PARTICLE DEGENERACY: ROOT CAUSE AND SOLUTION

Fred Daum & Jim Huang

Raytheon

ABSTRACT

We have solved the well known and important problem of particle degeneracy for particle filters. Our filter is roughly seven orders of magnitude faster than standard particle filters for the same estimation accuracy. The new filter is four orders of magnitude faster per particle, and it requires roughly three orders of magnitude fewer particles to achieve the same accuracy as a standard particle filter. Typically we beat the EKF or UKF accuracy by approximately two orders of magnitude for difficult nonlinear problems.

Index Terms— particle filter, nonlinear filter, extended Kalman filter, estimation, particle degeneracy, curse of dimensionality

1. INTRODUCTION

We derive a new theory for nonlinear filters that solves the well known problem of "particle degeneracy". Our filter differs radically from other particle filters in many ways: (1) we do not resample particles, thereby avoiding the major computational burden of particle filters; (2) we do not rely on a proposal density; (3) we compute the log of the unnormalized density using a particle flow, rather than as a pointwise multiply of two functions; and (4) we do not use importance sampling or any other MCMC method. Particle flow moves the particles to the correct region of state space for the computation of Bayes' rule; this solves the problem of "particle degeneracy" (see section 1 of [1] for a detailed explanation of this problem, including many references). For brevity, we refer the reader to [5] for a survey of the nonlinear filter problem, including notation and assumptions.

The workhorse algorithm for nonlinear filter problems is the extended Kalman filter (EKF), which is used in thousands of real World applications, including tracking, navigation, communications, robotics, multisensor data fusion, financial engineering, weather prediction, etc. The EKF often provides good estimation accuracy, but it is surprisingly bad in many important real World applications. Particle filters (PFs) promise to improve estimation accuracy by computing the complete non-Gaussian conditional probability density of the state given the history of measurements. But the real time computational complexity of PFs is orders of magnitude larger than for the EKF for typical problems. It is well known that PFs suffer from the curse of dimensionality [17]. In particular, figure 1 shows the estimation error of the classic particle filter normalized to optimal accuracy plotted vs. the number of particles for problems with various dimensions of the state vector of the plant (d = 1 to 10). Evidently, the PF provides optimal accuracy for low dimensional problems (d = 1 & 2 & 4) using a reasonable number of particles. However, for higher dimensional problems (d = 6 & 8 & 10), the accuracy of the PF is many orders of magnitude worse than optimal even with a very large number of particles. This phenomenon has been noticed by many other engineers and researchers. Moreover, there is now a very elegant mathematical proof that PFs generally suffer from the curse of dimensionality (see [7] for details and many references). The specific problem used for figure 1 is not a pathological example, but rather it is the easiest problem (Gaussian and linear and exactly solved by the Kalman filter); that is, the example used in figure 1 is not multimodal or non-smooth or highly nonlinear or otherwise difficult, but rather it is as smooth and as nice as possible. In the old days, before we first published figure 1, over eight years ago, one would often hear or read assertions that: "the PF beats the curse of dimensionality," but we no longer hear or read such assertions. Further details on this crucial issue, as well as a litany of attempts to mitigate it, are given in the excellent recent tutorials in [9] and [11], which are refreshingly honest.

It is well known (by some) that standard PFs cannot be run efficiently on parallel processors, owing to the bottleneck caused by resampling of particles (see [8] for details). That is, a normal engineer would naively expect that using M processors to implement a PF would result in roughly a factor of M speed-up in computation time, because the prediction of N particles is easily parallelizable, because the particles are decoupled from each other in the prediction step. But resampling of particles requires the N particles to be used jointly, thereby destroying the easy parallelization. In particular,

Signal Processing, Sensor Fusion, and Target Recognition XX, edited by Ivan Kadar,
Proc. of SPIE Vol. 8050, 80500W · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.877167

using thousands of processors in [8] results in less than an order of magnitude speedup, even after the resampling method was modified specifically to address this issue. In contrast, our particle flow filter does not resample particles, and hence it does not suffer from this bottleneck to parallelization, and thus it is embarrassingly parallelizable. This is in addition to the fact that our filter runs four orders of magnitude faster per particle than standard PFs, and it is also in addition to the fact that our flow filter requires many orders of magnitude fewer particles to achieve the same accuracy.

2. ROOT CAUSE OF PARTICLE DEGENERACY

Standard PFs suffer from particle degeneracy, which severely degrades filter accuracy. "Particle degeneracy" means that almost all of the particles are wasted, because they are in the wrong place in state space. Figure 3 shows a one dimensional example, which gives an intuitive notion of this problem. The ten green dots are particles, which are points at which we compute the numerical value of the probability density. In this example, the ten particles do an excellent job of representing the blue curve (the density of x before a measurement). The red curve is the likelihood of the latest measurement. Bayes' rule is simply computing the product of the blue and red curves. If we compute this product at each particle, it gives a very bad approximation to the product; in particular, there are no particles near the peak of the product of the red & blue curves. Hence, all of the ten particles are useless to represent the product. Of course we could use more particles to improve the situation; for example, using 100 particles rather than only 10 particles solves this problem for our d = 1 toy example. Unfortunately, this simple and obvious idea does not work in higher dimensions (e.g., d = 12), where we would need a huge number of particles to represent the product accurately. The effect of particle degeneracy is obvious in Figure1, where the number of particles required to achieve optimal accuracy grows by roughly one or two orders of magnitude for each additional dimension of the state vector. It also explains why N = 100 particles are required for d = 1, rather than only N =10 particles, which would be plenty to represent the posteriori density accurately for d = 1 if we only knew where to position the particles. This is the fundamental problem with current particle filters, which is called "particle degeneracy", and which is the root cause of the curse of dimensionality for PFs. Moreover, the problem of particle degeneracy gets worse with more accurate measurements; in particular, it is intuitively obvious from Figure 3 that if the red curve is narrower we would need many more particles to represent the product of the red and blue curves accurately.

Another obvious solution to particle degeneracy is to sample new particles from the likelihood. We can do this very efficiently, despite the fact that the likelihood is not a density of x. This is exactly what many standard PFs do, and in fact, the results in Figure 1 use this method. But it is obvious that this method does not work well in high dimensions. That is because we do not measure all components of the state vector, but rather in typical applications we only measure a few components. If we actually measured all components of the state vector, then this simple method would work in high dimensions.

A third obvious solution is to sample new particles from a good proposal density. For example, use an EKF or UKF to compute a Gaussian density, and use that for sampling new particles. This simple idea works well for examples in which the EKF or UKF are reasonably accurate, but not otherwise. If the EKF or UKF were reasonably accurate for a given problem, then one would wonder why we need a PF at all. Of course, there are examples of this type in the literature, but we cannot count on this happy situation in general. The fundamental problem is that it is not easy to compute a good proposal density. This notion is now well known among PF researchers.

Our solution to particle degeneracy is to move the particles to new positions in state space that are good for representing the product of the blue and red curves. Of course, this is a chicken and egg problem; how do you know good positions for particles to represent the product before you compute the product itself? However, this does not mean that the problem is hopeless; in particular, we have solved other chicken & egg problems before (e.g., optimal control computes the best control for a dynamical system without computing all possible future paths of the state vector). We solve this chicken & egg problem using the method shown in Figure 4. In particular, we compute the flow of probability density from the prior to the posteriori using a very simple definition, and we compute the flow of particles induced by this flow of density using calculus. The particle flow is derived in the next section.

The problem of particle degeneracy is well known and it is very important; see section 1.0 of [1] for many references and further discussion.

3. DERIVATION OF PDE FOR PARTICLE FLOW

We derived a PDE that governs the desired particle flow for Bayes' rule in [1], whereas in this paper we will give a more perspicuous derivation. The basic idea is to

compute a flow of particles induced by the following flow of the conditional unnormalized probability density of x:

$$\log p(x,\lambda) = \log g(x) + \lambda \log h(x)$$

in which x is the d-dimensional state vector of the plant, and g(x) denotes the prior unnormalized probability density of x, and h(x) is the likelihood, as explained in [1]. The variable λ goes continuously from 0 to 1, analogous to time but it is distinct from time. For λ = 0, the function p(x,λ) is equal to the prior density, whereas for λ =1, p(x,λ) is equal to the desired posteriori density of x conditioned on all measurements up to and including the current time. It turns out that we are forced to use logp(x,λ) rather than p(x,λ) in this flow, to avoid singularities or other problems. We did not decide to use the log for improved numerical reasons; however, logp is vastly superior to p for this reason also.

Next, we suppose that there exists a continuous flow of particles induced by the flow of probability density above, with the following dynamics:

$$\frac{dx}{d\lambda} = f(x,\lambda)$$

Therefore, the probability density of x will obey the Fokker-Planck equation throughout this flow:

$$\frac{\partial p(x,\lambda)}{\partial \lambda} = -Tr\left[\frac{\partial (p(x,\lambda) f(x,\lambda))}{\partial x}\right]$$

For simplicity we have assumed zero process noise (i.e., zero diffusion) in the flow of x; it is also possible to derive an analogous PDE for the particle flow with non-zero diffusion. Using the definition of p(x,λ) and simple calculus results in:

$$\frac{\partial \log p(x,\lambda)}{\partial \lambda} = \log h(x)$$

Combining the last two equations, along with some more calculus gives the following:

$$\log h(x) p(x,\lambda) = -p(x,\lambda) Tr\left[\frac{\partial f}{\partial x}\right] - \frac{\partial p}{\partial x} f$$

Assuming that p(x,λ) is nowhere vanishing, and dividing by p we obtain the desired PDE:

$$\log h = -div(f) - \frac{\partial \log p}{\partial x} f$$

As shown in [1], we can also put this PDE into divergence form by defining q(x,λ) = p(x,λ)f(x,λ):

$$div(q) = \eta$$

in which η = -p(x,λ)logh(x), and div denotes the divergence.

These two PDEs are both linear first order PDEs in the unknown functions (f or q). Also, both of these PDEs are highly underdetermined, because there is only one scalar valued equation, but the unknown function is d-dimensional. That is, we generally have many more unknowns than equations. Also, there are no boundary conditions on f or q. Therefore, as emphasized in [1], we have great freedom in computing f or q, which we can exploit to reduce computational complexity. Many distinct methods for solving these PDEs are described in [1] and [3], and we will introduce several new methods in this paper. The important issues in solving these PDEs are: (1) we want low real time computational complexity to solve this PDE; (2) we want stability of the particle flow (viewed as a dynamical system evolving in λ); (3) we want adequate accuracy of the particle flow to provide good filter accuracy; and (4) we need to decide how to pick a unique solution of the PDE. As shown in [1] and [3], there are many ways to achieve these goals (see figure 5 for a summary of solutions).

4. SEPARATION OF VARIABLES

As shown in [1], if we assume that both the prior density and the likelihood are Gaussian densities, then an exact solution for our particle flow corresponding to Bayes' rule is:

$$\frac{dx}{d\lambda} = A(\lambda)x + b(\lambda)$$

in which

$$A = -\frac{1}{2} PH^T (\lambda HPH^T + R)^{-1} H$$
$$b = (I + 2\lambda A)[(I + \lambda A)PH^T R^{-1} z + A\bar{x}]$$

where P is the covariance matrix of prediction error in x for the Gaussian prior density, and H is the measurement

matrix (i.e., z = Hx+v), and R is the covariance matrix of the measurement noise (v). The symbol $\bar{x}$ denotes the predicted value of x, corresponding to the mean value of the prior Gaussian density. Inspection of the matrix A shows that the particle flow is stable (under very mild conditions). In particular, if P and R are positive definite, then A is a negative semidefinite symmetric matrix. Therefore, the eigenvalues of A are real and nowhere positive, and thus the particle flow corresponding to Bayes' rule is Schur stable. It is well known that a linear time varying system can be unstable, despite having all of its eigenvalues in the left half plane; however, in our case, one can construct a suitable Lyapunov function to prove stability. Moreover, in all of our numerical experiments, the particle flow is always stable. One can easily compute H and P by using the standard EKF or UKF approximations. The real time computational complexity for this solution is extremely fast, because f is given by a formula which can be evaluated at each particle x. As shown in [1], the exact particle flow derived here is many orders of magnitude faster than standard particle filters for any given number of particles, because we do not resample particles.

The method of separation of variables described above can be generalized to solve our PDE using the exponential family of probability densities rather than Gaussian densities (e.g., see [13] & [15]).

5. DIRECT INTEGRATION

First, we write our PDE in a more perspicuous form:

$$\frac{\partial q_1}{\partial x_1} + \frac{\partial q_2}{\partial x_2} + \ldots + \frac{\partial q_d}{\partial x_d} = \eta$$

Suppose that we ask a freshman to solve this PDE for q as a homework problem. A typical freshman would say: "Well, that is easy! Just set all the components of q to zero except for one, and solve for the non-zero component of q by integration". This is a correct and exact solution, but it is far from the most general solution, and moreover, it is not a useful solution for particle flow, because the resulting flow is not stable (all but one of the eigenvalues of the Jacobian of the flow are zero), and also it does not move the particles in all coordinates. We need a stable flow that is full rank. One could try to fix this freshman solution by using superposition, because our PDE is linear with constant coefficients. In particular, form a superposition of d solutions with coefficients that sum to unity. But this does not give the correct solution, because the d freshman solutions are generally not compatible with each other. We need some kind of compatibility conditions as explained in [3].

Suppose that we have an exact solution of this PDE for a related problem with the same dimension d, but with a different right hand side, as follows:

$$div(\tilde{q}) = \tilde{\eta}$$

For example, we can obtain such exact solutions for any problem with Gaussian prior and likelihood, where the exact solution is given in the previous section. In particular, we can use the linearization of our problem to compute the matrices H, R and P used to compute A and b.

Combining these two equations, we get:

$$div(q - \tilde{q}) = \eta - \tilde{\eta}$$

We can now use the freshman solution described above (but applied to this new equation rather than div(q) = η) by picking all components of q but one equal to $\tilde{q}$. Finally, we can integrate the selected non-zero component to compute the exact solution as follows:

$$q_j = \tilde{q}_j + \int^{x_j} \eta(x,\lambda) - \tilde{\eta}(x,\lambda) dx_j$$

for any one j of our choice, and $q_i = \tilde{q}_i$ for i ≠ j.

We can exploit the freedom to choose the particular non-zero component of $q - \tilde{q}$ to make the evaluation of this integral fast and accurate. For example, we could pick j such that the integrand, $\eta - \tilde{\eta}$ is smooth and as flat as possible in x_j, or else is well approximated by a polynomial in the variable x_j. We could also pick j to give the most robustly stable flow. The computational complexity of this solution is minimal, owing to the use of our exact solution for d-1 components of q, and the fact that this is an integral with respect to only one component of x. In contrast to the freshman solution of div(q) = η this exact solution is generally full rank, and it is stable, owing to the stability of the flow corresponding to our exact solution for a related problem.

For example, if $\eta - \tilde{\eta}$ is approximately constant in x_j, then the following extremely simple approximation should be adequate:

$$q_j \approx \tilde{q}_j + x_j(\eta - \tilde{\eta})$$

One can generalize this type of approximation by using higher order Taylor series in x_j expanded about a good point. Moreover, the likelihood h(x) is known exactly as a formula, and thus it can be computed very fast at any point x; we can exploit this fact to speed-up the integration, either numerically or analytically. In contrast, the prior density, g(x) is only known at random points in d-dimensional state space, but the prior typically varies much more slowly in x than the likelihood, and this fact can be exploited to craft a fast and accurate approximation to the integral.

It is remarkable that we have derived an exact solution to our PDE for arbitrary smooth nowhere vanishing densities, rather than being specialized to Gaussian densities or densities from exponential families. This is true despite the fact that the related solution $\tilde{q}$ was based on a linear approximation like the EKF or UKF.

6. POISSON'S EQUATION

As shown in [1], if we assume that the function q = fp is the gradient of a scalar potential, V(x, λ), then we get Poisson's equation for the potential. It turns out that this is exactly the same as assuming that we want the unique minimum L2 norm solution. There is a vast literature on solving Poisson's equation fast and accurately in high dimensions for scattered data. In particular, the well known "fast multipole methods" (FMM) could be used here. We emphasize, however, that we do not need excellent accuracy of the flow, but rather one percent accuracy of the flow should be adequate for most practical applications. Therefore, we can exploit this fact by using faster and simpler methods than FMM, which has the reputation of complex code in order to achieve solutions within machine precision, or one part in one million or one part in a billion accuracy. We do not need such high accuracy here. For example, one could approximate the convolution integral for the gradient of the potential (equation (27) in [1]) using the following simple formula (for d > 2):

$$\frac{\partial V(x,\lambda)}{\partial x} = \int \log h(y)\, p(y,\lambda) \frac{c(2-d)(x-y)^T}{\|x-y\|^d} dy$$

$$\frac{\partial \mathrm{V}(\mathrm{x},\lambda)}{\partial \mathrm{x}} = \mathrm{E}\left[\log\mathrm{h}(\mathrm{y}) \frac{\mathrm{c}(2-\mathrm{d})(\mathrm{x}-\mathrm{y})^{\mathrm{T}}}{\|\mathrm{x}-\mathrm{y}\|^d} \right]$$

$$\frac{\partial \mathrm{V}(\mathrm{x}_\mathrm{i},\lambda)}{\partial \mathrm{x}} \approx \sum_{j \in S_i} \log h(x_j) \frac{c(2-d)(x_i - x_j)^T}{\|x_i - x_j\|^d}$$

in which E(.) denotes the expected value with respect to the density p(y,λ), the symbol x_i denotes the i^{th} particle, and c is defined in [1]. The summation is over the subset of k nearest neighbors of a given particle, rather than over all particles. If we summed over all particles, then the computational complexity would be quadratic in N, whereas the formula above has complexity linear in N. This is essentially the same computational issue for all methods of solution of Poisson's equation for N particles. The well known FMM algorithms use a different approximation to reduce computational complexity to NlogN or N (computing the effect of distant clumps of particles by averaging rather than treating each individual particle); this is a far-field approximation, which is well known and extremely useful in physics. Our approximation is similar to Coulomb's law, but in d-dimensions rather than d = 3, and it gives the velocity vector rather than the acceleration vector. It is pleasantly surprising that this approximation improves with higher dimension, owing to the rapid decay of the Newton potential with distance. Moreover, we emphasize that we have a very fast approximate k-NN (nearest neighbor) algorithm, which we developed several years ago for our incompressible flow filter [4]. Yet another approach to solving Poisson's equation accurately in high dimensions (with computational complexity that is linear in d) was recently reported in [6], which can be extended to scattered data.

One can derive a much more stable particle flow using equation (26) in [1] as follows. Recall that the gradient of the potential given by the convolution integral for Poisson's equation is denoted by $\frac{\partial V(x,\lambda)}{\partial x}$, and it can be computed exactly as follows:

$$= -\int \frac{c\left[\frac{\partial \log h(y)}{\partial y} + \log h(y)\frac{\partial \log p(y)}{\partial y}\right] p(y)dy}{\|x-y\|^{d-2}}$$

$$= -cE\left[\frac{\left(\frac{\partial \log h(y)}{\partial y} + \log h(y)\frac{\partial \log p(y,\lambda)}{\partial y}\right)}{\|x-y\|^{d-2}}\right]$$

$$\approx -c\sum_{j\in S_i}\left[\frac{\frac{\partial \log h(x_j)}{\partial x} + \log h(x_j)\frac{\partial \log p(x_j,\lambda)}{\partial x}}{\|x_i - x_j\|^{d-2}}\right]$$

This new particle flow is similar to the incompressible flow in certain formulaic ways, but of course we have not assumed incompressibility of the flow here. Moreover, the stability properties of this new flow are much better than the flow derived from d-dimensional Coulomb's law, which turns out to be somewhat problematical. There are very interesting and useful analogies with stability of flow in plasma physics which will be discussed elsewhere.

7. OTHER SOLUTIONS

There are many other methods to solve our first order linear underdetermined PDE, as explained in [1] and [3]; see figure 5 for a summary of many solutions, including: (1) separation of variables using the exponential family of densities rather than Gaussian densities; (2) generalized method of characteristics; (3) Fourier transform of the divergence form of our PDE (which has constant coefficients); (4) optimal control; (5) homotopy method inspired by Gromov's h-principle; (6) variational method of Gauss & Hertz (rather than standard variational method); (7) most general solution using the generalized inverse of a linear differential operator and the orthogonal projection; and hybrids of the above solutions. We emphasize that all of these methods are analytical solutions of our PDE, whereas we could also use numerical methods to solve our PDE (e.g., differential quadrature due to Bellman; meshfree PDE solutions using partition of unity or quasi-interpolation, etc.).

8. MORE GENERAL FLOW

As noted in the derivation of the PDE for our particle flow given in section 3, as well as in [1], we assumed that the diffusion (aka process noise) was zero for simplicity. We emphasize that here we are talking about the flow of particles corresponding to Bayes' rule, rather than the flow for the plant, which generally has non-zero process noise. However, it is easy to derive an analogous PDE for the flow of particles for Bayes' rule using non-zero diffusion. In this case we get the Fokker-Planck equation rather than a first order PDE. Fortunately, we have developed exact solutions to the Fokker-Planck equation for a wide class of problems for nonlinear filters, including Gaussian (i.e., Kalman filter), potential flow (i.e., Benes filter), as well as more general filters corresponding to the exponential family of probability densities (see [12]-[15] for details). Of course, the usual meaning of the word "solving" the Fokker-Planck equation is that we are given f(x,λ) and we wish to find p(x,λ), whereas in this paper we are doing the reverse; that is, we are given p(x,λ) and we want to find one of the corresponding flows f(x,λ). But our exact solutions given in [12]-[15] facilitate this reverse solution as well. We should actually consider f & p as a pair, rather than as a given function and the solution; in that sense, [12]-[15] provides a catalog of such pairs which we can exploit. Furthermore, in this paper, we obtained Poisson's equation by assuming that q was the gradient of a potential, whereas we could also ask for f as the gradient; this will result in a steady-state Fokker-Planck equation for the potential, and we can exploit the exact solutions in [12]-[15] here as well. Yet another way to solve the Fokker-Planck equation is using the Feynman-Kac formula, along with Dirac's approximation (see [16] for details). For high dimensional problems with scattered data (i.e., particles) and low accuracy requirements for the flow, we suspect that Monte Carlo would be the method of choice for solving the Fokker-Planck equation, based on numerical experiments and intuition. We will report on these more general particle flows elsewhere.

9. NUMERICAL EXPERIMENTS

We have made many numerical experiments to test the accuracy of the state vector estimate as well as the computational complexity of our new theory. Owing to page limitations for this paper, our numerical results are reported elsewhere (see [1] and [2]). In particular, our filter is four orders of magnitude faster (per particle) than standard particle filters, and we need roughly three orders of magnitude fewer particles to achieve the same estimation accuracy compared with standard particle filters. Therefore, the net reduction of computational complexity is approximately seven orders of magnitude.

We have tested our filter for many different conditions: (1) dimension of the state vector of the plant from 1 to 30; (2) stable as well as unstable plants; (3) various measurement nonlinearities (quadratic and cubic); (4) linear Gaussian problems with both stable and unstable plants; (5) various amounts of initial uncertainty in the state vector; (6) radar tracking problems with d = 6 with and without range measurements, and other applications. In general, our filter beats the EKF by roughly two orders of magnitude for difficult nonlinear problems, and our filter is essentially optimal for linear Gaussian problems.

ACKNOWLEDGEMENTS

We are extremely grateful to Professors Poisson and Coulomb and Gauss for paving the way for our research in particle flow. We thank Bhashyam Balaji for emphasizing that we can solve the Fokker-Planck equation using fast path integral methods, as well as for a lengthy and spirited flow of emails on probability density prediction, quantum field theory and particle flow.

REFERENCES

[1] Fred Daum, Jim Huang and Arjang Noushin, "Exact particle flow for nonlinear filters," Proceedings of SPIE Conference, Orlando Florida, April 2010.

[2] Fred Daum and Jim Huang, "Numerical experiments for nonlinear filters with exact particle flow induced by log-homotopy," Proceedings of SPIE Conference, Orlando Florida, April 2010.

[3] Fred Daum and Jim Huang, "Exact particle flow for nonlinear filters: seventeen dubious solutions to a first order linear underdeteremined PDE," Proceedings of IEEE Conference on Signals, Systems & Computers, Asilomar California, November 2010.

[4] Fred Daum, Jim Huang, Misha Krichman and Talia Kohen, "Seventeen dubious methods to approximate the gradient for nonlinear filters with particle flow," Proceedings of SPIE Conference, San Diego CA, August 2009.

[5] Fred Daum, "Nonlinear filters: beyond the Kalman filter," IEEE AES Systems Magazine, special tutorial issue, August 2005.

[6] Flavia Lanzara, Vladimir Maz'ya and Gunther Schmidt, "On the fast computation of high dimensional volume potentials," arXiv:0911.0443v1 [Math.NA], 2 November 2009.

[7] Paul Bui Quang, Christian Musso and Francois Le Gland, "An Insight into the Issue of Dimensionality in Particle Filtering," Proceedings of 13th international conference on information fusion, Edinburgh Scotland, July 2010.

[8] Gustaf Hendeby, Jeroen D. Hol, Rickard Karlsson, Fredrik Gustafsson, "A graphics processing unit implementation of the particle filter," Proceedings of 15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007.

[9] Arnaud Doucet and Adam Johansen, "A Tutorial on Particle Filters and Smoothing: Fifteen years later," available on-line, version 1.1, December 2008; also, to be published in [10].

[10] Dan Crisan and Boris Rozovsky (editors), "The Oxford University handbook of nonlinear filtering," Oxford University Press, 2011.

[11] Fredrik Gustafsson, "Particle filter theory and practice with positioning applications," IEEE Aerospace & Electronics Systems Magazine, special tutorial issue, volume 25 number 7, July 2010.

[12] Fred Daum, "Exact Finite Dimensional Nonlinear Filters," IEEE Trans. Automatic Control, Vol. AC-31, No. 7, pp. 616-622, July 1986.

[13] Fred Daum, "Solution of the Zakai Equation by Separation of Variables", IEEE Trans. Automatic Control, Vol. AC-32, No. 10, pp. 941-943, Oct. 1987.

[14] Fred Daum, "New Nonlinear Filters and Exact Solutions of the Fokker-Planck Equation", Proceedings of IEEE American Control Conference, Seattle, Wash., Pages 884-888, 1986.

[15] Fred Daum, "New Exact Nonlinear Filters", Chapter 8 in Bayesian Analysis of Time Series and Dynamic Models, ed. by J. C. Spall, New York: Marcel Dekker, Inc., 1988.

[16] Bhashyam Balaji, "Continuous-discrete path integral filtering," Entropy volume 11 pp. 402-430, August 2009.

[17] Fred Daum and Jim Huang, "The curse of dimensionality for particle filters," Proceedings of IEEE Aerospace Conference, Big Sky Montana, March 2003.

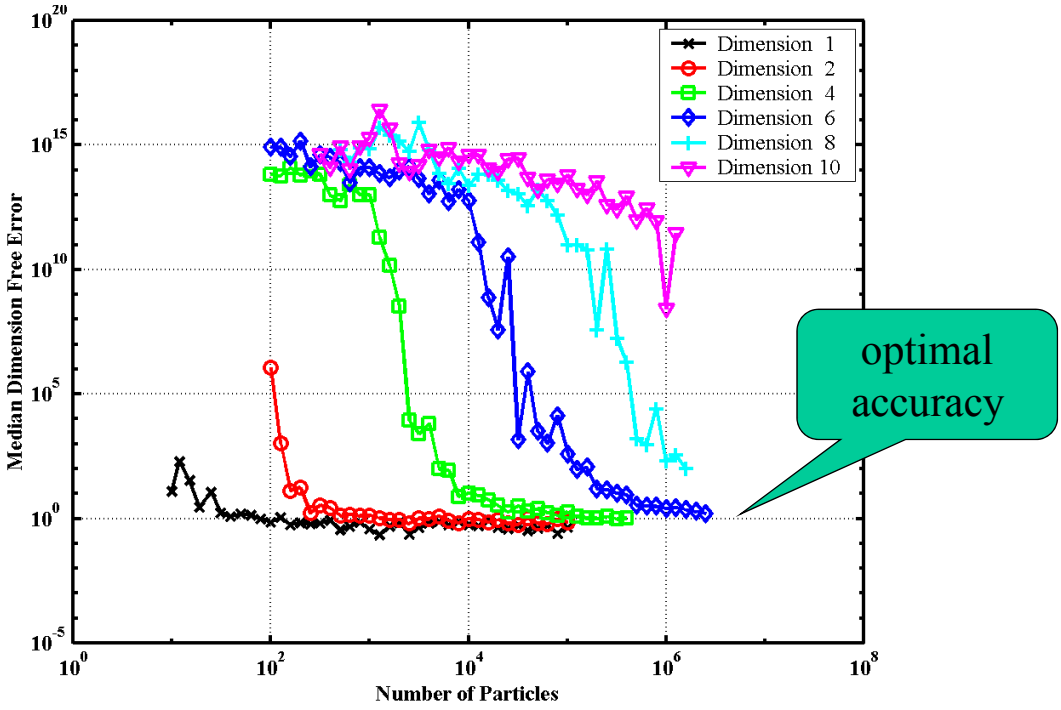

Figure 1 Curse of dimensionality for standard particle filters (from [17])

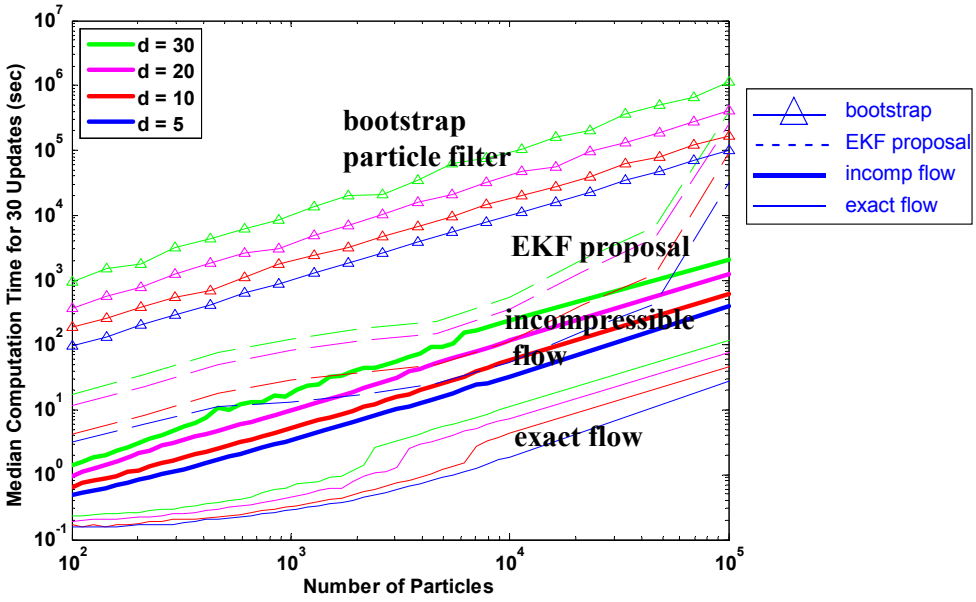

Figure 2 Comparison of runtime for four filters (MATLAB on a PC)

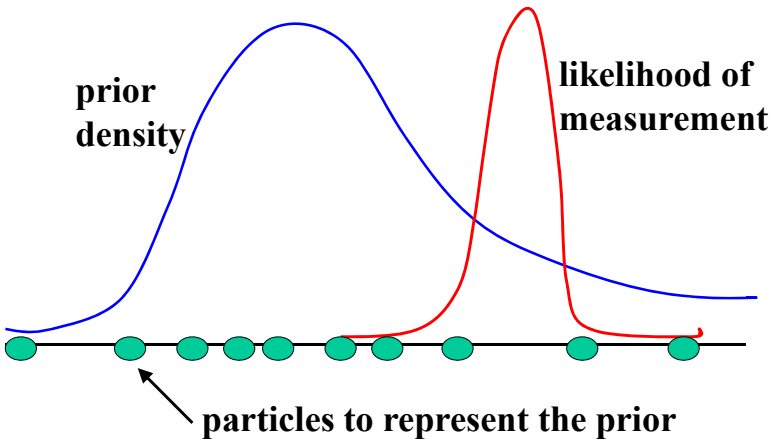

Figure 3 Intuition into particle degeneracy

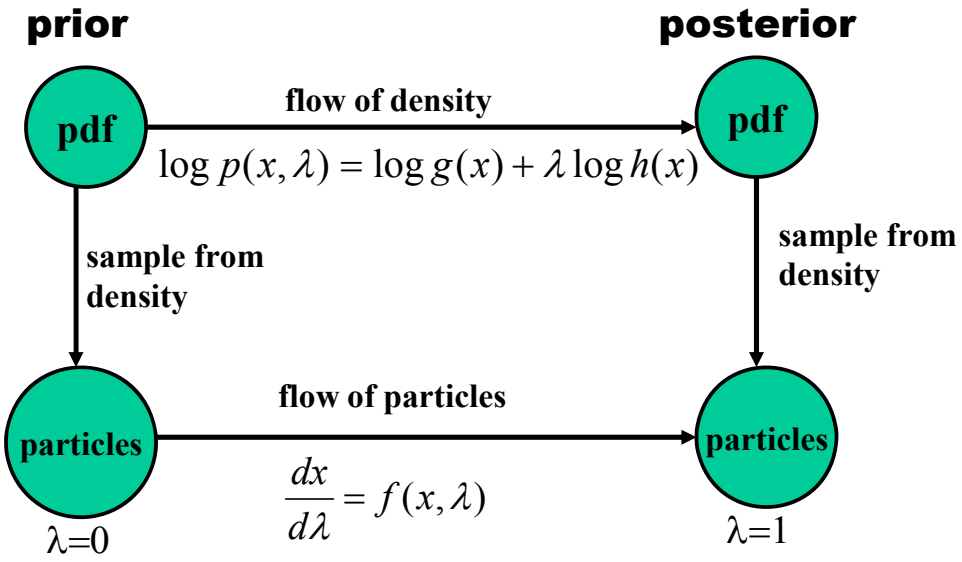

Figure 4 Induced flow of particles for Bayes' rule

method to solve PDE	method to pick unique solution	computation
1. generalized inverse of linear differential operator	minimum L2 norm	fast Poisson solver in d-dimensions
2. Poisson's equation	gradient of potential (assume irrotational flow)	fast Poisson solver in d-dimensions
3. generalized inverse of gradient of log-homotopy	assume incompressible flow (i.e., divergence free flow)	fast (but need to compute the gradient from random points)
4. most general solution	most robustly stable filter or random pick, etc.	fast (but need to compute the gradient from random points)
5. separation of variables (Gaussian)	pick solution of specific form (polynomial)	extremely fast (formula)
6. separation of variables (exponential family)	pick solution of specific form (finite basis functions)	very fast (formula)
7. variational formulation (Gauss & Hertz)	convex function minimization	ODEs
8. optimal control formulation	convex functional minimization	Euler-Lagrange PDEs (or maybe ODES for nice problem)
9. direct integration (of first order linear PDE in divergence form)	choice of d-1 arbitrary functions	one-dimensional integral
10. generalized method of characteristics	more conditions (e.g., curl = given & chain rule)	ODEs from chain rule
11. another homotopy (inspired by Gromov's h-principle)	initial condition of ODE & uniqueness of sol. to ODE	ODEs from homotopy
12. finite dimensional parametric flow (e.g., f = Ax+b)	non-singular matrix to invert	d^3 or d^6 (least squares for d or d^2 parameters, i.e., A & b)
13. Fourier transform of PDE (divergence form of PDE has constant coefficients)	minimum L2 norm or most stable flow or other method	Gaussian sum makes inverse very fast (by inspection)

Figure 5 Methods to solve first order linear highly underdetermined PDE for particle flow