussian-distributed. In that case, the expression above is non-centrally χ_M^2

distributed apart from a constant. This constant comes from the variance of $\gamma_j + H_j A(x_t^{n-1} - x_i^{n-1})$, which is equal to $H_j A P^{n-1} A^{\mathrm{T}} H_j^{\mathrm{T}} = V_x$, in which P^{n-1} is the covariance of the model state at time $n-1$. Hence, we find

$$-\log w_i = \frac{V_x}{2(V_\beta + V_y)} \sum_{j=1}^{M} \frac{\left[\gamma_j + H_j A(x_t^{n-1} - x_i^{n-1})\right]^2}{V_x}. \tag{13.52}$$

Apart from the constant in front, this expression is non-centrally χ^2_M distributed with variance $a^2 2(M + 2\lambda)$, where $a = V_x/(2(V_\beta + V_y)$ and $\lambda = (\sum_j \gamma_j^2)/V_x$.

We can estimate λ by realizing that for a large enough number of observations we expect $\sum_j (y_j^n - H_j x_t^n)^2 \approx M V_y$, and $\sum_j (y_j^n - H_j x_t^n) \approx 0$. Furthermore, when the dimension of the system under study is large, we expect $\sum_j (H_j \beta_t^n)^2 \approx M V_\beta$. Combining all these estimates, we find that the variance of $-\log w_i$ can be estimated as

$$\frac{M}{2}\left(\frac{V_x}{V_\beta + V_y}\right)^2 \left[1 + 2\left(\frac{V_\beta + V_y}{V_x}\right)\right]. \tag{13.53}$$

This expression shows that the only way to keep the variance of $-\log w_i$ low when the number of independent observations M is large is to have a very small variance in the ensemble: $V_x \approx (V_\beta + V_y)/M$. Clearly, when the number of observations is large (10 million in typical meteorological applications), this is not very realistic. This expression has been tested in several applications and holds within a factor 0.1 in all tests (van Leeuwen, 2012, unpublished manuscript).

It should be mentioned that a large variance of $-\log w_i$ does not necessarily mean that the weights will be degenerate, because the large variance could be due to a few outliers. However, we have shown that $-\log w_i$ is approximately non-centrally χ^2_M distributed for a linear model, so the large variance is not due to outliers but intrinsic in the sampling from such a distribution. Furthermore, there is no reason to assume that this variance will behave better for nonlinear models, especially because we did not make any assumptions on the divergent or contracting characteristics of the linear model.

From this analysis, we learn two things: it is the number of independent observations that determines the degeneracy of the filter, and the optimal proposal density cannot be used in systems with a very large number of independent accurate observations.

13.4.5 The implicit particle filter

In 2009, Chorin and Tu (2009) introduced this implicit particle filter. Although their paper is not very clear, with the theory and the application intertwined, discussions with them and later papers (Chorin et al., 2010; Morzfeld and Chorin, 2012) explain the method in more detail. Although they do not formulate their method in terms of a proposal density, to clarify the relation with the other particle filters this is the way it is presented here. In fact, as we shall see, it is closely related to the optimal proposal density discussed before when the observations are available at every time step.

The proposed samples are produced as follows. Assume we have m model time steps between observations. Draw a random vector ξ_i of length the size of the state vector times m. Each element of ξ_i is drawn from $N(0,1)$. The actual samples are now constructed by solving

$$-\log\left[p(y^n|x^n)p\left(x^{n-m+1:n}|x_i^{n-m}\right)\right] = \frac{\xi_i^{\mathrm{T}}\xi_i}{2} + \phi_i \tag{13.54}$$

for each particle x_i. The term ϕ_i is included to ensure that the equation has a solution, so $\phi_i \geq \min\left\{-\log\left[p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m})\right]\right\}$. Note that this can be written as

$$p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m}) = A\exp\left(-\frac{\xi^{\mathrm{T}}\xi}{2} - \phi_i\right) \tag{13.55}$$

for later reference. One can view this step as drawing from the proposal density $q_x(x^{n-m+1:n}|x_i^{n-m}, y^n)$ via the proposal density $q_\xi(\xi)$, where we have introduced the subscript to clarify the shape of the pdf. These two are related by a transformation of the probability densities as

$$q_x(x^{n-m+1:n}|x_i^{n-m}, y^n)\, dx^{n-m:n} = q_\xi(\xi)\, d\xi, \tag{13.56}$$

so that

$$q_x(x^{n-m:n}|x_i^{n-m}, y^n) = q_\xi(\xi)J, \tag{13.57}$$

in which J is the Jacobian of the transformation $\xi \to x$. We can now write the weights of this scheme as

$$\begin{aligned} w_i &= p(y^n|x_i^n)\frac{p(x_i^{n-m+1:n}|x_i^{n-m})}{q_x(x_i^{n-m+1:n}|x_i^{n-m}, y^n)} \\ &= p(y^n|x_i^n)\frac{p(x^{n-m+1:n}|x_i^{n-m})}{Jq_\xi(\xi_i)}. \end{aligned} \tag{13.58}$$

Using (13.55), we find that the weights are given by

$$w_i = A\frac{\exp(-\phi)}{J}. \tag{13.59}$$

To understand better how this works, let us consider the case of observations every model time step, and Gaussian observation errors, Gaussian model equation errors, and linear observation operator H. In that case, we have

$$\begin{aligned} -\log\left[p(y^n|x^n)p(x^{n-m+1}|x_i^{n-m})\right] &= \frac{1}{2}\left(y^n - Hx_i^n\right)^{\mathrm{T}} R^{-1}\left(y^n - Hx_i^n\right) \\ &\quad + \frac{1}{2}\left[x^n - f(x_i^{n-1})\right]^{\mathrm{T}} Q^{-1}\left[y^n - f(x_i^{n-1})\right] \\ &= \frac{1}{2}\left(x^n - \hat{x}_i^n)\right)^{\mathrm{T}} P^{-1}\left(x^n - \hat{x}_i^n\right) + \phi_i, \end{aligned} \tag{13.60}$$

in which $\hat{x}_i^n = f(x_i^{n-1}) + K[y^n - Hf(x_i^{n-1})]$, the maximum of the posterior pdf, and $P = (1 - KH)Q$, with $K = QH^{\mathrm{T}}(HQH^{\mathrm{T}} + R)^{-1}$. Comparing this with (13.55), we find $x^n = P^{1/2}\xi$, so J is a constant, and

$$\begin{aligned}\phi_i &= \min\left\{-\log\left[p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m})\right]\right\} \\ &= \frac{1}{2}\left[y^n - Hf(x_i^{n-1})\right]^{\mathrm{T}}(HQH^{\mathrm{T}} + R)^{-1}\left[y^n - Hf(x_i^{n-1})\right],\end{aligned} \tag{13.61}$$

and finally $w_i \propto \exp(-\phi_i)$. Comparing with the optimal proposal density, we see that when observations are present at every time step, the implicit particle filter is equal to the optimal proposal density, with the same degeneracy problem.

13.4.6 The equivalent-weights particle filter

At the beginning of this chapter, we discussed a very simple nudging scheme to pull the particles towards future observations, written as

$$x_i^n = f(x_i^{n-1}) + \beta_i^n + K^n\left[(y^{n+m} - H(x_i^{n-1})\right], \tag{13.62}$$

in which $n + m$ is the next observation time. Unfortunately, exploring the proposal density by simply nudging will not avoid degeneracy in high-dimensional systems with a large number of observations. Also, more complicated schemes, such as running a 4D-Var on each particle, which is essentially what Chorin and Tu (2009) propose, are likely to lead to strongly varying weights for the particles because its close relation to the optimal proposal density. (However, it must be said that no rigourous proof exists of this statement!) We can expect to have to do something more optimal.

To start, let us recall the expression we found for the proposal density weights in Section 13.4, equation (13.28):

$$w_i = p(y^n|x_i^n)\prod_{j=n-m+1}^{n}\frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^n)}. \tag{13.63}$$

We will use a modified model as explained in Section 13.4.3 for all but the last model time step, which will be different, as explained below. The last time step consists of two stages: first perform a deterministic time step with each particle that ensures that most of the particles have equal weight, and then add a very small random step to ensure that Bayes' theorem is satisfied (for details, see van Leeuwen, 2010, 2011). There are again infinitely many ways to do this. For the first stage we write down the weight for each particle using only a deterministic move, so ignoring the proposal density q for the moment:

$$\begin{aligned}-\log w_i = &-\log w_i^{\mathrm{rest}} + \frac{1}{2}(y^n - Hx_i^n)^{\mathrm{T}}R^{-1}(y^n - Hx_i^n) \\ &+ \frac{1}{2}\left[x_i^n - f(x_i^{n-1})\right]^{\mathrm{T}}Q^{-1}\left[x_i^n - f(x_i^{n-1})\right],\end{aligned} \tag{13.64}$$

in which w_i^{rest} is the weight accumulated over the previous time steps between observations, so the p/q factors from each time step. If H is linear, which is not essential

but we will assume for simplicity here, this is a quadratic equation in the unknown x_i^n. All other quantities are given. We calculate the minimum of this function for each particle i, which is simply given by

$$-\log w_i = -\log w_i^{\text{rest}} + \frac{1}{2}\left[y^n - Hf(x_i^{n-1})\right]^{\mathrm{T}} (HQH^{\mathrm{T}} + R)^{-1}\left[y^n - Hf(x_i^{n-1})\right]. \tag{13.65}$$

For N particles, this gives rise to N minima. Next, we determine a target weight as the weight that 80% of the particles can reach; i.e. 80% of the minimum $-\log w_i$ is smaller than the target value. (Note that we can choose another percentage; see e.g. Ades and van Leeuwen (2013), who investigate the sensitivity of the filter for values between 70% and 100%.) Define a quantity $C = -\log w_{\text{target}}$, and solve for each particle with a minimum weight larger than the target weight:

$$C = -\log w_i^{\text{rest}} + \frac{1}{2}\left[y^n - Hf(x_i^{n-1})\right]^{\mathrm{T}} (HQH^{\mathrm{T}} + R)^{-1}\left[y^n - Hf(x_i^{n-1})\right]. \tag{13.66}$$

So now we have found the positions of the new particles x_i^n such that all have equal weight. The particles that have a minimum larger than C will come back into the ensemble via a resampling step, to be discussed later.

The equation above has an infinite number of solutions for dimensions larger than 1. To make a choice we assume

$$x_i^n = f(x_i^{n-1}) + \alpha_i K[y^n - Hf(x_i^{n-1})], \tag{13.67}$$

in which $K = QH^{\mathrm{T}}(HQH^{\mathrm{T}} + R)^{-1}$, Q is the error covariance of the model errors, and R is the error covariance of the observations. Clearly, if $\alpha_i = 1$, we find the minimum back. We choose the scalar α_i such that the weights are equal, leading to

$$\alpha_i = 1 - \sqrt{1 - b_i/a_i}, \tag{13.68}$$

in which $a_i = 0.5x_i^{\mathrm{T}} R^{-1} HKx$ and $b_i = 0.5x_i^{\mathrm{T}} R^{-1} x_i - C - \log w_i^{\text{rest}}$. Here $x = y^n - Hf(x_i^{n-1})$, C is the chosen target weight level, and w_i^{rest} denotes the relative weights of each particle i up to this time step, related to the proposal density explained above.

Of course, this last step towards the observations cannot be fully deterministic. A deterministic proposal would mean that the proposal transition density q can be zero while the target transition density p is non-zero, leading to division by zero, because for a deterministic move the transition density is a delta function. The proposal transition density could be chosen as a Gaussian, but since the weights have q in the denominator, a draw from the tail of a Gaussian would lead to a very high weight for a particle that is perturbed by a relatively large amount. To avoid this, q is chosen in the last step before the observations as a mixture density

$$q(x_i^n | x_i') = (1 - \gamma)U(-a, a) + \gamma N(0, a^2), \tag{13.69}$$

in which x_i', the particle before the last random step, and γ and a are small. By choosing γ small, the change of having to choose from $N(0, a^2)$ can be made as small as desired. For instance, it can be made dependent on the number of particles N.

To conclude, the almost-equal-weight scheme consists of the following steps:

1. Use the modified model equations for each particle for all time steps between observations.
2. Calculate, for each particle i for each of these time steps,

$$w_i^n = w_i^{n-1} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1}, y^m)}. \tag{13.70}$$

3. At the last time step before the observations, calculate the maximum weight for each particle and determine $C = -\log w_{\text{target}}$.
4. Determine the deterministic moves by solving for α_i for each particle as outlined above.
5. Choose a random move for each particle from the proposal density (13.69).
6. Add these random move to each deterministic move, and calculate the full posterior weight.
7. Resample, and include the particles that have been neglected from step 4 on.

Finally, it is stressed again that we do solve the fully nonlinear data assimilation problem with this efficient particle filter, and the only approximation is in the ensemble size. All other steps are completely compensated for in Bayes' theorem via the proposal density freedom.

13.4.7 Application to the barotropic vorticity equations

A few results using the new particle filter with almost equal weights are shown here, (see van Leeuwen and Ades, 2013). Figure 13.5 shows the application of the method

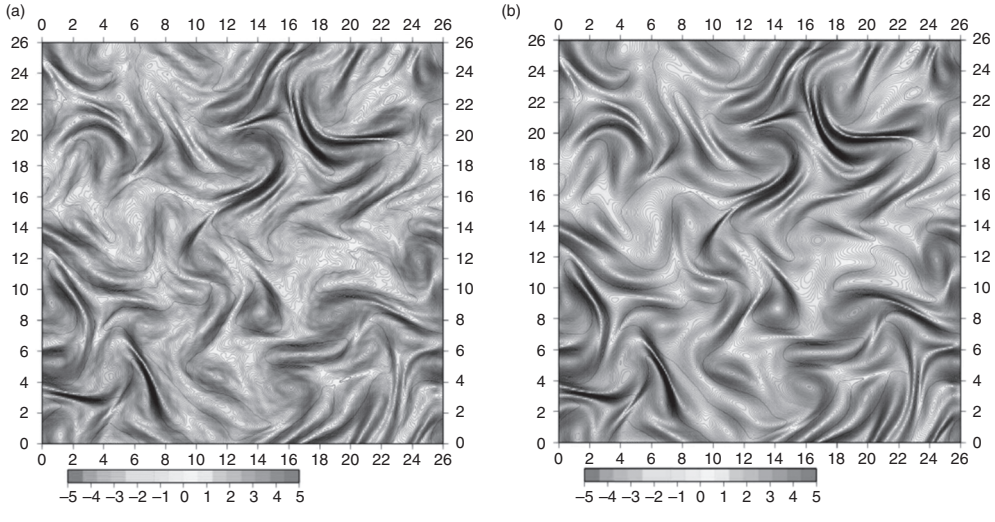


Fig. 13.5 Snapshots of (a) the particle filter mean and (b) the vorticity field of the truth at time 25. Note the highly chaotic state of the fields and the close to perfect tracking.

to the highly chaotic barotropic vorticity equation, governed by:

$$\frac{\partial q}{\partial t} - \frac{\partial \psi}{\partial y}\frac{\partial q}{\partial x} + \frac{\partial \psi}{\partial x}\frac{\partial q}{\partial y} = \beta,$$
$$q = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}, \tag{13.71}$$

in which q is the vorticity field, ψ is the stream function, and β is a random noise term representing errors in the model equations. It was chosen from a multivariate Gaussian with mean zero, variance 0.01, and decorrelation lengthscale 4 gridpoints. The equations are implemented on a 256×256 grid, using a semi-Lagrangian scheme with time step $\Delta t = 0.04$ and grid spacing $\Delta x = \Delta y = 1/256$, leading to a state dimension close to 65,000. The vorticity field was observed every 50 time steps on every gridpoint. The decorrelation timescale of this system is about 25 time steps, so, even though the full state is observed, this is a very hard highly nonlinear data assimilation problem. The observations were obtained from a truth run, and independent random measurement noise with standard deviation 0.05 was added to each observation.

Only 24(!) particles were used to track the posterior pdf. In the application of the new particle filter, we chose $K = 0.1$ in the nudging term (except for the last time step before the new observations, where the 'almost equal weight' scheme was used, as explained above), multiplied by a linear function that is zero halfway between the two updates and growing to one at the new observation time. The random forcing was the same as in the original model. This allows the ensemble to spread out owing to

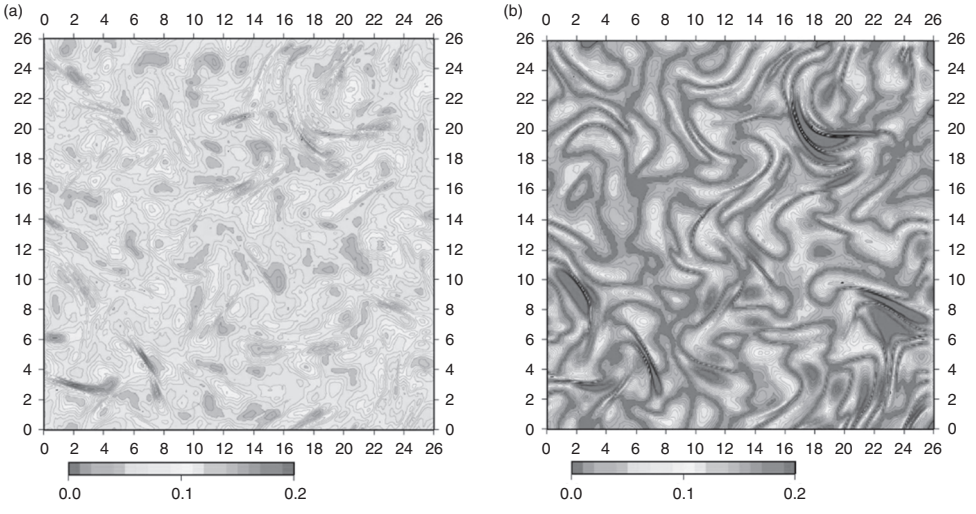


Fig. 13.6 Snapshots of the absolute value of (a) the mean truth misfit and (b) the standard deviation in the ensemble. The ensemble underestimates the spread at several locations, but, averaged over the field, it is slightly higher: 0.074 versus 0.056.

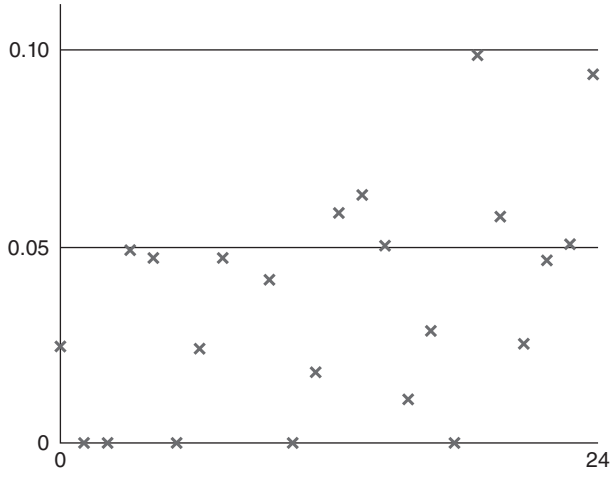


Fig. 13.7 Weights distribution of the particles before resampling. All weights cluster around 0.05, which is close to 1/24 for uniform weights (using 24 particles). The 5 particles with zero weights will be resampled. Note that the other particles form a smoother estimate.

the random forcing, and pulling harder and harder towards the new observation the closer it was to the new update time.

Figure 13.6 shows the difference between the mean and the truth after 50 time steps, together with the ensemble standard deviation compared to the absolute value of the mean-truth misfit. Clearly, the truth is well represented by the mean of the ensemble. Figure 13.3 shows that although the spread around the truth is underestimated at several locations, it is overestimated elsewhere.

Finally, Fig. 13.7 shows that the weights are distributed as they should be: they display small variance around the equal weight value 1/24 for the 24 particles. Note that the particles with zero weight had too small a weight to be included in the almost-equal weight scheme and will be resampled from the rest.

Because the weights vary so little, they weights can be used back in time, generating a smoother solution for this high-dimensional problem with only 24 particles.

13.5 Conclusions

To try to solve strongly nonlinear data assimilation problems, we have discussed particle filters in this chapter. They have a few strong assets, namely their full nonlinearity, the simplicity of their implementation (although this tends to be lost in more advanced variants), the fact that balances are automatically fulfilled (although, again, more advanced methods might break this), and, quite importantly, that their behaviour does not depend on a correct specification of the model state covariance matrix.

We have also seen their weaknesses in terms of efficiency—the filter degeneracy problem that plagues the simpler implementations. However, recent progress seems to

suggest that we are quite close to solving this problem with developments such as the implicit particle filter and the equivalent-weights particle filter. Also, the approximations are becoming more advanced too, and perhaps we do not need a fully nonlinear data assimilation method for real applications.

There is a wealth of new approximate particle filters that typically shift between a full particle filter and an ensemble Kalman filter, depending on the degeneracy encountered. Gaussian mixture models for the prior are especially popular. I have refrained from trying to give an overview here, since there is just too much going on in this area. A brief discussion is given in van Leeuwen (2009)—again not completely up to date. In a few years, time, we will have learned what is useful and what is not.

Specifically, I should like to mention the rank histogram filter of Anderson (2010). It approximates the prior ensemble in observation space with a histogram, assuming Gaussian tails at both end members. It then performs Bayes' theorem and multiplies this prior with the likelihood to form the posterior. Samples from this posterior are generated as follows. First, the cumulative probability of the posterior at each prior particle is calculated by integrating the posterior over the regions between the prior particles. We want the posterior particles to have equal probability $1/(N+1)$, and so cumulative probability $n/(N+1)$ for ordered particle n. Therefore, the position of each new particle is found by integrating the posterior pdf $1/(N+1)$ further from the previous new member. As Anderson shows, this entails solving a simple quadratic equation for each particle, with special treatment of the tails.

A few comments are in order. First, the prior is not assumed to be Gaussian, and the likelihood also can be highly non-Gaussian, which is good. However, a potential problem is that the above procedure is performed on each dimension separately, and it is unclear how to combine these dimensions into sensible particles. Localization has to be applied to keep the numerical calculations manageable, and inflation is also needed to avoid ensemble collapse. Also, as far as I can see, when the observations are correlated, the operations explained above have to be done in a higher-dimensional space, making the method more complicated. Finally, the method interpolates in state space, which potentially leads to unbalanced states. Anderson applied the method to a 28 000-dimensional atmospheric model with very promising results.

A word of caution is needed. The contents of this chapter express my present knowledge of the field, and no doubt miss important contributions. Also, the field is developing so rapidly—we have aroused the interest of applied mathematicians and statisticians to our geoscience problems—that it is becoming extremely hard to keep track of all interesting work. (That all these communities publish in their own journals does not make life easy, I am now reviewing particle filter articles in over 20 journals.)

Finally, it must be said that the methods discussed here have a strong bias to state estimation. One could argue that this is fine for prediction purposes, but for model improvement (and thus indirectly forecasting), parameter estimation is of more interest (and then there is the question of parameterisation estimation). Unfortunately, no efficient particle filter schemes exist for that problem. This is a growing field that needs much more input from bright scientists like you, reader!

References

Ades, M. and van Leeuwen, P. J. (2013). An exploration of the equivalent weights particle filter. *Q. J. R. Meterol. Soc.*, **139**, 820–840.

Anderson, J. (2010). A non-Gaussian ensemble filter update for data assimilation. *Mon. Weather. Rev.*, **138**, 4186–4198.

Chorin, A., Morzfeld, M., and Tu, X. (2010). Implicit particle filters for data assimilation. *Commun. Appl. Math. Comput. Sci.*, **5**, 221–240.

Chorin, A. J. and Tu, X. (2009). Implicit sampling for particle filters. *Proc. Nat. Acad. Sci. USA*, **106**, 17249–17254.

Doucet, A., De Freitas, N., and Gordon, N. (2001). *Sequential Monte-Carlo methods in Practice*. Springer-Verlag, Berlin.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel-approach to nonlinear non-Gaussian Bayesian state estimation. *IEE Proce. F: Radar Sig. Process.*, **140**, 107–113.

Morzfeld, M. and Chorin, A. J. (2012). Implicit particle filtering for models with partial noise, and an application to geomagnetic data assimilation. *Nonlin. Process. Geophys.*, **19**, 365–382.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: auxiliary particle filters. *J. Am. Statist. Assoc.*, **94**, 590–599.

Snyder, C., Bengtsson, T., Bickel, P., and Anderson, J. L. (2008). Obstacles to high-dimensional particle filtering. *Mon. Weather Rev.*, **136**, 4629–4640.

van Leeuwen, P. J. (2009). Particle filtering in geophysical systems. *Mon. Weather. Rev.*, **132**, 4089–4114.

van Leeuwen, P. J. (2010). Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Q. J. R. Meteoral. Soc.*, **136**, 1991–1999.

van Leeuwen, P. J. (2011). Efficient nonlinear data-assimilation in geophysical fluid dynamics. *Comput. Fluids*, **46**, 52–58.

van Leeuwen, P. J. and Ades, M. (2013). Efficient Fully nonlinear data assimilation for geophysical fluid dynamics. *Comput. Geosciences*, **55**, 16–27.