Peter Jan van Leeuwen

Nonlinear Data Assimilation for high-dimensional systems

–with geophysical applications –

December 22, 2014

Springer

Contents

Chapter 1
Introduction

1.1 What is data assimilation?

Loosely defined, data assimilation combines past knowledge of a system in the form of a numerical model with new information about that system in the form of observations of that system. It is a technique used every day in e.g. numerical weather forecasting, but its use is much more widespread. For instance it is used in ocean forecasting, hydrology, space weather, taffic control, image processing, etc, etc. Typically, the latter applications are considered inverse problems, and we discuss that view below.

This past knowledge is most generally represented by a probability measure. Since we will assume that the model of the system under study is finite dimensional we will assume this prior knowledge is available in the form of a probability density function, or pdf, $p(X = x)$, in which X is the random variable corresponding to model states, and $x \in \Re^{N_x}$ is a specific model state, represented as a state vector. This pdf is assumed to be known from model integrations.

The pdf denotes our uncertainty of the whereabouts of the true system, evolving according to

$$x_t^n = f_t(x_t^{n-1}) + \beta_t^n \tag{1.1}$$

in which the superscript is the time index, $f : \Re^{N_x} \rightarrow \Re^{N_x}$ maps the state one time step forward and denotes the deterministic part of the true evolution equation and β^n is the stochastic part of the true evolution. We have taken a discrete representation in time here. The dimension of the true system is typically much larger than of the modelled system. It could be infinite dimensional. Also, it is an interesting discussion wether the real system, so nature, is deterministic of stochastic. Some interpretations of quantum theory point to the latter, but the debate is still open. However, these discussion is not essential for our endeavour. The reason is that our (computer) model of the true system is always finite dimensional and will always contain errors, so our model system evolves like:

$$x^n = f(x^{n-1}) + \beta^n \tag{1.2}$$

in which the superscript is the time index, $f(x_t^{n-1})$ denotes that part of the true evolution equation that we have captured in our model, β^n is that part of the evolution equation that we have not modelled deterministically (e.g. because we don't know its exact form, or because we don't have the computer resources to take that part also into account, we come back to this later), represented here as a stochastic term. All we can ask for is the pdf of this model state.

The observations, represented by vector y, are taken from the real or true system under study and their uncertainty is also represented as a pdf: $p(Y = y|X = x_t)$, with $y \in \Re^{N_y}$ and in which x_t denotes the true system. This pdf is also assumed to be known, at least up to a normalisation constant. These observation vectors are finite dimensional as each actual measurement is a spatial and temporal average. Importantly, if we know this probability density as function of the state x_t, we know the probability density of this set of observations if they would arise from another state x, as $p(y|x)$, by just replacing x_t with x in the expression for $p(y|x_t)$. One issue here is of course that our model states are not at the same spatial and temporal resolution of the true system. This is one of the sources of the so-called representation errors (see e.g. Cohn, 1997; Van Leeuwen, 2014), a proper discussion of which would be a paper on its own. Here we assume we know what $p(y|x)$ is, with x a state vector from our numerical model.

At the heart of data assimilation lies Bayes Theorem. The theorem is easily derived from the definition of a conditional probability density (pdf from now on) as follows. The conditional pdf of state x given a set of observations y, so $p(x|y)$, can be defined as:

$$p(x,y) = p(x|y)p(y) \tag{1.3}$$

Similarly we can define the pdf of the observations y given state vector x as:

$$p(x,y) = p(y|x)p(x) \tag{1.4}$$

Equating these two expressions for the joint pdf $p(x,y)$ we find:

$$p(x|y) = \frac{p(y|x)}{p(y)}p(x) \tag{1.5}$$

which is Bayes theorem. Although the derivation is simple its interpretation is more subtle. Firstly $p(x)$ is the prior pdf which contains all our knowledge of the state vector before the observations are seen. Bayes Theorem then tells us how this pdf is updated by the observations y to form $p(x|y)$. What we need to do is multiply the prior pdf by $p(y|x)$, which is called the likelihood. To use this the observations have to be known, so y is a known vector, and $p(y|x)$ should be considered as a function of x. Finally, the result should be divided by $p(y)$, the marginal of the observations. We can write this function as:

$$p(y) = \int p(y,x)\, dx = \int p(y|x)p(x)\, dx \tag{1.6}$$

which shows that $p(y)$ is just a normalisation factor needed so that the posterior pdf integrates to one. The pdf $p(y)$ is NOT the pdf of the observations as such. It is the pdf of the observations before the actual measurement is made. This is typically hard to determine, so the expression in terms of $p(y|x)$ above gives a proper way to calculate it.

Bayes Theorem is a most powerful equation, and lies at the heart of all practical data-assimilation methods that have been developed (although it is sometimes difficult to trace it back to this because of the numerous implicit approximations made...).

1.2 How do inverse methods fit in?

As mentioned above, Bayes Theorem tells is that data assimilation is a multiplication problem: given the prior and the likelihood, the solution is the point-wise multiplication of the two. The argument that data assimilation is an inverse problem runs as follows. Reality is evolving in time, and at observation times we make observations of reality. Mathematically this can be expressed as:

$$y = H(\tilde{x}_t) + \varepsilon \tag{1.7}$$

in which $\tilde{x}_t$ is the true state at model resolution, H is the operator that maps that state to observation space, and ε is a realisation of the measurement noise.

The argument now is that data assimilation is an inverse problem because we don't know the truth and we start from the observations y and try to recover $\tilde{x}_t^n$, or its evolution over time. The problem with this argument is that because the observations contain errors we know right from the start that this is impossible. There will always be a whole range of model states or model evolutions that could have generated the observations, typically an infinite number. This is true even in the case when we observe every dimension of the state vector. So the inverse problem formulated in this way is not satisfactorily this far, but let us continue with this thinking as there are a few more issues with the approach.

Instead of the true solution one tries to find a 'best' solution. Mathematically this is formulated as a minimisation problem in which the true solution is approximated as the model state that is closest to all observations, e.g. in a least-squares sense or as minimum absolute distance.This part is quite often ad hoc. Why least squares? Only a coupling to the structure of the observation errors seems to make sense, and indeed, if the errors in the observations are Gaussian distributed a least squares approach will make sense.

An issue is that typically only part of the stat vector is observed, so even a least-squares approach will not lead to a unique solution. This is generally solved by introducing regularisation terms that are added to the least-squares penalty function. An example is a smoothness regularisation term that enforces e.g. small derivatives. Or it is assumed that the solution should be close to another full solution. This can

for instance be enforced again via an extra least-squares term (called ridge regularisation), or via an L1 term (the Lasso), or combinations of these. The literature on this subject is extensive and impressive, not least because of the extensive treatment of he numerical implementation, which is a big issue in itself. As an aside, one issue often overlooked with L1 regularisation (or any regularisation other than L2) is that the solution depends on the coordinate system used, which is a bit awkward. The regularisation term is typically chosen such that it leads to a well-defined unique solution of the inverse problem, both analytically, but also numerically. Or rather, the search is for a stable solution, meaning that a small change in the observations will lead to a small change in the solution. Given the fact that the observations do have errors and nature could have chosen another realisation of this observation error during the measurement process, looking for a stable solution does make sense.

However, the issue with the approach of adding regularisation terms is that they are ad hoc in the sense that they are not related to the physics of the problem but used to enforce a unique or stable solution. It then becomes unclear what the original problem was one is trying to solve, and different regularisations, not withstanding their sophistication, lead to different solutions. Also, it is good to realise that the search for the truth has been abandoned.

A more logical approach is to try to encapsulate all information one has on the solution in a prior before even looking at the observations. Since the prior knowledge is not exact we represent it by a probability density function (pdf). That leads naturally to a Bayesian view of the data assimilation problem. This prior is than changed by the observations using Bayes Theorem, and the solution is an adapted prior called the posterior pdf. The problem is always well defined, ill-posedness does not appear at all, and neither does uniqueness. To me, this is a logical framework as is comes close to the simplest description of knowledge enhancement by us humans: we know our view of the world is not exact, and observations are used to sharpen that view. (One question is of course if we actually work with probabilities, probably not... And indeed, if a new observation comes up of something we had no prior of we tend to become quite upset, and often simply ignore that observation.)

Although intuitively perhaps more appealing, it must be mentioned that enormous difficulties arise when trying to apply Bayes Theorem to spatio-temporal random fields. The size of the problem is overwhelming, perhaps even prohibitively so. Think just about trying to store the prior pdf. Assume we have a 100 dimensional system, and want to use ten probability bins for each dimension to store the pdf in a computer. To do that we would need to store 10^{100} probabilities, which, given that the total number of atoms in the universe is estimated to be about 10^{80}, is a large number. So, necessarily, one needs efficient representations of the pdf (and efficient is not well defined). Also, how accurately can we perform the multiplications needed in Bayes Theorem? Viewed this way, there are ill-posedness problems within the Bayesian view too.

But let us set this discussion aside and concentrate on the real problem at hand, how to generate a good approximation of the posterior pdf as it appears in the geophysical data-assimilation problem.

1.3 Issues in geophysical systems and popular present-day data-assimilation methods

Geophysical systems are special in the sense that the observations are always sparse, observation errors can be quite large, the observation operator $H(x)$ can be strongly nonlinear, and the models used are high-dimensional (up to 10^9 for present-day numerical weather forecasting), and they can be highly nonlinear. Furthermore, error structures in observations and model are poorly known.

On the other hand, we do have model equations that describe the evolution of the system and that are often based on well-trusted physical conservation laws. Typically they are not exact conservation laws as there is always missing physics or biology or chemistry, either because we just don't know what a proper physical description is (e.g. fluid turbulence), or because the computer resources are too small to incorporate all we know. Reality is a combination of these two in most cases.

Given the size of these systems we might as well regard the data-assimilation problem as hopeless. The solution is defined in the introduction as the posterior pdf. Even if this pdf is Gaussian we still need to describe its first two two moments, mean and covariance. For a state vector of size 10^9 the covariance has of the order of 10^{18} entries, more than any supercomputer in the world can store at this moment. Even when we realise that the majority of the entries might be very close to zero, long-range physical correlation do exist, so storing this matrix will at least fill up the largest supercomputer. (Note that as soon as computers become larger we will increase the resolution of the models used, so this problem will not go away.)

In real-world problems, however, the pdf is not Gaussian due to all nonlinearities involved. The pdf can have any shape which leads to even more problematic numbers as shown above.

So what can we do? Well, that is a difficult question, but we can study what has been done. At the moment two approaches have been followed. In the first it is assumed that the prior is Gaussian and the observations errors are Gaussian too, but the observation operator is allowed to be nonlinear. In this system one tries to find the most probable state of the system, the maximum of the posterior pdf, the (global) mode. This is found by looking for the minimum of $J(x) = -\log(p(x|y))$, which, because of the Gaussian assumptions mentioned above, leads to a sum of squares that has to be minimised. Note that the resulting so-called cost function, or penalty function, $J(x)$, is the same as the one appearing in the inverse literature as L2 regularisation or ridge regression, but now it is derived from first principles. This costfunction is minimised using gradient-descent techniques like conjugent gradient. The covariance matrix in the prior is stored in operator form, meaning that certain relations between model variables (called balances) are explored to reduce the actual size of the matrix to be stored. Although these relations are only approximately true they do allow one to find meaningful solutions that e.g. allow for skilful weather prediction. Unfortunately it is very difficult to find the posterior covariance, and this is typically not calculated in e.g. numerical weather prediction. Another problem is that, since the optimisation is a local problem, we can only use the in-

verse of the local curvature, the Hessian, as the covariance. If $H(x)$ is a nonlinear function the posterior is not a Gaussian, so the covariance that can be generated is only an approximation. We will discuss this method in more detail later.

Another popular method is the ensemble Kalman filter. This method represents the prior pdf by a number of model states, or ensemble members, typically of the order of 10-500 for geophysical applications. Observations are included by assuming that the prior is Gaussian, the observation errors are Gaussian and $H(x)$ is linear. In that case Bayes Theorem reduces to the Kalman filter, and different methods are employed to generate the posterior ensemble members, all using the Kalman filter equations in one form or the other. These equations are used even when $H(x)$ is nonlinear and the prior is actually non Gaussian, so it is unclear what this method actually does in nonlinear systems (see e.g. Le Gland et al., 2011). However, it is also used quite effectively in e.g. numerical weather prediction. Also this method is discussed later in more detail.

The latest developments in numerical weather prediction are to combine the variational and ensemble Kalman filter approaches to find 'the best of both worlds'. Unfortunately the development is rather ad hoc, driven by operational needs. A more systematic approach that can handle fully nonlinear systems is desperately needed.

1.4 Potential nonlinear data-assimilation methods for geophysical systems

This paper is an attempt to show a possible way forward. As will be shown most standard nonlinear data-assimilation methods are not suitable for high-dimensional systems as they require an enormous amount of model runs, and each model run is a very expensive affair. However, strong developments have been reported recently that make these methods much more efficient, so they are included in the discussion as we need to keep an eye on their further development. In real geophysical applications we cannot afford more than 10 to perhaps 1000 model runs, which rules out most methods explored in other fields. The only exception to date are particle filters. As will be shown particle filters can handle fully non-linear systems and can provide more information than just the mode. Until recently common knowledge was that particle filters are impractical when the dimension of the model is larger than say 10, because the effective ensemble size quickly reduces to one or two, unless a very large number of particles is used. However, recent developments contradict this and efficient particle filters have been generated for high-dimensional systems in nonlinear settings. One of these methods claims even to be efficient independent of the size of the system at hand. A crucial ingredient in these efficient particle filters is their use of observations in a early state to generate particles close to the high probability peaks of the posterior pdf, and this is where the traditional techniques used in the geosciences like variational methods and ensemble Kalman filter ideas will come in handy.

1.5 Organisation of this paper

In this paper we will study nonlinear data assimilation for geophysical problems in a unified framework that will explore existing suboptimal data assimilation methods. The next chapter discusses standard nonlinear data-assimilation techniques based on Markov Chains, and these methods are contrasted with particle filters.

In chapter 3 the basic idea behind particle filters is presented, followed by why this basic formulation can never work for large-dimensional systems. 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 by 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 the last couple of years, and these developments will be discussed here. We discuss resampling as a way to increase the efficiency of particle filters, and discuss why this is not enough for high-dimensional systems. Then we will discuss proposal densities, which form an important part of this paper. We show that they allow 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, illustrated by a high-dimensional example. This is the first example of undoubtedly an enormous growth in useful methods for extremely high dimensional systems encountered in the geosciences. The paper is closed by a comparison of several nonlinear data assimilation methods, both based on Metropolis-Hastings and several particle filters, on a simple system with variable dimension and a concluding section.

A final word of caution. Firstly, this survey is far from complete, although it tries to incorporate methods and ideas relevant for very high dimensional systems. Secondly, the paper does not strive for mathematical rigour, as will be clear from this introduction. Doing that would lengthen the paper considerably, at the expense of the overview. The philosophy employed here is that the ideas presented will encourage mathematicians to dive into this and help provide the necessary fundamentals.

Chapter 2
Nonlinear data-assimilation methods

In this chapter a few popular nonlinear data-assimilation methods will be discussed to help understanding the issues involved in nonlinear data assimilation, and in particular, the application of these methods in the geosciences. One particular feature of the geosciences that stands out is the high dimension of the systems involved, easily $10^5 - 10^9$, severely restricting the usefulness of several of the methods described here. We will briefly discuss the Gibbs sampler, several variants of Metropolis-Hastings, Hybrid Monte-Carlo, and Langevin sampling (MALA), and contrast them with particle filters. This is necessarily a very incomplete set, and one or more books can be written on all methods now available. Apart from particle filters, all of the methods discussed here have in common that they draw samples from a Markov Chain, so a new state vector trajectory is drawn based on the previous state-vector trajectory sample. While this is a logical strategy exploring previous good samples, it has as drawback that subsequent samples are dependent, leading to slow exploration of the state space. Indeed, as we will see, thousands of samples are typically needed for relatively small dimensional systems. Furthermore, extra samples are needed to obtain a good starting point of the Markov Chain, and most algorithms reject the majority of the samples generated to ensure convergence to the correct posterior pdf. In short, these Markov-Chain methods, even interesting new variants, are very inefficient. The method followed here is to present the basis ideas behind the methods rather than focussing on detailed proofs on concepts and convergence. More information on these methods can be found in e.g. Robert and Casella (2004). A different methodology is the particle filter, which propagates independent samples by construction, so that much more efficient algorithms can be derived. That is why particle filters are the main subject of this paper, and expanded upon in the next chapters.

The standard setting is that we want to sample state vectors, or state-vector trajectories, that can be used as an efficient representation of the posterior pdf. As discussed in the introduction we are interested in the best representation of the posterior pdf of the state (or trajectory) of our numerical model. This makes our data-assimilation problem discrete in space and time. Hence the state vector $x \in \Re^{N_x}$. Observations $y \in \Re^{N_y}$ are also assumed discrete in space and time.

2.1 The Gibbs sampler

The Gibbs sampler does not generate samples from the full posterior density directly but instead draws marginal samples in such a way that all these marginal draws together generate one new sample from the full density. Since each marginal is one- (or at least low-) dimensional, each of these draws is very cheap, making this a competitive scheme in certain circumstances. Let us see how it works in detail.

To sample from a complicated joint density, which is the posterior density in a data-assimilation problem, one can draw samples from the full marginals. Let us for simplicity write this joint density as $p(x^{(1)},...,x^{(N_x)|y})$ in which the upper index is the spacial coordinate. The Gibbs sampler explores a Markov Chain to generate samples and works as follows:

1) Choose a first sample $x_0 = (x_0^{(1)}, x_0^{(2)}, ..., x_0^{(N_x)})^T$ from some initial density p_0.
2) Obtain a new sample x_n from x_{n-1} by sampling the values:

$$\begin{aligned} x_n^{(1)} &\sim p(x_n^{(1)}|x_{n-1}^{(2)},...,x_{n-1}^{(N_x)}|y) \\ x_n^{(2)} &\sim p(x_n^{(2)}|x_n^{(1)},x_{n-1}^{(3)},...,x_{n-1}^{(N_x)}|y) \\ &\dots \\ x_n^{(d)} &\sim p(x_n^{(d)}|x_n^{(1)},...,x_n^{(N_x-1)}|y) \end{aligned} \tag{2.1}$$

3) Change n to $n+1$ and proceed to 2) until convergence.

Note that each new component $x_n^{(i)}$ is used immediately to draw the next component $x_n^{(i+1)}$. 'Convergence' here means that we have reached the stationary joint distribution $p(x^{(1)},...,x^{(N_x)})$, and one sample is produced. Using properties of Markov Chains this convergence can easily be proved, see e.g. Robert and Casella (2004). The number of samples needed to converge to the target pdf is called the burn-in period. Sampled generated during this period cannot be used to infer the characteristics of the posterior pdf, making this method less attractive. As soon as we have generated this one sample we can either start again with a sample from p_0 or we can continue starting from this sample. Obviously, in the latter case the new sample we generate will be dependent on the this sample, and we want to have independent samples. This means that the above procedure has to be performed several times to generate one next independent sample. How many times depends on the autocorrelation structure of this Markov Chain.

It is important to realise that the Gibbs sampler will only work if the marginal pdfs are easy to draw from. Any nonstandard pdf will make this method prohibitively expensive. As a general rule the Gibbs sampler can be used efficiently in two cases:

1) When the dimension of the system is small the conditional densities can be evaluated before hand, and sampling is just drawing from this small-dimensional density.

2) When the conditional densities are given in parametric form and samples can easily be generated from them.

If we now consider a large-scale geophysical application, e.g. numerical weather forecasting, the first is not the case, and the second is unclear.

As touched upon above, several sampling strategies are available to us, and typically one of the following is used:

1) Generate n chains using different starting points, each with m iterations to reach the stationary distribution. The costs to obtain n independent samples is mn.
2) Generate one single long chain and use the iterates after the m burn-in iterates to the stationary density. Sample only every kth value after the burn in period m. Costs for n independent samples $m+kn$ if k is large enough to ensure independent samples. Large enough is determined from the chain autocorrelation.
3) Combine 1) and 3), so l chains, typically $l < 10$, and keep each kth sample after m iterates. Costs for n independent samples is $lm + kn/l$, with k large enough, see 2).

Strategy 1) is not recommended because it is too expensive: if we want 100 independent samples and m is 10000 (which is a very low estimate), we would need 10^6 model runs, which is way too expensive indeed. If the posterior is relatively smooth without strong local maxima a single chain will do fine. Still, the cost is typically huge: taking $k = 10$, which is a rather low estimate again, we'd need about 20000 samples, still a lot. However, if the high-probability areas are almost disconnected a very long chain is needed to jump to another high-probability area, so a single chain is ineffective. The third option with say $l = 5$ will lead to 52000 samples. It must be said that multiple chains allow for parallel implementation, which would significantly reduce the turn-around time. On the other hand, for a high-dimensional model of say 10^{6-9} model variables these numbers are all way too large

Another way to increase the speed of convergence is the following. Instead of sampling one component of the state vector at a time one can sample a larger subset of the state vector. In high dimensional state spaces that is certainly recommended, especially when components are highly correlated. This is called block sampling.

In very high-dimensional spaces one typically encounters in the geosciences, one tries to generate a first sample directly on the stationary distribution to avoid the expensive and inefficient burn-in period. One of the possibilities is to first generate a 4DVar solution (a minimisation method that will be described in a later section) and start the Gibbs sampler from there. Note that this 4Dvar solution can be a local minimum. This has not been explored in any depth in the geosciences yet, but is potentially very interesting. Experience in numerical weather prediction shows that about 100 model runs (iterations) are needed to come close to the 4Dvar solution, which is much better than having to do the full burn in.

The following methods in this chapter all generate complete samples all at once, avoiding having to calculate the marginals first.

2.2 Metropolis-Hastings sampling

The Metropolis-Hastings sampler proposes a new sample, and then decides to accept it or not given some acceptance criterion. It works as follows:

1) Draw a starting point x_0 from some initial pdf p_0.
2) Move the chain to a new value z drawn from a proposal density $q(z|x_{n-1},y)$
3) Evaluate the acceptance probability of the move $\alpha(x_{n-1},z)$ with

$$\alpha(x_{n-1},z) = min\left\{1, \frac{p(y|z)p(z)}{p(y|x_{n-1})p(x_{n-1})}\frac{q(x_{n-1}|z,y)}{q(z|x_{n-1},y)}\right\} \tag{2.2}$$

4) Draw a random number u from $U(0,1)$ and accept the move , i.e. $x_n = z$ when $u < \alpha$, otherwise reject it, so $x_n = x_{n-1}$.
5) Change n to $n+1$ and return to 2) until convergence.

The **proposal density** is a very important density in Metropolis-Hastings, as it provides potential new samples. Most used proposal densities are the (multivariate) Gaussian and the Cauchy (or Lorentz) density centred around the current value of the chain, so x_{n-1} in this case. The covariance or width of these distributions determines the size of the step in state space and it typically adapted to the acceptance rate of the chain.

Obviously, this method is efficient if the acceptance rate is not too low. One way to achieve this is to make only small moves, so that α is close to 1, and hence the probability of acceptance is high. This would correspond to a small covariance or width of the proposal density. Unfortunately, this means that the chain moves very slowly through state space, and convergence will be slow. One somehow has to construct moves that are large enough to probe state space efficiently, while at the same time keep acceptance rates high. There are no general construction rules for the proposal q to do this. A practical solution is to monitor the acceptance rate, and adjust q such that acceptance rates are between $20\% - 50\%$.

The remarks in the previous chapter about the sampling schemes, e.g. one single long or multiple shorter chains are valid here too. Also the convergence criteria cary over directly, so the method needs a very large number of model runs, especially since most model runs will not be accepted, leading to a valid new sample, but positioned on the previous sample.

We now discuss some specific proposal densities often used in Metropolis-Hastings algorithms.

1) If the proposal density q is symmetric, i.e. $q(x|z,y) = q(z|x,y)$ the acceptance rate α reduces to $\alpha = min\{1, p(y|z)p(z)/(p(y|x_{n-1}p(x_{n-1})\}$, simplifying the calculation. This can be achieved for instance by choosing q a function of $|z-x|$.
2) An often used proposal density is that of the random walk, i.e. $z = x_{n-1} + \xi_n$, with ξ_n a random variable with distribution independent of the samples already obtained in the chain. Popular choices for q are Normal and Student's t. Clearly, the width of q determines the average size of the moves, and is typically chosen as a constant in the range $(0.5,3)$ times the covariance of the chain.

3) The proposal density can also be chosen as not depending on the previous estimate x_{n-1}. (Note that the actual transition density $p(z|x)$ still does depend on x_{n-1}, so the chain is Markov.) A popular choice is the prior density, i.e. our best guess of the density before the new observations come into play. In this case, the acceptance ratio becomes the ratio of the likelihoods as Bayes Theorem shows. The main advantage is that α becomes relatively easy to calculate. However, when the prior and the likelihood disagree the prior samples will not be well positioned to describe the posterior.
4) Instead of updating the complete state vector at once, the so-called *global Metropolis-Hastings sampler*, one can also update individual components, or groups of components of the state vector, the *local Metropolis-Hastings sampler*.
5) Gibbs sampling can be seen as a special case of Metropolis-Hastings in which the proposal density is the density conditional of a certain component given the current values of the other components, as is easy to see.
6) Recent schemes use variants of the so called preconditioned Crank-Nicolson algorithm, allowing for considerable speedup compared to the random walk method. It is described in detail below.

2.2.1 Crank-Nicolson Metropolis Hastings

Instead of using a simple random walk proposal, we can use a general Crank-Nicolson-like proposal as (see e.g. Cotter et al, 2013 for an overview):

$$\left(I+\frac{1}{2}TB^{-1}\right)z=\left(I-\frac{1}{2}TB^{-1}\right)x_{n-1}+\sqrt{2T}\xi \tag{2.3}$$

in which B is the covariance of the prior, T is a preconditioning matrix to be chosen later, $\xi \sim N(0,I)$.

To understand the advantages of this kind of proposal let us look at the case $K=B$. This leads to the preconditioned Crank-Nicolson (pCN) proposal

$$z=\sqrt{1-\beta^2}x_{n-1}+\beta\sqrt{B}\xi_n \tag{2.4}$$

in which $\beta=2\sqrt{2}/3$. The acceptance ratio becomes

$$\alpha(x_{n-1},z)=min\left\{1,\frac{p(y|z)p(z)}{p(y|x_{n-1})p(x_{n-1})}\frac{q(x_{n-1}|z)}{q(z|x_{n-1})}\right\}=min\left\{1,\frac{p(y|z)}{p(y|x_{n-1})}\right\} \tag{2.5}$$

The basic idea is to ensure that if no observations are present, every move is accepted. This leads to much better mixing of the algorithm, which is noteworthily of constant performance when the dimension of the system increases, in contrast with the standard random walk proposal. The reason for this performance is simply that the observations only inform about certain scales in the solution, and beyond those scales we rely on the prior. So if prior moves are always accepted beyond certain

scales, as in the pCN, constant performance with scale reduction can be expected and is indeed found by Cotter et al. (2013). We thus find that this method will be more efficient than a simple random walk in high-dimensional systems.

It turns out that also without preconditioning, so choosing $T = I$ leads to acceptance ratio

$$\alpha(x_{n-1}, z) = min\left\{1, \frac{p(y|z)p(z)}{p(y|x_{n-1})p(x_{n-1})}\frac{q(x_{n-1}|z)}{q(z|x_{n-1})}\right\} = min\left\{1, \frac{p(y|z)}{p(y|x_{n-1})}\right\} \quad (2.6)$$

so again the prior is not part of the acceptance ratio, meaning, again, that scales smaller than those described by the data play no role in the acceptance criterion.

2.3 Hybrid Monte-Carlo Sampling

As mentioned before, if new candidates in the Metropolis-Hastings algorithm are chosen as in a random walk, as is usually done, the distance travelled through state space grows only as the square of the number of steps taken. Larger steps do not solve this as they lead to low acceptance rates.

Using ideas from dynamical systems we can make the method more efficient, see e.g. Duane et al. (1987). The following is an introduction to dynamical systems, followed by the combination with Metropolis-Hasting to the hybrid scheme. The main difference with the standard Metropolis-Hastings scheme is that the hybrid method explores gradient information from the pdf, or more specifically $-\log p(x|y)$. In this sense it resembles variational methods to find the mode of the pdf, like 3DVar or 4DVar. Recall, however, that our goal with these sampling methods is not to find the mode, but to try to represent the full posterior pdf by a set of samples. The random ingredients in the method ensure that the methods do not just walk towards the mode.

2.3.1 Dynamical systems

We will exploit the evolution of a system under Hamiltonian dynamics to be defined shortly. Consider the evolution of state variable x under continuous time τ. In classical mechanics Newton's law describes how particles are accelerated by forces, the famous $F = ma$ in which m is the mass of the particle and a the acceleration, i.e. the second time derivative of the coordinates of a particle. This law being a second derivative in time, the evolution of a system of particles is fully determined if the forces are given, together with the position and velocities of the particles. So each particle is fully determined by its position and velocity coordinates, and these are related through $v^{(i)} = dx^{(i)}/dt$. The space spanned by the position and velocity variables is called phase space.

The probability density of the system can be written as:

$$p(x|y) = \frac{1}{Z_p} e^{-E(x)} \tag{2.7}$$

in which $E(x)$ is called the potential energy of the system when in state x. In a Hamiltonian system forces are conservative, which means that they can be written as the gradient of the potential energy, and Newton's law becomes:

$$\frac{dv}{d\tau} = -\frac{\partial E(x)}{\partial x} \tag{2.8}$$

The kinetic energy is defined as $K(v) = 1/2v^T v$, the sum of the squares of the velocities of the particles. The total energy of the system is the sum of the kinetic and potential energies:

$$H(x,v) = E(x) + K(v) \tag{2.9}$$

where H is called the Hamiltonian of the system. Because the velocities are the time derivatives of the positions and only the kinetic part of the Hamiltonian depends on the velocities, we find the Hamiltonian equations:

$$\begin{aligned} \frac{dx_i}{d\tau} &= \frac{\partial H}{\partial v_i} = v_i \\ \frac{dv_i}{d\tau} &= -\frac{\partial H}{\partial x_i} \end{aligned} \tag{2.10}$$

During the evolution of the system the Hamiltonian is constant:

$$\begin{aligned} \frac{dH}{d\tau} &= \sum_i \left\{ \frac{\partial H}{\partial x_i}\frac{dx_i}{d\tau} + \frac{\partial H}{\partial v_i}\frac{dv_i}{d\tau} \right\} \\ &= \sum_i \left\{ \frac{\partial H}{\partial x_i}\frac{\partial H}{\partial v_i} - \frac{\partial H}{\partial v_i}\frac{\partial H}{\partial x_i} \right\} = 0 \end{aligned} \tag{2.11}$$

which is just energy conservation in this case. Note that this is related directly to the fact that the forces are assumed to be conservative.

Another important feature of Hamiltonian systems is that they preserve volume in phase space, the so-called Liouville Theorem. This can be derived by calculating the divergence of the flow field in phase space. This flow field is given by $V = (dx/dt, dv/dt)$ and its divergence is:

$$\begin{aligned} divV &= \sum_i \left\{ \frac{\partial}{\partial x_i}\frac{dx_i}{d\tau} + \frac{\partial}{\partial v_i}\frac{dv_i}{d\tau} \right\} \\ &= \sum_i \left\{ -\frac{\partial}{\partial x_i}\frac{\partial H}{\partial v_i} + \frac{\partial}{\partial v_i}\frac{\partial H}{\partial x_i} \right\} = 0 \end{aligned} \tag{2.12}$$

Having established these relations we can generate the laws that describe the evolution of the density of the system in a new way. Define the joint density of positions and velocities as:

$$p(x,v|y) = \frac{1}{Z_H} e^{-H(x,v)} \tag{2.13}$$

Because both H and the volume in phase space are conserved, the Hamiltonian dynamics will leave $p(x,v)$ invariant. The connection with sampling becomes apparent when we realise that although H is constant, x and v may change, and large changes in x are possible when v is large, avoiding random walk behaviour.

To exploit this we have to integrate the Hamiltonian equations numerically. Obviously, this will lead to numerical errors, that we would like to minimise. Especially, scheme's that preserve Liouville's Theorem are of importance, as will become clear in the next section. One of those is the leap-frog scheme:

$$\begin{aligned} v(\tau+\varepsilon/2) &= v(\tau) - \frac{\varepsilon}{2}\frac{\partial E}{\partial x}(x(\tau)) \\ x(\tau+\varepsilon) &= x(\tau) + \varepsilon v(\tau+\varepsilon/2) \\ v(\tau+\varepsilon) &= v(\tau+\varepsilon/2) - \frac{\varepsilon}{2}\frac{\partial E}{\partial x}(x(\tau+\varepsilon)) \end{aligned} \tag{2.14}$$

Although each finite ε will lead to numerical errors, we will show in the next section that the hybrid scheme is able to compensate for them. But even with these numerical errors we see that numerical scheme follows Liouville's theorem because the first step changes all v by an amount that only depends on x. So along each line x is constant all v change the same amount, and this first step conserves volume. The same is true for the other two steps in the leap-frog scheme: the updated variable is changed by an amount that only depends on the other variable.

Finally, note that the main difference between the Metropolis-Hastings and the hybrid scheme is that the latter uses gradient information of the density (or rather the log density) while the former does not.

2.3.2 Hybrid Monte-Carlo

The hybrid scheme works as follows:

1) Add a random quantity to the velocity variables v by sampling from $p(v|x)$. Since that density is Gaussian it is relatively easy.
2) Update the position variable (which is the actual variable in the target density) with the leap-frog scheme, choosing a positive or negative value for ε each with probability $1/2$. Take L of those leap-frog steps.
3) Accept the new state (x^*, v^*) with probability

$$\alpha = min\{1, \exp[H(x,v) - H(x^*,v^*)]\} \tag{2.15}$$

4) Return to 1).

Note that one leap-frog step would make the scheme close to a random walk again, which is what we wanted to avoid. This is the reason to take L leap frog

steps, with L not too small. Also note that the stochastic first step is needed to change the total energy of the system. If H would remain constant the chain would not be ergodic.

An important question is how to choose the time step ε. Beskov et al. (2013) show that the time step in the leap-frog scheme should be of order $N_x^{1/4}$, leading to an acceptance rate of 0.651. This is related to a balance between the generating a proposal, which decreases as L increases (less time steps are needed to reach the set integration time), and the number of proposals needed to obtain acceptance, which increases as L increases.

If the numerical scheme used for solving the Hamiltonian equations was without error the value of H would not change, and each step will be accepted with probability one. Due to numerical errors, the value of H may sometimes decrease, leading to a bias in the Metropolis-Hastings scheme. To avoid this we need detailed balance. One way to achieve this is to choose the step in the leap-frog scheme random with equal probability for a positive or a negative value for ε, as we did in the scheme above.

In some applications a slight modification of the scheme is needed to avoid that the scheme returns to its initial position and ergodicity is lost. This can be avoided by choosing the step size in the leap-frog scheme random too. Finally, one could use a slightly different Hamiltonian in the leap-frog steps as long as the acceptance rate is determined using the correct Hamiltonian. This allows one to simplify the actual number of calculations needed.

In order to reduce the dependence in subsequent samples we can replace the Hamiltonian equations by (see Kim et al, 2003):

$$\begin{aligned} \frac{dx_i}{d\tau} &= Av_i \\ \frac{dv_i}{d\tau} &= -A^T \frac{\partial H}{\partial x_i} \end{aligned} \tag{2.16}$$

and try to choose matrix A such that the correlation between the subsequent samples is as small as possible. A disadvantage is that the samples are more expensive to generate. However, a useful property of a cleverly chosen matrix A is that matrix vector calculations can be performed efficiently in $N \log_2 N$ operations, leading to sometimes serious efficiency savings.

Also this method has not been applied to any real-sized geophysical problem, but we can make a judgement based on the number of operations needed to generate one sample. Firstly, one has to calculate $\partial E/\partial x$, which, for a nonlinear model especially, will not be trivial. Typically this will require numerical differentiation which is a complex task to code, and the model run needed to evaluate $\partial E/\partial x$ is typically more expensive than a standard forward run of the model. Secondly, the model has to be run twice for each leap-frog time step, for L leap frog time step. So if $L = 10$ we need the equivalent of 20 model runs to generate one new sample. So, again, to generate say 100 samples we need at least 2000 model runs.

A different approach is followed by Beskov et al. (2011), who study the formulation of Hybrid Monte Carlo on Hilbert spaces. Firstly they argue that when the acceptance rate only depends on the likelihood samples from the prior would always be accepted when the observations are worthless. This is useful because in that case the acceptance rate is robust under model refinement, as for the Crank-Nicholson proposals in Metropolis-Hastings. By formulating Hybrid Monte-Carlo on Hilbert spaces they enforce this requirement automatically. Then they notice that the covariance used for the velocity variables should have similar shape to that of the prior as large scales in the prior should allow for large velocities, so fast movement, in those directions where the length scale is large. The resulting Hamilton equations, and their numerical implementation via a modified leap-frog scheme outperforms some existing schemes.

It may be clear that a lot can be gained by studying more sophisticated splitting scheme of the Hamilton equations; the literature on this subject is growing fast, but is beyond the scope of this paper.

2.4 Langevin Monte-Carlo Sampling

If one uses the Hybrid Monte-Carlo algorithm with only one leap-frog step Langevin Monte-Carlo sampling results. The behaviour of the system is close to that of the random walk.

Typically one omits the acceptance step by accepting all moves directly. In this case, there is no need to calculate the values of the new momentum variables v at the end of the leap-frog step since they will immediately be replaced by new values from the conditional density at the start of the next iteration. So, there is no reason to represent them at all. The scheme consists of the following steps:

1) Draw d random values $\beta^{(i)}$ from d Gaussian distributions $N(0,1)$, where d is the dimension of the system.
2) Calculate a new value for the state vector via

$$x_n = x_{n-1} - \frac{\varepsilon^2}{2}\frac{\partial E(x)}{\partial x} + \varepsilon\beta \tag{2.17}$$

which follows from contracting the momentum and position update in the leap-frog scheme.

One can show that if ε is small the acceptance rate in the Metropolis-Hastings version converges to one, so ignoring the acceptance step is justified.

Computationally the method needs only one evaluation of $\partial E/\partial x$ per sample generated. However, the subsequent samples will not be independent, so we will have to take several samples, so model evaluations, to generate one independent sample, with the number of samples needed dependent on the decorrelation time of the sample time series. In practice this would add a factor of at least 10 to the

scheme, so, again, for independent samples one would need at least 1000 iterations of the scheme, not counting the burn-in period.

We can again make this algorithm much more efficient by exploring a Crank-Nicolson proposal (see Beskos et al, 2008) as follows:

$$\left(I+\frac{1}{2}TB^{-1}\right)z=\left(I-\frac{1}{2}TB^{-1}\right)x_{n-1}-TD\Phi(x_{n-1})+\sqrt{2T}\xi \tag{2.18}$$

in which $\Phi(x_{n-1})=-\log p(y|x_{n-1})$, and again T is the preconditioning matrix. This method has been coined Metropolis-adapted Langevin (MALA).

Beskos et al, (2008) show that the acceptance ratio is only a function of the likelihood, so again superior scaling is expected for those cases where the model has finer resolution than the observations. This exciting result opens the possibility, again, for high-dimensional applications.

The conclusion from these extensions to the random walk Metropolis Hastings is that by choosing an appropriate Crank-Nicolson-based proposal perfect scaling of acceptance ratio with model resolution can be achieved. This opens ways to apply Metropolis-Hastings-like algorithms to high dimensional systems. However, it must be mentioned that perfect scaling does not necessarily mean that the methods are also efficient in high-dimensional applications. The number of samples needed for high-dimensional systems is still much larger than 10-100.

2.5 Discussion and preview

The disadvantage of all methods discussed so far is that by construction one has to perform several model runs to generate one new independent sample from the posterior. Furthermore, if no good first guess is available one has to waste computer time on the burn-in period. The particle filter that we will discuss in the next sections doesn't have these problems, at least not by construction. If, however, a good first guess is available and the Markov chain mixes fast with reasonable acceptance rates some methods do show potential. An example is Metropolis-Hastings with a preconditioned Crank-Nicolson proposal, see e.g. Cotter et al. (2013). This is confirmed below when we compare several of these schemes with state-of-the-art particle filters.

On a practical level, since the samples are generated via a Markov Chain, it is not straightforward to make the algorithm parallel, unless more than one chain is used. Again, in a particle filter the model runs are typically independent from each other, so fully parallel. It is, in principle, possible to use 100 model runs for 100 independent samples from the posterior. As mentioned earlier, common knowledge on particle filters is that much more samples are needed (see e.g. Snyder et al, 2008), but the rest of this article will try to prove that the field has advanced to making particle filters efficient for systems with arbitrary dimensions.

Chapter 3
A simple Particle filter based on Importance Sampling

Particle filters are based on some form of Importance Sampling. A sample is drawn from a proposal density, often the prior, but instead of determining an acceptance ratio based on comparison with another sample, as in Metropolis-hastings-like algorithms, each sample is weighted with the probability of that sample in the proposal compared to the posterior probability of that sample. This is done for several samples, after which all weights are compared to each other, and normalised. So, instead of accepting or rejecting a sample, each sample is accepted, and carries a weight. When e.g. the sample mean is needed this is calculated as the weighted mean.

The above is the most straight-forward implementation of a particle filter and is called Basic Importance Sampling here. Basic Importance sampling is straightforward implementation of Bayes Theorem as we will show below. In the next chapters more sophisticated versions will be discussed.

3.1 Importance Sampling

The idea of importance sampling is as follows. Suppose one wants to draw samples from a pdf $p(x)$. The difficulty is that it is not easy to draw from this pdf, for instance because it is a combination of standard density functions. In importance sampling one generates draws from another pdf, the importance, or proposal pdf, from which it is easy to draw. To make these samples from the target pdf one reweighs the samples of the proposal by how probable that sample would be if drawn from the target pdf.

The idea is illustrated by the following example. Suppose we want to calculate the integral

$$I = \int f(x)p(x)\,dx \tag{3.1}$$

If we could draw samples from $p(x)$ directly we would evaluate the integral as follows, using N samples $x_i \sim p(x)$:

$$I_N = \int f(x) \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i)\, dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{3.2}$$

However, as mentioned above, we cannot make these draws efficiently. So what we do is introduce a pdf $g(x)$ from which it is easy to draw, e.g. a Uniform or a Gaussian pdf. We can now write:

$$I = \int f(x) p(x)\, dx = \int f(x) \frac{p(x)}{g(x)} g(x)\, dx \tag{3.3}$$

and draw samples $x_i \sim g(x)$, to find:

$$I = \int f(x) \frac{p(x)}{g(x)} \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i)\, dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \frac{p(x_i)}{g(x_i)} \tag{3.4}$$

So we can conclude that I am allowed to draw samples from a different density but I have to compensate for this by attaching an extra factor, or *weight* $p(x_i)/g(x_i)$ to each sample. In the following we explore this idea in Bayes theorem, using the posterior pdf as the target pdf and the prior pdf as the proposal pdf. In later chapter we will explore more exotic proposals to increase the efficiency of the method.

3.2 Basic Importance Sampling

The basic idea of particle filtering is to represent the prior pdf by a set of particles x_i, each of which is a full state vector or a full model trajectory as follows:

$$p(x) = \sum_{i=1}^{N} \frac{1}{N} \delta(x - x_i) \tag{3.5}$$

Using Bayes Theorem

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)\, dx} \tag{3.6}$$

leads to

$$p(x|y) = \sum_{i=1}^{N} w_i \delta(x - x_i) \tag{3.7}$$

in which the weights w_i are given by:

$$w_i = \frac{p(y|x_i)}{\sum_{j=1}^{N} p(y|x_j)} \tag{3.8}$$

which is just a number we can calculate directly as detailed in the previous section.

Weighting the particles just means that their relative importance in the probability density changes. For instance, if we want to know the expectation of the function

$f(x)$ we now have:

$$E[f(x)] = \int f(x)p(x)\,dx \approx \sum_{i=1}^{N} w_i f(x_i) \tag{3.9}$$

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

It is important to realise what we have done: we didn't draw samples from the posterior, as that is difficult, but we did draw samples from the prior. As mentioned in the first section in this chapter we should attach to each sample, or particle, a weight $p(x|y)/p(x)$. Because the posterior can be written as the normalised product of the prior and the likelihood, $p(x|y) = p(y|x)p(x)/p(y)$, only the normalised likelihood $p(y|x)/p(y)$ remains in the weights.

Let us now assume that the model is discrete in time and define $x^{0:n} = (x^0, ..., x^n)^T$ in which the superscript denotes the time index. A practical way to implement the particle filter is to calculate the one time state or the trajectory sequentially over time, which is where the name 'filter' comes from. The idea is to write the prior density as

$$p(x^{0:n}) = p(x^n|x^{0:n-1})p(x^{0:n-1}) \tag{3.10}$$

Using that the state vector evolution is Markov, i.e. all information needed to predict the future is contained in the present, and the past adds no extra information, we can write:

$$p(x^{0:n}) = p(x^n|x^{n-1})p(x^{0:n-1}) \tag{3.11}$$

Using this again on $p(x^{0:n-1})$, and on $p(x^{0:n-2})$, etc. we find:

$$p(x^{0:n}) = p(x^n|x^{n-1})p(x^{n-1}|x^{n-2})...p(x^1|x^0)p(x^0) \tag{3.12}$$

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:

$$x^n = f(x^{n-1}) + \beta^n \tag{3.13}$$

in which $f(x^{n-1})$ is the deterministic part of the model and β^n is the stochastic part of the model. The stochastic part is sometimes called the model error, and 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 from its density and evaluate the model equation given above. In fact, for normal, or Gaussian, distributed model errors β^n with mean zero and covariance Q, we can write:

$$p(x^n|x^{n-1}) = N(f(x^{n-1}), Q) \tag{3.14}$$

stating that the mean of the pdf of x^n given the state at time $n-1$ is equal to the deterministic model evolution from that given state at time $n-1$, and the covariance of that pdf comes from the covariance of the stochastic part.

Model errors can be additive or multiplicative. Additive errors do not depend on the state of the system, while while multiplicative errors do. We assume the model errors are additive in this paper to be able to derive analytical expressions later on and most progress has been made with this kind of errors.

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:

$$p(y^{1:n}|x^{0:n}) = p(y^n|x^n)...p(y^1|x^1) \tag{3.15}$$

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:

$$\begin{aligned} p(x^{0:n}|y^{1:n}) &= \frac{p(y^{1:n}|x^{0:n})p(x^{0:n})}{p(y^{1:n})} \\ &= \frac{p(y^n|x^n)...p(y^1|x^1)p(x^n|x^{n-1})...p(x^1|x^0)p(x^0)}{p(y^n)...p(y^1)} \\ &= \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)}...\frac{p(y^1|x^1)p(x^1|x^0)}{p(y^1)}p(x^0) \end{aligned} \tag{3.16}$$

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

$$p(x^{0:n}|y^{0:n}) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)}p(x^{0:n-1}|y^{1:n-1}) \tag{3.17}$$

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 $p(x^j|x_i^{j-1})$ for each particle i, and each time j. Hence for each particle x_i run the model forward from time $j-1$ to time j using the nonlinear model equations. The stochastic part 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 (3.8), normalise 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 j by one and repeat 2 and 3 until all observations have been processed.

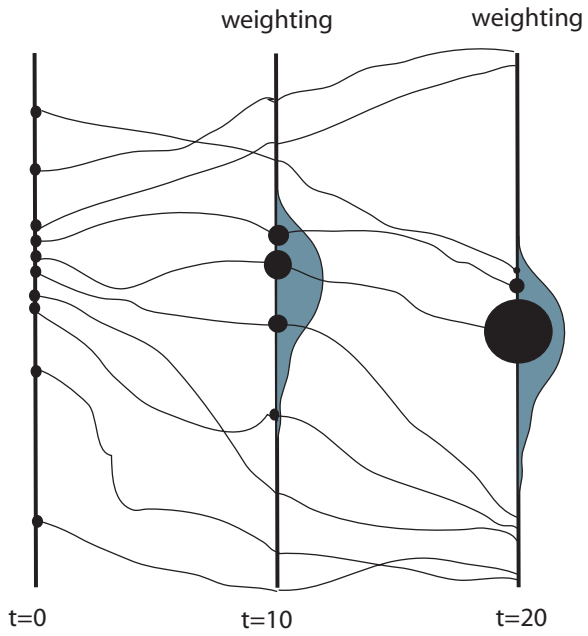


Fig. 3.1 *The standard particle filter with Importance Sampling. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have 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.*

It should be noted that the full posterior pdf over all time steps is created using a sequential algorithm. So, to find the marginal posterior pdf at time zero given all observations up to time n, $p(x^0|y^{1:n})$, one just has to use the full weights at time n over the whole trajectory of each particle. We thus see that the sequential filter algorithm gives rise to a smoother estimate.

The good thing about importance sampling is that the particles are not modified, so that dynamical balances present in a pure model integration (see introduction) are not destroyed by the data assimilation. 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.

It is stressed how simple this scheme is compared to traditional methods like 3- or 4DVar and (Ensemble) Kalman filters. (These schemes are discussed later for those readers not familiar with them.) The success of these schemes depends heavily on the accuracy and error covariances of the model state vector. In 3- and 4DVar this leads to complicated covariance structures to ensure balances etc. In Ensemble Kalman filters artificial tricks like covariance inflation and localisation are needed

to get good results in high dimensional systems. Particle filters do not have these difficulties because the covariance of the state vector is never used.

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 figure 1, at $t = 20$). So the variance in the weights gets very large. That means that the statistical information in the ensemble becomes too low to be meaningful. For instance, if we would calculate the mean, so the weighted mean in this case, this mean is determined by the one particle with weight close to one and the others have no influence because their weight is so low. Furthermore, the weighted sample variance would be (very close to) zero. Clearly, this results in a very poor representation of the posterior pdf. This is called *filter degeneracy* or *ensemble collapse*. It has given importance sampling a low profile until resampling was invented, see the next section.

Chapter 4
Reducing the variance in the weights

In the previous chapter we have seen how simple a particle filter with basic importance sampling is. However, we also noted that this filter will be degenerate for even very small-dimensional systems simply because we multiply weight upon weight. Several methods exists to counteract this behaviour, so to reduce the variance in the weights, and we discuss resampling and the Auxiliary Particle Filter here (see e.g. Van Leeuwen, 2009, for other methods).

4.1 Resampling

In resampling the posterior ensemble is resampled so that the weights become more equal (Gordon et al., 1993). The idea of resampling is simply that particles with very low weights are abandoned, while multiple copies of particles with high weight are kept. 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 $1/N$.

Sequential Importance Re-sampling is identical to Basic Importance Sampling discussed in the previous chapter 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 (3.8) and attach these weights to each corresponding particle.
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.

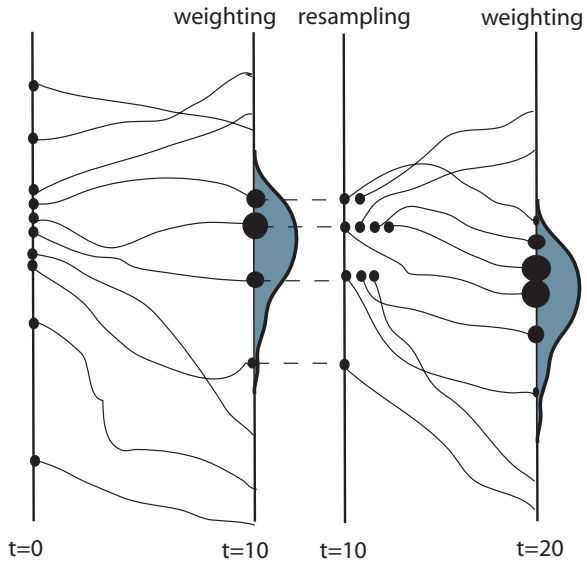


Fig. 4.1 *The Particle Filter with Resampling, also called Sequential Importance Resampling. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time 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 built up of lesser and lesser particles over time, until it consists of only one particle, loosing again all statistical meaning. However, as a filter the method is still very useful as at any time we have a full ensemble of particles given all previous observations.

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

1) *Probabilistic resampling*
   Most straightforwardly 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. This method has been named generalised Bernoulli for those versed in the sampling literature.

2) *Residual Sampling*
   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*
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 $N-1$ 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. While Residual Sampling reduces the sampling noise, it can bee shown that Stochastic Universal Sampling has lowest sampling noise.

Snyder et al. (2008) prove that resampling will not be enough to avoid filter collapse and argue it is related to the dimension of the state vector. The following argument shows a slightly different interpretation. Let us consider two -artificial- particles, one always 0.1σ away from a set of N_y independent observations, and the other 0.2σ away from them, in which σ is the standard deviation of each observations, assumed to be equal for all N_y Gaussian distributed observations. This is admittedly not something that would happen in reality but it does help in bringing two important points home. The point now is that these are two almost perfect particles, what more can you want? The weight of the first particle will be:

$$w_1 = A\exp\left[-\frac{1}{2}(y-H(x_1))^T R^{-1}(y-H(x_1))\right] = A\exp(-0.005N_y) \tag{4.1}$$

while the weight of the second particle is

$$w_2 = A\exp\left[-\frac{1}{2}(y-H(x_2))^T R^{-1}(y-H(x_2))\right] = A\exp(-0.02N_y) \tag{4.2}$$

The ratio of these two weights is

$$\frac{w_2}{w_1} = \exp(-0.015N_y) \tag{4.3}$$

Having a geophysical application in mind, let's assume we have a modest $N_y = 1000$ independent observations. This leads to a ratio of the weights of

$$\frac{w_2}{w_1} = \exp(-15) \approx 3\ 10^{-7} \tag{4.4}$$

so particle two has a negligible weight compared to particle 1 while both are absolutely excellent particles. We can draw two conclusions from this: 1) the number of independent observations is crucial, and 2) the accuracy of the observations is not the main factor, even inaccurate observations will give rise to filter degeneracy if the number of independent observations is large.

This argument is strengthened by the following. Consider the hypersphere with radius r around the observations in N_y dimensional space. The volume of that hypersphere is given by (see e.g. Huber, 1982):

$$V \propto \frac{r_y^N}{\Gamma(N_y/2-1)} \tag{4.5}$$

Let's take for the radius of this hypersphere 3σ we find for this volume, using Stirling for the factorial:

$$V \propto \left[\frac{9\sigma}{N_y/2}\right]^{N_y/2} \tag{4.6}$$

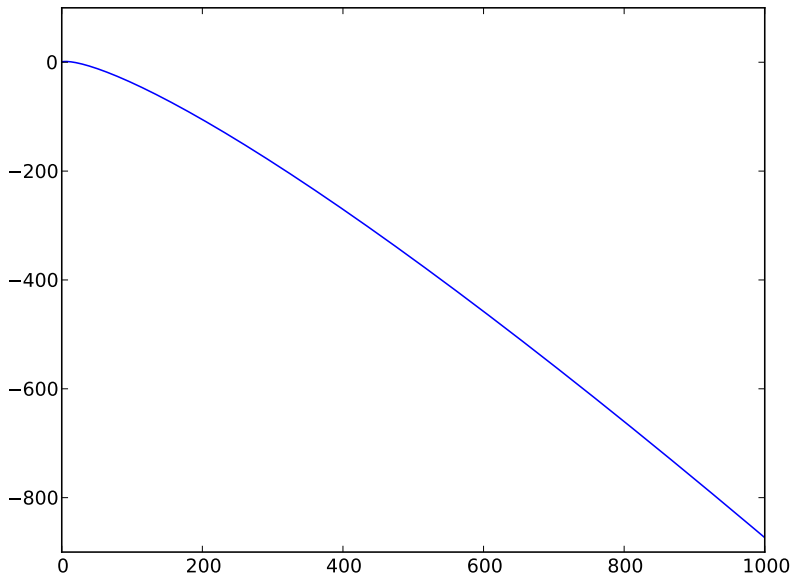


Fig. 4.2 $\log_{10}$ *of the volume in observation space of ball with radius 3 standard deviations (taken as 1 here) against dimension. Note the very rapid decrease of this function.*

This is a very quickly decreasing function of N_y as figure 4.2 shows. So it is highly unlikely for particles to end up in this hypersphere, so closer than 3σ from the observations, when N_y is large. Again we see that the number of observations is the crucial part of degeneracy.

So a reinterpretation of the Snyder et al (2008) results, backed up in a later section when we discuss the optimal proposal density, is that the number of independent observations is the real cause of filter degeneracy because they make the likelihood peak in only a very small portion of the observation space. However, no matter what the cause is, more is needed than simple resampling to solve the degeneracy problem.

4.2 The Auxiliary Particle Filter

Up to now we only considered resampling at the time of the actual observation, and one of the conclusions is that this is already too late: the weights of the particles are to diverse to avoid filter degeneracy. Indeed, as soon as we arrive at observation time and the weights are degenerate the particle filter cannot be saved: the information to save it is just not there. The only thing one can do is start the runs over again, perhaps in a smarter way.

So the question is, is it possible to do something to the particles before observation times? The answer is yes, and several schemes have been proposed. Perhaps the first idea that comes to mind is to run the particles towards the new observations, see how they are doing, and redo the runs with better positioned particles. This is indeed what the Auxiliary Particle Filter does, see Pitt and Shepherd, (1999) for details. The scheme runs as follows (see figure 4.3):

1 Integrate each particle from the previous observations at time $n-m$ to the new observations at time n. This can be done with simplified dynamics (e.g. without model noise) as the reason for these runs is to probe where the observations are.
2 Weight each particle with the new observations as

$$\tilde{w}_i \propto p(y^n|x_i^n) \tag{4.7}$$

where we assumed that the weights are equal at time $n-m$. These weights are called the 'first-stage weights' or the 'simulation weights'.
3 Resample the particles at time $n-m$ with these weights, and use these resampled particles k_i as second part of the proposal density by integrating each forward to n with the full stochastic model, so choosing from $p(x_{ki}^j|x_{k_i}^{j-1})$. Note that k_i connects the original particle i with its new position in state space, that of particle k from the resampling step at $n-m$.
4 Re-weight the members with weights

$$w_i^n = \frac{1}{A}\frac{p(y^n|x_i^n)}{\tilde{w}_{k_i}} \tag{4.8}$$

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

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

A drawback of this scheme is that 2N integrations have to be performed, 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 simplified model like a quasi-geostrophic model for the first set of integrations, and the full model for the second.

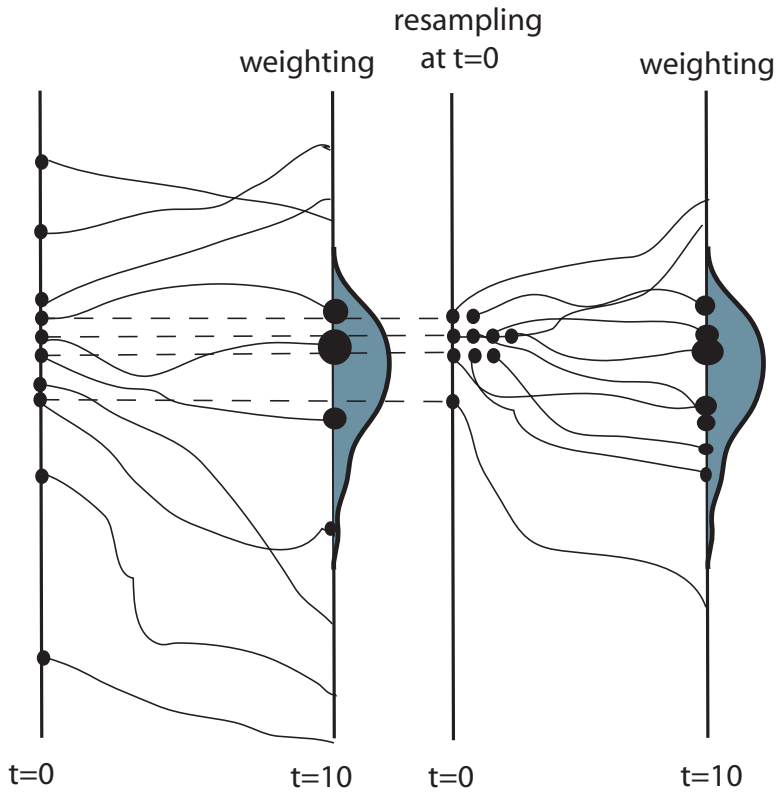


Fig. 4.3 *The Auxiliary Particle Filter. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight at time zero. 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.*

It is possible to do it even more times, zooming in into the likelihood, but at a cost of performing more and more integrations of the model. Unfortunately, my experience with this method is not encouraging for high-dimensional geophysical applications. Other methods based on similar principles, like the Guiding Particle Filter which employs resampling before the actual observations with increased observational errors, have been proposed, see the review by Van Leeuwen (2009), but again experience shows that these methods do not perform well on high-dimensional systems. The conclusion is that one has to 'tell the particles where to go', so give them information on where future observations are. This is indeed possible, as explored in the next section.

4.3 Localisation in particle filters

As we have seen (and will become even more clear later on) the problem with particle filters is the number of observations that form the likelihood. If we could somehow reduce this number particle filters would perform much better. One option is to combine different observations, sometimes called superobbing or summary statistics, but that will reduce the information in the observations, while typically the system is under observed from the start.

Another option is to split the observations into different local in space batches, and use separately Bayes Theorem on these different areas. This process is called localisation. It is a standard technique in Ensemble Kalman Filters. As will be discussed later, Ensemble Kalman Filters approximate the Kalman Filter by representing the mean and covariance of the prior pdf with a small number of model states, typically in the order of 10-500. In the real full covariance correlations between distant points in space will be very close to zero. However, when we try to represent the covariance with let's say 100 states, spurious long-range correlations will result: it is unlikely that 100 random numbers drawn from a zero-mean density add up to zero. To avoid this problem one cuts of the covariance after some distance, either smoothly, or via a hard cut off. In this way any point will only see the observations within this so-called localisation radius. In Ensemble Kalman Filters this procedure also helps with the conditioning of the data assimilation, but that is less of an issue here.

We could do the same in particle filtering. The issue, as pointed out by Van Leeuwen (2009), is that the weight of each particle will vary over space. The issue comes when trying to resample. Particle 1 might do very good in one area, so have a high weight there, and have a very low weight in another area. Should that particle be resampled or not? One option is to only resample it when where its weight is high. But that would mean that we have to somehow stitch particles from different areas together for the following forward integration of the model. Geophysical models are generally very sensitive to artificial large gradients, typically leading to unphysical behaviour. So this direct application of localisation is not possible.

However, recent developments using optimal transportation allow for smoother updates (Reich, 2013). The idea is that at each grid point each new particle can we written as a linear combination of the old particles. By making the coefficients smooth the result will be a smooth new particle. Although the new particles will be smooth in time, that does not necessarily mean they are also 'balanced', i.e. fulfil physical relations between different model variables to a large extend, see introduction. This idea has not been explored in geophysical systems yet, but is definitely of interest and needs further exploration.

Interestingly, using proposal densities is discussed in the next sections implicitly also induces localisation as each grid point sees only observations within an influence region set by the covariance matrix used in the proposal density. So localisation is fully explored in more advanced particle filters.

Chapter 5
Proposal densities

In this section we will concentrate on recent developments in using the so-called proposal transition density in solving the degeneracy problem.

First, we discuss how a proposal density is used in particle filtering, and discuss as simple example the Ensemble Kalman filter at the time of the present observations to move particles towards these observations before the likelihood weights are calculated. In the next chapter we discuss methods that change the model equations by informing intermediate steps between observations about the future observations to steer them in the right direction during the model integrations.

5.1 Proposal densities: theory

We are now to discuss a very interesting property of particle filters that is explored more and more in the geophysical community. As always we start from Bayes Theorem:

$$p(x^{0:n}|y^{0:n}) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n)} p(x^{0:n-1}|y^{1:n-1}) \tag{5.1}$$

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

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int p(x^n|x^{n-1})p(x^{n-1}|y^{1:n-1})\, dx^{n-1} \tag{5.2}$$

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

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int \frac{p(x^n|x^{n-1})}{q(x^n|x^{n-1}, y^n)} q(x^n|x^{n-1}, y^n)p(x^{n-1}|y^{1:n-1})\, dx^{n-1} \tag{5.3}$$

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 ensures we don't divide by zero. Assuming we have an equal-weight ensemble of particles from the previous analysis at time $n-1$:

$$p(x^{n-1}|y^{1:n-1}) = \sum_{i=1}^{N} \frac{1}{N} \delta(x^{n-1} - x_i^{n-1}) \tag{5.4}$$

we find:

$$p(x^n|y^{0:n}) = \sum_{i=1}^{N} \frac{1}{N} \frac{p(y^n|x^n)}{p(y^n)} \frac{p(x^n|x_i^{n-1})}{q(x^n|x_i^{n-1},y^n)} q(x^n|x_i^{n-1},y^n) \tag{5.5}$$

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.

As an example, let us write this modified model as

$$x^n = g(x^{n-1},y^n) + \hat{\beta}^n \tag{5.6}$$

If we assume that $\hat{\beta}^n$ is Gaussian distributed with zero mean and covariance $\hat{Q}$, so $\hat{\beta}^n \sim N(0,\hat{Q})$ we find for the proposed transition density:

$$q(x^n|x^{n-1},y^n) = N(g(x^{n-1},y^n),\hat{Q}) \tag{5.7}$$

so, as for the original transition density, the mean is the deterministic model, and the covariance is that of the stochastic term. But, of course, we don't have to use this Gaussian form, we can do 'whatever we want'.

Back to the general case, drawing from the proposal transition density leads to:

$$p(x^n|y^{0:n}) = \sum_{i=1}^{N} \frac{1}{N} \frac{p(y^n|x_i^n)}{p(y^n)} \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1},y^n)} \delta(x^n - x_i^n) \tag{5.8}$$

so the posterior pdf at time n can be written as:

$$p(x^n|y^{1:n}) = \sum_{i=1}^{N} w_i \delta(x^n - x_i^n) \tag{5.9}$$

with weights w_i given by:

$$w_i \propto p(y^n|x_i^n) \frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1},y^n)} \tag{5.10}$$

where we dropped the factors $1/N^2$ and $p(y^n)$ as they are the same for each particle. We recognise the first factor in this expression as the likelihood weight, and the second as a factor related to using the proposal transition density instead of the original transition density to propagate from time $n-1$ to n. This last factor is related

to the use of the proposed model instead of the original model. These equations form the basis for exploring proposal densities to find efficient particle filters.

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.

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

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int \prod_{j=n-m+1}^{n} p(x^j|x^{j-1}) p(x^{n-m}|y^{1:n-m})\, dx^{n-m:n-1} \tag{5.11}$$

where we used the Markov property of the model. Introducing a proposal transition density at each time step we find:

$$p(x^n|y^{0:n}) = \frac{p(y^n|x^n)}{p(y^n)} \int \prod_{j=n-m+1}^{n} \frac{p(x^j|x^{j-1})}{q(x^j|x^{j-1},y^n)} q(x^j|x^{j-1},y^n) p(x^{n-m}|y^{1:n-m})\, dx^{n-m:n-1} \tag{5.12}$$

Assume that the ensemble at time $n-m$ is an equal weight ensemble, so

$$p(x^{n-m}|y^{1:n-m}) = \sum_{i=1}^{N} \frac{1}{N} \delta(x^{n-m} - x_i^{n-m}) \tag{5.13}$$

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

$$w_i \propto p(y^n|x_i^n) \prod_{j=n-m+1}^{n} \frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1},y^n)} \tag{5.14}$$

So we find that the weights are multiplied by a p/q term each time step in which we use a modified model.

5.2 Moving particles at observation time

One can run the particles to observation time and try to improve the likelihood weights by a proposal density, as in:

$$p(x^n|y^n) = \frac{p(y^n|x^n)}{p(y)} \frac{p(x^n)}{q(x^n|y^n)} q(x^n|y^n) \tag{5.15}$$

The issue is that one has to be able to evaluate the p/q term, which needs the pdf of both. We do have freedom on q, which we can choose to be a Gaussian for instance, but $p(x^n)$ is not at our disposal. Approximating it with e.g. a Gaussian with mean and covariance determined from the ensemble of particles would reduce the method to standard Gaussian data assimilation, which is what we want to avoid.

So, to make progress we have to take the last time step before observations into account as we do know the pdf of $p(x^n|x_i^{n-1})$. The next sections will use the ensemble Kalman filter as proposal density using this last time step in the proposal.

5.2.1 The Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF) is a well-known and often used data-assimilation method on its own. Introduced by Evensen (1994, and corrected by Burgers et al. 1998 and Houtekamer and Mitchell, 1998), it is probably the most-used data assimilation method in the geosciences because of its simplicity. The basic idea is that the prior is assumed to be Gaussian, and also the observation errors are Gaussian, with only weakly nonlinear observation operator. In that case an extremely simple algorithm arises that has influenced the field in a hard to overestimate way. As we are dealing with nonlinear filtering in this paper, the assumption is that the prior is significantly non-Gaussian, so we will not treat this method as a separate data-assimilation method here, but instead only refer to it as a possible proposal density in a particle filter.

The Kalman filter can be derived from Bayes theorem by assuming Gaussian prior and likelihood, and a linear observation operator. This then leads to:

$$\begin{aligned} p(x|y) &= \frac{p(y|x)p(x)}{\int p(y|x)p(x)\,dx} \\ &\propto \exp\left[-\frac{1}{2}(x^n-\bar{x}_f^n)^T P_f^{-1}(x^n-\bar{x}_f^n) - \frac{1}{2}(y-Hx)^T R^{-1}(y-Hx)\right] \\ &\propto \exp\left[-\frac{1}{2}(x^n-\bar{x}_a^n)^T P_a^{-1}(x^n-\bar{x}_a^n)\right] \end{aligned} \tag{5.16}$$

in which $\bar{x}_a^n$ is found from completing the squares on the variable x^n as:

$$\bar{x}_a^n = \bar{x}_f^n + K(y - H\bar{x}_f^n) \tag{5.17}$$

in which K is the Kalman gain given by:

$$K = P_f^n H^T (HP_f^n H^T + R)^{-1} \tag{5.18}$$

The posterior covariance is found as:

$$P_a^n = (1 - KH)P_f^n \tag{5.19}$$

This posterior mean and covariance describe the posterior covariance completely. The specialty of the Kalman filter comes from the fact that the mean and the covariance are propagated between observations with the -assumed linear- model equations:

$$x_f^n = F x_a^{n-1} \tag{5.20}$$

and

$$P_f^n = F P_a^{n-1} F^T + Q \tag{5.21}$$

in which Q is the covariance matrix of the model errors, which are assumed to be distributed according to $N(0,Q)$. These equations can be derived from the Kolmogorov or Fokker-Planck equation as the evolution equations for mean and covariance under linear dynamics (see e.g. Jazwinski, 1970). When the model equations are nonlinear the Kalman Filter is still used as the Extended Kalman Filter in which the model is linearised. However, it can be shown that the evolution equation for the error covariance can be unstable in this approximation. Furthermore, when the dimension of the model state is very large, say 10^9, the Kalman filter covariance becomes too large to be stored.

These two considerations led to the development of the Ensemble Kalman Filter (EnKF). The Ensemble Kalman Filter (EnKF) was introduced by Evensen (1994) and modified by Burgers et al (1998) to correct for a too narrow posterior ensemble, see also Houtekamer and Mitchell (1988). The original formulation by Burgers et al. (1998) considered the EnKF an ensemble approximation to the full pdfs involved. Between observations the pdf evolution is approximated by the evolution of its ensemble representation, in which each member evolves according to the stochastic model equations:

$$x_i^n = f(x_i^{n-1}) + \beta_i^n \tag{5.22}$$

Since each member uses the full nonlinear model the instability problem in the Extended Kalman filter is eliminated. At observation time each ensemble member is forced to follow the transformation of the mean in the Kalman filter, with a few small modifications:

$$x_{i,a}^n = x_{i,f}^n + K^e \left(y - H(x_{i,f}^n) - \varepsilon_i \right) \tag{5.23}$$

(This update scheme is called the stochastic EnKF in recent literature.) The first modification is that the model equivalent of the observations, $H(x_{i,f}^n)$, is perturbed by random vector drawn from the observation error pdf $N(0,R)$. The idea behind this is that each ensemble member is statistically indistinguishable from the true evolution of the system. Since the observation is a direct measurement of the true state perturbed by the observation error, each ensemble member is treated in the same way. (Note that Burgers et al. (1998) use a more pragmatic argument.) And indeed, in using this extra perturbation the posterior ensemble has a covariance which is statistically identical to that to the full Kalman filter. The second modification is the use of the full nonlinear observation operator H in the equation above.

More generally, when the observation operator is nonlinear three approaches can be followed. In the first approach the observation operator is linearised around the present forecast. More sophisticated is the second approach in which the state vector is augmented with the model equivalent of the observations with a nonlinear observation operator, so the new state vector becomes $x_i^{new} = [x^T, H(x)^T]^T$. Observing this state vector leads to a linear observatory operator, and we can use the formalism outlined above. (Note that the relations between the model state x and the observed

model state $H(x)$ is nonlinear, so the prior will not be Gaussian, but that assumption is still made, consistent with the Kalman filter idea. Note also that this is allowed here as it is just a way to generate proposal samples, not to find a best solution to the data-assimilation problem.)

The third approach is to find the mode of the posterior for each ensemble member assuming that each member has a Gaussian prior with covariance determined directly (or after localisation and inflation) from the ensemble (Maximum Likelihood Ensemble Filter by Zupanski, 2005, who used a square-root formulation, see also the Ensemble Randomised Maximum Likelihood Filter of Gu and Oliver (2007), who use the stochastic EnKF). This mode can be found as

$$x_{i,a}^n = argmin\left(\frac{1}{2}(x-x_{i,f}^n)^T P_f^{-1}(x-x_{i,f}^n)+\frac{1}{2}(y^n-H(x)^T R^{-1}(y^n-H(x)\right) \quad (5.24)$$

The solution to this minimisation problem is found iteratively. This method is called the Maximum Likelihood Ensemble Filter (MLEF), while the method searches for the maximum a posteriori state. It has been shown to outperform the EnKF when the observation operator is strongly nonlinear. A second modification is that the observation operator can be nonlinear, and is linearised only in the ensemble Kalman filter gain Matrix $K^e = P_f^e H^T (HP_f^e H^T + R)^{-1}$, with the superscript e denoting the ensemble representation of the covariance.

Before we discuss the EnKF as proposal density four issues should be noted. Firstly, the perturbation of the innovation by ε_i leads to extra sample variance, which is unwanted. Several variants of the EnKF have been derived that avoid these extra perturbations (see e.g. Tippett et al, 2004), all based on square-root approximations of the full Kalman filter.

Secondly, the algorithm depicted above is not very efficient, for instance it looks as if the full error covariance needs still to be stored. Several efficient methods have been developed to avoid this storage problem. They are all based on ensemble representations of the covariance matrix. Define the ensemble perturbation matrix from the ensemble mean as:

$$X_f = \frac{1}{\sqrt{N-1}}(x_{1,f}-\bar{x}_f,...,x_{N,f}-\bar{x}_f) \quad (5.25)$$

in which the first subscript denotes the ensemble member number. The prior covariance can now be written as

$$P_f = X_f X_f^T \quad (5.26)$$

The calculations needed in the Kalman gain matrix are

$$P_f H^T = X_f X_f^T H^T = X_f \left(HX_f\right)^T \quad (5.27)$$

showing that we need only to store matrices of order *model state dimension times observation dimension*. Furthermore

$$HP_f H^T = HX_f X_f^T H^T = \left(HX_f\right)\left(HX_f\right)^T \quad (5.28)$$

is only of size *observation dimension times observation dimension*.

Thirdly, if the ensemble size is N, the rank of the covariance matrix is at most $N-1$, so all observations can only express themselves in a $N-1$ dimensional space, which is quite restrictive if the number of observations is large. Furthermore, model covariances tend to drop to zero when the distance between two points is large. However, when the covariance is constructed using only a limited ensemble size it is unlikely that zero covariances are indeed represented as zero. The issues is that one needs a large number of random numbers drawn from a pdf with mean zero to get a zero mean. This has led to a technique called *localisation* in which covariances over large distances are set to zero. Several localisation methods exist, but no best method has been found. One of the issues is that geophysical systems often exhibit subtle relations between model variables (so-called *model balances*) that can easily be destroyed by localisation.

Finally, the finite ensemble size means that only a finite space is spanned by the ensemble members. This means that the covariance matrix will always be an underestimate of the true covariance, potentially leading to filter divergence in which the ensemble mean deviates strongly from the truth but the ensemble error covariance remains small. A very crude solution is to increase all covariances in the covariance matrix by a fixed number called the *inflation factor*. The crudeness is related to the fact that the covariance is inflated in the directions in which we do have covariance information from the ensemble, not in the missing directions. Nevertheless, this procedure does lead to much better performance of the EnKF. See e.g. the book by Evensen (2009) for more details on these matters.

5.2.2 The Ensemble Kalman Filter as proposal density

As a first example we will explore proposal densities with the Gaussian of the EnKF as the proposal density, as proposed by Papadakis et al. (2010). They call it the Weighted Ensemble Kalman Filter (WEnKF). We know from the chapter on resampling that waiting until the observation time is too late. So we have to combine the last time step of the model before observation time with the EnKF algorithm. Let us assume we have used the EnKF at observation time, so we know what has happened in this last time step: the model is propagated forward with the model equations and the particles are moved around during the EnKF analysis step. So 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:

$$x_i^n = f(x_i^{n-1}) + \beta_i^n \tag{5.29}$$

Hence 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:

$$p(x_i^n|x_i^{n-1}) \propto \exp\left[-1/2\left(x_i^n - f(x_i^{n-1})\right)^T Q^{-1}\left(x_i^n - f(x_i^{n-1})\right)\right] \tag{5.30}$$

This will give us a number for each $[x_i^{n-1},\ x_i^n]$ combination, so each particle. (Note that Papadakis et al, 2010, and also Beyou et al, 2013 make the mistake equating this with the proposal transition density, so they ignore the p/q term from the proposal. This action leads to less variance in the weights, so better performance of the scheme. Livings, private communication)

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

$$x_i^n = x_i^{n,old} + K^e\left(y - \varepsilon_i - H(x_i^{n,old})\right) \tag{5.31}$$

in which ε_i is the random error drawn from $N(0,R)$ as explained before. The particle prior to the analysis comes from that of the previous time step through the stochastic model:

$$x_{i,f}^n = f(x_i^{n-1}) + \beta_i^n \tag{5.32}$$

Combining these two gives:

$$x_{i,a}^n = f(x_i^{n-1}) + \beta_i^n + K^e\left(y - \varepsilon_i - H(x_i^{n-1}) - H(\beta_i^n))\right) \tag{5.33}$$

or

$$x_{i,a}^n = f(x_i^{n-1}) + K^e\left(y - H(f(x_i^{n-1}))\right) + (1 - K^e H)\beta_i^n - K^e\varepsilon_i \tag{5.34}$$

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 spread in the transition density going from x_i^{n-1} to x_i^n. Assuming both model and observation errors to be independent Gaussian distributed we find for this transition density:

$$q(x_i^n|x_i^{n-1}y^n) \propto \exp\left[-1/2\,(x_i^n - \mu_i^n)^T\,\Sigma_i^{-1}\,(x_i^n - \mu_i^n)\right] \tag{5.35}$$

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

$$\mu_i^n = f(x_i^{n-1}) + K^e\left(y - H(x_i^{n-1})\right) \tag{5.36}$$

and the covariance Σ_i is given by:

$$\Sigma_i = (1 - K^e H)Q(1 - K^e H)^T + K^e R K^{eT} \tag{5.37}$$

where we assumed as usual that the model and observation errors are uncorrelated. It should be realised that x_i^n does depend on all $x_{j,f}^n$ via the Kalman gain, that involves the error covariance P^e. Hence we have actually 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 a very large ensemble P^e would

be a variable that depends only on the system, not on specific realisations of that system. This is different from the terms related to x_i^n, that will depend on the specific realisation for β_i^n even when the ensemble size is very large. (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 realising 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, again ignoring normalisation factors as they are the same for each particle. The full procedure is as follows (see figure 5.1):

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 normalise them.
6 Resample

It is good to realise that the EnKF step is only used to draw the particles close to the observations. Hence if the weights are still varying too much, one can do the EnKF step with much smaller observational errors, or do it several times. This might look like over fitting but it is not since the only thing we do in probabilistic sense is to move particles to those positions in state space where the likelihood is large and all tricks we do with the EnKF are just part of the proposal, and can be compensated for by using the correct weights.

Unfortunately, as mentioned above, there is a problem with the WEnKF as proposed by Papadakis (2010). When this mistake is corrected for it becomes clear that this method does not work in high-dimensional systems (Livings, private communication). Again, more is needed, and the next chapter shows ways to move forward.

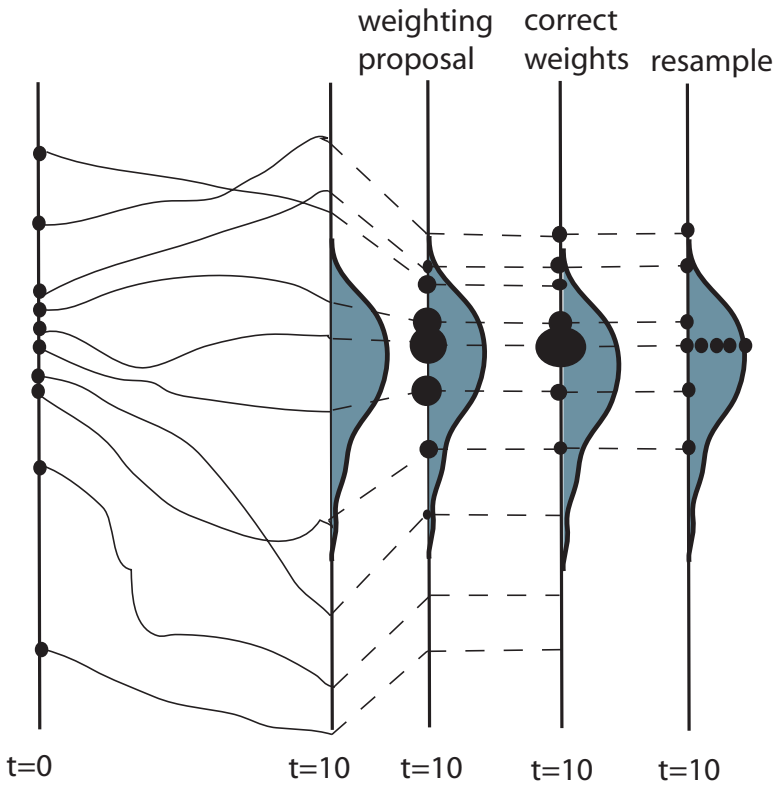


Fig. 5.1 *The particle filter with proposal density. The model variable runs along the vertical axis, the weight of each particle corresponds to the size of the bullets on this axis. The horizontal axis denotes time, with observations at a time interval of 10 time units. All particles have equal weight 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.*

Chapter 6
Changing the model equations

In this chapter we will explore particle filters in which the model equations are changed directly, fully exploring the proposal density freedom. We have in fact done this earlier with the EnKF as proposal, but there we used the EnKF equations to define the proposal density in the last time step before the observations. And in 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. 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, and that is what we will explore in this chapter. By keeping track of the weights associated with this it can be assured that the correct problem is solved, and the particles remain 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 transition density related to the original model. And the efficiency gain exists in the fact that these samples can be dependent on the future observations. To see how this works consider the following example. Write the model equations as:

$$x_i^j = f(x_i^{j-1}) + \beta_i^j \tag{6.1}$$

Let us recall how this equation is related to the transition density of the original model, $p(x_i^j|x_i^{j-1})$. The probability to end up in state x_i^j starting from particle position x_i^{j-1} is related to β_i^j. For instance, if $\beta_i^j = 0$, so no model error, or a so-called perfect model, this probability is 1 if the x_i^j, x_i^{j-1} pair fulfils the perfect model equations $x_i^j = f(x_i^{j-1})$, and zero otherwise. In this case $p(x_i^j|x_i^{j-1})$ is be a delta function centred on $f(x_i^{j-1})$. In the more realistic case the model error is nonzero and the transition density will depend on the distribution of the stochastic random forcing. Assuming Gaussian random forcing with mean zero and covariance Q, so $\beta_i^j \sim N(0,Q)$, we find

$$p(x_i^j|x_i^{j-1}) \propto N(f(x_i^{j-1}),Q) \tag{6.2}$$

so the mean of the Gaussian is the deterministic move, and the covariance of the transition density is determined by the covariance of the stochastic forcing.

Let us now explore a modified model equation, one that 'knows' about future observations, and actually draws the model to those observations. A very simple example is to add a term that relaxes the model to the future observation, like

$$x_i^j = f(x_i^{j-1}) + \beta_i^j + T^j(y^n - H(x_i^{j-1})) \tag{6.3}$$

in which n is the next observation time. The observation operator H does not contain any model integrations, it is just the evaluation of x_i^{j-1} in observation space, simply because x_i^n is not known yet. Clearly, each particle i will now be pulled towards the future observations, with relaxation strength related to matrix T^j. In principle, we are free to choose this matrix, but it is reasonable to assume that it is related to the error covariance of the future observation R, and of the model equations Q. Possible forms will be discussed in the examples later.

Of course, we can't just change the model equations, we have to compensate for this change via the weight of the particles, as explained in chapter 4, where we showed that the proposal density turns up in the weights. Each time step the weight of each particle changes with

$$w_i^j = \frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^j)} \tag{6.4}$$

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

$$p(x_i^j|x_i^{j-1}) \propto \exp\left[-\frac{1}{2}\left(x_i^j - f(x_i^{j-1})\right)^T Q^{-1}\left(x_i^j - f(x_i^{j-1})\right)\right] \tag{6.5}$$

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

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

$$\begin{aligned} q(x_i^j|x_i^{j-1}y^n) &\propto \exp\left[-\frac{1}{2}\left(x_i^j - f(x_i^{j-1}) - T^j(y^n - H(x^{j-1})\right)^T Q^{-1}\left(x_i^j - f(x_i^{j-1}) - T^j(y^n - H(x^{j-1})\right)\right] \\ &= \exp\left[-\frac{1}{2}\beta_i^{jT} Q^{-1}\beta_i^j\right] \end{aligned} \tag{6.6}$$

Again, since we have already chosen 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.

6.1 The 'Optimal' proposal density

In the literature the so-called 'optimal proposal density' is defined (see e.g. Doucet et al, 2000) by taking $q(x^n|x^{n-1},y^n) = p(x^n|x^{n-1},y^n)$. It is argued this choice results in optimal weights, and explains the word 'optimal'. However, it is easy to show that particle filters would never be useful for high-dimensional systems if this choice led to optimal weights, as shown below. Let us study the simplest 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 as $N(0,Q)$ and observation errors are Gaussian distributed according to $N(0,R)$. First, using Bayes Theorem we can write:

$$p(x^n|x^{n-1},y^n) = \frac{p(y^n|x^n)p(x^n|x^{n-1})}{p(y^n|x^{n-1})} \tag{6.7}$$

where we used $p(y^n|x^n,x^{n-1}) = p(y^n|x^n)$, i.e. we don't learn anything new for the pdf of the observations from x^{n-1} when we know x^n. Using this proposal density gives posterior weights:

$$\begin{aligned} w_i &= p(y^n|x_i^n)\frac{p(x_i^n|x_i^{n-1})}{q(x_i^n|x_i^{n-1},y^n)} \\ &= p(y^n|x_i^n)\frac{p(x_i^n|x_i^{n-1})}{p(x_i^n|x_i^{n-1},y^n)} \\ &= p(y^n|x_i^{n-1}) \end{aligned} \tag{6.8}$$

To evaluate this term we can expand it as:

$$w_i = \int p(y^n,x^n|x^{n-1})\,dx^n = \int p(y^n|x^n)p(x^n|x^{n-1})\,dx^n \tag{6.9}$$

in which we 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:

$$w_i \propto \exp\left[-\frac{1}{2}\left(y^n - Hf(x_i^{n-1})\right)^T (HQH^T+R)^{-1}\left(y^n - Hf(x_i^{n-1})\right)\right] \tag{6.10}$$

We will now obtain an order of magnitude estimate for the variance of $-\log w_i$ as function of i. First expand $y - Hf(x_i^{n-1})$ to $y - Hx_t^n + H\left(x_t^n - f(x_i^{n-1})\right)$ in which x_t^n 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 $y - Hx_t^n + H\left(f(x_t^{n-1}) - f(x_i^{n-1})\right) + H\beta_t^n$. 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:

$$-\log(w_i) = \frac{1}{2(V_\beta + V_y)} \sum_{j=1}^{M} \left[y_j - H_j x_t^n + H_j \beta_t^n + H_j \left(f(x_t^{n-1}) - f(x_i^{n-1}) \right) \right]^2 \quad (6.11)$$

The variance of $-\log w_i$ arises from varying ensemble index i. Introduce the constant $\gamma_j = y_j^n - H_j x_t^n + H_j \beta_t^n$ and assume that the model can be linearised as $f(x_i^{n-1}) \approx A x_i^{n-1}$, leading to:

$$-\log(w_i) = \frac{1}{2(V_\beta + V_y)} \sum_{j=1}^{M} \left[\gamma_j + H_j A (x_t^{n-1} - x_i^{n-1}) \right]^2 \quad (6.12)$$

A following 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-central χ^2_M 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 $H_j A P^{n-1} A^T H_j^T$, in which P^{n-1} is the covariance of the model state at time $n-1$. Defining $V_x = H_j A P^{n-1} A^T H_j^T$, we find:

$$-\log(w_i) = \frac{V_x}{2(V_\beta + V_y)} \sum_{j=1}^{M} \frac{\left[\gamma_j + H_j A (x_t^{n-1} - x_i^{n-1}) \right]^2}{V_x} \quad (6.13)$$

Apart from the constant in front the expression above is non-central 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 an order of magnitude for λ by realising 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

$$\frac{M}{2} \left(\frac{V_x}{V_\beta + V_y} \right)^2 \left(1 + 2 \left(\frac{V_\beta + V_y}{V_x} \right) \right) \quad (6.14)$$

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: $V_x \approx (V_\beta + V_y)/M$, see figure 6.1. Clearly, when the number of observations is large (100 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, 2014, unpublished manuscript to be submitted).

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 on the low end. However, we have shown that $-\log(w_i)$ is approximately non-central 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. There is no reason to expect that this variance will behave better for nonlinear models, especially because we didn't make any assumptions on the divergent or contracting characteristics of the linear model.

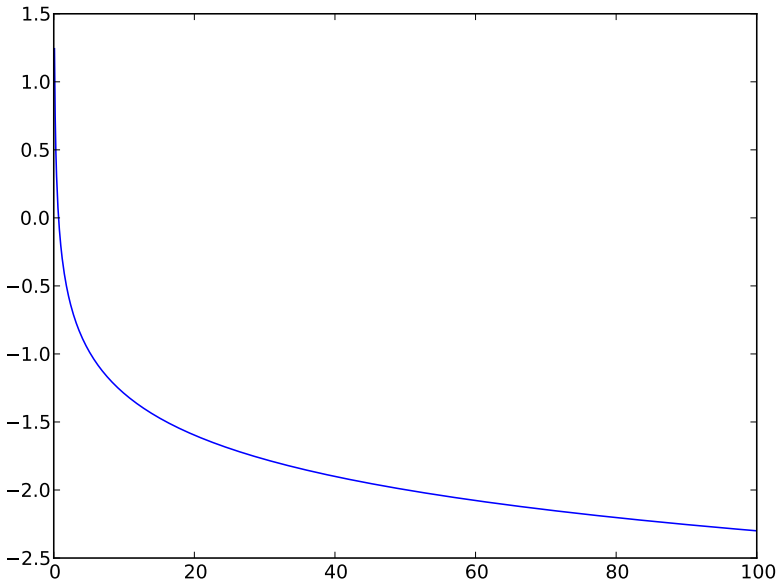


Fig. 6.1 $\log_{10}$ *of the factor multiplying M in the variance of* $-\log w_i$ *as function of* $(V_\beta + V_y)/V_x$. *Note that even when the latter is 100 a system with more than* 1000 *observations will be degenerate.*

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

6.2 The Implicit Particle Filter

The Implicit Particle Filter was introduced in Chorin and Tu (2009) and developed further in Chorin et al., (2010) and Morzfeld et al., (2012). As we shall see, the method is closely related to the 'optimal proposal density' discussed above when the observations are available at every time step.

The method works as follows. Assume we have m model time steps between observations. Define the function $F(..)$ as:

$$F_i(x^{n-m+1:n}) = -\log(p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m})) \tag{6.15}$$

and its minimum

$$\phi_i = \min(F_i(x^{n-m+1:n})) \tag{6.16}$$

Draw a random vector ξ_i of length the size of the state vector times m, with each element of ξ_i is drawn from $N(0,1)$. The actual samples are now constructed by solving

$$F_i(x^{n-m+1:n}) = \frac{\xi_i^T \xi_i}{2} + \phi_i \tag{6.17}$$

for each particle x_i. The term ϕ_i is included to ensure that the equation above has a solution. 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 introduced the subscript to clarify the shape of the pdf. These two are related by a transformation of the probability densities as

$$q_x(x^{n-m+1:n}|x_i^{n-m}, y^n)dx^{n-m:n} = q_\xi(\xi)d\xi \tag{6.18}$$

so that

$$q_x(x^{n-m:n}|x_i^{n-m}, y^n) = q_\xi(\xi)J_i^{-1} \tag{6.19}$$

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

$$\begin{aligned} w_i &= p(y^n|x_i^n)\frac{p(x_i^{n-m+1:n}|x_i^{n-m})}{q_x(x_i^{n-m+1:n}|x_i^{n-m}, y^n)} \\ &= p(y^n|x_i^n)\frac{p(x^{n-m+1:n}|x_i^{n-m})}{q_\xi(\xi_i)}J_i \\ &= \frac{\exp\left(-F_i(x^{n-m+1:n})\right)}{\exp\left(-1/2\xi^T\xi\right)}J_i \\ &= \frac{\exp(-F_i\left(x^{n-m+1:n}\right))}{\exp\left(-F_i(x^{n-m+1:n}) + \phi_i\right)}J_i = \exp(-\phi_i)\,J_i \end{aligned} \tag{6.20}$$

where we used (6.17).

To understand better what this means in terms of the variance of the weights let's 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

$$\begin{aligned} -\log(p(y^n|x^n)p(x^n|x_i^{n-1})) &= \frac{1}{2}\left(y^n - Hx_i^n)\right)^T R^{-1}\left(y^n - Hx_i^n\right)) \\ &+ \frac{1}{2}\left(x^n - f(x_i^{n-1})\right)^T Q^{-1}\left(y^n - f(x_i^{n-1})\right) \\ &= \frac{1}{2}\left(x^n - \hat{x}_i^n)\right)^T P^{-1}\left(x^n - \hat{x}_i^n\right) + \phi_i \end{aligned} \tag{6.21}$$

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 $K = QH^T(HQH^T + R)^{-1}$. Comparing this with (6.17) we find $x_i^n = \hat{x}_i^n + P^{1/2}\xi_i$, so J is a constant, and

$$\begin{aligned}\phi_i &= \min(-\log(p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m}))) \\ &= \frac{1}{2}\left(y^n - Hf(x_i^{n-1})\right)^T (HQH^T + R)^{-1}\left(y^n - Hf(x_i^{n-1})\right)\end{aligned} \tag{6.22}$$

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. Although the Implicit Particle Filter can be used with nonlinear H and more general error pdfs there is no reason to assume that it it will not suffer from filter degeneracy in these more general cases. too.

We have not discussed yet how to actually solve the equation

$$F_i(x^{n-m+1:n}) = \frac{\xi_i^T \xi_i}{2} + \phi_i \tag{6.23}$$

for each particle. Clearly, since F_i is a scalar this equation has an infinite number of solutions. When the model equation is linear and the observation operator H is linear too, this is a quadratic equation in trajectory $x_i^{n-m+1:n}$. In that case F_i can be rewritten as:

$$F_i(x^{n-m+1:n}) = (x_i^{n-m+1:n} - x_i^a)^T P^{-1} (x_i^{n-m+1:n} - x_i^a) + \phi_i \tag{6.24}$$

in which x_i^a and P follow from completing the squares in F_i, as:

$$x_i^a = Mx_i^{n-m} + K(y - H^n M x_i^{n-m}) \tag{6.25}$$

with Mx_i^{n-m} generates the trajectory $(x_i^{n-m+1}, ..., x_i^n)$ using the deterministic model, $H^n M x_i^{n-m} = H^n x_i^n$, and K is the generalised Kalman gain given by:

$$K = BH^{nT}(H^n B H^{nT} + R)^{-1} \tag{6.26}$$

with B the trajectory of the covariance using the deterministic model and evolution equation:

$$B^j = M^{j-1} B^{j-1} M^{j-1} + Q^j \tag{6.27}$$

where Q is the error covariance of the model equations. The covariance P is given by:

$$P = (1 - KH^n)B \tag{6.28}$$

One possible solution is to choose

$$x_i^{n-m+1:n} = x_i^a + P^{1/2}\xi_i \tag{6.29}$$

If the model and/or H is nonlinear, we can either linearise the system and solve the above iteratively (see next section), or use a method called the random map, in which we assume (see Morzfeld et al.)

$$x_i^{n-m+1:n} = x_i^a + \lambda(\xi_i)\xi_i \tag{6.30}$$

in which $\lambda(\xi_i)$ is a scalar function of ξ_i. This leads to a highly nonlinear but scalar equation for λ that is typically easier to solve than solving the high-dimensional problem iteratively.

Finding ϕ_i, the minimum of F_i is close to the solution of a variational method called 4DVar, which we will discuss in the next section, followed by using 4DVar as a proposal density.

6.3 Variational methods as proposal densities

A data-assimilation technique widely used in the geosciences, especially in numerical weather forecasting, is a variational technique called 4DVar. This method tries to find the mode of the posterior pdf joint over time, so the pdf of model trajectories over a certain time window. It is interesting that this mode is chosen, and not the mode of the marginal pdf at final time. Given the nonlinear models used in numerical weather forecasting, these two modes are not the same. The reason is one of computational efficiency as will be explained below.

The prior and likelihood cannot have any shape for this method to be efficient. Typically Gaussian assumptions are made for the prior, the observation errors, and, if they are included, errors in the model equations. Because the model equations are nonlinear the observation operator, which takes the model state at time zero to the observation time and projects it to observation space, is also nonlinear. On top of that, the actual observation operator at observation time can also be a nonlinear operator. In what follows we will discuss how 4DVar is used in present-day numerical weather forecasting, followed by how 4DVar can be made a useful tool for nonlinear non-Gaussian data assimilation.

6.3.1 4DVar as stand-alone method

Let us first look at the case when model errors are ignored, and the unknown is the model state at the start of a time window, so-called *strong-constraint 4DVar*. Hence the solution to the data-assimilation problem is taken as the mode of the marginal pdf at time zero given a set of observations within the window.

Let us have a look at the algorithm in more detail, following Fisher and Auvinen, (2012). The posterior can then be written as:

$$p(x^0|y) \propto p(y|x^0)p(x^0) \propto \exp(-J(x^0)) \tag{6.31}$$

in which $J(x^0)$ is given by:

$$J\left(x^0\right) = \frac{1}{2}\left(x^0 - x_b\right)^T B^{-1}\left(x^0 - x_b\right)$$

$$+ \sum_{j=1}^{M} \frac{1}{2} \left(y^j - \tilde{H}^j\left(x^0\right)\right)^T R^{-1} \left(y^j - \tilde{H}^j(x^0)\right) \tag{6.32}$$

in which $\tilde{H}^j(x^0)$ takes the model state at time zero, propagates it using the deterministic nonlinear model to the time of observation set j, and projects the resulting model state into observation space. The minimisation of $J\left(x^0\right)$ is constrained to the state following the deterministic model equations:

$$x^i = f(x^{i-1}) \tag{6.33}$$

Because the model equations are nonlinear the posterior is not Gaussian in x^0, and the cost function $J(x^0)$ is not quadratic in x^0. This means that the solution cannot be written down in closed form and numerical procedures are needed. In 4DVar, as the name suggests, the minimum of the costfunction (so the maximum of the posterior), is found through variational methods.

Instead of solving the constrained optimisation problem one usually solves the unconstrained problem using Lagrange multipliers:

$$\begin{aligned} J\left(x^0\right) = & \frac{1}{2} \left(x^0 - x_b\right)^T B^{-1} \left(x^0 - x_b\right) \\ & + \sum_{j=1}^{M} \frac{1}{2} \left(y^j - H^j\left(x^j\right)\right)^T R^{-1} \left(y^j - H^j(x^j)\right) \\ & + \sum_{i=1}^{n} \left(\lambda^i\right)^T \left(x^i - f(x^{i-1})\right) \end{aligned} \tag{6.34}$$

with λ^i the Lagrange multiplier fields at times i, and $H^j(x^j)$ maps model state x^j into observation space. Note that the minimisation is now not just over x^0 but over the whole trajectory. The solution of the minimisation problems follows from the calculus of variations as:

$$\frac{\partial J}{\partial x^0} = B^{-1}\left(x^0 - x_b\right) - {F^0}^T \lambda^1 = 0 \tag{6.35}$$

$$\frac{\partial J}{\partial \lambda^i} = x^i - f(x^{i-1}) = 0 \tag{6.36}$$

$$\frac{\partial J}{\partial x^i} = \lambda^i - {F^i}^T \lambda^{i+1} - {H^j}^T R^{-1}\left(y^j - H^j(x^j)\right) = 0 \tag{6.37}$$

$$\frac{\partial J}{\partial x^n} = \lambda^n - {H^n}^T R^{-1}\left(y^n - H^n(x^n)\right) = 0 \tag{6.38}$$

in which F^i is the linearisation of $f(..)$ at time i.

This constitutes a two-point boundary value problem which is solved iteratively. One starts from a first guess for x^0 and integrates the deterministic model forward to time n. Then the adjoint equations for the adjoint variables λ are solved backwards in time from λ^n to λ^1. Finally the expression for λ^1 is used to find a new guess for x^0 and the whole process is repeated until convergence.

Common practice in numerical weather prediction is not to use minimisation methods for non-quadratic cost functions, but instead solve a series of quadratic minimisation methods using Gauss-Newton iteration. This procedure is, perhaps confusingly, called *incremental 4DVar*. Each forward-backward integration is called an *inner loop*, and an *outer loop* is an integration of the full nonlinear system. As an example, the UK Met Office runs a 4Dvar with 50 inner loops and one outer loop, and the European Centre for Medium-range Weather Forecasting ECMWF runs 50 inner loops and 2 outer loops in their 4Dvar.

A popular minimisation method in meteorology is conjugent gradient. The rate of convergence depends on the condition number of the Hessian A, which is defined as the curvature of the cost function, and is given by:

$$A = \frac{\partial^2 J}{\partial x^2} = B^{-1} + \sum_{i=1}^{n} F^{i^T} \tilde{H}^{i^T} R^{-1} \tilde{H}^i F^i \tag{6.39}$$

The condition number is given as the ratio of the largest to the smallest eigenvalue of A:

$$\kappa(A) = \frac{\mu_1}{\mu_{N_x}} \tag{6.40}$$

in which N_x is the dimension of the state vector. It measures how circular symmetric the cost function is. If $\kappa(A) = 1$ all eigenvalues are equal, the costfunction has a perfectly circular symmetric shape, and a gradient descent algorithm will find the minimum in one iteration. However, a large condition number is related to a highly elliptical cost function, and a large number of iterations will be needed for convergence. Since the condition number is typically high related to very small eigenvalues a technique called preconditioning is used. In preconditioning a coordinate transformation is used to reduce the condition number. Define

$$\chi = L^{-1}(x^0 - x_b) \tag{6.41}$$

with

$$B^{-1} = LL^T \tag{6.42}$$

to find for the cost function:

$$\begin{aligned} J\left(x^0\right) = & \frac{1}{2}\chi^T\chi \\ & + \sum_{j=1}^{M} \frac{1}{2}\left(y^j - \tilde{H}^j\left(L\chi + x_b\right)\right)^T R^{-1}\left(y^j - \tilde{H}^j(L\chi + x_b)\right) \end{aligned} \tag{6.43}$$

with Hessian

$$\hat{A} = 1 + L^T \sum_{i=1}^{n} F^{i^T} \tilde{H}^{i^T} R^{-1} \tilde{H}^i F^i L \tag{6.44}$$

This preconditioning will make $\mu_{N_x} = 1$, strongly decreasing the condition number.

But we can do better by exploring the eigenvalues of A. Let is write the result of preconditioning on the Hessian as:

$$\hat{A} = L^T A L \tag{6.45}$$

The best preconditioning would be $L = A^{-1/2}$, which would make all eigenvalues equal to one, resulting in only one minimisation step needed. So the question becomes, can we find an approximation of the Hessian that can easily be inverted? To answer this question let us decompose the Hessian in terms of its eigenvectors as:

$$A = \sum_{i=1}^{K} \mu_i v_i v_i^T \tag{6.46}$$

Define the preconditioning matrix using the first p eigenvectors as:

$$L^{-1} = 1 + \sum_{i=1}^{p} (\sigma_k^{1/2} - 1) v_i v_i^T \tag{6.47}$$

which leads to

$$\hat{A} = L^T A L = \sum_{i=1}^{p} \sigma_i \mu_i v_i v_i^T + \sum_{i=p+1}^{N_x} \mu_i v_i v_i^T \tag{6.48}$$

Now choosing $\sigma_i \mu_i < \mu_{p+1}$ leads to a condition number

$$\kappa(\hat{A}) = \frac{\mu_{p+1}}{\mu_{N_x}} = \mu_{p+1} \tag{6.49}$$

The eigenvectors and eigenvalues can be found from the Lanczos algorithm, which is used to minimise the cost function and at the same time calculate the eigenstructure of the Hessian. This is used by e.g. ECMWF to calculate the eigenstructure in the first inner loops and use them in the next inner loops, resulting in super linear convergence for the second inner loop.

Most centres are now moving away from strong-constrained 4Dvar and are working to include model errors, in a procedure called *weak-constrained 4Dvar*. The posterior now becomes:

$$p(x|y) \propto p(y|x)p(x) \tag{6.50}$$

in which $x = (x^0, ..., x^n)^T$ denotes a whole trajectory of the model. The model equations are written as:

$$x^j = f(x^{i-j}) + \beta^j \tag{6.51}$$

in which β^j is the stochastic part of the model equation at time j. Typically, but not always, β is taken white in time, leading to a Markov process and the prior pdf can be written as:

$$p(x) = p(x^n|x^{n-1})p(x^{n-1}|x^{n-2}), ..., p(x^1|x^0)p(x^0) \tag{6.52}$$

In weak-constrained 4dvar the assumption is made that the model errors are Gaussian distributed, so that we can write

$$p(x|y) \propto p(y|x)p(x) \propto \exp(-J(x)) \tag{6.53}$$

in which $J(x)$ is given by:

$$\begin{aligned} J(x) = & \frac{1}{2}\left(x^0 - x_b)\right)^T B^{-1}\left(x^0 - x_b\right) \\ & + \sum_{j=1}^{M} \frac{1}{2}\left(y^j - H^j(x^j))\right)^T R^{-1}\left(y^j - H^j(x^j))\right) \\ & + \sum_{j=1}^{M} \frac{1}{2}\left(x^j - f(x_i^{j-1})\right)^T Q^{-1}\left(x^j - f(x_i^{j-1})\right) \end{aligned} \tag{6.54}$$

in which Q is the covariance of the model errors. This would allow for extension of the window length as the stochastic part of the model equations tend to limit the memory of the system leading to much better convergence of the algorithm. To make the long-window 4Dvar affordable parallelisation is a must. In the following we discuss methods to make the algorithm parallel. Because the 4DVar is an iterative algorithm the parallelisation can only be achieved by splitting up the time window into smaller chunks which are then treated in parallel.

The costfunction can be viewed in two ways: either the unknowns are the states at all time steps, or the unknowns are the state at time zero and all stochastic forcing terms. Let us concentrate on the first view first. To ease the presentation we concentrate on the inner loops linearised around a first-guess model trajectory from a pure model run (with or without stochastic forcing). Denote the latest estimate in the minimisation as $x^k_{(i)}$ with (i) the iteration number, for each time k and write $\delta x^0 = x^0 - x^0_{(i)}$ and $b = x_b - x^0_{(i)}$. We also have

$$x^j - f(x^{j-1}) = x^j - x^j_{(i)} + x^j_{(i)} - f(x^{j-1} - x^{j-1}_{(i)} + x^{j-1}_{(i)}) \approx \delta x^j - F^j \delta x^{j-1} + \beta^j_{(i)} \tag{6.55}$$

in which we defined $\beta^j_{(i)} = x^j_{(i)} - f(-x^{j-1}_{(i)})$, the estimated stochastic term at iteration (i). The innovation can be approximated as

$$H^j(x^j) - y^j = H(\delta x^j + x^j_{(i)}) - y \approx H\delta x - d \tag{6.56}$$

with $d = H(x^j_{(i)}) - y$.

Using this notation the costfunction can be written compactly as:

$$J(\delta x) = (L\delta x - c)^T D^{-1}(L\delta x - c) + (H\delta x - d)^T R^{-1}(H\delta x - d) \tag{6.57}$$

with $c = (b, \beta^1, ..., \beta^n)^T$, and matrices:

$$L = \begin{pmatrix} I & & & & \\ -F^1 & I & & & \\ & -F^2 & I & & \\ & & \cdots & & \\ & & & -F^n & I \end{pmatrix} \tag{6.58}$$

$$D = \begin{pmatrix} B & & & \\ & Q & & \\ & & Q & \\ & & \cdots & \\ & & & Q \end{pmatrix} \tag{6.59}$$

This formulation can easily be made parallel because the operations $H^j \delta x^j$ and $F^j \delta x^j$ can be computed independently over each sub interval j. However, the preconditioning:

$$\delta x = L^{-1}(D^{1/2}\chi + c) \tag{6.60}$$

leads to difficulties as L^{-1} is equivalent to solving equations of the form $L\delta x = g$ for δx, which has to be solved by forward substitution as (see definition of L):

$$\delta x^j = g^j + F^j \delta x^{j-1} \tag{6.61}$$

which is sequential and connote be made parallel. Approximating L doesn't work because D has small eigenvalues (large spatial scales have near zero variance) and D appears directly in the Hessian for χ. Hence it is very difficult to find efficient and cheap preconditioners.

The alternative view uses as unknown $\gamma = L\delta x$ leading to cost function

$$J(\gamma) = (\gamma - c)^T D^{-1}(\gamma - c) + (HL^{-1}\gamma - d)^T R^{-1}(HL^{-1}\gamma - d) \tag{6.62}$$

Also this form cannot be made parallel because we need L^{-1}.

A solution has been found which explores the Saddle Point Formulation. Rewrite the last cost function as:

$$J(\delta x) = (\gamma - c)^T D^{-1}(\gamma - c) + (\delta w - d)^T R^{-1}(\delta w - d) \tag{6.63}$$

with conditions $\gamma = L\delta x$ and $\delta w = H\delta x$. We can write this as un unconstrained optimisation problem by introducing Lagrange multipliers as:

$$\begin{aligned} \mathscr{L}(\delta x, \beta, \delta w, \lambda, \mu) = {} & (\gamma - c)^T D^{-1}(\gamma - c) + \\ & (\delta w - d)^T R^{-1}(\delta w - d) + \\ & \lambda^T(\gamma - L\delta x) + \mu^T(\delta w - H\delta x) \end{aligned} \tag{6.64}$$

The extremum is found by using calculus of variations leading to:

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \lambda} = 0 \quad &=> \quad \gamma = L\delta x \\ \frac{\partial \mathscr{L}}{\partial \mu} = 0 \quad &=> \quad \delta w = H\delta x \\ \frac{\partial \mathscr{L}}{\partial \gamma} = 0 \quad &=> \quad D^{-1}(\gamma - b) + \lambda = 0 \\ \frac{\partial \mathscr{L}}{\partial \delta w} = 0 \quad &=> \quad R^{-1}(\delta w - d) + \mu = 0 \end{aligned}$$

$$\frac{\partial \mathscr{L}}{\partial \delta x} = 0 \quad => \quad L^T \lambda + H^T \mu = 0 \tag{6.65}$$

Elimination γ and δw leads to the system

$$\begin{aligned} D\lambda + L\delta x &= b \\ R\mu + H\delta x &= d \\ L^T \lambda + H^T \mu &= 0 \end{aligned} \tag{6.66}$$

There are a few advantages of this algorithm. Firstly, there are no inverse matrices involved so one can work directly with the covariances. Secondly, L^{-1} does not appear, so it is relatively easy to generate a parallel algorithm. Finally, precondition is possible with L^{-1} because that matrix can be approximated without making the preconditioning very dependent on this approximation. A possibility is

$$L^{-1} = 1 + (1-L) + (1-l)^2 + ... + (1-L)^{N_x} \tag{6.67}$$

which can be truncated at some order $p << N_x - 1$.

6.3.2 What does 4Dvar actually calculate?

As mentioned above, 4Dvar calculates the mode of a pdf. In strong-constraint 4DVar this is the mode of the marginal pdf at the beginning of the time window in which we have observations. If the model is linear the evolution of this mode to the end of the window, where the actual forecast begins, leads to the mode of the marginal pdf at the end of the time window. This is, indeed, the best starting position for the forecast.

However, when the model is nonlinear evolving the mode of the pdf at the beginning of the window can lead to a very unlikely state at the end of the time window, see figure 6.2. So this could be a very bad initial guess for a forecast. Unfortunately, not much attention to this problem is given in the numerical weather literature.

In weak-constraint 4Dvar the model errors are included and the method calculates the mode of the joint pdf over time. However, also this mode might not lead to the best initial condition for a forecast at the end of the window. The best initial condition for a forecast would be the mode of the marginal pdf at the end of the window. For a nonlinear model the mode of the joint pdf and the mode of the marginal pdf at the end of the window will not be the same, see figure 6.3. This is another issue that is largely ignored.

From the above we can conclude that present-day implementations of 4DVar will not provide the best starting point for a forecast, while that is the very reason for its existence in numerical weather prediction. Also, it does not provide an uncertainty estimate, so it is altogether not a good approximation to the posterior pdf. However, it can be extremely useful to generate effective particles, as explained in the next section.

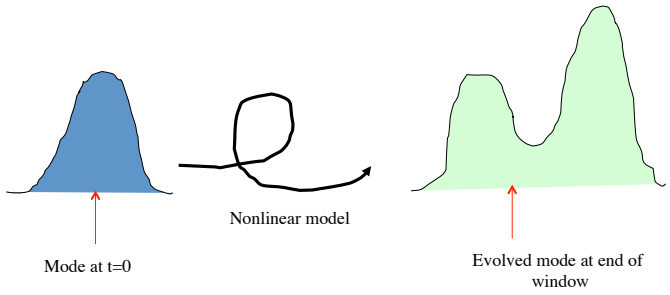


Fig. 6.2 *Illustration of the problem with strong-constraint 4Dvar: the mode at time zero will not be the mode at the end of the window when the model is nonlinear.*

6.3.3 4DVar in a proposal density

The algorithms discussed above cannot be directly used in a particle filter because in a particle filter the particles at time zero, i.e. from the previous posterior pdf, are

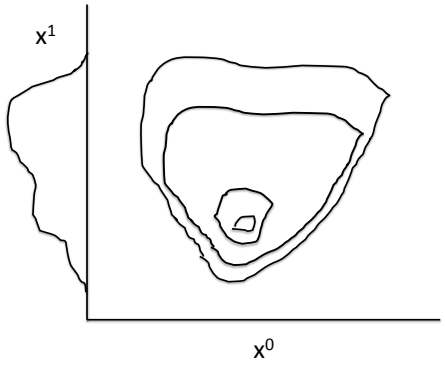


Fig. 6.3 *Illustration of the problem with weak-constraint 4Dvar: the mode of the joint pdf of the state at two time points $p(x^1,x^0)$ will not be the mode of the marginal pdf $p(x^1)$ at the end of the window when the model is nonlinear.*

fixed. So the background term is not part of the cost function. It becomes immediately clear that one has to include model errors to allow these particles to evolve away from the fixed deterministic model, as is consistent with out knowledge of the quality of the models used: model errors are always present in geophysical models, and they tend to be substantial. This means that the now standard strong-constraint 4DVar cannot be used, and we have to rely on the much more limited experience with weak-constraint 4Dvar. This is the reason why we discussed weak-constrained 4Dvar in the previous section. It turns out that leaving out the prior at time zero in the costfunction will still allow us to explore the saddle-point formalism.

The costfunction to be minimised on each particle now becomes:

$$J(x) = \frac{1}{2}\sum_{j=1}^{M}\left(y^j - H_j(x^j)\right)^T R^{-1}\left(y^j - H_j(x^j)\right) + \frac{1}{2}\left(x^n - f(x_i^{n-1})\right)^T Q^{-1}\left(y^n - f(x_i^{n-1})\right) \tag{6.68}$$

in which the state at time zero is given by the position of that specific particle at time zero, so from the posterior pdf at time zero.

The 4DVar on each particle is a deterministic solution, and, since the proposal density cannot be a delta function, we have to include a stochastic part somewhere in the algorithm. A natural way to do this is to use the 4DVar in an Implicit Particle Filter. As explained earlier we first draw a random vector from a multivariate Gaussian and then solve for the particle trajectories from:

$$F_i(x^{n-m+1:n}) - \frac{\xi_i^T \xi_i}{2} - \phi_i = 0 \tag{6.69}$$

This equation can be solved efficiently using an iterative Newton method, which will need solving an adjoint equation at each iteration. The connection with 4Dvar is that finding the minimum of the cost function leads to $\partial J/\partial x = 0$, so finding the zero crossing of $\partial J/\partial x$. This is the same problem as finding the zero crossing of (6.69).

The weights will be given by

$$w_i \propto \exp(-\phi_i)\, J_i \tag{6.70}$$

in which J_i is the Jacobian of the transformation, not to be confused with the costfunction. In this we recognise the standard Implicit Particle Filter formulation, as pointed out in Morzfeld et al, (2012) and Atkins et al. (2013).

A special case appears when we assume that the particles at $n-m$ are not fixed but to be drawn from a known $p(x^{n-m})$. In this case we define

$$F(x^{n-m:n}) = -\log(p(y^n|x^n)p(x^{n-m:n})) \tag{6.71}$$

and its minimum

$$\phi = \min(F(x^{n-m:n})) \tag{6.72}$$

where we note that ϕ does not depend on the particle index i. Again we draw a random vector ξ_i of length the size of the state vector times m. The particles are now constructed by solving

$$F(x^{n-m:n}) = \frac{\xi_i^T \xi_i}{2} + \phi \tag{6.73}$$

and the weights become:

$$\begin{aligned} w_i &= p(y^n|x_i^n)\frac{p(x_i^{n-m:n})}{q_x(x_i^{n-m:n}|y^n)} \\ &= p(y^n|x_i^n)\frac{p(x^{n-m:n})}{q_\xi(\xi_i)}J_i \\ &= \frac{\exp(-F_i(x^{n-m:n})}{\exp(-1/2\xi^T\xi)}J_i \\ &= \frac{\exp(-F_i(x^{n-m:n})}{\exp(-F_i(x^{n-m:n})+\phi)}J_i \\ &\propto J_i \end{aligned} \tag{6.74}$$

so the weights are only dependent on the Jacobian J_i and not dependent on ϕ_i, leading to much better behaviour of the filter. To use this better behaving scheme we have to be able to draw samples from $p(x^{n-m})$. Doing that we break the sequential nature of the filter, and we have to assume a pdf which it is easy to draw from, like a Gaussian.

6.4 The Equivalent-Weights Particle Filter

We have seen that even sophisticated schemes such as the Implicit Particle Filter are likely to show degeneracy when the number of independent observations is large. In this section we discuss a scheme that is not degenerate by construction.

As before we assume we have observations at times $n-m$ and n, and we have an equal weight ensemble of particles at $n-m$, e.g. from a resampling step. We start as in the Implicit Particle Filter with defining:

$$F_i(x^{n-m+1:n}) = -\log(p(y^n|x^n)p(x^{n-m+1:n}|x_i^{n-m}) \tag{6.75}$$

which is minus the logarithm of the numerator of the final weights. Note that the subscript i refers to the particle position at time $n-m$. Its minimum for each particle is given by

$$\phi_i = \min(F_i(x^{n-m+1:n})) \tag{6.76}$$

The trick in the Equivalent-Weights particle filter is to set a target weight w_{target} that a set number of particles has to obtain. This target weight relates to a value of ϕ as $\phi_{target} = -\log w_{target}$. Then each particle trajectory is moved in state space between

times $n-m$ and n to obtain this weight, so we solve:

$$F_i(x^{n-m+1:n}) = \phi_{target} \tag{6.77}$$

We can again explore the wrk-constrained 4DVar algorithm to find a solution to this equation. Call the solution to this equation x_i^*. Solving this equation corresponds to a deterministic move of the particle. Of course,the movement of a particle cannot be fully deterministic. A deterministic proposal would mean that the proposal transition density q is a delta function, and as it appears in the denominator of the weights this leads to division by zero. So we add a stochastic term to the deterministic solution, leading to:

$$x_i^{n-m+1:n} = x_i^* + \beta_i^{n-m+1:n} \tag{6.78}$$

If the amplitude of β is small the stochastic move will be small, and the final weight of the particle will not change much from the target weight, leading to particles with equivalent weights.

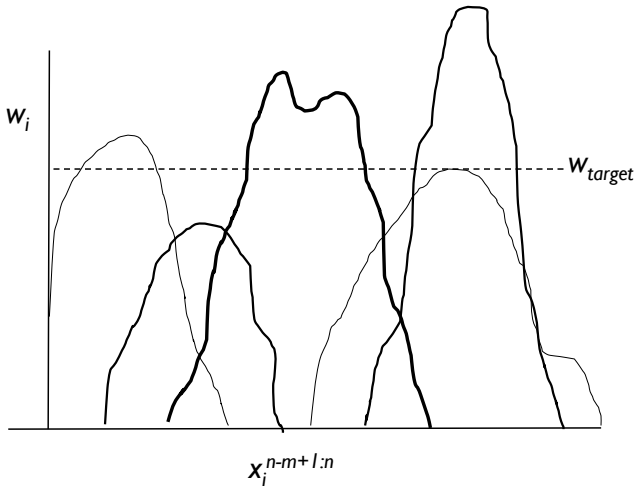


Fig. 6.4 *Schematic of the weights for 5 particles as function of the position of their trajectory through state space. The target weight is set such that 80% of the particles can reach it.*

This scheme has three choices built in that we will discuss below. A first question is how to choose the target weight. Figure 5 contains a schematic that explains the procedure. Firstly, we cannot choose the target weight as the maximum weight that any particle can reach, as only one particle can reach that weight. No matter how the other particles are moved in state space at times $n-m+1:n$ their weight will always be smaller. Secondly, choosing the target weight as the maximum weight

of the worst particle is also not a good idea. This would mean that most particles will have to move quite far from their optimal value for the weight, so away from the observations, and away from where the deterministic model would like to push them. The best choice is somewhere in between. In our experience a target weight that 80% of the particles can reach works well.

This means that we have to move 80% of the particles such that their weight becomes equal to the target weight. Figure 5 shows a 1-dimensional example in which there are typically 2 solutions for each particle in the 80%. However, when the system is 2-dimensional the solution will lie on the intersection of a plane, defining the target weight, and the shape of the weight for each particle, which is a circular shape, so there are an infinite number of solutions. The question then becomes which solution to choose, and how to find it. We can borrow ideas from the Implicit Particle Filter literature to solve this problem, i.e. using the random map.

And what should one do with the other 20%? They cannot reach the target weight so the above procedure does not apply to them. In fact, since their weight will be much lower than that of the other particles when the number of independent observations is large, we can ignore their evolution completely and resample them form the other 80%.

The final choice to make is on the shape of the proposal density, so the density from which final stochastic move is drawn. The proposal transition density could be chosen 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 can be chosen as a mixture density

$$q(x_i^{n-m+1:n}|x_i^{n-m}) = (1-\gamma)U(-a,a) + \gamma N(0,a^2) \tag{6.79}$$

in which 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. a has to be small to ensure the stochastic movement is small so that the weight of the particles will not change much and this close to the target weight.

We can study how the scheme behaves for linear models and linear observation operators H. In that case we can write, as for the implicit particle filter:

$$F_i(x^{n-m+1:n}) = (x_i^{n-m+1:n} - x_i^a)^T P^{-1} (x_i^{n-m+1:n} - x_i^a) + \phi_i \tag{6.80}$$

so we find:

$$(x_i^{n-m+1:n} - x_i^a)^T P^{-1} (x_i^{n-m+1:n} - x_i^a) + \phi_i = \phi_{target} \tag{6.81}$$

A possible solution to this equation can be written as:

$$x_i^a = M x_i^{n-m} + \alpha_i K(y - H^n M x_i^{n-m}) \tag{6.82}$$

in which α_i is a scalar, and, as before, Mx_i^{n-m} generates a trajectory $(x_i^{n-m+1},...,x_i^n)$ using the deterministic model. Clearly, if $\alpha = 1$ we find the solution as proposed in the Implicit Particle Filter. We choose the scalar α_i such that the weights are equal,

so we use this expression in equation (6.81), leading to

$$\alpha = 1 - \sqrt{1 - b_i/a_i} \tag{6.83}$$

in which $a_i = 0.5 d_i^T R^{-1} H^n K d$ and $b_i = 0.5 d_i^T R^{-1} d_i - \phi_{target}$. Here $d_i = y^n - H^n M x_i^{n-m}$.

It is good to realise what we have achieved: a particle filter that generates samples from the high-probability area of the posterior pdf which is not degenerate by construction. The only drawback is that the scheme has one tuning parameter, the percentage of particles kept within the scheme. Experience with a simplified version of the scheme explored below shows that the quality of the method does depend on this choice.

6.4.1 Convergence of the EWPF

A simple requirement for any particle filter seems to be that it should converge to the full posterior from Bayes Theorem when the number of particles increases. However, it is easy to show that this is not the case for the Equivalent-Weights Particle Filter. The main issue is the percentage of particles kept in the equal-weights step. Whatever percentage is chosen we will always loose the tails of the density. Furthermore, a very high percentage will push all particles away from their highest weight, pushing them towards the tails of the posterior (but also never pushing far enough into the tails to cover them because the percentage will always be finite).

There is, however, a very simple remedy. One can make the proposal moves converge to that of the original model with increasing number of particles by e.g. multiplying the deterministic moves away from the original deterministic part of the model by a factor that decreases these terms to zero when N increases. For instance, we can multiply α_i is the equivalent-weights step by a factor $1000/(1000+N)$. It is possible to tune this such as to obtain a smooth transition between the EWPF when N is small, and the SIR when N grows larger.

However, I don't consider this a fruitful way. The EWPF generates high-probability particles from the posterior pdf when N is small. Experiments shown above and in e.g. Ades and Van Leeuwen (2012, 2014) show that the method is very effective doing so, and the ensemble statistics are good, even for very high dimensional systems.

6.4.2 Simple implementations for high-dimensional systems

So far the Equivalent-Weights Particle Filter has been tested on several high-dimensional geophysical systems, but only in a form in which the actual equivalent-weights step is used at the final step before observations. This implementation

avoids complicated iterative searches over several time steps. Between observations a relaxation scheme is used to ensure the particles are reasonably positioned before the Equivalent-Weights step. At the beginning of this chapter we discussed a very simple relaxation scheme to pull the particles towards future observations, written as:

$$x_i^j = f(x_i^{j-1}) + \beta_i^j + T^j(y^n - H(x_i^{j-1}) \tag{6.84}$$

in which n is the next observation time. Unfortunately, this proposal density will not avoid degeneracy in high-dimensional systems with a large number of observations, so we will need an equivalent-weights step at the final time step.

To start, let us recall the expression we found for the proposal density weights in equation 5.14:

$$w_i = p(y^n|x_i^n) \prod_{j=n-m+1}^{n} \frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^n)} \tag{6.85}$$

The model with the extra relaxation term will be used for all but the last time step before observations. This last step will employ the equivalent-weights step as explained in the previous section: 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, see also Van Leeuwen (2010, 2011) for details. 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:

$$-\log w_i = F_i(x^n) \tag{6.86}$$

which for Gaussian distributed model errors and observation errors reduces to

$$F_i(x^n) = \log w_i^{rest} + \frac{1}{2}(y^n - Hx_i^n)^T R^{-1}(y^n - Hx_i^n) + \frac{1}{2}(x_i^n - f(x_i^{n-1}))^T Q^{-1}(x_i^n - f(x_i^{n-1})) \tag{6.87}$$

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 as 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

$$\phi_i = -\log w_i^{rest} + \frac{1}{2}\left(y^n - Hf(x_i^{n-1})\right)^T (HQH^T + R)^{-1}\left(y^n - Hf(x_i^{n-1})\right) \tag{6.88}$$

For N particles this given 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 ϕ_i is smaller than the target value. (Note that we can choose another percentage, see e.g. Ades and Van Leeuwen (2012) who investigate the sensitivity of the filter for values between 70% and 100%.) Define a quantity $C = -\log w_{target}$, and solve for each particle with a minimum weight larger than the target weight

$$F_i(x^n) = -\log(w_{target}) \tag{6.89}$$

So now we have found the positions of the new particles x_i^* such that all 80% have equal weight. The particles that have a larger minimum than $-\log w_{target}$ 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

$$x_i^n = f(x_i^{n-1}) + \alpha_i K[y^n - Hf(x_i^{n-1})] \tag{6.90}$$

in which $K = QH^T(HQH^T + R)^{-1}$, 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

$$\alpha = 1 - \sqrt{1 - b_i/a_i} \tag{6.91}$$

in which $a_i = 0.5x_i^T R^{-1} HKx$ and $b_i = 0.5x_i^T R^{-1} x_i - \log w_{target} - \log w_i^{rest}$. Here $x = y^n - Hf(x_i^{n-1})$, 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 and as explained in the previous section we choose a mixture density:

$$q(x_i^n|x_i^*) = (1-\gamma)U(-a,a) + \gamma N(0,a^2) \tag{6.92}$$

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 equivalent-weight scheme explored in several applications so far 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

$$w_i^j = w_i^{j-1} \frac{p(x_i^j|x_i^{j-1})}{q(x_i^j|x_i^{j-1}, y^n)} \tag{6.93}$$

3 At the last time step before the observations calculate the maximum weights for each particles and determine $F_i(x^n) = -\log w_{target}$
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 (6.92).
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.

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, although the performance of the scheme is dependent on the correct tuning of the scheme parameters T^j and the percentage of particles kept in the equivalent weights step. This method has been studied quite extensively in low-dimensional models (Ades and Van Leeuwen, 2012) and used successfully in several high-dimensional applications, see e.g. Van Leeuwen and Ades (2013), Ades and Van Leeuwen (2014) for an application in a 65,000 dimensional barotropic vorticity equation. We are testing its performance in even higher-dimensional climate models with millions degrees of freedom at the moment.

To illustrate its performance we follow Ades and Van Leeuwen (2014) by showing some insightful extra results. Ades and Van Leeuwen (2014) applied the method to the so-called barotropic vorticity equation model, with a dimension of 65,536. The model solves the evolution equation for the vorticity q:

$$\frac{\partial q}{\partial t} + J(\psi, q) = \beta \tag{6.94}$$

in which ψ is the streamfunction, related to the vorticity as $q = \Delta\psi$, and Δ is the 2-dimensional Laplace operator. The Jacobian $J(a,b) = a_x b_y - a_y b_x$ in which subscripts denote derivatives to the zonal and meridional coordinates x and y, respectively. β is a random space and time dependent forcing representing missing physics. The equations are discretised over a equidistant horizontal grid on a torus, with 256x256 grid points. The evolution equation is solved with a semi-Lagrangian scheme with a time step $\Delta t = 0.04$. The time stepper itself was Euler-Maruyama. After each time step the stream function ψ is calculated using FFTs. This stream function field is then used in the next time step to advect the vorticity fiild further.

This system describes the nonlinear evolution of interacting vortices on a torus. The data assimilation experiment is a so-called identical twin experiment, in which first a truth run is generated using the stochastic model equations and observations are taken from this truth run. These artificial observations are then perturbed by the prescribed observation errors and then assimilated into model runs with different initial condition and stochastic forcing realisations, but using the same model dynamics and statistics for model errors and observation errors.

We compare the results of a data experiment using the SIR filter, which only uses resampling, and the Equivalent-Weights Particle Filter, both with 32 particles. 80% of the particles was kept.

In our case the experiment was run for 1150 time steps with observations every 50 time steps. As the decorrelation timescale, determined from a single model run as the time scale at which the autocorrelation drops below $1/e$, is 42 time steps, the system has time to develop nonlinearly between observation times. The observations are assumed to be uncorrelated in space and time, with standard deviation 0.05. In this experiment we observe every grid point at observation time.

The initial condition was a random field of vortices with length scales of 10 grid points. The initial ensemble was generated by adding Gaussian random fields with variance 0.025^2 and spatial correlations given by

$$\rho(d) = \left(1 + \frac{|d|}{L}\right) \exp\left(-\frac{|d|}{L}\right) \tag{6.95}$$

in which d is the Eulerian distance between grid points and L is the decorrelation length scale of 5 grid points. This correlation structure was also used for the model error covariance, but now with variance $0.025^2 \Delta t$. This random error gives perturbations to the deterministic model move of about 10%.

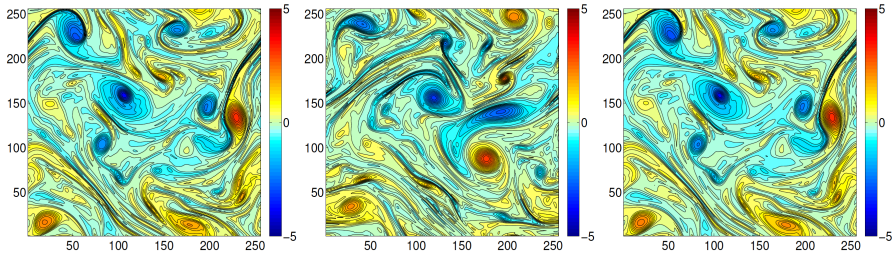


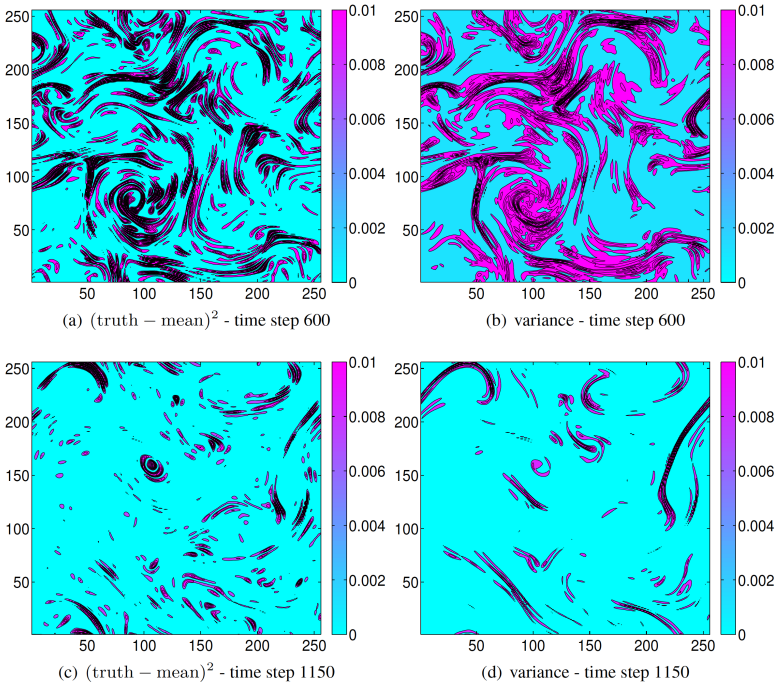
Fig. 6.5 *Vorticity field of the truth (left), the mean of the SIR filter (middle) and the mean of the EWPF (right) at time 1150. Note that the EWPF is quite close to the truth, while the SIR is completely off.*

Figure 6.7 shows the truth, the mean of the ensemble using the SIR filter that only used resampling, and the EWPF. We clearly see that the EWPF manages to capture the position and strength of the vorices, but also the majority of the filament structure. In contrast, the SIR result is strongly different from the truth. This filter is indeed degenerate from the first observation set at day 50 on, as expected.

Figure 6.8 explores the quality of the ensemble spread. It shows the squared error $(mean - truth)^2$ for the EWPF at time 600 and time 1150, compared to the variance in the ensemble. If the filter does well these two should be quite similar. It should be noted that the variance is a statistical measure, while the $(mean - truth)^2$ is a realisation, so the latter is expected to be much more variable, as indeed shown in the figure. Note that the structures in squared error and variance are similar. Also note that the spatial average is 0.046 versus 0.031 at time 600, and 0.0032 versus 0.0030 at time 1150, showing that the squared error and the variance are similar in magnitude.

To further analyse the quality of the ensemble we calculated the rank histogram. This histogram is constructed by ranking the truth in the ensemble for a collection of grid points at all observation times. Figure 6.9 shows that the rank histogram is slightly bulging, meaning that the truth ranks more in the middle of the ensemble than at the extremes. This shows that the ensemble is slightly over dispersive: it's spread slightly too wide. The variance plots above showed that the variance was slightly too small compared to the squared error, so how can we relate these two results? The answer is simply that the pdfs are not Gaussian and have heavier tailes.

At last we have a look at the weight distribution in the EWPF. Figure 6.10 shows the weights before resampling in the EWPF at time 600. There is a small variation



(a) $(\text{truth} - \text{mean})^2$ - time step 600 (b) variance - time step 600

(c) $(\text{truth} - \text{mean})^2$ - time step 1150 (d) variance - time step 1150

Fig. 6.6 *Comparison of the squared error* $(mean - truth)^2$ *with the variance in the ensemble at time 600 and at time 1150 for the EWPF. Note that the structures in error and variance are similar and that the size is also similar. See text for a more detailed discussion.*

due to the added random forcing after the step that makes the weights equal, but apart from that 25 particles (80 %) have very similar weights, showing that the filter is not degenerate for this high-dimensional example. This is, of course, due to the construction of the filter. Note that 7 particles weights have a much lower weight; they come back via resampling.

6.4.3 Comparison of nonlinear data assimilation methods

In this section we compare the standard particle filter (SIR), Equivalent-Weights Particle Filter (EWPF) and the Implicit Particle Filter (IPF) with the following Metropolis-Hastings based methods: the random walk Metropolis Hastings (MH), the preconditioned Crank-Nicholson Metropolis-Hastings (MHpCN), and Metropolis-adjusted Langevin (MALA). We use a simple model given by:

$$x^1 = x^0 + \eta \tag{6.96}$$

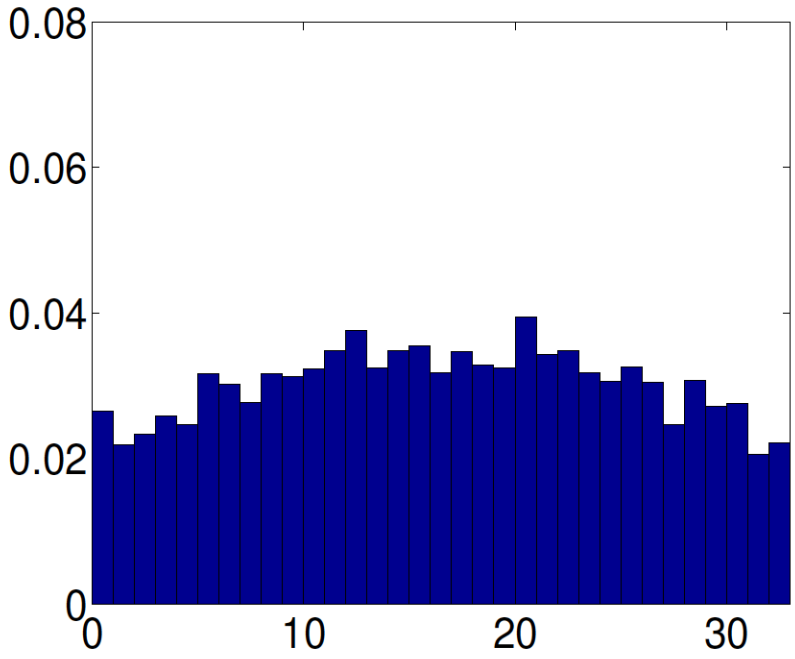


Fig. 6.7 *Rank histogram of the ranking of the truth in the ensemble, averaged over several grid points and over observation times. The non-flat shape illustrates that the ensemble is slightly overdispersive.*

The distribution of x^0 is taken as independent Gaussian for simplicity. The state vector is N_x-dimensional and we will investigate the performance of the filters when N_x increases. Finally, the model errors are also independent Gaussian. The state x^1 is observed as

$$y = x^1 + \varepsilon \tag{6.97}$$

in which the observation errors are also taken independent and Gaussian.

While this system is completely Gaussian, so linear, so the traditional linear data-assimilation methods can be used, we will explore several nonlinear data-assimilation methods here. The idea is that if the methods fail in this simple linear case it is unlikely they will perform better in a nonlinear setting.

Several experiments were performed with different values for initial, model, and observational errors. Here we report on the following experimental settings, mentioning that other settings give similar results: the initial mean is 0 for each variable, the initial variance is 1 for each variable, the model variance is 0.01, and the observation error variance is 0.16. The number of ensemble members or particles or MCMC samples is 10 in all experiments. For the equivalent-weights we keep 80% of the particles. For the Metropolis-Hastings sampler we set the width of the Gaussian proposal such that the acceptance rate is $30-50\%$. This let to the lowest

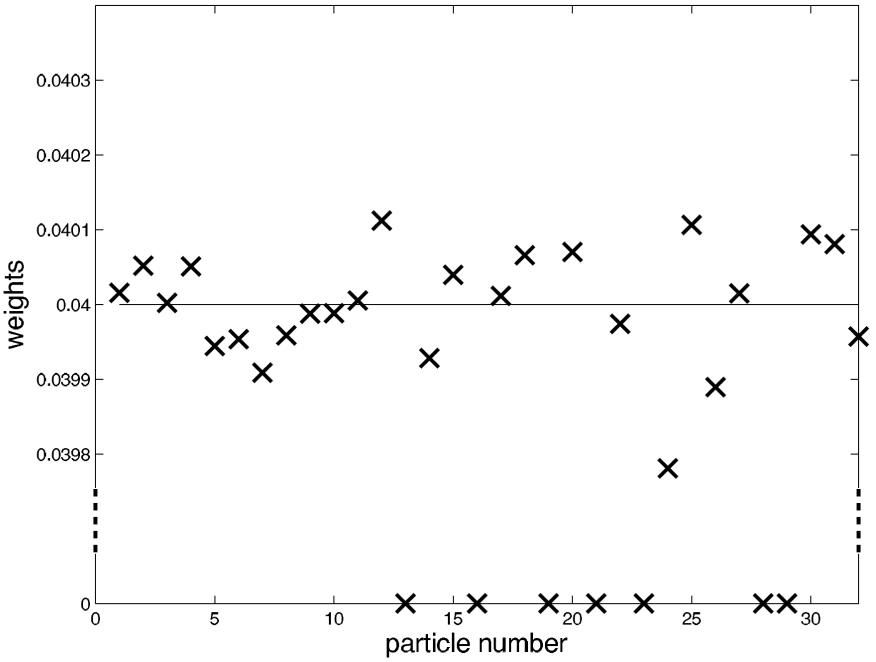


Fig. 6.8 *Weights of the particles before resampling. Note that the filter is not degenerate by construction as 25 particles have a good weight. The other 7 particles come back into the ensemble via resampling.*

root-mean-square error (RMSE). For Metropolis-Hastings with the preconditioned Crank-Nicholson proposal we scale the factor γ such that the acceptance rate is $30-50\%$. This let to the lowest RMSE. Finally, for Metropolis-adjusted Langevin we set the step size δ such that the acceptance rate was $30-50\%$, again leading to best performance.

Figure 6.5 shows the effective ensemble size in the three particle filters tested, defined as

$$N_{eft} = \frac{1}{\sum w_i^2} \tag{6.98}$$

for the standard particle filter with proposal density equal to the prior (SIR), the Equivalent-Weights Particle Filter (EWPF) and the Implicit Particle Filter (IPF), which is equal to a particle filter using the so-called Optimal Proposal density. The results shown are averages over 100 experiments. We can clearly see that apart from a state dimension of 1, all filters are degenerate, except for the EWPF, in which we find constant effective ensemble sizes of 8 out of 10, identical to the percentage of particles kept in the equivalent-weights procedure.

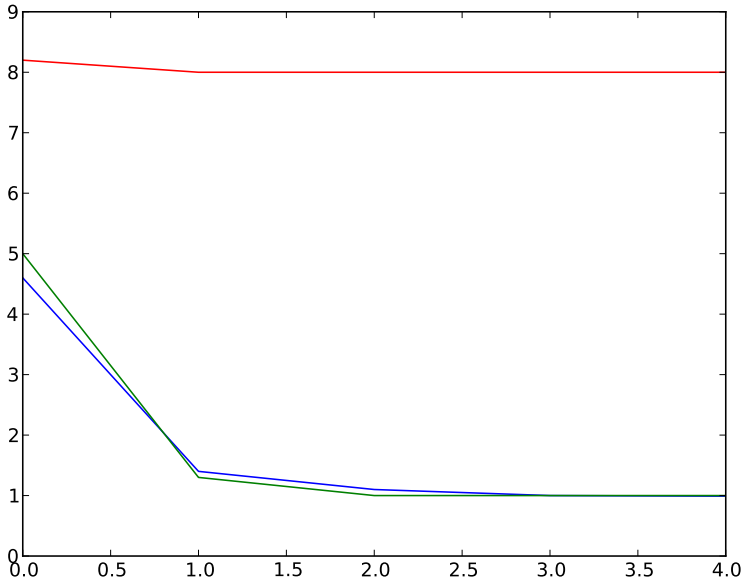


Fig. 6.9 *Effective ensemble size for SIR (blue), EWPF (red), and IPF (green) as function of the power of the state dimension, so* 10^0 *to* 10^4.

Figure 6.6 shows the root-mean-square error (RMSE) of the ensemble mean from the truth for each method, averaged over state space. The results shown are again averages over 100 experiments.

The theoretical posterior variance per dimension is 0.37 for this case, so a good filter would have a RMSE close to that. For dimension 1 we observe that most methods do reach that value, apart from MH, which is related to the small sample size. However, when the state dimension increases to 10 or higher only the EWPF keeps a proper RMSE, all other methods get a way too high value, close to the prior estimate. Of note is the MHpCN which settles around a value of 0.6. It is unclear why this method outperforms the others in this way (apart from the EWPF).

One might wonder why the IPF does not do much better than the standard particle filter. The reason is simply that the members are moved to better positions but the filter is degenerate, so the mean only consist of the best particle, which does depend on the initial particle position, so the RMSE will converge to that of the prior. Note that, for this specific case in which the prior at time zero is a Gaussian one could take the drawing from that Gaussian into the sampling via the proposal q, in which case all samples would have equal weight as this system is linear. In that case the IPF reduces to the Ensemble Smoother of Van Leeuwen and Evensen (1996, or the Ensemble Kalman Smoother of Evensen and Van Leeuwen, 2000). The point here is

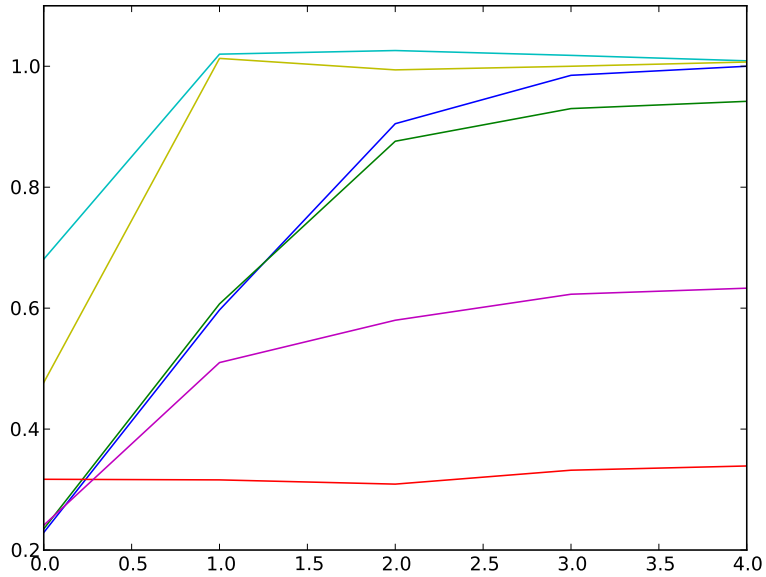


Fig. 6.10 *State-space averaged root-mean-square error of ensemble mean for SIR (blue), EWPF (red), IPF (green), MH (light blue), MHpCN (purple), MALA (light green) as function of the power of the state dimension, so* 10^0 *to* 10^4.

that, in general, the prior at time zero is non-Gaussian as it arises from a sequential application of the algorithm so the starting point at time zero is a number of particles with unknown distribution.

The results here do depend on the specific choice of the system parameters, i.e. the model error covariance, the observation covariance, and the covariance of the initial ensemble. For instance, the IPF has a much lower RMSE (0.45 for $N_x = 10$) when the model variance is multiplied by a factor 50. However, the effective ensemble size remains below 2, even for a 10-dimensional system, showing that the filter cannot represent the pdf with a low number of particles, which is the goal in geophysical data assimilation. These findings are consistent with our order of magnitude estimates on the optimal proposal density earlier. The point is, however, that even when varying these system parameters the general conclusion remains that only the EWPF is able to keep the effective number of particles high and the RMSE low.

Chapter 7
Conclusions

The nonlinear data-assimilation problem for high-dimensional systems is a hard one and this paper tries to summarise what has been done, providing new links and a few new ideas. It was shown that data-assimilation is not an inverse but a multiplication problem, and the goal of data assimilation is to try to represent the posterior pdf as efficiently as possible. That does not mean, however, that inverse methods are not important for the data-assimilation problem. The vast literature on inverse methods has resulted a huge variety of methods that can be, and are, explored in the more advanced Markov-Chain Monte-Carlo methods and in Particle Filters. Examples discussed here include so-called variational methods, but there is much more to explore, like iterative Tikhonov regularisation. Of specific mention is are variants that can handle non-smooth problems, see e.g. Steward et al, (2012) for a geophysical example. All these methods can be part of the proposal densities in the Bayesian framework, and this seems to be the natural way to combine the Bayesian and the inverse problems fields, and indeed their communities. In nonlinear systems the pdf is typically not of a shape that can be described by a few parameters, so one typically resorts to representing the posterior pdf by a set of samples from that pdf. This then automatically leads to Monte-Carlo methods to solve the nonlinear data-assimilation problem.

Although there is a vast variety of nonlinear data-assimilation methods available only a few are actually applicable to high-dimensional problems. The problems in mind are of geophysical origin, in which the state dimension can easily exceed one million. We discussed the applicability of Markov-Chain based methods like the Gibbs Sampler, Metropolis-Hastings, Hybrid Monte-Carlo and (Metropolis-adjusted) Langevin Monte-Carlo. All these methods rely on the notion of a Markov Chain that is used to generate the next sample from the previous sample. This immediately points to a problem with these methods: the next sample is typically not independent of the previous sample, so several subsequent samples have to be discarded before a true new independent sample from the posterior is found. This makes these methods less efficient from the start.

Particle filters are among the few that do not have this problem. They have a few strong assets, i.e. their full nonlinearity, the simplicity of their implementation (al-

though 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 the 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 like the Implicit Particle Filter and the Equivalent-Weights Particle Filter. Interestingly, by exploring proposal densities we are effectively using some sort of localisation as each grid point typically only sees observations within an influence region set by the covariance matrix used in the proposal. This does reduce the degeneracy problems considerably. Direct localisation can also now be explored by using ideas from optimal transportation. Strong new developments are expected when exploring these ideas further.

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. Especially Gaussian mixture models for the prior are popular. I have refrained from trying to give an overview here, there is just too much going on in this area. A brief discussion is given in Van Leeuwen (2009), but that is largely out of date now. In a few years time we will have learned what is useful and what is not.

Finally, it must be said that the methods discussed above 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 what about 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.

Chapter 8
Acknowledgements

I thank the members Data Assimilation Research Centre (DARC) for numerous discussions on the topics of this paper, especially Melanie Ades, Javier Amezcua, David Livings, Phil Browne, and Sanita Carvalho-Vetra. None of them is in anyway responsible for the contents. I also thank funding from the National Environment Research Council (NERC) via the National Centre of Earth Observation (NCEO) and via several other grants, and the EU FP7 project SANGOMA.

References

Ades, M., and P.J. Van Leeuwen, An exploration of the Equivalent-Weights Particle Filter , Q. J. Royal Meteorol. Soc., doi:10.1002/qj.1995, 2012.

Ades, M., and P.J. van Leeuwen, The equivalent-weights Particle Filter in a high-dimensional system, Monthly Weather Rev., accepted, 2014. Chorin A.J., X. Tu Implicit sampling for particle filters. *PNAS* 106, 17249-17254. 2009.

Atkins,E., M. Morzfeld, X. Tu, and A.J. Chorin, Implicit particle methods and their connection with variational data assimilation, Monthly Weather Review 141, 17861803, 2013.

Beskos, A., G. Roberts, A. Stuart, and J. Vos, MCMC methods for diffusion bridges, Stoch. Dyn. 8, 319-350, 2008.

Beskos, A., Pinski, F.J., Sanz-Serna, J.M. and Stuart, A.M.. Hybrid Monte Carlo on Hilbert spaces. Stochastic Process. Appl. 121 22012230. MR2822774, 2011.

Beskos, A., N. Pillai, G. Roberts, J.-M. Sanz-Serna and A.M. Stuart, Optimal tuning of the hybrid Monte Carlo algorithm. Bernoulli 19, 1501-1534, 2013.

Beyou, S., A. Cuzol, S. Subrahmanyam Gorthi, and E. Memin. Weighted ensemble transform Kalman filter for image assimilation. Tellus A, 65, 2013.

Cotter, S.L., G.O. Roberts, A.M. Stuart, and D. White, MCMC methods for functions: Modifying old algorithms to make them faster, Stat. Science, 28, 424-446, DOI:10.1214/13-STS421, 2013.

Doucet, A., N. De Freitas, and N. Gordon, *Sequential Monte-Carlo methods in practice*, Springer, Berlin, 2001.

Duane, S., A.D. Kennedy, B. Pendleton, and D. Roweth, Hybrid Monte-Carlo, Phys. Lett. B, 195, 216-222, 1987. Evensen, G. Data Assimilation: The Ensemble Kalman Filter, Springer, 2009.

Fisher, M., and H. Auvinen, Long window 4DVar, Proceedings of the ECMWF Seminar on Data Assimilation for Atmosphere and Ocean, 6-9 September 2011, 2012.

Gordon,N.J., D.J. Salmond, and A.F.M. Smith., "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", *IEE Proceedings-F*, 140, 107-113, 1993.

Gu Y, Oliver DS, An iterative ensemble Kalman filter for multiphase fluid flow data assimilation. SPE J 12(4):438446, 2007.

Huber, G., 1982: Gamma function derivation of n-sphere volumes. American Mathematical Monthly, 89, 301302.
Jazwinski, A.H. Stochastic Filtering and Filtering Theory, Academic Press, 1970.
Kim, S., G.L. Eyink, J.M. Restrepo, F.J. Alexander, and G. Johnson, 2003: Ensemble filtering for nonlinear dynamics, *Monthly Weather Rev.*, **131**, 2586-2594.

Le Gland, F., V. Monbet and V-D. Tran, Large sample asymptotics for the ensemble Kalman filter, in Handbook on Nonlinear Filtering, D. Crisan and B. Rozovskii, editors, pp. 598-631, Oxford University Press, Oxford, 2011.
Morzfeld, M., X. Tu, E. Atkins, and A.J. Chorin, A random map implementation of implicit filters, Journal of Computational Physics 231, 20492066, 2012.
Papadakis N., E. Memin, A. Cuzol, and N. Gengembre. Data assimilation with the weighted ensemble Kalman filter. Tellus, 62A:673 697, 2010.
Pitt, M.K., and N. Shephard, 1999: Filtering via simulation: Auxilary particle filters, *J. American Stat. Ass.*, **94**, 590-599.
S. Reich, A non-parametric ensemble transform method for Bayesian inference. SIAM J Sci Comput, 35 , A2013-A2014, 2013.
Robert, Ch.P., and G. Casella, Monte-Carlo Statistical Methods, Springer, 2004.
Snyder,C., T. Bengtsson, P. Bickel, and J. Anderson "Obstacles to high-dimensional particle filtering", *Mon. Wea. Rev.*, 136, 4629-4640, 2008.
Steward, J.L., I.M.Navon,M. Zupanski and N. Karmitsa. Impact of Non-Smooth Observation Operators on Variational and Sequential Data Assimilation for a Limited-Area Shallow Water Equations Model. Quart. Jour. Roy. Met Soc. , 138, 323–339, 2012.
Van Leeuwen, P.J., 2002: Ensemble Kalman Filters, Sequential Importance Resampling and beyond, *ECMWF workshop on the role of the upper ocean in medium and extended range forecasting, 13-15 November 2002*., ECMWF, Reading, UK, 46-56.
Van Leeuwen, P.J. Particle Filtering in Geosciences, *Monthly Weather Rev.*, 137, 4089-4114, 2009.
Van Leeuwen, P.J. , Nonlinear Data Assimilation in geosciences: an extremely efficient particle filter, *Quart. J. Roy. Meteor. Soc.*., 136, 1991-1996, 2010.
Van Leeuwen, P.J: Efficient non-linear Data Assimilation in Geophysical Fluid Dynamics *Computers and Fluids*, doi:10,1016/j.compfluid.2010.11.011, 2011.
Zupanski, M., Maximum Likelihood Ensemble Filter: Theoretical aspects, Mon. Weather Rev., 133, 1710 1726, 2005.