Data Assimilation

Kody J.H. Law, Andrew M. Stuart and Kostas Zygalakis
KAUST, Warwick University and Southampton University
kodylaw@gmail.com, a.m.stuart@warwick.ac.uk, k.zygalakis@soton.ac.uk

Abstract: These lecture notes provide an introduction to the subject of data assimilation, based on an underlying formulation as a Bayesian inverse problem, and from an applied mathematics perspective. Various standard methods are derived, analyzed and discussed from this Bayesian standpoint. Furthermore, continuous time formulations of the data assimilation problem are obtained, by taking a high observation frequency limit of the discrete time formulation. Study of these continuous time formulations provide insight into the mechanisms at play when dynamical models are combined with data; the continuous time limits also provide explicit algorithms appropriate for high frequency data. Explicit calculations, numerical examples and MATLAB code are provided throughout the notes in order to illustrate the theory.

Contents

1. Discrete Time: Formulation

In this Chapter we introduce the mathematical framework for discrete-time data assimilation. Section 1.1 describes the mathematical models we use for the underlying signal, which we wish to recover, and for the data, which we use for the recovery. In section 1.2 we introduce a number of examples used throughout the text to illustrate the theory. Sections 1.3 and 1.4 respectively describe two key problems related to the conditioning of the signal v on the data y, namely **smoothing** and **filtering**; in section 1.5 we describe how these two key problems are related. Section 1.6 proves that the smoothing problem is well-posed and, using the connection to filtering described in 1.5, that the filtering problem is well-posed; here well-posedness refers to continuity of the desired conditioned probability distribution with respect to the observed data. Section 1.7 discusses approaches to evaluating the quality of data assimilation algorithms. In section 1.8 we describe various illustrations of the foregoing theory and conclude the Chapter with section 1.9 devoted to a bibliographyical overview.

Throughout $\left(\langle\cdot,\cdot\rangle, |\cdot|\right)$ denotes the Euclidean inner-product and norm on $\mathbb{R}^\ell$, for any integer ℓ. For any positive-definite symmetric $A \in \mathbb{R}^{\ell\times\ell}$, we introduce the inner-product $\langle\cdot,\cdot\rangle_A = \langle A^{-\frac{1}{2}}\cdot, A^{-\frac{1}{2}}\cdot\rangle$ and the resulting norm $|\cdot|_A = |A^{-\frac{1}{2}}\cdot|$. The symbol $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the natural numbers and $\mathbb{Z}^+ = \{1, 2, \dots\}$ the positive integers. We use > 0 (resp. ≥ 0) to denote positive definite (resp. positive semi-definite) for real symmetric matrices.

1.1. Set-Up

Let $\Psi : \mathbb{R}^n \to \mathbb{R}^n$ and consider the Markov chain $v = \{v_j\}_{j\in\mathbb{N}}$ defined by the random map

$$v_{j+1} = \Psi(v_j) + \xi_j,\ j \in \mathbb{N}, \tag{1.1a}$$

$$v_0 \sim N(m_0, C_0), \tag{1.1b}$$

where $\xi = \{\xi_j\}_{j\in\mathbb{N}}$ is an i.i.d. sequence, independent of v_0, with $\xi_0 \sim N(0, \Sigma)$. Because (v_0, ξ) is a random variable, so too is the solution sequence $\{v_j\}_{j\in\mathbb{N}}$: the **signal**, which determines the state of the system at each discrete time instance. For simplicity we assume that u and ξ are independent. The probability distribution of the random variable v quantifies the uncertainty in predictions arising from this **stochastic dynamics** model.

In many applications, models such as (1.1) are supplemented by observations of the system as it evolves; this information then changes the probability distribution on the signal, typically reducing the uncertainty. To describe such situations we assume that we are given **data** or **observations** $y = \{y_j\}_{j\in\mathbb{Z}^+}$ defined as follows. At each discrete time instance we observe a (possibly nonlinear) function of the signal, with additive noise:

$$y_{j+1} = h(v_{j+1}) + \eta_{j+1},\ j \in \mathbb{N}, \tag{1.2}$$

where $h : \mathbb{R}^n \to \mathbb{R}^m$ and $\eta = \{\eta_j\}_{j\in\mathbb{Z}^+}$ is an i.i.d. sequence, independent of (v_0, ξ), with $\eta_1 \sim N(0, \Gamma)$. The function h is known as the **observation operator**. The objective of **data assimilation** is to determine information about the signal v, given data y. Mathematically we wish to solve the problem of conditioning the random variable v on the observed data y, or problems closely related to this.

We will also be interested in the case where the dynamics is deterministic and (1.1) becomes

$$v_{j+1} = \Psi(v_j),\ j \in \mathbb{N}, \tag{1.3a}$$

$$v_0 \sim N(m_0, C_0). \tag{1.3b}$$

In this case, which we refer to as **deterministic dynamics**, we are interested in the random variable v_0, given the observed data y; note that v_0 determines all subsequent values of the signal v.

Finally we mention that in many applications the function Ψ is the solution operator for an ordinary differential equation (ODE) of the form

$$\frac{dv}{dt} = f(v),\ t \in (0, \infty), \tag{1.4a}$$

$$v(0) = v_0. \tag{1.4b}$$

Then, assuming the solution exists for all $t \geq 0$, there is a one-parameter semi-group of operators $\Psi(\cdot;t)$, parameterized by time $t \geq 0$, with the properties that

$$\begin{aligned} v(t) &= \Psi(v_0;t),\ t \in (0,\infty), && (1.5a)\\ \Psi(u;t+s) &= \Psi\big(\Psi(u;s);t\big),\ t,s \in \mathbb{R}^+, u \in \mathbb{R}^n && (1.5b)\\ \Psi(u;0) &= u \in \mathbb{R}^n. && (1.5c) \end{aligned}$$

In this situation we assume that $\Psi(u) = \Psi(u;\tau)$, i.e. the solution operator over τ time units, where τ is the time between observations; thus we implicitly make the simplifying assumption that the observations are made at equally spaced time-points, and denote the state by $v_j = \Psi(v_{j-1};\tau)$. We use the notation $\Psi^{(j)}(\cdot)$ to denote the $j-$fold composition of Ψ with itself. Thus, in the case of continuous time dynamics, $\Psi(\cdot;j\tau) = \Psi^{(j)}(\cdot)$.

Throughout these notes we will make frequent reference to the concept of *ergodicity*. This refers to the idea that time averages of quantities evaluated along the trajectory of a dynamical system converge to an ergodic average which is independent of the initial condition. In the simplest setting, where the ergodic average is with respect to an *invariant measure* μ_∞ with Lebesgue density ρ_∞ we obtain, for μ_∞ almost every v_0 and some class of test functions $\phi : \mathbb{R}^n \to \mathbb{R}$,

$$\frac{1}{J}\sum_{j=1}^{J} \phi(v_j) \to \int_{\mathbb{R}^n} \rho_\infty(v)\phi(v)dv; \tag{1.6}$$

more generally we obtain

$$\frac{1}{J}\sum_{j=1}^{J} \phi(v_j) \to \int_{\mathbb{R}^n} \phi(v)\mu_\infty(dv). \tag{1.7}$$

Probability measures and their corresponding Lebesgue densities, or probability density functions (pdfs) will be discussed in more detail in section 1.3.

1.2. Guiding Examples

Throughout these notes we will use the following examples to illustrate the theory and algorithms presented.

Example 1.1. *We consider the case of one dimensional linear dynamics where*

$$\Psi(v) = \lambda v \tag{1.8}$$

for some scalar $\lambda \in \mathbb{R}$. Figure 1 compares the behaviour of the stochastic dynamics (1.1) and deterministic dynamics (1.1) for the two values $\lambda = 0.5$ and $\lambda = 1.05$. We set $\Sigma = \sigma^2$ and in both cases 50 iterations of the map are shown.

We observe that the presence of noise does not significantly alter the dynamics of the system for the case when $|\lambda| > 1$, since for both the stochastic and deterministic models $|v_j|$ is as $j \to \infty$. However, the effects of stochasticity are more pronounced when $|\lambda| < 1$, since in this case the deterministic map satisfies $v_j \to 0$ whilst, for the stochastic model, v_j fluctuates randomly around 0.

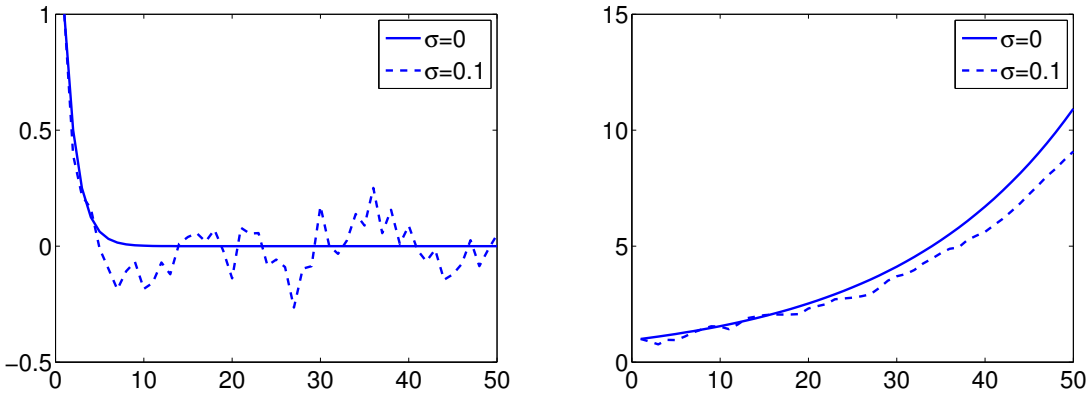
Using (1.1a), together with the linearity of Ψ and the Gaussianity of the noise ξ_j, we obtain

$$\mathbb{E}(v_{j+1}) = \lambda\mathbb{E}(v_j), \quad \mathbb{E}(v_{j+1}^2) = \lambda^2\mathbb{E}(v_j^2) + \sigma^2.$$

If $|\lambda| > 1$ then the modulus of the mean and the variance both explode as $j \to \infty$. On the other hand, if $|\lambda| < 1$, we see that $\mathbb{E}(v_j) \to 0$ and $\mathbb{E}(v_j^2) \to \sigma_\infty^2$ where

$$\sigma_\infty^2 = \frac{\sigma^2}{1-\lambda^2}. \tag{1.9}$$

Indeed, since v_0 is Gaussian, the model (1.1a) with linear Ψ and Gaussian noise ξ_j gives rise to a random variable v_j which is also Gaussian. Thus, from the convergence of the mean and variance of v_j, we conclude



(a) $\lambda = 0.5$ (b) $\lambda = 1.05$

FIG 1. *Behaviour of (1.1) for Ψ given by (1.8) for different values of λ and $\Sigma = \sigma^2$.*

that v_j converges weakly to the random variable $N(0, \sigma_\infty^2)$. This is an example of ergodicity as expressed in (1.6), (1.7); the measure μ_∞ is the Gaussian $N(0, \sigma_\infty^2)$ and the density ρ_∞ is the Lebesgue density of this Gaussian.

Example 1.2. *Now consider the case of two dimensional linear dynamics. In this case*

$$\Psi(v) = Av, \tag{1.10}$$

with A a 2×2 dimensional matrix of one of the following three forms:

$$A_1 = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} \lambda & \alpha \\ 0 & \lambda \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

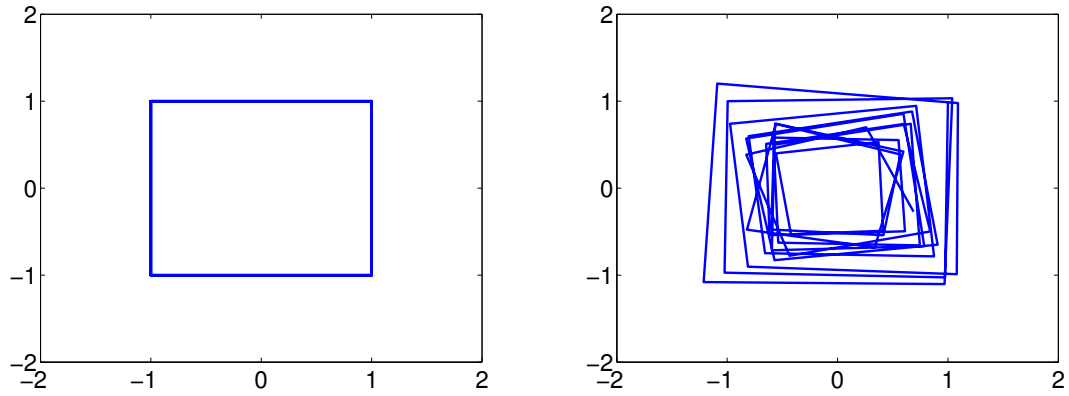
For $\ell = 1, 2$ the behaviour of (1.1) for $\Psi(u) = A_\ell u$ can be understood from the analysis underling the previous Example 1.1 and the behaviour is similar, in each coordinate, depending on whether the λ value on the diagonal is smaller than, or larger than, 1. However, the picture is more interesting when we consider the third choice $\Psi(u) = A_3 u$ as, in this case, the matrix A_3 has purely imaginary eigenvalues and corresponds to a rotation by $\pi/2$ on the plane; this is illustrated in Figure 2a. Addition of noise into the dynamics gives a qualitatively different picture: now the step j to $j+1$ corresponds to a rotation by $\pi/2$, composed with a random shift of origin; this is illustrated in Figure 2b.

Example 1.3. *We now consider our first nonlinear example, namely the one-dimensional dynamics for which*

$$\Psi(v) = \alpha \sin v. \tag{1.11}$$

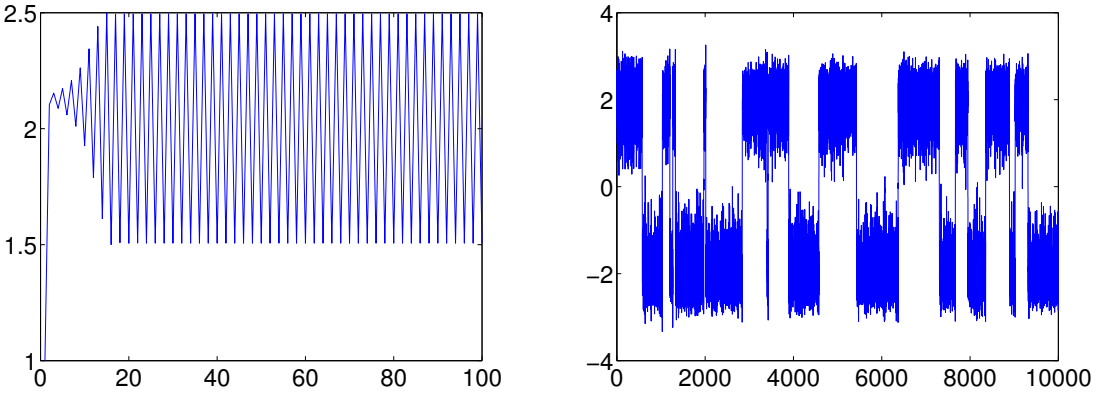
Figure 3 illustrates the behaviour of (1.1) for this choice of Ψ both for deterministic and stochastic dynamics. The behaviour of the system is significantly affected by noise. In the case of deterministic dynamics, Figure 3a, we see that eventually iterates of the discrete map converge to a period 2 solution. Although only one period 2 solution is seen in this single trajectory, we can deduce that there will be another period 2 solution, related to this one by the symmetry $u \mapsto -u$. This second solution is manifest when we consider stochastic dynamics. Figure 3b demonstrates that the inclusion of noise significantly changes the behaviour of the system. The signal now exhibits bistable behaviour and, within each mode of the behavioural dynamics, vestiges of the period 2 dynamics may be seen: the upper mode of the dynamics is related to the period 2 solution shown in Figure 3a and the lower mode to the period 2 solution found from applying the symmetry $u \mapsto -u$.

A good way of visualizing ergodicity is via the empirical measure, *or* histogram, *generated by a trajectory of the dynamical system. Equation (1.6) formalizes the idea that the histogram, in the large J limit, converges to the probability density function of a random variable, independently of the starting point v_0. Thinking in*



(a) Deterministic dynamics (b) Stochastic dynamics, $\sigma = 0.1$

FIG 2. *Behaviour of (1.1) for* Ψ *given by* (1.10)*, and* $\Sigma = \sigma^2$.



(a) Deterministic dynamics (b) Stochastic dynamics, $\sigma = 0.25$

FIG 3. *Behaviour of (1.1) for* Ψ *given by* (1.11) *for* $\alpha = 2.5$ *and* $\Sigma = \sigma^2$*, see also* `p1.m` *in Section 4.1.1.*

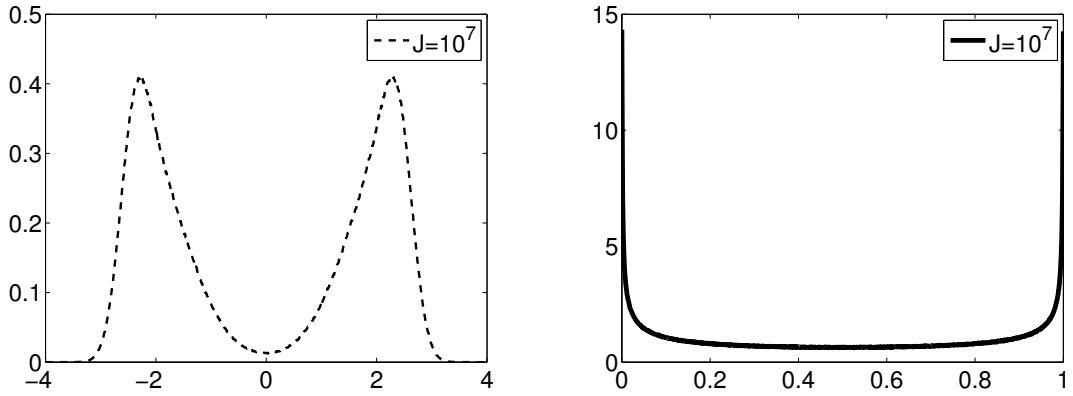
terms of pdfs of the signal, or functions of the signal, and neglecting time-ordering information, is a very useful viewpoint throughout these notes.

Histograms visualize complex dynamical behaviour such as that seen in Figure 3b by ignoring time-correlation in the signal and simply keeping track of where *the solution goes in time, but not the* order *in which places are visited. This is illustrated in Figure 4a, where we plot the histogram corresponding to the dynamics shown in Figure 3b, but calculated using a simulation of length* $J = 10^7$*. We observe that the system quickly forgets its initial condition and spends an almost equal proportion of time around the positive and negative period* 2 *solutions of the underlying deterministic map. The Figure 4a would change very little if the system were started from a different initial condition, reflecting ergodicity of the underlying map.*

Example 1.4. *We now consider a second one-dimensional and nonlinear map, for which*

$$\Psi(v) = rv(1 - v). \tag{1.12}$$

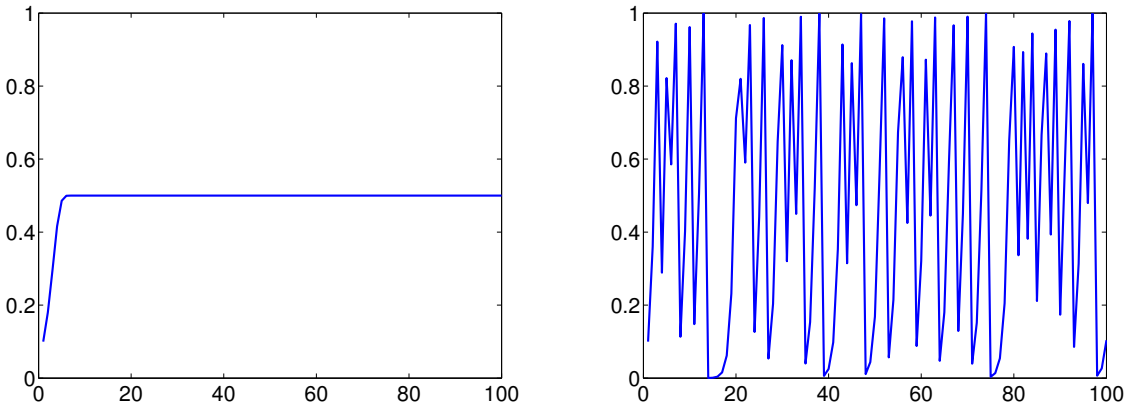
We consider initial data $v_0 \in [0, 1]$ *noting that, for* $r \in [0, 4]$*, the signal will then satisfy* $v_j \in [0, 1]$ *for all* j*, in the case of the deterministic dynamics* (1.3)*. We confine our discussion here to the deterministic case which can itself exhibit quite rich behaviour. In particular, the behaviour of (1.3, 1.12) can be seen in Figure 5 for*



(a) Example 1.3, as in Figure 3b

(b) Example 1.4, as in Figure 5b

FIG 4. *Probability density functions for* $v_j, j = 0, \cdots, J$, *for* $J = 10^7$



(a) Deterministic dynamics, $r = 2$

(b) Deterministic dynamics, $r = 4$

FIG 5. *Behaviour of (1.1) for* Ψ *given by* (1.12).

the values of $r = 2$ *and* $r = 4$. *These values of* r *have the desirable property that it is possible to determine the signal analytically. For* $r = 2$ *one obtains*

$$v_j = \frac{1}{2} - \frac{1}{2}(1 - 2v_0)^{2^j}, \tag{1.13}$$

which implies that, for any value of $v_0 \neq 0, 1$, $v_j \to 1/2$ *as we can also see in Figure 5a. For* $v_0 = 0$ *the solution remains at the unstable fixed point* 0, *whilst for* $v_0 = 1$ *the solution maps onto* 0 *in one step, and then remains there. In the case* $r = 4$ *the solution is given by*

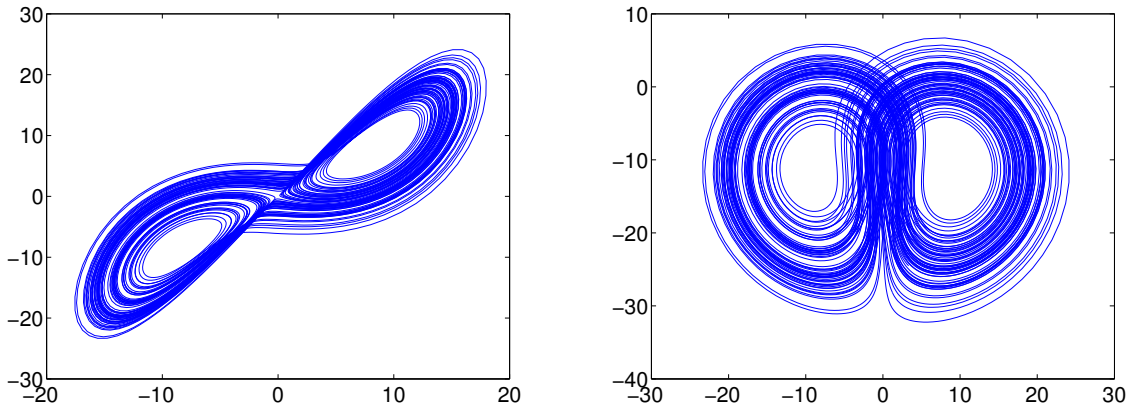
$$v_j = 4\sin^2(2^j \pi\theta), \quad \textit{with} \quad v_0 = 4\sin^2(\pi\theta) \tag{1.14}$$

This solution can also be expressed in the form

$$v_j = \sin^2(2\pi z_j). \tag{1.15}$$

where

$$z_{j+1} = \begin{cases} 2z_j, & 0 \leq z_j < \frac{1}{2}, \\ 2z_j - 1, & \frac{1}{2} \leq z_j < 1, \end{cases}$$



(a) u_1 vs u_2 (b) u_2 vs u_3

FIG 6. *Projection of the Lorenz'63 attractor onto two different pairs of coordinates.*

and using this formula it is possible to show that this map produces chaotic dynamics for almost all initial conditions. This is illustrated in Figure 5b, where we plot the first 100 *iterations of the map. In addition, in Figure 4b, we plot the pdf using a long trajectory of* v_j *of length* $J = 10^7$*, demonstrating the ergodicity of the map. In fact there is an analytic formula for the steady state value of the pdf for* $J \to \infty$ *and it is given by*

$$\rho(x) = \pi^{-1} x^{-1/2} (1-x)^{-1/2}. \tag{1.16}$$

Example 1.5. *Turning now to maps* Ψ *derived from differential equations, the simplest case is to consider linear autonomous dynamical systems of the form*

$$\frac{dv}{dt} = Lv, \tag{1.17a}$$
$$v(0) = v_0. \tag{1.17b}$$

Then $\Psi(u) = Au$ *with* $A = \exp(Lh)$.

Example 1.6. *The Lorenz '63 model is perhaps the simplest continuous-time system to exhibit sensitivity to initial conditions and chaos. It is a system of three coupled non-linear ordinary differential equations whose solution* $v \in \mathbb{R}^3$*, where* $v = (v_1, v_2, v_3)$*, satisfies*[1]

$$\frac{dv_1}{dt} = \alpha(v_2 - v_1), \tag{1.18a}$$
$$\frac{dv_2}{dt} = -\alpha v_1 - v_2 - v_1 v_3, \tag{1.18b}$$
$$\frac{dv_3}{dt} = v_1 v_2 - b v_3 - b(r + \alpha). \tag{1.18c}$$

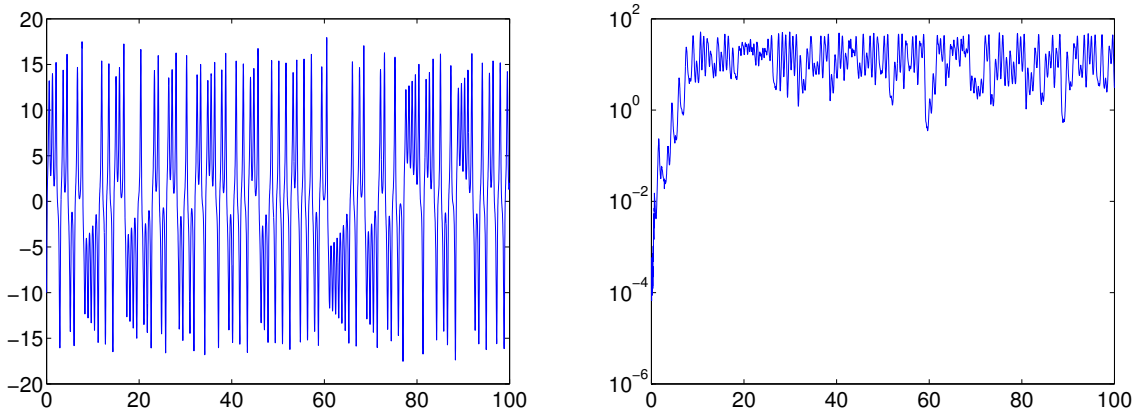
Note that we have employed a coordinate system where the origin in the original version of the equations proposed by Lorenz is shifted. In the coordinate system that we employ here we have equation (1.4) with vector field f *satisfying*

$$\langle f(v), v \rangle \leq a - b|v|^2 \tag{1.19}$$

for some $a, b > 0$*. This implies the existence of an absorbing set:*

$$\limsup_{t \to \infty} |v(t)|^2 < R \tag{1.20}$$

[1]Here index denotes components of the solution, not discrete time.



(a) u_1 as a function of time

(b) Evolution of error for a small perturbation

FIG 7. *Dynamics of the Lorenz'63 model in the chaotic regime* $(\alpha, b, r) = (10, \frac{8}{3}, 28)$

for any $R > a/b$. *Mapping the ball* $B(0; R)$ *forward under the dynamics gives the global attractor for the dynamics. In Figure 6 we visualize this attractor, projected onto two different pairs of coordinates at the classical parameter values* $(\alpha, b, r) = (10, \frac{8}{3}, 28)$.

Throughout these notes we will use the classical parameter values $(\alpha, b, r) = (10, \frac{8}{3}, 28)$ *in all of our numerical experiments; at these values the system is ergodic and exhibits sensitive dependence with respect to the initial condition. A trajectory of* u_1 *versus time can be found in Figure 7a and in Figure 7b we illustrate the evolution of a small perturbation to the initial condition which generated Figure 7a; to be explicit we plot the evolution of the error in the Euclidean norm* $|\cdot|$, *for an initial perturbation of magnitude* 10^{-4}. *Figure 6 suggests that the ergodic measure* μ_∞ *is supported on a strange set with Lebesgue measure zero, and this is indeed the case; for this example there is no Lebesgue density* ρ_∞ *for the invariant measure, reflecting the fact that the attractor has a fractal dimension less than three.*

Example 1.7. *The Lorenz '96 model is a simple dynamical system, of tunable dimension, which was designed as a caricature of the dynamics of Rossby waves in atmospheric dynamics. The equations have a periodic "ring" formulation and take the form*[2]

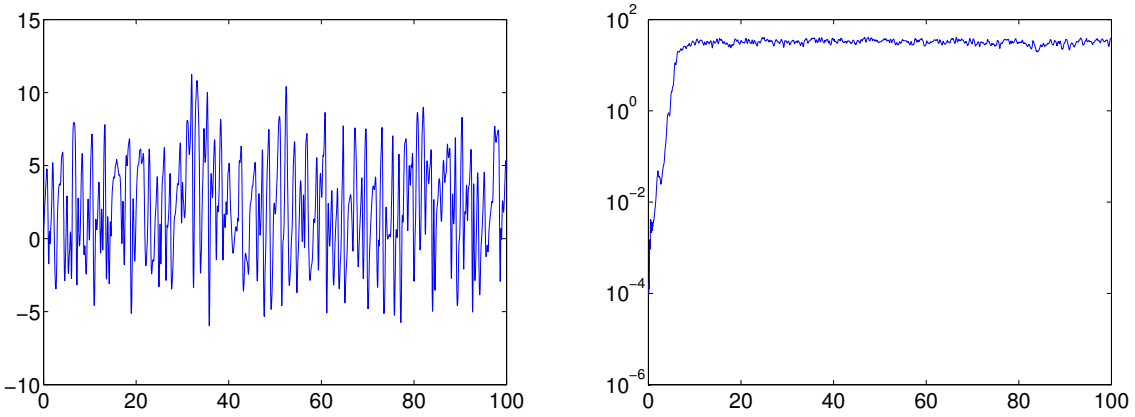
$$\frac{dv_k}{dt} = v_{k-1}\big(v_{k+1} - v_{k-2}\big) - v_k + F, \quad k \in \{1, \cdots, K\}, \tag{1.21a}$$

$$v_0 = v_K, \quad v_{K+1} = v_1, \quad v_{-1} = v_{K-1}. \tag{1.21b}$$

Equation (1.21) satisfies the same dissipativity property (1.19) satisfied by the Lorenz '63 model, for appropriate choice of $a, b > 0$, *and hence also satisfies the absorbing ball property (1.20) and thus has a global attractor.*

In Figure 8a we plot a trajectory of v_1 *versus time for* $F = 8$ *and* $K = 40$. *Furthermore, as we did in the case of the Lorenz '63 model, we also show the evolution of the Euclidean norm of the error* $|\cdot|$ *for an initial perturbation of magnitude* 10^{-4}*; this is displayed in Figure 8b and clearly demonstrates sensitive dependence on initial conditions. We visualize the attractor, projected onto two different pairs of coordinates, in Figure 9.*

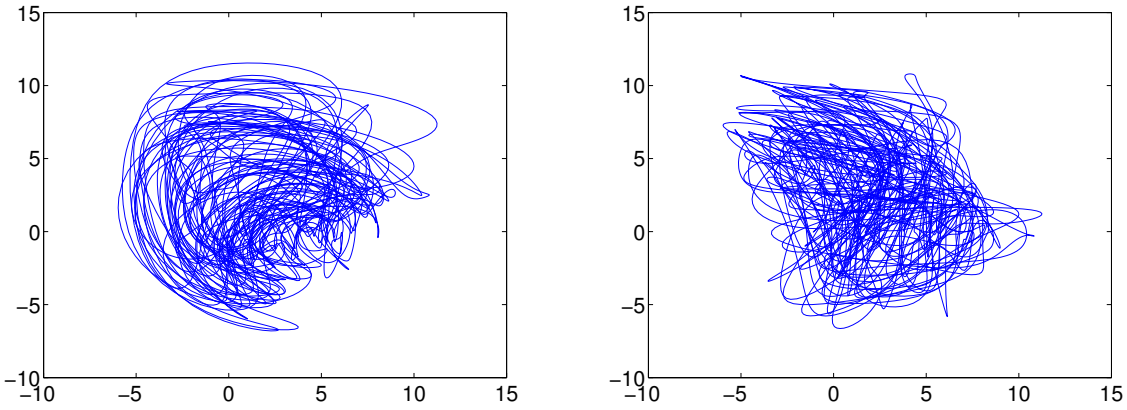
[2]Again, here index denotes components of the solution, not discrete time.



(a) v_1 as a function of time

(b) Evolution of error for a small initial perturbation

FIG 8. *Dynamics of the Lorenz'96 model in the chaotic regime* $(F, K) = (8, 40)$



(a) v_1 vs v_K

(b) v_1 vs v_{K-1}

FIG 9. *Projection of the Lorenz'96 attractor onto two different pairs of coordinates.*

1.3. Smoothing Problem

1.3.1. Probabilistic Formulation of Data Assimilation

Together (1.1) and (1.2) provide a probabilistic model for the jointly varying random variable (v, y). In the case of deterministic dynamics, (1.3) and (1.2) provide a probabilistic model for the jointly varying random variable (v_0, y). Thus in both cases we have a random variable (u, y), with $u = v$ (resp. $u = v_0$) in the stochastic (resp. deterministic) case. Our aim is to find out information about v, in the stochastic case, or v_0 in the deterministic case, from observation of a single instance of y. The natural probabilistic approach to this problem is to try and find the random variable u given y, denoted $u|y$. This constitutes the Bayesian formulation of the problem of determining information about the signal arising in a noisy dynamical model, based on noisy observations of that signal. We will refer to the conditioned random variable $u|y$, in the case of either the stochastic dynamics or deterministics dynamics, as the **smoothing distribution**. It is a random variable which contains all the probabilistic information about the signal, given our observations. The key formula which drives this approach is given by Bayes' Theorem which we now discuss in abstract form.

1.3.2. Bayesian Probability

Throughout these notes we will consider random variables on $\mathbb{R}^\ell$ for a range of different integers ℓ. In all cases we assume that the random variable has a positive Lebesgue density, everywhere on $\mathbb{R}^\ell$, as this will simplify the presentation. Given a random variable a we write $\mathbb{P}(a)$ to denote the Lebesgue density associated with this random variable; we call this the pdf (probability density function). For any function $f : \mathbb{R}^\ell \to \mathbb{R}^p$ we use $\mathbb{E}f(a)$ to denote the expected value of the random variable $f(a)$ on $\mathbb{R}^p$. Sometimes a will be thought of as varying with respect to a number of different probability measures, with different Lebesgue densities. In that situation, when we wish to denote the particular measure μ that we are using at a given point in our arguments, then we may write $\mathbb{P}^\mu$ and $\mathbb{E}^\mu$ to denote the corresponding pdf and expectation. If the pair (a, b) is a jointly varying random variable then $a|b$ denotes the random variable a given b, and we denote its pdf $\mathbb{P}(a|b)$. The pushforward of a pdf ρ on $\mathbb{R}^l$ under a map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^\ell$ is denoted $\Phi \star \rho$. It may be calculated explicitly by means of change of variable formula under an integral. Indeed if Φ is invertible then $\Phi \star \mathbb{P}(v_j|Y_j) := \mathbb{P}(\Phi^{-1}(v)|Y_j)|D\Phi^{-1}(v)|$.

If $(a, b) \in \mathbb{R}^\ell \times \mathbb{R}^m$ is a random variable then we denote the associated pdf by $\mathbb{P}(a, b)$, and we denote the pdfs of the conditional random variables $a|b$ and $b|a$ by $\mathbb{P}(a|b)$ and $\mathbb{P}(b|a)$ respectively. The rules of conditional probability show that

$$\mathbb{P}(a, b) = \mathbb{P}(a|b)\mathbb{P}(b), \tag{1.22a}$$

$$\mathbb{P}(a, b) = \mathbb{P}(b|a)\mathbb{P}(a). \tag{1.22b}$$

We will make frequent use of these formulae in the notes. We also note that integrating over the variable b in (1.22a) gives the identity

$$\mathbb{P}(a) = \int_{\mathbb{R}^m} \mathbb{P}(a, b)db \tag{1.23a}$$

$$\int_{\mathbb{R}^m} \mathbb{P}(a|b)\mathbb{P}(b)db. \tag{1.23b}$$

Bayes' formula for pdfs, which we will use frequently throughout the notes, states that

$$\mathbb{P}(a|b) = \frac{1}{\mathbb{P}(b)}\mathbb{P}(b|a)\mathbb{P}(a) \tag{1.24}$$

and it also follows from (1.23): the formula follows in a straightforward way from dividing one of the two formulae for the definitions of the conditioned random variables $a|b$ and $b|a$ by the other. When Bayes' formula (1.24) is used in statistics then typically b is the data and a the unknown about which we wish to find information, using the data. In this context we refer to $\mathbb{P}(a)$ as the **prior**, to $\mathbb{P}(b|a)$ as the **likelihood**

and to $\mathbb{P}(a|b)$ as the **posterior**. The beauty of Bayes's formula as a tool in applied mathematics is that the likelihood is often easy to determine explicitly, given reasonable assumptions on the observational noise, whilst there is considerable flexibility inherent in modelling prior knowledge via probabilities, to give the prior. Combining the prior and likelihood as in (1.24) gives the posterior, which is the random variable of interest; whilst the probability distributions used to define the likelihood $\mathbb{P}(b|a)$ (via a probability density on the data space) and prior $\mathbb{P}(a)$ (via a probability on the space of unknowns) may be quite simple, the resulting posterior probability distribution can be very complicated. A second key point to note about Bayes' formula in this context is that $\mathbb{P}(b)$, which normalizes the posterior to a pdf, may be hard to determine explicitly, but algorithms exist to find information from the posterior without knowing this normalization. We return to this point later.

1.3.3. Stochastic Dynamics

We wish to find the signal v from (1.1) from a single instance of data y given by (1.2). To be more precise we wish to condition the signal on a discrete time interval $\mathbb{J}_0 = \{0, ..., J\}$, given data on the discrete time interval $\mathbb{J} = \{1, ..., J\}$; we refer to $\mathbb{J}_0$ as the data assimilation window. We define $v = \{v_j\}_{j\in\mathbb{J}_0}, y = \{y_j\}_{j\in\mathbb{J}}, \xi = \{\xi_j\}_{j\in\mathbb{J}_0}$ and $\eta = \{\eta_j\}_{j\in\mathbb{J}_0}$. The smoothing distribution here is the distribution of the conditioned random variable $v|y$. Recall that we have assumed that u, ξ and η are mutually independent random variables. With this fact in hand we may apply Bayes' formula to find the pdf $\mathbb{P}(v|y)$.

Prior The prior on v is specified by (1.1), together with the independence of u and ξ and the i.i.d. structure of ξ. First note that, using (1.22) and the i.i.d. structure of ξ in turn, we obtain

$$\begin{aligned}\mathbb{P}(v) &= \mathbb{P}(v_J, v_{J-1}, \cdots, v_0)\\ &= \mathbb{P}(v_J|v_{J-1}, \cdots, v_0)\mathbb{P}(v_{J-1}, \cdots, v_0)\\ &= \mathbb{P}(v_J|v_{J-1})\mathbb{P}(v_{J-1}, \cdots, v_0).\end{aligned}$$

Proceeding inductively gives

$$\mathbb{P}(v) = \Pi_{j=0}^{J-1}\mathbb{P}(v_{j+1}|v_j)\mathbb{P}(v_0).$$

Now

$$\mathbb{P}(v_0) \propto \exp\Big(-\frac{1}{2}\big|C_0^{-\frac{1}{2}}(v_0 - m_0)\big|^2\Big)$$

whilst

$$\mathbb{P}(v_{j+1}|v_j) \propto \exp\Big(-\frac{1}{2}\big|\Sigma^{-\frac{1}{2}}\big(v_j - \Psi(v_{j-1})\big)\big|^2\Big).$$

Thus

$$\mathbb{P}(v) \propto \exp(-\mathsf{J}(v))$$

where

$$\mathsf{J}(v) := \frac{1}{2}\big|C_0^{-\frac{1}{2}}(v_0 - m_0)\big|^2 + \sum_{j=0}^{J-1}\frac{1}{2}\big|\Sigma^{-\frac{1}{2}}\big(v_{j+1} - \Psi(v_j)\big)\big|^2. \tag{1.25}$$

The pdf $\mathbb{P}(v) = \rho_0(v)$ proportional to $\exp(-\mathsf{J}(v))$ determines a prior measure μ_0 on $\mathbb{R}^{|\mathbb{J}_0|\times n}$.

Likelihood The likelihood of the data $y|v$ is determined as follows. It is a (Gaussian) probability distribution on $\mathbb{R}^{|\mathbb{J}|\times m}$, with pdf $\mathbb{P}(y|v)$ proportional to $\exp(-\Phi(v; y))$, where

$$\Phi(v; y) = \sum_{j=0}^{J-1}\frac{1}{2}\big|\Gamma^{-\frac{1}{2}}\big(y_{j+1} - h(v_{j+1})\big)\big|^2. \tag{1.26}$$

To see this note that, because of the i.i.d nature of the sequence η, it follows that

$$\begin{aligned}\mathbb{P}(y|v) &= \Pi_{j=0}^{J-1}\mathbb{P}(y_{j+1}|v)\\ &= \Pi_{j=0}^{J-1}\mathbb{P}(y_{j+1}|v_{j+1})\\ &\propto \Pi_{j=0}^{J-1}\exp\Big(-\frac{1}{2}\big|\Gamma^{-\frac{1}{2}}\big(y_{j+1} - h(v_{j+1})\big)\big|^2\Big)\\ &= \exp(-\Phi(v; y)).\end{aligned}$$

In the applied literture m_0 and C_0 are often referred to as the **background mean** and **background covariance** respectively; we refer to Φ as the **model-data misfit** functional.

Using Bayes' formula (1.24) we can combine the prior and the likelihood to determine the posterior distribution, that is the smoothing distribution, on $v|y$. We denote the measure with this distribution by μ.

Theorem 1.8. *The posterior smoothing distribution on $v|y$ for the stochastic dynamics model (1.1), (1.2) is a probability measure μ on $\mathbb{R}^{\mathbb{J}_0|\times n}$ with pdf $\mathbb{P}(v|y) = \rho(v)$ proportional to $\exp(-\mathsf{I}(v;y))$ where*

$$\mathsf{I}(v;y) = \mathsf{J}(v) + \Phi(v;y). \tag{1.27}$$

Proof. Bayes' formula (1.24) gives us

$$\mathbb{P}(v|y) = \frac{\mathbb{P}(y|v)\mathbb{P}(v)}{\mathbb{P}(y)}.$$

Thus, ignoring constants of proportionality which depend only on y,

$$\begin{aligned}\mathbb{P}(v|y) &\propto \mathbb{P}(y|u)\mathbb{P}(v_0)\\ &\propto \exp(-\Phi(v;y))\exp(-\mathsf{J}(v))\\ &= \exp(-\mathsf{I}(v;y)).\end{aligned}$$

□

Note that, although the preceding calculations required only knowledge of the pdfs of Gaussian distributions, the resulting posterior distribution is non-Gauassian in general, unless Ψ and h are linear. This is because, unless Ψ and h are linear, $\mathsf{I}(\cdot;y)$ is not quadratic.

1.3.4. Deterministic Dynamics

It is also of interest to study the posterior distribution on the initial condition in the case where the model dynamics contains no noise, and is given by (1.3); this we now do. Recall that $\Psi^{(j)}(\cdot)$ denotes the $j-$fold composition of $\Psi(\cdot)$ with itself. In the following we sometimes refer to J_0 as the **background** penalization, and m_0 and C_0 as the background mean and covariance; we refer to Φ_0 as the **model-data misfit** functional.

Theorem 1.9. *The posterior smoothing distribution on $v_0|y$ for the the deterministic dynamics model (1.3), (1.2) is a probability measure ν on $\mathbb{R}^n$ with density $\mathbb{P}(v_0|y) = \varrho(v_0)$ proportional to $\exp(-\mathsf{I}_0(v_0;y))$ where*

$$\mathsf{I}_0(v_0;y) = \mathsf{J}_0(v_0) + \Phi_0(v_0;y) \tag{1.28a}$$

$$\mathsf{J}_0(v_0) = \frac{1}{2}\left|C_0^{-\frac{1}{2}}(v_0 - m_0)\right|^2 \tag{1.28b}$$

$$\Phi_0(v_0;y) = \sum_{j=0}^{J-1}\frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\Big(y_{j+1} - h\big(\Psi^{(j+1)}(v_0)\big)\Big)\right|^2. \tag{1.28c}$$

Proof. We again use Bayes' Theorem which states that

$$\mathbb{P}(v_0|y) = \frac{\mathbb{P}(y|v_0)\mathbb{P}(v_0)}{\mathbb{P}(y)}.$$

Thus, ignoring constants of proportionality which depend only on y,

$$\begin{aligned}\mathbb{P}(v_0|y) &\propto \mathbb{P}(y|v_0)\mathbb{P}(v)\\ &\propto \exp\big(-\Phi_0(v_0;y)\big)\exp\big(-\frac{1}{2}|C_0^{-\frac{1}{2}}(v_0 - m_0)|^2\big)\\ &= \exp(-\mathsf{I}(v_0;y)).\end{aligned}$$

To see that $\mathbb{P}(y|v_0)$ is proportional to $\exp\big(-\Phi_0(v_0;y)\big)$ follows from the fact that $y_j|v_0$ form an i.i.d sequence of Gaussian random variables $N\big(h(v_j),\Gamma\big)$ with $v_j = \Psi^{(j)}(v_0)$. □

1.4. Filtering Problem

Note that the smoothing problem considered in the previous section involves, potentially, conditioning v_j on data y_k with $k > j$. Such conditioning can only be performed *off-line* and is of no use in *on-line* scenarios where we want to determine information on the state of the signal *now* hence using only data from the past up to the present. To study this situation, let $Y_j = \{y_l\}_{l=1}^j$ denote the accumulated data up to time j. **Filtering** is concerned with the sequential updating of $\mathbb{P}(v_j|Y_j)$, the pdf of $v_j|Y_j$, as the index j is incremented. This update is defined by the following two-step procedure which provides a prescription for computing $\mathbb{P}(v_{j+1}|Y_{j+1})$ from $\mathbb{P}(v_j|Y_j)$ via a two step-procedure: **prediction** which computes the mapping $\mathbb{P}(v_j|Y_j) \mapsto \mathbb{P}(v_{j+1}|Y_j)$ and **analysis** which computes $\mathbb{P}(v_{j+1}|Y_j) \mapsto \mathbb{P}(v_{j+1}|Y_{j+1})$ by application of Bayes' formula .

Prediction Note that $\mathbb{P}(v_{j+1}|Y_j, v_j) = \mathbb{P}(v_{j+1}|v_j)$ because Y_j contains noisy and indirect information about v_j and cannot improve upon perfect knowledge of the variable v_j. Thus, by (1.23), we deduce that

$$\mathbb{P}(v_{j+1}|Y_j) = \int_{\mathbb{R}^n} \mathbb{P}(v_{j+1}|Y_j, v_j)\mathbb{P}(v_j|Y_j)dv_j \tag{1.29a}$$

$$= \int_{\mathbb{R}^n} \mathbb{P}(v_{j+1}|v_j)\mathbb{P}(v_j|Y_j)dv_j \tag{1.29b}$$

Note that, since the forward model equation (1.1) determines $\mathbb{P}(v_{j+1}|v_j)$, this prediction step provides the map from $\mathbb{P}(v_j|Y_j)$ to $\mathbb{P}(v_{j+1}|Y_j)$. This prediction step simplifies in the case of deterministic dynamics (1.3); in this case it simply corresponds to computing the pushforward of $\mathbb{P}(v_j|Y_j)$ under the map Ψ.

Analysis Note that $\mathbb{P}(y_{j+1}|v_{j+1}, Y_j) = \mathbb{P}(y_{j+1}|v_{j+1})$ because because Y_j contains noisy and indirect information about v_{j+1} and cannot improve upon perfect knowledge of the variable v_{j+1}. Thus, using Bayes' formula (1.24), we deduce that

$$\begin{aligned}\mathbb{P}(v_{j+1}|Y_{j+1}) &= \mathbb{P}(v_{j+1}|Y_j, y_{j+1}) \\ &= \frac{\mathbb{P}(y_{j+1}|v_{j+1}, Y_j)\mathbb{P}(v_{j+1}|Y_j)}{\mathbb{P}(y_{j+1}|Y_j)} \\ &= \frac{\mathbb{P}(y_{j+1}|v_{j+1})\mathbb{P}(v_{j+1}|Y_j)}{\mathbb{P}(y_{j+1}|Y_j)}.\end{aligned} \tag{1.30}$$

Since the observation equation (1.2) determines $\mathbb{P}(y_{j+1}|v_{j+1})$, this analysis step provides a map from $\mathbb{P}(v_{j+1}|Y_j)$ to $\mathbb{P}(v_{j+1}|Y_{j+1})$.

Filtering Update Together, then, the prediction and analysis step provide a mapping from $\mathbb{P}(v_j|Y_j)$ to $\mathbb{P}(v_{j+1}|Y_{j+1})$. Indeed if we let μ_j denote the probability measure on $\mathbb{R}^n$ corresponding to the density $\mathbb{P}(v_j|Y_j)$ and $\widehat{\mu}_{j+1}$ be the probability measure on $\mathbb{R}^n$ corresponding to the density $\mathbb{P}(v_{j+1}|Y_j)$ then the prediction step maps μ_j to $\widehat{\mu}_{j+1}$ whilst the analysis step maps $\widehat{\mu}_{j+1}$ to μ_{j+1}. However, there is, in general, no easily usable closed form expression for the density of μ_j, namely $\mathbb{P}(v_j|Y_j)$. Nevertheless, formulae (1.29), (1.30) form the starting point for numerous algorithms to approximate $\mathbb{P}(v_j|Y_j)$.

1.5. Filtering and Smoothing are Related

The filtering and smoothing approaches to determining the signal from the data are distinct, but related. They are related by the fact that in both cases the solution computed at the *end* of any specified time-interval is conditioned on the same data; this is made precise in the following.

Theorem 1.10. *Let $\mathbb{P}(v|y)$ denote the smoothing distribution on the discrete time interval $j \in \mathbb{J}_0$, and $\mathbb{P}(v_J|Y_J)$ the filtering distribution at time $j = J$ for the stochastic dynamics model (1.1). Then the marginal of the smoothing distribution on v_J is the same as the filtering distribution at time J:*

$$\int \mathbb{P}(v|y)dv_0dv_1...dv_{J-1} = \mathbb{P}(v_J|Y_J).$$

Proof. Note that $y = Y_J$. Since $v = (v_0, ..., v_{J-1}, v_J)$ the result follows trivially. □

Remark 1.11. *Note that the marginal of the smoothing distribution on say v_j, $j < J$ is* not *equal to the filter $\mathbb{P}(v_j|Y_j)$. This is because the smoother induces a distribution on v_j which is influenced by the entire data stream $Y_J = y = \{y_l\}_{l\in\mathbb{J}}$; in contrast the filter at j involves only the data $Y_j = \{y_l\}_{l\in\{1,...,j\}}$.*

It is also interesting to mention the relationship between filtering and smoothing in the case of noise-free dynamics. In this case the filtering distribution $\mathbb{P}(v_j|Y_j)$ is simply found as the pushforward of the smoothing distribution on $\mathbb{P}(v_0|Y_j)$ under $\Psi^{(j)}$, that is under j applications of Ψ.

Theorem 1.12. *Let $\mathbb{P}(v_0|y)$ denote the smoothing distribution on the discrete time interval $j \in \mathbb{J}_0$, and $\mathbb{P}(v_J|Y_J)$ the filtering distribution at time $j = J$ for the deterministic dynamics model (1.3). Then the pushforward of the smoothing distribution on v_0 under $\Psi^{(J)}$ is the same as the filtering distribution at time J:*

$$\Psi^{(J)} \star \mathbb{P}(v_0|Y_J) = \mathbb{P}(v_J|Y_J).$$

1.6. Well-Posedness

Well-posedness of a mathematical problem refers, generally, to the existence of a unique solution which depends continuously on the specified data. We have shown, for both filtering and smoothing, how to construct a probabilistic solution to the problem of determining the signal given the data. We now investigate the continuous dependence of this solution on the data.

To this end, let μ and μ' denote two probability measures which have strictly positive Lebesgue densities ρ and ρ' on $\mathbb{R}^\ell$. Throughout this section, all integrals are over $\mathbb{R}^\ell$. Define the **Hellinger distance** between μ and μ' as

$$\begin{aligned} d_{\text{Hell}}(\mu,\mu') &= \sqrt{\frac{1}{2}\int_{\mathbb{R}^\ell}\left(1-\sqrt{\frac{\rho'(x)}{\rho(x)}}\right)^2\rho(x)dx} \\ &= \left(\frac{1}{2}\mathbb{E}^\mu\left(1-\sqrt{\frac{\rho'(x)}{\rho(x)}}\right)^2\right)^{\frac{1}{2}}. \end{aligned} \tag{1.31}$$

It is of interest to relate this to the perhaps better known **total variation distance** (TV) defined by

$$\begin{aligned} d_{\text{TV}}(\mu,\mu') &= \frac{1}{2}\int_{\mathbb{R}^\ell}\left|1-\frac{\rho'(x)}{\rho(x)}\right|\rho(x)dx \\ &= \frac{1}{2}\mathbb{E}^\mu\left|1-\frac{\rho'(x)}{\rho(x)}\right|. \end{aligned} \tag{1.32}$$

The Hellinger and total variation distances are related as follows:

Lemma 1.13. *Assume that two measures μ and μ' are both absolutely continuous with respect to Lebesgue measure with strictly positive Lebesgue densities. Then*

$$\frac{1}{\sqrt{2}}d_{\text{TV}}(\mu,\mu') \le d_{\text{Hell}}(\mu,\mu') \le d_{\text{TV}}(\mu,\mu')^{\frac{1}{2}}. \tag{1.33}$$

Proof. Let $\mu(dx) = \rho(x)dx$ and $\mu'(dx) = \rho'(x)dx$. Then

$$\frac{d\mu'}{d\mu}(x) = \frac{\rho'(x)}{\rho(x)}$$

and we have

$$\begin{aligned}
d_{\mathrm{TV}}(\mu,\mu') &= \frac{1}{2}\int \Big|\sqrt{1}-\sqrt{\frac{d\mu'}{d\mu}}\Big|\Big|\sqrt{1}+\sqrt{\frac{d\mu'}{d\mu}}\Big|d\mu \\
&\le \sqrt{\Big(\frac{1}{2}\int\Big(\sqrt{1}-\sqrt{\frac{d\mu'}{d\mu}}\Big)^2 d\mu\Big)}\sqrt{\Big(\frac{1}{2}\int\Big(\sqrt{1}+\sqrt{\frac{d\mu'}{d\mu}}\Big)^2 d\mu\Big)} \\
&\le \sqrt{\Big(\frac{1}{2}\int\Big(\sqrt{1}-\sqrt{\frac{d\mu'}{d\mu}}\Big)^2 d\mu\Big)}\sqrt{\Big(\int\Big(1+\frac{d\mu'}{d\mu}\Big)d\mu\Big)} \\
&= \sqrt{2}d_{\mathrm{Hell}}(\mu,\mu')
\end{aligned}$$

as required for the lower bound. Here, and in the remainder of the proof, all integrals are over $\mathbb{R}^\ell$.

For the upper bound note that, for any positive a, b,

$$|\sqrt{a}-\sqrt{b}| \le \sqrt{a}+\sqrt{b}.$$

Thus

$$\begin{aligned}
d_{\mathrm{Hell}}(\mu,\mu')^2 &\le \frac{1}{2}\int \Big|\sqrt{1}-\sqrt{\frac{d\mu'}{d\mu}}\Big|\Big|\sqrt{1}+\sqrt{\frac{d\mu'}{d\mu}}\Big|d\mu \\
&= \frac{1}{2}\int \Big|1-\frac{d\mu'}{d\mu}\Big|d\mu \\
&= d_{\mathrm{TV}}(\mu,\mu').
\end{aligned}$$

□ □

Why do we bother to introduce the Hellinger distance, rather than working with the more familiar TV distance? The answer stems from the following calculations. Let $f:\mathbb{R}^\ell \to \mathbb{R}^p$. Then

$$\begin{aligned}
|\mathbb{E}^\mu f(x) - \mathbb{E}^{\mu'} f(x)| &\le \int |f(x)||\rho(x)-\rho'(x)|dx \\
&= \int \sqrt{2}|f(x)||\sqrt{\rho(x)}+\sqrt{\rho'(x)}|\cdot\frac{1}{\sqrt{2}}|\sqrt{\rho(x)}-\sqrt{\rho'(x)}|dx \\
&\le \left(\int 2|f(x)|^2|\sqrt{\rho(x)}+\sqrt{\rho'(x)}|^2dx\right)^{\frac{1}{2}}\left(\frac{1}{2}\int|\sqrt{\rho(x)}-\sqrt{\rho'(x)}|^2dx\right)^{\frac{1}{2}} \\
&\le \left(\int 4|f(x)|^2(\rho(x)+\rho'(x))dx\right)^{\frac{1}{2}}\left(\frac{1}{2}\int\left(1-\frac{\sqrt{\rho'(x)}}{\sqrt{\rho(x)}}\right)^2\rho(x)dx\right)^{\frac{1}{2}} \\
&= 2(\mathbb{E}^\mu|f(x)|^2+\mathbb{E}^{\mu'}|f(x)|^2)^{\frac{1}{2}}d_{\mathrm{Hell}}(\mu,\mu').
\end{aligned}$$

Thus the Hellinger metric provides a direct way of estimating changes in expectation of square integrable functions:

$$|\mathbb{E}^\mu f(x) - \mathbb{E}^{\mu'} f(x)| \le 2(\mathbb{E}^\mu|f(x)|^2+\mathbb{E}^{\mu'}|f(x)|^2)^{\frac{1}{2}}d_{\mathrm{Hell}}(\mu,\mu'). \tag{1.34}$$

In particular if two measures μ and μ' are $\mathcal{O}(\epsilon)$ close in the Hellinger metric, and if the function $f(x)$ is square integrable with respect to x distributed according to μ and μ', then expectations of $f(x)$ with respect to μ and μ' are also $\mathcal{O}(\epsilon)$ close. To get an analogous result using $\mathcal{O}(\epsilon)$ closeness in the TV metric requires the stronger assumption that f is finite almost surely with respect to both μ and μ'. Denote the almost sure

upper bound by $f_{\max}$. Under this assumption

$$\begin{aligned}|\mathbb{E}^{\mu} f(x) - \mathbb{E}^{\mu'} f(x)| &\leq \int |f(x)||\rho(x) - \rho'(x)|dx \\ &\leq 2f_{\max}\left(\frac{1}{2}\int |\rho(x) - \rho'(x)|dx\right) \\ &\leq 2f_{\max}\left(\frac{1}{2}\int \left|1 - \frac{\rho'(x)}{\rho(x)}\right| \rho(x)dx\right) \\ &= 2f_{\max} d_{\mathrm{TV}}(\mu, \mu').\end{aligned}$$

Thus if two measures μ and μ' are $\mathcal{O}(\epsilon)$ in the TV metric and the function f is bounded almost surely with respect to both measures then expectations of $f(x)$ with respect to μ and μ' are also $\mathcal{O}(\epsilon)$ close. However, if $f(x)$ is only square-integrable with respect to μ and μ' then expectations of $f(x)$ with respect to μ and μ' can only be proved to be $\mathcal{O}(\epsilon^{\frac{1}{2}})$ close in general, as follows from (1.33) and (1.34). For these reasons we work with the Hellinger metric.

We let μ_0 denote the prior measure on v for the smoothing problem arising in stochastic dynamics, and μ and μ' the posterior measures resulting from two different instances of the data, y and y' respectively. The following theorem shows that the posterior measure is in fact Lipschitz continuous, in the Hellinger metric, with respect to the data.

Theorem 1.14. *Consider the smoothing problem arising from the stochastic dynamics model (1.1). Assume that* $\mathbb{E}^{\mu_0}(\sum_{j=0}^{J-1} 1 + |h(v_{j+1})|^2)^{\frac{1}{2}} < \infty$. *Then, for* $|y|, |y'| \leq r$ *there exists* $c = c(r)$ *such that*

$$d_{\mathrm{Hell}}(\mu, \mu') \leq c|y - y'|.$$

Proof. Let ρ_0, ρ and ρ' denote the Lebesgue densities on μ_0, μ and μ' respectively. Then

$$\begin{aligned}\rho_0(v) &= \frac{1}{Z_0}\exp(-\mathsf{J}(v)), \\ \rho(v) &= \frac{1}{Z}\exp(-\mathsf{J}(v) - \Phi(v; y)), \\ \rho'(v) &= \frac{1}{Z'}\exp(-\mathsf{J}(v) - \Phi(v; y')),\end{aligned}$$

where

$$\begin{aligned}Z_0 &= \int \exp(-\mathsf{J}(v))dv, \\ Z &= \int \exp(-\mathsf{J}(v) - \Phi(v; y))dv, \\ Z' &= \int \exp(-\mathsf{J}(v) - \Phi(v; y'))dv.\end{aligned}$$

Hence, in particular, $\mu_0(dv) = \rho_0(v)dv, \mu(dv) = \rho(v)dv$ and $\mu'(dv) = \rho'(v)dv$. Furthermore

$$\exp\big(-\mathsf{J}(v)\big)dv = Z_0\rho_0(v)dv = Z_0\mu_0(dv).$$

We use this last identity repeatedly in what follows.

Thus we have

$$\begin{aligned}d_{\mathrm{Hell}}(\mu, \mu')^2 &= \frac{1}{2}\int |\sqrt{\rho(v)} - \sqrt{\rho'(v)}|^2 dv \\ &= \frac{1}{2}\int Z_0\left|\frac{1}{\sqrt{Z}}e^{-\frac{1}{2}\Phi(v;y)} - \frac{1}{\sqrt{Z'}}e^{-\frac{1}{2}\Phi(v;y')}\right|^2 \mu_0(dv) \\ &\leq I_1 + I_2,\end{aligned}$$

where

$$I_1 = Z_0 \int \frac{1}{Z} |e^{-\frac{1}{2}\Phi(v;y)} - e^{-\frac{1}{2}\Phi(v;y')}|^2 \mu_0(dv)$$

and

$$\begin{aligned} I_2 &= Z_0 \Big| \frac{1}{\sqrt{Z}} - \frac{1}{\sqrt{Z'}} \Big|^2 \int e^{-\Phi(v;y')} \mu_0(dv) \\ &= Z' \Big| \frac{1}{\sqrt{Z}} - \frac{1}{\sqrt{Z'}} \Big|^2. \end{aligned}$$

Since $\Phi(v;y) \geq 0$ and $\Phi(v;y') \geq 0$ we have

$$\begin{aligned} |Z - Z'| &\leq Z_0 \int |e^{-\Phi(v;y)} - e^{-\Phi(v;y')}| \mu_0(dv) \\ &\leq Z_0 \int |\Phi(v;y) - \Phi(v;y')| \mu_0(dv). \end{aligned}$$

By definition

$$\begin{aligned} |\Phi(v;y) - \Phi(v;y')| &\leq \frac{1}{2} \sum_{j=0}^{J-1} |y_{j+1} - y'_{j+1}|_\Gamma |y_{j+1} + y'_{j+1} - 2h(v_{j+1})|_\Gamma \\ &\leq \frac{1}{2} \left(\sum_{j=0}^{J-1} |y_{j+1} - y'_{j+1}|_\Gamma^2 \right)^{\frac{1}{2}} \left(\sum_{j=0}^{J-1} |y_{j+1} + y'_{j+1} - 2h(v_{j+1})|_\Gamma^2 \right)^{\frac{1}{2}} \\ &\leq c(r)|y - y'| \left(\sum_{j=0}^{J-1} 1 + |h(v_{j+1})|^2 \right)^{\frac{1}{2}}. \end{aligned}$$

Thus

$$|Z - Z'| \leq c(r)|y - y'|.$$

Hence, since $Z, Z' > 0$, $I_2 \leq c(r)|y - y'|$. A similar argument shows that $I_1 \leq c(r)|y - y'|$ and the proof is complete. □

Corollary 1.15. *Let $f : \mathbb{R}^N \to \mathbb{R}^p$ be such that $\mathbb{E}^{\mu_0}|f(v)|^2 < \infty$ and assume further that $\mathbb{E}^{\mu_0} \left(\sum_{j=0}^{J-1} 1 + |h(v_{j+1})|^2 \right)^{\frac{1}{2}} < \infty$. Then*

$$|\mathbb{E}^{\mu} f(x) - \mathbb{E}^{\mu'} f(x)| \leq c|y - y'|.$$

Proof. First note that, since $\Phi(v;y) \geq 0$, $\mathbb{E}^{\mu}|f(v)|^2 \leq c\mathbb{E}^{\mu_0}|f(v)|^2$, and similarly for μ'. The result follows from (1.34) and Theorem 1.14. □

Corollary 1.16. *Let $g : \mathbb{R}^n \to \mathbb{R}^p$ be such that $\mathbb{E}^{\mu_0}|g(v_J)|^2 < \infty$ and assume further that $\mathbb{E}^{\mu_0} \left(\sum_{j=0}^{J-1} 1 + |h(v_{j+1})|^2 \right)^{\frac{1}{2}} < \infty$. If μ_J and μ'_J denote the filtering distributions at time J corresponding to data Y_J, Y'_J respectively, then*

$$|\mathbb{E}^{\mu_J} g(x) - \mathbb{E}^{\mu'_J} g(x)| \leq c|Y_J - Y'_J|.$$

Proof. Since, by Theorem 1.10, μ_J is the marginal of the smoother on the v_J coordinate, the result follows from Corollary 1.15 by choosing $f(v) = g(v_J)$. □

A similar theorem, and corollaries, may be proved for the case of deterministic dynamics (1.3), and the posterior $\mathbb{P}(v_0|y)$. We state the theorem and leave its proof to the reader. We let ν_0 denote the prior Gaussian measure $N(m_0, C_0)$ on v_0 for the smoothing problem arising in deterministic dynamics, and μ and μ' the posterior measures on v_0 resulting from two different instances of the data, y and y' respectively.

Theorem 1.17. *Consider the smoothing problem arising from the deterministic dynamics model (1.3). Assume that* $\mathbb{E}^{\nu_0}\left(\sum_{j=0}^{J-1} 1 + \left|h\left(\Psi^{(j+1)}(v_0)\right)\right|^2\right)^{\frac{1}{2}} < \infty$. *Then, for* $|y|, |y'| \leq r$ *there exists* $c = c(r)$ *such that*

$$d_{\text{Hell}}(\mu, \mu') \leq c|y - y'|.$$

1.7. Assessing The Quality of Data Assimilation Algorithms

It is helpful when studying algorithms for data assimilation to ask two questions: (i) how informative is the data we have?; (ii) how good is our algorithm at extracting this information? We take these in turn.

Answering question (i) is independent of any particular algorithm: it concerns the properties of the Bayesian posterior pdf itself. In some cases we will be interested in studying the properties of the probability distribution on the signal, or the initial condition, for a particular instance of the data generated from a particular instance of the signal, which we call the **truth**. In this context we will use the notation $y^\dagger = \{y_j^\dagger\}$ to denote the *realization* of the data generated from a particular realization of the truth $v^\dagger = \{v_j^\dagger\}$. We first discuss properties of the smoothing problem for stochastic dynamics. **Posterior consistency** concerns the question of the limiting behaviour of $\mathbb{P}(v|y^\dagger)$ as either $J \to \infty$ (large data sets) or $|\Gamma| \to 0$ (small noise). A key question is whether $\mathbb{P}(v|y^\dagger)$ converges to the truth in either of these limits; this might happen, for example, if $\mathbb{P}(v|y^\dagger)$ becomes closer and closer to a Dirac probability measure centred on $v^\dagger$. When this occurs we have Bayesian posterior consistency and it is of interest to study the rate at which the limit is attained. Such questions concern the information content of the data; they do not refer to any algorithm and therefore they are not concerned with the quality of any particular algorithm. When considering filtering, rather than smoothing, a particular instance of this question concerns marginal distributions: for example one may be concerned with posterior consistency of $\mathbb{P}(v_J|y_J^\dagger)$ with respect to a Dirac on $v_J^\dagger$ in the filtering case, see Theorem 1.10; for the case of deterministic dynamics the distribution $\mathbb{P}(v|y^\dagger)$ is completely determined by $\mathbb{P}(v_0|y^\dagger)$ (see Theorem 1.12) so one may discuss posterior consistency of $\mathbb{P}(v_0|y^\dagger)$ with respect to a Dirac on $v_0^\dagger$.

Here it is appropriate to mention the important concept of **model error**. In many (in fact most) applications the physical system which generates the data stream $\{y_j\}$ can be (sometimes significantly) different from the mathematical model used, at least in certain aspects. This can be thought of conceptually by imagining data generated by (1.2), with $v^\dagger = \{v_j^\dagger\}$ governed by the deterministic dynamics

$$v_{j+1}^\dagger = \Psi_{\text{true}}(v_j^\dagger),\ j \in \mathbb{N}. \tag{1.35a}$$

$$v_0^\dagger = u \sim N(m_0, C_0). \tag{1.35b}$$

Here the function Ψ_{true} governs the dynamics of the truth which underlies the data. We assume that the true solution operator is not known to us exactly, and seek instead to combine the data with the stochastic dynamics model (1.1); the noise $\{\xi_j\}$ is used to allow for the discrepancy between the true solution operator Ψ_{true} and that used in our model, namely Ψ. It is possible to think of many variants on this situation. For example, the dynamics of the truth may be stochastic; or the dynamics of the truth may take place in a higher-dimensional space than that used in our models, and may need to be projected into the model space. Statisticians sometimes refer to the situation where the data source differs from the model used as **model misspecification**.

We now turn from the information content, or quality, of the data to the quality of algorithms for data assimilation. We discuss three approaches to assessing quality. The first fully Bayesian approach can be defined independently of the quality of the data. The second estimation approach entangles the properties of the algorithm with the quality of the data. We discuss these two approaches in the context of the smoothing problem for stochastic dynamics. The reader will easily see how to generalize to smoothing for deterministic dynamics, or to filtering. The third approach is used in operational numerical weather prediction and judges quality by the ability to predict.

Bayesian Quality Assessment. Here we assume that the algorithm under consideration provides an approximation $\mathbb{P}_{\text{approx}}(v|y)$ to the true posterior distribution $\mathbb{P}(v|y)$. We ask the question: how close is

$\mathbb{P}_{\text{approx}}(v|y)$ to $\mathbb{P}(v|y)$. We might look for a distance measure between probability distributions, or we might simply compare some important moments of the distributions, such as the mean and covariance. Note that this version of quality assessment does not refer to the concept of a true solution $v^\dagger$. We may apply it with $y = y^\dagger$, but we may also apply it when there is model error present and the data comes from outside the model used to perform data assimilation. However, if combined with Bayesian posterior consistency, when $y = y^\dagger$, then the triangle inequality relates the output of the algorithm to the truth $v^\dagger$.

Signal Estimation Quality Assessment. Here we assume that the algorithm under consideration provides an approximation to the signal v underliying the data, which we denote by v_{approx}, i.e. that v_{approx} attempts to *track* the signal. If the algorithm actually provides a probability distribution, then this estimate might be, for example, the mean. We ask the question: if the algorithm is applied in the situation where the data $y^\dagger$ generated from the the signal $v^\dagger$, how close is v_{approx} to $v^\dagger$? There are two importants effects at play here: the first is the information content of the data – does the data actually contain enough information to allow for accurate reconstruction of the signal in principle; and the second is the role of the specific algorithm used – does the specific algorithm in question have the ability to extract this information when it is present. This approach thus measures the overall effect of these two in combination.

Forecast Skill. In many cases the goal of data assimilation is to provide better forecasts of the future, for example in numerical weather prediction. In this context data assimilation algorithms can be benchmarked by their ability to make forecasts. This can be discussed in both the Bayesian Quality and Signal Estimation senses. For simplicity of exposition we discuss Bayesian Estimation forecast skill in the context of stochastic dynamics. The Bayesian k-lag forecast skill can be defined by studying the distance between the approximation $\mathbb{P}_{\text{approx}}(v|y)$ and $\mathbb{P}(v|y)$ when both are pushed forward from the end-point of the data assimilation window by k applications of the dynamical model (1.1); this model defines a Markov transition kernel which is applied $k-$times to produce a forecast. We discuss Signal Estimation forecast skill in the context of deterministic dynamics. Using v_{approx} at the end point of the assimilation window as an initial condition, we run the model (1.3) forward by k steps and compare the output with $v^\dagger_{j+k}$. In practical application, this forecast methodology inherently confronts the effect of model error, since the data used to test forecasts is real data which is not generated by the model used to assimilate, as well as information content in the data and algorithm quality.

Consistency Checks. We describe an important consistency check that may be applied to the output of an algorithm. Given an approximation of the updated signal $\{v_{\text{approx},j}\}$ for $j \in \mathbb{J}_0$ the data assimilation window, we can consider the corresponding approximation just before the update, i.e. the one-step forecasts $\widehat{v}_{\text{approx},j}$ based on the previous updated estimate $v_{\text{approx},j-1}$. We can study the approximate *innovations*, or one-step forecast-observation discrepancies given by

$$d_j = [h(\widehat{v}_{\text{approx},j}) - y_j]. \tag{1.36}$$

If we assume that $\widehat{v}_{\text{approx},j}$ is a good approximation to $\mathbb{E}(v_j|Y_{j-1})$ for $j \in \mathbb{J}$ and we also have an approximation of the covariance $\widehat{C}_{\text{approx},j}$ of the conditioned signal $v_j|Y_{j-1}$, then the statistics of $\{d_j\}_{j\in\mathbb{J}}$ can be compared with this approximation, given the assumed Gaussian statistics of the model. In particular if $h(\cdot) := H\cdot$ is linear, and J is large enough we would expect that

$$\frac{1}{J}\sum_{j\in\mathbb{J}} d_j \approx 0, \quad \text{and} \quad \frac{1}{J}\sum_{j\in\mathbb{J}} d_j \otimes d_j \approx H\left(\frac{1}{J}\sum_{j\in\mathbb{J}} \widehat{C}_{\text{approx},j}\right) H^T + \Gamma.$$

If the empirical statistics of the innovations are inconsistent with the assumed model, then they can be used to improve the model used in the future; this is known as *reanalysis*. There are a number of variants on this consistency check, and generalizations to nonlinear observation operators are also possible. We discuss the idea of the variant known as *rank histograms* at the end of Chapter 3.

1.8. Illustrations

In order to build intuition concerning the probabilistic viewpoint on data assimilation we describe some simple examples where the posterior distribution may be visualized easily. For this reason we concentrate

on the case of one-dimensional deterministic dynamics; the posterior pdf $\mathbb{P}(v_0|y)$ for deterministic dynamics is given by Theorem 1.9. It is one-dimensional when the dynamics is one-dimensional. In section 2 we will introduce more sophisticted sampling methods to probe probability distributions in higher dimensions which arise from noisy dynamics and/or from high dimensional models.

Figure 10 concerns the scalar linear problem from Example 1.1 (recall that throughout this section we consider only the case of deterministic dynamics) with $\lambda = 0.5$. We employ a prior $N(4, 5)$, we assume that $h(v) = v$, and we set $\Gamma = \gamma^2$ and consider two different values of γ and two different values of J, the number of observations. The figure shows the posterior distribution in these various parameter regimes. The true value of the initial condition which underlies the data is $v_0^\dagger = 0.5$. For both $\gamma = 1.0$ and 0.1 we see that, as the number of observations J increases, the posterior distribution appears to converge to a limiting distribution. However for smaller γ the limiting distribution has much smaller variance, and is centred closer to the true initial condition at 0.5. Both of these observations can be explained, using the fact that the problem is explicitly solvable: we show that for fixed γ and $J \to \infty$ the posterior distribution has a limit, which is a Gaussian with non-zero variance. And for fixed J as $\gamma \to 0$ the posterior distribution converges to a Dirac measure (Gaussian with zero variance) centred at the truth $v_0^\dagger$.

To see these facts we start by noting that from Theorem 1.9 the posterior distribution on $v_0|y$ is proportional to the exponential of

$$\mathsf{I}_0(v_0; y) = \frac{1}{2\gamma^2} \sum_{j=0}^{J-1} |y_{j+1} - \lambda^{j+1} v_0|^2 + \frac{1}{2\sigma_0^2} |v_0 - m_0|^2$$

where σ_0^2 denotes the prior variance C_0. As a quadratic form in v_0 this defines a Gaussian posterior distribution and we may complete the square to find the posterior mean m and variance σ_{post}^2:

$$\frac{1}{\sigma_{\text{post}}^2} = \frac{1}{\gamma^2} \sum_{j=0}^{J-1} \lambda^{2(j+1)} + \frac{1}{\sigma_0^2} = \frac{1}{\gamma^2} \Big(\frac{\lambda^2 - \lambda^{2J+2}}{1 - \lambda^2} \Big) + \frac{1}{\sigma_0^2}$$

and

$$\frac{1}{\sigma_{\text{post}}^2} m = \frac{1}{\gamma^2} \sum_{j=0}^{J-1} \lambda^{(j+1)} y_{j+1} + \frac{1}{\sigma_0^2} m_0.$$

We note immediately that the posterior variance is independent of the data. Furthermore, if we fix γ and let $J \to \infty$ then for any $|\lambda| < 1$ we see that the large J limit of the posterior variance is determined by

$$\frac{1}{\sigma_{\text{post}}^2} = \frac{1}{\gamma^2} \Big(\frac{\lambda^2}{1 - \lambda^2} \Big) + \frac{1}{\sigma_0^2}$$

and is non-zero; thus uncertainty remains in the posterior, even in the limit of large data. On the other hand, if we fix J and let $\gamma \to 0$ then $\sigma_{\text{post}}^2 \to 0$ so that uncertainty disappears in the limit. It is then natural to ask what happens to the mean. To this end we assume that the data is itself generated by the linear model of Example 1.1 so that

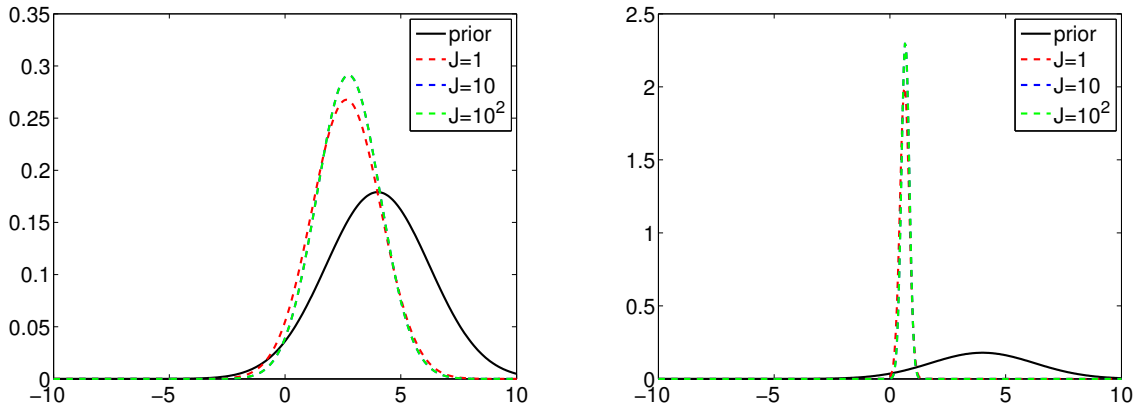
$$y_{j+1} = \lambda^{j+1} v_0^\dagger + \gamma \zeta_{j+1}$$

where ζ_j is an i.i.d. Gaussian sequence with $\zeta_1 \sim N(0, 1)$. Then

$$\frac{1}{\sigma_{\text{post}}^2} m = \frac{1}{\gamma^2} \Big(\frac{\lambda^2 - \lambda^{2J+2}}{1 - \lambda^2} \Big) v_0^\dagger + \frac{1}{\gamma} \sum_{j=0}^{J-1} \lambda^{(j+1)} \zeta_{j+1} + \frac{1}{\sigma_0^2} m_0.$$

Using the formula for σ_{post}^2 we obtain

$$\Big(\frac{\lambda^2 - \lambda^{2J+2}}{1 - \lambda^2} \Big) m + \frac{\gamma^2}{\sigma_0^2} m = \Big(\frac{\lambda^2 - \lambda^{2J+2}}{1 - \lambda^2} \Big) v_0^\dagger + \gamma \sum_{j=0}^{J-1} \lambda^{(j+1)} \zeta_{j+1} + \frac{\gamma^2}{\sigma_0^2} m_0.$$



(a) $\gamma = 1$ (b) $\gamma = 0.1$

FIG 10. *Posterior distribution for Examples 1.1 for different levels of observational noise. The true initial condition used in both cases is $v_0 = 0.5$, while we have assumed that $C_0 = 5$ and $m_0 = 4$ for the prior distribution.*

From this it follows that, for fixed J and as $\gamma \to 0$, $m \to v_0^\dagger$, almost surely with respect to the noise realization $\{\zeta_j\}_{j\in\mathbb{J}}$. This is an example of posterior consistency.

We now study Example 1.4 in which the true dynamics are no longer linear. We start our investigation taking $r = 2$ and investigate the effect of choosing different prior distributions. Before discussing the properties of the posterior we draw attention to two facts. Firstly, as Figure 5a shows, the system converges in a small number of steps to the fixed point at $1/2$ for this value of $r = 2$. And secondly the initial conditions v_0 and $1 - v_0$ both result in the same trajectory, if the initial condition is ignored. The first point implies that, after a small number of steps, the observed trajectory contains very little information about the initial condition. The second point means that, since we observe from the first step onwards, only the prior can distinguish between v_0 and $1 - v_0$ as the starting point.

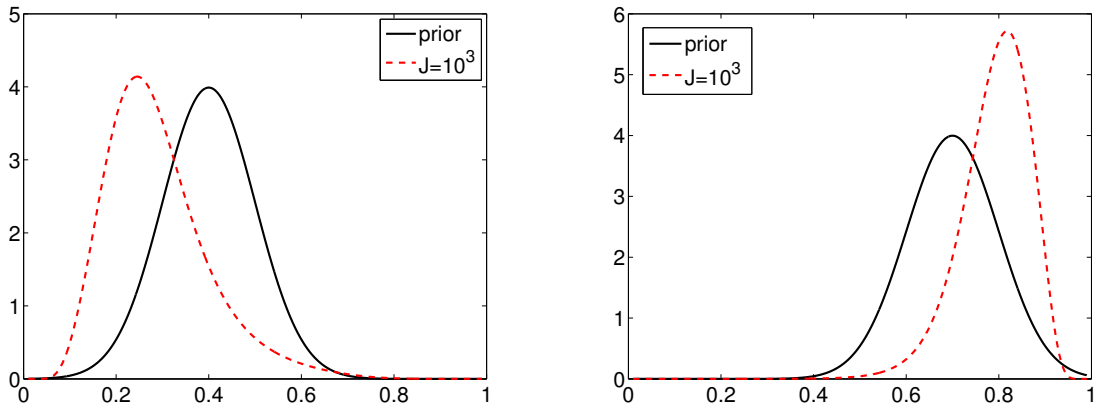
Figure 11 concerns an experiment in which the true initial condition underlying the data is $v_0^\dagger = 0.1$. Two different priors are used, both with $C_0 = 0.01$, giving a standard deviation of 0.1, but with different means. The figure illustrates two facts: firstly, even with 10^3 observations, the posterior contains considerable uncertainty, reflecting the first point above. Secondly the prior mean has an important role in the form of the posterior pdf: shifting the prior mean to the right, from $m_0 = 0.4$ to $m_0 = 0.7$, results in a posterior which favours the initial condition $1 - v_0^\dagger$ rather than the truth $v_0^\dagger$.

This behaviour of the posterior changes completely if we assume a flatter prior. This is illustrated in Figure 12 where we consider the prior $N(0.4, C_0)$ with $C_0 = 0.5$ and 5 respectively. As we increase the prior covariance the mean players a much weaker role than in the preceding experiments: we now obtain a bimodal posterior centred around both the true initial condition $v_0^\dagger$, and also around $1 - v_0^\dagger$.

In Figure 13 we consider the quadratic map (1.12) with $r = 4$. We use the prior $N(0.5, 0.01)$ and observational standard deviation $\gamma = 0.2$. Here, after only five observations the posterior is very peaked, although because of the $v \mapsto 1 - v$ symmetry mentioned above, there are two symmetrically related peaks; see Figure 13a. It is instructive to look at the negative of the logarithm of the posterior pdf which, upto an additive constant, is given by $\mathsf{I}_0(v_0; y)$ in Theorem 1.9. The function $\mathsf{I}_0(\cdot; y)$ is shown in Figure 13b. Its complexity indicates the considerable complications underlying solution of the smoothing problem. We will return to this last point in detail later. Here we simply observe that normalizing the posterior distribution requires evaluation of the integral

$$\int e^{-\mathsf{I}(v_0,y)} dv_0.$$

This integral may often be determined almost entirely by very small intervals of v_0, meaning that this calculation requires some care. We note, however, that the sampling methods that we will describe in the next chapter do not require evaluation of this integral.

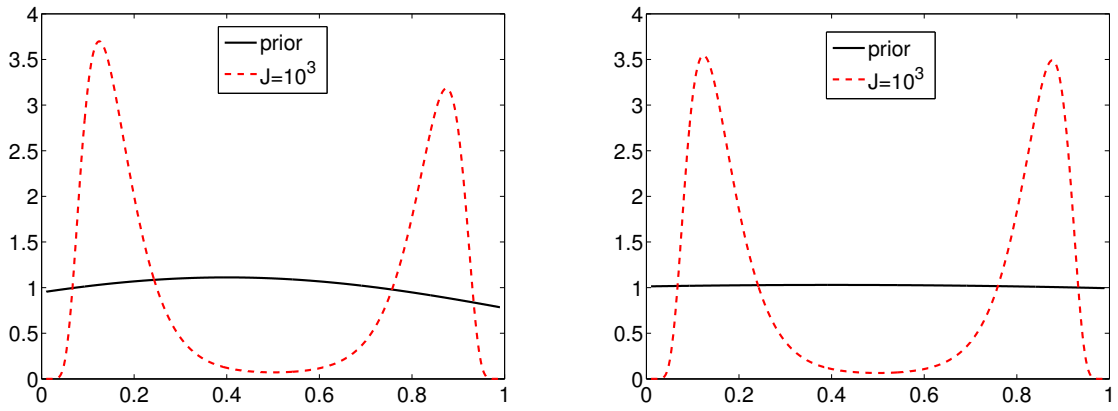


(a) $r = 2$, $m_0 = 0.4$ (b) $r = 2$ $m_0 = 0.7$

FIG 11. *Posterior distribution for Example 1.4 for $r = 2$ in the case of different means for the prior distribution. We have used $C_0 = 0.01$, $\gamma = 0.1$ and true initial condition $v_0 = 0.1$, see also* `p2.m` *in Section 4.1.2.*

1.9. Bibliographic Notes

- Section 1.1 Data Assimilation is a subject which has its roots in the geophysical sciences, and is driven by the desire to improve inaccurate models of complex dynamically evolving phenomena by means of incorporation of data. The book [Kal03] describes data assimilation from the viewpoint of atmospheric weather prediction, whilst the book [Ben02] describes the subject from the viewpoint of oceanography. These two subjects were the initial drivers for evolution of the field. However, other applications are increasingly using the methodology of data assimilation, and the oil industry in particular is heavily involved in the use, and development, of algorithms in this area [ORL08]. The recent book [Aba13] provides a perspective on the subject the the viewpoint of physics and nonlinear dynamical systems. The article [ICGL97] is a useful one to read because it establishes a notation which is now widely used in the applied communities and the articles [Nic03, AJSV08] provide simple introductions to various aspects of the subject from a mathematical perspective. The special edition of the journal PhysicaD, devoted to Data Assimilation, [IJ07], provides an overview of the state of the art around a decade ago. Throughout we assume that the model noise ξ and observational noise η are Gaussian for convenience only and that this could be easily generalized.
- Section 1.2 The subject of deterministic discrete time dynamical systems of the form (1.3) is overviewed in numerous texts; see [Wig03] and the references therein, and Chapter 1 of [SH96], for example. The subject of stochastic discrete time dynamical systems of the form (1.1), and in particular the property of ergodicity which underlies Figure 4, is covered in some depth in [MT93]. The exact solutions of the quadratic map (1.12) for $r = 2$ and $r = 4$ may be found in [Sch70] and [Lor64] respectively. The Lorenz '63 model was introduced in [Lor63]. Not only does this paper demonstrate the possibility of chaotic behviour and sensitivity with respect to initial conditions, but it also makes a concrete connection between the three dimensional continuous time dynamical system and a one-dimensional chaotic map of the form (1.1). Furthermore, a subsequent computer assisted proof demonstrated rigorously that the ODE does indeed exhibit chaos [Tuc99, Tuc02]. The book [Spa82] discusses properties of the Lorenz '63 model in some detail and the book [Fal86] discussed properties such as fractcal dimension. The shift of origin that we have adopted for the Lorenz '63 model is explained in [Tem97]; it enables the model to be written in an abstract form which includes many geophysical models of interest, including the Lorenz '96 model introduced in [Lor96], and the Navier-Stokes equation on a two-dimensional torus [MW06, Tem97]. We now briefly describe this common abstract form. The vector $u \in \mathbb{R}^J$ ($J = 3$ for



(a) $r = 2$, $C_0 = 0.5$ (b) $r = 2$ $C_0 = 5$

FIG 12. *Posterior distribution for Example 1.4 for $r = 2$ in the case of different covariance for the prior distribution. We have used $m_0 = 0.4$, $\gamma = 0.1$ and true initial condition $v_0 = 0.1$.*

Lorenz '63, J arbitrary for Lorenz' 96) solves the equation

$$\frac{du}{dt} + Au + B(u,u) = f, \quad u(0) = u_0, \tag{1.37}$$

where there is $\lambda > 0$ such that, for all $w \in \mathbb{R}^J$,

$$\langle Aw, w\rangle \geq \lambda |w|^2, \quad \langle B(w,w), w\rangle = 0.$$

Taking the inner-product with u shows that

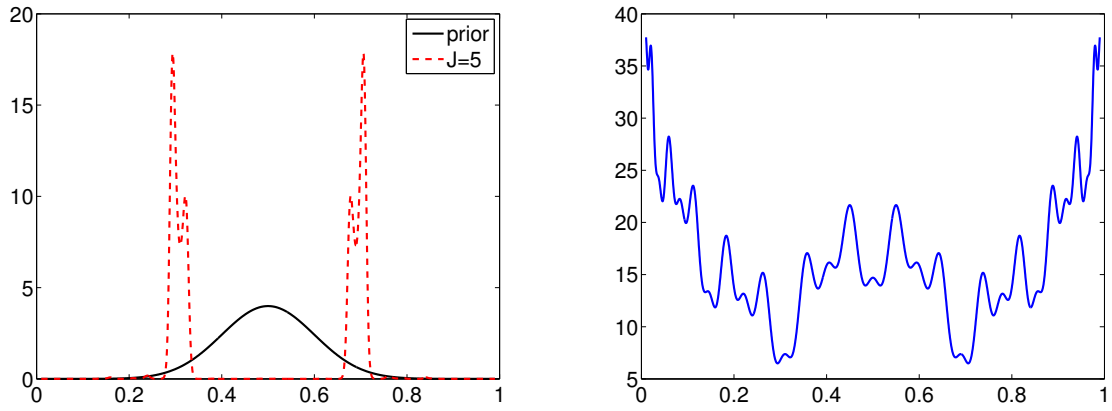
$$\frac{1}{2}\frac{d}{dt}|u|^2 + \lambda |u|^2 \leq \langle f, u\rangle.$$

If f is constant in time then this inequality may be used to show that (1.19) holds:

$$\frac{1}{2}\frac{d}{dt}|u|^2 \leq \frac{1}{2\lambda}|f|^2 - \frac{\lambda}{2}|u|^2.$$

Integrating this inequality gives the existence of an absoring set and hence leads to the existence of a global attractor; see [Tem97] or Chapter 2 of [SH96], for example.

- Section 1.3 contains the formulation of Data Assimilation as a fully nonlinear and non-Gaussian problem in Bayesian statistics. This formulation is not yet the basis of practical algorithms in the geophysical systems such as weather forecasting. This is because global weather forecast models involve $n = \mathcal{O}(10^9)$ unknowns, and incorporate $m = \mathcal{O}(10^6)$ data points daily; sampling the posterior on $\mathbb{R}^n$ given data in $\mathbb{R}^m$ in an online fashion, useable for forecasting, is beyond current algorithmic and computational capability. However the fully Bayesian perspective provides a fundamental mathematical underpinning of the subject, from which other more tractable approaches can be systematically derived. See [Stu10] for discussion of the Bayesian approach to inverse problems. Historically, data assimilation has not evolved from this Bayesian perspective, but has rather evolved out of the control theory perspective. This perspective is summarized well in the book [Jaz70]. However, the importance of the Bayesian perspective is increasingly being recognized in the applied communities. In addition to providing a starting point from which to derive approximate algorithms, it also provides a gold standard against which other more *ad hoc* algorithms can be benchmarked; this use of Bayesian methodology was suggested in [LS12] in the context of meteorology (see discussion that follows), and then employed in [ILS13] in the context of subsurface inverse problems arising in geophysics.



(a) $r = 4$ (b) $\mathsf{I}(v_0; y)$ $r = 4, J = 5$

FIG 13. *Posterior distribution for Example 1.4 for* $r = 4$. *We have used* $C_0 = 0.01, m_0 = 0.5$, $\gamma = 0.2$ *and true initial condition* $v_0 = 0.3$.

- Section 1.4 describes the filtering, or sequential, approach to data assimilation, within the fully Bayesian framework. For low dimensional systems the use of particle filters, which may be shown to rigorously approximate the required filtering distribution as it evolves in discrete time, has been enormously succesful; see [DG01] for an overview. Unfortunately, these filters can behave poorly in high dimensions [BLB08, BBL08, SBBA08]. Whilst there is ongoing work to overcome these problems with high-dimensional particle filtering, see [BCJ11, vL10a] for example, this work has yet to impact practical data assimilation in, for example, operational weather forecasting. For this reason the *ad hoc* filters, such as 3DVAR, Extended Kalman Filter and Ensemble Kalman Filter, described in Chapter 3, are of great practical importance. Their analysis is hence an important challenge for applied mathematicians.
- Section 1.6 Data assimilation may be viewed as an inverse problem to determine the signal from the observations. Inverse problems in differential equations are often ill-posed when viewed from a classical non-probabilistic perspective. One reason for this is that the data may not be informative about the whole signal so that many solutions are possible. However taking the Bayesian viewpoint, in which the many solutions are all given a probability, allows for well-posedness to be established. This idea is used for data assimilation problems arising in fluid mechanics in [CDRS09], for inverse problems arising in subsurface geophysics in [DS11, DHS12] and described more generally in [Stu10]. Well-posedness with respect to changes in the data is of importance in its own right, but also more generally because it underpins other stability results which can be used to control perturbations. In particular the effect of numerical approximation on integration of the forward model can be understood in terms of its effect on the posterior distribution; see [CDS10].
- Section 1.7 The subject of posterior consistency is central to the theory of statistics in general [VdV00], and within Bayesian statistics in particular [Ber85, BS94, GGR99]. Assessing the quality of data assimilation algorithms is typically performed in the "signal estimation" framework using **identical twin experiments** in which the data is generated from the same model used to estimate the signal; see [IJ07] and the references therein. The idea of assessing "Bayesian quality" has only recently been used within the data assimilation literature; see [LS12] where this approach is taken for the Navier-Stokes inverse problem formulated in [CDRS09]. The evaluation of algorithms by means of forecast skill is enormously influential in the field of numerical weather prediction and drives a great deal of algorithmic selection. The use of information theory to understand the effects of model error, and to evaluate filter performance, is introduced in [MGH12] and [MB13] respectively.

2. Discrete Time: Smoothing Algorithms

The formulation of the data assimilation problem described in the previous chapter is probabilistic, and its computational resolution requires the probing of a posterior probability distribution on signal given data. This probability distribution is on the signal sequence $\{v_j\}_{j=0}^J$ when the underlying dynamics is stochastic and given by (1.1); the posterior is specified in Theorem 1.8 and is proportional to $\exp\big(-\mathsf{I}(v;y)\big)$ given by (1.27). On the other hand, if the underlying dynamics is deterministic and given by (1.3), then the probability distribution is on the initial condition v_0 only; it is given in Theorem 1.9 and is proportional to $\exp\big(-\mathsf{I}_0(v_0;y)\big)$, with I_0 given by (1.28). Generically, in this chapter, we refer to the unknown variable as u, denoting v in the case of stochastic dynamics or v_0 in the case of deterministic dynamics.

In general the probability distributions of interest cannot be described by a finite set of parameters, except in a few simple situations such as the Gaussian scenario where the mean and covariance determine the distribution in its entirety. When the probability distributions cannot be described by a finite set of parameters, an expedient computational approach is through the idea of **Monte Carlo sampling**. The basic idea is to approximate a measure ν by a set of N samples $\{w^n\}_{n\in\mathbb{Z}^+}$ drawn, or approximately drawn, from ν to obtain the measure $\nu^N \approx \nu$ given by:

$$\nu^N = \frac{1}{N}\sum_{n=1}^{N} \delta_{w^n}. \tag{2.1}$$

If the w^n are exact draws from ν then the resulting approximation ν^N converges to the true measure ν as $N \to \infty$. [3] For example if $w = \{v_j\}_{j=0}^J$ is governed by the probability distribution μ_0 defined by the unconditioned dynamics (1.1), and with pdf determined by (1.25), then exact samples are easily to generate, simply by running the dynamics model forward in discrete time. However for the complex probability measures of interest here, where the dynamics is conditioned on data, exact samples are typically not possible and so instead we use the idea of **Monte Carlo Markov Chain (MCMC)** methods which provide a methodology for generating approximate samples.

In section 2.1 we provide some background concerning MCMC) methods, and in particular, the **Metropolis-Hastings** variant of MCMC , and show how they can be used to explore the posterior distribution. It can be very difficult to sample the probability distributions interest accurately, because of the two problems of high dimension and sensitive dependence on initial conditions. Whilst we do not claim to introduce the optimal algorithms to deal with these issues, we do discuss such issues in relation to the samplers we introduce, and we provide references to the active research ongoing in this area. Furthermore, although sampling of the posterior distribution may be computationally infeasible in many situations, where possible, it provides an important benchmark solution, enabling other algorithms to be compared against a "gold standard."

However, because sampling the posterior distribution can be prohibitively expensive, a widely used computational methodology is simply to find the point which maximizes the probability, using techniques from optimization. These are the **variational methods**, also known as **4DVAR**. We introduce this approach to the problem in section 2.2. In section 2.3 we provide numerical illustrations which showcase the MCMC and variational methods. The section concludes with bibliographic notes in 2.4.

2.1. Markov Chain-Monte Carlo Methods

In the case of stochastic dynamcs, equation (1.1), the posterior distribution of interest is the measure μ on $\mathbb{R}^N$, $N = n|\mathbb{J}_0|$, with density $\mathbb{P}(v|y)$ given in Theorem 1.8; in the case of deterministic dynamics, equation (1.3), it is the measure μ on $\mathbb{R}^n$ with density $\mathbb{P}(v_0|y)$ given in Theorem 1.9. In this section we describe the idea of Markov Chain-Monte Carlo (MCMC) methods for exploring such probability distributions.

We will start by describing the Metropolis-Hastings methodology for creating a Markov chain which is invariant for a general measure μ on $\mathbb{R}^\ell$ with pdf ρ. We then describe applications of this method to the problem of noise-free dynamics and noisy dynamics respectively. When describing the generic Metropolis-Hastings methodology we will use u (with indices) to denote the state of the Markov chain and w (with

[3] Indeed we prove such a result in Lemma 3.7 in the context of the particle filter.

indices) the proposed moves. Thus the current state u and proposed state w live in the space where signal sequences v lie, in the case of stochastic dynamics, and in the space where initial conditions v_0 lie, in the case of deterministic dynamics.

2.1.1. Metropolis-Hastings Methods

We are given a probability density function $\rho : \mathbb{R}^\ell \to \mathbb{R}^+$, with $\int \rho(u)du = 1$. Now consider a Markov transition kernel $q : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}^+$ with the property that $\int q(u,w)dw = 1$ for every $u \in \mathbb{R}^\ell$. In this section we use an expedient abuse of notation by writing, for fixed u, the function $q(u,w)$ to denote a pdf and, simultaneously, a probability measure $q(u,dw)$. We create a Markov chain $\{u^{(k)}\}_{k\in\mathbb{N}}$ which is invariant for ρ as follows. Define

$$a(u,w) = 1 \wedge \frac{\rho(w)q(w,u)}{\rho(u)q(u,w)} \tag{2.2}$$

The algorithm is:

1. Set $k = 0$ and choose $u^{(0)} \in \mathbb{R}^\ell$.
2. $k \to k+1$.
3. Draw $w^{(k)} \sim q(u^{(k-1)}, \cdot)$.
4. Set $u^{(k)} = w^{(k)}$ with probability $a(u^{(k-1)}, w^{(k)})$, $u^{(k)} = u^{(k-1)}$ otherwise.
5. Go to step 2.

At each step key in the algorithm there are two sources or randomness: that required for drawing $w^{(k)}$ in step 3; and that required for accepting or rejecting $w^{(k)}$ as the next $u^{(k)}$ in step 4. These two sources of randomness are chosen to be independent of one another. Furthermore, all the randomness at discrete algorithmic time k is independent of randomness at preceding discrete algorithmic times, conditional on $u^{(k-1)}$. Thus the whole procedure gives a Markov chain. If $\mathsf{z} = \{\mathsf{z}^{(j)}\}_{j\in\mathbb{N}}$ is an i.i.d. sequence of $U[0,1]$ random variables then we may write the algorithm as follows:

$$\begin{aligned} w^{(k)} &\sim q(u^{(k-1)}, \cdot) \\ u^{(k)} &= w^{(k)}\mathbb{I}\big(\mathsf{z}^{(j)} \le a(u^{(k-1)}, w^{(k)})\big) + u^{(k-1)}\mathbb{I}\big(\mathsf{z}^{(j)} > a(u^{(k-1)}, w^{(k)})\big). \end{aligned}$$

Here $\mathbb{I}$ denotes the indicator function of a set. We let $p : \mathbb{R}^\ell \times \mathbb{R}^\ell \to \mathbb{R}^+$ denote the transition kernel of the resulting Markov chain, and we let p^k denote the transition kernel over k steps. Thus $p^k(u,A) = \mathbb{P}(u^{(k)} \in A | u^{(0)} = u)$. Similarly as above, for fixed u, $p^k(u,dw)$ notes a probability measure on $\mathbb{R}^\ell$ with density $p^k(u,w)$. The resulting algorithm is known as a Metropolis-Hastings MCMC algorithm.

Remark 2.1. *The following two observations are central to Metropolis-Hastings MCMC methods.*

- *The construction of Metropolis-Hastings MCMC methods is designed to ensure* detailed balance*:*

$$\rho(u)p(u,w) = \rho(w)p(w,u). \tag{2.3}$$

  Once this condition is obtained it follows trivially that measure μ with density ρ is invariant since, integrating over u, we obtain

$$\begin{aligned} \int \rho(u)p(u,w)dy &= \int \rho(w)p(w,u)du \\ &= \rho(w)\int p(w,u)du \\ &= \rho(w). \end{aligned}$$

  This means that if the Markov chains is distributed according to measure with density ρ initially then it will be distributed according to the same measure at the next step, and hence for all algorithmic time.
- *Note that, in order to implement Metropolis-Hastings MCMC methods, it is not necessary to know the normalisation constants for $\rho(\cdot)$ and $q(u,\cdot)$ since only their ratios appear in a.*

The Metropolis-Hastings algorithm defined above satisfies the following, which requires definition of TV distance given in section 1.6:

Theorem 2.2. *If $u^{(0)} \sim \mu$ with Lebesque density ρ, then $u^{(k)} \sim \mu$ for all $k \in \mathbb{Z}^+$. Thus, if the Markov chain is ergodic, then for any bounded continuous $\varphi : \mathbb{R}^\ell \to \mathbb{R}$,*

$$\frac{1}{K}\sum_{k=1}^{K} \varphi(u^{(k)}) \xrightarrow{a.s.} \mathbb{E}^\mu \varphi(u)$$

for μ a.e. initial condition $u^{(0)}$. In particular, if there is probability measure with pdf p on $\mathbb{R}^\ell$ and $\varepsilon > 0$ such that, for all $u \in \mathbb{R}^\ell$ and all Borel sets $A \subseteq \mathbb{R}^\ell$, $p(u, A) \geq \varepsilon \mathsf{p}(A)$ then, for all $u \in \mathbb{R}^\ell$,

$$d_{\mathrm{TV}}\big(p^k(u,\cdot), \mu\big) \leq 2(1-\varepsilon)^k. \tag{2.4}$$

Furthermore, there is then a $C > 0$ such that

$$\frac{1}{K}\sum_{k=1}^{K} \varphi(u^{(k)}) = \mathbb{E}^\mu \varphi(x) + C\xi_K K^{-\frac{1}{2}} \tag{2.5}$$

where ξ_K converges weakly to $N(0,1)$ as $K \to \infty$.

We now describe some exemplars of Metropolis-Hastings methods tailored to the data assimilation problem. These are not to be taken as optimal MCMC methods for data assimilation, but rather as examples of how to construct proposal distributions $q(u,\cdot)$ for Metropolis-Hastings methods in the context of data assimilation; in any given application the proposal distribution plays a central role in the efficiency of the MCMC method and tailoring it to the specifics of the problem can have significant impact on efficiency of the MCMC method.

2.1.2. Deterministic Dynamics

In the case of deterministic dynamics (1.3), the measure of interest is a measure on the initial condition v_0 in $\mathbb{R}^n$. Perhaps the simplest Metropolis-Hastings algorithm is the **Random walk Metropolis** sampler which employs a Gaussian proposal, centred at the current state; we now illustrate this for the case of deterministic dynamics. Recall that the measure of interest is ν with pdf ϱ. Furthermore $\varrho \propto \exp\big(-\mathsf{I}_0(v_0; y)\big)$ as given in Theorem 1.9.

The Random walk Metropolis method proceeds as follows: given that we are at $v^{(k-1)} \in \mathbb{R}^n$, a current sample from the posterior distribution on the initial condition, we propose

$$w^{(k)} = u^{(k-1)} + \beta \iota^{(k-1)}$$

where $\iota^{(k-1)} \sim N(0, C_{\mathrm{prop}})$ for some symmetric positive-definite proposal covariance C_{prop} and small parameter $\beta > 0$; natural choices for this proposal covariance include the identity I or the prior covariance C_0. Because of the symmetry of such a random walk proposal it follows that $q(w,u) = q(u,w)$ and hence that

$$\begin{aligned} a(u,w) &= 1 \wedge \frac{\varrho(w)}{\varrho(u)} \\ &= 1 \wedge \exp\big(\mathsf{I}_0(u;y) - \mathsf{I}_0(w;y)\big). \end{aligned}$$

Thus, in particular, the proposed move to w is accepted with probability one if the value of I_0 is decreased by moving to w from the current state u. On the other hand, if I_0 increases then the proposed stated is accepted only with some probability less than one. Recall that I_0 is the sum of the prior penalization (background) and the model-data misfit functional:

$$\mathsf{I}_0(v_0;y) = \frac{1}{2}\left|C_0^{-\frac{1}{2}}(v_0 - m_0)\right|^2 + \sum_{j=0}^{J-1} \frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\Big(y_{j+1} - h\big(\Psi^{(j+1)}(v_0)\big)\Big)\right|^2.$$

The algorithm thus has a very natural interpretation in terms of the data assimilation problem. The algorithm has two key tuning parameters: the proposal covariance C_{prop} and the scale parameter β. The covariance can encode any prior knowledge, or guesses, about the relative strength of correlations in the model; given this the parameter β should be tuned to give an acceptance probability that is neither close to 0 not to 1. Numerical results illustrating the method are given in section 2.3.

2.1.3. Stochastic Dynamics

We now apply the Metropolis-Hastings methodology to the data assimilation smoothing problem in the case of the stochastic dynamics model (1.1). Thus the probability measure is on an entire signal sequence $\{v_j\}_{j=0}^J$ and not just on v_0; hence it lives on $\mathbb{R}^N$, with $N = n|\mathbb{J}_0| = n(J+1)$. It is possible to apply the random walk method to this situation, too, but we take the opportunity to introduce several different Metropolis-Hastings methods, in order to highlight the flexibility of the methodology. Furthermore, it is also possible to take the ideas behind the proposals introduced in this section and apply them in the case of deterministic dynamics.

In what follows recall the measures μ_0 and μ defined in section 1.3, with densities ρ_0 and ρ, representing (respectively) the measure on sequences v generated by (1.1) and the same measure when conditioned on data y from (1.2). We now construct two Markov chains $\{u^{(k)}\}_{k\in\mathbb{N}}$ which are invariant with respect to μ. These will both be Metropolis-Hastings methods and hence we need only specify the transition kernel $q(u, w)$, and identify the resulting acceptance probability $a(u, w)$. The sequence $\{w^{(k)}\}_{k\in\mathbb{Z}^+}$ will denote the proposals.

Independence Dynamics Sampler Here we choose the proposal $w^{(k)}$, independently of the current state $u^{(k-1)}$, from the prior μ_0 with density ρ_0. Thus we are simply proposing independent draws from the dynamical model (1.1), with no information from the data used in the proposal. Important in what follows is the observation that

$$\frac{\rho(v)}{\rho_0(v)} \propto \exp(-\Phi(v; y)). \tag{2.6}$$

With the given definition of proposal we have that $q(u, w) = \rho_0(w)$ and hence that

$$\begin{aligned} a(u, w) &= 1 \wedge \frac{\rho(w)q(w, u)}{\rho(u)q(u, w)} \\ &= 1 \wedge \frac{\rho(w)/\rho_0(w)}{\rho(u)/\rho_0(u)} \\ &= 1 \wedge \exp(\Phi(u; y) - \Phi(w; y)). \end{aligned}$$

The resulting MCMC method always accepts moves which decrease the model-data misfit functional $\Phi(.; y)$ given in (1.26); if this functional increases then the move is accepted with a probability less than one.

As we will see in the illustrations section 2.3 below, this method can be quite inefficient because of frequent rejections. These rejections are caused by attempts to move far from the current state, and in particular to proposed states which are based on the underlying stochastic dynamics, but not on the observed data. This typically leads to increases in the model-data misfit functional $\Phi(.; y)$. Even if data is not explicitly used in constructing the proposal, then this effect can be ameliorated by making local proposals, which do not move far from the current state. These are exemplified in the following MCMC algorithm.

The pCN Method. It is helpful in what follows to introduce the measure ϑ_0 with density π_0 found from μ_0 and ρ_0 in the case where $\Psi \equiv 0$. Thus

$$\pi_0(v) \propto \exp\left(-\frac{1}{2}\left|C_0^{-\frac{1}{2}}(v_0 - m_0)\right|^2 - \sum_{j=0}^{J-1}\frac{1}{2}|\Sigma^{-\frac{1}{2}}v_{j+1}|^2\right) \tag{2.7}$$

and hence ϑ_0 is a Gaussian measure, independent in each component v_j for $j = 0, \cdots, J$. We denote the mean by m and covariance by C, noting that $m = (m_0^T, 0^T, \cdots, 0^T)^T$ and that C is block diagonal with first block C_0 and the remainder all being Σ. Thus $\vartheta_0 = N(m, C)$. The basic idea of this method is to make proposals with the property that, if $\Psi \equiv 0$ so that the dynamics is Gaussian and with no time correlation, and if $h \equiv 0$ so that the data is totally uninformative, then the proposal would be accepted with probability

one. Making small incremental proposals of this type thereby incorporates the effects of $\Psi \neq 0$ and $h \neq 0$ through the accept-reject mechanism. We describe the details of how this works.

Recall the prior on the stochastic dynamics model with density $\rho_0(v) \propto \exp\big(-\mathsf{J}(v)\big)$ given by (1.25). It will be useful to rewrite π_0 as follows:

$$\pi_0(v) \propto \exp(-\mathsf{J}(v) + G(v)),$$

where

$$G(v) = \sum_{j=0}^{J-1} \left(\frac{1}{2} \left| \Sigma^{-\frac{1}{2}} \Psi(v_j) \right|^2 - \left\langle \Sigma^{-\frac{1}{2}} v_{j+1}, \Sigma^{-\frac{1}{2}} \Psi(v_j) \right\rangle \right). \tag{2.8}$$

We note that

$$\frac{\rho(v)}{\pi_0(v)} \propto \exp(-\Phi(v; y) - G(v)). \tag{2.9}$$

Recall the Gaussian measure $\vartheta_0 = N(m, C)$ defined via its pdf in (2.7). The pCN method is a variant of random walk type methods, based on the following proposal

$$w^{(k)} = m + \left(1 - \beta^2\right)^{\frac{1}{2}} \left(u^{(k-1)} - m \right) + \beta \iota^{(k-1)}, \tag{2.10}$$

$$\beta \in (0, 1), \quad \iota^{(k-1)} \sim N(0, C).$$

Here $\iota^{(k-1)}$ is assumed to be independent of $v^{(k-1)}$. This proposal preserves ϑ_0 and would be accepted all the time in the absence of data ($h \equiv 0$), and if $\Psi \equiv 0$. To see this preservation of ϑ_0 notice that if $v^{(k-1)} \sim \vartheta_0$ then $\mathbb{E} w^{(k)} = m$ and

$$\begin{aligned} \mathbb{E}\left(w^{(k)} - m\right) \otimes \left(w^{(k)} - m\right) &= \left(1 - \beta^2\right) \mathbb{E}\left(u^{(k-1)} - m\right) \otimes \left(u^{(k-1)} - m\right) + \beta^2 \mathbb{E} \iota^{(k-1)} \otimes \iota^{(k-1)} \\ &= (1 - \beta^2) C + \beta^2 C \\ &= C, \end{aligned}$$

where C is the covariance under ϑ_0. This shows that the proposal preserves ϑ_0 and, in fact that the resulting proposal satisfies detailed balance with respect to measure ϑ_0 with density π_0:

$$\frac{\pi_0(w) q(w, u)}{\pi_0(u) q(u, w)} = 1, \tag{2.11}$$

thus the Markov chain

$$u^{(k)} = m + \left(1 - \beta^2\right)^{\frac{1}{2}} \left(u^{(k-1)} - m \right) + \beta \iota^{(k-1)}$$

has ϑ_0 as an invariant measure. By use of (2.11) and (2.9) we deduce that the acceptance probability for this method is

$$\begin{aligned} a(u, w) &= 1 \wedge \frac{\rho(w) q(w, u)}{\rho(u) q(u, w)} \\ &= 1 \wedge \frac{\rho(w)/\pi_0(w)}{\rho(u)/\pi_0(u)} \\ &= 1 \wedge \exp(\Phi(u; y) - \Phi(w; y) + G(u) - G(w)). \end{aligned}$$

Recall that the proposal preserves the underlying Gaussian structure of the stochastic dynamics model; the accept-reject mechanism then introduces non-Gaussianity into the stochastic dynamics model, via G, and introduces the effect of the data, via Φ. By choosing β small, so that $w^{(k)}$ is close to $v^{(k-1)}$, we can make $a(v^{(k-1)}, w^{(k)})$ reasonably large and obtain a useable algorithm. This is illustrated in section 2.3.

Notice that, if $\Psi \equiv 0$ as assumed to define the measure ϑ_0, then the noise sequence $\{\xi_j\}_{j=1}^{\infty}$ is indentical with the signal sequence $\{v_j\}_{j=1}^{\infty}$. More generally, even if $\Psi \neq 0$, the noise sequence $\{\xi_j\}_{j=1}^{\infty}$, together with v_0, uniquely determines the signal sequence $\{v_j\}_{j=0}^{\infty}$. This motivates a different formulation of the

smoothing problem for stochastic dynamics where one views the noise sequence and initial condition as the unknown, rather than the signal sequence itself. Here we study the implication of this perspective for MCMC methodology, in the context of the pCN Method, leading to what we refer to as the **pCN Dynamics Sampler**, because the proposal samples from the dynamics as in the Independence Dynamics Sampler, while the step-size is controlled to ensure good acceptance probability as in the pCN Method.

To give details of this approach we notice that once v_0 and $\xi := \{\xi_j\}_{j\in\{0,\dots,J-1\}}$ are known then the sequence $\{v_j\}_{j\in\mathbb{J}_0}$ from (1.1) is determined and we formulate the smoothing problem in terms of $u := (v_0, \xi)$ rather than in terms of $\{v_j\}_{j\in\mathbb{J}_0}$. The data as given in (1.2) can be concatenated and written as

$$y = \mathcal{G}(u) + \eta$$

where $y = (y_1, \cdots, y_J)$ and $\eta = (\eta_1, \cdots, \eta_J)$ and $\mathcal{G}$ denotes the mapping from initial conditions v_0 and random forces ξ to the observation space. For simplicity we assume that the noise η is a Gaussian $N(0, \Gamma)$. Putting the prior ϑ_0, with Gaussian density specified in (2.7), on $u = (v_0, \xi)$ and conditioning on y gives a posterior μ on $u|y$ which has the Lebesgue density proportional to $e^{-\mathsf{I}(u;y)}$ with

$$\mathsf{I}(u;y) = \frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\left(y - \mathcal{G}(u)\right)\right|^2 + \frac{1}{2}\left|C_0^{-\frac{1}{2}}(v_0 - m_0)\right|^2 + \sum_{j=0}^{J-1}\frac{1}{2}\left|\Sigma^{-\frac{1}{2}}\xi_{j+1}\right|^2$$

where the first term comes from the likelihood $\mathbb{P}(y|u)$, which is non-Gaussian, and the second two terms come from the prior ϑ_0. Now using the pCN proposal (2.10) with $v^{(i)}$ replaced by $u^{(i)}$, one can see the acceptance probability is now given by

$$a(u, w) = 1 \wedge \exp\left(\frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\left(y - \mathcal{G}(u)\right)\right|^2 - \frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\left(y - \mathcal{G}(w)\right)\right|^2\right).$$

2.2. Variational Methods

Sampling the posterior using MCMC methods can be prohibitively expensive. Furthermore, if the probability is peaked at one, or a small number of places, then simply locating these peaks may be sufficient in an applied context. This is the basis for variational methods which seek to maximize the posterior probability, thereby locating such peaks. In this section we show how to characterize these peaks in the posterior probability, leading to problems in the calculus of variations. In contrast to the previous section on MCMC methods we do no attempt to overview the vast literature on optimization algorithms; references are given in the bibliographic notes of section 2.4.

and are hence termed **variational methods**. In the atmospheric sciences they are called **4DVAR** since they incorporate data over three spatial dimensions and one temporal dimension, in order to estimate the state. In Bayesian statistics the methods are called **MAP estimators**: maximum *a posteriori* estimators. First we consider the case of stochastic dynamics.

Theorem 2.3. *Consider the data assimilation problem for stochastic dynamics (1.1). The density $\rho(v) = \mathbb{P}(v|y)$ on $\mathbb{R}^N$ associated with the posterior probability μ, is maximized where $\mathsf{I}(v;y)$ given in (1.27) is minimized. Furthermore, if $B_\delta(z)$ denotes a ball in $\mathbb{R}^N$ of radius δ, centred at z, then if Ψ, h are continuous,*

$$\lim_{\delta\to 0}\frac{\mathbb{P}^\mu(B_\delta(z_1))}{\mathbb{P}^\mu(B_\delta(z_2))} = \exp(\mathsf{I}(z_2;y) - \mathsf{I}(z_1;y)).$$

Proof. Since

$$\begin{aligned}\mu(dv) &= \frac{1}{Z}\exp(-\mathsf{I}(v;y))dv\\ &= \rho(v)dv\end{aligned}$$

the first result is clear. For the second note that

$$\begin{aligned}\mathbb{P}^{\mu}(B_\delta(z)) &= \frac{1}{Z}\int_{|v-z|<\delta} \exp\big(-\mathsf{I}(v;y)\big)dv \\ &= \frac{1}{Z}\int_{|v-z|<\delta} \Big(\exp(-\mathsf{I}(z;y)) + e(\delta;v)\Big)dv\end{aligned}$$

where $e(\delta;v)\to 0$ as $\delta\to 0$, uniformly for $v\in B_\delta(z)$. This is because $\mathsf{I}(\cdot;y)$ inherits continuity from $\Psi(\cdot)$ and $h(\cdot)$. The result follows since $|\mathsf{I}(z;y)|$ is finite for every $z\in\mathbb{R}^N$. □

Remark 2.4. *The second statement in Theorem 2.3 may appear a little abstract. However, unlike the first statement, it can be generalised to infinite dimensions, as is required in continuous time. We state it this for precisely this reason.*

In applications to meteorology the variational method just described is known as **weak constraint 4DVAR**, and we denote this as **w4DVAR** in what follows. This generalizes the standard **4DVAR** method which may be derived in the limit $\Sigma\to 0$ so that the prior on the model dynamics (1.1) is deterministic, but with a random initial condition, as in (1.3). In this case the appropriate minimization is of $\mathsf{I}_0(v_0;y)$ given by (1.28). This has the advantage of being a lower dimensional minimization problem than weak constraint 4DVAR; however it is often a harder minimization problem, especially when the dynamics is chaotic. The following theorem may be proved similarly to Theorem 2.3.

Theorem 2.5. *Consider the data assimilation problem for deterministic dynamics (1.3). The density $\rho(v_0) = \mathbb{P}(v_0|y)$ on $\mathbb{R}^n$ associated with the posterior probability ν, is maximized where $\mathsf{I}_0(v_0;y)$ given in (1.28) is minimized. Furthermore, if $B_\delta(z)$ denotes a ball in $\mathbb{R}^n$ of radius δ, centred at z, then if Ψ, h are continuous,*

$$\lim_{\delta\to 0}\frac{\mathbb{P}^{\nu}(B_\delta(z_1))}{\mathbb{P}^{\nu}(B_\delta(z_2))} = \exp(\mathsf{I}_0(z_2;y) - \mathsf{I}_0(z_1;y)).$$

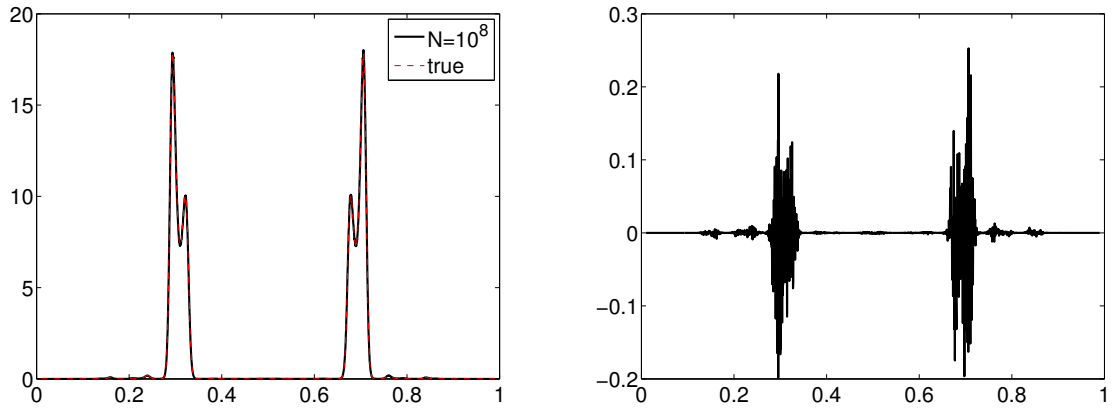
2.3. Illustrations

We describe a range of numerical experiments which illustrate the application of MCMC methods and variational methods to the smoothing problems which arise in both deterministic and stochastic dynamics.

The first illustration concerns use of the Random walk Metropolis algorithm to study the smoothing distribution for Example 1.4 in the case of determinstic dynamics where our aim is to find $\mathbb{P}(v_0|y)$. Recall Figure 13a which shows the true posterior pdf, found by plotting the formula given in Theorem 1.8. We now approximate the true posterior pdf by the MCMC method, using the same parameters, namely $m_0 = 0.5, C_0 = 0.01, \gamma = 0.2$ and $v_0^\dagger = 0.3$. In Figure 14 we compare the posterior pdf calculated by the Random walk Metropolis method (denoted by ρ^N, the histogram of the output of the Markov chain) with the true posterior pdf ρ. The two distributions are almost indistinguishable when plotted together in Figure 14a; in Figure 14b we plot their difference, which as we can see is small, relative to the true value.

We now turn to the use of MCMC methods to sample the smoothing pdf $\mathbb{P}(v|y)$ in the case of stochastic dynamics (1.1), using both the independence dynamics sampler and the pCN method. Before describing application of numerical methods we study the ergodicity of the independence dynamics sampler in a simple, but illustrative, setting. For simplicity assume that the observation operator h is bounded so that, for all $u\in\mathbb{R}^N$, $|h(u)|\le h_{\max}$. Then

$$\begin{aligned}\Phi(u;y) &\le \sum_{j=0}^{J-1}\big(|\Gamma^{-\frac{1}{2}}y_{j+1}|^2 + |\Gamma^{-\frac{1}{2}}h(u_{j+1})|2\big) \\ &\le |\Gamma^{-\frac{1}{2}}|^2\Big(\sum_{j=0}^{J-1}|y_{j+1}|^2 + Jh_{\max}^2\Big) \\ &\le |\Gamma^{-\frac{1}{2}}|^2\Big(|Y_J|^2 + Jh_{\max}^2\Big) \\ &=: \Phi_{\max}.\end{aligned}$$



(a) ρ^N vs ρ (b) $\rho^N - \rho$

FIG 14. *Comparison of the posterior for Example 1.4 for $r = 4$ using Random walk metropolis and equation (1.28) directly as in the second* MATLAB *program. We have used $J = 5$ $C_0 = 0.01, m_0 = 0.5$, $\gamma = 0.2$ and true initial condition $v_0 = 0.3$, see also* `p3.m` *in Section 4.2.1. We have used $N = 10^8$ samples from the MCMC algorithm.*

Since $\Phi \geq 0$ this shows that every proposed step is accepted with probability exceeding $e^{-\Phi_{\max}}$ and hence that, since proposals are made with the prior measure μ_0 describing the unobserved stochastic dynamics,

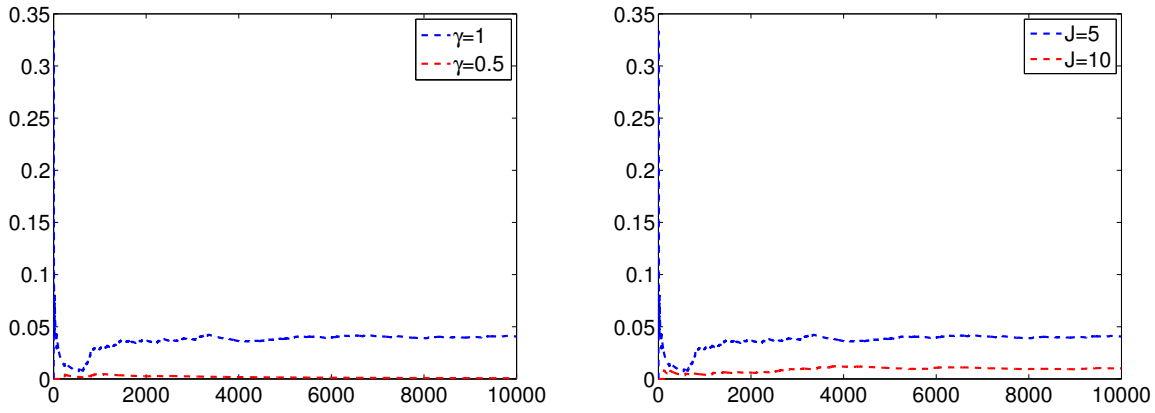
$$p(u, A) \geq e^{-\Phi_{\max}} \mu_0(A).$$

Thus Theorem 2.2 applies and, in particular, (2.4) and (2.5) hold, with $\varepsilon = e^{-\Phi_{\max}}$, under these assumptions. This positive result, also indicates the potential difficulties with the independence dynamics sampler. The independence sampler relies on draws from the prior matching the data well. Where the data set is large ($J \gg 1$) or the noise covariance small ($|\Gamma| \ll 1$) this will happen infrequently and the MCMC method will reject frequently and be inefficient. To illustrate this we consider application of the method to the Example 1.3, using the same parameters as in Figure 3; specifically we take $\alpha = 2.5$ and $\Sigma = \sigma^2 = 1$. We now sample the posterior distribution and then plot the resulting accept-reject ratio a for the independence dynamics sampler, employing different values of noise Γ and different sizes of the data set J. This is illustrated in Figure 15.

In addition, in Figure 16, we plot the output, and the running average of the output, projected into the first element of the vector $v^{(k)}$, the initial condition – remember that we are defining a Markov chain on $\mathbb{R}^{J+1}$ – for $K = 10^5$ steps. Figure 16a clearly exhibits the fact that there are many rejections caused by the low average acceptance probability. Figure 16b shows that the running average has not converged after 10^5 steps, indicating that the chains needs to be run for longer. If we run the Markov chain over $K = 10^8$ steps then we do get convergence. This is illustrated in Figure 17. In Figure 17a we see that the running average has converged to its limiting value when this many steps are used. In Figure 17b where we plot the marginal probability distribution for the first element of $v^{(k)}$, calculated from this converged Markov chain.

In order to get faster convergence when sampling the posterior distribution we turn to application of the pCN method. Unlike the independence dynamics sampler, this contains a tunable parameter which can vary the size of the proposals. In particular, the possibility of making small moves, with resultant higher acceptance probability, makes this a more flexible method than the independence dynamics sampler. In Figure 18 we show application of the pCN sampler, again considering Example 1.3 for $\alpha = 2.5$, $\Sigma = \sigma^2 = 1$ and $\Gamma = \gamma^2 = 1$, with $J = 10$, the same parameters used in Figure 17.

We now turn to variational methods; recall Theorems 2.3 and 2.5 in the stochastic and deterministic cases respectively. In Figure 19a we plot the MAP (4DVAR) estimator for our Example 1.4 for the case $r = 2$ choosing exactly the same parameters and data as for Figure 10a, $J = 10^2$. In this case the function $\mathsf{I}_0(\cdot; y)$ is quadratic and has a unique global minimum. A straightforward minimization routine will easily find this:



(a) $J = 5$, different values of γ

(b) $\gamma = 1$, different values of J

FIG 15. *Accept-reject probability of the independence sampler for Example 1.3 for* $\alpha = 2.5$, $\Sigma = \sigma^2 = 1$ *and* $\Gamma = \gamma^2$ *for different values of* γ *and* J.

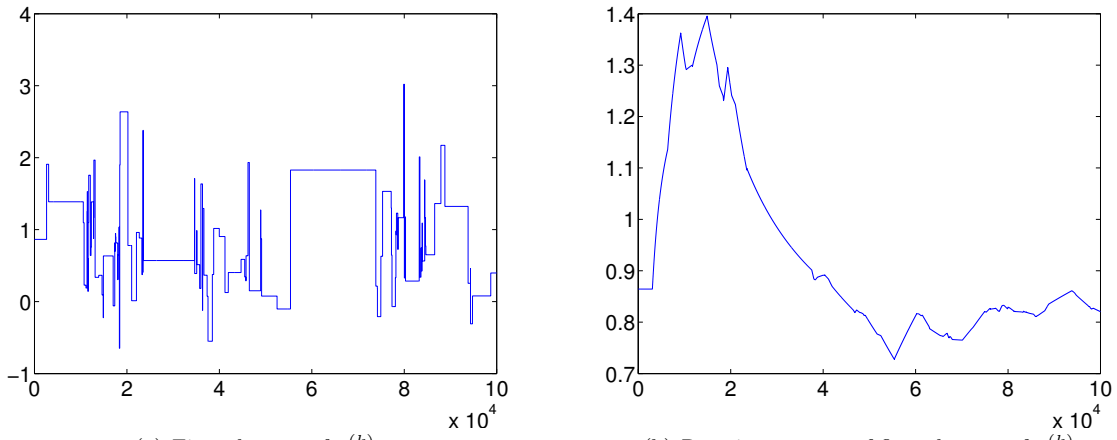
we employed standard MATLAB optimization software initialized at three different points. From all three starting points chosen the algorithm finds the correct global minimizer.

In Figure 19b we plot the MAP (4DVAR) estimator for our Example 1.4 for the case $r = 4$ choosing exactly the same parameters and data as for Figure 13. We again employ the MATLAB optimization routine initialized at three different points. Now the behaviour is very different from that observed in the preceding example. The value obtained for our MAP estimator depends crucially on the choice of initial condition in our minimization procedure. In particular, of the choices of starting point presented, only when we start from 0.2 are we able to find the global minimum of $\mathsf{I}_0(v_0; y)$. By Theorem 2.5 this global minimum corresponds to the maximum of the posterior distribution, we see that finding the MAP estimator is a difficult task for this problem. Starting with the other two initial conditions displayed we converge to one of the many local minima of $\mathsf{I}_0(v_0; y)$; these local minima are in fact regions of very low probability, as we can see in Figure 13a. This illustrates the care required when computing 4DVAR solutions in cases where the forward problem exhibits sensitivity to initial conditions.

Figure 20 shows application of the w4DVAR method, or MAP estimator given by Theorem 2.3, in the case of the Example 1.3 with parameters set at $J = 5, \gamma = \sigma = 0.1$. In contrast to the previous example, this is no longer a one-dimensional minimization problem: we are minimzing $\mathsf{I}(v; y)$ given by (1.27) over $v \in \mathbb{R}^6$, given the data $y \in \mathbb{R}^5$. The figure shows that there are at least 2 local minimizers for this problem, with v_1 closer to the truth than v_2, and with $I(v_1; y)$ considerably smaller that $I(v_2; y)$. However v_2 has a larger basin of attraction for the optimization software used: many initial conditions lead to v_2, while fewer lead to v_1. Furthermore, whilst be believe that v_1 is the global minimizer, it is difficult to state this with certainty, even for this relatively low-dimensional model.

2.4. Bibliographic Notes

- Section 2.1. Monte Carlo Markov Chain methods have a long history, initiated through the 1953 paper [MRTT53] and then generalized to an abstract formulation in the 1970 papr [Has70]. The subject is overviewed from an algorithmic point of view in [Liu01]. Theorem 2.2 is contained in [MT93], and that reference also contains many other converegence theorems for Markov chains; in particular we note that it is often possible to increase substantially the class of functions φ to which the theorem applies by means of Lyapunov function techniques, which control the tails of the probability distribution. The specific forms of the pCN-MCMC methods which we introduce here has been chosen to be particularly effective in high dimensions; see [CRSW13] for an overview, [BRSV08] for the introduction of pCN and



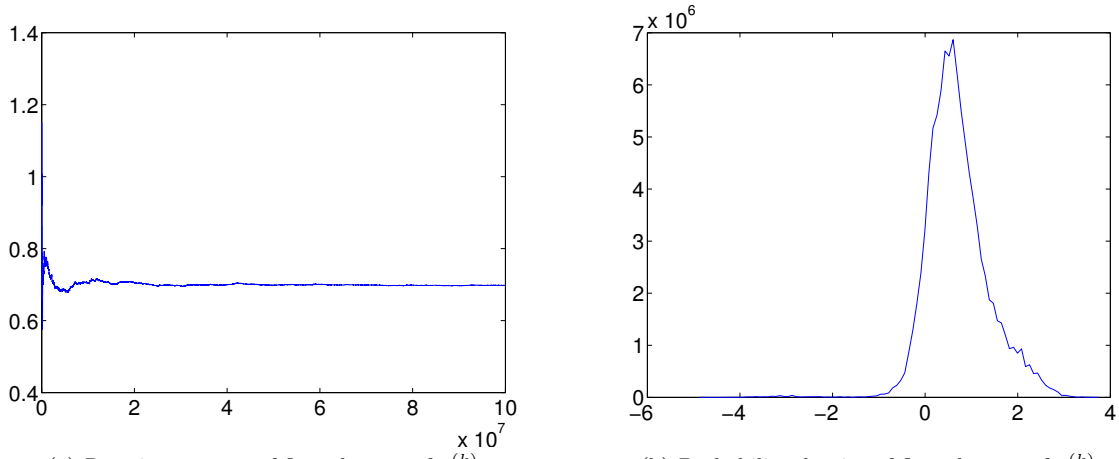
(a) First element of $v^{(k)}$ (b) Running average of first element of $v^{(k)}$

FIG 16. *Output and running average of the independence dynamics sampler after* $K = 10^5$ *steps, for Example 1.3 for* $\alpha = 2.5$, $\Sigma = \sigma^2 = 1$ *and* $\Gamma = \gamma^2 = 1$, *with* $J = 10$, *see also* `p4.m` *in Section 4.2.2.*

other methods for sampling probability measures in infinite dimensions, in the context of conditioned diffusions, and [CDS11a] for the application to a data assimilation problem.

The key point about pCN methods is that the proposal is reversible with respect to an underlying Gaussian measure. If $\Psi \neq 0$ then this Gaussian measure is far from the measure governing the actual dynamics. In contrast this Gaussian measure is *precisely* the measure governing the noise and initial condition, giving the pCN Dynamics Sampler a natural advantage. In particular, notice that the acceptance probability is now determined only by the model-data misfit for the pCN Dynamics Sampler, and is *not* having to account for incorporation of the dynamics as it does in the original pCN method; this typically improves the acceptance rate of the pCN Dynamical Sampler over the standard pCN method. Therefore, this method may be preferable, particularly in the case of unstable dynamics. The pCN Dynamics Sampler was introduced in [CDS11b] and further trialled in [HLS14]; it shows considerable promise.

- Section 2.2 Variational Methods, known as 4DVAR in the metereology community, have the distinction, when compared with the *ad hoc* non-Gaussian filters described in later sections, of being well-founded statistically: they correspond to the maximum *a posteriori* estimator for the fully Bayesian posterior distribution on model state given data. See [Zup97] and the references therein for a discussion of the applied context. See [CDRS09] for a more theoretical presentation. The European Centre for Medium-Range Weather Forecasts (ECMWF) weather prediction code, which is based on 4DVAR, is the best weather predictor, worldwide, according to a widely adopted metric by much the prediction skill of forecasts is measured. The subject of algorithms for optimization is vast and we have not attempted to cover it in these notes; the reader is directed to [NW99] for details.



(a) Running average of first element of $v^{(k)}$

(b) Probability density of first element of $v^{(k)}$

FIG 17. *Running average and probability density of the first element of $v^{(k)}$ for the independence dynamics sampler after $N = 10^8$ steps, for Example 1.3 for $\alpha = 2.5$, $\Sigma = \sigma^2 = 1$ and $\Gamma = \gamma^2$, with $\gamma = 1$ and $J = 10$, see also* `p4.m` *in Section 4.2.2.*

3. Discrete Time: Filtering Algorithms

In this chapter we describe various algorithms for the filtering problem. We start in section 3.1 with the Kalman filter which provides an exact algorithm to determine the filtering distribution for linear Gaussian problems; since the filtering distribution is Gaussian in this case, the algorithm comprises an iteration for the mean and covariance at each time. In section 3.2 we show how the idea of Kalman filtering may be used to combine dynamical model with data for nonlinear problems; in this case the posterior distribution is not Gaussian, but the algorithms proceed by invoking a Gaussian ansatz in the analysis step of the filter. This results in algorithms which do not provably approximate the true filtering distribution; in various forms they are, however, robust to use in high dimension. In section 3.3 we introduce the particle filter methodology which leads to provably accurate estimates of the true filtering distribution but which is, in its current forms, poorly behaved in high dimensions.

3.1. Linear Gaussian Problems: The Kalman Filter

This algorithm provides a sequential method for updating the filtering distribution $\mathbb{P}(v_j|Y_j)$ from time j to time $j+1$, when Ψ and h are linear maps. In this case the filtering distribution is Gaussian and it can be characterized entirely through the mean and covariance. We let

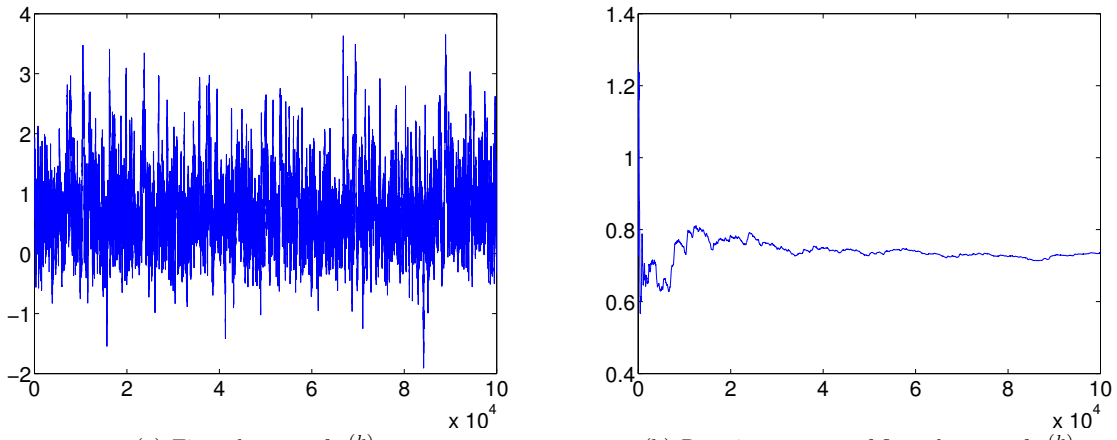
$$\Psi(v) = Mv, \ h(v) = Hv \tag{3.1}$$

for matrices $M \in \mathbb{R}^{n\times n}, H \in \mathbb{R}^{m\times n}$. We assume that $m \le n$ and $\mathrm{Rank}(H) = m$. We let (m_j, C_j) denote the mean and covariance of $v_j|Y_j$, noting that this random variable is Gaussian for each j since all maps are linear and all noise is Gaussian additive. We let $(\widehat{m}_{j+1}, \widehat{C}_{j+1})$ denote the mean and covariance of $v_{j+1}|Y_j$, noting that this too is a Gaussian random variable. We now derive the map $(m_j, C_j) \mapsto (m_{j+1}, C_{j+1})$, using the intermediate variables $(\widehat{m}_{j+1}, \widehat{C}_{j+1})$ so that we may compute the prediction and analysis steps separately.

Theorem 3.1. *Assume that $C_0, \Sigma > 0$. Then $C_j > 0$ for all $j \in \mathbb{Z}^+$ and*

$$C_{j+1}^{-1} = (MC_jM^T + \Sigma)^{-1} + H^T\Gamma^{-1}H \tag{3.2a}$$

$$C_{j+1}^{-1}m_{j+1} = (MC_jM^T + \Sigma)^{-1}Mm_j + H^T\Gamma^{-1}y_{j+1}. \tag{3.2b}$$



(a) First element of $v^{(k)}$ (b) Running average of first element of $v^{(k)}$

FIG 18. *Running average and probability density of the first element of $u^{(k)}$ for the pCN sampler after $N = 10^5$ steps, for Example 1.3 for $\alpha = 2.5$, $\Sigma = \sigma^2 = 1$ and $\Gamma = \gamma^2 = 1$, with $J = 10$, see also* `p5.m` *in Section 4.2.3.*

Proof. The prediction step is determined by (1.1) in the case $\Psi(\cdot) = M\cdot$:

$$v_{j+1} = Mv_j + \xi_j,\ \xi_j \sim N(0, \Sigma).$$

From this it is clear that

$$\mathbb{E}(v_{j+1}|Y_j) = \mathbb{E}(Mv_j|Y_j) + \mathbb{E}(\xi_j|Y_j).$$

Since ξ_j is independent of Y_j we have

$$\widehat{m}_{j+1} = Mm_j. \tag{3.3}$$

Similarly

$$\begin{aligned}\mathbb{E}\big((v_{j+1} - \widehat{m}_{j+1}) \otimes (v_{j+1} - \widehat{m}_{j+1})|Y_j\big) &= \mathbb{E}\big(M(v_j - m_j) \otimes M(v_j - m_j)|Y_j\big) + \mathbb{E}\big(\xi_j \otimes \xi_j|Y_j\big) \\ &\quad + \mathbb{E}\big(M(v_j - m_j) \otimes \xi_j|Y_j\big) + \mathbb{E}\big(\xi_j \otimes M(v_j - m_j)|Y_j\big).\end{aligned}$$

Again, since ξ_j is independent of Y_j and of v_j, we have

$$\begin{aligned}\widehat{C}_{j+1} &= M\mathbb{E}((v_j - m_j) \otimes (v_j - m_j)|Y_j)M^T + \Sigma \\ &= MC_jM^T + \Sigma.\end{aligned} \tag{3.4}$$

Now we consider the analysis step. By (1.30), which is just Bayes' rule, and using Gaussianity, we have

$$\exp\Big(-\frac{1}{2}\big|C_{j+1}^{-\frac{1}{2}}(v - m_{j+1})\big|^2\Big) \propto \exp\Big(-\frac{1}{2}\big|\Gamma^{-\frac{1}{2}}(y_{j+1} - Hv)\big|^2 - \frac{1}{2}\big|\widehat{C}_{j+1}^{-\frac{1}{2}}(v - \widehat{m}_{j+1})\big|^2\Big). \tag{3.5}$$

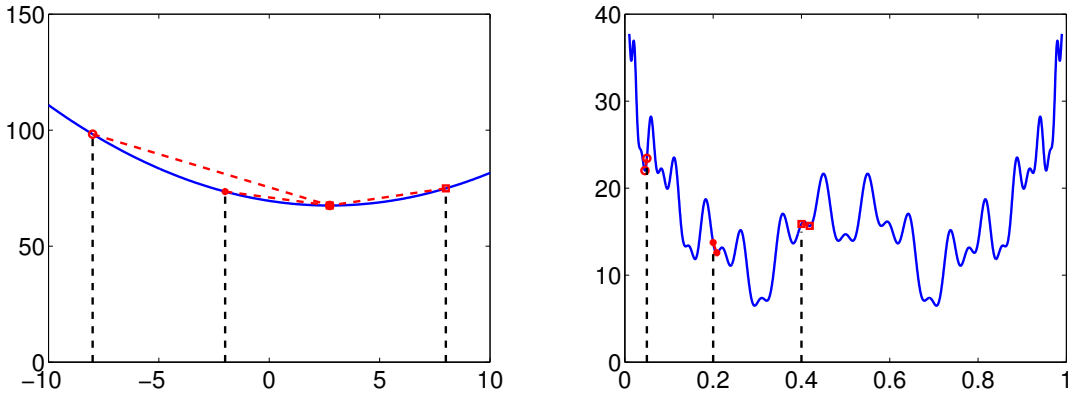
Equating quadratic terms in v gives

$$C_{j+1}^{-1} = \widehat{C}_{j+1}^{-1} + H^T\Gamma^{-1}H \tag{3.6}$$

and equating linear terms in v gives

$$C_{j+1}^{-1}m_{j+1} = \widehat{C}_{j+1}^{-1}\widehat{m}_{j+1} + H^T\Gamma^{-1}y_{j+1} \tag{3.7}$$

Substituting the expressions (3.3) and (3.4) for $(\widehat{m}_{j+1}, \widehat{C}_{j+1})$ gives the desired result. It remains to verify that $C_j > 0$ for all $j \in \mathbb{Z}^+$ so that the formal calculations above make sense. To this end we notice that, since Σ and $C_j > 0$ (inductive hypothesis, true for $j = 0$), we have $MC_jM^T + \Sigma > 0$ and hence $(MC_jM^T + \Sigma)^{-1} > 0$. Thus $C_{j+1}^{-1} > 0$ and hence $C_{j+1} > 0$. □



(a) $I(v_0; y)$ and its minima for Example 1.1

(b) $I(v_0; y)$ and its minima for Example 1.4

FIG 19. *Finding local minima of $I(v_0; y)$ for Examples 1.1 and 1.4. The values and the data used are the same as for Figures 10a and 13b. $(\circ, \star, \square)$ denote three different initial conditions for the starting the minimization process. $(-8, -2, 8)$ for Example 1.1 and $(0.05, 0.2, 0.4)$ for Example 1.4.*

Corollary 3.2. *Under the assumptions of Theorem 3.1, the formulae for the Kalman filter given there may be rewritten as follows:*

$$\begin{aligned} d_{j+1} &= y_{j+1} - H\widehat{m}_{j+1} \\ S_{j+1} &= H\widehat{C}_{j+1}H^T + \Gamma \\ K_{j+1} &= \widehat{C}_{j+1}H^T S_{j+1}^{-1} \\ m_{j+1} &= \widehat{m}_{j+1} + K_{j+1}d_{j+1} \\ C_{j+1} &= (I - K_{j+1}H)\widehat{C}_{j+1}, \end{aligned}$$

with $(\widehat{m}_{j+1}, \widehat{C}_{j+1})$ given in (3.3), (3.4).

Proof. By (3.6) we have

$$C_{j+1}^{-1} = \widehat{C}_{j+1}^{-1} + H^T\Gamma^{-1}H$$

and application of Lemma 3.4 below gives

$$\begin{aligned} C_{j+1} &= \widehat{C}_{j+1} - \widehat{C}_{j+1}H^T(\Gamma + H\widehat{C}_{j+1}H^T)^{-1}H\widehat{C}_{j+1} \\ &= \Big(I - \widehat{C}_{j+1}H^T(\Gamma + H\widehat{C}_{j+1}H^T)^{-1}H\Big)\widehat{C}_{j+1} \\ &= (I - \widehat{C}_{j+1}H^T S_{j+1}^{-1}H)\widehat{C}_{j+1} \\ &= (I - K_{j+1}H)\widehat{C}_{j+1} \end{aligned}$$

as required. Then the identity (3.7) gives

$$\begin{aligned} m_{j+1} &= C_{j+1}\widehat{C}_{j+1}^{-1}\widehat{m}_{j+1} + C_{j+1}H^T\Gamma^{-1}y_{j+1} \\ &= (I - K_{j+1}H)\widehat{m}_{j+1} + C_{j+1}H^T\Gamma^{-1}y_{j+1} \end{aligned} \tag{3.8}$$

Now note that, again by (3.6),

$$C_{j+1}(\widehat{C}_{j+1}^{-1} + H^T\Gamma^{-1}H) = I$$

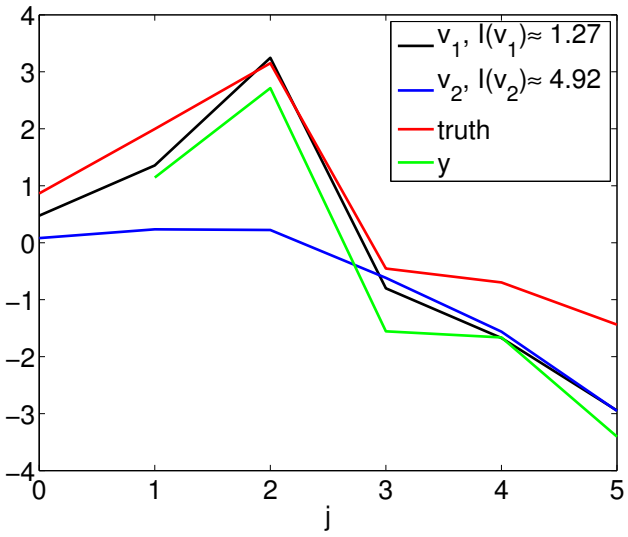


FIG 20. *Weak constraint 4DVAR for $J = 5, \gamma = \sigma = 0.1$, illustrating two local minimizers v_1 and v_2, see also* `p6.m` *in Section 4.2.4.*

so that

$$\begin{aligned} C_{j+1}H^T\Gamma^{-1}H &= I - C_{j+1}\widehat{C}_{j+1}^{-1} \\ &= I - (I - K_{j+1}H) \\ &= K_{j+1}H. \end{aligned}$$

Since H has rank m we deduce that

$$C_{j+1}H^T\Gamma^{-1} = K_{j+1}.$$

Hence (3.8) gives

$$m_{j+1} = (I - K_{j+1}H)\widehat{m}_{j+1} + K_{j+1}y_{j+1} = \widehat{m}_{j+1} + K_{j+1}d_{j+1}$$

as required. □

Remark 3.3. *The key difference between the Kalman update formulae in Theorem 3.1 and in Corollary 3.2 is that, in the former matrix inversion takes place in the state space, with dimension n, whilst in the latter matrix inversion takes place in the data space, with dimension m. In many applications $m \ll n$, as the observed subspace dimension is much less than the state space dimension, and thus the formulation in Corollary 3.2 is frequently employed in practice.*

The following matrix identity was used to derive the formulation of the Kalman Filter in which inversion takes place in the data space.

Lemma 3.4. Woodbury Matrix Identity *Let $A \in \mathbb{R}^{p\times p}, U \in \mathbb{R}^{p\times q}, C \in \mathbb{R}^{q\times q}$ and $V \in \mathbb{R}^{q\times p}$. If A and C are positive then $A + UCV$ is invertible and*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\Big(C^{-1} + VA^{-1}U\Big)^{-1}VA^{-1}.$$

3.2. Approximate Gaussian Filters

Here we introduce a family of methods, based on invoking a Gaussian ansatz, which may be applied to non-Gaussian problems. The update equation for the Kalman filter mean, (3.7), can be derived by minimizing

the following model/data compromise functional, derived from (3.5),

$$J(v) := \frac{1}{2}|\Gamma^{-\frac{1}{2}}(y_{j+1} - Hv)|^2 + \frac{1}{2}|\widehat{C}_{j+1}^{-\frac{1}{2}}(v - \widehat{m}_{j+1})|^2. \tag{3.9}$$

Whilst the Kalman filter itself is restricted to linear, Gaussian problems, the formulation via minimization generalizes to nonlinear problems. Noting that $\widehat{m}_{j+1} = Mm_j$, and that $\Psi(\cdot) = M\cdot$, $h(\cdot) = H\cdot$, we see that a natural generalization of (3.9) to the nonlinear case is to minimize

$$J(v) := \frac{1}{2}|\Gamma^{-\frac{1}{2}}(y_{j+1} - h(v))|^2 + \frac{1}{2}|\widehat{C}_{j+1}^{-\frac{1}{2}}(v - \Psi(m_j))|^2$$

and then to set

$$m_{j+1} = \arg\min_v J(v).$$

For simplicity we consider the case where observations are linear and $h(v) = Hv$ leading to the update algorithm $m_j \mapsto m_{j+1}$ defined by

$$\begin{aligned} J(v) &= \frac{1}{2}|\Gamma^{-\frac{1}{2}}(y_{j+1} - Hv)|^2 + \frac{1}{2}|\widehat{C}_{j+1}^{-\frac{1}{2}}(v - \Psi(m_j))|^2 \\ m_{j+1} &= \arg\min_v J(v). \end{aligned} \tag{3.10}$$

By the arguments used in Corollary 3.2 we deduce the following update formulae:

$$m_{j+1} = (I - K_{j+1}H)\Psi(m_j) + K_{j+1}y_{j+1} \tag{3.11a}$$

$$K_{j+1} = \widehat{C}_{j+1}H^T S_{j+1}^{-1} \tag{3.11b}$$

$$S_{j+1} = H\widehat{C}_{j+1}H^T + \Gamma \tag{3.11c}$$

The next three sections each correspond to algorithms derived in this way, namely by minimizing $J(v)$, but corresponding to different choices of the model covariance $\widehat{C}_{j+1}$. We sometimes refer to these three algorithms collectively as **approximate Gaussian filters**. This is because they invoke a Gaussian approximation when updating the estimate of the signal via (3.10). Specifically this update is the correct update for the mean if the assumption that $\mathbb{P}(v_{j+1}|Y_j) = N\big(\Psi(m_j), \widehat{C}_{j+1}\big)$ is invoked for the prediction step. In general the approximation implied by this assumption will not be a good one and this can invalidate the statistical accuracy of the resulting algorithms. However the resulting algorithms may still have desirable properties in terms of signal estimation; we will demonstrate that this is indeed so.

3.2.1. 3DVAR

This algorithm is derived from (3.11) by simply fixing the model covariance $\widehat{C}_{j+1} \equiv \widehat{C}$ for all j. Thus we obtain

$$m_{j+1} = (I - KH)\Psi(m_j) + Ky_{j+1} \tag{3.12a}$$

$$K = \widehat{C}H^T S^{-1}, \quad S = H\widehat{C}H^T + \Gamma \tag{3.12b}$$

We now describe two methodologies which generalize 3DVAR by employing model covariances which evolve from step j to step $j+1$: the extended and ensemble Kalman filters. We present both methods in basic form but conclude the section with some discussion of methods widely used in practice to improve their practical performance.

3.2.2. Extended Kalman Filter

The idea of the extended Kalman filter (ExKF) is to propagate covariances according to the linearization of (1.1), and propagate the mean, using (1.1). Thus we obtain, from modification of Corollary 3.2 and (3.3),

(3.4)

$$\text{Prediction} \begin{cases} \widehat{m}_{j+1} &= \Psi(m_j) \\ \widehat{C}_{j+1} &= D\Psi(m_j)C_j D\Psi(m_j)^T + \Sigma \end{cases}$$

$$\text{Analysis} \begin{cases} S_{j+1} &= H\widehat{C}_{j+1}H^T + \Gamma \\ K_{j+1} &= \widehat{C}_{j+1}H^T S_{j+1}^{-1} \\ m_{j+1} &= (I - K_{j+1}H)\widehat{m}_{j+1} + K_{j+1}y_{j+1} \\ C_{j+1} &= (I - K_{j+1}H)\widehat{C}_{j+1} \end{cases}$$

3.2.3. Ensemble Kalman Filter

The ensemble Kalman filter (EnKF) generalizes the idea of approximate Gaussian filters in a significant way: rather than using the minimization procedure (3.10) to update a *single* estimate of the mean, it is used to generate an *ensemble* of particles which all satisfy the model/data compromise inherent in the minimization; the mean and covariance used in the minimization are estimated using this ensemble, thereby adding further coupling to the particles, in addition to that introduced by the data.

The EnKF is executed in a variety of ways and we start by describing one of these, the perturbed observation EnKF:

$$\text{Prediction} \begin{cases} \widehat{v}_{j+1}^{(n)} &= \Psi(v_j^{(n)}) + \xi_j^{(n)},\ n = 1, ..., N \\ \widehat{m}_{j+1} &= \frac{1}{N}\sum_{n=1}^{N} \widehat{v}_{j+1}^{(n)}, \\ \widehat{C}_{j+1} &= \frac{1}{N-1}\sum_{n=1}^{N}(\widehat{v}_{j+1}^{(n)} - \widehat{m}_{j+1})(\widehat{v}_{j+1}^{(n)} - \widehat{m}_{j+1})^T \end{cases}$$

$$\text{Analysis} \begin{cases} S_{j+1} &= H\widehat{C}_{j+1}H^T + \Gamma \\ K_{j+1} &= \widehat{C}_{j+1}H^T S_{j+1}^{-1} \\ v_{j+1}^{(n)} &= (I - K_{j+1}H)\widehat{v}_{j+1}^{(n)} + K_{j+1}y_{j+1}^{(n)} \\ y_{j+1}^{(n)} &= y_{j+1} + \eta_{j+1}^{(n)} \end{cases}$$

Here $\eta_j^{(n)}$ are i.i.d. draws from $N(0, \Gamma)$ and $\xi_j^{(n)}$ are i.i.d. draws from $N(0, \Sigma)$. Perturbed observation refers to the fact that each particle sees an observation perturbed by an independent draw from $N(0, \Gamma)$. This procedure gives the Kalman Filter in the linear case in the limit of infinite ensemble. Even though the algorithm is motivated through our general approximate Gaussian filters framework, notice that the ensemble is not prescribed to be Gaussian. Indeed it evolves under the full nonlinear dynamics in the prediction step. This fact, together with the fact that covariance matrices are not propagated explicitly, other than through the empirical properties of the ensemble, has made the algorithm very appealing to practitioners.

3.2.4. Square Root Ensemble Kalman Filters

We now describe another popular variant of the EnKF. The idea of this variant is to define the analysis step in such a way that an ensemble of particles is produced whose empirical covariance *exactly* satisfies the Kalman identity

$$C_{j+1} = (I - K_{j+1}H)\widehat{C}_{j+1} \tag{3.13}$$

which relates the covariances in the analysis step to those in the prediction step. This is done by mapping the mean of the predicted ensemble according to the standard Kalman update, and introducing a linear transformation of the differences between the particle positions and their mean to enforce (3.13). Doing so eliminates a sampling error inherent in the perturbed observation approach. The resulting algorithm has the following form:

$$\text{Prediction} \begin{cases} \widehat{v}_{j+1}^{(n)} &= \Psi(v_j^{(n)}) + \xi_j^{(n)},\ n = 1, ..., N \\ \widehat{m}_{j+1} &= \frac{1}{N}\sum_{n=1}^N \widehat{v}_{j+1}^{(n)}, \\ \widehat{C}_{j+1} &= \frac{1}{N-1}\sum_{n=1}^N (\widehat{v}_{j+1}^{(n)} - \widehat{m}_{j+1})(\widehat{v}_{j+1}^{(n)} - \widehat{m}_{j+1})^T \end{cases}$$

$$\text{Analysis} \begin{cases} S_{j+1} &= H\widehat{C}_{j+1}H^T + \Gamma \\ K_{j+1} &= \widehat{C}_{j+1}H^T S_{j+1}^{-1} \\ m_{j+1} &= (I - K_{j+1}H)\widehat{m}_{j+1} + K_{j+1}y_{j+1} \\ v_{j+1}^{(n)} &= m_{j+1} + \zeta_{j+1}^{(n)} \end{cases}$$

Here the $\{\zeta_{j+1}^{(n)}\}_{n=1}^N$ are designed to have sample covariance $C_{j+1} = (I - K_{j+1}H)\widehat{C}_{j+1}$. There are several ways to do this and we now describe one of them, refered to as the ensemble transform Kalman filter (ETKF).

If we define

$$\widehat{X}_{j+1} = \frac{1}{\sqrt{N-1}}\left[\widehat{v}_{j+1}^{(1)} - \widehat{m}_{j+1}, \ldots, \widehat{v}_{j+1}^{(N)} - \widehat{m}_{j+1}\right]$$

then $\widehat{C}_{j+1} = \widehat{X}_{j+1}\widehat{X}_{j+1}^T$. We now seek a transformation T_{j+1} so that, if $X_{j+1} = \widehat{X}_{j+1}T_{j+1}^{\frac{1}{2}}$, then

$$C_{j+1} := X_{j+1}X_{j+1}^T = (I - K_{j+1}H)\widehat{C}_{j+1}. \tag{3.14}$$

Note that the X_{j+1} (resp. the $\widehat{X}_{j+1}$) correspond to Cholesky factors of the matrices C_{j+1} (resp. $\widehat{C}_{j+1}$) respectively. We may now define the $\{\zeta_{j+1}^{(n)}\}_{n=1}^N$ by

$$X_{j+1} = \frac{1}{\sqrt{N-1}}\left[\zeta_{j+1}^{(1)}, \ldots, \zeta_{j+1}^{(N)}\right].$$

We now demonstrate how to find an appropriate transformation T_{j+1}. We assume that T_{j+1} is symmetric and positive-definite and the standard matrix square-root is employed. Choosing

$$T_{j+1} = \left[I_K + (H\widehat{X}_{j+1})^T\Gamma^{-1}(H\widehat{X}_{j+1})\right]^{-1}$$

we see that

$$\begin{aligned} X_{j+1}X_{j+1}^T &= \widehat{X}_{j+1}T_{j+1}\widehat{X}_{j+1}^T \\ &= \widehat{X}_{j+1}\left[I_K + (H\widehat{X}_{j+1})^T\Gamma^{-1}(H\widehat{X}_{j+1})\right]^{-1}\widehat{X}_{j+1}^T \\ &= \widehat{X}_{j+1}\left\{I_K - (H\widehat{X}_{j+1})^T\left[(H\widehat{X}_{j+1})(H\widehat{X}_{j+1})^T + \Gamma\right]^{-1}(H\widehat{X}_{j+1})\right\}\widehat{X}_{j+1}^T \\ &= (I - K_{j+1}H)\widehat{C}_{j+1} \end{aligned}$$

as required, where the transformation between the second and third lines is justified by Lemma 3.4. It is important to ensure that **1**, the vector of all ones, is an eigenvector of the transformation T_{j+1}, and hence of $T_{j+1}^{\frac{1}{2}}$, so that the mean of the ensemble is preserved. This is guaranteed by T_{j+1} as defined.

3.2.5. Tuning Non-Gaussian Filters

In practical implementations, especially for high dimensional problems, the basic forms of the ExKF and EnKF as described here are prone to poor behaviour. In Examples 3.12 and 3.13 we have already shown the role of **variance inflation** for 3DVAR and this type of approach is also fruitfully used within ExKF and EnKF. A basic version of variance inflation is to replace the estimate $\widehat{C}_{j+1}$ in (3.11) by $\epsilon^2 I + \widehat{C}_{j+1}$; doing so prevents $\widehat{C}_{j+1}$ from having a null-space and can be particularly important when the EnKF is used in high dimensional systems where the number of ensemble members, N, is always less than the dimension

d of the state space. In this situation $\widehat{C}_{j+1}$ necessarily has a null-space of dimension at least $d-N$. It can also be important for the ExKF where the evolving dynamics can lead, asymptotically in j, to degenerate $\widehat{C}_{j+1}$ with non-trivial null-space. Notice also that this form a various inflation can be thought of as using 3DVAR-like covariance updates, in the directions not described by the ensemble covariance. This can be beneficial in terms of the ideas underlying Theorem 3.10 where the key idea is that K close to the identity can help ameliorate growth in the underlying dynamics. This may also be achieved by replacing the estimate $\widehat{C}_{j+1}$ in (3.11) by $(1+\epsilon^2)\widehat{C}_{j+1}$. This is another commonly used inflation tactic; not, however, that it lacks the benefit of rank correction.

Another methodology which is important for practical implementation of the EnKF is **localization**. This is used to reduce unwanted correlations in $\widehat{C}_j$ between points which are separated by large distances in space. The underlying assumption is that the correlation between points decays proportionally to their distance from one another, and as such is increasingly corrupted by the sample error in ensemble methods. The sample covariance is hence modified to remove correlations between points separated by large distances in space. A typical convolution kernel to achieve this would be

$$\rho_{ik} = \exp\{-|i-k|^2\}.$$

Localization can have the further benefit of increasing rank, as for the first type of variance inflation described above. If used in addition to inflation of the second type described above one accomplishes both rank correction as well as stability, while eliminating assumed spurious correlations. It is worth noting that in many systems which are not governed by PDE, or indeed even in those governed by PDE but in a spectral representation [4], there are not such decay of correlations and localization may eliminate useful information. In such a case rank correction is still desirable, but the first type of inflation may be the preferable approach to accomplish this.

3.3. The Particle Filter

In this section we introduce an important class of filtering methods known as *particle filters*. In contrast to the filters introduced in the preceding section, the particle filter can be *proved* to reproduce the true posterior filtering distribution in the large particle limit and, as such, has a priveledged places amongst all the filters introduced in this book. We will describe the method in its basic form – the *bootstrap filter* – and then give a proof of convergence. It is important to appreciate that the form of particle filter introduced here is far from state-of-the-art and that far more sophisticated versions are used in practical applications. Nonetheless, despite this sophisticiation, particle filters do not perform well in applications such as those arising in geophysical applications of data assimilation, because the data in those applications places very strong constraints on particle locations, making efficient algorithms very hard to design. It is for this reason that we have introduced particle filters after the approximate Gaussian filters introduced in the preceding section. The filters in the preceding section tend to be more robust to data specifications. However they do all rely on the invocation of ad hoc Gaussian assumptions in their derivation and hence do not provably produce the correct posterior filtering distribution, notwithstanding their ability, in partially observed small noise scenarios, to correctly identify the signal itself, as in Theorem 3.10. Because it can provably reproduce the correct filtering distribution, the particle filter thus plays an important role, conceptually, even though it is not, in current form, a practical algorithm in geophysical applications. With further innovations it may, in time, form the basis for practical algorithms.

3.3.1. Approximation by Dirac Measures

All probability measures which possess density with respect to Lebesgue measure can be approximated by a finite convex combination of Dirac probability measures; an example of this is the **Monte Carlo sampling** idea that we decribed at the start of Chapter 2, and also underlies the ensemble Kalman filter of 3.2.3. In

[4] In the case of spectral representation, one may expect similar effect by transforming the operator ρ into the given spectral basis.

practice the idea of approximation by a convex set of probability measures requires the determination of the locations and weights associated with these Dirac measures. Particle filters are sequential algorithms which use this idea to approximate the true filtering distribution $\mathbb{P}(v_j|Y_j)$.

Recall the prediction and analysis formulae from (1.29) and (1.30) which can be summarized as

$$\mathbb{P}(v_{j+1}|Y_j) = \int_{\mathbb{R}^n} \mathbb{P}(v_{j+1}|v_j)\mathbb{P}(v_j|Y_j)dv_j, \tag{3.15a}$$

$$\mathbb{P}(v_{j+1}|Y_{j+1}) = \frac{\mathbb{P}(y_{j+1}|v_{j+1})\mathbb{P}(v_{j+1}|Y_j)}{\mathbb{P}(y_{j+1}|Y_j)}. \tag{3.15b}$$

Now let μ_j be the probability measure on $\mathbb{R}^n$ corresponding to the density $\mathbb{P}(v_j|Y_j)$ and $\widehat{\mu}_{j+1}$ be the probability measure on $\mathbb{R}^n$ corresponding to the density $\mathbb{P}(v_{j+1}|Y_j)$. Then we may write (3.15) as

$$\widehat{\mu}_{j+1}(\cdot) = (P\mu_j)(\cdot) := \int_{\mathbb{R}^n} \mathbb{P}(\cdot|v_j)\mu_j(dv_j) \tag{3.16a}$$

$$\frac{d\mu_{j+1}}{d\widehat{\mu}_{j+1}}(v_{j+1}) = \frac{\mathbb{P}(y_{j+1}|v_{j+1})}{\mathbb{P}(y_{j+1}|Y_j)}. \tag{3.16b}$$

The formula (3.16b) for the *density* of μ_{j+1} with respect to that of $\widehat{\mu}_{j+1}$ has a straightforward interpretation: the righthand-side quantifies how to reweight expectations under $\widehat{\mu}_{j+1}$ so that they become expectations under μ_{j+1}. We write the update formulae this way because it makes sense in the absence of Lebesgue densities; in particular we will want to apply it in situations where Dirac masses are used.

Particle filters proceed by finding an N-particle Dirac measure approximation of the form

$$\mu_j \approx \mu_j^N := \sum_{n=1}^N w_j^{(n)} \delta_{v_j^{(n)}}. \tag{3.17}$$

The approximate distribution is completely defined by particle positions $v_j^{(n)}$ and weights $w_j^{(n)}$ respectively. Thus the objective of the method is to find an update rule for $\{v_j^{(n)}, w_j^{(n)}\}_{n=1}^N \mapsto \{v_{j+1}^{(n)}, w_{j+1}^{(n)}\}_{n=1}^N$. The weights must sum one. This may be achieved an application of Bayesian probability, combined with judicious approximation by Dirac measures.

3.3.2. Bootstrap Filter (Sequential Importance Resampling)

The simplest particle filter is as follows. First each particle $v_j^{(n)}$ is updated by proposing a new candidate particle $\widehat{v}_{j+1}^{(n)}$ according to the Markov kernel P. This creates an approximation to $\widehat{\mu}_{j+1}$ which we can think of as a prior distribution. Secondly each new particle is re-weighted according to the desired distribution μ_{j+1} given by (3.16b); this corresponds to applying Bayes' formula (1.24) to obtain the posterior distribution. The required calculations are very straightforward because of the assumed form of the measures as sums of Dirac's, as we now explain.

Note from (3.16a) that, making the particle approximation (3.17) for μ_j,

$$\widehat{\mu}_{j+1}(\cdot) \approx \sum_{n=1}^N w_j^{(n)} \mathbb{P}(\cdot|v_j^{(n)}). \tag{3.18}$$

This distribution is not, in general, of Dirac form (although it will be in the case of deterministic dynamics). However we may make a Dirac approximation as follows: if we choose $\widehat{v}_{j+1}^{(n)}$ as a sample from $P(\widehat{v}_j^{(n)}, \cdot)$ the Markov kernel with transition density $\mathbb{P}(\widehat{v}_{j+1}^{(n)}|\widehat{v}_j^{(n)})$ then a natural approximation to $\widehat{\mu}_{j+1}$ is the measure

$$\widehat{\mu}_{j+1}^N = \sum_{n=1}^N w_j^{(n)} \delta_{\widehat{v}_{j+1}^{(n)}}. \tag{3.19}$$

We now apply Bayes' formula in the form (3.16b). Using an approximation proportional to (3.19) for $\widehat{\mu}_{j+1}$ we obtain

$$\mu_{j+1} \approx \mu_{j+1}^N := \sum_{n=1}^N w_{j+1}^{(n)} \delta_{\widehat{v}_{j+1}^{(n)}}. \tag{3.20}$$

where

$$w_{j+1}^{(n)} = \widehat{w}_{j+1}^{(n)} \Big\backslash \left(\sum_{n=1}^N \widehat{w}_{j+1}^{(n)} \right), \qquad \widehat{w}_{j+1}^{(n)} = w_j^{(n)} \mathbb{P}\left(y_{j+1} | \widehat{v}_{j+1}^{(n)} \right). \tag{3.21}$$

The first equation in the preceding is required for normalization.

We could simply set $v_{j+1}^{(n)} = \widehat{v}_{j+1}^{(n)}$ at this point and the algorithm resulting from doing so is the basic form of *sequential importance sampling*. However, in practice, some weights become very small and for this reason it is desirable to add an additional resampling step where the $\{v_{j+1}^{(n)}\}$ are drawn from (3.20) and then all given equal weights, yielding the sequential importance *re*-sampling algorithm. Because the initial measure $\mathbb{P}(v_0)$ is not in Dirac form it is convenient to place this resampling step at the *start* of each iteration, rather than at the end as we have presented here, as this naturally introduces a particle approximation of the initial measure. This reordering makes no difference to the iteration we have described and doing so results in the following algorithm, where Y_0 denotes the empty vector (no observations at the start):

1. Set $j = 0$ and $\mu_0^N(v_0)dv_0 = \mathbb{P}(v_0)dv_0$.
2. Draw $v_j^{(n)} \sim \mu_j^N$, $n = 1, \ldots, N$.
3. Set $w_j^{(n)} = 1/N$, $n = 1, \ldots, N$.
4. Draw $\widehat{v}_{j+1}^{(n)} \sim \mathbb{P}(v_j^{(n)}|\cdot)$.
5. Define $w_{j+1}^{(n)}$ by (3.21)and μ_{j+1}^N by (3.20).
6. $j + 1 \to j$.
7. Go to step 2.

This algorithm is conceptually intuitive, proposing that each particle moves according to the dynamics of the underlying model itself, and is then reweighted according to the likelihood of the proposed particle, i.e. according to the data. The resulting method is termed the **bootstrap filter**. We will comment on important improvements to this basic algorithm in the the following section and in the bibliographic notes. Here we prove convergence of this basic method, as the number of particles goes to infinity, thereby demonstrating the potential power of the bootstrap filter and more sophisticated variants on it.

It is instructive to write the algorithm in the following form. We build the algorithm from the following three operations on measures: P corresponding to moving a point currently at v according to the Markov kernel $\mathbb{P}(\cdot|v)$ describing the dynamics given by (1.1a); S^N denotes sampling N i.i.d. points from a measure and approximating that measure by an equally weighted sum of Dirac's at the sample points; and C_j denotes the application of Bayes' formula with likelihood proportional to

$$g_j(\cdot) = \mathbb{P}(y_{j+1}|\cdot).$$

The preceding algorithm may be written as

$$\mu_{j+1}^N = C_j S^N P \mu_j^N. \tag{3.22}$$

There is a slight trickery here in writing application of the sampling S^N *after* application of P, but some reflection shows that this is well-justified: applying P followed by S^N can be shown, by first conditioning on the initial point and sampling with respect to P, and then sampling over the distribution of the initial point, to be the algorithm as defined. The true filtering distribution simply satisfies the iteration

$$\mu_{j+1} = C_j P \mu_j. \tag{3.23}$$

Thus analyzing the particle filter requires estimation of the error induced by application of S^N (the *resampling error*) together with estimation of the rate of accumulation of this error in time.

The operators C_j, P and S^N map the space $\mathcal{P}(\mathbb{R}^n)$ of probability measures on $\mathbb{R}^n$ into itself according to the following:

$$(C_j\mu)(dv) = \frac{g_j(v)\mu(dv)}{\int_{\mathbb{R}^n} g_j(v)\mu(dv)}, \tag{3.24a}$$

$$(P\mu)(dv) = \int_{\mathbb{R}^n} \mathbb{P}(v', dv)\mu(dv'), \tag{3.24b}$$

$$(S^N\mu)(dv) = \frac{1}{N}\sum_{n=1}^{N} \delta_{v^{(n)}}(dv), \quad v^{(n)} \sim \mu \quad \text{i.i.d..} \tag{3.24c}$$

Let $\mu = \mu_\omega$ denote, for each ω, an element of $\mathcal{P}(\mathbb{R}^n)$. If we then assume that ω is a random variable, and let $\mathbb{E}^\omega$ denote expectation over ω, then we may define a "root mean square" distance $d(\cdot,\cdot)$ between two random probability measures μ_ω, ν_ω, as follows:

$$d(\mu,\nu) = \sup_{|f|\leq 1}\sqrt{\mathbb{E}^\omega|\mu(f) - \nu(f)|^2},$$

where we have used the convention that $\mu(f) = \int_{\mathbb{R}^n} f(v)\mu(dv)$ for measurable $f : \mathbb{R}^n \to \mathbb{R}$, and similar for ν. This distance does indeed generate a metric and, in particular, satisfies the triangle inequality.

Theorem 3.5. *We assume in the following that there exists $\kappa \in (0,1]$ such that for all $v \in \mathbb{R}^n$ and $j \in \mathbb{N}$*

$$\kappa < g_j(v) < \kappa^{-1}.$$

Then

$$d(\mu_J^N, \mu_J) \leq \sum_{j=1}^{J} (2\kappa^{-2})^j \frac{1}{\sqrt{N}}.$$

Proof. The desired result follows in a straightforward way from the following three facts, whose proof we postpone to three lemmas at the end of the section:

$$\sup_{\mu\in\mathcal{P}(\mathbb{R}^n)} d(S^N\mu, \mu) \leq \frac{1}{\sqrt{N}}, \tag{3.25a}$$

$$d(P\nu, P\mu) \leq d(\nu,\mu) \tag{3.25b}$$

$$d(C_j\nu, C_j\mu) \leq 2\kappa^{-2} d(\nu,\mu). \tag{3.25c}$$

By the triangle inequality we have, for $\nu_j^N = P\mu_j^N$,

$$\begin{aligned} d(\mu_{j+1}^N, \mu_{j+1}) &= d(C_j S^N P\mu_j^N, C_j P\mu_j) \\ &\leq d(C_j P\mu_j^N, C_j P\mu_j) + d(C_j S^N P\mu_j^N, C_j P\mu_j^N) \\ &\leq 2\kappa^{-2}\Big(d(\mu_j^N, \mu_j) + d(S^N\nu_j^N, \nu_j^N)\Big) \\ &\leq 2\kappa^{-2}\Big(d(\mu_j^N, \mu_j) + \frac{1}{\sqrt{N}}\Big). \end{aligned}$$

Iterating, after noting that $\mu_0^N = \mu_0$, gives the desired result. □

Remarks 3.6. *This important theorem shows that the particle filter actually reproduces the true filtering distribution, in the large particle limit. We make some comments about this.*

- *This theorem shows that, at any fixed J, the filtering distribution μ_j is well-approximated by the bootstrap filtering distribution μ_j^N in the sense that, as the number of particles $N \to \infty$, the approximating measure converges to the true measure. However, since $\kappa < 1$, the number of particles required to decrease the upper bound on the error beneath a specified tolerance grows with J.*

- *In many applications the likelihoods g_j may not be bounded from above or below, uniformly in j, and more refined analysis is required.*
- *If the Markov kernel P is ergodic then it is possible to obtain bounds which are* uniform *in J.*
- *Considering the case of deterministic dynamics shows just how difficult it may be to make the theorem applicable in practice: if the dynamics is deterministic then the original set of samples from μ_0, $\{v_0^{(n)}\}_{n=1}^N$ give rise to a set of particles $v_j^{(n)} = \Psi^{(j)}(v_0^{(n)})$; in other words the particle positions are* unaffected by the data. *This is clearly a highly undesirable situation, in general, since there is no reason at all why the pushforward under the dynamics of the initial measure μ_0 should have substantial overlap with the filtering distribution. This example motivates the improved proposals of the next section.*

Before describing improvements to the basic particle filter, we prove the three lemmas which underly the convergence proof.

Lemma 3.7. *The sampling operator satisfies*

$$\sup_{\mu\in\mathcal{P}(\mathbb{R}^n)} d(S^N\mu,\mu) \leq \frac{1}{\sqrt{N}}.$$

Proof. Let ν be an element of $\mathcal{P}(\mathbb{R}^n)$ and $\{v^{(k)}\}_{k=1}^N$ i.i.d. with $v^{(1)}\sim\nu$. Then

$$S^N\nu(f) = \frac{1}{N}\sum_{j=1}^N f\left(v^{(k)}\right)$$

and, defining $\overline{f} = f - \nu(f)$, we deduce that

$$S^N\nu(f) - \nu(f) = \frac{1}{N}\sum_{j=1}^N \overline{f}\left(v^{(k)}\right).$$

It is straightforward to see that

$$\mathbb{E}\overline{f}\left(v^{(k)}\right)\overline{f}\left(v^{(l)}\right) = \delta_{kl}\mathbb{E}\left|\overline{f}\left(v^{(k)}\right)\right|^2.$$

Furthermore, for $|f|\leq 1$,

$$\mathbb{E}\left|\overline{f}\left(v^{(1)}\right)\right|^2 = \mathbb{E}\left|f\left(v^{(1)}\right)\right|^2 - \left|\mathbb{E}f\left(v^{(1)}\right)\right|^2 \leq 1.$$

It follows that, for $|f|\leq 1$,

$$\mathbb{E}^\omega\left|\nu(f) - S^N\nu(f)\right|^2 = \frac{1}{N^2}\sum_{j=1}^N \mathbb{E}\left|\overline{f}\left(v^{(k)}\right)\right|^2 \leq \frac{1}{N}.$$

Since the result is independent of ν we may take the supremum over all probability measures and obtain the desired result. □

Lemma 3.8. *Since P is a Markov kernel we have*

$$d(P\nu,P\mu)\leq d(\nu,\mu).$$

Proof. Define

$$q(v') = \int_{\mathbb{R}^n}\mathbb{P}(dv|v')f(v),$$

that is the expected value of f under one-step of the Markov chain given by (1.1a), started from v'. Clearly

$$\sup_v|q(v)|\leq\sup_v|f(v)|.$$

It also follows that

$$|P\nu(f) - P\nu'(f)| = |\nu(q) - \nu'(q)|.$$

Thus

$$\begin{aligned} d(P\nu, P\nu') &= \sup_{|f|\leq 1} \left(\mathbb{E}^\omega |P\nu(f) - P\nu'(f)|^2\right)^{\frac{1}{2}} \\ &\leq \sup_{|q|\leq 1} \left(\mathbb{E}^\omega |\nu(q) - \nu'(q)|^2\right)^{\frac{1}{2}} \\ &= d(\nu, \nu') \end{aligned}$$

as required. □

Lemma 3.9. *Under the Assumptions of Theorem 3.5 we have*

$$d(C_j\nu, C_j\mu) \leq 2\kappa^{-2} d(\nu, \mu).$$

Proof. Notice that for $|f|_\infty < \infty$ we can rewrite

$$(C_j\nu)(f) - (C_j\mu)(f) = \frac{\nu(fg_j)}{\nu(g_j)} - \frac{\mu(fg_j)}{\mu(g_j)} \tag{3.26a}$$

$$= \frac{\nu(fg_j)}{\nu(g_j)} - \frac{\mu(fg_j)}{\nu(g_j)} + \frac{\mu(fg_j)}{\nu(g_j)} - \frac{\mu(fg_j)}{\mu(g_j)} \tag{3.26b}$$

$$= \frac{\kappa^{-1}}{\nu(g_j)}[\nu(\kappa f g_j) - \mu(\kappa f g_j)] + \frac{\mu(fg_j)}{\mu(g_j)} \frac{\kappa^{-1}}{\nu(g_j)}[\mu(\kappa g_j) - \nu(\kappa g_j)]. \tag{3.26c}$$

Now notice that $\nu(g_j)^{-1} \leq \kappa^{-1}$ and that $\mu(fg_j)/\mu(g_j) \leq 1$ since the expression corresponds to an expectation with respect to measure found from μ by reweighting with likelihood proportional to g_j. Thus

$$|(C_j\nu)(f) - (C_j\mu)(f)| \leq \kappa^{-2}|\nu(\kappa f g_j) - \mu(\kappa f g_j)| + \kappa^{-2}|\nu(\kappa g_j) - \mu(\kappa g_j)|.$$

Since $|\kappa g_j| \leq 1$ it follows that

$$\mathbb{E}^\omega |(C_j\nu)(f) - (C_j\mu)(f)|^2 \leq 4\kappa^{-4} \sup_{|f|\leq 1} \mathbb{E}^\omega |\nu(f) - \mu(f)|^2$$

and the desired result follows. □

3.3.3. Improved Proposals

In the particle filter described in the previous section we propose according to underlying unobserved dynamics, and then apply Bayes' formula to incorporate the data. The final point in Remarks 3.6 demonstrates that this may result in a very poor set of particles with which to approximate the filtering distribution. Cleverer proposals, which use the data, can lead to improved performance and we outline this methodology here.

Instead of moving the particles $\{v_j^{(n)}\}_{n=1}^N$ according to the Markov kernel P, we use a Markov kernel Q_j with density $\mathbb{Q}(v_{j+1}|v_j, Y_{j+1})$. The weights $w_{j+1}^{(n)}$ are found, as before, by applying Bayes' formula for each particle, and then weighting appropriately; the result is found from (3.21) by reweighting according to the ratio of the kernel of the true dynamics to the kernel of the proposed dynamics, which gives

$$\widehat{w}_{j+1}^{(n)} = w_j^{(n)} \frac{\mathbb{P}\left(y_{j+1}|\widehat{v}_{j+1}^{(n)}\right) \mathbb{P}\left(\widehat{v}_{j+1}^{(n)}|v_j^{(n)}\right)}{\mathbb{Q}\left(\widehat{v}_{j+1}^{(n)}|v_j^{(n)}, Y_{j+1}\right)}. \tag{3.27}$$

Normalization then requires that we apply (3.21) as before. This results in the following algorithm:

1. Set $j = 0$ and $\mu_0^N(v_0)dv_0 = \mathbb{P}(v_0)dv_0$.
2. Draw $v_j^{(n)} \sim \mu_j^N$, $n = 1, \ldots, N$.
3. Set $w_j^{(n)} = 1/N$, $n = 1, \ldots, N$.
4. Draw $\widehat{v}_{j+1}^{(n)} \sim \mathbb{Q}(\cdot|v_{j+1}^{(n)}, Y_{j+1})$.
5. Define $w_{j+1}^{(n)}$ by (3.27) and $\mathbb{P}^N(v_{j+1}|Y_{j+1})$ by (3.20).
6. $j + 1 \to j$.
7. Go to step 2.

The so-called **optimal proposal** is found by choosing

$$\mathbb{Q}\left(v_{j+1}|v_j^{(n)}, Y_{j+1}\right) \equiv \mathbb{P}\left(v_{j+1}|y_{j+1}, v_j^{(n)}\right)$$

which results in

$$\widehat{w}_{j+1}^{(n)} = w_j^{(n)} \mathbb{P}\left(y_{j+1}|v_j^{(n)}\right). \tag{3.28}$$

The above can be seen by observing that the definition of conditional probability gives

$$\begin{aligned} \mathbb{P}\left(y_{j+1}|\widehat{v}_{j+1}^{(n)}\right) \mathbb{P}\left(\widehat{v}_{j+1}^{(n)}|v_j^{(n)}\right) &= \mathbb{P}\left(y_{j+1}, \widehat{v}_{j+1}^{(n)}|v_j^{(n)}\right) \\ &= \mathbb{P}\left(\widehat{v}_{j+1}^{(n)}|y_{j+1}, v_j^{(n)}\right) \mathbb{P}\left(y_{j+1}|v_j^{(n)}\right). \end{aligned} \tag{3.29}$$

Substituting the optimal proposal into (3.27) then immediately gives (3.28).

This small difference may seem trivial at a glance, and at a large cost of evaluating the normalizing constant. However, in the case of nonlinear Gaussian Markov models as we study here, the distribution and the weights are given in closed form. If the dynamics is highly nonlinear or the model noise is much larger than the observational noise then the variance of the weights for the optimal proposal may be much smaller than for the standard proposal. The corresponding particle filter is also more likely to track rare events, such as a transition between wells in a double-well potential. For deterministic dynamics the optimal proposal reduces to the standard proposal.

3.4. Stability Analyses

An important question concerning filters in general is their behaviour when iterated over long times and, in particular, their ability to recover the true signal underlying the data if iterated for long enough. In this section we present some basic stability results for filters to illustrate the key issue which affects the ability of filters to accurately recover the signal when iterated for long enough. This key idea is that the data must be sufficiently rich to stabilize any inherent instabilities within the underlying dynamical model (1.1); in rough terms it is necessary to observe only the unstable directions as the dynamics of the model itself will enable recovery of the true signal within the stable directions. We illustrate this idea first, in subsection 3.4.1, for the explicitly solvable case of the Kalman filter in one dimension, and then, in subsection 3.4.2, for the 3DVAR method.

3.4.1. The Kalman Filter in One Dimension

We consider the case of one dimensional dynamics with

$$\Psi(v) = \lambda v, \quad h(v) = v,$$

while we will also assume that

$$\Sigma = \sigma^2, \quad \Gamma = \gamma^2.$$

With these definitions equations (3.2a,b) become

$$\frac{1}{c_{j+1}} = \frac{1}{\sigma^2+\lambda^2 c_j} + \frac{1}{\gamma^2}, \tag{3.30a}$$

$$\frac{m_{j+1}}{c_{j+1}} = \frac{\lambda m_j}{\sigma^2+\lambda^2 c_j} + \frac{1}{\gamma^2} y_{j+1}, \tag{3.30b}$$

which, after some algebraic manipulations, give

$$c_{j+1} = g(\lambda, c_j), \tag{3.31a}$$

$$m_{j+1} = \left(1 - \frac{c_{j+1}}{\gamma^2}\right)\lambda m_j + \frac{c_{j+1}}{\gamma^2} y_{j+1}, \tag{3.31b}$$

where we have defined

$$g(c) := \frac{\gamma^2(\lambda^2 c + \sigma^2)}{\gamma^2 + \lambda^2 c + \sigma^2}. \tag{3.32}$$

We wish to study the behaviour of the Kalman filter as $j \to \infty$, *i.e.* when more and more data points are assimilated into the model. Note that the covariance evolves independently of the data $\{y_j\}_{j\in\mathbb{Z}^+}$ and satisfies an autonomous dynamical system; we first study the asymptotic properties of this map. The fixed points $c^\star$ of (3.31a) satisfy

$$c^\star = \frac{\gamma^2(\lambda^2 c^\star + \sigma^2)}{\gamma^2 + \lambda^2 c^\star + \sigma^2}, \tag{3.33}$$

and thus solve the quadratic equation

$$\lambda (c^\star)^2 + \left(\gamma^2(1-\lambda^2) + \sigma^2\right) c^\star - \gamma^2\sigma^2.$$

We see that, provided $\lambda\gamma\sigma \neq 0$, one root is positive and one negative. The roots are given by

$$c^\star_\pm = \frac{-(\gamma^2+\sigma^2-\gamma^2\lambda^2) \pm \sqrt{(\gamma^2+\sigma^2-\gamma^2\lambda^2)^2 + 4\lambda^2\gamma^2\sigma^2}}{2\lambda^2}. \tag{3.34}$$

We observe that the update formula for the covariance ensures that, provided $c_0 > 0$ then $c_j > 0$ for all $j \in \mathbb{Z}^+$. It also demonstrates that $c_j \leq \gamma^2$ for all $j \in \mathbb{Z}^+$. Thus we fix our attention on non-negative covariances. We will now study the stability of the non-negative fixed points.

We first start with the case $\sigma = 0$, which corresponds to no model error. In this case we obtain

$$c^\star_+ = 0, \quad c^\star_- = \frac{\gamma^2(\lambda^2-1)}{\lambda^2},$$

and

$$g'(c^\star_+) = \lambda^2, \quad g'(c^\star_-) = \lambda^{-2},$$

which implies that when $\lambda^2 < 1$, $c^\star_+$ is an asymptotically stable fixed point, while when $\lambda^2 > 1$, $c^\star_-$ is an asymptotically stable fixed point. When $|\lambda| = 1$ the two roots are coincident at the origin and neutrally stable. In fact for the case $\sigma = 0$ it is possible to solve (3.30a) to obtain for $\lambda^2 \neq 1$

$$\frac{1}{c_j} = \left(\frac{1}{\lambda^2}\right)^j \frac{1}{c_0} + \frac{1}{\gamma^2}\left[\frac{\left(\frac{1}{\lambda^2}\right)^j - 1}{\frac{1}{\lambda^2} - 1}\right]. \tag{3.35}$$

This explicit formula shows that the fixed point $c^\star_+$ (resp. $c^\star_-$) is globally asymptotically stable, and exponentially attracting on $\mathbb{R}^+$, when $\lambda^2 < 1$ (resp. $\lambda^2 > 1$). Notice also that $c^\star_- = \mathcal{O}(\gamma^2)$ so that when $\lambda^2 > 1$, the asymptotic variance of the filter scales as the observational noise variance. Furthermore, when $\lambda^2 = 1$ we may solve (3.30a) to obtain

$$\frac{1}{c_j} = \frac{1}{c_0} + \frac{j}{\gamma^2},$$

showing that $c_-^\star = c_+^\star = 0$ is globally asymptotically stable on $\mathbb{R}^+$, but is only algebraically attracting.

We now study the stability of the fixed points $c_+^\star$ and $c_-^\star$ in the case of $\sigma^2 > 0$ corresponding to the case of the imperfect model scenario. To this end we prove some bounds on $g'(c^\star)$ that will be also be useful when we study the behaviour of the error between the true signal and the estimated mean; here, and in what follows in the remainder of this example, prime denotes differentiation with respect to c. We start by noting that

$$g(c) = \gamma^2 - \frac{\gamma^4}{(\gamma^2 + \lambda^2 c + \sigma^2)}, \tag{3.36}$$

and so

$$g'(c) = \frac{\lambda^2 \gamma^4}{(\gamma^2 + \lambda^2 c + \sigma^2)^2}.$$

Using the fact that $c^\star$ is a fixed point of (3.31a), together with equation (3.36), we obtain

$$g'(c^\star) = \frac{1}{\lambda^2} \frac{(c^\star)^2}{\left(c^\star + \frac{\sigma^2}{\lambda^2}\right)^2} \quad \text{and} \quad g'(c^\star) = \lambda^2 \left(1 - \frac{c^\star}{\gamma^2}\right)^2.$$

We can now see that from the first equation we obtain the following two bounds, since $\sigma^2 > 0$:

$$g'(c^\star) < \lambda^{-2}, \quad \text{for} \quad \lambda \in \mathbb{R}, \quad \text{and} \quad g'(c^\star) < 1, \quad \text{for} \quad \lambda^2 = 1,$$

while from the second equality and the fact that, since $c^\star$ satisfies (3.36), $c^\star < \gamma^2$ we obtain

$$g'(c^\star) < \lambda^2.$$

when $c^\star > 0$. We thus conclude that when $\sigma^2 > 0$ the fixed point $c_+^\star$ of (3.31a) is always stable independently of the value of the parameter λ.

We will now study the behaviour of the mean of the Kalman filter in the case of one dimensional dynamics. We assume that the data $\{y_j\}_{j\in\mathbb{N}}$ is generated from a true signal $\{v_j^\dagger\}_{j\in\mathbb{Z}^+}$ governed by the equation

$$v_{j+1}^\dagger = \lambda v_j^\dagger$$

with the addition of noise $\{\eta_j\}_{j\in\mathbb{N}}$ which is a drawn from i.i.d. sequences with variance γ^2. From (3.31b) we have

$$m_{j+1} = \left(1 - \frac{c_{j+1}}{\gamma^2}\right) \lambda m_j + \frac{c_{j+1}}{\gamma^2}(\lambda v_j^\dagger + \eta_{j+1})$$

while we also have

$$v_{j+1}^\dagger = \left(1 - \frac{c_{j+1}}{\gamma^2}\right) \lambda v_j^\dagger + \frac{c_{j+1}}{\gamma^2} \lambda v_j^\dagger.$$

Defining the error $e_j = m_j - v_j^\dagger$ between the estimated mean and the true signal and subtracting the previous two equations we obtain

$$e_{j+1} = \left(1 - \frac{c_{j+1}}{\gamma^2}\right) \lambda e_j + \frac{c_{j+1}}{\gamma^2} \eta_{j+1}. \tag{3.37}$$

Having obtained (3.37) it is interesting to ask what happens to the error in the case when more and more data are assimilated in the model, *i. e* $j \to \infty$. To address this question it is useful to observe that since η_{j+1} is a Gaussian random variable then e_j is also Gaussian. Thus it suffices to characterise the behaviour of $\mathbb{E}e_j$ and $\mathbb{E}e_j^2$ to obtain information about the long time behaviour of the error. By taking expectations in (3.37) and using the fact that η_{j+1} has zero mean, and it is conditionally independent of e_j, we obtain

$$\mathbb{E}e_{j+1} = \lambda \left(1 - \frac{c_{j+1}}{\gamma^2}\right) \mathbb{E}e_j, \tag{3.38}$$

while by taking the square of (3.37), and using the fact that $\mathbb{E}(\eta_{j+1}^2) = \gamma^2$ we have

$$\mathbb{E}e_{j+1}^2 = \lambda^2 \left(1 - \frac{c_{j+1}}{\gamma^2}\right)^2 \mathbb{E}e_j^2 + \frac{c_{j+1}^2}{\gamma^2}. \tag{3.39}$$

Equation (3.38) can be solved to obtain

$$\mathbb{E}e_j = \lambda^j \left[\prod_{i=0}^{j}\left(1-\frac{c_{i+1}}{\gamma^2}\right)\right]\mathbb{E}e_0, \tag{3.40}$$

while in a similar way we obtain for the solution of (3.39)

$$\mathbb{E}e_j^2 = \lambda^{2j}\left[\prod_{i=0}^{j-1}\left(1-\frac{c_{i+1}}{\gamma^2}\right)^2\right]\mathbb{E}e_0^2 + \sum_{i=0}^{j-1}\left\{\left[\prod_{k=i+1}^{j}\left(1-\frac{c_k}{\gamma^2}\right)\right]\lambda^{2(j-i)}\frac{c_i^2}{\gamma^2}\right\} + \frac{c_j^2}{\gamma^2} \tag{3.41}$$

Using the fact derived earlier, that $c_j \leq \gamma^2, j \geq 1$, we see that in the case $|\lambda|^2 < 1$

$$|\lambda|\left(1-\frac{c_j}{\gamma^2}\right) \leq \lambda^\star < 1, \quad \forall j \geq 1 \tag{3.42}$$

Furthermore, in the case where $|\lambda| > 1$ and $\sigma^2 \geq 0$, or when $|\lambda| = 1$ and $\sigma^2 \neq 0$, using (3.36) and (3.33) we obtain

$$|\lambda|\left(1-\frac{c_+^\star}{\gamma^2}\right) = \frac{|\lambda|\gamma^2}{\gamma^2+\lambda^2 c_+^\star+\sigma^2} = \frac{|\lambda|c_+^\star}{\lambda^2 c_+^\star+\sigma^2} < 1, \quad \text{when} \quad \sigma^2 > 0. \tag{3.43}$$

Now since $c_j \to c_+^\star$ we deduce that for every $\epsilon > 0$ there is $J = J(\epsilon)$ such that

$$|c_j - c_+^\star| < \epsilon, \quad \forall j > J$$

which together with (3.43) implies that

$$|\lambda|\left(1-\frac{c_{j+1}}{\gamma^2}\right) \leq \lambda^\star < 1, \quad \forall j > J. \tag{3.44}$$

Equations (3.42) and (3.44) together with (3.40) imply that

$$\mathbb{E}e_j \to 0, \quad \forall \lambda \tag{3.45}$$

when $\sigma^2 \neq 0$. Furthermore, using the fact that $|c_j| \leq \gamma^2$, $j \geq 1$ we see that (3.37) implies that

$$\mathbb{E}e_j^2 \leq \lambda^{2j}\left[\prod_{i=0}^{j-1}\left(1-\frac{c_{i+1}}{\gamma^2}\right)^2\right]\mathbb{E}e_0^2 + \sum_{i=0}^{j-1}\left\{\left[\prod_{k=i+1}^{j}\left(1-\frac{c_k}{\gamma^2}\right)\right]\lambda^{2(j-i)}\gamma^2\right\} + \gamma^2$$

which again using (3.42) and (3.44) implies that

$$\mathbb{E}e_j^2 \leq C\gamma^2, \quad \forall \lambda, j \geq J \tag{3.46}$$

when $\sigma^2 > 0$, for J sufficiently large. We are thus now left with considering the case $\sigma^2 = 0, |\lambda| = 1$ for the asymptotic behaviour of the error e_j. Note that in this case, the limiting covariance matrix is equal to zero but the convergence to it, unlike the case $|\lambda| < 1$ is only algebraic, since in this case (3.35) implies that

$$c_j = \frac{\gamma^2 c_0}{\gamma^2 + jc_0}.$$

We deduce that in this case

$$\lambda\left(1-\frac{c_{j+1}}{\gamma^2}\right) = 1 - \frac{\gamma^2 c_0}{\gamma^2+jc_0} \leq 1, \quad \forall j \geq 1$$

which similarly to the case $\sigma^2 > 0$ implies that (3.45) and (3.46) hold also for the case $|\lambda| = 1, \sigma^2 = 0$. Table 1 summarises the behaviour of the Kalman filter in the case of one-dimensional dynamics. Furthermore, the limiting behaviour of the error e_j satisfies , for all the possible combinations λ and σ^2,

$$e_j \to N(0,\alpha), \quad \text{where} \quad \alpha \leq C\gamma^2.$$

We learn from this that, in the one-dimensional case, the Kalman filter can recover the true signal stably and accurately provided that unstable directions are observed.

	Limiting covariance for $\sigma^2 = 0$	Limiting covariance for $\sigma^2 > 0$
$\|\lambda\| < 1$	$c_j \to 0$ (exponentially)	$c_j \to c_+^\star = \mathcal{O}(\gamma^2)$ (exponentially)
$\|\lambda\| = 1$	$c_j \to 0$ (algebraically)	$c_j \to c_+^\star = \mathcal{O}(\gamma^2)$ (exponentially)
$\|\lambda\| > 1$	$c_j \to c_-^\star = \mathcal{O}(\gamma^2)$ (exponentially)	$c_j \to c_+^\star = \mathcal{O}(\gamma^2)$ (exponentially)

TABLE 1

Summary of the limiting behaviour of covariance c_j for Kalman filter applied to one dimensional dynamics

3.4.2. The 3DVAR Filter

Recall the 3DVAR filter (3.12). Armed with our insight from the analysis of the one-dimensional Kalman filter, it is natural to ask when the 3DVAR filter will recover the true signal. To this end we assume that

$$y_{j+1} = Hv_{j+1}^\dagger + \epsilon_j \tag{3.47}$$

where the true signal $\{v_j^\dagger\}_{j\in\mathbb{N}}$ satisfies

$$v_{j+1}^\dagger = \Psi(v_j^\dagger),\ j \in \mathbb{N} \tag{3.48a}$$

$$v_0^\dagger = u \tag{3.48b}$$

and, for simplicity, we assume that

$$\sup_{j\in\mathbb{N}} |\epsilon_j| = \epsilon. \tag{3.49}$$

We have the following result.

Theorem 3.10. *Assume that (3.49) holds and that $\widehat{C}$ can be chosen so that $(I - KH)\Psi : \mathbb{R}^n \to \mathbb{R}^n$ is globally Lipschitz with constant $a < 1$ in some norm $\|\cdot\|$, then there is constant $c > 0$ such that*

$$\limsup_{j\to\infty} \|m_j - v_j^\dagger\| \le \frac{c}{1-a}\epsilon.$$

Proof. We may write (3.12), (3.48), using (3.47), as

$$\begin{aligned} m_{j+1} &= (I - KH)\Psi(m_j) + KH\Psi(v_j^\dagger) + K\epsilon_j \\ v_{j+1}^\dagger &= (I - KH)\Psi(v_j^\dagger) + KH\Psi(v_j^\dagger). \end{aligned}$$

Subtracting, and letting $e_j = m_j - v_j^\dagger$ gives, for some finite constant c independent of j,

$$\begin{aligned} \|e_{j+1}\| &\le \|(I - KH)\Psi(m_j) - (I - KH)\Psi(v_j^\dagger)\| + \|K\epsilon_j\| \\ &\le a\|e_j\| + c\epsilon. \end{aligned}$$

Applying Gronwall gives the desired result. □

We refer to a map with Lipschitz constant less than 1 as a *contraction* in what follows.

Remark 3.11. *This simple theorem shows that it may be possible to construct filters which lock-on to a small neighbourhood of the true signal underlying the data, and can recover from being initialized far from the truth. This illustrates a key idea in filtering: the question of of whether a $\widehat{C}$ can be chosen to make $(I - KH)\Psi$ into a contraction involves a subtle interplay between the underlying unobserved dynamics, encapsulated in Ψ, and the observation operator H. In rough terms the question of making $(I - KH)\Psi$ into a contraction is the question as to whether the unstable parts of the dynamics are observed; if they are then it is typically the case that $\widehat{C}$ can be designed to obtain the desired contraction.*

Example 3.12. *Assume that $H = I$, so that the whole system is observed, that $\Gamma = \gamma^2 I$ and $\widehat{C} = \sigma^2 I$. Then, for $\eta^2 = \frac{\gamma^2}{\sigma^2}$*

$$S = (\sigma^2 + \gamma^2)I, \quad K = \frac{\sigma^2}{(\sigma^2 + \gamma^2)}I$$

and

$$(I - KH) = \frac{\gamma^2}{(\sigma^2 + \gamma^2)} I = \frac{\eta^2}{(1 + \eta^2)} I.$$

Thus, if $\Psi : \mathbb{R}^n \to \mathbb{R}^n$ is globally Lipschitz with constant $\lambda > 0$ in the Euclidean norm, $|\cdot|$, then $(I - KH)\Psi$ is globally Lipschitz with constant $a < 1$, if η is chosen so that $\frac{\eta^2 \lambda}{1+\eta^2} < 1$. Thus, by choosing η sufficiently small the filter can be made to contract. This corresponds to trusting the data sufficiently in comparison to the model. It is a form of **variance inflation**.

Example 3.13. *Assume that there is a partition of the state space in which $H = (I, 0)^T$, so that only part of the system is observed. Set $\Gamma = \gamma^2 I$ and $\widehat{C} = \sigma^2 I$. Then, with η as in the previous example,*

$$I - KH = \begin{pmatrix} \frac{\eta^2}{1+\eta^2} I & 0 \\ 0 & I \end{pmatrix}.$$

Now variance inflation may help to stabilize the filter, but more is required: it is clear from this simple example that making $(I - KH)\Psi(\cdot)$ into a contraction will require a relationship between the subspace in which we observe and the space in which the dynamics of the map is expanding and contracting. For example, if $\Psi(u) = Lu$ and

$$L = \begin{pmatrix} 2I & 0 \\ 0 & aI \end{pmatrix}$$

then

$$(I - KH)L = \begin{pmatrix} \frac{2\eta^2}{1+\eta^2} I & 0 \\ 0 & aI \end{pmatrix}$$

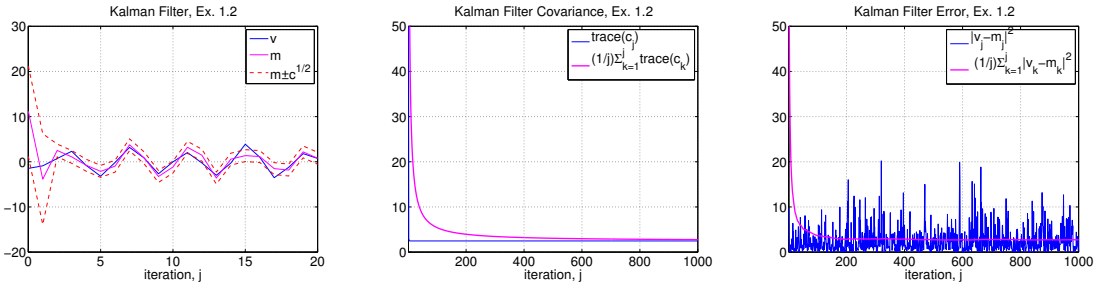
When $|a| < 1$ this can be made into a contraction by choosing η sufficiently small; but for $|a| \geq 1$ this is no longer possible. This illustrates the intuitive idea that the observations should be sufficiently rich to ensure that the unstable directions within the dynamics can be tamed by observing them.

3.5. Illustrations

Our first illustration concerns the Kalman filter applied to the linear system of Example 1.2 with $A = A_3$. We assume that $H = (1, 0)$ so that we observe only the first component of the system and the model and observational covariances are $\Sigma = I$ and $\Gamma = 1$, where I is the 2×2 identity. The problem is initialized with mean 0 and covariance $10\, I$. Figure 21a shows the behaviour of the filter on the unobserved component, showing how the mean locks onto a small neighbourhood of the truth and how the one-standard deviation confidence intervals computed from the variance on the second component also shrink from a large initial value to an asymptotic small value; this value is determined by the observational noise variance in the first component. In Figure 21b the trace of the covariance matrix is plotted demonstrating that the total covariance matrix asymptotes to a small limiting matrix. And finally Figure 21c shows the mean-square error and its running average. We will employ similar figures (a), (b) and (c) in the examples which follow in this section.

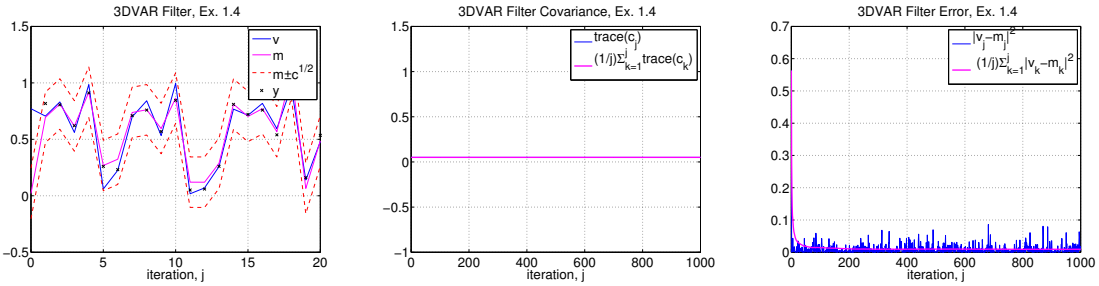
Our next illustration shows the 3DVAR methodology applied to the Example 1.4 with $r = 2.5$. We consider noise-free dynamics and observational variance of $\gamma^2 = 10^{-2}$. The fixed model covariance is chosen to be $c = \gamma^2/\eta$ with $\eta = 0.2$. The resulting algorithm performs well tracking the truth with asymptotic time-averged mean-square error of size roughly 10^{-2}. See Figure 22.

We now compare the performance of 3DVAR, ExKF and EnKF on Example 1.3 with $\alpha = 2.5$ and model noise variance 0.3. The observational noise variance is set at 1. For 3DVAR, we take $\eta = 0.5$. For the EnKF we take 100 ensemble members. Figures 23, 24 and 25 summarize the algorithm behaviour for 3DVAR, ExKF and EnKF respectively. Notice from Fig. 24 that the ExKF has small error for most of the simulation, but that sporadic large (in comparison to those seen for 3DVAR and EnKF) excursions are seen in the error.



(a) Solution. (b) Covariance. (c) Error.

FIG 21. *Kalman filter applied to the linear system of Example 1.2 with* $A = A_3$, $H = (1, 0)$, $\Sigma = I$, *and* $\Gamma = 1$, *see also* `p7.m` *in Section 4.3.1. The problem is initialized with mean* 0 *and covariance* $10\, I$.



(a) Solution. (b) Covariance. (c) Error.

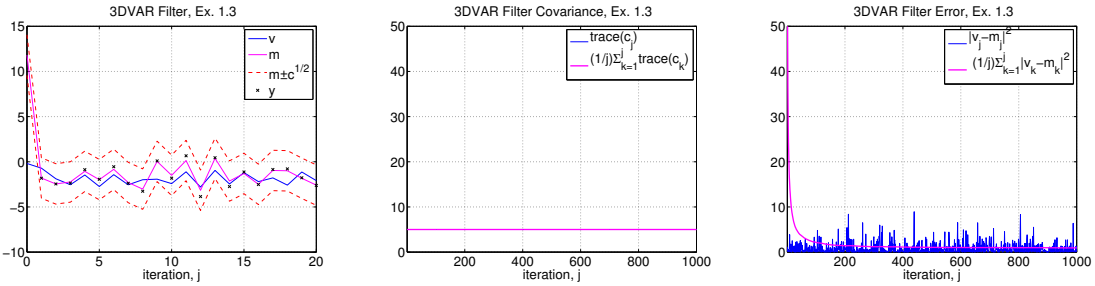
FIG 22. *3DVAR methodology applied to the logistic map Example 1.4 with* $r = 4$, $\gamma^2 = 10^{-2}$, *and* $c = \gamma^2/\eta$ *with* $\eta = 0.2$, *see also* `p8.m` *in Section 4.3.2.*

Figure 26 compares the errors incurred by the three methods, demonstrating that the EnKF is the most accurate method on average, with ExKF the least accurate on average. Notice from Fig. 26(a) that the error distribution of 3DVAR is the widest, and both it and EnKF remain consistently accurate. The distribution of ExKF is similar to EnKF, except with fat tails associated to the destabilization intervals seen in Fig. 24.

Finally, ETKF (Fig. 27), and the particle filters with optimal (Fig. 29) and standard (Fig. 28) proposals are also compared also on Example 1.3 with $\alpha = 2.5, \sigma = 0.3$, and $\gamma = 1$ with 100 ensemble members each. Notice from Fig. 27(c) that ETKF is more prone to destabilization than EnKF with perturbed observations in Fig. 25(c). Also, notice from Fig. 28(c) that the particle filter with standard proposal is more prone to destabilization than the optimal proposal in Fig. 29(c), although the optimal proposal still loses track sometimes. Fig. 30(a) shows that the error distribution of all these filters is similar, and ETKF and optimal proposal SIRS also remain consistently accurate. The distribution of standard SIRS is similar to the optimal proposal, except with fat tails associated to the destabilization intervals seen in Fig. 28, which leads to the larger RMSE, similar to ExKF.

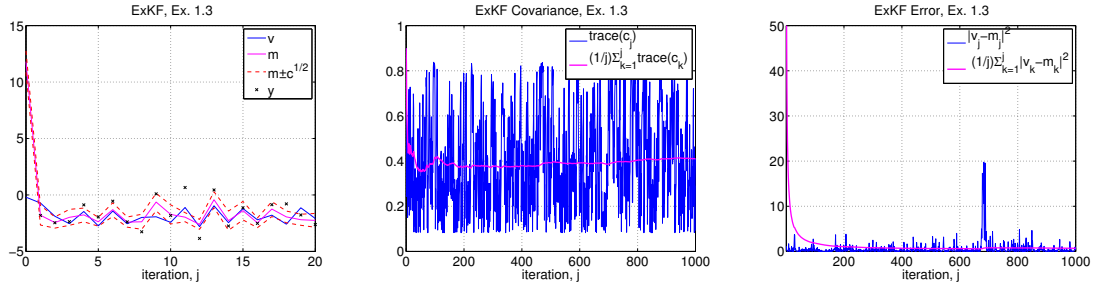
3.6. Bibliographic Notes

- Section 3.1 The Kalman Filter has found wide-ranging application to low dimensional engineering applications where the linear Gaussian model is appropriate, since its introduction in 1960 [Kal60]. In addition to the original motivation in control of flight vehicles, it has grown in importance in the fields of econometric time-series analysis, and signal processing [Har91]. It is also important because it plays a key role in the development of the *ad hoc* non-Gaussian filters which are the subject of the next section. The idea behind the Kalman filter, to optimally combine model and data, is arguably one of the most important ideas in applied mathematics over the last century: the impact of the paper



(a) Solution. (b) Covariance. (c) Error.

FIG 23. *3DVAR for the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3, \gamma = 1$, *and* $\eta = 0.2$, *see also* `p9.m` *in Section 4.3.3.*

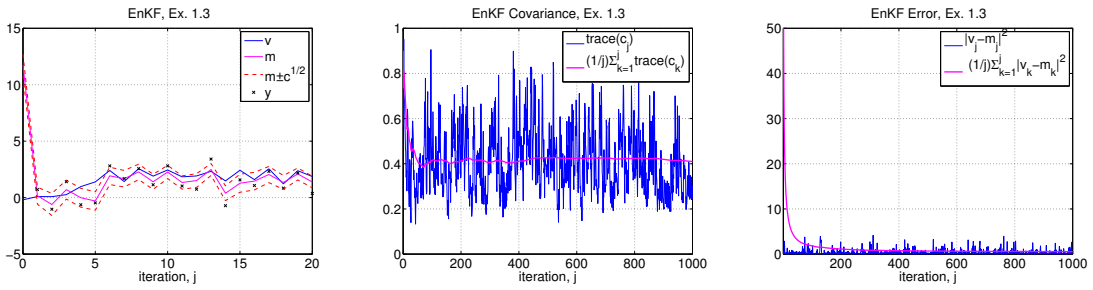


(a) Solution. (b) Covariance. (c) Error.

FIG 24. *ExKF on the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3$, *and* $\gamma = 1$, *see also* `p10.m` *in Section 4.3.4.*

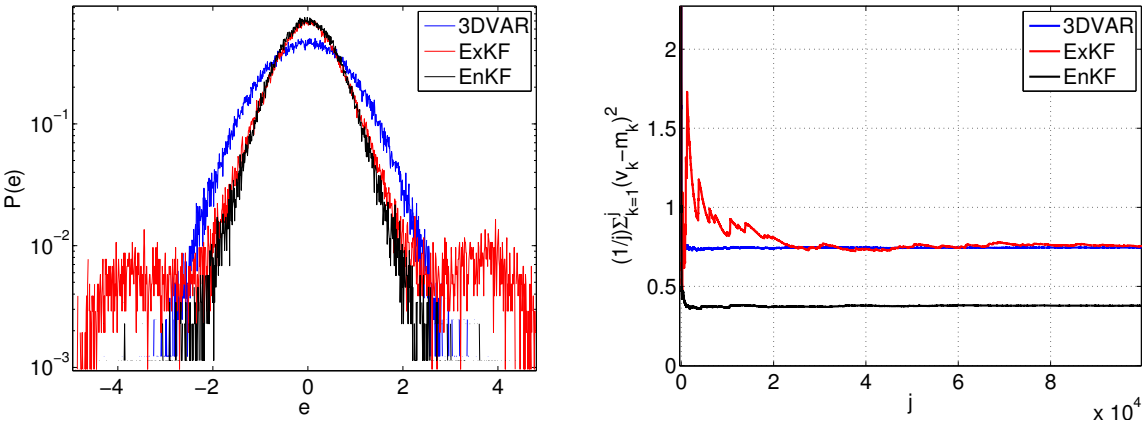
[Kal60] on many applications domains has been huge.

- Section 3.2 All the non-Gaussian Filters we discuss are based on modifying the Kalman filter so that it may be applied to non-linear problems. The development of new filters is a very active area of research and the reader is directed to the book [MH12], together with the articles [CMT10],[MHG10] and [VL09] for insight into some of the recent developments with an applied mathematics perspective.
  The 3DVAR algorithm was proposed at the UK Met Office in 1986 [Lor00, LBB+00], and was subsequently developed by the US National Oceanic and Atmospheric Administration [PD92] and by the European Centre for Medium-Range Weather Forecasts (ECMWF) in [CAH+98]. The 3DVAR algorithm is prototypical of the many more sophisticated filters which are now widely used in practice and it is thus natural to study it.
  The extended Kalman filter was developed in the control theory community and is discussed at length in [Jaz70]. It is not practical to implement in high dimensions, and low-rank extended Kalman filters are then used instead; see [LS12] for a recent discussion.
  The ensemble Kalman filter uses a set of particles to estimate covariance information, and may be viewed as an approximation of the extended Kalman filter, designed to be suitable in high dimensions. See [Eve06] for an overview of the methodology, written by one of its originators, and [VLE96] for an early example of the power of the method.
  Note that the Γ appearing in the perturbed observation EnKF can be replaced by the sample covariance $\tilde{\Gamma}$ of the $\{\eta_{j+1}^{(k)}\}_{k=1}^K$ and this is often done in pracice. The sample covariance of the updated ensemble in this case is equal to $(I - \tilde{K}_{j+1}H)\widehat{C}_{j+1}$ where $\tilde{K}_{j+1}$ is the gain corresponding to the sample covariance $\tilde{\Gamma}$.
  Following the great success of the ensemble Kalman filter algorithm, in a series of papers [TAB+03, BEM01, And01, WH02], the square-root filter framework was (re)discovered. The idea goes back to at least [And68]. We focused the discussion above in Sec. 3.2.4 to the ETKF, but note that one may derive different transformations. For example, the singular evolutive interpolated Kalman (SEIK) filter



(a) Solution. (b) Covariance. (c) Error.

FIG 25. *EnKF on the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3$, $\gamma = 1$ *and* $N = 100$, *see also* `p11.m` *in Section 4.3.5.*



(a) Logscale histograms. (b) Running average root mean square e.

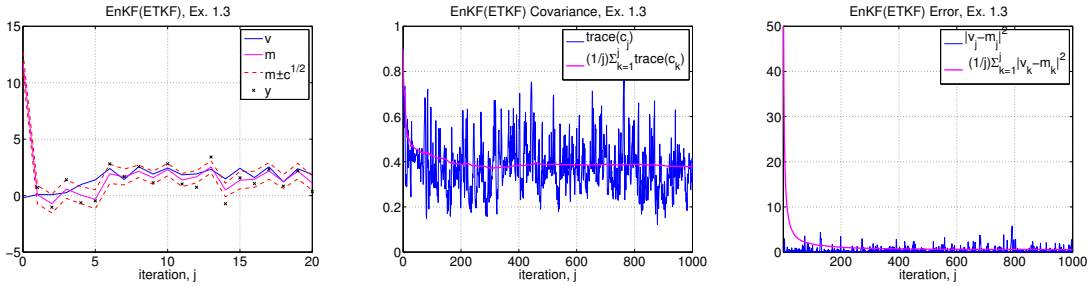
FIG 26. *Convergence of* e *for each filter for the sin map Example 1.3, corresponding to solutions from Figs. 23, 24, 25.*

proceeds by first projecting the ensemble into the $K-1$-dimensional mean-free subspace, and then identifying a $K-1 \times K-1$ matrix transformation, effectively prescribing a $K \times (K-1)$ matrix transformation L_j as opposed to the $K \times K$ rank $K-1$ matrix $T_j^{1/2}$ proposed in ETKF. The former is unique up to unitary transformation, while the latter is unique only up to unitary transformations which have $\mathbf{1}$ as eigenvector. Other alternative transformations may take the forms A_j or $\tilde{K}_j$ such that $X_j = A_j \widehat{X}_j$ or $X_j = (I - \tilde{K}H)\widehat{X}_j$. These are known as the ensemble adjustment Kalman filter (EAKF) and the ensemble square-root filter (ESRF) respectively. See the following papers for details about the ETKF [BEM01], the EAKF [And01], and the ESRF [WH02]. A review of all three is given in [TAB+03]. The SEIK filter was introduced in [PVG98] and is compared with the other square root filters in [NJSH12].

More details regarding tuning of filters through inflation can be found in [AA99, FK05, Jaz70]. An early reference illustrating the benefits and possible implementation of localization is [HM01].

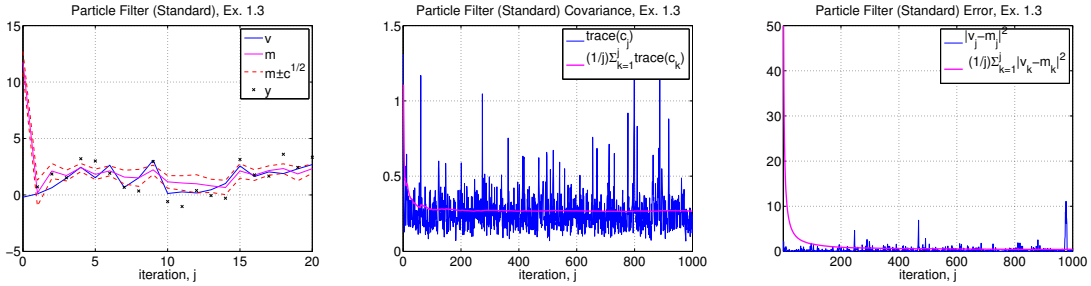
- Section 3.3. In the linear case, the extended Kalman filter of course coincides with the Kalman filter; furthermore, in this case the perturbed observation ensemble Kalman filter reproduces the true posterior distribution in the large particle limit [Eve06]. However the filters introduced in section 3.2 do not produce the correct posterior distribution when applied to general nonlinear problems. The particle filter does recover the true posterior distribution as the number of particles tends to infinity, as we show in Theorem 3.5. This proof is adapted from the very clear exposition in [RvH13].

  For more refined analyses of the convergence of particle filters see, for example, [CD02, DMG01] and references therein. As explained in Remarks 3.6 the constant appearing in the convergence results may



(a) Solution. (b) Covariance. (c) Error.

FIG 27. *ETKF on the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3$, $\gamma = 1$ *and* $N = 100$, *see also* `p12.m` *in Section 4.3.6.*



(a) Solution. (b) Covariance. (c) Error.

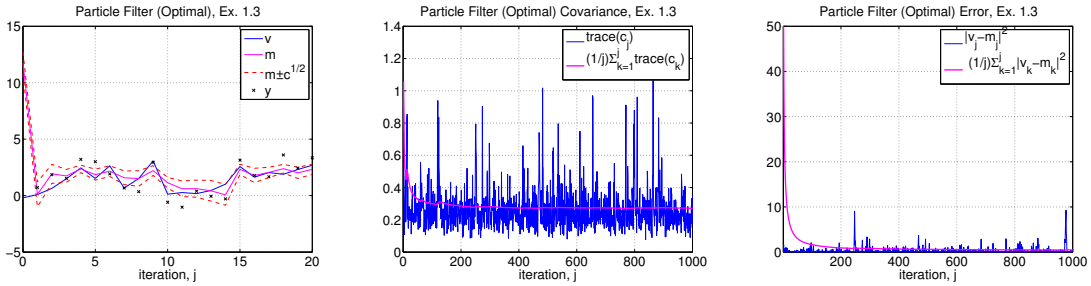
FIG 28. *Particle Filter (standard proposal) on the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3$, $\gamma = 1$ *and* $N = 100$, *see also* `p13.m` *in Section 4.3.7.*

depend exponentially on time if the mixing properties of the transition kernel $\mathbb{P}(dv_j|v_{j-1})$ are poor (the undesirable properties of deterministic dynamics illustrate this). This is also interesting work studying the effect of the dimension [SBBA08]. A proof in the case of Dobrushin ergodic coefficient for transition kernel may be found in [DMG01]; the assumptions there on the transition and observation kernels are very strong, and are generally not satisfied in practice, but studies indicate comparable results may hold under less stringent conditions.

For a derivation and discussion of the optimal proposal, introduced in section 3.3.3, see [DGA00] and references therein. We also mention here the implicit filters developed by Chorin and co-workers [CT09, CT10, CMT10]. These involve solving an implicit nonlinear equation for each particle which includes knowledge of the next set of observed data. This has some similarities to the method proposed in [vL10b] and both are related to the optimal proposal mentioned above.

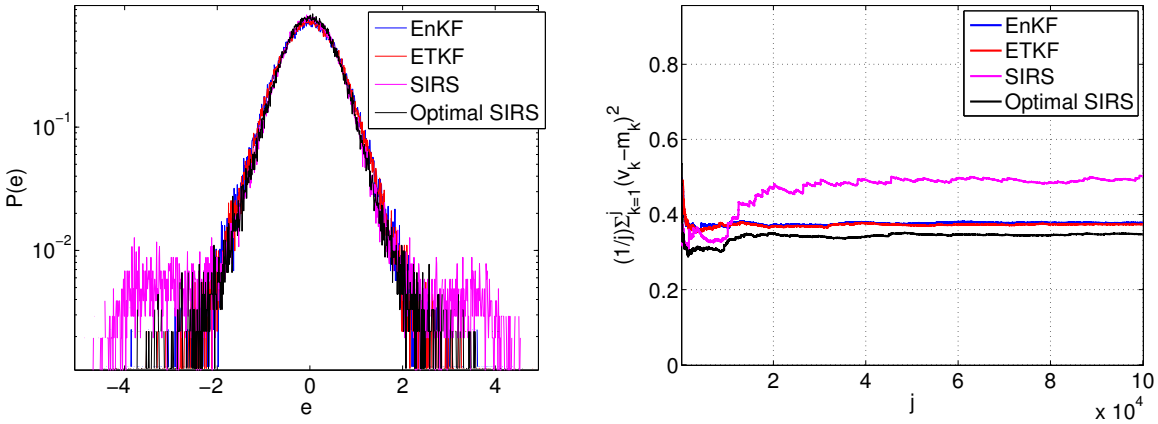
- Section 3.4. The stability of the Kalman filter is a well-studied subject and the book [LR95] provides an excellent overview from the perspective of linear algebra. For extensions to the extended Kalman filter see [Jaz70]. Theorem 3.10 provides a glimpse into the mechanisms at play within 3DVAR, and approximate Gaussian filters in general, in determining stability and accuracy: the incorporation of data can convert unstable dynamical systems, with positive Lyapunov exponents, into contractive non-autnonomous dynamical systems, thereby leading, in the case of small observational noise, to filters which recover the true signal within a small error. This idea was highlighted in [CGTU08] and first studied rigorously for the 3DVAR method applied to the Navier-Stokes equation in [BLL$^+$12]; this work was subsequently generalied to a variety of different models in [PMLvL12, MLPvL13, LSS14]. It is also of note that these analyses of 3DVAR build heavily on ideas developed in [HOT11] for a specialized form of data assimilation in which the observations are noise-free.

  Similar ideas are studied for the perturbed observation EnKF in [KLS13]; in this case it is necessary to introduce a form a variance inflation to get a result analogous to Theorem 3.10.



(a) Solution. (b) Covariance. (c) Error.

FIG 29. *Particle Filter (optimal proposal) on the sin map Example 1.3 with* $\alpha = 2.5$, $\sigma = 0.3$, $\gamma = 1$ *and* $N = 100$, *see also* `p14.m` *in Section 4.3.8.*



(a) Logscale histograms. (b) Running average root mean square e.

FIG 30. *Convergence of* $e = v - m$ *for both versions of EnKF in comparison to the particle filters for the sin map Ex. 1.3, corresponding to solutions from Figs. 25, 27, 28, and 29.*

In the context of filter stability it is important to understand the *optimality* of the true filtering distribution. We observe that all of the filtering algorithms that we have described produce an estimate of the probability distribution $\mathbb{P}(v_j|Y_j)$ that depends only on the data Y_j. There is a precise sense in which the true filtering distribution can be used to find a lower bound on the accuracy that can be achieved by any of these approximate algorithms. We let $\mathbb{E}(v_j|Y_j)$ denote the mean of v_j under the probability distribution $\mathbb{P}(v_j|Y_j)$ and let $E_j(Y_j)$ denote any estimate of the state v_j based only on data Y_j. Now consider all possible random data sets Y_j generated by the model (1.1), (1.2), noting that the randomness is generated by the initial condition v_0 and the noises $\{\xi_j, \eta_j\}$; in particular, conditioning on Y_j to obtain the probability distribution $\mathbb{P}(v_j|Y_j)$ can be thought of as being induced by conditioning on the obervational noise $\{\eta_k\}_{k=1,\dots,j}$. Then $E_j^*(Y_j) := \mathbb{E}(v_j|Y_j)$ minimizes the mean-square error with respect to the random model (1.1), (1.2) [Lue68, Jaz70, Kal60]:

$$\mathbb{E}\|v_j - E_j^*(Y_j)\|^2 \le \mathbb{E}\|v_j - E_j(Y_j)\|^2 \tag{3.50}$$

for all $E_j(Y_j)$. Thus the algorithms we have described can do no better at estimating the state of the system than can be achieved, in principle, from the conditional mean of the state given the data $\mathbb{E}(v_j|Y_j)$. This lower bound holds on average over all instances of the model. An alternative way to view the inequality (3.50) is as a means to providing upper bounds on the true filter. For example, under the conditions of Theorem 3.10 the righthand side of (3.50) is, asymptotically as $j \to \infty$, of size

$\mathcal{O}(\epsilon^2)$; thus we deduce that

$$\limsup_{j\to\infty} \mathbb{E}\|v_j - \mathbb{E}(v_j|Y_j)\|^2 \leq C\epsilon^2.$$

- Section 3.5 We mention here the *rank histogram*. This is another consistency check on the output of ensemble or particle based approximations of the filtering distribution. The idea is to consider scalar observed quantities consisting of generating ordered bins associated to that scalar and then keeping track of the statistics over time of the data y_j with respect to the bins. For example, if one has an approximation of the distribution consisting of N equally-weighted particles, then a rank histogram for the first component of the state consists of three steps, each carried out at each time j. First, add a random draw from the observational noise $N(0, \Gamma)$ to each particle after the prediction phase of the algorithm. Secondly order the particles according to the value of their first component, generating $N-1$ bins between the values of the first component of each particle, and with one extra bin on each end. Finally, rank the current observation y_j between 1 and $N+1$ depending on which bin it lands in. Proceeding to do this at each time j, a histogram of the rank of the observations is obtained. The "spread" of the ensemble can be evaluated using this diagnostic. If the histogram is uniform, then the spread is consistent. If it is concentrated to the center, then the spread is overestimated. If it is concentrated at the edges, then the spread is underestimated. This consitency check on the statistical model used was introduced in [And96] and is widely adopted throughout the data assimilation community.

4. Discrete Time: MATLAB Programs

This chapter is dedicated to illustrating the theory and algorithms, as presented in the previous chapters, through a few short and easy to follow MATLAB programs. These programs are provided for two reasons: (i) for some readers they will form the best route by which to appreciate the details of the algorithms we describe; (ii) for other readers they will be a useful starting point to develop their own codes: whilst ours are not necessarily the optimal implementations of the algorithms discussed in these notes, they have been structured to be simple to understand, to modify and to extend. In particular the code may be readily extended to solve problems more complex than those described in the Examples 1.1-1.7 which we will use for most of our illustrations.

Before getting into details we highlight out a few principles that have been adopted in the programs and in accompanying text of this chapter. First, notation is consistent between programs, and matches the text in the previous sections of the book as far as possible. Second, since many of the elements of the individual programs are repeated, they will be described in detail only in the text corresponding to the program in which they first appear; the short annotations explaining them will be repeated within the programs however. Third, one should always remember that documentation is available at the command line for *any* built-in functions of MATLAB and this can be accessed using the `help` command; for example the documentation for the command `help` can be accessed by typing `help help`.

4.1. Chapter 1 Programs

The programs `p1.m` and `p2.m` used to generate the figures in Chapter 1 are presented in this section. Thus these algorithms simply solve the dynamical system (1.1), and process the resulting data.

4.1.1. p1.m

The first program `p1.m` illustrates how to obtain sample paths from equations (1.1) and (1.3). In particular the program simulates sample paths of the equation

$$u_{j+1} = \alpha \sin(u_j) + \xi_j, \tag{4.1}$$

with $\xi \sim N(0, \sigma^2)$ and $\alpha = 2.5$, both for deterministic ($\sigma = 0$) and stochastic dynamics ($\sigma \neq 0$) corresponding to Example 1.3. In line 5 the variable `J` is defined, which corresponds to the number of forward steps that we will take. The parameters α and σ are set in lines 6-7. The seed for the random number generator is set to `sd`$\in \mathbb{N}$ in line 8 using the command `rng(sd)`. This guarantees the results will be reproduced exactly by running the program with this same `sd`. Different choices of `sd`$\in \mathbb{N}$ will lead to different streams of random numbers used in the program, which may also be desirable in order to observe the effects of different random numbers on the output. This command will be called in the preamble of all of the programs that follow. In line 9, two vectors of length `J` are created named `v` and `vnoise`; after running the program, these two vectors contain the solution for the case of deterministic ($\sigma = 0$) and stochastic dynamics ($\sigma = 0.25$) respectively. After setting the initial conditions in line 10, the desired map is iterated, without and with noise, in lines $12 - 15$. Note that the only difference between the forward iterations of `v` and `vnoise` is the presence of the `sigma*randn` term, which corresponds to the generation of a random variable sampled from $N(0, \sigma^2)$. Lines 17-20, graph the trajectories with and without noise to produce Figure 3. Figures 1,2 and 5 were obtained by simply modifying lines $12 - 15$ of this program, in order to create sample paths for the corresponding Ψ for the other three examples, and Figure 4 was generated from output of this program.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p1.m - behaviour of sin map (Ex. 1.3)
%%% with and without observational noise

J=10000;% number of steps
alpha=2.5;% dynamics determined by alpha
sigma=0.25;% dynamics noise variance is sigma^2
sd=1;rng(sd);% Choose random number seed
v=zeros(J,1);  vnoise=zeros(J,1);% preallocation for saving time
v(1)=1;vnoise(1)=1; %initial conditions

for i=1:J-1
    v(i+1)=alpha*sin(v(i));
    vnoise(i+1)=alpha*sin(vnoise(i))+sigma*randn;
end

figure(1), plot([1:1:J],v),
xlabel('j'), ylabel('v'), title('noise-free dynamics')
figure(2), plot([1:1:J],vnoise),
xlabel('j'), ylabel('v'), title('noisy dynamics')
```

4.1.2. p2.m

The second program presented here, `p2.m`, is designed to visualize the posterior distribution in the case of one dimensional deterministic dynamics. For clarity, the program is separated into three main sections. The `setup` section in lines 5-10 defines the parameters of the problem. The model parameter is now given by `r` in line 6, which determines the dynamics of the forward model, in this case given by the logistic map (1.12)

$$v_j = rv_{j-1}(1 - v_{j-1}). \tag{4.2}$$

The dynamics are taken as deterministic, so the parameter `sigma` does not feature here. The parameter `r`$= 2$ so that the dynamics are not chaotic, as the explicit solution given in Example 1.4 shows. New parameters `m0` and `C0` define the prior distribution $v_0 \sim N(m_0, C_0)$, and `gamma` defines the observational noise $\eta_j \sim N(0, \gamma^2)$.

The `truth` section in lines 14-20 generates the true reference trajectory (or, truth) `vt` in line 18 given by (4.2), as well as the observations `y` in line 19 given by

$$y_j = v_j + \eta_j. \tag{4.3}$$

Note that the index of `y(:,j)` corresponds to observation of `H*v(:,j+1)`. This is due to the fact that the first index of an array in matlab is `j=1`, while the initial condition is v_0, and the first observation is of v_1. So, effectively the indices of `y` are correct as corresponding to the text and Eq. (4.3), but the indices of `v` are one off. The memory for these vectors is *preallocated* in line 14. This is not necessary because MATLAB would simply *dynamically allocate* the memory in its absence, but it would slow down the computations due to the necessity of allocating new memory each time the given array changes size. One may comment this line to observe the effect, which becomes significant when `J` becomes sufficiently large.

The `solution` section after line 24 computes the solution, in this case the point-wise representation of the posterior smoothing distribution on the scalar initial condition. The point-wise values of initial condition are given by the vector `v0`(v_0) defined in line 24. There are many ways to construct such vectors, this convention defines the initial (0.01) and final (0.99) values and a uniform step size 0.0005. One may also use the command `v0=linspace(0.01,0.99,1961)`, defining the *number* of intermediate points $N = 1961$, rather than the stepsize 0.0005. The corresponding vectors of values of `Phi0` (Φ_0), `J0` (J_0), and `I0`(I_0) are computed in lines 29, 32, and 34 for each value of `v0`, as related by the equation

$$I_0(v_0; y) = J_0(v_0) + \Phi_0(v_0; y), \tag{4.4}$$

where $J_0(v_0)$ is the **background** penalization and $\Phi_0(v_0; y)$ is the **model-data misfit** functional given by (1.28b) and (1.28c) respectively. The function $I_0(v_0; y)$ is the negative log-posterior as given in Theorem 1.9. Having obtained $I_0(v_0; y)$ we calculate $\mathbb{P}(v_0|y)$ in lines 37-38, using the formula

$$\mathbb{P}(v_0|y) = \frac{\exp(-I_0(v_0; y))}{\int \exp(-I_0(v_0; y))}. \tag{4.5}$$

The trajectory v corresponding to the given value of v_0 (`v0(i)`) is denoted by `vv` and is replaced for each new value of `v0(i)` in lines 28 and 31 since it is only required to compute `I0`. The command `trapz(v0,exp(-I0))` in line 37 approximates the denominator of the above by the trapezoidal rule, i.e. the summation

$$\texttt{trapz}(\texttt{v0}, \texttt{exp}(-\texttt{I0})) = \sum_{i=1}^{N-1} (\texttt{v0(i+1)} - \texttt{v0(i)}) * (\texttt{I0(i+1)} + \texttt{I0(i)})/2. \tag{4.6}$$

The rest of the program deals with plotting our results and in this instance it coincides with the output of Figure 11b. Again simple modifications of this program were used to produce Figures 10, 12 and 13. Note that `rng(sd)` in line 8 allows us to use the same random numbers every time the file is executed generated with the seed `sd` as described in the previous section 4.1.1. Commenting this line out would result in the creation of data sets y different from the ones used to obtain Figure 11b.

```
clear; set(0,'defaultaxesfontsize',20); format long
%%% p2.m smoothing problem for the deterministic logistic map (Ex. 1.4)
%% setup

J=1000;% number of steps
r=2;% dynamics determined by r
gamma=0.1;% observational noise variance is gamma^2
C0=0.01;% prior initial condition variance
m0=0.7;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed

%% truth

vt=zeros(J+1,1); y=zeros(J,1);% preallocate space to save time
vt(1)=0.1;% truth initial condition
for j=1:J
    % can be replaced by Psi for each problem
    vt(j+1)=r*vt(j)*(1-vt(j));% create truth
    y(j)=vt(j+1)+gamma*randn;% create data
end

%% solution

v0=[0.01:0.0005:0.99];% construct vector of different initial data
Phi0=zeros(length(v0),1); I0=Phi0; J0=Phi0; vv=zeros(J,1);% preallocate space to save time
% loop through initial conditions vv0, and compute log posterior I0(vv0)
for j=1:length(v0)
   vv(1)=v0(j);
   J0(j)=1/2/C0*(v0(j)-m0)^2;  %background penalization
     for i=1:J
        vv(i+1)=r*vv(i)*(1-vv(i));
        Phi0(j)=Phi0(j)+1/2/gamma^2*(y(i)-vv(i+1))^2; %model-data misfit functional
     end
    I0(j)=Phi0(j)+J0(j);
end

constant=trapz(v0,exp(-I0));% approximate normalizing constant
P=exp(-I0)/constant;% normalize posterior distribution

logprior=1/2/C0*(v0-m0).^2;% compute log prior
constant=trapz(v0,exp(-logprior));% approximate normalizing constant
prior=exp(-logprior)/constant;% normalize prior distribution

figure(1),plot(v0,prior,'k','LineWidth',2)
hold on, plot(v0,P,'r--','LineWidth',2), xlabel 'v_0',
legend 'prior' J=10^3
```

4.2. Chapter 2 Programs

The programs `p3.m-p6.m`, used to generate the figures in Chapter 2, are presented in this section. Various MCMC algorithms used to sample the posterior smoothing distribution are given. Optimization algorithms used to obtain solution of the 4DVAR and w4DVAR variational methods are also given. Although our general development of MCMC methods concerned a notation of u for the state of the chain and w for the proposal, the programs described here use the letter v for the state of the Markov chain, to keep the connection with the underlying dynamics model.

4.2.1. p3.m

The MATLAB program `p3.m` is the first of the Markov Chain Monte Carlo (MCMC) algorithms given. It contains an implementation of the Random walk Metropolis (RWM) Algorithm from section 2.1.2 to determine the posterior distribution on the initial condition arising from the deterministic logistic map of Example 1.4 given by (4.2). Note that in this case, since the the underlying dynamics are deterministic and hence completely determined by the initial condition, the RMW algorithm will provide samples from a probability distribution on $\mathbb{R}$.

As in program `p2.m` the program is divided into 3 sections `setup` where parameters are defined, `truth` where the truth and data are generated, and `solution` where the solution is computed, this time monte-carlo samples from the posterior smoothing distribution. The parameters in lines 5-10 and the true solution (here taken as only the initial condition, rather than the trajectory it gives rise to) `vt` in line 14 are taken to be the same as those used to generate Figure 13. The temporary vector `vv` generated in line 19 is the trajectory corresponding to the truth (`vv(1)=vt` in line 14), and used to calculate the observations `y` in line 20. The true value `vt` will also be used as the initial sample in the MCMC for this and subsequent MCMC programs. This scenario is not possible in the case that the data is not simulated, however it is useful in the case that the data is simulated as it is here, because it can reduce the *burn-in* time, i.e. the time necessary for the current sample in the chain to reach the target distribution, or the high-probability region of the state-space. Therefore, `I0` will be necessary to compute the acceptance probability as described below. It is computed in lines 15-23 exactly as in lines 25-34 of program `p2.m`, as described around (4.4).

In the `solution` section some additional MCMC parameters are defined.In line 28 the number of samples is set to `N` $=10^5$.For the parameters and specific data used here, this is sufficient for convergence of the Markov chain. In line 30 the step-size parameter `beta` is pre-set such that the algorithm for this particular posterior distribution has a reasonable acceptance probability, or ratio of accepted vs. rejected moves. A general rule of thumb for this is that it should be somewhere around 0.5, to ensure that the algorithm is not too correlated because of high rejection rate (acceptance probability near zero) and that it is not too correlated because of small moves (acceptance probability near one). The vector `V` defined in line 29 will save all of the samples. This is an example where pre-allocation is *very* important. Try using the commands `tic` and `toc` before and respectively after the loop in lines 33-50 in order to time the chain both with and without pre-allocation. [5] In line 34 a move is proposed according to the equation

$$w^{(k)} = v^{(k-1)} + \beta\iota^{(k-1)}$$

where v(`v`) is the current state of the chain (initially taken to be equal to the true initial condition v_0), $\iota^{(k-1)}$=`randn` is an i.i.d. standard normal, and `w` represents $w^{(k)}$. Indices are not used for `v` and `w` because they will be replaced at each iteration.

The temporary variable `vv` is again used for the trajectory corresponding to $w^{(k)}$ as a vehicle to compute the value the proposed $I(w^{(k)}; y)$, denoted in line 42 by `I0prop = J0prop + Phiprop`. In lines 44-46 the decision to accept or reject the proposal is made based on the acceptance probability

$$a(v^{(k-1)}, w^{(k)}) = 1 \wedge \exp(I_0(v^{(k-1)}; y) - I_0(w^{(k)}; y)).$$

[5] In practice, one may often choose to collect certain statistics from the chain "on-the-fly" rather than saving every sample, particularly if the state-space is high-dimensional where one pays a premium price in memory for each sample.

In practice this corresponds to drawing a uniform random number `rand` and replacing `v` and `I0` in line 45 with `w` and `I0prop` if `rand<exp(I0-I0prop)` in line 44. The variable `bb` is incremented if the proposal is accepted, so that the running ratio of accepted moves `bb` to total steps `n` can be computed in line 47. This approximates the average acceptance probability. The current sample $v^{(k)}$ is stored in line 48. Notice that here one could replace `v` by `V(n-1)` in line 34, and by `V(n)` in line 45, thereby eliminating `v` and line 48, and letting `w` be the only temporary variable. However, the present construction is favorable because, as mentioned above, in general one may not wish to save every sample.

The samples `V` are used in lines 51-53 in order to visualize the posterior distribution. In particular, bins of width `dx` are defined in line 51, and the command `hist` is used in line 52. `Z = hist(V,v0)` means first the real-number line is split into M bins with centers defined according to `v0(i)` for $i = 1, \ldots, M$, with the first and last bin corresponding to the negative, respectively positive, half-lines. Second, `Z(i)` counts the number of `k` for which `V(k)` is in the bin with center determined by `v0(i)`. Again, `trapz` (4.6) is used to compute the normalizing constant in line 53, directly within the plotting command. The choice of the location of the histogram bins allows for a direct comparison with the posterior distribution calculated from the MATLAB program `p2.m` by directly evaluation of $I_0(v; y)$ defined in 4.4 for different values of initial conditions v. This output is then compared with the corresponding output of `p2.m` for the same parameters in Figure 14.

```
clear; set(0,'defaultaxesfontsize',20); format long
%%% p3.m MCMC RWM algorithm for logistic map (Ex. 1.4)
%% setup

J=5;% number of steps
r=4;% dynamics determined by alpha
gamma=0.2;% observational noise variance is gamma^2
C0=0.01;% prior initial condition variance
m0=0.5;% prior initial condition mean
sd=10;rng(sd);% Choose random number seed

%% truth

vt=0.3;vv(1)=vt;% truth initial condition
J0=1/2/C0*(vt-m0)^2; %background penalization
Phi0=0; %initialization model-data misfit functional
for j=1:J
    % can be replaced by Psi for each problem
    vv(j+1)=r*vv(j)*(1-vv(j));% create truth
    y(j)=vv(j+1)+gamma*randn;% create data
    Phi0=Phi0+1/2/gamma^2*(y(j)-vv(j+1))^2;% compute model-data misfit functional
end
I0=J0+Phi0; %compute log posterior of the truth

%% solution
% Markov Chain Monte Carlo: N forward steps of the
% Markov Chain on R (with truth initial condition)
N=1e5;% number of samples
V=zeros(N,1);% preallocate space to save time
beta=0.05;% step-size of random walker
v=vt;% truth initial condition (or else update I0)
n=1; bb=0; rat(1)=0;
while n<=N
    w=v+sqrt(2*beta)*randn;% propose sample from random walker
    vv(1)=w;
    J0prop=1/2/C0*(w-m0)^2; %background penalization
    Phi0prop=0;
    for i=1:J
            vv(i+1)=r*vv(i)*(1-vv(i));
            Phi0prop=Phi0prop+1/2/gamma^2*(y(i)-vv(i+1))^2;
    end
    I0prop=J0prop+Phi0prop;  %compute log posterior of the proposal

    if rand<exp(I0-I0prop)% accept or reject proposed sample
        v=w; I0=I0prop; bb=bb+1;% update the Markov chain
    end
    rat(n)=bb/n;% running rate of acceptance
    V(n)=v;% store the chain
    n=n+1;
end
dx=0.0005; v0=[0.01:dx:0.99];
Z=hist(V,v0);% construct the posterior histogram
figure(1), plot(v0,Z/trapz(v0,Z),'k','Linewidth',2)% visualize the posterior
```

4.2.2. p4.m

The MATLAB program `p4.m` contains an implementation of the independence dynamics sampler for stochastic dynamics, as introduced in section 2.1.3. Thus the posterior distribution is on the entire signal $\{v_j\}_{j\in\mathbb{J}}$. The forward model in this case is from Example 1.3, given by (4.1). The smoothing distribution $\mathbb{P}(v|Y)$ is therefore over the state-space $\mathbb{R}^{J+1}$.

The sections `setup`, `truth`, and `solution` are defined exactly as in program 4.2.1, except this time the smoothing distribution is over the entire path. Since the state-space is now the path-space, rather than the initial condition as it was in program 4.2.1, the truth `vt`$\in \mathbb{R}^{J+1}$ is now a vector. Its initial condition is taken as a draw from $N(m_0, C_0)$ in line 16, and the trajectory is computed in line 19, so that at the end `vt`$\sim \rho_0$. Again, $v^\dagger$ (`vt`) will be the initial condition in the Markov chain and so $\Phi(v^\dagger; y)$ is computed in line 22. Recall from section 2.1.3 that only $\Phi(\cdot; y)$ is required to compute the acceptance probability in this algorithm.

Notice that the collection of samples `V`$\in \mathbb{R}^{N\times J+1}$ pre-allocated in line 29 is already becoming quite large in this case, illustrating the memory issue which arises when the state-space and number of samples increase.

The current state of the chain $v^{(k)}$, and the value of $\Phi(v^{(k)}; y)$ are again denoted `v` and `Phi`, while the proposal $w^{(k)}$ and the value of $\Phi(w^{(k)}; y)$ are again denoted `w` and `Phiprop`, as in program 4.2.1. As discussed in section 2.1.3, the proposal $w^{(k)}$ is an independent sample from the prior distribution ρ_0, similarly to $v^\dagger$, and it is constructed in lines 33-38. The acceptance probability used in line 39 is now

$$a(v^{(k-1)}, w^{(k)}) = 1 \wedge \exp(\Phi(v^{(k-1)}; y) - \Phi(w^{(k)}; y)),$$

The rest of the program is the same as 4.2.1. The outputs of this program are used to plot Figures 15, 16, and 17. Note that in the case of Figure 17, we have used $N = 10^8$ samples.

```
clear; set(0,'defaultaxesfontsize',20); format long
%%% p4.m MCMC INDEPENDENCE DYNAMICS SAMPLER algorithm
%%% for sin map (Ex. 1.3) with noise
%% setup

J=10;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=1;% dynamics noise variance is sigma^2
C0=1;% prior initial condition variance
m0=0;% prior initial condition mean
sd=10;rng(sd);% Choose random number seed

%% truth

vt(1)=m0+sqrt(C0)*randn;% truth initial condition
Phi=0;
for j=1:J
    vt(j+1)=alpha*sin(vt(j))+sigma*randn;% create truth
    y(j)=vt(j+1)+gamma*randn;% create data
    % calculate log likelihood of truth, Phi(v;y) from (1.11)
    Phi=Phi+1/2/gamma^2*(y(j)-vt(j+1))^2;
end

%% solution
% Markov Chain Monte Carlo: N forward steps of the
% Markov Chain on R^{J+1} with truth initial condition
N=1e5;% number of samples
V=zeros(N,J+1);% preallocate space to save time
v=vt;% truth initial condition (or else update Phi)
n=1; bb=0; rat(1)=0;
while n<=N
    w(1)=sqrt(C0)*randn;% propose sample from the prior distribution
    Phiprop=0;
    for j=1:J
    w(j+1)=alpha*sin(w(j))+sigma*randn;% propose sample from the prior distribution
    Phiprop=Phiprop+1/2/gamma^2*(y(j)-w(j+1))^2;% compute likelihood
    end
    if rand<exp(Phi-Phiprop)% accept or reject proposed sample
       v=w; Phi=Phiprop; bb=bb+1;% update the Markov chain
    end
   rat(n)=bb/n;% running rate of acceptance
   V(n,:)=v;% store the chain
   n=n+1;
end
```

4.2.3. p5.m

The independence dynamics sampler 4.2.2 may be very inefficient as typical random draws from the dynamics may be unlikely to fit the data, and will hence be rejected. The fifth MATLAB program `p5.m` gives an implementation of the pCN algorithm from section 2.1.3 which is designed to overcome this issue.

This program is almost identical to `p4.m`, and so only the points at which it differs will be described. First, since the acceptance probability is given by

$$a(v^{(k-1)}, w^{(k)}) = 1 \wedge \exp(\Phi(v^{(k-1)}; y) - \Phi(w^{(k)}; y) + G(v^{(k-1)}) - G(w^{(k)})),$$

the quantity

$$G(u) = \sum_{j=0}^{J-1} \Big(\frac{1}{2} |\Sigma^{-\frac{1}{2}} \Psi(u_j)|^2 - \langle \Sigma^{-\frac{1}{2}} u_{j+1}, \Sigma^{-\frac{1}{2}} \Psi(u_j) \rangle \Big)$$

will need to be computed, both for $v^{(k)}$, denoted by `v` in lines 32 and 45 where its value is denoted by `G` ($v^{(0)} = v^\dagger$ and $G(v^\dagger)$ is computed in line 21), and for $w^{(k)}$, denoted by `w` in line 37, where its value is denoted by `Gprop` in line 40. This program is used to generate Figure 18

As discussed in section 2.1.3 the proposal $w^{(k)}$ is given by

$$w^{(k)} = m + (1 - \beta^2)^{\frac{1}{2}} (v^{(k-1)} - m) + \beta \iota^{(k-1)},$$

where $\iota^{(k-1)} \sim N(0, C)$ are i.i.d. and denoted by `iota` in line 36. C is the covariance of the Gaussian measure π_0 given in Equation (2.7) corresponding to the case of trivial dynamics $\Psi = 0$, and m is the mean of π_0. The value of m is given by `m` in line 34.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p5.m MCMC pCN algorithm for sin map (Ex. 1.3) with noise
%% setup

J=10;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=1;% dynamics noise variance is sigma^2
C0=1;% prior initial condition variance
m0=0;% prior initial condition mean
sd=10;rng(sd);% Choose random number seed

%% truth

vt(1)=sqrt(C0)*randn;% truth initial condition
G=0;Phi=0;
for j=1:J
    vt(j+1)=alpha*sin(vt(j))+sigma*randn;% create truth
    y(j)=vt(j+1)+gamma*randn;% create data
    % calculate log density from (1.--)
    G=G+1/2/sigma^2*((alpha*sin(vt(j)))^2-2*vt(j+1)*alpha*sin(vt(j)));
    % calculate log likelihood phi(u;y) from (1.11)
    Phi=Phi+1/2/gamma^2*(y(j)-vt(j+1))^2;
end

%% solution
% Markov Chain Monte Carlo: N forward steps of the
% Markov Chain on R^{J} with truth initial condition
N=1e5;% number of samples
V=zeros(N,J+1);
beta=0.2;% step-size of pCN walker
v=vt;% truth initial condition (or update G + Phi)
n=1; bb=0; rat=0;
m=[m0;zeros(1,J)];
while n<N
    iota=[sqrt(C0)*randn,sigma*randn(1,J)];% Gaussian prior sample
    w=m+sqrt(1-beta^2)*(v-m)+beta*iota;% propose sample from the pCN walker
    Gprop=0;Phiprop=0;
    for j=1:J
        Gprop=Gprop+1/2/sigma^2*((alpha*sin(w(j)))^2-2*w(j+1)*alpha*sin(w(j)));
        Phiprop=Phiprop+1/2/gamma^2*(y(j)-w(j+1))^2;
    end

    if rand<exp(Phi-Phiprop+G-Gprop)% accept or reject proposed sample
        v=w;Phi=Phiprop;G=Gprop;bb=bb+1;% update the Markov chain
    end
       rat(n)=bb/n;% running rate of acceptance
       V(n,:)=v;% store the chain
       n=n+1;
end
%plot samples, truth, observations
figure;nn=round(rand*N);plot([0:J],V(nn,:));hold;plot([0:J],vt,'r','Linewidth',2);
plot([1:J],y,'g','Linewidth',2);for b=1:1e3; nn=round(rand*N);plot([0:J],V(nn,:));end
plot([0:J],vt,'r','Linewidth',2);plot([1:J],y,'g','Linewidth',2);hold
xlabel('j');legend('Posterior samples','truth','observations')
% trace plot and histogram
jay=J/2;[rho,ex]=hist(V(:,jay),100);figure;plot(V(1:1e4,jay));hold
plot(rho*5e3/max(rho),ex,'r');title(strcat('v_{',num2str(jay-1),'}|Y_{',num2str(J),'}'))
legend(strcat('v^{(n)}_{',num2str(jay-1),'}|Y_{',num2str(J),'}'),'histogram');xlabel('n')
```

4.2.4. p6.m

The next MATLAB program `p6.m` contains an implementation of the weak constrained variational algorithm w4DVAR discussed in section 2.2. Again the forward model will be given by Example 1.11 (4.1). The `setup` and `truth` sections are similar to the previous programs, except G, Φ, etc. need not be computed here. Furthermore, note that this program is written as a function rather than as a script like the previous programs were. This choice was made so that the MATLAB built-in function `fminsearch` can be used for optimization in the `solution` section, and the program can still be self-contained. To use this built-in function it is necessary to define an *auxiliary* objective function `I` to be optimized. The function `fminsearch` can be used within a script, but the auxiliary function would then have to be written separately, so we cannot avoid functions altogether unless we write the optimization algorithm by hand. We avoid the latter in order not to divert the focus of this text from the data assimilation problem, and algorithms to solve it, to the problem of how to optimize an objective function.

The auxiliary objective function `I` in this case is $I(\cdot; y)$ from equation (1.27) given by

$$\mathsf{I}(\cdot; y) = \mathsf{J}(\cdot) + \Phi(\cdot; y), \tag{4.7}$$

where

$$\mathsf{J}(u) := \frac{1}{2}\left|C_0^{-\frac{1}{2}}(u_0 - m_0)\right|^2 + \sum_{j=0}^{J-1} \frac{1}{2}\left|\Sigma^{-\frac{1}{2}}\big(u_{j+1} - \Psi(u_j)\big)\right|^2, \tag{4.8}$$

and

$$\Phi(u; y) = \sum_{j=0}^{J-1} \frac{1}{2}\left|\Gamma^{-\frac{1}{2}}\big(y_{j+1} - h(u_{j+1})\big)\right|^2. \tag{4.9}$$

It is defined in lines 37-44. It takes as inputs (`u,y,sigma,gamma,alpha,m0,C0,J`), and gives output `out`$= I(u; y)$ where $u \in R^{J+1}$ (given all the other parameters in its definition – the issue of identifying the input to be optimized over is discussed also below).

The initial guess for the optimization algorithm `uu` is taken as a standard normal random vector over $\mathbb{R}^{J+1}$ in line 27. In line 24, a standard normal random matrix of size 100^2 is drawn and thrown away. This is so one can easily change the input, e.g. to `randn(z)` for `z`$\in \mathbb{N}$, and induce different random initial vectors `uu` for the optimization algorithm, while keeping the data fixed by the random number seed `sd` set in line 12. One may also use the truth `vt` as initial guess by uncommenting line 28. In particular, if the output of the minimisation procedure is different for different initial conditions, then it is possible that the objective function $I(\cdot; y)$ has multiple minima, and hence the posterior distribution $\mathbb{P}(\cdot|y)$ is multi-modal. As we have already seen in Figure 19 this is certainly true even in the case of scalar deterministic dynamics, when the underlying map gives rise to a chaotic flow.

The MATLAB optimization function `fminsearch` is called in line 30. The *function handle* command `@(u)I(u,` $\cdots$`)` is used to tell `fminsearch` that the objective function `I` is to be considered a function of `u`, even though it may take other parameter values as well (in this case, `y,sigma,gamma,alpha,m0,C0`, and `J`). The outputs of `fminsearch` are the value `vmap` such that `I(vmap)` is minimum, the value `fval = I(vmap)`, and the `exit flag` which takes the value 1 if the algorithm has converged. The reader is encouraged to use the `help` command for more details on this and other MATLAB functions used in the notes. The results of this minimisation procedure are plotted in lines 33-34 together with the true value $v^\dagger$ as well as the data y. In Figure 20 such results are presented, including two minima which were found with different initial conditions.

```
function this=p6
clear;set(0,'defaultaxesfontsize',20);format long
%%% p6.m weak 4DVAR for sin map (Ex. 1.3)
%% setup

J=5;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1e0;% observational noise variance is gamma^2
sigma=1;% dynamics noise variance is sigma^2
C0=1;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed

%% truth

vt(1)=sqrt(C0)*randn;% truth initial condition
for j=1:J
    vt(j+1)=alpha*sin(vt(j))+sigma*randn;% create truth
    y(j)=vt(j+1)+gamma*randn;% create data
end

%% solution

    randn(100);% try uncommenting or changing the argument for different
               % initial conditions -- if the result is not the same,
               % there may be multimodality (e.g. 1 & 100).
    uu=randn(1,J+1);% initial guess
    %uu=vt;     % truth initial guess option

[vmap,fval,exitflag]=fminsearch(@(u)I(u,y,sigma,gamma,alpha,m0,C0,J),uu)% solve with blackbox
% exitflag=1 ==> convergence

figure;plot([0:J],vmap,'Linewidth',2);hold;plot([0:J],vt,'r','Linewidth',2)
plot([1:J],y,'g','Linewidth',2);hold;xlabel('j');legend('MAP','truth','y')

%% auxiliary objective function definition
function out=I(u,y,sigma,gamma,alpha,m0,C0,J)

Phi=0;JJ=1/2/C0*(u(1)-m0)^2;
for j=1:J
    JJ=JJ+1/2/sigma^2*(u(j+1)-alpha*sin(u(j)))^2;
    Phi=Phi+1/2/gamma^2*(y(j)-u(j+1))^2;
end
out=Phi+JJ;
```

4.3. Chapter 3 Programs

The programs `p7.m-p14.m`, used to generate the figures in Chapter 3, are presented in this section. Various filtering algorithms used to sample the posterior filtering distribution are given, involving both Gaussian approximation and particle approximation. Since these algorithms are run for very large times (large J), they will only be divided in two sections, `setup` in which the parameters are defined, and `solution` in which *both* the truth and observations are generated, *and* the online assimilation of the current observation into the filter solution is performed. The generation of truth can be separated into a `truth` section as in the previous sections, but two loops of length J would be required, and loops are notoriously inefficient in MATLAB , so the present format is preferred. These programs are very similar, and their output is also similar, giving rise to Figures 21-30. With the exception of `p7.m` and `p8.m`, the forward model is given by Example 1.11 (4.1), and the output is identical, given for `p9.m` through `p14.m` in Figures 23-25 and 27-29. Figures 26 and 30 compare the filters from the other Figures. `p7.m` features a two-dimensional linear forward model, and `p8.m` features the forward model from Example 1.12 (4.2).

4.3.1. p7.m

The first filtering MATLAB program is `p7.m` which contains an implementation of the Kalman Filter applied to Example 1.2:

$$v_{j+1} = Av_j + \xi_j, \quad \text{with} \quad A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

and observed data given by

$$y_{j+1} = Hv_{j+1} + \eta_{j+1}$$

with $H = (1, 0)$ and Gaussian noise. Thus only the first component of v_j is observed.

The parameters and initial condition are defined in the `setup` section, lines 3-19. The vectors `v`, `m` $\in \mathbb{R}^{N\times J}$, `y`$\in \mathbb{R}^J$, and `c` $\in \mathbb{R}^{N\times N\times J}$ are preallocated to hold the truth, mean, observations, and covariance over the `J` observation times defined in line 5. In particular, notice that the true initial condition is drawn from $N(m_0, C_0)$ in line 16, where $m_0 = 0$ and $C_0 = 1$ are defined in lines 10-11. The initial *estimate* of the distribution is defined in lines 17-18 as $N(m_0', C_0')$, where $m_0' \sim N(0, 100I)$ and $C_0' = 100C_0$ so that the code may test the ability of the filter to lock onto the true distribution given a poor initial estimate.

The main `solution` loop then follows in lines 20-33. The truth `v` and the data that are being assimilated `y` are sequentially generated within the loop, in lines 24-25. The filter prediction step, in lines 27-28, consists of computing $\widehat{m}_j$ and $\widehat{C}_j$ as defined in (3.3) and (3.4) respectively:

$$\widehat{m}_j = Am_{j-1}, \quad \widehat{C}_j = AC_{j-1}A^T + \Sigma,$$

Notice that indices are not used for the transient variables `mhat` and `chat` representing $\widehat{m}_j$ and $\widehat{C}_j$ because they will not be saved from one iteration to the next. In lines 30-33 we implement the analysis formulae for the Kalman filter from Corollary 3.2. In particular, the innovation (1.36) between the observation of the estimated mean and the actual observation is first computed in line 30

$$d_j = y_j - H\widehat{m}_j. \tag{4.10}$$

Again `d` which represents d_j does not have any index for the same reason as above. Next, the Kalman gain defined in Corollary 3.2 is computed in line 31

$$K_j = \widehat{C}_j H^T (H\widehat{C}_j H^T + \Gamma)^{-1}. \tag{4.11}$$

Once again index is not used for the transient variable `K` representing K_j. Notice the "forward slash" `/` is used to compute `B/A=B` $\mathtt{A}^{-1}$. This is an internal function of MATLAB which will analyze the matrices `B` and `A` to determine the best method for inversion, for example if `A` is banded then –, with Gaussian elimination

as last resort. The reverse operation can be done with the "backslash" \ , i.e. `B\A=B^-1A`. The update given in Corollary 3.2 is completed in lines 30-32 with the equations

$$m_j = \widehat{m}_j + K_j d_j \quad \text{and} \quad C_j = (I - K_j H)\widehat{C}_j. \tag{4.12}$$

Finally, in lines 36-50 the outputs of the program are used to plot the mean and the covariance as well as the mean square error of the filter as functions of the iteration number j, as shown in Figure 21.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p7.m Kalman Filter, Ex. 1.2
%% setup

J=1e3;% number of steps
N=2;%dimension of state
I=eye(N);% identity operator
gamma=1;% observational noise variance is gamma^2*I
sigma=1;% dynamics noise variance is sigma^2*I
C0=eye(2);% prior initial condition variance
m0=[0;0];% prior initial condition mean
sd=10;rng(sd);% Choose random number seed
A=[0 1;-1 0];% Dynamics determined by A

m=zeros(N,J);v=m;y=zeros(J,1);c=zeros(N,N,J);% pre-allocate
v(:,1)=m0+sqrtm(C0)*randn(N,1);% initial truth
m(:,1)=10*randn(N,1);% initial mean/estimate
c(:,:,1)=100*C0;% initial covariance
H=[1,0];% observation operator

%% solution % assimilate!

for j=1:J
    v(:,j+1)=A*v(:,j) + sigma*randn(N,1);% truth
    y(j)=H*v(:,j+1)+gamma*randn;% observation

    mhat=A*m(:,j);% estimator predict
    chat=A*c(:,:,j)*A'+sigma^2*I;% covariance predict

    d=y(j)-H*mhat;% innovation
    K=(chat*H')/(H*chat*H'+gamma^2);% Kalman gain
    m(:,j+1)=mhat+K*d;% estimator update
    c(:,:,j+1)=(I-K*H)*chat;% covariance update
end

figure;js=21;plot([0:js-1],v(2,1:js));hold;plot([0:js-1],m(2,1:js),'m');
plot([0:js-1],m(2,1:js)+reshape(sqrt(c(2,2,1:js)),1,js),'r--');
plot([0:js-1],m(2,1:js)-reshape(sqrt(c(2,2,1:js)),1,js),'r--');
hold;grid;xlabel('iteration, j');
title('Kalman Filter, Ex. 1.2');

figure;plot([0:J],reshape(c(1,1,:)+c(2,2,:),J+1,1));hold
plot([0:J],cumsum(reshape(c(1,1,:)+c(2,2,:),J+1,1))./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');axis([1 1000 0 50]);
title('Kalman Filter Covariance, Ex. 1.2');

figure;plot([0:J],sum((v-m).^2));hold;
plot([0:J],cumsum(sum((v-m).^2))./[1:J+1],'m','Linewidth',2);grid
hold;xlabel('iteration, j');axis([1 1000 0 50]);
title('Kalman Filter Error, Ex. 1.2')
```

4.3.2. p8.m

The MATLAB program is `p8.m` contains an implementation of the 3DVAR method applied to the chaotic logistic map of Example 1.4 (4.2) for $r = 4$.

As in the previous section, the parameters and initial condition are defined in the `setup` section, lines 3-16. In particular, notice that the truth initial condition `v(1)` and initial mean `m(1)`, are now initialized in lines 12-13 with a *uniform* random number using the command `rand`, so that they are in the interval $[0, 1]$ where the model is well-defined. Indeed the solution will eventually become unbounded if initial conditions are chosen outside this interval. With this in mind, we set the dynamics noise `sigma = 0` in line 8, i.e. deterministic dynamics, so that the true dynamics themselves do not go unbounded.

Notice the small stabilization parameter η (`eta`) from Example 3.12 is set in line 14, representing the ratio in uncertainty of the observations vs. the model, or equivalently our trust of the model over the observations: $\eta = 0$ means the model is irrelevant while $\eta \to \infty$ would mean the observations are irrelevant. This then gives rise to the constant scalar covariance `C` and resultant constant scalar gain `K` not to be confused with the changing K_j in (4.11), temporarily defined by `K` in line 31 of `p7.m`.

The main `solution` loop follows in lines 20-33. Up to the different forward model, lines 21-22, 24, and 26-27 of this program are identical to lines 24-25, 27, 30, and 32 of `p7.m` described in section 4.3.1. The only other difference is that the covariance updates are not here because of the constant covariance assumption underlying the 3DVAR algorithm.

The 3DVAR filter may in principle generate estimated mean `mhat` outside $[0, 1]$, because of the noise in the data. In order to flag potential unbounded trajectories of the filter an extra stopping criteria is included in lines 29-32. As a fun exercise, try setting `sigma`$\neq 0$ in line 8. Then the signal will eventually go unbounded regardless of how small the noise variance is chosen. In this case the estimate will surely blowup while tracking the unbounded signal. Otherwise, if η is chosen appropriately so as to stabilize the filter it is extremely unlikely that the estimate will ever blowup. Finally, similarly to `p7.m`, in the last lines of the program we use the outputs of the program in order to produce Figure 22, namely plotting the mean and the covariance as well as the mean square error of the filter as functions of the iteration number j.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p8.m 3DVAR Filter, deterministic logistic map (Ex. 1.4)
%% setup

J=1e3;% number of steps
r=4;% dynamics determined by r
gamma=1e-1;% observational noise variance is gamma^2
sigma=0;% dynamics noise variance is sigma^2
sd=10;rng(sd);% Choose random number seed

m=zeros(J,1);v=m;y=m;% pre-allocate
v(1)=rand;% initial truth, in [0,1]
m(1)=rand;% initial mean/estimate, in [0,1]
eta=2e-1;% stabilization coefficient 0 < eta << 1
C=gamma^2/eta;H=1;% covariance and observation operator
K=(C*H')/(H*C*H'+gamma^2);% Kalman gain

%% solution % assimilate!

for j=1:J
    v(j+1)=r*v(j)*(1-v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    mhat=r*m(j)*(1-m(j));% estimator predict

    d=y(j)-H*mhat;% innovation
    m(j+1)=mhat+K*d;% estimator update

    if norm(mhat)>1e5
        disp('blowup!')
        break
    end
end
js=21;% plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(C),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(C),'r--');hold;grid;xlabel('iteration, j');
title('3DVAR Filter, Ex. 1.4')

figure;plot([0:J],C*[0:J].^0);hold
plot([0:J],C*[0:J].^0,'m','Linewidth',2);grid
hold;xlabel('iteration, j');title('3DVAR Filter Covariance, Ex. 1.4');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('3DVAR Filter Error, Ex. 1.4')
```

4.3.3. p9.m

A variation of program `p8.m` is given by `p9.m`, where the 3DVAR filter is implemented for Example 1.3 given by (4.1). Indeed the remaining programs of this section will all be for the same Example 1.3 so this will not be mentioned again. In this case, the initial condition is again taken as a draw from the prior $N(m_0, C_0)$ as in `p7.m`, and the initial mean estimate is taken again as $m_0' \sim N(0, 100I)$ so that the code may test the ability of the filter to lock onto the signal given a poor initial estimate. Furthermore, for this problem there is no need to introduce the stopping criteria present in the case of `p8.m` since the underlying deterministic dynamics are dissipative. The output of this program is shown in Figure 23.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p9.m 3DVAR Filter, sin map (Ex. 1.3)
%% setup

J=1e3;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=3e-1;% dynamics noise variance is sigma^2
C0=9e-2;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed

m=zeros(J,1);v=m;y=m;% pre-allocate
v(1)=m0+sqrt(C0)*randn;% initial truth
m(1)=10*randn;% initial mean/estimate
eta=2e-1;% stabilization coefficient 0 < eta << 1
c=gamma^2/eta;H=1;% covariance and observation operator
K=(c*H')/(H*c*H'+gamma^2);% Kalman gain

%% solution % assimilate!

for j=1:J
    v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    mhat=alpha*sin(m(j));% estimator predict

    d=y(j)-H*mhat;% innovation
    m(j+1)=mhat+K*d;% estimator update

end

js=21;% plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(c),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(c),'r--');hold;grid;xlabel('iteration, j');
title('3DVAR Filter, Ex. 1.3')

figure;plot([0:J],c*[0:J].^0);hold
plot([0:J],c*[0:J].^0,'m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('3DVAR Filter Covariance, Ex. 1.3');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('3DVAR Filter Error, Ex. 1.3')
```

4.3.4. p10.m

The next MATLAB program is `p10.m`. This program contains an implementation of the extended Kalman Filter.This program is essentially the same as of `p7.m`, except with a different forward model. Since the dynamics are scalar, the observation operator is defined as `H`$= 1$ in line 16. The predicting covariance $\widehat{C}_j$ is not given in closed form as it was for the linear problem `p7.m`. Instead, as described in section 3.2.2, it is approximated using the *linearization* of the forward map around m_j, in line 26

$$\widehat{C}_j = (\alpha \cos(m_{j-1}))\, C_{j-1}\, (\alpha \cos(m_{j-1}))\,.$$

The poor initial estimate of the distribution in this (in lines 15-16) and subsequent programs is always given as $N(m_0', C_0')$, where $m_0' \sim N(0, 100I)$ and $C_0' = 10C_0$. This will not be iterated again. The output of this program is shown in Figure 24.

```
1  clear;set(0,'defaultaxesfontsize',20);format long
2  %%% p10.m  Extended Kalman Filter, sin map (Ex. 1.3)
3  %% setup
4
5  J=1e3;% number of steps
6  alpha=2.5;% dynamics determined by alpha
7  gamma=1;% observational noise variance is gamma^2
8  sigma=3e-1;% dynamics noise variance is sigma^2
9  C0=9e-2;% prior initial condition variance
10 m0=0;% prior initial condition mean
11 sd=1;rng(sd);% Choose random number seed
12
13 m=zeros(J,1);v=m;y=m;c=m;% pre-allocate
14 v(1)=m0+sqrt(C0)*randn;% initial truth
15 m(1)=10*randn;% initial mean/estimate
16 c(1)=10*C0;H=1;% initial covariance and observation operator
17
18 %% solution % assimilate!
19
20 for j=1:J
21
22     v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
23     y(j)=H*v(j+1)+gamma*randn;% observation
24
25     mhat=alpha*sin(m(j));% estimator predict
26     chat=alpha*cos(m(j))*c(j)*alpha*cos(m(j))+sigma^2;% covariance predict
27
28     d=y(j)-H*mhat;% innovation
29     K=(chat*H')/(H*chat*H'+gamma^2);% Kalman gain
30     m(j+1)=mhat+K*d;% estimator update
31     c(j+1)=(1-K*H)*chat;% covariance update
32
33 end
34
35 js=21;% plot truth, mean, standard deviation, observations
36 figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
37 plot([0:js-1],m(1:js)+sqrt(c(1:js)),'r--');plot([1:js-1],y(1:js-1),'kx');
38 plot([0:js-1],m(1:js)-sqrt(c(1:js)),'r--');hold;grid;xlabel('iteration, j');
39 title('ExKF, Ex. 1.3')
40
41 figure;plot([0:J],c);hold
42 plot([0:J],cumsum(c)./[1:J+1]','m','Linewidth',2);grid
43 hold;xlabel('iteration, j');
44 title('ExKF Covariance, Ex. 1.3');
45
46 figure;plot([0:J],(v-m).^2);hold;
47 plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
48 hold;xlabel('iteration, j');
49 title('ExKF Error, Ex. 1.3')
```

4.3.5. p11.m

The program `p11.m` contains an implementation of the ensemble Kalman Filter, with perturbed observations (PO) as described in section 3.2.3.The structure of this program is again very similar to `p7.m` and `p10.m`, except now an ensemble of particles, of size `N` defined in line 12, is retained as an approximation of the filtering distribution. The ensemble $\{v^{(n)}\}_{n=1}^N$ represented by the matrix `U` is then constructed out of draws from this Gaussian in line 18, and the mean m_0' is reset to the ensemble sample mean.

In line 27 the predicting ensemble $\{\widehat{v}_j^{(n)}\}_{n=1}^N$ represented by the matrix `Uhat` is computed from a realization of the forward map applied to each ensemble member. This is then used to compute the ensemble sample mean $\widehat{m}_j$ (`mhat`) and covariance $\widehat{C}_j$ (`chat`). There is now an ensemble of innovations representing the perturbed observations, with a new i.i.d. realization $y_j^{(n)} \sim N(y_j, \Gamma)$ for each ensemble member, computed in line 31

$$d_j^{(n)} = y^{(n)} - H\widehat{v}_j^{(n)}.$$

The Kalman gain K_j (`K`) is computed using (4.11), the same as in `p7.m` and `p10.m` and the ensemble of updates are computed in line 33

$$v_j^{(n)} = \widehat{v}_j^{(n)} + K_j d_j^{(n)}.$$

The output of this program is shown in Figure 25. Furthermore, long simulations of length $J = 10^5$ were performed for this and the previous two programs `p9.m` and `p10.m` and their errors are compared in Figure 26.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p11.m Ensemble Kalman Filter (PO), sin map (Ex. 1.3)
%% setup

J=1e3;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=3e-1;% dynamics noise variance is sigma^2
C0=9e-2;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed
N=100;% number of ensemble members

m=zeros(J,1);v=m;y=m;c=m;U=zeros(J,N);% pre-allocate
v(1)=m0+sqrt(C0)*randn;% initial truth
m(1)=10*randn;% initial mean/estimate
c(1)=10*C0;H=1;% initial covariance and observation operator
U(1,:)=m(1)+sqrt(c(1))*randn(1,N);m(1)=sum(U(1,:))/N;% initial ensemble

%% solution % assimilate!

for j=1:J

    v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    Uhat=alpha*sin(U(j,:))+sigma*randn(1,N);% ensemble predict
    mhat=sum(Uhat)/N;% estimator predict
    chat=(Uhat-mhat)*(Uhat-mhat)'/(N-1);% covariance predict

    d=y(j)+gamma*randn(1,N)-H*Uhat;% innovation
    K=(chat*H')/(H*chat*H'+gamma^2);% Kalman gain
    U(j+1,:)=Uhat+K*d;% ensemble update
    m(j+1)=sum(U(j+1,:))/N;% estimator update
    c(j+1)=(U(j+1,:)-m(j+1))*(U(j+1,:)-m(j+1))'/(N-1);% covariance update

end

js=21;% plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(c(1:js)),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(c(1:js)),'r--');hold;grid;xlabel('iteration, j');
title('EnKF, Ex. 1.3')

figure;plot([0:J],c);hold
plot([0:J],cumsum(c)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('EnKF Covariance, Ex. 1.3');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('EnKF Error, Ex. 1.3')
```

4.3.6. p12.m

The program `p12.m` contains a particular square-root filter implementation of the ensemble Kalman filter, namely the ETKF filter mentioned earlier and described in detail in section 3.2.4. The program thus is very similar to `p11.m` for EnKF with perturbed observations. In particular, the filtering distribution of the state is again approximated by an ensemble of particles. The predicting ensemble $\{\widehat{v}_j^{(n)}\}_{n=1}^N$ (`Uhat`), mean $\widehat{m}_j$(`mhat`), and covariance $\widehat{C}_j$ (`chat`) are computed exactly as in `p11.m`. However, this time the covariance is kept in factorized form $\widehat{X}_j\widehat{X}_j^\top = \widehat{C}_j$ in lines 29-30, with factors denoted `Xhat`. The transformation matrix is computed in line 31

$$T_j = \left(I_N + \widehat{X}_j^\top H^\top \Gamma^{-1} H \widehat{X}_j\right)^{-\frac{1}{2}},$$

and $X_j = \widehat{X}_j T_j$ (`X`) is computed in line 32, from which the covariance $C_j = X_j X_j^\top$ is reconstructed in line 38. A single innovation d_j is computed in line 34 and a single updated mean m_j is then computed in line 36 using the Kalman gain K_j (4.11) computed in line 35. This is the same as in the KF and ExKF of `p7.m` and `p10.m`, in contrast to the EnKF(PO) of `p11.m`. The ensemble is then updated to `U` in line 37 using the formula

$$v_j^{(n)} = m_j + X_j^{(n)}\sqrt{N-1},$$

where $X_j^{(n)}$ is the n^{th} column of X_j.

Notice that the operator which is factorized and inverted has dimension N, which in this case is large in comparison to the state and observation dimensions. This naturally should be the case for computing sample statistics, but in practical application it rarely is the case. Indeed the state dimension is usually the largest, with the observation dimension coming next, and the ensemble size is much smaller than either of these. This is why the ETKF has become a very popular method. So do not let its relative inefficiency here with respect to the other filters lead to any misconception. The results are shown in Figure 27.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p12.m Ensemble Kalman Filter (ETKF), sin map (Ex. 1.3)
%% setup

J=1e3;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=3e-1;% dynamics noise variance is sigma^2
C0=9e-2;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed
N=100;% number of ensemble members

m=zeros(J,1);v=m;y=m;c=m;U=zeros(J,N);% pre-allocate
v(1)=m0+sqrt(C0)*randn;% initial truth
m(1)=10*randn;% initial mean/estimate
c(1)=10*C0;H=1;% initial covariance and observation operator
U(1,:)=m(1)+sqrt(c(1))*randn(1,N);m(1)=sum(U(1,:))/N;% initial ensemble

%% solution % assimilate!

for j=1:J

    v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    Uhat=alpha*sin(U(j,:))+sigma*randn(1,N);% ensemble predict
    mhat=sum(Uhat)/N;% estimator predict
    Xhat=(Uhat-mhat)/sqrt(N-1);% centered ensemble
    chat=Xhat*Xhat';% covariance predict
    T=sqrtm(inv(eye(N)+Xhat'*H'*H*Xhat/gamma^2));% right-hand square-root transform
    X=Xhat*T;% transformed centered ensemble

    d=y(j)-H*mhat;randn(1,N);% innovation
    K=(chat*H')/(H*chat*H'+gamma^2);% Kalman gain
    m(j+1)=mhat+K*d;% estimator update
    U(j+1,:)=m(j+1)+X*sqrt(N-1);% ensemble update
    c(j+1)=X*X';% covariance update

end

js=21;% plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(c(1:js)),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(c(1:js)),'r--');hold;grid;xlabel('iteration, j');
title('EnKF(ETKF), Ex. 1.3');

figure;plot([0:J],c);hold
plot([0:J],cumsum(c)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('EnKF(ETKF) Covariance, Ex. 1.3');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('EnKF(ETKF) Error, Ex. 1.3')
```

4.3.7. p13.m

The program `p13.m` is an implementation of the standard sequential importance resampling algorithm from section 3.3.2, described in detail in section 3.3.

The `setup` section is almost identical to the EnKF methods, because this methods also relies on particle approximations of the filtering distribution. However, this method consistently estimates even non-Gaussian distributions. The truth and data generation and ensemble prediction in lines 24-27 are the same as in `p11.m` and `p12.m`. The way this prediction in line 27 is phrased in section 3.3.2 is $\widehat{v}_j^{(n)} \sim \mathbb{P}(\cdot|v_{j-1}^{(n)})$. An ensemble of innovations $\{d_j^{(n)}\}_{n=1}^N$ are required again, but all using the *same* observation, as computed in line 28. Assuming $w_{j-1}^{(n)} = 1/N$, then

$$\widehat{w}_j^{(n)} \propto \mathbb{P}(y_j|v_j^{(n)}) \propto \exp\left\{-\frac{1}{2}\left|d_j^{(n)}\right|_\Gamma^2\right\},$$

where $d_j^{(n)}$ is the innovation of the n^{th} particle, as given in (3.21). The vector of un-normalized weights $\{\widehat{w}_j^{(n)}\}_{n=1}^N$ (`what`) are computed in line 29 and normalized to $\{w_j^{(n)}\}_{n=1}^N$ (`w`) in line 30. Lines 32-39 implement the resampling step. First, the cumulative distribution function of the weights $W \in [0,1]^N$ (`ws`) is computed in line 32. Notice W has the properties that $W_1 = w_j^{(1)}$, $W_n \leq W_{n+1}$, and $W_N = 1$. Then N uniform random numbers $\{u^{(n)}\}_{n=1}^N$ are drawn. For each $u^{(n)}$, let n^* be such that $W_{n^*-1} \leq u^{(n)} < W_{n^*}$. This corresponds to drawing the $(n^*)^{th}$ element from the discrete measure defined by $\{w_j^{(n)}\}_{n=1}^N$. Therefore, this n^* (`ix`) is found in line 34, and the n^{th} particle $v_j^{(n)}$ (`U(j+1,n)`) is set to be equal to $\widehat{v}_j^{(n^*)}$ (`Uhat(ix)`) in line 37. `find` function! The sample mean and covariance are then computed in lines 41-42. The rest of the program follows the others, generating the output displayed in Figure 28.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p13.m Particle Filter (SIRS), sin map (Ex. 1.3)
%% setup

J=1e3;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=3e-1;% dynamics noise variance is sigma^2
C0=9e-2;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed
N=100;% number of ensemble members

m=zeros(J,1);v=m;y=m;c=m;U=zeros(J,N);% pre-allocate
v(1)=m0+sqrt(C0)*randn;% initial truth
m(1)=10*randn;% initial mean/estimate
c(1)=10*C0;H=1;% initial covariance and observation operator
U(1,:)=m(1)+sqrt(c(1))*randn(1,N);m(1)=sum(U(1,:))/N;% initial ensemble

%% solution % Assimilate!

for j=1:J

    v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    Uhat=alpha*sin(U(j,:))+sigma*randn(1,N);% ensemble predict
    d=y(j)-H*Uhat;% ensemble innovation
    what=exp(-1/2*(1/gamma^2*d.^2));% weight update
    w=what/sum(what);% normalize predict weights

    ws=cumsum(w);% resample: compute cdf of weights
    for n=1:N
        ix=find(ws>rand,1,'first');% resample: draw rand \sim U[0,1] and
        % find the index of the particle corresponding to the first time
        % the cdf of the weights exceeds rand.
        U(j+1,n)=Uhat(ix);% resample: reset the nth particle to the one
        % with the given index above
    end

    m(j+1)=sum(U(j+1,:))/N;% estimator update
    c(j+1)=(U(j+1,:)-m(j+1))*(U(j+1,:)-m(j+1))'/N;% covariance update

end

js=21;% plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(c(1:js)),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(c(1:js)),'r--');hold;grid;xlabel('iteration, j');
title('Particle Filter (Standard), Ex. 1.3');

figure;plot([0:J],c);hold
plot([0:J],cumsum(c)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('Particle Filter (Standard) Covariance, Ex. 1.3');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');title('Particle Filter (Standard) Error, Ex. 1.3')
```

4.3.8. p14.m

The program `p14.m` is an implementation of the optimal proposal sequential importance resampling algorithm from section 3.3.3. The `setup` section and truth and observation generation are again the same as in the previous programs.

The difference of this particle filter arises arises because the importance sampling proposal kernel Q_j with density $\mathbb{P}(v_j|v_{j-1}, y_j)$ is used to sample $\widehat{v}_j^{(n)}$ given $v_{j-1}^{(n)}$, rather than the kernel P with density $\mathbb{P}(v_j|v_{j-1})$.

Observe that if $v_{j-1}^{(n)}$ and y_j are both fixed, then $\mathbb{P}\left(v_j|v_{j-1}^{(n)}, y_j\right)$ is the density of the Gaussian with mean $m'^{(v)}$ and covariance Σ' given by

$$m'^{(n)} = \Sigma'\left(\Sigma^{-1}\Psi\left(v_{j-1}^{(n)}\right) + H^\top\Gamma^{-1}y_j\right), \quad \left(\Sigma'\right)^{-1} = \Sigma^{-1} + H^\top\Gamma^{-1}H.$$

Therefore, Σ' (`Sig`) and the ensemble of means $\left\{m'^{(n)}\right\}_{n=1}^N$ (vector `em`) are computed in lines 27 and 28 and used to sample $\widehat{v}_j^{(n)} \sim N(m'^{(n)}, \Sigma')$ in line 29 for all of $\left\{\widehat{v}_j^{(n)}\right\}_{n=1}^N$ (`Uhat`).

Now the weights are therefore updated by (3.28) rather than (3.21), i.e. assuming $w_{j-1}^{(n)} = 1/N$, then

$$\widehat{w}_j^{(n)} \propto \mathbb{P}\left(y_j|v_{j-1}^{(n)}\right) \propto \exp\left\{-\frac{1}{2}\left|y_j - \Psi\left(v_{j-1}^{(n)}\right)\right|^2_{\Gamma+\Sigma}\right\}.$$

This is computed in lines 31-32, using an auxiliary "innovation" vector `d` in line 31. Notice that this is *not* the same innovation as in the other programs. Lines 35-45 are again identical to lines 32-42 of program `p13.m`, performing the resampling step and computing sample mean and covariance.

The output of this program was used to produce Figure 29 similar to the other filtering algorithms. Furthermore, long simulations of length $J = 10^5$ were performed for this and the previous three programs `p11.m`, `p12.m` and `p13.m` and their errors are compared in Figure 26.

```
clear;set(0,'defaultaxesfontsize',20);format long
%%% p14.m Particle Filter (SIRS, OP), sin map (Ex. 1.3)
%% setup

J=1e3;% number of steps
alpha=2.5;% dynamics determined by alpha
gamma=1;% observational noise variance is gamma^2
sigma=3e-1;% dynamics noise variance is sigma^2
C0=9e-2;% prior initial condition variance
m0=0;% prior initial condition mean
sd=1;rng(sd);% Choose random number seed
N=100;% number of ensemble members

m=zeros(J,1);v=m;y=m;c=m;U=zeros(J,N);% pre-allocate
v(1)=m0+sqrt(C0)*randn;% initial truth
m(1)=10*randn;% initial mean/estimate
c(1)=10*C0;H=1;% initial covariance and observation operator
U(1,:)=m(1)+sqrt(c(1))*randn(1,N);m(1)=sum(U(1,:))/N;% initial ensemble

%% solution % Assimilate!

for j=1:J

    v(j+1)=alpha*sin(v(j)) + sigma*randn;% truth
    y(j)=H*v(j+1)+gamma*randn;% observation

    Sig=inv(inv(sigma^2)+H'*inv(gamma^2)*H);% optimal proposal covariance
    em=Sig*(inv(sigma^2)*alpha*sin(U(j,:))+H'*inv(gamma^2)*y(j));% optimal proposal mean
    Uhat=em+sqrt(Sig)*randn(1,N);% ensemble optimally importance sampled

    d=y(j)-H*alpha*sin(U(j,:));% ensemble innovation
    what=exp(-1/2/(sigma^2+gamma^2)*d.^2);% weight update
    w=what/sum(what);% normalize predict weights

    ws=cumsum(w);% resample: compute cdf of weights
    for n=1:N
        ix=find(ws>rand,1,'first');% resample: draw rand \sim U[0,1] and
        % find the index of the particle corresponding to the first time
        % the cdf of the weights exceeds rand.
        U(j+1,n)=Uhat(ix);% resample: reset the nth particle to the one
        % with the given index above
    end

    m(j+1)=sum(U(j+1,:))/N;% estimator update
    c(j+1)=(U(j+1,:)-m(j+1))*(U(j+1,:)-m(j+1))'/N;% covariance update

end

js=21;%plot truth, mean, standard deviation, observations
figure;plot([0:js-1],v(1:js));hold;plot([0:js-1],m(1:js),'m');
plot([0:js-1],m(1:js)+sqrt(c(1:js)),'r--');plot([1:js-1],y(1:js-1),'kx');
plot([0:js-1],m(1:js)-sqrt(c(1:js)),'r--');hold;grid;xlabel('iteration, j');
title('Particle Filter (Optimal), Ex. 1.3');

figure;plot([0:J],c);hold
plot([0:J],cumsum(c)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('Particle Filter (Optimal) Covariance, Ex. 1.3');

figure;plot([0:J],(v-m).^2);hold;
plot([0:J],cumsum((v-m).^2)./[1:J+1]','m','Linewidth',2);grid
hold;xlabel('iteration, j');
title('Particle Filter (Optimal) Error, Ex. 1.3')
```

Acknowledgements The authors are grateful to Sergios Agapiou and to Yuan-Xiang Zhang for help in preparation of an early draft of these lecture notes. They wish to thank Mel Ades and Håkon Hoel for useful feedback which helped to improve the lecture notes. AMS is grateful to EPSRC, ERC, ESA and ONR for financial support. KJHL is a member of the SRI-UQ Center at KAUST.

References

[AA99] JL Anderson and SL Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, 127(12):2741–2758, 1999.

[Aba13] H Abarbanel. *Predicting the Future: Completing Models of Observed Complex Systems*. Springer, 2013.

[AJSV08] A. Apte, C.K.R.T Jones, A.M. Stuart, and J. Voss. Data assimilation: mathematical and statistical perspectives. *Int. J. Num. Meth. Fluids*, 56:1033–1046, 2008.

[And68] A. Andrews. A square root formulation of the kalman covariance equations. *AIAA Journal*, 6(6):1165–1166, 1968.

[And96] JL Anderson. A method for producing and evaluating probabilistic forecasts from ensemble model integrations. *Journal of Climate*, 9(7):1518–1530, 1996.

[And01] JL Anderson. An ensemble adjustment kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

[BBL08] T. Bengtsson, P. Bickel, and B. Li. Curse of dimensionality revisited: the collapse of importance sampling in very large scale systems. *IMS Collections: Probability and Statistics: Essays in Honor of David Freedman*, 2:316–334, 2008.

[BCJ11] A. Beskos, D. Crisan, and A. Jasra. On the stability of sequential Monte Carlo methods in high dimensions. *Arxiv preprint arXiv:1103.3965*, 2011.

[BEM01] CH Bishop, BJ Etherton, and SJ Majumdar. Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects. *Monthly weather review*, 129(3):420–436, 2001.

[Ben02] A. Bennett. *Inverse Modeling of the Ocean and Atmosphere*. Cambridge, 2002.

[Ber85] James O Berger. *Statistical Decision Theory and Bayesian analysis*. Springer Verlag, 1985.

[BLB08] P. Bickel, B. Li, and T. Bengtsson. Sharp failure rates for the bootstrap particle filter in high dimensions. *IMS Collections: Pushing the Limits of Contemporary Statistics*, 3:318–329, 2008.

[BLL+12] CEA Brett, KF Lam, KJH Law, DS McCormick, MR Scott, and AM Stuart. Accuracy and stability of filters for dissipative pdes. *Physica D: Nonlinear Phenomena*, 245:34–45, 2012.

[BRSV08] A. Beskos, G. O. Roberts, A. M. Stuart, and J. Voss. MCMC methods for diffusion bridges. *Stochastic Dynamics*, 8(3):319–350, Sep 2008.

[BS94] J.M. Bernardo and A.F.M. Smith. *Bayesian Theory*. Wiley, 1994.

[CAH+98] P. Courtier, E. Andersson, W. Heckley, D. Vasiljevic, M. Hamrud, A. Hollingsworth, F. Rabier, M. Fisher, and J. Pailleux. The ECMWF implementation of three-dimensional variational assimilation (3d-Var). I: Formulation. *Quart. J. R. Met. Soc.*, 124(550):1783–1807, 1998.

[CD02] D Crisan and A Doucet. A survey of convergence results on particle filtering methods for practitioners. *Signal Processing, IEEE Transactions on*, 50(3):736–746, 2002.

[CDRS09] S.L. Cotter, M. Dashti, J.C. Robinson, and A.M. Stuart. Bayesian inverse problems for functions and applications to fluid mechanics. *Inverse Problems*, 25:doi:10.1088/0266–5611/25/11/115008, 2009.

[CDS10] SL Cotter, M Dashti, and AM Stuart. Approximation of Bayesian inverse problems for PDEs. *SIAM Journal on Numerical Analysis*, 48(1):322–345, 2010.

[CDS11a] S.L. Cotter, M. Dashti, and A.M. Stuart. Variational data assimilation using targetted random walks. *Int. J. Num. Meth. Fluids*, 2011.

[CDS11b] S.L. Cotter, M. Dashti, and A.M. Stuart. Variational data assimilation using targetted random walks. *Int. J. Num. Meth. Fluids*, 2011.

[CGTU08] A. Carrassi, M. Ghil, A. Trevisan, and F. Uboldi. Data assimilation as a nonlinear dynamical systems problem: Stability and convergence of the prediction-assimilation system. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18:023112, 2008.

[CMT10] A. Chorin, M. Morzfeld, and X. Tu. Implicit particle filters for data assimilation. *Communications in Applied Mathematics and Computational Science*, page 221, 2010.

[CRSW13] S. Cotter, G. Roberts, A. Stuart, and D. White. Mcmc methods for functions: modifying old algorithms to make them faster. *Statistical Science*, 28:424–446, 2013.

[CT09] A. Chorin and X. Tu. Implicit sampling for particle filters. *Proc. Nat. Acad. Sc.*, 106:17249–17254, 2009.

[CT10] A. Chorin and X. Tu. Interpolation and iteration for nonlinear filters. *Math. Modlling and Num. Anal.*, To appear, 2010.

[DG01] N. Doucet, A. de Frietas and N. Gordon. *Sequential Monte Carlo in Practice*. Springer-Verlag, 2001.

[DGA00] A Doucet, S Godsill, and C Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[DHS12] M. Dashti, S. Harris, and A.M. Stuart. Besov priors for Bayesian inverse problems. *Inverse Problems and Imaging*, 6:183–200, 2012.

[DMG01] P Del Moral and A Guionnet. On the stability of interacting processes with applications to filtering and genetic algorithms. In *Annales de l'Institut Henri Poincaré (B) Probability and Statistics*, volume 37, pages 155–194. Elsevier, 2001.

[DS11] M. Dashti and A.M. Stuart. Uncertainty quantification and weak approximation of an elliptic inverse problem. *SIAM J. Num. Anal.*, 49:2524–2542, 2011.

[Eve06] G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2006.

[Fal86] Kenneth J Falconer. *The geometry of fractal sets*, volume 85. Cambridge university press, 1986.

[FK05] M Fisher, Mand Leutbecher and GA Kelly. On the equivalence between Kalman smoothing and weak-constraint four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 131(613):3235–3246, 2005.

[GGR99] S Ghosal, JK Ghosh, and RV Ramamoorthi. Consistency issues in Bayesian nonparametrics. *Statistics Textbooks and Mongraphs*, 158:639–668, 1999.

[Har91] A.C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge Univ Pr, 1991.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[HLS14] V. Ha Haong, K.J.H. Law, and A.M. Stuart. Determining white noise forcing from eulerian observations in the navier-stokes equation. *Stochastic Partial Differential Equations*, 2014.

[HM01] PL Houtekamer and HL Mitchell. A sequential ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001.

[HOT11] K. Hayden, E. Olson, and E.S. Titi. Discrete data assimilation in the Lorenz and 2d Navier-Stokes equations. *Physica D: Nonlinear Phenomena*, 2011.

[ICGL97] K. Ide, M. Courier, M. Ghil, and A. Lorenc. Unified notation for assimilation: Operational, sequential and variational. *J. Met. Soc. Japan*, 75:181–189, 1997.

[IJ07] K. Ide and C.K.R.T. Jones. Special issue on the mathematics of data assimilation. *PhysicaD*, 230:vii–viii, 2007.

[ILS13] M.A. Iglesias, K.J.H. Law, and A.M. Stuart. Evaluation of Gaussian approximations for data assimilation in reservoir models. *Computational Geosciences*, 17:851–885, 2013.

[Jaz70] A.H. Jazwinski. *Stochastic processes and filtering theory*, volume 63. Academic Pr, 1970.

[Kal60] RE Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[Kal03] E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge, 2003.

[KLS13] DTB Kelly, KJH Law, and AM Stuart. Well-posedness and accuracy of the ensemble kalman filter in discrete and continuous time. *arxiv http://arxiv.org/abs/1310.3167*, 2013.

[LBB+00] A. C. Lorenc, S. P. Ballard, R. S. Bell, N. B. Ingleby, P. L. F. Andrews, D. M. Barker, J. R. Bray, A. M. Clayton, T. Dalby, D. Li, T. J. Payne, and F. W. Saunders. The Met. Office global three-dimensional variational data assimilation scheme. *Quart. J. R. Met. Soc.*, 126(570):2991–3012, 2000.

[Liu01] Jun S. Liu. *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer,

2001.

[Lor63] E.N. Lorenz. Deterministic nonperiodic flow1. *Atmos J Sci*, 20:130–141, 1963.

[Lor64] EN Lorenz. The problem of deducing the climate from the governing equations. *Tellus*, 16(1):1–11, 1964.

[Lor96] E.N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, pages 1–18, 1996.

[Lor00] A. C. Lorenc. Analysis methods for numerical weather prediction. *Quart. J. R. Met. Soc.*, 112(474):1177–1194, 2000.

[LR95] Peter Lancaster and Leiba Rodman. *Algebraic Riccati Equations*. Oxford University Press, 1995.

[LS12] K.J.H. Law and A.M. Stuart. Evaluating data assimilation algorithms. *Monthly Weather Review*, 140:3757–3782, 2012.

[LSS14] KJH Law, A Shukla, and AM Stuart. Analysis of the 3DVAR filter for the partially observed Lorenz '63 model. *Discrete and Continuous Dynamical Systems A*, 34:1061–1078, 2014.

[Lue68] David G Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1968.

[MB13] A.J. Majda and M. Branicki. Quantifying filter performance for turbulent dynamical systems through information theory. *Submitted*, 2013.

[MGH12] A.J. Majda, D. Giannakis, and I. Horenko. Information theory, model error and predictive skill of stochastic models for complex nonlinear systems. *PhysicaD*, 241:1735–1752, 2012.

[MH12] A.J. Majda and J. Harlim. *Filtering Complex Turbulent Systems*. Cambridge University Press, 2012.

[MHG10] A.J. Majda, J. Harlim, and B. Gershgorin. Mathematical strategies for filtering turbulent dynamical systems. *Disc. Cont. Dyn. Sys.*, 2010.

[MLPvL13] AJF Moodey, AS Lawless, RWE Potthast, and PJ van Leeuwen. Nonlinear error dynamics for cycled data assimilation methods. *Inverse Problems*, 29(2):025002, 2013.

[MRTT53] N. Metropolis, R.W. Rosenbluth, M.N. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[MT93] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Communications and Control Engineering Series. Springer-Verlag London Ltd., London, 1993.

[MW06] A.J. Majda and X. Wang. *Nonlinear Dynamics and Statistical Theories for Geophysical Flows*. Cambridge, Cambridge, 2006.

[Nic03] N.K. Nichols. Data assimilation: aims and basic concepts. In *Data Assimilation for the Earth System, Editors R. Swinbank, V.Shutyaev, W.A.Lahoz*, pages 9–20. Kluwer, 2003.

[NJSH12] L Nerger, T Janjic, J Schröter, and W Hiller. A unification of ensemble square root kalman filters. *Monthly Weather Review*, 140(7):2335–2345, 2012.

[NW99] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

[ORL08] D.S. Oliver, A.C. Reynolds, and N. Liu. *Inverse theory for petroleum reservoir characterization and history matching*. Cambridge Univ Pr, 2008.

[PD92] D. F. Parrish and J. C. Derber. The national meteorological centers spectral statistical-interpolation analysis system. *Monthly Weather Review*, 120(8):1747–1763, 1992.

[PMLvL12] R.W.E. Potthast, A.J.F. Moodey, A.S. Lawless, and P.J. van Leeuwen. On error dynamics and instability in data assimilation. *Preprint*, 2012.

[PVG98] DT Pham, J Verron, and L Gourdeau. Singular evolutive Kalman filters for data assimilation in oceanography. *Oceanographic Literature Review*, 45(8), 1998.

[RvH13] P Rebeschini and R van Handel. Can local particle filters beat the curse of dimensionality? *arXiv preprint arXiv:1301.6585*, 2013.

[SBBA08] T. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review.*, 136:4629–4640, 2008.

[Sch70] Ernst Schröder. Ueber iterirte functionen. *Mathematische Annalen*, 3(2):296–322, 1870.

[SH96] A. M. Stuart and A. R. Humphries. *Dynamical systems and numerical analysis*, volume 2 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1996.

[Spa82] C. Sparrow. *The Lorenz equations: bifurcations, chaos, and strange attractors*, volume 41 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1982.

[Stu10] A. M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numer.*, 19:451–559, 2010.

[TAB+03] MK Tippett, JL Anderson, CH Bishop, TM Hamill, and JS Whitaker. Ensemble square root filters. *Monthly Weather Review*, 131(7):1485–1490, 2003.

[Tem97] R. Temam. *Infinite-Dimensional Dynamical Systems in Mechanics and Physics*, volume 68 of *Applied Mathematical Sciences*. Springer-Verlag, New York, second edition, 1997.

[Tuc99] W. Tucker. The Lorenz attractor exists. *C. R. Acad. Sci. Paris Sér. I Math.*, 328(12):1197–1202, 1999.

[Tuc02] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Found. Comput. Math.*, 2(1):53–117, 2002.

[VdV00] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[VL09] P.J. Van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review*, 137:4089–4114, 2009.

[vL10a] P.J. van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[vL10b] PJ van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[VLE96] P.J. Van Leeuwen and G. Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 124:2892–2913, 1996.

[WH02] JS Whitaker and TM Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130(7):1913–1924, 2002.

[Wig03] S. Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2003.

[Zup97] D. Zupanski. A general weak constraint applicable to operational 4DVAR data assimilation systems. *Monthly Weather Review*, 125:2274–2292, 1997.