Hindawi
Journal of Electrical and Computer Engineering
Volume 2020, Article ID 9541624, 9 pages
https://doi.org/10.1155/2020/9541624

Research Article

Differential Evolution Optimized a Second-Order Divided Difference Particle Filter

Ting Cao,[1] **Huo-tao Gao**,[1] **Chun-feng Sun**,[2] **and Guo-bao Ru**[1]

[1]*Electromagnetic Wave and Modern Radar Lab, School of Electronic Information, Wuhan University, Wuhan 430079, China*
[2]*School of Physics and Electronic Information Engineering, Hubei Engineering University, Xiaogan 432000, China*

Correspondence should be addressed to Huo-tao Gao; gaoght863@163.com

Received 28 June 2019; Revised 31 August 2019; Accepted 30 January 2020; Published 24 February 2020

Academic Editor: Yagang Zhang

In order to improve the estimation accuracy of particle filter algorithm in a nonlinear system state estimation problem, a new algorithm based on the second-order divided difference filter to generate the proposed distribution and the differential evolution algorithm for resampling is proposed. The second-order divided difference based on Strling's interpolation formula is used to generate approximations to nonlinear dynamics, which avoids the evaluation of the Jacobian derivative matrix and is easy to implement. Cholesky factorization is used to ensure the positive definiteness of the covariance matrix. The truncated errors of the local linearization are reduced to a certain extent, and the approximation degree of the proposed distribution to the posterior probability of the system state is improved. The differential evolution algorithm is used to replace the traditional resampling algorithm, which effectively mitigates the problem of particle degradation. Monte Carlo simulation experiments show the effectiveness of the new algorithm.

1. Introduction

Particle filters (PF) are an effective solution to a nonlinear dynamic and/or measurement filtering problem, which has been widely used in many fields such as economics [1, 2], visual tracking [3, 4], navigation position [5], and radio communication [6]. The PF is a recursive Bayesian estimation method based on the Monte Carlo simulation. It exploits a set of random particles (samples) and their associated weights to represent the posterior probability density function. Despite achieving the state-of-the-art performance, existing PF approaches are restricted to two aspects: the sample degeneracy and impoverishment. The sample degeneracy appears when the weight focuses on a few particles only and the rest have negligible weight, and impoverishment is due to the loss of diversity among the particles in resampling process when high-weight particles are chosen many times. In general, there are three methods to improve the performance of the PF. The first is to increase the particle number. This is often impractical because increasing the number of particles yields very limited improved estimation accuracy, but requires significantly increased computational burden. Therefore, we rely on two other methods: (1) select an effective proposal distribution and (2) resampling.

To obtain better proposal distribution, many nonlinear filtering techniques have been put forward. For instance, by linearizing nonlinear functions around the current state at first-order Taylor approximations, extended Kalman filter (EKF) is used to generate the proposal distribution, and then the extended particle filter (EPF) is proposed. However, the EKF has slow convergence, low estimated accuracy and may suffer from large errors when the system has strong nonlinearity. The unscented particle filter (UPF) [7] utilizes the unscented Kalman filter (UKF) to generate the proposal distribution. By translating the analytic integral operator into the approximate summation operator through a set of deterministic points, the UKF can accurately calculate the mean and covariance of nonlinear functions up to the second order (third in Gaussian prior) of the Taylor series expansion. Therefore, the UPF outperforms EPF in accuracy. However, as the system dimension increases, the computation burden

increases rapidly, this limits the real-time performance of UPF. To address the computation load, several simplex point selection strategies have been developed. In 2003, Julier proposed spherical simplex unscented transformation (SSUT) [8]. In [9], the spherical simplex unscented particle filter resulted from using a spherical simplex unscented Kalman filter (SSUKF) for the proposal distribution generation within a particle filter framework. Zhao [10] proposed a global sampling strategy which generated proposal distribution from the mean and covariance value of the whole particles instead of from each particle. Aiming at the degeneracy problem of particle filter algorithm, many researchers proposed some improved algorithms, such as the gradient particle filtering [11] and the box particle filtering [12].

The second-order divided difference (DD2) filter [13] is based on a similar linear regression to the UKF, but is able to more accurately estimate the state covariance resulting in a more precise set of weighted sample points to estimate from. Besides, the DD2 filter provides a faster processing speed than the UKF because it does not need to predict forward every positive and negative sigma point in separate stages from when the measurement prediction, measurement prediction covariance, and cross-covariance are interpreted.

Resampling is a key operation in particle filtering. Some improved techniques based on intelligent optimization approaches were therefore proposed to modify the resampling scheme, which includes Genetic Particle Filter [14], Ant Colony Particle Filter [15] and Particle Swarm Particle Filter [16]. Recently, scholars are eager to the research of Artificial Neural Network (ANN) and their applications [17–22]. The application of BP or Convolution Network/PF has indicated an attractive research direction in combining particle filtering and ANN optimization.

In this paper, we propose using the DD2 filter to generate the proposal distribution. Meanwhile, aiming at the problem of Particle Filter algorithm such as particles degeneracy and particles impoverishment, a differential evolution algorithm is used to make the particles of DD2 filter move toward the global optimum, which optimizes the resampling process and relieves the problem of particles degeneracy and impoverishment.

The rest of the paper is organized as follows. In Section 2, we briefly reviewed the general second-order divided difference filter. Section 3 gives details of the proposed particle filter. Experiments and analysis are presented in Section 4. Finally, the work is concluded in Section 5.

2. The Second-Order Divided Difference Filter

In this section, a generalized version of the DD2 filter will be outlined. DD2 filter exploits polynomial approximations obtained with a Stirling interpolation formula for the derivation of state estimators for nonlinear systems. Conceptually, the principle underlying the DD2 filters is much like that of the EKF and its higher order relatives. The estimator becomes more precise than estimators based on Taylor approximations; yet the implementation is significantly simpler as no derivatives are required. Additionally, by using the Cholesky factors of covariance matrices, the numerical properties of estimation algorithms are generally improved.

Many nonlinear filtering problems can be described in the form of the dynamic state space (DSS) model as follows:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}\right) \quad \text{(Process equation)}, \tag{1}$$

$$\mathbf{y}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{v}_k\right) \quad \text{(Measurement equation)}, \tag{2}$$

where $\mathbf{x}_k$ and $\mathbf{y}_k$ are the state vector and observation at time instant k. Process system noise $\mathbf{w}_{k-1}$ and the measurement noise $\mathbf{v}_k$ are random vectors that derive the dynamic system through the nonlinear state transition function $\mathbf{f}(\cdot)$ and the nonlinear measurement function $\mathbf{h}(\cdot)$, respectively.

We consider expanding the nonlinear function $\mathbf{y} = \mathbf{f}(\mathbf{x})$, by polynomial approximations based on String's interpolation formula. The second-order expansion around the point $\mathbf{x} = \overline{\mathbf{x}}$ gives the following approximation:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\overline{\mathbf{x}}) + \mathbf{f}'_{\mathbf{DD}}(\overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}}) + \frac{\mathbf{f}''_{\mathbf{DD}}(\overline{\mathbf{x}})}{2!}(\mathbf{x} - \overline{\mathbf{x}})^2, \tag{3}$$

where

$$\begin{aligned} \mathbf{f}'_{\mathbf{DD}}(\overline{\mathbf{x}}) &= \frac{\mathbf{f}(\overline{\mathbf{x}} + \lambda) - \mathbf{f}(\overline{\mathbf{x}} - \lambda)}{2\lambda}, \\ \mathbf{f}''_{\mathbf{DD}}(\overline{\mathbf{x}}) &= \frac{\mathbf{f}(\overline{\mathbf{x}} + \lambda) + \mathbf{f}(\overline{\mathbf{x}} - \lambda) - 2\mathbf{f}(\overline{\mathbf{x}})}{\lambda^2}, \end{aligned} \tag{4}$$

where λ is the interval step-size for approximation. Inserting the full Taylor series in place of $\mathbf{f}(\overline{\mathbf{x}} + \lambda)$ and $(\overline{\mathbf{x}} - \lambda)$, we can obtain

$$\begin{aligned} &\mathbf{f}(\overline{\mathbf{x}}) + \mathbf{f}'_{\mathbf{DD}}(\overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}}) + \frac{\mathbf{f}''_{\mathbf{DD}}(\overline{\mathbf{x}})}{2!}(\mathbf{x} - \overline{\mathbf{x}})^2 \\ &= \mathbf{f}(\overline{\mathbf{x}}) + \mathbf{f}'(\overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}}) + \frac{\mathbf{f}''(\overline{\mathbf{x}})}{2!}(\mathbf{x} - \overline{\mathbf{x}})^2 \\ &\quad + \left(\frac{\mathbf{f}^{(3)}(\overline{\mathbf{x}})}{3!}\lambda^2 + \frac{\mathbf{f}^{(5)}(\overline{\mathbf{x}})}{5!}\lambda^4 + \cdots\right)(\mathbf{x} - \overline{\mathbf{x}}) \\ &\quad + \left(\frac{\mathbf{f}^{(4)}(\overline{\mathbf{x}})}{4!}\lambda^2 + \frac{\mathbf{f}^{(6)}(\overline{\mathbf{x}})}{6!}\lambda^4 + \cdots\right)(\mathbf{x} - \overline{\mathbf{x}})^2. \end{aligned} \tag{5}$$

The first three terms on the right hand side of equation (5) are identical with the second order Taylor series expansion and are independent of the interval length λ. The "remainder" term corresponds to the higher order terms of the Taylor series expansion of $\mathbf{f}$ and is controlled by λ. Proper interval lengths can make sure that the remainder term in some sense will be close to the higher order terms of the full Taylor series. As we can see, it is the remainder term that makes the interpolation formula more accurate than Taylor approximation in some applications.

The following procedure outlines the implementation of the DD2 filters.

(1) Square Cholesky factorization:

$$\begin{aligned}\widehat{\mathbf{P}}_{k-1} &= \widehat{\mathbf{S}}_{\mathbf{x}}\widehat{\mathbf{S}}_{\mathbf{x}}^{T},\\ \overline{\mathbf{P}}_{k-1} &= \overline{\mathbf{S}}_{\mathbf{x}}\overline{\mathbf{S}}_{\mathbf{x}}^{T},\end{aligned} \tag{6}$$

$$\begin{aligned}\mathbf{Q} &= \mathbf{S}_{\mathbf{w}}\mathbf{S}_{\mathbf{w}}^{T},\\ \mathbf{R} &= \mathbf{S}_{\mathbf{v}}\mathbf{S}_{\mathbf{v}}^{T},\end{aligned} \tag{7}$$

where $\widehat{\mathbf{S}}_{\mathbf{x}}$, $\overline{\mathbf{S}}_{\mathbf{x}}$, $\mathbf{S}_{\mathbf{w}}$, and $\mathbf{S}_{\mathbf{v}}$ are the Cholesky factorization of the predicted state error covariance $\widehat{\mathbf{P}}_{k-1}$, corrected state error $\overline{\mathbf{P}}_{k-1}$, process noise covariance $\mathbf{Q}$ and measurement noise covariance $\mathbf{R}$, respectively.

$$\mathbf{S}^{(1)}_{\mathbf{x}\widehat{\mathbf{x}}_{k-1}} = \frac{1}{2\lambda}\left\{\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1}+\lambda\widehat{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{w}}_{k-1}\right)-\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1}-\lambda\widehat{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{w}}_{k-1}\right)\right\}, \tag{8}$$

$$\mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}_{k-1}} = \frac{1}{2\lambda}\left\{\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}+\lambda\mathbf{S}_{\mathbf{w},j}\right)-\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}-\lambda\mathbf{S}_{\mathbf{w},j}\right)\right\}, \tag{9}$$

$$\begin{aligned}\mathbf{S}^{(2)}_{\mathbf{x}\widehat{\mathbf{x}}_{k-1}} = \frac{\sqrt{\lambda^2-1}}{2\lambda^2}&\left(\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1}+\lambda\widehat{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{w}}_{k-1}\right)\right.\\ &\left.+\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1}-\lambda\widehat{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{w}}_{k-1}\right)-2\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}\right)\right),\end{aligned} \tag{10}$$

$$\begin{aligned}\mathbf{S}^{(2)}_{\mathbf{x}\mathbf{w}_{k-1}} = \frac{\sqrt{\lambda^2-1}}{2\lambda^2}&\left(\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}+\lambda\mathbf{S}_{\mathbf{w},j}\right)\right.\\ &\left.+\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}-\lambda\mathbf{S}_{\mathbf{w},j}\right)-2\mathbf{f}_i\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}\right)\right),\end{aligned} \tag{11}$$

where $\widehat{\mathbf{S}}_{\mathbf{x},j}$ denotes the jth column of $\widehat{\mathbf{S}}_{\mathbf{x}}$ and similarly for the other factors. The Cholesky factor of the predicted covariance $\overline{\mathbf{S}}_{\mathbf{x}_k}$ is obtained by the House-holder transformation of the compound matrix.

$$\overline{\mathbf{S}}_{\mathbf{x}_k} = \left[\begin{array}{cccc}\mathbf{S}^{(1)}_{\mathbf{x}\widehat{\mathbf{x}}_{k-1}} & \mathbf{S}^{(1)}_{\mathbf{x}\mathbf{w}_{k-1}} & \mathbf{S}^{(2)}_{\mathbf{x}\widehat{\mathbf{x}}_{k-1}} & \mathbf{S}^{(2)}_{\mathbf{x}\mathbf{w}_{k-1}}\end{array}\right]. \tag{12}$$

(2) The predicted state $\overline{\mathbf{x}}_k$:

$$\begin{aligned}\overline{\mathbf{x}}_k = &\frac{\lambda^2-n_x-n_w}{\lambda^2}\mathbf{f}\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}\right)\\ &+\frac{1}{2\lambda^2}\sum_{p=1}^{n_x}\left(\mathbf{f}\left(\widehat{\mathbf{x}}_{k-1}+\lambda\widehat{\mathbf{S}}_{\mathbf{x},p},\overline{\mathbf{w}}_k\right)+\mathbf{f}\left(\widehat{\mathbf{x}}_{k-1}-\lambda\widehat{\mathbf{S}}_{\mathbf{x},p},\overline{\mathbf{w}}_{k-1}\right)\right)\\ &+\frac{1}{2\lambda^2}\sum_{p=1}^{n_w}\left(\mathbf{f}\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}+\lambda\mathbf{S}_{\mathbf{w},p}\right)+\mathbf{f}\left(\widehat{\mathbf{x}}_{k-1},\overline{\mathbf{w}}_{k-1}-\lambda\mathbf{S}_{\mathbf{w},p}\right)\right),\end{aligned} \tag{13}$$

where n_x is the size of the state dimension and n_w is the size of the process noise dimension.

$$\mathbf{S}^{(1)}_{\mathbf{y}\overline{\mathbf{x}}_k} = \frac{1}{2\lambda}\left(\mathbf{h}_i\left(\overline{\mathbf{x}}_k+\lambda\overline{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{v}}_k\right)-\mathbf{h}_i\left(\overline{\mathbf{x}}_k-\lambda\overline{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{v}}_k\right)\right), \tag{14}$$

$$\mathbf{S}^{(1)}_{\mathbf{y}\mathbf{v}_k} = \frac{1}{2\lambda}\left(\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{v}_k+\lambda\mathbf{S}_{\mathbf{v},j}\right)-\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{v}_k-\lambda\mathbf{S}_{\mathbf{v},j}\right)\right), \tag{15}$$

$$\begin{aligned}\mathbf{S}^{(2)}_{\mathbf{y}\overline{\mathbf{x}}_k} = \frac{\sqrt{\lambda^2-1}}{2\lambda^2}&\left\{\mathbf{h}_i\left(\overline{\mathbf{x}}_k+\lambda\overline{\mathbf{S}}_{\mathbf{x},j},\overline{\mathbf{v}}_k\right)\right.\\ &\left.+\mathbf{h}_i\left(\overline{\mathbf{x}}_k-\lambda\overline{\mathbf{S}}_{\mathbf{x},j},\overline{v}_k\right)-2\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k\right)\right\},\end{aligned} \tag{16}$$

$$\begin{aligned}\mathbf{S}^{(2)}_{\mathbf{y}\mathbf{v}_k} = \frac{\sqrt{\lambda^2-1}}{2\lambda^2}&\left\{\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k+\lambda\mathbf{S}_{\mathbf{v},j}\right)\right.\\ &\left.+\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k-\lambda\mathbf{S}_{\mathbf{v},j}\right)-2\mathbf{h}_i\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k\right)\right\},\end{aligned} \tag{17}$$

$$\widehat{\mathbf{S}}_{\mathbf{y}_k} = \left[\begin{array}{cccc}\mathbf{S}^{(1)}_{\mathbf{y}\overline{\mathbf{x}}_k} & \mathbf{S}^{(1)}_{\mathbf{y}\mathbf{v}_k} & \mathbf{S}^{(2)}_{\mathbf{y}\overline{\mathbf{x}}_k} & \mathbf{S}^{(2)}_{\mathbf{y}\mathbf{v}_k}\end{array}\right]. \tag{18}$$

The predicted measurement $\overline{\mathbf{y}}_k$:

$$\begin{aligned}\overline{\mathbf{y}}_k = &\frac{\lambda^2-n_x-n_v}{\lambda^2}\mathbf{h}\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k\right)+\frac{1}{2\lambda^2}\sum_{p=1}^{n_x}\left(\mathbf{h}\left(\overline{\mathbf{x}}_k+\lambda\overline{\mathbf{S}}_{\mathbf{x},p},\overline{\mathbf{v}}_k\right)\right.\\ &\left.+\mathbf{h}\left(\overline{\mathbf{x}}_k-\lambda\overline{\mathbf{S}}_{\mathbf{x},p},\overline{\mathbf{v}}_k\right)\right)+\frac{1}{2\lambda^2}\sum_{p=1}^{n_v}\left(\mathbf{h}\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k+\lambda\mathbf{S}_{\mathbf{v},p}\right)\right.\\ &\left.+\mathbf{h}\left(\overline{\mathbf{x}}_k,\overline{\mathbf{v}}_k-\lambda\mathbf{S}_{\mathbf{v},p}\right)\right),\end{aligned} \tag{19}$$

where n_v is the size of the observation noise dimension.

(3) Update stage:

$$\widehat{\mathbf{x}}_k = \overline{\mathbf{x}}_k+\mathbf{K}_k\left(\mathbf{y}_k-\overline{\mathbf{y}}_k\right), \tag{20}$$

$$\widehat{\mathbf{S}}_{\mathbf{x}_k} = \left[\begin{array}{cccc}\overline{\mathbf{S}}_{\mathbf{x}_k}-\mathbf{K}_k\mathbf{S}^{(1)}_{\mathbf{y}\overline{\mathbf{x}}_k} & \mathbf{K}_k\mathbf{S}^{(1)}_{\mathbf{y}\mathbf{v}_k} & \mathbf{K}_k\mathbf{S}^{(2)}_{\mathbf{y}\overline{\mathbf{x}}_k} & \mathbf{K}_k\mathbf{S}^{(2)}_{\mathbf{y}\mathbf{v}_k}\end{array}\right], \tag{21}$$

$$\mathbf{K}_k = \widehat{\mathbf{S}}_{\mathbf{x}_k}\left(\mathbf{S}^{(1)}_{\mathbf{y}\overline{\mathbf{x}}_k}\right)^{T}\left(\widehat{\mathbf{S}}_{\mathbf{y}_k}\widehat{\mathbf{S}}_{\mathbf{y}_k}^{T}\right)^{-1}, \tag{22}$$

$$\widehat{\mathbf{P}}_{\mathbf{k}} = \widehat{\mathbf{S}}_{\mathbf{x}_k}\widehat{\mathbf{S}}_{\mathbf{x}_k}^{T}. \tag{23}$$

3. Differential Evolution Optimized a DD2 Particle Filter

Differential evolution (DE), proposed by Storn and Price [23], is an efficient and powerful population-based stochastic search technique for solving optimization problems over continuous search domain, which has been broadly applied in many scientific and engineering fields. It generates new candidate solutions by combining the parent individual and several other individuals of the same population, capable of dealing with nondifferentiable, nonlinear, and multimodal objective functions.

In this paper, we adopt the DD2 filter to generate the proposal distribution. Meanwhile, the latest measurements are employed to promote the proposal distribution for generating particles more intelligently. After the DD2 importance sampling, the DE technique is utilized to optimize the resample process. The procedure of optimization views each particle as a target vector of the current population in

the search space. There is a weight associated with each particle to determine if the particle is good or not. The population of the DE evolves by mutation, crossover, and selection. All of these operations can improve the valid particle number and reduce sample impoverishment. Thus, the new particles can be better approximate the true state by exploiting new areas, and the effectiveness and diversity of particles can be improved.

The description of the proposed algorithm is summarized as follows:

(1) Initiation: $k = 0$

Draw N particles $\{\mathbf{x}_0^{(i)}\}_{i=1}^N$ from the prior $\mathbf{p}(\hat{\mathbf{x}}_0)$ and set weight $\boldsymbol{\omega}_0^{(i)} = 1/N$, $i = 1, \ldots, N$.

(2) Importance sampling:

(i) Update particles $\{\mathbf{x}_k^{(i)}\}_{i=1}^N$ with the DD2 filter using equations (6)~(23) to obtain $\{\overline{\mathbf{x}}_k^{(i)}\}_{i=1}^N$ and $\{\overline{P}_k^{(i)}\}_{i=1}^N$.

(ii) Sample particles $\{\hat{\mathbf{x}}_k^{(i)}\}_{i=1}^N$ from the proposal distribution $\mathbf{q}(\hat{\mathbf{x}}_k^{(i)} \mid \hat{\mathbf{x}}_{0:\,k-1}^{(i)}, \mathbf{z}_{1:\,k}) = \mathcal{N}(\overline{\mathbf{x}}_k^{(i)}, \overline{P}_k^{(i)})$

(iii) Compute the particle weight and normalize:

$$\boldsymbol{\omega}_k^{(i)} \propto \boldsymbol{\omega}_k^{(i)} \frac{\mathbf{p}\left(\mathbf{z}_k \mid \hat{\mathbf{x}}_k^{(i)}\right)\mathbf{p}\left(\hat{\mathbf{x}}_k^{(i)} \mid \hat{\mathbf{x}}_{k-1}^{(i-1)}\right)}{\mathbf{q}\left(\hat{\mathbf{x}}_k^{(i)} \mid \hat{\mathbf{x}}_{0:\,k-1}^{(i)}, \mathbf{z}_{1:k}\right)}, \tag{24}$$

$$\widetilde{\boldsymbol{\omega}}_k^{(i)} = \frac{\boldsymbol{\omega}_k^{(i)}}{\sum_{i=1}^N \boldsymbol{\omega}_k^{(i)}}. \tag{25}$$

(3) Resampling

In a D-dimensional real parameter space $\mathcal{R}^D$, let the sampled particles $\{\hat{\mathbf{x}}_k^{(i)}\}_{i=1}^N$ denote the state vector of current population $\{\hat{\mathbf{x}}_{i,g} = \hat{\mathbf{x}}_{i,1,g}, \ldots, \hat{\mathbf{x}}_{i,D,g}, i = 1, \ldots, N\}$ at generation $G = 0$. Regard each particle and its weight as the target vector and fitness function, respectively. The resampling is composed of the following three steps.

(i) Mutation

Here, we use the "DE/rand/1" mutation strategy to generate mutant vector $\mathbf{V}_{l,g}$.

$$\mathbf{V}_{l,g} = \hat{\mathbf{x}}_{r_1,g} + \mathbf{F} \cdot \left(\hat{\mathbf{x}}_{r_2,g} - \hat{\mathbf{x}}_{r_3,g}\right), \quad l \neq r_1 \neq r_2 \neq r_3, \tag{26}$$

The indices r_1, r_2, and r_3 are mutually distinct integers randomly chosen from the range $[1, N]$, and all are different from the base index l. $\mathbf{F}$ is a positive mutation scaling factor which affects the differential variation between individuals.

(ii) Crossover

To improve the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation.

$$\%\ \mathbf{U}_{l,g} = \mathbf{U}_{j,l,g} = \begin{cases} \mathbf{V}_{j,l,g}, & \text{if } \left(\text{rand}_j[0,1] \le C_r \text{ or } j = j_{\text{rand}}\right), \\ \hat{\mathbf{x}}_{j,l,g}, & \text{otherwise.} \end{cases} \tag{27}$$

Where j_{rand} is a randomly chosen integer between 0 and 1, $\text{rand}_j[0, 1]$ is a uniformly distributed random number in $[1, D]$, which is generated for each j, and $C_r \varepsilon [0, 1]$ is the crossover rate. By mutation and crossover operation, we can enlarge the search space and the particles can be shifted around real state.

(iii) Selection

With the abovementioned crossover and mutation process, the system generates a large number of particles, and thus the new particles can better approximate the true state by exploiting new areas. Subsequently, we select particles which have large fitness (weight) value. The higher the fitness value is, the more likely the particle is selected in the DE algorithm.

$$\dot{\mathbf{x}}_{l,g+1} = \begin{cases} \mathbf{U}_{l,g}, & \text{if weight}\left(\mathbf{U}_{l,g}\right) \ge \text{weight}\left(\hat{\mathbf{x}}_{l,g}\right). \\ \hat{\mathbf{x}}_{l,g}, & \text{otherwise.} \end{cases} \tag{28}$$

When all individuals of the new generation are calculated, the mutation step is carried out to maintain the diversity of the population. That is to say, the mutation, crossover and selection procedure repeat until the optimum is found or the maximum generation number G is reached. For this reason, the DE adopts an iterative approach for particle generation. As the particles generally become better and better as the iterations go on, the DE can decisively improve the particle quality. Thanks to the improved particle quality, particle samples are moved towards regions where particles have larger values of posterior density function. As a result, the impoverishment problem suffered in the particle filter can be effectively handled, and the degeneracy phenomenon is alleviated obviously.

(4) Output: a set of samples that can be used to approximate the posterior distribution as follows:

$$\widetilde{x}_k = \sum_{i=1}^N \dot{x}_k^{(i)} \dot{\widetilde{\boldsymbol{\omega}}}_k^{(i)}. \tag{29}$$

The basic steps of the proposed algorithm are summarized in Figure 1.

4. Simulation Experiments

In this section, the estimation performance and computational cost of the proposed particle filters are evaluated on a state estimation problem and compared with that of the standard sampling importance-resampling particle filter (SIR-PF), extended particle filter (EPF), unscented particle filter (UPF), and the second-order divided difference particle filter (DD2PF). A typical nonlinear process model and measurement model are as follows [24–29]:

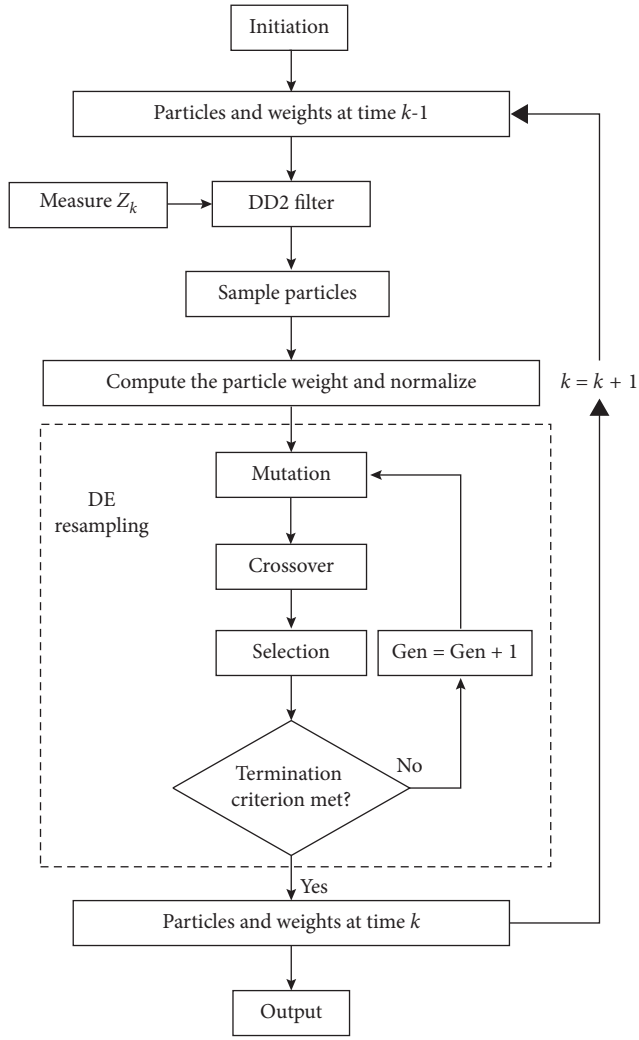

Figure 1: The basic steps of the proposed algorithm.

$$x_k = 1 + \sin(0.04\pi k) + 0.5x_{k-1} + u_{k-1}, \tag{30}$$

$$y_k = \begin{cases} 0.2x_k^2 + \mathbf{v}_k, & k \le 30, \\ 0.5x_k + \mathbf{v}_k - 2, & k > 30, \end{cases} \tag{31}$$

where $\mathbf{u}_k$ is a Gamma $\varsigma_a(3, 2)$ random variable modelling the process noise. The measurement noise $\mathbf{v}_k$ is drawn from a Gaussian distribution $\mathcal{N}(\mathbf{v}_k; 0, 10^{-5})$. The scale factor F, the crossover probability C_r, and the maximum number of generations G are set based on trial-and-error procedures. In this case, we set $F = 0.9$, $C_r = 0.6$, and $G = 20$.

To qualify a fair comparison of the estimates produced by each of the five filters, the estimates are averaged across a Monte Carlo simulation consisting of 200 runs. Each run is implemented with a different noise sample. Ten and two hundred particles are used, respectively. In order to evaluate performance of the proposed algorithm, we use the traditional measure of performance: the Root Mean Squared Error (RMSE). The RMSE of each run is defined by

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(\tilde{\mathbf{x}}_k - x_k\right)^2}. \tag{32}$$

For reference, the true state and the state estimate of the different particle filter for a random Monte Carlo run are presented in Figures 2 and 3. As Figures 2 and 3 show, states estimated by DD2PFDE is the most closed to the true state. Figures 4 and 5 show the variation of estimation RMSE with observation time for all particle filters. In addition, Table 1 summarizes the performance of different filters for 200 MC tests. The table demonstrates the means and variances of the mean square-error of the state estimates as well as the average processing time in seconds of each filter.

As can be seen from Figure 2, the estimates for the DD2PF, UPF, EPF, and the SIR-PF may still be far away from the true values. The reason for the failure is that only one particle may be significantly weighted and the others are much less weighted. But on the contrary, in Figure 2, the proposed particle filter DD2PFDE can track successfully throughout the whole period. Owing to the adding of

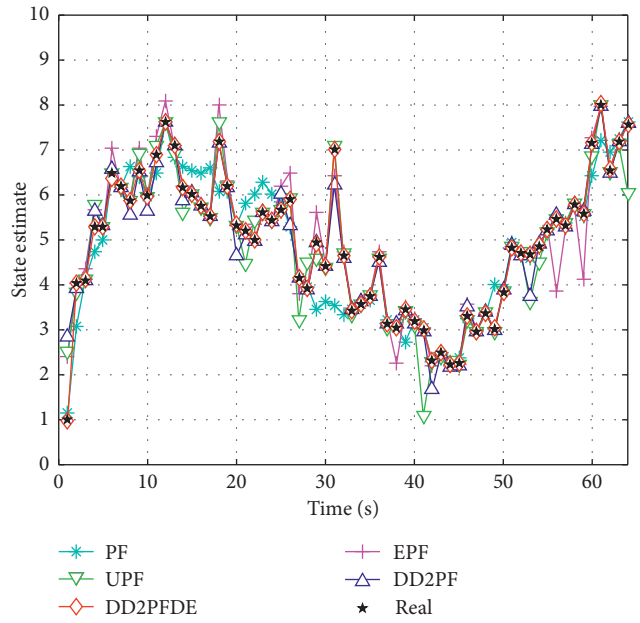

Figure 2: Filtering result comparison of five filters in a random simulation run, where particle numbers $N = 10$.

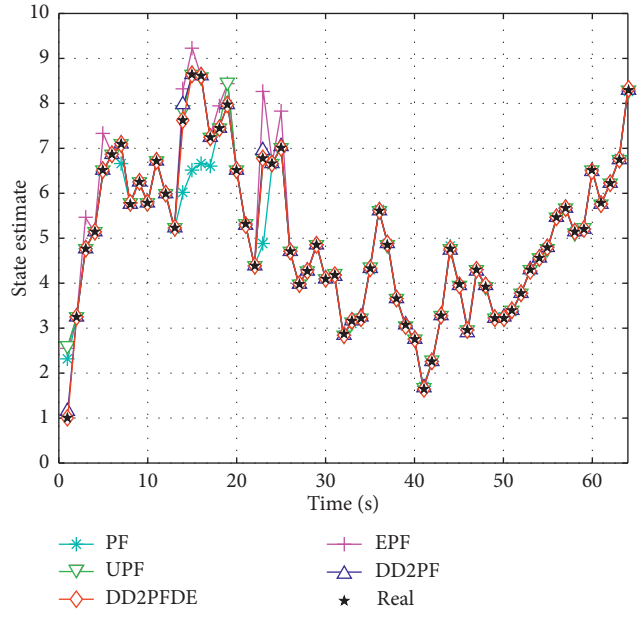

Figure 3: Filtering result comparison of five filters in a random simulation run, where particle numbers $N = 200$.

mutation, crossover and selection, which can be seen in Section 3, and particles in DD2PFDE can exploit more wide areas to find suitable particles to approximate the target's true state. And DD2PFDE can maintain the diversity of the particle set. All of these aforementioned guarantee that the proposed filter can successfully estimate the target.

In Figures 3 and 5, improvements on the five algorithms with 200 particles are readily seen, not only with estimation accuracy, but also in the robustness of the performance. Only with greater numbers, will more particles be considerably weighted through the likelihood function, and a posterior estimate is calculated from the normalized weighted sum of particle estimates, resulting to be much closer to the true values.

As tabulated in Table 1, when the number of particles used by five algorithms is the same, we can see the SIR-PF, EPF, DD2PF, UPF, and DD2PFDE runtime increases in turn. EPF and UPF take longer than SIR-PF due to the addition of EKF and UKF filtering before particle sampling. The EKF provides a faster processing speed than the UKF because it does not need to augment the state dimension, select, and calculate the sigma points in separate procedures. Comparing with the other four algorithms, the DD2PFDE is the most time consuming. This is due to the fact that both the

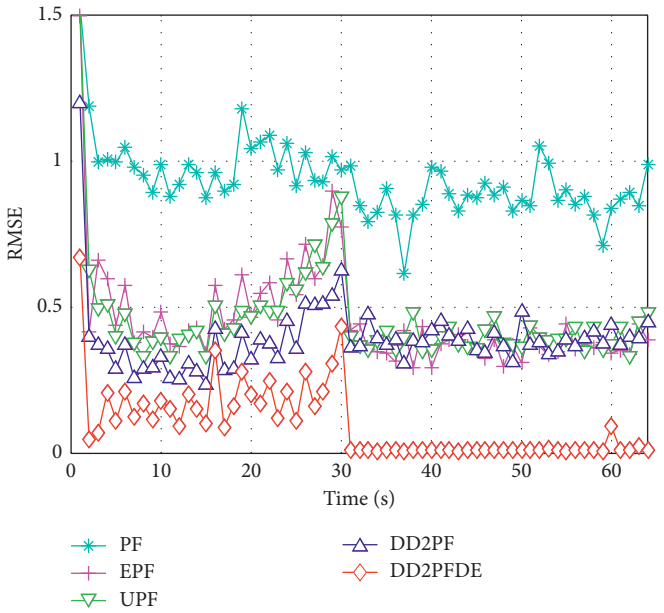

Figure 4: RMSE of SIR-PF, EPF, UPF, DD2PF, and DD2PFDE for 200 Monte Carlo simulations, where particle numbers $N = 10$.

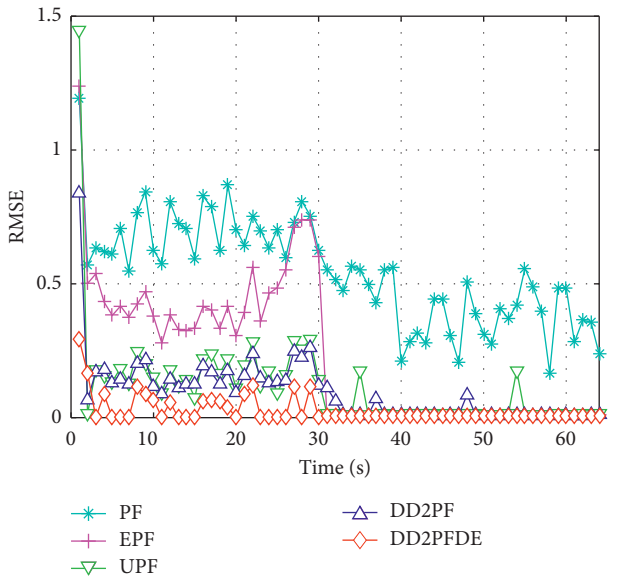

Figure 5: RMSE of SIR-PF, EPF, UPF, DD2PF, and DD2PFDE for 200 Monte Carlo simulations, where particle numbers $N = 200$.

Table 1: Estimation results and computation times averaged over 200 Monte Carlo runs.

Algorithm	Mean	Variance	Time(s)	Particles
SIR-PF	0.9431	0.0218	0.13011	
EPF	0.4915	0.0076	0.21287	
UPF	0.4911	0.0051	0.39452	10
DD2PF	0.4101	0.0050	0.30124	
DD2PFDE	0.1598	0.0043	0.51271	
SIR-PF	0.5803	0.0372	1.45103	
EPF	0.3503	0.0107	3.55714	
UPF	0.2220	0.0082	6.85312	200
DD2PF	0.1527	0.0041	5.56231	
DD2PFDE	0.0769	0.0031	9.43107	
SIR-PF	0.2190	0.0068	2.67380	1800

DD2 filter and the DE iteration are introduced into DD2PFDE.

In comparison with UKF, DD2 filter presents better accuracy and less computation cost and complexity. Therefore, DD2PF provides better performance than UPF. In order to evaluate the efficiency of the proposed algorithm correctly, we increase the number of PF particles to 1800. Two hundred Monte Carlo experiments show that the RMSE mean is 0.2190, and the average computation time is 2.6738 s. Note that 10 DD2PFDE particles achieve higher accuracy (the RMSE mean is 0.1598) and spend less time (the average run time is 0.51271 s) than 1800 PF particles, which indicate that the particle utilization efficiency of the proposed algorithm is higher. Because DE can mutate and select sampling with fitness function, better particles are selected and various particles are generated. By using the DE schemes, the diversity of the particles is increased. As a result, the performance of estimation is enhanced. Despite the proposed algorithm running time is longer than other algorithms; it can decrease the degeneracy phenomenon and alleviate the problem of sample impoverishment.

5. Conclusions

In this paper, we propose an efficient second-order divided difference particle filter based on differential evolution. The filter adopts the DD2 filter to generate the proposal distribution, and the differential evolution algorithm to optimize the resampling process. The DD2 filter linearizes the nonlinear dynamic model by using a multivariable extension of Stirling's interpolation formula rather than the derivative-based Taylor series approximation. The filter provides excellent accuracy without the need to analytically calculate Jacobians. In addition, by using square root factorizations of covariance, the numerical properties of estimation are generally improved. Because the proposed algorithm incorporates the latest measurement information into the sampled particles, the sampled particles are optimized from the source, and the differential evolution algorithm is used to iteratively optimize the sampled particles, thus effectively alleviate the problem of particle depletion and making improved algorithm particle utilization. As a result, the accuracy of the filter would be further enhanced. We illustrated the feasibility of our proposed filtering algorithm for testing nonlinear filtering problems. The simulation results demonstrate that the proposed particle filter can perform better than the other sequential particle filters and can be easier to implement.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form part of an ongoing study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (61671333), the Natural Science Foundation of Hubei Province (2014CFA093), and the Fundamental Research Funds for the Central Universities (2042015gf0029).

References

[1] O. E. Barndorff-Nielsen and N. Shephard, "Non-Gaussian Ornstein-Uhlenbeck-based models and some of their uses in financial economics," *Journal of the Royal Statistical Society: Series B*, vol. 63, no. 2, pp. 167–241, 2001.

[2] F. A. L. Aiube, E. A. H. Baidya, and E. A. Tito, "Analysis of commodity prices with the particle filter," *Energy Economics*, vol. 30, no. 2, pp. 597–605, 2008.

[3] T. Zhang, C. Xu, and M.-H. Yang, "Learning multi-task correlation particle filters for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 365–378, 2019.

[4] M. Firouznia, K. Faez, H. Amindavar, and J. A. Koupaei, "Chaotic particle filter for visual object tracking," *Journal of Visual Communication and Image Representation*, vol. 53, pp. 1–12, 2018.

[5] F. Gustafsson, F. Gunnarsson, N. Bergman et al., "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[7] A. Doucet, N. D. Freitas, and E. Wan, "The unscented particle filter," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, MIT Press, Hong Kong, China, October 2000.

[8] S. J. Julier, "The spherical simplex unscented transformation," in *Proceedings of the IEEE American Control Conference*, pp. 2430–2434, Denver, CO, USA, June 2003.

[9] W. Guo, C. Han, and M. Lei, "Research on particle filter based on spherical unscented transformation," in *Proceedings of the 7th World Congress on Intelligent Control & Automation*, pp. 8388–8392, Chongqing, China, June 2008.

[10] Y. Zhao, "An improved unscented particle filter with global sampling strategy," *Journal of Computational Engineering*, vol. 2014, Article ID 175820, 6 pages, 2014.

[11] S. Godsill and T. Clapp, "Improvement strategies for Monte Carlo particle filter," in *Sequential Monte Carlo Methods in Practice*, Springer, New York, NY, USA, 2001.

[12] F. Wang and Y. Lin, "Improving particle filter with a new sampling strategy," in *Proceedings of the 2009 4th International Conference On Computer Science & Education*, pp. 408–412, Nanning, China, July 2009.

[13] T. S. Schei, "A finite-difference method for linearization in nonlinear estimation algorithms," *Automatica*, vol. 33, no. 11, pp. 2053–2058, 1997.

[14] T. Higuchi, "Monte Carlo filter using the genetic algorithm operators," *Journal of Statistical Computation and Simulation*, vol. 59, no. 1, pp. 1–23, 1997.

[15] J. Zhong and Y.-F. Fung, "Case study and proofs of ant colony optimisation improved particle filter algorithm," *IET Control Theory & Applications*, vol. 6, no. 5, pp. 689–697, 2012.

[16] W. Xian, B. Long, M. Li, and H. Wang, "Prognostics of lithium-ion batteries based on the verhulst model, particle swarm optimization and particle filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 1, pp. 2–17, 2014.

[17] E. S. Wang, X. K. Li, and T. Pang, "A particle filtering algorithm based on the BP neural network," *CAAI Transactions on Intelligent Systems*, vol. 9, no. 6, pp. 709–713, 2014.

[18] Y. Li, J. Li, and J. Zhong, "Research on modelling and optimisation of RBF neural network based on particle filter," *International Journal of Modelling, Identification and Control*, vol. 11, no. 3-4, p. 218, 2010.

[19] H. Chu, K. Wang, and X. Xing, "Target tracking via particle filter and convolutional network," *Journal of Electrical and Computer Engineering*, vol. 2018, Article ID 5381962, 9 pages, 2018.

[20] M. A. Naidoo, L. E. Olivier, and I. K. Craig, "Combined neural network and particle filter state estimation with application to a run-of-mine ore mill," *IFAC Proceedings Volumes*, vol. 46, no. 32, pp. 397–402, 2013.

[21] E. Wang, P. Tao, C. Ming et al., "Application of neural network aided particle filter in GPS receiver autonomous integrity monitoring," in *Proceedings of the China Satellite Navigation Conference (CSNC) 2014*, vol. 2, Nanjing, China, May 2014.

[22] S. R. D. A. Carolina, W. B. Da silva, and J. C. Sampaio Dutra, "propylene polymerization reactor control and estimation using a particle filter and neural network," *Macromolecular Reaction Engineering*, vol. 11, no. 6, 2017.

[23] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[24] R. Merwe, A. Doucet, N. Freitas et al., "The unscented particle filter," Technical Report CUED/F INFENG/TR 380, The Cambridge University Engineering Department, Cambridge, England, 2000.

[25] G. Tong, Z. Fang, and X. Xu, "A particle swarm optimized particle filter for nonlinear system state estimation," in *Proceedings of the 2006 IEEE International Conference on Evolutionary Computation*, pp. 438–442, Vancouver, BC, Canada, July 2006.

[26] Q. Cheng and P. Bondon, "A new unscented particle filter," in *Proceedings of the IEEE International Conference on Acoustics*, pp. 3417–3420, Las Vegas, NV, USA, March–April 2008.

[27] Y. Tian and L. Chen, "Unscented particle filter algorithm based on artificial fish swarm algorithm," in *Proceedings of the Eighth International Conference on Natural Computation*, IEEE, pp. 1123–1126, Chongqing, China, May 2012.

[28] J. Zuo, Y. Jia, and Q. Gao, "Simplified unscented particle filter for nonlinear/non-Gaussian Bayesian estimation," *Journal of Systems Engineering and Electronics*, vol. 24, no. 3, pp. 537–544, 2013.

[29] Y. Wu, J. Wang, X. Lu, and Y. Cao, "Modified sequential importance resampling filter," *Journal of Systems Engineering and Electronics*, vol. 26, no. 3, pp. 441–449, 2015.