A High-Dimensional Particle Filter Algorithm

Jameson Quinn[*],

Abstract:

Online data assimilation in time series models over a large spatial extent is an important problem in both geosciences and robotics. Such models are intrinsically high-dimensional, rendering traditional particle filter algorithms ineffective. Though methods that begin to address this problem exist, they either rely on additional assumptions or lead to error that is spatially inhomogeneous. I present a novel particle-based algorithm for online approximation of the filtering problem on such models, using the fact that each locus affects only nearby loci at the next time step. The algorithm is based on a Metropolis-Hastings-like MCMC for creating hybrid particles at each step. I show simulation results that suggest the error of this algorithm is uniform in both space and time, with a lower bias, though higher variance, as compared to a previously-proposed algorithm.

1. Background

Filtering problems arise in many applied contexts, whenever noisy observations over time must be combined, using an explicit dynamical model, into a best-guess distribution of a current state. In cases where the system being modeled involves processes over a large spatial extent, such as models of weather or other large-scale fluid dynamics, filtering is also called data assimilation.[3] This is an active area of research, with broad applications in predictive geoscience[4][20] and robotics[19]. In fact, it is considered to be among the central problems in both of these disciplines.[1]

The basic filtering problem is as follows. We model the state of the system at time t as a random variable X_t. In our context, X_t will have values x_l at each spacial locus l, for a large number of loci. We assume $X_0, ..., X_T$ form a Markov chain with known and sampleable densities for both the initial state (π_0) and transition function (P, which maps states or densities at time $t-1$ to densities at time t). We also have a series of observations $Y_1, ..., Y_T$, and we assume that each Y_t depends only on the corresponding X_t according to a known and sampleable observation density $f(Y_t|X_t)$. This is shown graphically in Figure 1.

At each time step t, we wish to estimate the filtering distribution: that is, the probability density π_t of $X_t|\{Y_1, ..., Y_t\}$. Because the X_t are Markovian, and Y_t depends only X_t, we can write π_t recursively as

$$\pi_t(\cdot) = \boldsymbol{E}_{X_{t-1} \sim \pi_{t-1}}[P(X_t \in \cdot | Y_t, X_{t-1})] \tag{1.1}$$

To minimize subscripts, we adopt the following notation:

- We abbreviate the filtering distributions π_{t-1} and π_t by τ and π respectively.
- We abbreviate samples from X_{t-1} and X_t by $\boldsymbol{x}$ and $\boldsymbol{z}$ respectively.
- When $\boldsymbol{y}_{t-1}$ is not relevant, we write $\boldsymbol{y}$ for $\boldsymbol{y}_t$.
- Superscripts should not be read as exponentiation for these and similar entities.

[*]Thanks to Mira Bernstein, Pierre Jacob, and Luke Miratrix for constructive criticism of this manuscript.

[1] According to the papers cited above, there is "much focus in the [geoscience] literature on the assimilation of data and numerical models pertain[ing] to the sampling of high-dimensional probability density functions" [4], and "The SLAM [simultaneous location and mapping] problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots."[19]
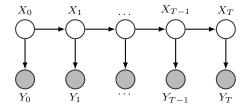

$X_0 \quad X_1 \quad \ldots \quad X_{T-1} \quad X_T$

$Y_0 \quad Y_1 \quad \ldots \quad Y_{T-1} \quad Y_T$

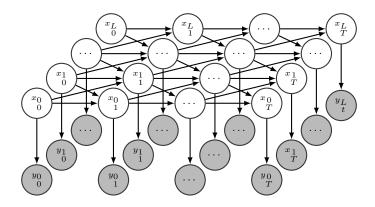FIG 1. *Graphical model of a low-dimensional filtering problem.*

FIG 2. *Graphical model of a (simple) high-dimensional filtering problem.*

The recursive formula for π_t suggests the possibility of online calculation, with only a constant computing time required to update from $\tau = \pi_{t-1}$ to $\pi = \pi_t$. However, unless we assume a particular parametric form for π, there is no finite set of sufficient statistics that could stand in for the full distribution. Thus, aside from very simple special cases, exact calculation is impossible; we look for an approximation instead.

A widely-used recursive algorithm for approximating the filtering distribution is the bootstrap particle filter. Assuming we have a sampleable distribution $\hat{\tau}$ at time $t-1$ that approximates the true filtering distribution τ, we proceed as follows:

1. Sample M iid particles $\boldsymbol{x}^{1..M}$ from $\hat{\tau}$. (Note that if we have been following the algorithm up to step $t-1$, then $\hat{\tau}$ takes the form of step (3) below.)
2. For each $\boldsymbol{x}^i$, progress it to get candidate particle $\boldsymbol{z}^i \sim \mathsf{P}\boldsymbol{x}^i$.
3. Find weights $w^i \equiv f(\boldsymbol{y}|\boldsymbol{z}^i)$. The set of weighted particles forms

$$\hat{\pi} \equiv \frac{\sum_{i=1}^{M} w^i \delta(\boldsymbol{z}^i)}{\sum_{i=1}^{M} w^i}.$$

Here $\delta(a)$ is the Dirac delta density; for example, $\frac{1}{2}(\delta(0) + \delta(1))$ is the Bernoulli distribution with $p = \frac{1}{2}$.

A key property of the particle filter algorithm is that, for large enough M, the Monte Carlo error remains under control, even as the time steps accumulate.[6] Specifically, let $\hat{\pi}_t^M$ be the approximation to π_t obtained using M particles, as above. Let $\mathcal{F}$ be the set of functions from the domain of π_t to $(-1, 1)$. For each $f \in \mathcal{F}$, denote $\boldsymbol{E}_{X \sim \pi_t}(f(X))$ and $\boldsymbol{E}_{X \sim \hat{\pi}_t^M}(f(X))$ by $\pi_t(f)$ and $\hat{\pi}_t^M(f)$ respectively. Then

$$\sup_{\mathcal{F}} \boldsymbol{E}|\pi_t(f) - \hat{\pi}_t^M(f)| \leq \frac{C}{\sqrt{M}}, \tag{1.2}$$

for some constant C that *does not depend on t*. The outer expectation here is taken over the randomness of the algorithm itself; that is, considering the distribution $\hat{\pi}$ as itself a random variable, while π is fixed.[14, p. 2814]

Now suppose that we are interested in modeling processes with a large spatial extent. For instance, in a weather model, one might use a lattice of points that cover the region of interest, with various continuous values (temperature, humidity, pressure, wind, etc.) recorded at each locus. If X_t contains information about L separate spatial loci, and the state space at each locus has dimension K, then the full state space of X_t has dimension KL. In practical applications, this can easily be 10^7 or more.[21]

To see why, consider a schematic diagram of the model (Figure 2), where the state of the system at time t and locus l is denoted $x_{\underset{t}{l}}$.

Note the following assumptions implicit in the diagram:

- $(y_{\underset{t}{l}}|x_{\underset{t}{l}}) \perp\!\!\!\perp x_{\underset{t}{k \neq l}}$. This assumption will be used in the following for simplicity, although I believe it can be relaxed in practice with only minor additional complications.
- More crucially, $x_{\underset{t}{l}}$ depends on $x_{\underset{t-1}{k}}$ only for k in some small spatial "neighborhood" $\mathcal{N}(l)$ of l. The precise composition of $\mathcal{N}(l)$ depends on model assumptions as well as the way that the L loci are positioned in

space. The diagram depicts the case where the loci are all laid out along a single spatial dimension, so that the immediate neighbors of locus l are loci $l-1$ and $l+1$. In practical applications, the loci would more likely be connected in a 2D or 3D grid.

I will discuss this locality of dynamics assumption further below, as it is key to the performance of the algorithm I propose in this paper.

Technically speaking, the error bounds in 1.2 still apply: errors are stable over time and inversely proportional to the square root of the number of particles ($\propto 1/\sqrt{M}$). But it is widely recognized that the bootstrap particle filter is no longer a practical solution in this context, due to weight degeneracy.[18][5][1] The problem is that, in the resampling step, the majority of the weight will tend to be carried by only a small fraction of the proposed particles. To see why, note that the log likelihood of each particle is the sum of its log likelihood at each locus. Since these terms are roughly independent, as L increases, the empirical distribution of log likelihoods of the particles comes to resemble a Gaussian distribution with variance that scales linearly with L. Thus the weights come to be distributed approximately according to a log-normal distribution, whose skewness increases exponentially with L. Thus, the fraction of particles with above-average weight will shrink exponentially with L. [2]

In fact, without an exponentially large number of particles, not only will one of them tend to have more weight than all the others, but it is likely that there is some missing value whose weight would dominate even our best particle. At this point the particle filter ceases to be a useful approximation of the filtering distribution. That is, although the constant C in Equation 1.2 does not depend on time, it does grow exponentially with L. This is what is known as *curse of dimensionality* for particle filters.

1.1. Existing state of the art (Rebeschini and van Handel, 2015)

There have been various proposals for dealing with this curse of dimensionality in general. In fact, there are three different recent survey articles reviewing and comparing these, by Septier and Peters[16], Morzfield *et al.*[11], and Farchi and Bocquet.[7] Some of these prior methods do not use the locality of dynamics assumption, which I believe limits their effectiveness. This includes Gilks and Berzuini,[8] who suggest rejuvenating particles with MCMC steps, targeted to the filtering distribution, to avoid the duplication problem from resampling; and Goodsill and Clapp,[9] who propose using bridging densities such as annealed quasi-filtering densities to solve the problem of lack of overlap between the progressed density $\mathsf{P}\tau \equiv (\boldsymbol{z}|\boldsymbol{y}_1,...,\boldsymbol{y}_{t-1})$ and the likelihood $f(\boldsymbol{y}_t|\boldsymbol{z})$.

The two previous proposals that do use locality of dynamics come from Poterjoy[12][13] and from Rebeschini and van Handel.[14] Of these two, Rebeschini and van Handel's proposal is more generally applicable, so I will explain it further below. Poterjoy, on the other hand, suggests a scheme that, as given, is limited to situations of sparse observations; it uses estimated covariance matrices to blend resampled particle filter values at a local scale with unresampled values at a meso-scale.

Rebeschini and van Handel's proposal is called the block particle filter; also sometimes termed the localized particle filter. In simple terms, they replace the global resampling step of the bootstrap particle filter with a local resampling step which constructs new particles by resampling neighborhoods independently.

They begin their theoretical discussion by offering an overall point of view of the problem which very much inspired the current work. They focus on the decay of correlations between local values as spatial distance increases, which they say is "in essence a spatial counterpart of the much better-understood [temporal] stability property of nonlinear filters". This decay of correlations, discussed further below, is a product of the locality-of-dynamics assumption shown in the diagrams above.

Rebeschini and van Handel partition the loci $1,...,L$ of the progressed particles into J zones $\{\mathcal{Z}_j\}$, where each zone consists of a small number of (contiguous) loci. They then weight and resample values from each zone independently. This produces what I would call "Frankenstein" particles, sewn together from pieces which tend to fit well with observations locally, but which may often come from progressing different particles from time $t-1$.

The precise steps of the block particle filter algorithm are as follows:

1. Given $\hat{\tau}$, a sampleable distribution that approximates the ideal filtering distribution τ, sample M iid particles $\boldsymbol{x}^{1..M}$ from $\hat{\tau}$. (Note that if $\hat{\tau}$ takes the form of the output of step (3) below, then this amounts to

---

[2]Bickel *et al.*[5] formalize this line of argument, showing that the variance of the log likelihood may be seen as an estimate of the effective state dimension depicted by the measurements $\boldsymbol{y}$.

sampling, independently with replacement, a k_j^i for each zone Z_j; $1 \leq j \leq J$ and particle i with probability $w_{Z_j}^{k_j^i}/\Sigma_n w_{Z_j}^n$; then putting those together to make the final particles, so that $l \in Z_j \Rightarrow z_l^i = z_l^{k_j^i}$.)

2. For each $\boldsymbol{x}^i$, progress it to get $\boldsymbol{z}^i \sim \mathsf{P}\boldsymbol{x}$.
3. Find weights for each particle for each zone: $w_{Z_j}^i = \prod_{l \in Z_j} f(y_l|z_l^i)$. Then

$$\hat{\pi} \equiv \bigotimes_{j=1}^{J} \frac{\sum_{i=1}^{M} w_{Z_j}^i \delta(\boldsymbol{x}_{Z_j}^i)}{\sum_{i=1}^{M} w_{Z_j}^i}$$

Intuitively, cutting the progressed particles into zones and reconstituting them is a way to solve the exponential curse of dimensions. However, it gives rise to a different problem: it breaks any covariances across zone boundaries, whether those come from covariances at the $t-1$ time step or are induced by the transition kernel using a history that overlaps the boundary. These broken covariances lead to error at the boundaries, which does not disappear even as number of particles goes to infinity.

It could also then lead to unrealistic dynamics near the boundaries at later time steps, especially if the forward density operator P is nonlinear. [14, p. 2829] For instance, imagine a weather model in which the hypothetical air pressure in a particle varied reasonably within each zone, but a discontinuity at the zone boundary led to a prediction of a tornado forming in the next time step. Note that the algorithm proposed in this paper avoids such discontinuities, but for unrelated reasons can be ill-suited to modeling models with nonlinear dynamics; I will address this issue in later work with a proposed extension to my algorithm.

For the block particle filter method to be useful, covariances between local values must tend to decay with distance. If such decay of correlations holds, then, far enough from a zone boundary, the dynamics return to normal.

This intuition helps explain the error bounds that these researchers prove their algorithm obeys. They show that the error for the value at a given locus l satisfies

$$|||\pi - \hat{\pi}^M|||_{\{l\}} \leq \alpha(\frac{e^{\beta|Z_l|}}{\sqrt{M}} + e^{-\gamma \inf |l-b \in Z_l^C|}) \tag{1.3}$$

where the constants $\alpha, \beta, \gamma > 0$ do not depend on t. (They define a norm for this purpose which measures the distance between random distributions; I will not reproduce this definition here, as in this this result is merely a guide for intuition.)

One can see that there is a tradeoff: using smaller zones and/or more particles allows better guesses for a given zone to control the term $\frac{e^{\beta|Z_l|}}{\sqrt{M}}$, while using larger zones makes it possible to move away from boundary effects and control the term $e^{-\gamma \inf |m-b \in Z_l^C|}$. In practice, Rebeschini and van Handel give a simple example where it would still be possible to control average error per locus, based on finding an optimal balance between adding particles and increasing neighborhood size, but in that example the inverse of their error bound grows only logarithmically with the computing resources/number of particles — in other words, as tolerances tighten, the required number of particles can still grow exponentially. Thus, although their algorithm in practice gives error far lower than that of the bootstrap particle filter, it has not fully overcome the problem of needing exponential computing cost as dimension grows, especially if one wishes to achieve fixed error bounds that are lower than what comes easily from a moderate neighborhood size.

2. The Finkelstein solution

In this section, I will sketch out the basic outlines of a recursive algorithm in which each particle at time t is composed of values at different loci which are drawn from state vectors progressed from different particles at time $t-1$. The choice of which values for a given locus combine with which values at other loci is made by running a separate Metropolis-Hastings MCMC to create each composite particle, proposing to replace one locus value at each MCMC step. What acceptance ratio ρ to use for those proposals will be discussed in later sections.

I name this the Finkelstein algorithm, after the character Sally Finkelstein from the movie "The Nightmare Before Christmas". I have already compared the block particle filter to a Frankenstein solution, in which the progressed particles are chopped up and then randomly sewn back together. In that algorithm, the suitability of the values in each zone of each particle is measured against observation, but not against the other zones

on which it borders. Finkelstein can improve on this. Though herself originally a Frankenstein's-monster-like creation of a stereotypical mad doctor, now that she has been animated, Sally is able to lose her body parts and sew them back on, and thus presumably to choose for herself only those body parts that best fit together. In my terms, Finkelstein would be able to run an MCMC process on her own body, targeting whatever distribution she pleases.

A key aspect of this algorithm is the Metropolis-Hastings acceptance ratio ρ. When choosing a formula for such a ratio, the key question is, what distribution do we wish to target? I'll begin by showing an algorithm that targets the natural unnormalized density:

$$
\begin{aligned}
f_{\boldsymbol{x} \sim \tau}(\boldsymbol{z}|\boldsymbol{y}) &= \int f(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{x})\tau(\boldsymbol{x})d\boldsymbol{x} \\
&\propto \int f(\boldsymbol{y}, \boldsymbol{z})f(\boldsymbol{z}, \boldsymbol{x})\tau(\boldsymbol{x})d\boldsymbol{x} \\
&= [\Pi_l f(y_l|z_l)] \int [\Pi_l f(z_l|\boldsymbol{x})]\tau(\boldsymbol{x})d\boldsymbol{x}
\end{aligned}
\tag{2.1}
$$

It will turn out that targeting this density still suffers a similar curse of dimensionality as the bootstrap particle filter, so I will modify the algorithm, such that its stationary distribution is not precisely the above expression. Still, this expression is still the starting point; by approximately targeting it, I approximately target π.

Here are the steps of the basic Finkelstein algorithm. I do not include a formula for the acceptance probability ρ here; I will develop and discuss several alternatives for such a formula, based on modifications of the density above, in the following sections.

1. Assume we have $\hat{\tau}^M$, a sampleable distribution which in some sense approximates the ideal filtering distribution τ, and which must be of the form $\frac{1}{M}\sum_{i=1}^{M}\delta(\boldsymbol{x}^i)$. Note that unlike the cases of the bootstrap and block particle filters, the final $\hat{\pi}$ produced by this algorithm already has equally-weighted particles; resampling is not required.
2. For each particle $\boldsymbol{x}^i$, progress it to get a full particle $\tilde{\boldsymbol{z}}^i \sim \mathsf{P}\boldsymbol{x}^i$ whose local values are known as $\tilde{z}_l^i$. (In later steps, I will assume for simplicity that there are no duplicate values at any locus, so $i \neq j \Rightarrow \tilde{z}_l^i \neq \tilde{z}_l^j$, but relaxing this assumption should be straightforward if necessary.)
3. Find likelihood weights for each such local value, denoted $w_l^i \equiv f(y_l|\tilde{z}_l^i)$; and forward densities conditional on $\boldsymbol{x}^j$ for all j (including $j = i$), denoted $f_l^{j \to i} \equiv f_{\mathsf{P}}(\tilde{z}_l^i|\boldsymbol{x}^j)$.
4. In parallel, for $k = 1, \ldots, M$, do the following:

   (a) Create a new proposal particle by independently sampling each locus of a vector $\boldsymbol{\iota}^0 \in \{1..M\}^L$. This vector specifies the source for the value of $\tilde{z}$ that is being considered at each locus; that is, after running the MCMC for S steps, the final value $\boldsymbol{\iota}^S$ will be used to define $\boldsymbol{z}^k$ by setting $z_l^k = \tilde{z}_l^{\iota_l^S}$. The initial sampling at each locus uses the probabilities

   $$
   P(\iota_l^0 = i) = w_l^i / \sum_j w_l^j.
   $$

   (Note that these initial sampling probabilities are arbitrary and, if the MCMC successfully runs until convergence, irrelevant. The probabilities above represent a reasonable starting point that should converge reasonably well, but it may be possible to get even faster convergence through some sampling scheme that is not independent across loci.)

   (b) Run a Metropolis-Hastings MCMC chain targeting an approximation of the filtering distribution, for $s = 1, ..., S$ steps to (assumed) convergence:

   i. Choose a spatial locus $\lambda(s) \in \{1, \ldots, L\}$ uniformly at random. For brevity, I will refer to this as λ, suppressing the dependency on s, in the steps that follow.

   ii. Sample a proposed particle ι^* from which to draw the replacement value $\tilde{z}_\lambda^{\iota^*}$ for locus λ, with probability
   $$
   P(\iota^* = i) = w_\lambda^i / \sum_j w_\lambda^j.
   $$

As with λ itself, I am omitting here the subscript s, even though this will be resampled at each step. Note that unlike $\boldsymbol{\iota}^s$, which is a vector of one integer per locus, this ι^* is only one integer, which determines the source for the proposed value only at locus λ. So for convenience in the formulas below, I will also define the vector $\boldsymbol{\iota}^{**}$ such that $\iota_\lambda^{**} = \iota^*$ and $\forall k \in \{1, ..., \lambda - 1, \lambda + 1, ..., L\} : \iota_k^{**} = \iota_k^{s-1}$

   iii. Accept this proposed change with M-H probability: $1 \wedge \rho(\boldsymbol{\iota}^{s-1}, \lambda, \iota^*)$, where ρ is defined below. In case of acceptance, $\boldsymbol{\iota}^s := \boldsymbol{\iota}^{**}$; Otherwise, in case of rejection, make no change, so that $\boldsymbol{\iota}^s := \boldsymbol{\iota}^{s-1}$.

  (c) Let $z_l^k = \tilde{z}_l^{\iota_l^S}$.

We would like to tune the proposal and acceptance probabilities so that this algorithm targets the distribution $(\boldsymbol{z}|\boldsymbol{y}, \{\boldsymbol{x}^i\})$. Insofar as we succeed, the full algorithm will be very similar to a bootstrap particle filter, which similarly targets that distribution. Thus, on a purely intuitive level, it is unsurprising that this should work if the MCMC does. But the whole process depends on the validity of the MCMC, which in turn depends on the M-H acceptance probability $\rho(\boldsymbol{\iota}^{s-1}, \lambda, \iota^*)$, left unspecified above.

3. Developing a working formula for ρ

In this section, I'll first develop some motivating understanding of what ρ should be like, then develop two specific formulas for ρ:

1. The first formula, ρ_{full}, is for illustrative purposes only. Though it is, by construction, asymptotically correct–that is, the algorithm using ρ_{full} approaches the correct filtering distribution as the number of particles M approaches infinity–it is unsuitable for use in practice. Not only does it lead to impractically high computational costs for a specific M, it also suffers from similar dimensionality problems as the bootstrap particle filter.
2. The second formula ρ_{local} only considers values of the MCMC in some local neighborhood of $\lambda(s)$. It thus does not suffer the weight degeneracy problem of the bootstrap particle filter or ρ_{full}, while still being possible to calculate in computing time that's polynomial in number of particles, and linear in dimension and number of time steps.

In the next section, I'll develop a further formula for ρ that improves the computational characteristics, as well as proposing some other computational optimizations.

3.1. ρ_{full}: full acceptance probability

I will begin by using a standard Metropolis-Hastings ratio to derive a ρ_{full} that targets the (unnormalized) density 2.1. That expression is promising in one sense: it suggests that one can judge the fit of local proposals drawn from two different particles using an expression involving the transition kernel forward density. However, the fact that it involves a sum over all particles, of a product over all loci, makes ρ_{full} computationally unworkable in practice. And even if there were enough available computational power to calculate this at every step of an MCMC, taking a product over loci would lead to similar problems as the naive high-dimensional bootstrap particle filter has.

If we propose replacement z_l^i for the value at a given locus l with probability proportional to some value v_l^i, we can construct a Metropolis-Hastings acceptance probability in the usual way, by multiplying a ratio of target densities (proposed over current) by a ratio of proposal densities (current over proposed).

(To avoid nested subscripts/superscripts in the following, I leave out redundant indices for locus; thus using the notation $f_k^{j \to \iota^{**}}$ rather than $f_k^{j \to \iota_k^{**}}$, using $f_k^{j \to \iota(s-1)}$ rather than $f_k^{j \to \iota_k^{s-1}}$, and using $w_k^{\iota(s-1)}$ rather than $w_k^{\iota_k^{s-1}}$.)

$$\rho_{\text{full}} \equiv \frac{[w_\lambda^{\iota^*} \Pi_{k \neq \lambda} w_k^{\iota(s-1)}] \, \Sigma_{j=1}^M [\Pi_k f_k^{j \to \iota^{**}}]}{[w_\lambda^{\iota(s-1)} \Pi_{k \neq \lambda} w_k^{\iota(s-1)}] \, \Sigma_{j=1}^M [\Pi_k f_k^{j \to \iota(s-1)}]} \cdot \frac{v_\lambda^{\iota(s-1)} \, \Sigma_{j=1}^M f_\lambda^{j \to \iota(s-1)}}{v_\lambda^{\iota^{**}} \, \Sigma_{j=1}^M f_\lambda^{j \to \iota^{**}}} \tag{3.1}$$

Let's consider these terms from left to right, taking the version of each term as it appears in the numerator:

1. $w_\lambda^{\iota^*}$: The likelihood at the locus in question; the term which depends on $\boldsymbol{y}$. I will set $v_\lambda^{\iota^*}$ so as to cancel this out.

2. $\Pi_{k\neq\lambda}w_k^{\iota(s-1)}$: The likelihoods at other loci. These cancel out naturally.
3. $\Sigma_j[\Pi_k f_k^{j\to\iota^{**}}]$: This sum of products term is the heart of the calculation at each MCMC step. Each product is the forward likelihood of the current/proposed hybrid particle conditional on a given history; the sum of products is proportional to the forward likelihood of the current/proposed hybrid particle conditional on $\hat{\tau}^M$.
4. $v_\lambda^{\iota(s-1)}$: Weights that define the proposal distribution and can be chosen arbitrarily (up to normalization).
5. $\Sigma_{j=1}^M f_\lambda^{j\to\iota(s-1)}$: The part of the proposal density that's due to $\hat{\tau}^M$; the probability density that a given value would have been in $\{\tilde{z}_\lambda^i : 0 < l < L\}$ to be available to be sampled. To someone used to other variations of particle filtering algorithms, it may seem counterintuitive to include this term; usually, taking advantage of the forward density is a key aspect of how the algorithm works, not something that needs canceling out. However, from the perspective of a Metropolis-Hastings construction, it is necessary to include this in order to target the intended (unnormalized) density 2.1. On a more intuitive level, one might note that the forward density values $f_\lambda^{j\to\iota(s-1)}$ for each j appear both here and in the sum of products term in the denominator; so if one did not include this term, that would in a sense be double counting these forward density values by allowing them to cause an increased proposal density and then also increase the acceptance ratio.

The unnormalized proposal weights v_l^i can be set at will; I will let $v_l^i \equiv w_l^i$, so that these terms cancel out. Now ρ is just a ratio of sums of products, multiplied by a ratio of the forward mean proposal densities $\bar{f}_l^i$. I could have set v_l^i to also cancel out $\bar{f}_l^i$, but as seen later, this would slow convergence.

The quantities $\bar{f}_l^i$ do not change for different MCMC chains or for different steps in each chain and can be precalculated in $O(LM)$ time. We thus have

$$\rho_{\text{full}} \equiv \frac{\Sigma_{j=1}^M[\Pi_{k=1}^L f_k^{j\to\iota^{**}}]}{\Sigma_{j=1}^M[\Pi_{k=1}^L f_k^{j\to\iota(s-1)}]} \frac{\Sigma_{j=1}^M f_l^{j\to\iota(s-1)}}{\Sigma_{j=1}^M f_l^{j\to\iota^{**}}} \tag{3.2}$$

By construction, this acceptance probability satisfies the conditions for detailed balance of a Metropolis-Hastings MCMC, and thus converges to the desired target stationary distribution 2.1 under standard regularity assumptions. At convergence, the algorithm will give M samples from the correct target distribution, but with the constraint that the value for each sample at each locus must be available in $\tilde{z}^{1..M}$. As $M \to \infty$, the set of values available at each locus will become dense, so the conditionality will not be restrictive. Thus, asymptotically, one would expect that each z^i should be a sample from the correct filtering distribution, conditional on $x \sim \hat{\tau}^{M_{t-1}}$.

Yet ρ_{full} is not useful in practice, for two reasons. First, on a relatively trivial level, actually calculating the sum of products terms once for each step of the MCMC would be computationally prohibitive; though not exponential, the resources required would be extreme.

More importantly, unless M_{t-1} is exponentially high, $\mathsf{C}_t\mathsf{P}\hat{\tau}^{M_{t-1}}$ is not a good approximation of $\mathsf{C}_t\mathsf{P}\tau$, because of a curse of dimensionality very similar to that which causes the bootstrap particle filter to fail in high dimensions. In ρ_{full}, the acceptance probability is based on a sum over histories of products over loci of likelihoods. Following a similar logic as Bickel *et al.*[5], discussed above, for showing weight degeneracy in the bootstrap particle filter, we see that, assuming that the likelihoods associated with distantly-separated loci are roughly independent, then as dimension increases the distribution of these products over loci will approach a log-normal. Since the variance of that distribution will grow with dimension, the sum is likely to be degenerate unless number of particles grows exponentially with dimension; just one history particle will contribute more to the sum than all others put together.

Consider the example of a weather model of the continental United States, where imperfect measurements of atmospheric conditions are taken daily over a set of cities. In this case, the sum would be degenerate because, although any given proposal particle (weather map for today) might accord well with a given history (possible weather map for yesterday) for some cities, you'd nevertheless need to consider an exponentially large number of possible histories before finding one which accords well across *all* cities with a given realistic present.

In essence, rather than resolving the high-dimensionality problem at time t, we've merely pushed it off to time $t-1$; because of the high variance of the product terms $\Pi_{k=1}^L f_k^{j\to\iota^{**}}$, the sums will tend to be dominated by the product term for a single history j, losing most of the benefits of a high number of particles.

3.2. ρ_{local}: *Simplifying the product terms by focusing on local neighborhoods*

The main computational burden of calculating ρ_{full} comes from the sum over histories of a product over loci. To deal with these computational issues, as well as with the degeneracy of the sum, it would be good to take this product over fewer terms. To do so, I restrict the product over loci to only consider loci in some neighborhood of the locus l which the proposal would change.

This idea gains some support from the decay of correlations property of that Rebeschini and van Handel (2015) demonstrate. This is a complex issue which occupies a significant portion of their paper, but to summarize briefly: they assume particle filters with local dynamics, and both forward densities and observation likelihoods that are strongly bounded away from zero and infinity, Given those assumptions (which they argue are probably stronger than necessary in most practical cases), they show that changing the value at locus k cannot change the conditional distribution of the value at l by more than a quantity that falls exponentially as the distance between k and l increases. Thus, it would seem logical that, in calculating an acceptance probability to target the distribution at l, one may safely ignore faraway loci k.

To use this idea for the Finkelstein algorithm, assume there is a natural distance metric $d(l, k)$ over loci; for instance, if loci were arranged in a square lattice, $d(l, k)$ could be the ℓ_1-distance. Use this distance to define neighborhood balls $\mathcal{B}_r(\lambda) \equiv \{l : d(\lambda, l) \le r\}$, and use the natural notation that $\boldsymbol{x}_{\mathcal{B}_r(\lambda)} \equiv \{x_l : l \in \mathcal{B}_r(\lambda)\}$. Thus, the new ρ would be:

$$\rho_{\text{local}} \equiv \frac{\Sigma_{j=1}^{M}[\Pi_{k \in \mathcal{B}_r(\lambda)} f_k^{j \to \iota^{**}}]}{\Sigma_{j=1}^{M}[\Pi_{k \in \mathcal{B}_r(\lambda)} f_k^{j \to \iota(s-1)}]} \frac{\Sigma_{j=1}^{M} f_\lambda^{j \to \iota(s-1)}}{\Sigma_{j=1}^{M} f_\lambda^{j \to \iota^*}} \tag{3.3}$$

If this works, it will have finally conquered the curse of dimensions. For any locus l, there are only $|\mathcal{B}_r(l)|$ terms in each product; a quantity which does not depend on the overall dimension of the problem, only on the local connectivity. We can therefore choose a fixed M large enough such that this sum of products is not degenerate (not dominated by just one of the products) for $N \equiv \max_l |\mathcal{B}_r(l)|$ loci.

However, it should be noted that with this acceptance probability, the algorithm is no longer strictly speaking Metropolis-Hastings. In particular, the overall MCMC is no longer guaranteed to obey detailed balance. If one repeatedly replaced the values of a single locus l, the MCMC would, by the standard Metropolis-Hastings construction, show detailed balance at a unique stationary distribution with a density proportional to:

$$f_l(z_l | z_1, ..., z_{l-1}, z_{l+1}, ..., z_L) \equiv (\Pi_{\lambda=1}^{L} \mathbb{1}_{z_\lambda \in \{\tilde{z}_\lambda\}}) \Sigma_{j=1}^{M}[\Pi_{k \in \mathcal{B}_r(l)} f_{\mathsf{P}}(z_k | \boldsymbol{x}^j)] \tag{3.4}$$

This function is not only of z_l, but of all z_k such that $k \in \mathcal{B}_r(l)$. However, since this density is not the same for two different values of l, this detailed balance can and will break down. The MCMC is still uniformly ergodic, so a unique stationary distribution still exists; but without detailed balance, we lack the nice guarantees that Metropolis-Hastings would offer as to what that target distribution is. At present, then, my use of ρ_{local}, and all later versions of ρ that build on it, is based on empirical validation, as seen below in the simulation section, not rigorous theory.

4. Computational optimizations

4.1. ρ_{sampled}: *a version of ρ_{local} which replaces numerator and denominator by unbiased estimators*

Running the Finkelstein algorithm with acceptance probability ρ_{local} does not require exponential computation, but even polynomial amounts of computation can be daunting in practice. Recall that the of sums over all histories in ρ_{local} are proportional to the forward likelihood conditional on $\hat{\tau}^M$, the M-particle approximation of the filtering distribution τ. At each step, we can save computation by, effectively, estimating τ with only an arbitrary fixed number $H << M$ particles; that is, by using only H history terms for these sums and using those to get unbiased Horvitz-Thompson estimators of the totals. This leads to ρ_{sampled}. Note that the specific H history particles used will change from step to step and locus to locus in the MCMC, thus taking advantage of the full M particles in equilibrium.

The idea of using unbiased estimators to calculate the Metropolis-Hastings acceptance ratio is not new, and, as Andrieu and Roberts 2009[2] show, this can be made to conserve the stationary distribution, provided that any randomness used in finding the estimator is maintained as part of an expanded Metropolis-Hastings

parameter space. This could work for ρ_{full}. But now that we are working from ρ_{local}, this is impossible because the MCMC is already not true Metropolis-Hastings with a single common parameter space. As discussed above, the target distribution of the particle as a whole is different when changing values at different loci, although there's reason to hope that the difference for nearby loci is small. Thus, ρ_{sampled} will inevitably have a different stationary distribution from ρ_{local}. Nevertheless, in the algorithm below, in an attempt to ensure that the stationary distribution changes as little as possible, I expand the parameter space of ρ_{sampled} with a matrix $\boldsymbol{\eta}^s$. This ensures that the same H histories used when accepting a value at a locus are also used when deciding whether to change that value later.

We revise the algorithm from section 2 as follows:

1. As before, assume we have $\hat{\tau}^M$.
2. As before, for each particle $\boldsymbol{x}^i$, progress it to get a full particle $\tilde{\boldsymbol{z}}^i \sim \mathsf{P}\boldsymbol{x}^i$.
3. As before, find likelihood weights $w_l^i \equiv f(y_l|z_l^i)$ and forward densities $f_l^{j \to i} \equiv f_{\mathsf{P}}(z_l^i|\boldsymbol{x}^j)$ for all $i, j \in \{1, \dots, M\}$ and $l \in \{1, \dots, L\}$.
4. In parallel, for $k = 1, \dots, M$, do the following:

   (a) As before, sample $\boldsymbol{\iota}^0 \in \{1..M\}^L$ to initialize the state of the new proposal particle.

   (b) In addition, initialize an $L \times H$ matrix $\boldsymbol{\eta}^0$ with entries in $\{1...M\}$, sampled iid with probabilities

   $$P(\eta_{l,h}^0 = i) = \frac{g(f_l^{i \to \iota_l^0})}{\Sigma_j g(f_l^{j \to \iota_l^0})},$$

   where g is an arbitrary monotonically increasing function.

   Each entry $\eta_{l,h}^0$ gives the index i of one history $\boldsymbol{x}^i$ which we will later use to estimate the denominator of ρ. The significance of g will be explained in more detail below.

   (c) As before, run a Metropolis-Hastings MCMC chain, for steps $s = 1, ..., S$, updating $\boldsymbol{\eta}$ and $\boldsymbol{\iota}$ at each step:

      i. As before, choose a spatial locus $\lambda(s)$ (aka λ) uniformly at random.

      ii. As before, sample a proposed replacement $\iota^*(s)$ (aka ι^*) for locus λ, with probability

      $$P(\iota^* = i) = w_\lambda^i / \sum_j w_\lambda^j.$$

      Also, define $\boldsymbol{\iota}^{**}$ as before.

      iii. In addition, sample (iid) a set $(\eta_1^*, ..., \eta_H^*)$ of histories by which to judge this proposed replacement, where

      $$P(\eta_h^* = i) = \frac{g(f_\lambda^{i \to \iota^*})}{\Sigma_j g(f_\lambda^{j \to \iota^*})}.$$

      Define the matrix $\boldsymbol{\eta}^{**}$ by

      $$\eta_{l,h}^{**} := \begin{cases} \eta_h^* & \text{if } l = \lambda \\ \eta_{l,h}^{s-1} & \text{if } l \neq \lambda \end{cases}$$

      iv. Finally, define

      $$\rho_{\text{sampled}} \equiv \frac{\Sigma_{h \in \{1..H\}} \frac{1}{g_\lambda(\eta_h^*, \iota^{**})} [\Pi_{l \in \mathcal{B}_r(\lambda)} f_l^{\eta_h^* \to \iota^{**}}]}{\Sigma_{h \in \{1..H\}} \frac{1}{g_\lambda(\eta_h^*, \iota^{s-1})} [\Pi_{l \in \mathcal{B}_r(\lambda)} f_l^{\eta_h^* \to \iota^{s-1}}]} \frac{\Sigma_{j=1}^M f_\lambda^{j \to \iota^{s-1}}}{\Sigma_{j=1}^M f_\lambda^{j \to \iota^{**}}}, \tag{4.1}$$

      where

      $$g_l(i, j) \equiv \frac{g(f_l^{i \to j})}{\Sigma_k g(f_l^{k \to j})}.$$

      As usual, we accept the proposed replacement with M-H probability $1 \wedge \rho_{\text{sampled}}$. In case of acceptance, let $\boldsymbol{\iota}^s := \boldsymbol{\iota}^{**}$ and $\boldsymbol{\eta}^s := \boldsymbol{\eta}^{**}$. Otherwise, make no change, so that $\boldsymbol{\iota}^s := \boldsymbol{\iota}^{s-1}$ and $\boldsymbol{\eta}^s := \boldsymbol{\eta}^{s-1}$.

(d) As before, set $z_l^k = \tilde{z}_l^{\iota_l^S}$.

In addition to showing that using unbiased estimators can conserve the target distribution when the parameter space is expanded, Andrieu and Roberts also discuss the case where the parameter space is not expanded. They show that this case still has a stationary distribution, which converges to the original target distribution as more samples are taken. With minor modifications, the same proof applies to the MCMC using ρ_{sampled}; the stationary distribution of ρ_{sampled} is not the same as that of ρ_{local}, but converges to it as $H \to \infty$.

A few words on the function g. First, note that the sampling probabilities for $\boldsymbol{\eta}^s$ are in principle arbitrary, so could be a function of the full current vector $\boldsymbol{\iota}^s$. In the above discussion, however, they are a function of only $f_l^{i \to \iota^*}$; this simplifies both notation and computation. Second, g should be chosen to be some non-decreasing function of $f_l^{i \to \iota^*}$, so that the variance in $\Pi_{l \in \mathcal{B}_r(\lambda)} f_l^{\eta_{\tilde{h}}^* \to \iota^{**}}$ is at least partially offset by that in g, increasing the efficiency of the estimation process. Another way of saying this is that we should make it more likely to sample plausible histories than implausible ones. Possible choices for g are discussed in Section 4.3 below. Whatever g is chosen, the denominator $\Sigma_j g(f_\lambda^{j \to \iota^*})$ can be precalculated for each possible choice of ι^*, meaning that this does not meaningfully increase computing requirements per MCMC step.

How much computation does ρ_{sampled} save? The ρ_{local} algorithm requires running M different MCMC chains, with each of L loci going through S steps, and at each step calculating a ρ using a sum over M histories of a product over the up to N locations in the relevant $\mathcal{B}_r(l)$. The total computation cost is at least $O(M^2LNS)$. This is better than $O(M^2L^2S)$ that ρ_{full} would have taken, but still somewhat burdensome. To get ρ_{sampled}, on the other hand, we only calculate the product of locus likelihoods for an arbitrary number H of histories rather than all M of them. Thus, the total computation cost falls to $O(MHLNS)$; since H and N are arbitrary constants that can be set independently from the full size M and L respectively, this is a substantial improvement.

4.2. Discussion of proposal weights

In the above discussion, the proposal weights v_l^i, used by all versions of ρ, are arbitrary. That is, any proposals with nonzero weights could be used; the v_l^i are accounted for out in the $\bar{F}$ term.

Ideally, these proposal weights should both be tuned for maximum efficiency of the MCMC; that is, insofar as it does not substantially increase the computational costs per step, to try to ensure that the variance of the acceptance probability is as low as possible (approaching the Gibbs sampling case where it's uniformly 1) while maintaining disperse (high-variance) proposal values for good ergodicity/mixing.

For v_l^i, that is similar to the idea of an optimal proposal distribution, which is common in the particle filter literature.[17] The low-dimensional bootstrap particle filter uses $(z|x^{1..M})$ as a proposal, then reweights using $(y|z)$. In such a case, the idea of an ideal proposal distribution is that if you could propose from $(z|x^{1..M}, y)$, the reweighting step would not be necessary.

Applying a similar idea to v_l^i, it becomes clear why I have set it equal to w_l^i. Of course, including a factor of w_l^i in v_l^i helps these terms cancel and thus simplifies the calculation of ρ. But if simplicity of calculating ρ were the only consideration, I could have set v_l^i to $w_l^i \Sigma_{j=1}^M f_l^{j \to i}$, so that the ratio $\frac{\Sigma_{j=1}^M f_l^{j \to \iota(s-1)}}{\Sigma_{j=1}^M f_l^{j \to \iota(s)*}}$ would cancel out too.

But setting $v_l^i \equiv w_l^i$ ensures that the proposal density for z_l, conditional on $\boldsymbol{x}^{1..M}$ and $\boldsymbol{y}$, is $(z_l|\boldsymbol{x}^{1..M}, y_l)$ — not too far from the ideal $(z_l|\boldsymbol{x}^{1..M}, \boldsymbol{y}, \boldsymbol{\zeta}_{\mathcal{B}_r(l)\backslash l}^{s-1})$ which would allow an acceptance probability of uniformly 1. That's because the density of $(z_l|\boldsymbol{x}^{1..M})$ is included implicitly through the progression procedure, while that of $(z_l|y_l)$ is handled explicitly through w_l^i.

4.3. Refining the history sampling weights

What about the history sampling weights $g_l^{i \to j}$? As above, these are arbitrary. Ideally, to minimize the variance of the acceptance probability, they would approximate:

$$g_l^{i \to j}(\boldsymbol{\zeta}_{\mathcal{B}_r(l)\backslash l}^{s-1}) \propto f_{\mathsf{P}}(\tilde{z}_l^j|\boldsymbol{x}^i, \boldsymbol{\zeta}_{\mathcal{B}_r(l)}^{s-1}) \tag{4.2}$$

$$= \Pi_{\kappa \in \mathcal{B}_r(l)} f_\kappa^{i \to j} \tag{4.3}$$

... because this expression in the Horvitz-Thompson inverse sampling weight term would cancel exactly with its relative contribution to the estimated sum of procucts, so that the overall estimator would be governed solely by the sum of weights term in the denominator.

It is computationally infeasible to calculate these quantities exactly for each step of the MCMC, so I use $g_l^{i \rightarrow j}$. In the simulation below, I've tested two formulas for these weights:

1. $g_{\text{uniform}}(x) = 1$
2. $g_{\text{bentlog}}(x) = \frac{\log(f_\lambda^{i \rightarrow j}) - \log(\min(f_\lambda^{i \rightarrow j})}{\alpha} + max(0, \log(x) - \log(\max(x) + \beta)$, where $\min(x)$ and $\max(x)$ are the precalculated minimum and maximum values of $f_\lambda^{i \rightarrow j}$ and α, β are positive constants.

Both of these options are simply computationally-convenient first attempts; though simulations show g_{bentlog} is an improvement over g_{uniform}, it is surely not optimal in this regard. In further work, I will look into using proposal distributions that are conditional on the current values at other loci, not just on the observations at the current locus.

4.4. Theoretical limitations of the algorithms in this paper

Both the block particle filter and the Finkelstein particle filter proposed here are intended to deal with the curse of dimensionality. However, both may fail in cases where forward densities — that is, the relative probabilities of given states at time t conditional on the state at time $t-1$ — are concentrated around particular values, and thus insufficiently ergodic. Rebeschini and Van Handel's error bounds rely on a strong ergodicity assumption, bounding the forward density away from 0 in a way that they themselves acknowledge is unrealistic in real-world cases. In a separate paper, they explore further the kind of problems that can arise when this assumption does not apply, and the regime where that failure occurs in practice.[15]

For the Finkelstein particle filter, I am not giving any formal proofs of performance, but it is clear that if ergodicity is poor enough, my algorithm will also break down. For example, suppose that the forward density from history $\boldsymbol{x}^j$ to raw locus value z_l^i is less than ϵ if $i \neq j$, and otherwise greater than 100ϵ. In that case, the MCMC will strongly tend to get stuck in states where all locus values come from the same history. Metaphorically, Sally Finkelstein would be too picky, never selecting nearby body parts that didn't match perfectly. The result would then reduce to the bootstrap particle filter, with more useless computational cost.

Do the real-world problems to which these algorithms are applied have enough ergodicity for them to function? For these algorithms to be appropriate, we'd need a situation with enough nonlinear effects that simple Kalman filters don't suffice; yet also one which still has plenty of new randomness at each time step, such that even if the forward density is not actually strongly ergodic, it is at least diffuse enough for these algorithms to work. In SLAM (simultaneous location and mapping) models for robotics applications, such situations arise. But in fluid dynamics models, chaotic dynamics are the rule. Such models can be deterministic or nearly so, with highly concentrated forward densities, yet still have interesting dynamics. Uncertainty in the initial conditions is amplified at each time step, so even in a deterministic model with new measurement tending to reduce the uncertainty at each time step, overall uncertainty can remain in equilibrium.

Due to the "picky Sally" problem explained just above, the Finkelstein algorithm as explained in current paper does not deal well with such deterministic or nearly-deterministic models. However, in a follow-up paper, I will offer a modification of this algorithm to address such models.

5. Numerical simulations

5.1. Setup

Filtering algorithms cannot be expected to precisely infer the underlying true state of the hidden Markov model. Instead, the goal is merely to infer its conditional distribution, and most of the interest of the problem lies in the fact that this distribution remains non-degenerate; as we acquire more information in order to narrow down the possible states, the state itself evolves, so we never catch up.

Thus, we cannot simply follow the recipe of a more traditional simulation study of a technique for parameter inference. In traditional parameter inference, the object of interest is the true parameter value(s), which can be arbitrarily chosen when running a simulation. Although inference algorithms may yield an inferred distribution for the parameter(s), the interpretation of this distribution as a confidence distribution (for frequentist methods)

or a credible distribution (for Bayesian methods) is in some sense not inherent to the problem; for example, in the case of Bayesian methods, a credible interval is only as valid as the priors that produce it. However, in this case, we are not making arbitrary assumptions in order to get the best or the most robust performance; the assumptions are given by the problem, and the aim is to calculate a true probability distribution. In order to efficiently measure an algorithm's performance, we'd like a setting where the correct value of the object of interest – not the point value, but the conditional distribution – is known.

So, in order to run a simulation study, I fall back on a linear Gaussian model, where the Kalman filter algorithm gives an analytically-correct filtering distribution. Of course, given that such an analytic solution does exist, one would never in practice use an inexact filtering algorithm such as those discussed by this paper. However, the ability of our more-general algorithm to roughly reproduce the results of a Kalman filter is, at the least, encouraging.

The particular linear Gaussian model I use is a model based on a progression matrix P, a novelty matrix N, and a measurement error covariance matrix E:

$$z = Px + \delta \tag{5.1}$$

$$y = z + \epsilon \tag{5.2}$$

$$\delta \sim \mathcal{N}(0, N) \tag{5.3}$$

$$\epsilon \sim \mathcal{N}(0, E) \tag{5.4}$$

P is a tridiagonal matrix. N and E are diagonal matrices with periodic structure, so that some loci are best learned about through their neighbors. Specifically, N's elements alternate between a higher and a lower variance, each value occurring at every 2nd locus, while E has a lower variance at every 5th locus.:

$$P \equiv \begin{bmatrix} b & c & & & 0 \\ a & b & c & & \\ & a & b & \ddots & \\ & & \ddots & \ddots & c \\ 0 & & & a & b \end{bmatrix} \tag{5.5}$$

$$a \equiv .4; b \equiv .35; c \equiv .05; a + b + c = .8 < 1 \tag{5.6}$$

$$N \equiv \begin{bmatrix} 1 & & & & & 0 \\ & q & & & & \\ & & 1 & & & \\ & & & q & & \\ & & & & \ddots & \\ 0 & & & & & q \end{bmatrix} \tag{5.7}$$

$$q = .25 \tag{5.8}$$

$$
E \equiv
\begin{array}{c}
\begin{array}{ccccccc} 1 & 2 & \dots & 4 & 5 & 6 & \dots & d \end{array} \\
\begin{array}{c} 1 \\ 2 \\ \vdots \\ 4 \\ 5 \\ 6 \\ \vdots \\ d \end{array}
\begin{bmatrix} e & & & & & & 0 \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & e & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ 0 & & & & & & 1 \end{bmatrix}
\end{array}
\tag{5.9}
$$

$$e = .16 \tag{5.10}$$

$$\tag{5.11}$$

The state was initialized at mean 0 and variance 5 independently at each locus. The model was run for 10 time steps, and for the particle filtering algorithms the outcome variables of 5 separate runs were averaged.
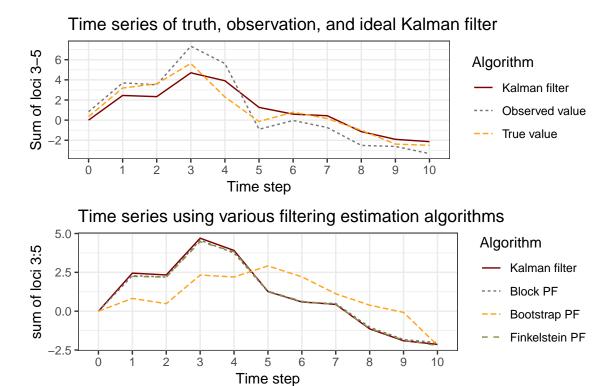
FIG 3. *Time series from a single run of each algorithm for 10 time steps, in a model with 90 dimensions. Parameters for each algorithm are given in text.*

A number of parameters were tried for the algorithms, but a good set of numbers for comparing different models was: 400 particles for the Finkelstein variants, $400^2 = 160000$ particles for the bootstrap particle filter, and $400^2/5 = 32000$ particles for the block particle filter algorithm. These numbers were chosen so that each algorithm would take roughly comparable computing time; the only step that requires computing power that is quadratic in the number of particles is precalculating the forward densities $f_l^{j\to i}$ in the Finkelstein algorithm.

All results for Figures 4 and 5 are for a 30-dimensional model. Results for both Finkelstein and block particle filter algorithms remained materially similar as model dimension was varied from 30 to 90, demonstrating that the Finkelstein and Frankenstein algorithms' errors are roughly independent of dimension, as expected.

The Finkelstein algorithm was used with ρ_{sampled}, with 45 histories per location and two formulas for the history sampling probabilities g (g_{uniform}, and g_{bentlog} with $\alpha = \beta = 5$). The neighborhood width was $r \equiv 1$, which is to say that $\max |\mathcal{B}_r(l)| = 3$. Similarly, the zone size for the block particle filter algorithm was 3.

Note that I am not the first to simulate outcomes for the block particle filter. Although Rebeschini and van Handel's paper originally proposing it relied on proofs rather than simulations, more recent papers have implemented it and given results.[10] [7] [22] The results there are not directly comparable with those given here due to different models used.

5.2. Results

To get an intuition for this situation, I will begin by showing the evolution of a single run of the model. The top panel of figure 3 shows the evolution over time of the true value ($\sum_{l=3}^{5} z_l$), the observed value ($\sum_{l=3}^{5} y_l$), and the filtering distribution as calculated by a Kalman filter ($\boldsymbol{E}_\pi(\sum_{l=3}^{5} z_l | \boldsymbol{y}_1, ..., \boldsymbol{y}_t)$, for the sum of loci 3-5. The bottom panel shows the the mean of the estimated filtering distribution for each of the four algorithms I tested — Kalman filter (analytically correct), bootstrap particle filter, block particle filter, and Finkelstein particle filter.

In the upper panel of Figure 3, one can see that the observations vary relatively widely around the truth, while the Kalman filter mean follows those observations with more conservative moves, thus staying closer to the true
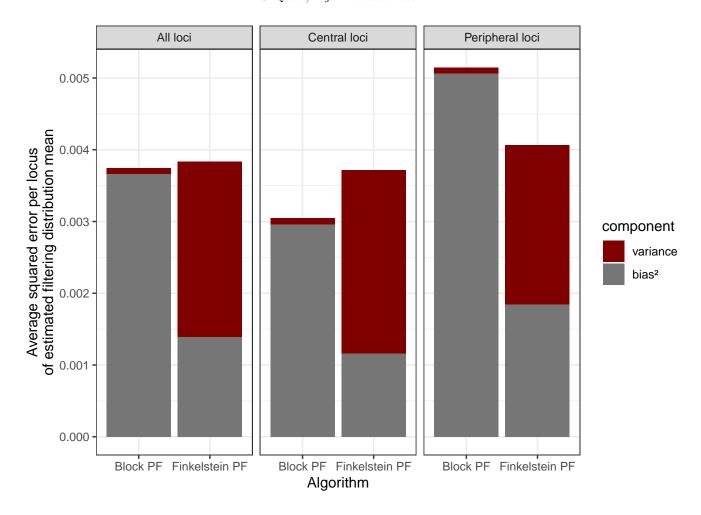
FIG 4. *Breakdown of average squared error per locus.*

value. In the lower panel, one can see that the bootstrap particle filter falls to the curse of dimensionality; while the block and Finkelstein particle filters both manage to approximate the correct Kalman filter distribution relatively well.

To compare outcomes of the two working algorithms in greater depth, Figure 4 shows the average squared error per locus: that is, the squared difference between the estimated distribution mean and the correct mean as given by the Kalman filter, conditional on a single fixed series of observations $(y_1, ..., y_{10})$. Note that we are measuring error relative to the mean of the Kalman filter rather than to z_l; this is because the Kalman filter result is the ideal filtering distribution that we are trying to capture here. Though it's not visible in these graphs, both algorithms do a relatively good job of reproducing the variance of the filtering distribution; this is within 3% of the true value for both algorithms.

Figure 4 makes two things clear. First, there is a bias/variance tradeoff between the block particle filter and the Finkelstein particle filter; with comparable run times, the block filter has a higher bias but almost no variance in its distribution error. Second, as expected, the performance of the block particle filter differs for loci that are central to their neighborhood Z_j, as opposed to loci that are on the border of their neighborhood. (The small apparent difference in performance of the Finkelstein algorithm between the two kinds of loci is largely an artifact of the specific realization of $(y_1, ..., y_{10})$ that was used to generate this graph. The Finkelstein algorithm does not use fixed neighborhoods $\{Z_j\}$, so the distinction between central and peripheral is simply does not apply, except for the first and last loci overall.)

The error of the Finkelstein algorithm, like that of the block particle filter, appears to remain stable over time, as seen in Figure 5. This shows the time evolution of the KL divergence between the true filtering distribution, as calculated using the Kalman filter, and a Gaussian with mean vector and covariance matrix inferred from
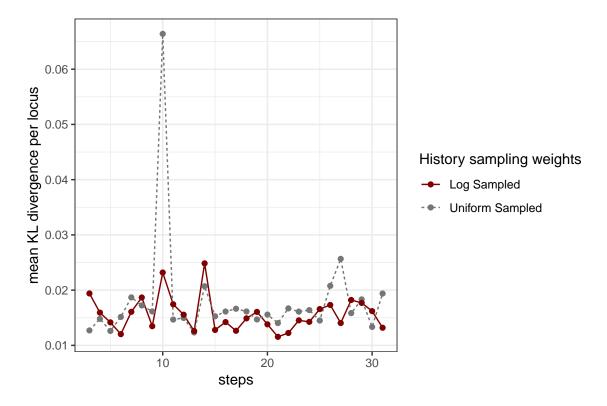
FIG 5. *Stability of KL divergence across time steps.*

a Finkelstein particle filter. It appears that uniform sampling (g_{uniform}) can occasionally be unsuccessful in sampling good histories, as reflected by the spikes in that line; log sampling (g_{bentlog}) had superior stability.

6. Conclusion

I have introduced the novel Finkelstein particle filtering algorithm for estimating the filtering distribution of models with high dimensionality due to large spatial extent. In such models, the simple bootstrap particle filtering algorithm is unusable. But, as with the previously-proposed block particle filter, my algorithm relies on the locality of dynamics to resolve this problem, focusing on a small area at a time.

Using simulations, I have showed that the error of means of my algorithm has lower bias but higher variance than the block particle filter, given comparable parameters. All in all, the total squared error of means of the Finkelstein algorithm is more homogeneous across loci than that of the block algorithm; lower for loci peripheral to a neighborhood in the block particle filter, but higher for those which are central. I also give empirical evidence that the error of this algorithm is stable over time, making it a candidate for online data assimilation tasks.

It is commonplace to prefer variance over bias when such a tradeoff is possible, because this allows improving precision with additional computing power by independent reruns of the algorithm. That improved precision would certainly be possible in this case with the Finkelstein algorithm. This picture is slightly complicated by the fact that such computing power might enable better results from the block particle filter by increasing the neighborhood size. But there are several problems with just increasing neighborhood size. Above all, computing power (that is, number of particles) needed could be up to exponential in neighborhood size, while it's just quadratic in number of Finkelstein particles or linear in independent Finkelstein runs. Second, unlike number of particles, neighborhood size comes in sizeable discrete intervals; it may not be possible to effectively use a small additional amount of computing power. And finally, to reduce the bias of the block particle filter, neighborhood size must be increased up-front, while the variance Finkelstein particle filter can in be reduced by independent runs (perhaps even by different scientists).

Thus, I believe that the Finkelstein particle filter algorithm offers meaningful advantages over prior proposals. In future work, I will extend this to cover chaotic dynamics in a deterministic or quasi-deterministic model.

References

[1] S. Agapiou et al. "Importance Sampling: Intrinsic Dimension and Computational Cost". In: *Statistical Science* 32.3 (Aug. 2017), pp. 405–431. ISSN: 0883-4237, 2168-8745. DOI: 10.1214/17-STS611.

[2] Christophe Andrieu and Gareth O. Roberts. "The Pseudo-Marginal Approach for Efficient Monte Carlo Computations". In: *The Annals of Statistics* 37.2 (Apr. 2009), pp. 697–725. ISSN: 0090-5364. DOI: 10.1214/07-AOS574. arXiv: 0903.5480.

[3] A. Apte et al. "Data Assimilation: Mathematical and Statistical Perspectives". In: *International Journal for Numerical Methods in Fluids* 56.8 (2008), pp. 1033–1046. ISSN: 1097-0363. DOI: 10.1002/fld.1698.

[4] Thomas Bengtsson, Peter Bickel, and Bo Li. "Curse-of-Dimensionality Revisited: Collapse of the Particle Filter in Very Large Scale Systems". In: *Institute of Mathematical Statistics Collections*. Beachwood, Ohio, USA: Institute of Mathematical Statistics, 2008, pp. 316–334. ISBN: 978-0-940600-74-4. DOI: 10.1214/193940307000000518.

[5] Peter Bickel, Bo Li, and Thomas Bengtsson. *Sharp Failure Rates for the Bootstrap Particle Filter in High Dimensions*. Institute of Mathematical Statistics, 2008. ISBN: 978-0-940600-75-1. DOI: 10.1214/074921708000000228.

[6] Olivier Cappe, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer, June 2009. ISBN: 978-0-387-28982-3.

[7] Alban Farchi and Marc Bocquet. "Review Article: Comparison of Local Particle Filters and New Implementations". In: *Nonlinear Processes in Geophysics* 25.4 (Nov. 12, 2018), pp. 765–807. ISSN: 1023-5809. DOI: https://doi.org/10.5194/npg-25-765-2018.

[8] Walter R Gilks and Carlo Berzuini. "Following a Moving Target – Monte Carlo Inference for Dynamic Bayesian Models". In: *Journal of the Royal Statistical Society*. Statistical Methodology 63.1 (2001), pp. 127–146. DOI: 1369-7412/01/63127.

[9] Simon Godsill and Tim Clapp. "Improvement Strategies for Monte Carlo Particle Filters". In: *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2000, pp. 139–158.

[10] M. Morzfeld, D. Hodyss, and J. Poterjoy. "Variational Particle Smoothers and Their Localization". In: *Quarterly Journal of the Royal Meteorological Society* 144.712 (2018), pp. 806–825. ISSN: 1477-870X. DOI: 10.1002/qj.3256. URL: http://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3256.

[11] Matthias Morzfeld, Xin T. Tong, and Youssef M. Marzouk. "Localization for MCMC: Sampling High-Dimensional Posterior Distributions with Local Structure". In: (Oct. 20, 2017). arXiv: 1710.07747 [math, stat].

[12] Jonathan Poterjoy. "A Localized Particle Filter for High-Dimensional Nonlinear Systems". In: *Monthly Weather Review* 144.1 (Oct. 20, 2015), pp. 59–76. ISSN: 0027-0644. DOI: 10.1175/MWR-D-15-0163.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-15-0163.1.

[13] Jonathan Poterjoy, Ryan A. Sobash, and Jeffrey L. Anderson. "Convective-Scale Data Assimilation for the Weather Research and Forecasting Model Using the Local Particle Filter". In: *Monthly Weather Review* 145.5 (Mar. 14, 2017), pp. 1897–1918. ISSN: 0027-0644. DOI: 10.1175/MWR-D-16-0298.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-16-0298.1.

[14] Patrick Rebeschini and Ramon van Handel. "Can Local Particle Filters Beat the Curse of Dimensionality?" In: *The Annals of Applied Probability* 25.5 (Oct. 2015). 00070 MR: MR3375889, pp. 2809–2866. ISSN: 1050-5164, 2168-8737. DOI: 10.1214/14-AAP1061.

[15] Patrick Rebeschini and Ramon van Handel. "Phase Transitions in Nonlinear Filtering". In: *Electronic Journal of Probability* 20 (2015). ISSN: 1083-6489. DOI: 10.1214/EJP.v20-3281.

[16] François Septier and Gareth W. Peters. "An Overview of Recent Advances in Monte-Carlo Methods for Bayesian Filtering in High-Dimensional Spaces". In: *Theoretical Aspects of Spatial-Temporal Modeling*. Ed. by Gareth William Peters and Tomoko Matsui. Tokyo: Springer Japan, 2015, pp. 31–61. ISBN: 978-4-431-55335-9. DOI: 10.1007/978-4-431-55336-6_2.

[17] Chris Snyder. *Particle Filters, the "Optimal" Proposal and High-Dimensional Systems*. Reading, UK: ECMWF, 2011, p. 10.

[18] Chris Snyder et al. "Obstacles to High-Dimensional Particle Filtering". In: *Monthly Weather Review* 136.12 (Dec. 1, 2008), pp. 4629–4640. ISSN: 0027-0644. DOI: 10.1175/2008MWR2529.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/2008MWR2529.1.

[19] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. "Simultaneous Localization and Mapping". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Springer Handbooks. Cham:

Springer International Publishing, 2016, pp. 1153–1176. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_46.

[20] Peter Jan Van Leeuwen, Yuan Cheng, and Sebastian Reich. *Nonlinear Data Assimilation*. Frontiers in applied dynamical systems reviews and tutorials 2. Cham: Springer, 2015. ISBN: 978-3-319-18347-3 978-3-319-18346-6.

[21] Peter Jan van Leeuwen. "Particle Filtering in Geophysical Systems". In: *Monthly Weather Review* 137.12 (Dec. 1, 2009), pp. 4089–4114. ISSN: 0027-0644. DOI: 10.1175/2009MWR2835.1.

[22] Xu Yaxian. "Sequential Monte Carlo Algorithms Fir High-Dimensional Filtering and Smoothing". Thesis. Apr. 6, 2018. URL: http://scholarbank.nus.sg/handle/10635/146933.