

# Implicit Particle Filtering *via* a Bank of Nonlinear Kalman Filters

Iman Askari <sup>a</sup>, Mulugeta A. Haile <sup>b</sup>, Xuemin Tu <sup>c</sup>, Huazhen Fang <sup>a</sup>

<sup>a</sup>*Information & Smart Systems Laboratory, Department of Mechanical Engineering, University of Kansas*

<sup>b</sup>*Vehicle Technology Directorate, U.S. Army Research Laboratory*

<sup>c</sup>*Department of Mathematics, University of Kansas*

---

## Abstract

The implicit particle filter seeks to mitigate particle degeneracy by identifying particles in the target distribution's high-probability regions. This study is motivated by the need to enhance computational tractability in implementing this approach. We investigate the connection of the particle update step in the implicit particle filter with that of the Kalman filter and then formulate a novel realization of the implicit particle filter based on a bank of nonlinear Kalman filters. This realization is more amenable and efficient computationally.

*Key words:* Nonlinear state estimation; particle filter; implicit particle filter

---

## 1 Introduction

State estimation has found significant application in all areas of science and engineering to accelerate technological advancements. The particle filtering (PF) approach has arisen as a crucial means, especially for nonlinear non-Gaussian systems. At the core, PF updates an ensemble of particles *via* sequential importance sampling such that their empirical distribution approximates the target distribution of the system's state. However, the PF often suffers from particle degeneracy, whereby a majority of the particle's weights diminish to almost zero, leading to a poor approximation of the target distribution. Even though one can use a large number of particles so that at least some of them have significant weights, the number of particles needed often grows catastrophically with the system's dimension.

A survey of the literature indicates an ongoing search for principled ways to address the above issue. A straightforward, popular method is resampling, which randomly replaces low-weight particles with high-weight ones [1]. More sophisticated mechanisms have also been developed, e.g., [2,3,4,5]. Among them, the method of implicit sampling has inspired much attention. Implicit sampling exploits the idea that fewer particles will

be needed as long as they lie in the high-probability regions of the target distribution [2]. By design, it constructs probability distributions assumed for the particles and utilizes them as a reference to select particles from the target distribution's high-probability regions. The resultant implicit PF (IPF) shows effectiveness in keeping the number of particles manageable even for high-dimensional systems [6,7]. However, implementing IPF is non-trivial as it entails solving a nonlinear optimization problem and then a nonlinear algebraic equation. While different numerical approaches are used in [6,7,8,9], the computation is expensive and tedious. This motivates the development of efficient alternative approaches to approximate the IPF.

This paper proposes that the IPF framework can be approximately realized by a bank of nonlinear Kalman filters (KFs). This realization shows more computational tractability while maintaining the IPF's merit to sample from regions of high probability to achieve high estimation performance. Centering around this insight, we first introduce implicit importance sampling. This will provide a basis to understand the IPF and a link to connect the IPF with the KFs. Then, we synthesize the approach of using a bank of nonlinear KFs to approximately execute the IPF. Here, we particularly use the extended and unscented KF (EKF and UKF, respectively), as they are among the most commonly used KFs. Finally, we discuss the connections between the proposed IPF realization with several existing PF methods and validate our approach through simulation.

---

*Email addresses:* askari@ku.edu (Iman Askari), mulugeta.a.haile.civ@mail.mil (Mulugeta A. Haile), xuemin@ku.edu (Xuemin Tu), fang@ku.edu (Huazhen Fang).

## 2 Implicit Importance Sampling

In general, a Bayesian estimation problem involves computation of the conditional expectation of an arbitrary function as follows:

$$\mathbb{E}[g(x) | y_{1:T}] = \int g(x) p(x | y_{1:T}) dx, \quad (1)$$

where  $x \in \mathbb{R}^n$ ,  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is an arbitrary function, and  $p(x | y_{1:T})$  is the posterior probability density function (PDF) of  $x$  conditioned on discrete measurements  $y_{1:T} = \{y_1, y_2, \dots, y_T\}$  [1]. The above integral often prohibits closed-form evaluation for a nonlinear  $g$ . As a remedy, the Monte Carlo method approximates  $p(x | y_{1:T})$  by a set of particles (samples) and then computes the integral in (1) based on the particles. Yet, drawing particles directly from  $p(x | y_{1:T})$  can be challenging. Importance sampling thus proposes to sample from a different distribution  $q(x)$ , called importance distribution, and weight the particles accordingly. However, it may require an enormous number of particles to achieve just fair accuracy, because the particles taken from  $q(x)$  may fall into low-probability regions of  $p(x | y_{1:T})$ . To ameliorate this problem, a promising approach is *implicit importance sampling*, as introduced below.

Consider the task of drawing particles  $x^i$  for  $i = 1, \dots, N$  from high-probability regions of  $p(x | y_{1:T})$ . To this end, we can introduce a reference random vector  $\xi$  with a known PDF  $p(\xi)$  to create a map  $\xi \rightarrow x^i$  that aligns the target  $p(x^i | y_{1:T})$  with reference  $p(\xi)$  density. This notion delivers a two-fold benefit. First, the map will allow to connect high-probability samples of  $\xi$  to highly probable  $x^i$ . Second, one can choose  $p(\xi)$  such that it is easy to sample, thus reducing the difficulty of sampling  $x^i$ . Proceeding forward, write  $X$  as a shorthand for  $x^i$ , and define  $F(X) = -\log(p(X | y_{1:T}))$  and  $s(\xi) = -\log(p(\xi))$ . The highest probabilities of  $p(X | y_{1:T})$  and  $p(\xi)$  will appear at points where  $\min F(X)$  and  $\min s(\xi)$  are achieved, respectively. Then, to align  $p(X | y_{1:T})$  with  $p(\xi)$  around each other's highest probability point, we can let

$$F(X) - \min F(X) = s(\xi) - \min s(\xi), \quad (2)$$

which implies a map  $X(\xi)$ . With  $X(\xi)$ , one can draw a particle from the high-probability regions of  $\xi$  and map it to obtain a particle of  $X$ , which is ensured to lie in the high-probability regions of  $p(X | y_{1:T})$ . Here, a trick to pick a high-probable particle for  $\xi$  is by using an auxiliary probability distribution that has the same support but concentrates on the high-probability regions of  $p(\xi)$ .

Next is deciding the sampling weight for  $X$ . Based on the above, the importance distribution  $\pi(\cdot)$  is the PDF of  $X(\xi)$  as a mapping of  $\xi$  given  $p(\xi)$ . Assume that  $X(\xi)$  is invertible for simplicity of exposition (the result

is readily generalizable to the non-invertible case based on [10, Corollary 11.3]). Then, by [10, Corollary 11.2], it follows that

$$\pi(X(\xi)) = |J|^{-1} \cdot p(\xi),$$

where  $\pi(X)$  denotes the importance distribution of  $X$ ,  $J = dX(\xi)/d\xi$ , and  $|J|$  is the absolute value of  $J$ 's determinant. The weight of  $X$  using (2) hence is

$$\begin{aligned} \tilde{w} &= \frac{p(X | y_{1:T})}{\pi(X(\xi))} = \frac{\exp(-F(X))}{|J|^{-1} \cdot p(\xi)} \\ &= |J| \cdot \exp[-\min F(X) + \min s(\xi)]. \end{aligned} \quad (3)$$

For  $x^i$ , the normalized weight is then given by

$$w^i = \frac{\tilde{w}^i}{\sum_{j=1}^N \tilde{w}^j} = \frac{|J^i|}{\sum_{j=1}^N |J^j|}. \quad (4)$$

As a result, (1) is approximated as

$$\mathbb{E}[g(x) | y_{1:T}] \approx \sum_{i=1}^N w^i g(x^i). \quad (5)$$

In above, (2)-(5) comprise the method of implicit importance sampling. More details are offered in [11,12]. While this method promises to use fewer particles, it is computationally challenging to find  $\min F(X)$  and then solve (2), due to the often nonlinear non-convex  $F$ . To find an alternative way forward, we examine a Gaussian special case here to gain an insight into tackling this issue.

---

**Gaussian Case Study:** Consider the following Gaussian approximation:

$$p\left(\begin{bmatrix} x \\ y_{1:T} \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{bmatrix} \bar{m} \\ \bar{y}_{1:T} \end{bmatrix}, \begin{bmatrix} P^x & P^{xy} \\ (P^{xy})^\top & P^y \end{bmatrix}\right).$$

It then follows that  $p(x | y_{1:T}) = \mathcal{N}(\tilde{m}, \tilde{P}^x)$ , where

$$\begin{aligned} \tilde{m} &= \bar{m} + P^{xy} (P^y)^{-1} (y_{1:T} - \bar{y}_{1:T}), \\ \tilde{P}^x &= P^x - P^{xy} (P^y)^{-1} (P^{xy})^\top. \end{aligned}$$

Suppose  $p(\xi) \sim \mathcal{N}(0, I)$  without loss of generality. The map  $X(\xi)$  derived from (2) is

$$X(\xi) = \tilde{m} + \sqrt{\tilde{P}^x} \xi.$$

After a high-probability particle  $\xi^i$  is drawn from  $p(\xi)$ , it can be mapped by  $X(\xi)$  to compute the particle  $x^i$ .

Its weight can be determined using (3)-(4). Specifically,

$$x^i = \tilde{m} + \sqrt{\tilde{P}^x} \xi^i, \quad w^i = \frac{1}{N}.$$

Based on (5), we have

$$\mathbb{E}[g(x) | y_{1:T}] \approx \frac{1}{N} \sum_{i=1}^N g(x^i).$$

The above case study indicates that implicit importance sampling is efficiently executable in a Gaussian setting. Extending this insight, we identify that, under the Gaussian approximation, the IPF can be approximately implemented as a bank of KFs.

### 3 Approximation of IPF as a Bank of KFs

Consider the problem of estimating the state of a nonlinear system taking the form

$$\begin{cases} x_{k+1} = f(x_k) + w_k, \\ y_k = h(x_k) + v_k, \end{cases} \quad (6)$$

where  $x_k \in \mathbb{R}^{n_x}$  is the state,  $y_k \in \mathbb{R}^{n_y}$  is the measurement, and  $w_k$  and  $v_k$  are zero-mean white noises with covariances  $Q$  and  $R$ , respectively. For the purpose of state estimation, it is of interest to consider the conditional PDF  $p(x_{0:k} | y_{1:k})$ . By the Markovian property of (6),  $p(x_{0:k} | y_{1:k})$  satisfies the following recursive relation:

$$p(x_{0:k} | y_{1:k}) \propto p(y_k | x_k) p(x_k | x_{k-1}) p(x_{0:k-1} | y_{1:k-1}).$$

One can assume that  $p(x_{0:k-1} | y_{1:k-1})$  has been made available at time  $k-1$ , as a result of the preceding recursive updates, and that it can be weakly approximated by the empirical distribution of particles  $x_{0:k-1}^i$  for  $i = 1, \dots, N$  with weights  $w_{k-1}^i$ . Therefore, to draw new particles  $x_k^i$  at time  $k$ , we only need to consider

$$p(x_k^i | y_{1:k}) \propto p(y_k | x_k^i) p(x_k^i | x_{k-1}^i). \quad (7)$$

Using  $X_k$  as a shorthand for  $x_k^i$ , we define

$$F_i(X_k) = -\log(p(y_k | X_k) p(X_k | x_{k-1}^i)),$$

To pick  $X_k$  in high-probability regions, we choose a reference random vector  $\xi_k$  with  $p(\xi_k) = \exp(-s(\xi_k))$ , and following (2), let

$$F_i(X_k) - \min F_i(X_k) = s(\xi_k) - \min s(\xi_k). \quad (8)$$

Taking a high-probability sample  $\xi_k^i$  from  $p(\xi_k)$  and solving (8), we can obtain a desired particle  $x_k^i$ , with weight

$$\begin{aligned} \tilde{w}_k^i &= w_{k-1}^i \frac{p(y_k | X_k) p(X_k | x_{k-1}^i)}{\pi(X_k(\xi_k))} \\ &= w_{k-1}^i |J_k^i| \cdot \exp[-\min F_i(X_k) + \min s(\xi_k)], \end{aligned} \quad (9)$$

where  $J_k^i = dX_k(\xi_k^i)/d\xi_k^i$ . The normalized weight then is

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^N \tilde{w}_k^j}. \quad (10)$$

Here, (8)-(10) illustrate the particle computation and weighting underlying the IPF. However, a difficulty for the implementation lies in finding  $\min F_i(X_k)$  and solving the algebraic equation (8). Even though the literature presents several useful methods based on iterative computation [7], their computational complexity is still not-trivial. To address the challenge, we take inspiration from the Gaussian Case Study in Section 2 to solve (8).

First, let us introduce a Gaussian approximation:

$$p\left(\begin{bmatrix} x_k^i \\ y_k \end{bmatrix} \middle| x_{k-1}^i\right) \sim \mathcal{N}\left(\begin{bmatrix} \tilde{m}_k^i \\ \tilde{y}_k^i \end{bmatrix}, \begin{bmatrix} \tilde{P}_k^i & \tilde{P}_k^{xy,i} \\ (\tilde{P}_k^{xy,i})^\top & \tilde{P}_k^{yy,i} \end{bmatrix}\right). \quad (11)$$

By (11), we have

$$p(x_k^i | y_k, x_{k-1}^i) \sim \mathcal{N}(\tilde{m}_k^i, \tilde{P}_k^i), \quad (12)$$

where

$$\tilde{m}_k^i = \tilde{m}_k^i + \tilde{P}_k^{xy,i} (\tilde{P}_k^{yy,i})^{-1} (y_k - \tilde{y}_k^i), \quad (13a)$$

$$\tilde{P}_k^i = \tilde{P}_k^i - \tilde{P}_k^{xy,i} (\tilde{P}_k^{yy,i})^{-1} (\tilde{P}_k^{xy,i})^\top. \quad (13b)$$

Assuming  $p(y_k | x_k^i) = p(y_k | x_k^i, x_{k-1}^i)$  and inserting it into (7), we obtain

$$p(x_k^i | y_{1:k}) \propto p(x_k^i | y_k, x_{k-1}^i) p(y_k | x_{k-1}^i). \quad (14)$$

Given (14), we use (8) to align  $p(x_k^i | y_{1:k})$  with the high-probability regions of a standard Gaussian random vector  $\xi_k$ , with  $p(\xi_k) \sim \mathcal{N}(0, I)$ . Then, referring to the Gaussian Case Study in Section 2, the desired particle  $x_k^i$  can be expressed as

$$x_k^i = \tilde{m}_k^i + \sqrt{\tilde{P}_k^i} \xi_k^i, \quad (15)$$

---

**Algorithm 1: The KF-IPF Algorithm**


---

- Initialize a set of particles  $x_0^i$  for  $i = 1, \dots, N$  with  $w_0^i$  and  $P_0^i$  at  $k = 0$
  - Implement the following for  $x_k^i$  for  $i = 1, \dots, N$  from  $k = 1, \dots, T$ 
    - Compute  $\bar{m}_k^i$  and  $\bar{P}_k^i$  using  $x_{k-1}^i$  and  $P_{k-1}^i$  via KF prediction
    - Compute  $\bar{y}_k^i$ ,  $\bar{P}_k^{y,i}$  and  $\bar{P}_k^{xy,i}$  as in KF prediction
    - Compute  $\tilde{m}_k^i$  and  $\tilde{P}_k^i$  via KF update in (13a)-(13b)
    - Compute  $x_k^i$  and  $w_k^i$  via (15)-(17)
    - Do resampling if necessary
    - Output the state estimate
    - $k \leftarrow k + 1$
- 

where  $\xi_k^i$  is a high-probability sample of  $p(\xi_k)$ . Based on (9)-(10), its weight is computed as

$$w_k^i = \frac{w_{k-1}^i p(y_k | x_{k-1}^i)}{\sum_{j=1}^N w_{k-1}^j p(y_k | x_{k-1}^j)}. \quad (16)$$

In practice, it is important that  $\xi_k^i$  are drawn from the high probability regions of  $p(\xi_k)$ . An implementation trick to ensure this is to sample from  $\mathcal{N}(0, \alpha I)$  with  $0 < \alpha \ll 1$ . We emphasize that  $\alpha$  can be considered as a tunable parameter to dictate the desired high probability region of  $p(\xi_k) \sim \mathcal{N}(0, I)$ . It hence must take a small enough value to ensure that only highly probable particles with respect to  $\mathcal{N}(0, I)$  are drawn. This is especially true when only a small number of particles are used. The covariance associated to  $x_k^i$  is given as

$$P_k^i = \tilde{P}_k^i. \quad (17)$$

Note that the formulation in (11)-(13) is equivalent to the well-known KF update procedure. This connection suggests the viability of approximately implementing the IPF as a bank of parallel KFs applied to individual particles. Specifically, given  $x_{k-1}^i$  at time  $k - 1$ , one can compute  $\bar{m}_k^i$  and  $\bar{P}_k^i$  via KF prediction and then compute  $\tilde{m}_k^i$  and  $\tilde{P}_k^i$  via KF update [13]; going further,  $\tilde{m}_k^i$  combines with  $\xi_k^i$  to generate  $x_k^i$ . Different nonlinear KFs can be leveraged to enable the prediction-update of the particles. Here, we highlight the use of the EKF and UKF to approximate the mean and covariance statistics of the Gaussian approximation in (11). The EKF and UKF have found wide use with proven effectiveness for various systems, and their detailed equations can be found in [13]. As extensions of the standard linear KF, both EKF and UKF at their core aim to track the conditional statistics (mean and covariance) of a nonlinear system's state recursively using accumulating measurement data. To achieve this end, EKF linearizes the system's nonlinear functions, and UKF leverages the unscented transform.

Summarizing the above, Algorithm 1 outlines the KF-IPF algorithm framework. Based on this framework, approximate realizations based on the EKF or UKF can be readily derived and named as E-IPF and U-IPF, respectively. The details are omitted here for the sake of space.

The proposed KF-IPF algorithm provides an amenable way to implement the IPF by approximating it as a bank of KFs. The algorithm circumvents the need to numerically find  $\min F_i(X_k)$  and solve the nonlinear algebraic equation (8) with computationally intensive numerical schemes in existing IPF literature. By design, the KF-IPF can also exploit the advantages of a nonlinear KF. For instance, it can achieve derivative-free computation if the UKF is used. These features will enable it to facilitate the computation of the IPF considerably. Further, the algorithm inherits the capability of the IPF in mitigating the issue of particle degeneracy. Also, note that the development of the KF-IPF algorithm uses the Gaussian approximation in (11), which plays a utilitarian role in solving (8) via the KF update. In effect, the approximation serves to facilitate computation. Despite it, the KF-IPF algorithm, as an IPF method, is still well applicable to non-Gaussian estimation problems. The simulation study in Section 4 further highlights this.

The literature also includes PF methods based on banks of EKF and UKF [14,15], referred to as EPF or UPF, respectively. However, they are different from the proposed KF-IPF algorithm on several aspects. First, the EPF/UPF is designed to use a bank of EKF/UKF to approximate PF's optimal importance distribution,  $p(x_k | x_{k-1}, y_k)$ . However, the KF-IPF is based on the notion of *implicit importance sampling*, which looks for high-probability regions of the target distribution and then locates particles in these regions. Second, in terms of implementation, in the sampling step (15), the EPF/UPF draws particles from the Gaussian approximation made by the EKF/UKF, which is analogous to the case where the KF-IPF directly draws  $\xi_k^i$  from  $\mathcal{N}(0, I)$ . However, a significant risk, in this case, is that  $x_k^i$  will fall in the low-probability regions when a marginally probable  $\xi_k^i$  is taken—this is also likely why the EPF and UPF sometimes fail to perform adequately, as noticed in [1]. By contrast, the actual implementation of the KF-IPF requires focusing particles only on the high-probability regions. For instance, to ensure this, we suggested drawing particles  $\xi_k^i$  from  $\mathcal{N}(0, \alpha I)$  with  $0 < \alpha \ll 1$  so that the resultant  $x_k^i$  is highly probable. Finally, in the particle weighting step (16), the KF-IPF utilizes  $p(y_k | x_{k-1}^i)$  instead of  $p(y_k | x_k^i)$  as in the case of the EPF/UPF. To sum up, the KF-IPF is distinct from the EPF/UPF, while contributing a new insight into the broad notion of using KF banks to implement the PF.

## 4 Numerical Simulation

In this section, we assess the performance of the KF-IPF algorithm on the 40-dimensional Lorenz'96 model [16]:

$$\dot{x}_j = f(x, F) = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F, \quad (18)$$

where  $j = 1, \dots, n_x = 40$  is the dimension index,  $x_{-1} = x_{n_x-1}$ ,  $x_0 = x_{n_x}$ ,  $x_{n_x+1} = x_1$ ,  $F = 5$ . The simulation run of (18) is based on discretization using the explicit fourth-order Runge-Kutta method. The discret-time system with process noise  $w$  is expressed as:

$$x_{j,k+1} = x_{j,k} + \frac{\Delta t}{6} (h_1 + 2h_2 + 2h_3 + h_4) + w_{j,k},$$

where

$$\begin{aligned} h_1 &= f(x_{j,k}, F), \quad h_2 = f(x_{j,k} + \Delta t \cdot h_1/2, F), \\ h_3 &= f(x_{j,k} + \Delta t \cdot h_2/2, F), \quad h_4 = f(x_{j,k} + \Delta t \cdot h_3, F), \end{aligned}$$

$w_{j,k} \sim \mathcal{U}(-0.5, 0.5)$  is added as the i.i.d. process noise, and  $\Delta t = 0.01$  is the discretization time-step. In addition, we consider a nonlinear measurement function that partially observes every other component of the state:

$$y_{l,k} = x_{2l-1,k} + \sin(x_{2l-1,k}) + v_{l,k},$$

where  $l = 1, \dots, n_y = 20$ , and  $v_l \sim \mathcal{U}(-0.5, 0.5)$ . We evaluate the state estimation performance of the E-IPF and U-IPF for the above system by the root mean squared error (RMSE) metric over 50 Monte Carlo (MC) runs. Each run is initialized randomly and with a bias to the true state. Meanwhile, we run the IPF based on numerical iteration (I-IPF) in [17], EPF, and UPF for comparison. The PDF of the reference random variable  $p(\xi_k)$  for all the considered IPFs is set to  $\mathcal{N}(0, \alpha I)$ , where  $\alpha = 0.05$ . Further, the individual Kalman filters use the covariances of  $w$  and  $v$  as  $Q$  and  $R$ , respectively, in their estimation run.

Fig. 1 shows the estimation performance of the different filters using different numbers of particles. The observations are summarized as follows. First, the KF-IPF algorithm provides advantageous estimation. Specifically, the E-IPF performs better than the EPF in terms of accuracy, with both using 1,000 particles; further, the U-IPF with only ten particles delivers considerably higher accuracy than the UPF with 100 particles. Second, compared to the I-IPF, the KF-IPF represents a more favorable and promising approach. Even though the I-IPF is understandably more accurate than the E-IPF (since it is based on iterative linearization and solution rather than the first-order linearization involved in the E-IPF), the U-IPF consistently outperforms the I-IPF, and this comes at using just ten particles rather than 100 by the I-IPF. The observations also suggest the

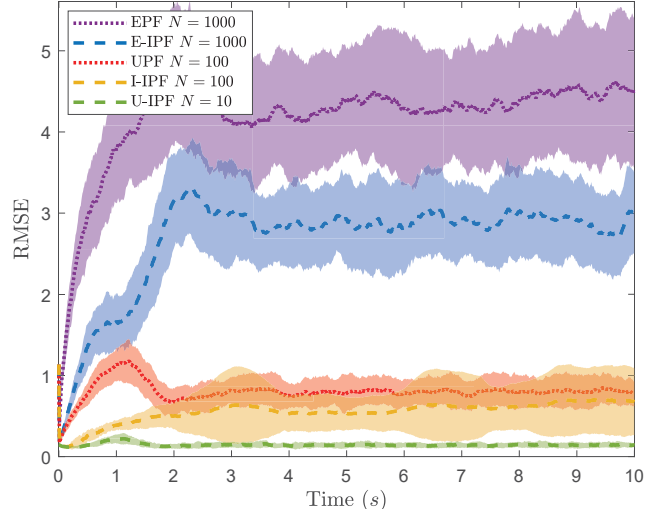


Fig. 1. RMSE performance of the EPF/E-IPF (with 1,000 particles) and U-IPF (with only 10 particles) versus UPF/I-IPF (with 100 particles). The shaded regions correspond to the  $\pm\sigma$  bounds by the filters over 50 Monte Carlo runs.

Table 1  
Average Computation Time

	EPF	E-IPF	UPF	I-IPF	U-IPF
$N$	1,000	1,000	100	100	10
Time (s)	3,462	3,362	652	578.9	46.4

need to use a good KF to well execute the KF-IPF algorithm. The simulation was run using MATLAB R2020b on a workstation equipped with a 3.5GHz Intel Core i9-10920X CPU, 128GB of RAM. Table 1 summarizes the average computation time of a MC run for the considered filters. Overall, the computation time increases with the number of particles. The U-IPF and I-IPF are computationally the fastest among the filters. The simulation results indicate the advantages and promise of the KF-IPF algorithm for nonlinear non-Gaussian estimation.

## 5 Conclusion

The IPF has emerged as a vibrant tool for dealing with particle degeneracy confronting the PF. However, the present IPF methods are computationally cumbersome, which may limit their application. In this paper, we proposed approximately implementing the IPF framework *via* nonlinear KFs for higher computational tractability. We presented implicit importance sampling and then, on this basis, derived the IPF realization based on a bank of nonlinear KFs. We discussed the EKF and UKF realizations of the IPF. Further, we examined how the proposed realization relates to some existing PF methods. Finally, we demonstrated the effectiveness of KF-based IPF through the use of a simulation.



## Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments. I. Askari and H. Fang were partially supported by the U.S. National Science Foundation under Awards CMMI-1763093 and CMMI-1847651. X. Tu was partially supported by the U.S. National Science Foundation under Award DMS-1723066.

## References

- [1] S. Sarkka, Bayesian Filtering and Smoothing, Cambridge University Press, 2013.
- [2] A. J. Chorin, X. Tu, Implicit sampling for particle filters, *Proceedings of the National Academy of Sciences* 106 (41) (2009) 17249–17254.
- [3] T. Yang, P. G. Mehta, S. P. Meyn, Feedback particle filter, *IEEE Transactions on Automatic Control* 58 (10) (2013) 2465–2480.
- [4] D. Raihan, S. Chakravorty, Particle Gaussian mixture filters-I, *Automatica* 98 (2018) 331–340.
- [5] P. M. Stano, Z. Lendek, R. Babuška, Saturated particle filter: Almost sure convergence and improved resampling, *Automatica* 49 (1) (2013) 147–159.
- [6] A. J. Chorin, M. Morzfeld, X. Tu, Implicit particle filters for data assimilation, *Communications in Applied Mathematics and Computational Science* 5 (2) (2010) 221–240.
- [7] A. J. Chorin, M. Morzfeld, X. Tu, A survey of implicit particle filters for data assimilation, in: *State-Space Models: Applications in Economics and Finance*, Springer, New York, NY, 2013, pp. 63–88.
- [8] C. Su, X. Tu, Sequential implicit sampling methods for Bayesian inverse problems, *SIAM/ASA Journal on Uncertainty Quantification* 5 (1) (2017) 519–539.
- [9] B. Weir, R. N. Miller, Y. H. Spitz, A potential implicit particle method for high-dimensional systems, *Nonlinear Processes in Geophysics* 20 (6) (2013) 1047–1060.
- [10] J. Jacod, P. Protter, *Probability Essentials*, Springer-Verlag Berlin Heidelberg, 2004.
- [11] E. Atkins, M. Morzfeld, A. J. Chorin, Implicit particle methods and their connection with variational data assimilation, *Monthly Weather Review* 141 (6) (2013) 1786–1803.
- [12] M. Morzfeld, X. Tu, J. Wilkening, A. Chorin, Parameter estimation by implicit sampling, *Communications in Applied Mathematics and Computational Science* 10 (2) (2015) 205–225.
- [13] H. Fang, N. Tian, Y. Wang, M. Zhou, M. A. Haile, Nonlinear Bayesian estimation: from Kalman filtering to a broader horizon, *IEEE/CAA Journal of Automatica Sinica* 5 (2) (2018) 401–417.
- [14] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, *Statistics and Computing* 10 (2000) 197–208.
- [15] R. van der Merwe, A. Doucet, N. de Freitas, E. Wan, The unscented particle filter, in: *Advances in Neural Information Processing Systems* 13, 2001, pp. 584–590.
- [16] E. N. Lorenz, Predictability: a problem partly solved, In *Proceedings of the seminar on Predictability* 1 (1995) 1–18.
- [17] A. J. Chorin, X. Tu, An iterative implementation of the implicit nonlinear filter, *ESAIM: Mathematical Modelling and Numerical Analysis* 46 (3) (2012) 535–543.