World Electric Vehicle Journal

MDPI

Article

State of Charge Estimation for Power Battery Base on Improved Particle Filter

Xingtao Liu [1], Xiaojie Fan [1], Li Wang [1,*] and Ji Wu [1,2]

[1] School of Automotive and Transportation Engineering, Hefei University of Technology, Hefei 230009, China
[2] Engineering Research Center for Intelligent Transportation and Cooperative Vehicle-Infrastructure of Anhui Province, Hefei 230009, China
* Correspondence: wangli@hfut.edu.cn

Abstract: In this paper, an improved particle filter (Improved Particle Swarm Optimized Particle Filter, IPSO-PF) algorithm is proposed to estimate the state of charge (SOC) of lithium-ion batteries. It solves the problem of inaccurate posterior estimation due to particle degradation. The algorithm divides the particle population into three parts and designs different updating methods to realize self-variation and mutual learning of particles, which effectively promotes global development and avoids falling into local optimum. Firstly, a second-order RC equivalent circuit model is established. Secondly, the model parameters are identified by the particle swarm optimization algorithm. Finally, the proposed algorithm is verified under four different driving conditions. The results show that the root mean square error (RMSE) of the proposed algorithm is within 0.4% under different driving conditions, and the maximum error (ME) is less than 1%, showing good generalization. Compared with the EKF, PF, and PSO-PF algorithms, the IPSO-PF algorithm significantly improves the estimation accuracy of SOC, which verifies the superiority of the proposed algorithm.

Keywords: lithium-ion battery; state of charge; particle filter; particle swarm optimization

Citation: Liu, X.; Fan, X.; Wang, L.; Wu, J. State of Charge Estimation for Power Battery Base on Improved Particle Filter. *World Electr. Veh. J.* **2023**, *14*, 8. https://doi.org/10.3390/wevj14010008

Academic Editor: Michael Fowler

Received: 25 October 2022
Revised: 23 November 2022
Accepted: 28 November 2022
Published: 28 December 2022

1. Introduction

Replacing traditional fuel vehicles with new energy vehicles is an effective way to deal with the energy crisis and environmental pollution [1]. Lithium-ion batteries have become the main source of power for electric vehicles owing to their low cost and low pollution. The state of charge (SOC) estimation is one of the core functions of battery management systems of electric vehicles, which can avoid overcharge and over-discharge of batteries, achieve balanced management of battery packs, and effectively guide users to charge the batteries. Accurate estimation of the SOC can improve battery working efficiency, prolong battery life, and play an important role in ensuring the stable and safe operation of battery packs [2].

At present, the SOC estimation methods mainly include the ampere-hour integration (Ah) method, open-circuit voltage (OCV) method, model-based method, and data-driven method [3]. Despite the Ah integration method being widely used in practical applications as its simple principle, its error will increase with the accumulation of errors as it is an open-loop form of estimation method, which leads to the deterioration of estimation accuracy [4,5]. The OCV method obtains the relationship between the open circuit voltage and the SOC through the discharge experiments and then estimates the SOC based on the relationship above. However, this method requires a long battery resting time, which is time-consuming and difficult to apply to online estimation [6–8]. The data-driven method achieves accurate state estimation by obtaining a black-box model through training a large amount of data. But its internal algorithm is complex, which leads to excessive computation, and its accuracy will be significantly reduced when the amount of data is small [9–11]. It is widely used for model-based methods as it has low complexity, high precision, and easy parameter identification complexity [12,13]. Model-based filtering

algorithms include the Kalman filter (KF) and Particle filter (PF). For the reason of the traditional KF algorithm can only deal with linear problems, many improved KF algorithms are derived. Qiu et al. [14] used the least squares method to identify the parameters of the battery model and then combined it with the extended Kalman filter (EKF) algorithm to estimate the SOC. The results show that the method could accurately estimate the SOC. Tan et al. [15] proposed an improved unscented Kalman filter (UKF) algorithm for the problem that the voltage observation data is susceptible to environmental interference, which could cause a decrease in the accuracy. The results show that the algorithm has strong robustness and high precision. Liu et al. [16] proposed a Drift-Ah integration method for the influence of drift current on SOC estimation and used the volumetric Kalman filter (CKF) algorithm to estimate the SOC. The results show that the method can effectively reduce the influence of drift current and improve estimation accuracy. Li et al. [17] proposed an improved CKF algorithm for the noise in the CKF algorithm that cannot satisfy the real-time dynamic characteristics during the iterative process. The experimental results show that the proposed method has higher estimation accuracy and better stability than the EKF algorithm. Tang et al. [18] proposed a method that fuses the EKF algorithm, HIF algorithm, and AEKF algorithm to estimate SOC estimation, which shows a lower computational burden.

However, the improved KF algorithms are still limited to the Gaussian noise environment and difficult to apply in practical scenarios, while the PF algorithm [19–21] has a greater application prospect for it is not limited by Gaussian noise. Wu et al. [22] proposed an adaptive PF algorithm based on a multi-model to solve the problem of accuracy degradation of the multi-target tracking problem with unknown noise statistics when solved by the traditional filter. The results show that the algorithm has strong anti-interference ability and high accuracy. Li et al. [23] used the PF algorithm to process underwater acoustic signals under non-Gaussian conditions. The simulation results show that the PF algorithm has significantly improved tracking performance compared with the EKF algorithm. Shao et al. [24] used the PF algorithm to estimate the SOC and validated the accuracy under electric vehicle driving conditions in Beijing. The results show that the PF algorithm has better estimation performance and has the same computational complexity compared with the EKF algorithm and UKF algorithm. Aiming at the problem that the reduction of the SOC estimation accuracy due to errors introduced by drift current when the current sensor collected the current, Liu et al. [25] used the drift current as a parameter of the temperature compensation model. Then a two-particle filter estimator was designed to estimate the SOC and the drift current at the same time. The results showed that the algorithm has high SOC estimation accuracy and good robustness. After several iterations of the conventional particle filter algorithm, the particle diversity decreases, and the weight difference becomes larger. Since the likelihood function used in particle update is a normal distribution, only a small part of the particles will increase in weight after the update, while most particles will degenerate. Eventually, only a small number of particles can represent the posterior probability distribution, resulting in a significant decrease in estimation accuracy. To improve the particle diversity, Zhao et al. [26] used the EKF as the proposed distribution of the PF algorithm. The simulation experiments showed that the algorithm significantly improves the accuracy of SOC, but the method is computationally intensive. Wu et al. [27] proposed an unscented particle filter (UPF) algorithm by improving the PF algorithm with the UKF algorithm. The results showed that the algorithm is more stable and has better performance than the PF algorithm, but the method lacks practicality. Bi et al. [28] optimized the update process of the PF algorithm by using an artificial immune algorithm to increase the diversity of particles. The results showed that the estimation accuracy was improved by about 40% compared with that before optimization, but the method caused an excessive computation as a large number of particles were adopted.

As the research listed above, some algorithms have complex update processes and an excessive number of particles, which lead to poor practicality and degradation of SOC estimation accuracy. This paper proposes an improved particle swarm-optimized particle

filter (IPSO-PF). It uses a particle swarm algorithm to effectively overcome the problem of decreasing particle diversity in the PF algorithm. The overall framework of the article is rough as follows. Section 2 will establish a second-order RC equivalent circuit model. The proposed IPSO-PF algorithm will be introduced in Section 3. Section 4 will use the particle swarm algorithm to identify the parameters offline and compare the SOC estimation results with the EKF, PF, PSO-PF, and IPSO-PF algorithms. Then the estimation accuracy of the proposed algorithm is verified by experiments. Specifically, the contributions are presented as follows.

(1) The PSO algorithm is used to identify the parameters of the second-order RC model to obtain a battery model with high accuracy.
(2) This paper designs self-mutation and mutual learning of particles to drive particles to high-likelihood regions, which effectively suppresses the degradation of particle weights without increasing the number of particles and increasing particle diversity.
(3) Compared with the PF algorithm, the IPSO-PF algorithm has better particle diversity and much higher estimation accuracy in any SOC stage under different driving conditions.

2. Battery Modeling and Parameter Identification

2.1. Battery Modeling

The battery SOC reflects the remaining charge inside the battery and is usually calculated by the ampere-hour integration method [29]. The discretized expression is shown below:

$$SOC_k = SOC_{k-1} + \eta \frac{\Delta T \cdot I_k}{C_n} \tag{1}$$

where SOC_k represents the SOC value at time k, SOC_{k-1} represents the SOC value at time $k-1$, η represents the charge/discharge efficiency which is taken as 1 in this paper, C_n represents the rated capacity of the battery, I_k represents the current value at time k (positive for charging and negative for discharging).

Accurate SOC estimation requires an accurate battery model, and the equivalent circuit model requires fewer parameters to be identified and is the most widely used in practical applications compared to electrochemical models. The second-order RC equivalent circuit model uses two RC networks to simulate the concentration polarization characteristics and electrochemical polarization characteristics inside the battery, respectively. It has higher accuracy compared with the first-order model and lower complexity compared with models that have more orders. Therefore, the second-order RC equivalent circuit model is selected in this paper for SOC estimation, as shown in Figure 1.

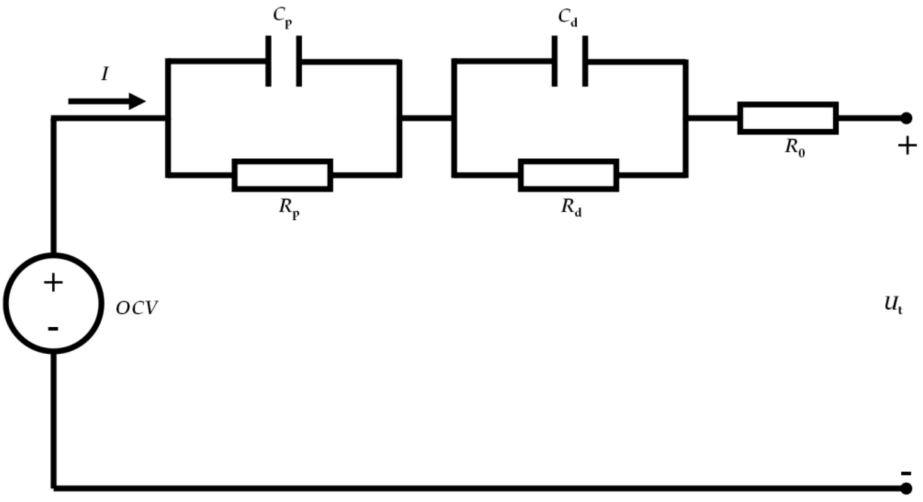

Figure 1. Second-Order RC Equivalent Circuit Model.

Where U_t is the terminal voltage of the battery, R_0 is the internal resistance of the battery, R_p and R_d are the polarization resistance, C_p and C_d are the polarization capacitance, OCV is the open circuit voltage of the battery. There has a relatively stable nonlinear relationship between OCV and SOC, which can be expressed as $OCV = f(SOC)$. The open circuit voltage can be approximated by the terminal voltage after a certain period of battery resting. In this paper, the terminal voltage obtained after each 10% SOC change is selected as the open-circuit voltage through the small current condition. Then the data points were curve-fitted to obtain the following sixth-order polynomial relationship:

$$OCV = 9.04SOC^6 - 21.29SOC^5 + 13.02SOC^4 + 3.92SOC^3 - 5.87SOC^2 + 2.02SOC + 3.34 \quad (2)$$

The following functional relationship can be obtained using the above model according to Kirchhoff's law:

$$\begin{cases} I = \frac{U_p}{R_p} + C_p \frac{dU_p}{d_t} \\ I = \frac{U_d}{R_d} + C_d \frac{dU_d}{d_t} \\ OCV = U_t + U_p + U_d + IR_0 \end{cases} \quad (3)$$

where U_p and U_d represent the concentration polarization voltage and the electrochemical polarization voltage, respectively. The following state space model can be obtained by discretizing Equation (3) and then combining Equation (1).

$$\begin{pmatrix} SOC_k \\ U_{p,k} \\ U_{d,k} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{-\frac{\Delta T}{R_p C_p}} & 0 \\ 0 & 0 & e^{-\frac{\Delta T}{R_d C_d}} \end{pmatrix} \begin{pmatrix} SOC_{k-1} \\ U_{p,k-1} \\ U_{d,k-1} \end{pmatrix} + \begin{pmatrix} \eta \frac{\Delta T}{C_n} \\ R_p \left(1 - e^{-\frac{\Delta T}{R_p C_p}}\right) \\ R_d \left(1 - e^{-\frac{\Delta T}{R_d C_d}}\right) \end{pmatrix} I_k + \omega_k \quad (4)$$

$$U_{t,k} = OCV_k + U_{p,k} + U_{d,k} + I_k R_0 + v_k \quad (5)$$

where ω_k is the process noise and v_k is the measurement noise.

2.2. Parameter Identification

The accuracy of battery model parameters greatly affects the SOC estimation accuracy, so it is necessary to accurately identify the parameters, R_0, R_p, C_p, R_d, and C_d in the battery model. Since the calculational effort of online identification is greater compared to that of offline identification, this paper uses the particle swarm optimization algorithm for offline identification of the model under incremental current conditions. The current and voltage of the incremental current condition are shown in Figures 2 and 3, respectively.

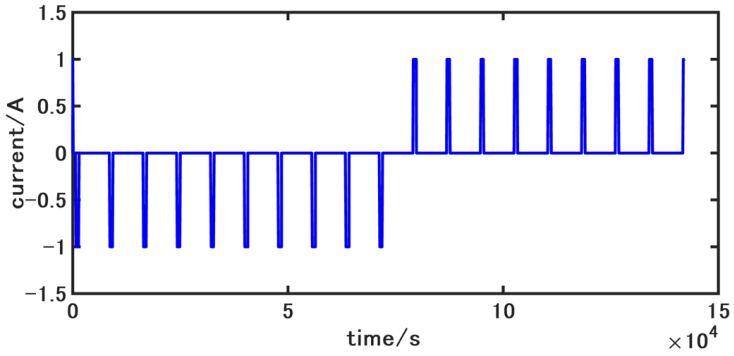

Figure 2. Current of Incremental Current Condition.

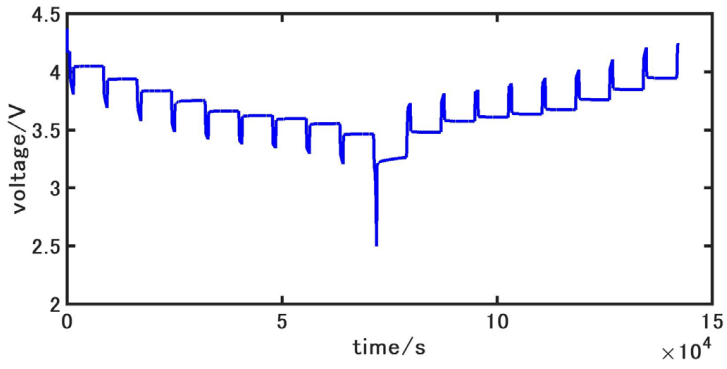

Figure 3. Voltage of Incremental Current Condition.

The particle swarm optimization algorithm obtains the optimal solution by evaluating the results of the fitness function and has a higher identification accuracy than the least squares algorithm [30]. The algorithm treats every feasible solution in the D-dimensional target search space as a particle, and each particle has its velocity and position. In parameter identification, that is, the set of five parameters is treated as a basic particle. During the iterative search process, the particle measures the quality of the current position based on its fitness and then updates it according to the global optimal position G_{best} and the individual optimal position P_{best}. Finally obtains the optimal solution. The specific steps of the PSO algorithm are shown as follows.

Step 1. Initialize the population.

$$\left\{x_{(i)}\right\}_{i=1}^{\text{m}} = \left\{R_{0(i)}, R_{\text{p}(i)}, R_{\text{d}(i)}, C_{\text{p}(i)}, C_{\text{d}(i)}\right\}_{i=1}^{\text{m}} \tag{6}$$

$$\left\{v_{(i)}\right\}_{i=1}^{\text{m}} = \left\{v_{0(i)}, v_{\text{p}(i)}, v_{\text{d}(i)}, v_{\text{p}(i)}, v_{\text{d}(i)}\right\}_{i=1}^{\text{m}} \tag{7}$$

$$\left\{P_{\text{best}(i)}\right\}_{i=1}^{\text{m}} = \left\{R_{0\text{best}(i)}, R_{\text{pbest}(i)}, R_{\text{dbest}(i)}, C_{\text{pbest}(i)}, C_{\text{dbest}(i)}\right\}_{i=1}^{\text{m}} \tag{8}$$

$$G_{\text{best}} = \left\{R_{0\text{best}}, R_{\text{pbest}}, R_{\text{dbest}}, C_{\text{pbest}}, C_{\text{dbest}}\right\}_{i=1}^{\text{m}} \tag{9}$$

Step 2. Set the fitness function.

$$fitness(i) = \sqrt{\sum_{\text{k}=1}^{\text{T}} \left[U_{\text{t}}(k) - U_{\text{p}}(k)\right]^2} \tag{10}$$

where $U_{\text{t}}(k)$ is the actual voltage at the current moment, $U_{\text{p}}(k)$ is the predicted voltage at the current moment, $iter_{now}$ represents the current number of iterations, and $iter_{max}$ represents the maximum number of iterations.

Step 3. Update the speed and position.

While $iter_{now} < iter_{max}$:

$$v_i^D = wv_i^D + c_1 r_1 \left(P_{\text{best}(i)}^D - x_i^D\right) + c_2 r_2 \left(G_{\text{best}}^D - x_i^D\right) \tag{11}$$

$$x_{(i)} = x_{(i)} + v_{(i)} \tag{12}$$

where w is the inertial weight that can balance the global search and local development, c_1 and c_2 are the learning factors that indicate the individual cognition and social cognition of the particles, respectively r_1 and r_2 are the random numbers generated on (0,1).

Step 4. Output the optimal value $G_{\text{best}}^{\text{D}}$.

3. SOC Estimation Based on IPSO-PF Algorithm

The core idea of the IPSO-PF algorithm is to iteratively optimize the prior state sets of particle filtering by using an improved particle swarm optimization algorithm and then update and resample it with the latest observations to obtain more accurate posterior state sets, thus realizing an improvement of the particle filtering algorithm. This section introduces the basic principles of the PF algorithm and the IPSO algorithms, respectively, and applies the IPSO-PF algorithm to the SOC estimation.

3.1. PF Algorithm

Particle filtering is a recursive Bayesian filter based on Monte Carlo sampling, and the core idea is to achieve integrated operations with weighted sums by using discrete random samples. Assume that the state space model of the nonlinear system is represented as follows:

$$\begin{cases} x_k = f(x_{k-1}, u_k) + \omega_k \\ y_k = h(x_k, u_k) + v_k \end{cases} \tag{13}$$

where u_k is the input, y_k is the output. The PF algorithm uses the system model prediction to obtain the prior probability density, then get the posterior probability density by correcting the prior probability density through the latest observations, and finally achieve the state estimation. The specific steps of the PF algorithm are shown as follows:

Step 1. k = 0, initialize the particle sets: $\left\{x_0^{(i)}\right\}_{i=1}^{N}$, $\left\{w_0^{(i)}\right\}_{i=1}^{N} = 1/N$.

Step 2. k = 1, 2, . . . , T, execute the following steps:

1. Prediction:

$$\left\{\widetilde{x}_k^{(i)}\right\}_{i=1}^{N} = f\left(\left\{x_{k-1}^{(i)}\right\}_{i=1}^{N}, u_k\right) \tag{14}$$

$$\left\{\widetilde{y}_k^{(i)}\right\}_{i=1}^{N} = h\left(\left\{\widetilde{x}_k^{(i)}\right\}_{i=1}^{N}, u_k\right) \tag{15}$$

2. Update:

$$w_k^{(i)} = \left(\frac{1}{\sqrt{2\pi R_k}}\right) \mathrm{e}^{-\frac{\left(y_k - \widetilde{y}_k^{(i)}\right)^2}{2R_k}} \tag{16}$$

$$\widetilde{w}_{\mathrm{k}}^{(\mathrm{i})} = \frac{w_{\mathrm{k}}^{(\mathrm{i})}}{\sum\limits_{\mathrm{i}=1}^{\mathrm{N}} w_{\mathrm{k}}^{(\mathrm{i})}} \tag{17}$$

3. Resample: obtain the posterior state sets $\left\{x_k^{(i)}\right\}_{i=1}^{N}$, $\left\{w_k^{(i)}\right\}_{i=1}^{N} = 1/N$ after resampling the particle sets $\left\{\widetilde{x}_k^{(i)}, \widetilde{w}_k^{(i)}\right\}_{i=1}^{N}$.
4. Estimation:

$$\hat{x}_k = \sum_{i=1}^{\mathrm{N}} w_k^{(i)} x_k^{(i)} \tag{18}$$

3.2. IPSO Algorithm

The likelihood function used by the PF algorithm is a normal distribution type, which is prone to excessive differences between particles, then causes degradation of particles, and finally leads to the reduction of the estimation accuracy. For the defects of the PF, this paper uses an improved particle swarm algorithm [31] to optimize the sampling process before the particle filtering update period. Traditional particle swarm algorithm is prone to fall into local optimal solutions in the iterative search process, so this paper makes the following improvements: the particle population is divided into three parts, and different update strategies are used to realize the self-mutation and mutual learning of particles so that the algorithm can jump out of the local optimum, accelerate convergence and improve

the estimation accuracy. The size of the fitness value reflects the distance of the particle from the real state. The fitness values of the particle population are sorted from small to large and then divided into three parts according to the mean value $\overline{f}$ and the standard deviation σ, as shown in Figure 4.

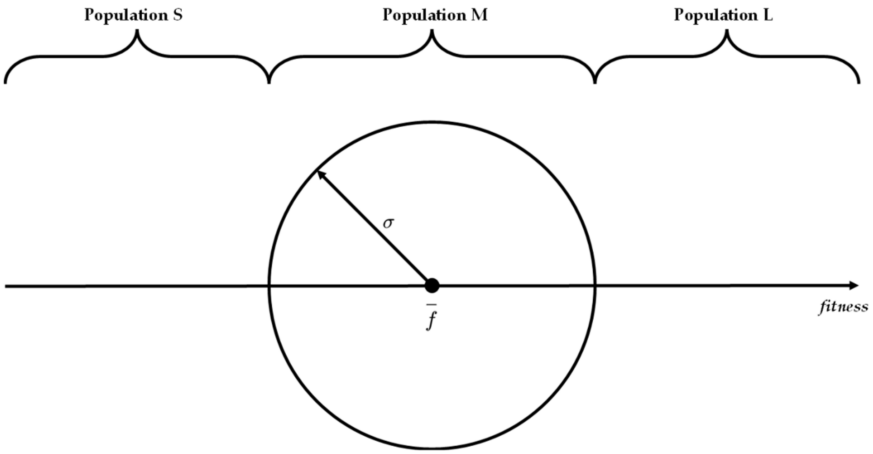

Figure 4. Population Division.

The mean value $\overline{f}$ and the standard deviation σ are defined as follows:

$$\overline{f} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} fitness(\mathrm{i})}{\mathrm{N}} \tag{19}$$

$$\sigma = \sqrt{\frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} fitness(\mathrm{i}) - \overline{f}}{\mathrm{N}}} \tag{20}$$

Particles with a smaller portion of the fitness values (population S) are far from the true state and need to accelerate closer to the global optimal state. The updated rule is as follows:

$$x^{(\mathrm{S})} = x^{(\mathrm{S})} + c_2\left(G_{\mathrm{best}} - x^{(\mathrm{S})}\right) + c_3\left(\overline{x} - x^{(\mathrm{S})}\right) \tag{21}$$

where c_3 is the learning factor, $\overline{x}$ is the particle at the average fitness position. The right side of Equation (21) consists of three parts: the first item indicates the ability of the particle to maintain its position, the second item indicates that the particle learns from particles close to the true state, the third item indicates that the particle is pulled by the particles located in the intermediate states and can avoid particle degradation.

Particles with a middle portion of the fitness values (population M) are in the intermediate state that plays a role in balancing global search and local development, therefore adopting the standard update rule, namely the Equation (11). The nonlinear differential decreasing strategy is used to update the inertia weights. In the early stage of the iteration, the inertia weight is large, and the particle can perform a large range of global searches. In the late stage of the iteration, the inertia weight is small, and the particle has a strong local development ability.

Particles with a large portion of the fitness values (population L) are close to the true state, therefore using self-variation updates to avoid falling into local optimum:

$$x^{(L)} = x^{(\mathrm{L})} \cdot (1 + \alpha \cdot C(0,1)) \tag{22}$$

$$\alpha = \frac{iter_{\max} - iter_{\mathrm{now}}}{iter_{\mathrm{now}}} \tag{23}$$

where α indicates the adaptive factor, $C(0,1)$ indicates the Cauchy random number. Equation (22) is the adaptive Cauchy formula which controls the particles to carry out the global

search in the early stage of the iteration and accelerates the convergence in the late stage of the iteration.

3.3. SOC Estimation

The IPSO-PF algorithm can be obtained by introducing the IPSO algorithm into the sampling process of the PF algorithm. The IPSO algorithm takes the set of prior particles computed by the PF algorithm as the elementary particles, yielding sets closer to the truly distributed states through the self-variation of the particles themselves and the mutual learning between the populations. Then the PF algorithm updates this set to obtain the posterior set. The specific steps of the IPSO-PF algorithm are shown as follows.

Step 1. k = 0, initialize the particle sets: $\left\{x_0^{(i)}\right\}_{i=1}^{N}$, $\left\{w_0^{(i)}\right\}_{i=1}^{N} = 1/N$.

Step 2. k = 1, 2, . . . , T, execute the following steps:

1. Prediction.

$$\left\{\widetilde{x}_k^{(i)}\right\}_{i=1}^{N} = f\left(\left\{x_{k-1}^{(i)}\right\}_{i=1}^{N}, u_k\right) \tag{24}$$

$$\left\{\widetilde{y}_k^{(i)}\right\}_{i=1}^{N} = h\left(\left\{\widetilde{x}_k^{(i)}\right\}_{i=1}^{N}, u_k\right) \tag{25}$$

2. IPSO optimization.

   a. Initialize the population.

$$\left\{x_{(i)}\right\}_{i=1}^{N} = \left\{\widetilde{y}_k^{(i)}\right\}_{i=1}^{N} \tag{26}$$

$$\left\{v_{(i)}\right\}_{i=1}^{N} = rand \tag{27}$$

$$\left\{P_{\mathrm{best}(i)}\right\}_{i=1}^{N} = \{fitness(i)\}_{i=1}^{N} \tag{28}$$

$$G_{\mathrm{best}} = \max\left(\{fitness(i)\}_{i=1}^{N}\right) \tag{29}$$

   b. Set the fitness function.

$$fitness(i) = \mathrm{e}^{-\frac{1}{2R_k}(x_{\mathrm{t}} - x_{\mathrm{p}(i)})2} \tag{30}$$

   c. Update the speed and position.

$$v_{(i)} = wv_{(i)} + c_1 r_1\left(P_{\mathrm{best}(i)} - x_{(i)}\right) + c_2 r_2\left(G_{\mathrm{best}} - x_{(i)}\right) \tag{31}$$

$$x_{(i)} = x_{(i)} + v_{(i)} \tag{32}$$

   d. Output the optimal value.

$$\left\{\widetilde{y}_k^{(i)}\right\}_{i=1}^{N} = \left\{x_{(i)}\right\}_{i=1}^{N} \tag{33}$$

3. Update:

$$w_k^{(i)} = \left(\frac{1}{\sqrt{2\pi R_k}}\right)\mathrm{e}^{-\frac{\left(y_k - \widetilde{y}_k^{(i)}\right)^2}{2R_k}} \tag{34}$$

$$\widetilde{w}_{\mathrm{k}}^{(\mathrm{i})} = \frac{w_{\mathrm{k}}^{(\mathrm{i})}}{\sum\limits_{\mathrm{i}=1}^{\mathrm{N}} w_{\mathrm{k}}^{(\mathrm{i})}} \tag{35}$$

4. Resample: obtain the posterior state sets $\left\{x_k^{(i)}\right\}_{i=1}^{N}$, $\left\{w_k^{(i)}\right\}_{i=1}^{N} = 1/N$ after resampling the particle sets $\left\{\widetilde{x}_k^{(i)}, \widetilde{w}_k^{(i)}\right\}_{i=1}^{N}$.
5. Estimation:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^{(i)} x_k^{(i)} \tag{36}$$

Then, the SOC estimation is performed using the proposed IPSO-PF algorithm. The flowchart of SOC estimation is shown in Figure 5. Firstly, the battery model parameters are obtained by the particle swarm algorithm offline. Secondly, the prior state sets of the current moment are generated by sampling the estimation state of the previous moment through the system model. Thirdly, the prior state sets are divided into three subsets according to fitness, and different strategies are used to realize the update of the sets. Then, the weights of the optimized particle state sets are normalized, and the posterior state sets of the current moment are obtained after resampling. Finally, the state can be estimated.

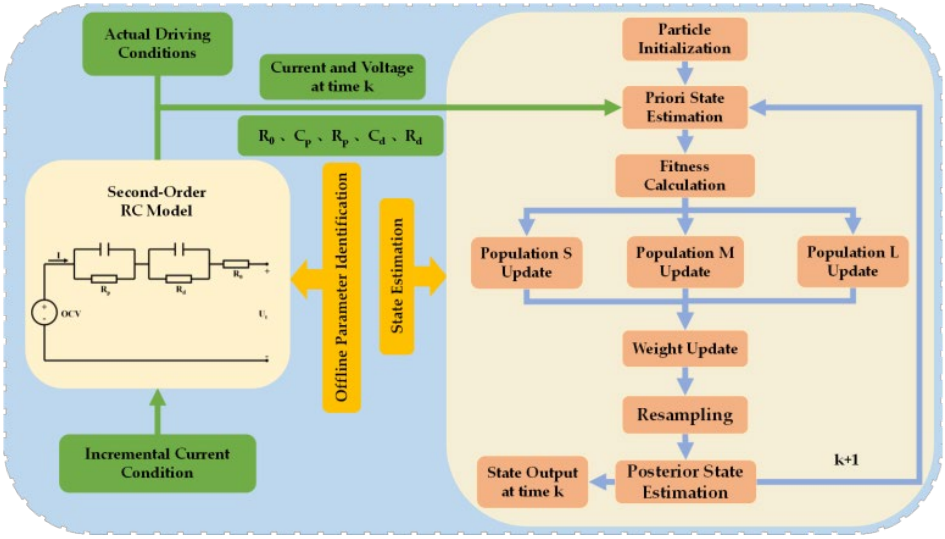

Figure 5. Flow Chart of the SOC Estimation.

4. Experimental Results and Analysis

In this paper, a LiNiMnCo/Graphite lithium-ion battery is selected as the research object, and its basic information is shown in Table 1. The experimental temperature is set to 25 °C, and the sampling period is set to 1s. The number of particles of the PF algorithm and PSO algorithm are both set to 100, the number of iterations of the PSO and IPSO-PF algorithm is set to 200, the maximum and minimum inertia weights are set to 0.9 and 0.4, respectively, and the learning factors c_1, c_2 and c_3 are set as 2.

Table 1. Basic Information about the Battery.

Type	Normal Voltage	Normal Capacity	Cutoff Voltage
INR 18650-20R	3.6 V	2.0 Ah	2.5 V/4.2 V

In order to validate the accuracy of the model and the SOC estimation. The Dynamic Street Test (DST), the Federal Urban Driving Schedule (FUDS), the Intense Driving Schedule (US06 Supplemental FTP Driving Schedule, US06), and the Beijing Dynamic Stress Test (BJDST) from the University of Maryland battery data [32] are selected for the experiment. All tests were performed for 80% battery level and 50% battery level at 0 °C, 25 °C and 45 °C. We chose the data that the battery tests were performed for 80% initial SOC at 25 °C

as they are the most commonly used data. Firstly, the battery is discharged to 80% SOC, then the battery is discharged according to the four working conditions, and the current and voltage are recorded. The current curves are obtained as shown in Figures 6–9. It can be seen that the current fluctuation frequency of the four conditions is high, and the charge/discharge ratio is large, which is relatively close to the actual usage.

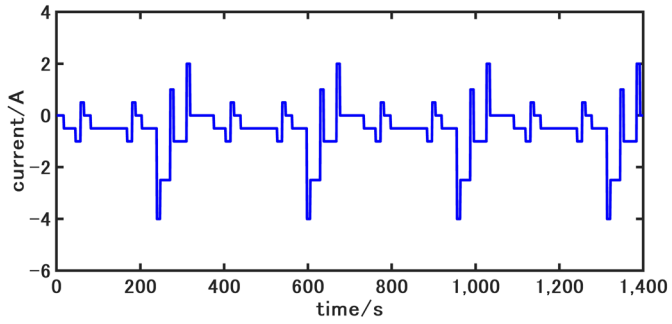

Figure 6. Current of DST Driving Condition.

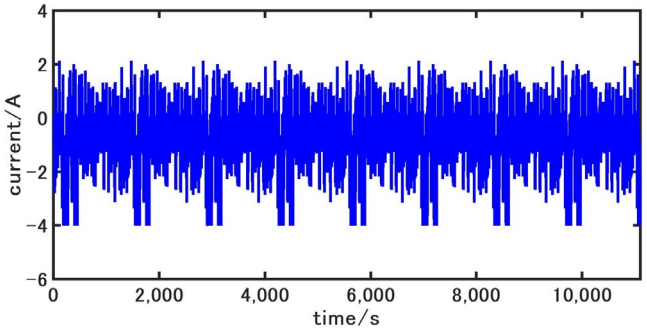

Figure 7. Current of FUDS Driving Condition.

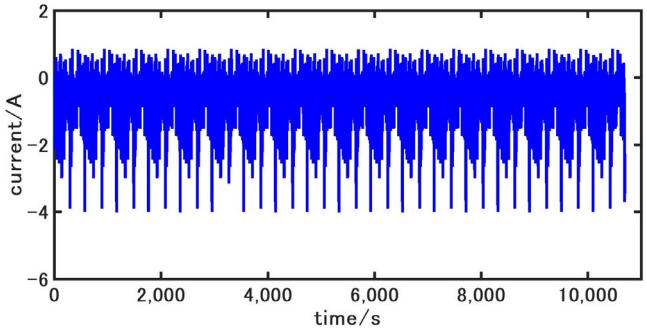

Figure 8. Current of US06 Driving Condition.

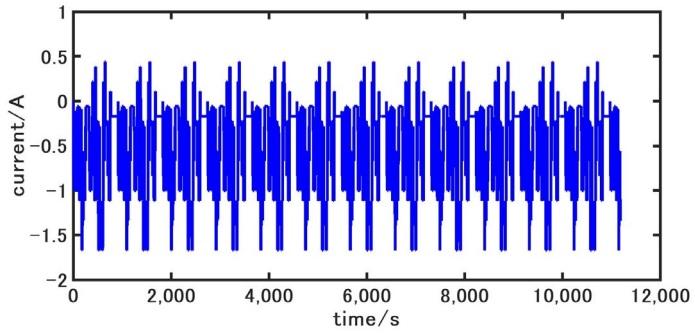

Figure 9. Current of BJDST Driving Condition.

4.1. Model Verification

The root mean square error of terminal voltage is used as an index to evaluate the accuracy of the model, and the parameters of the charging process and the discharging process are identified respectively, then the average values are taken as the model parameters. The fitness curve and parameter identification results are shown in Figure 10 and Table 2, respectively. It can be seen from Figure 10 that the fitness converges to 0.83% after only 50 iterations, which shows the fast convergence speed of the particle swarm algorithm. Meanwhile, we also compare the computation costs and accuracy of the first-order, second-order, and third-order RC models. The results are shown in Table 3. It can be seen from the RMSE of Table 3 that the third order has little improvement in accuracy compared to the second order, while the first order has poor accuracy compared to the second order. Since the parameters are identified offline and the time difference is not significant, the computation time can be neglected. Therefore, it can be considered that the second-order RC model is the optimal choice. In addition, the model parameter identification results are consistent with our theoretical expectations, and we need to verify further whether it is in line with the practical application under different driving conditions.

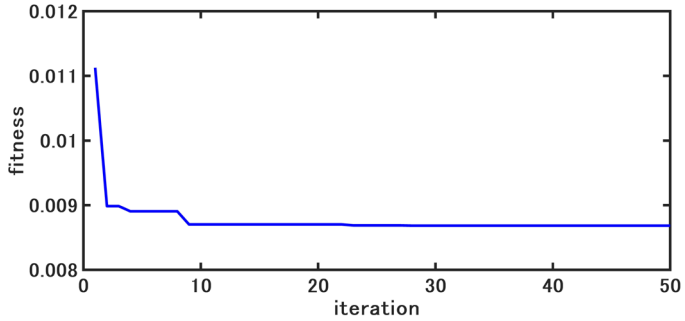

Figure 10. Fitness Curve.

Table 2. Model Parameters.

R_0/Ω	R_p/Ω	R_d/Ω	C_p/F	C_d/F
0.0687	0.0131	0.0035	1359.7	432.6

Table 3. Model Comparison.

Model	Computation Cost	RMSE (%)
first-order	3.55	0.8351
second-order	8.24	0.8337
third-order	15.88	0.8338

In order to verify the accuracy of the second-order RC model, the four driving condition data are selected for verification. Since electric vehicles that drive under extreme loads during actual driving are relatively rare, the battery data from 80% to 10% SOC are selected for experiments in this paper. The battery is discharged according to the current shown in Figures 6 and 7, and the terminal voltage and SOC of the battery are recorded. The above currents are used to simulate the model, and the terminal voltage of the battery model is obtained. The error curves of the model terminal voltage are shown in Figures 11–14. As can be seen, the errors of the voltage of the battery model are controlled within 20 mV overall, and the fluctuation range is small. The maximum errors are only 34.4 mV, 29.9 mV, 35.7 mV, and 21.6 mV, respectively, which shows that the second-order RC equivalent circuit model built in this paper has high accuracy and can well reflect the dynamic working characteristics of the battery.

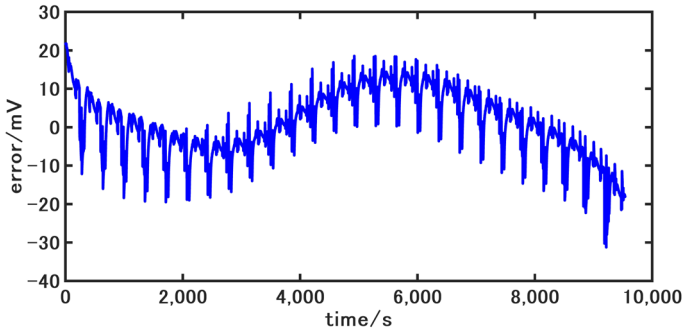

Figure 11. Errors of Terminal Voltage under DST Driving Condition.

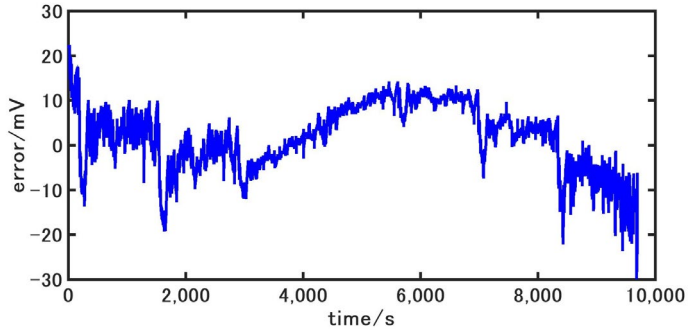

Figure 12. Errors of Terminal Voltage under FUDS Driving Condition.

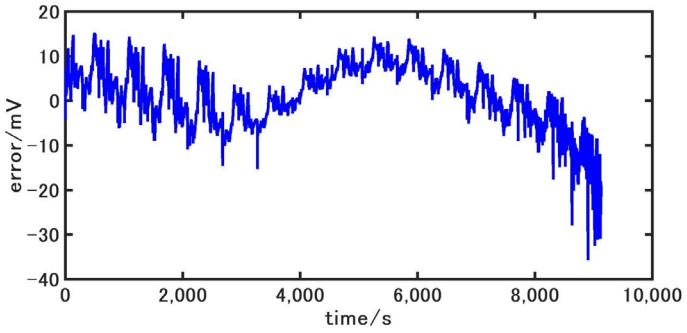

Figure 13. Errors of Terminal Voltage under US06 Driving Condition.

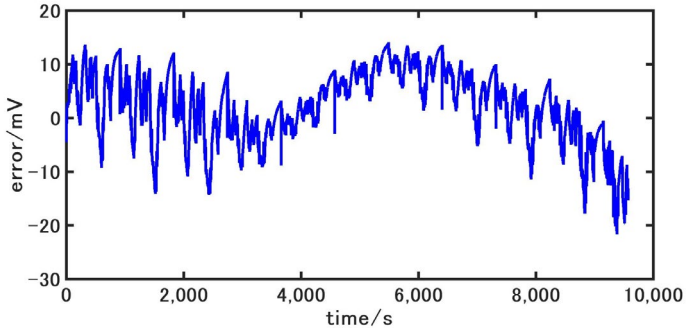

Figure 14. Errors of Terminal Voltage under BJDST Driving Condition.

4.2. SOC Estimation Verification

In order to verify the superiority of the algorithm proposed in this paper, the algorithm is verified using the DST driving condition, and the experimental results are shown in Figures 15–17. As can be seen from Figures 15 and 16, the difference in particle weights of the PF algorithm is big, and the number of particles with large weights is gradually

reduced while the particles with small weights are gradually degraded. Therefore, the sets of particles will gradually deviate from the true state and eventually leads to a decrease in the estimation accuracy. In contrast, the particle weights of the IPSO-PF algorithm have smaller differences and are more uniformly distributed, which can effectively guarantee the diversity of particles and prevent particle degradation, thus improving the estimation accuracy. It can be further illustrated by Figure 17 that the SOC estimation results of the IPSO-PF algorithm are closer to the true value. The EKF algorithm has the worst performance, and it gradually diverges at the end of the estimation phase, resulting in a maximum error approaching 2%. The estimation error of the PF algorithm dissipates with the increase of time, and the maximum error of the PSO-PF algorithm is too large due to the population falling into the local optimal solution, while the IPSO-PF algorithm controls the error within a certain range.

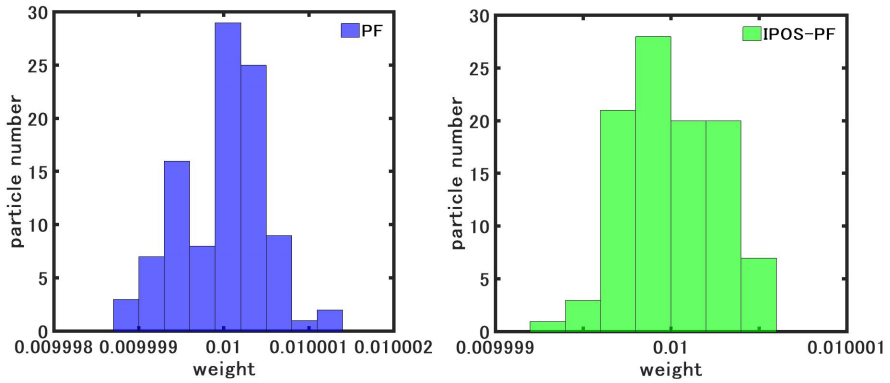

Figure 15. Weight Distribution at SOC = 0.5.

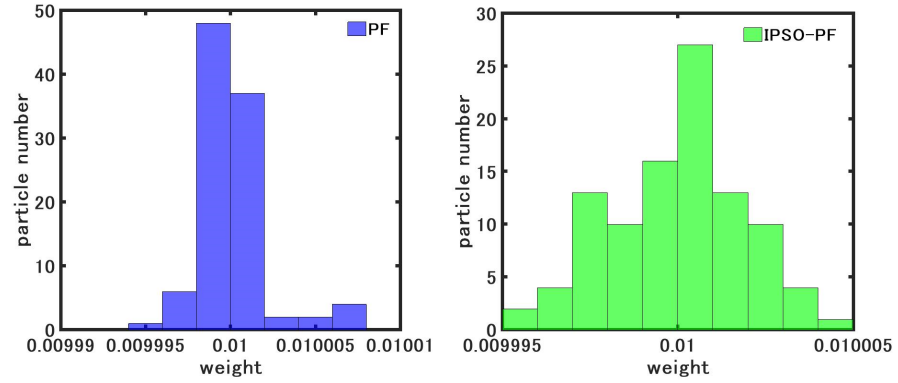

Figure 16. Weight Distribution at SOC = 0.2.

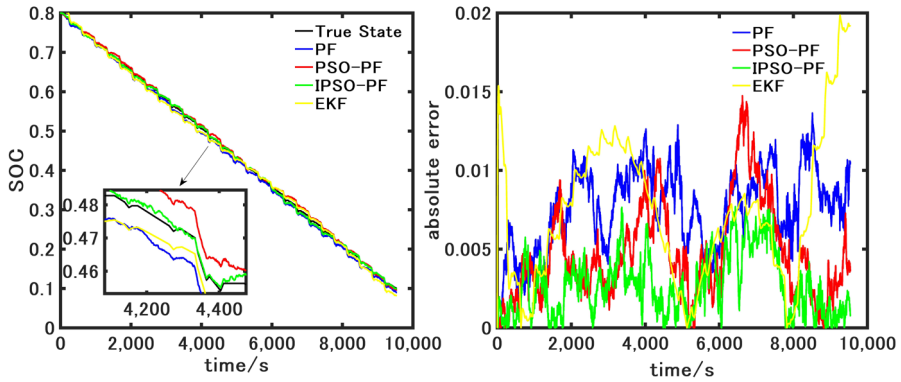

Figure 17. SOC Estimation Results and Errors of DST Driving Condition.

To intuitively compare the estimation performance of the different algorithms under DST driving conditions, three evaluation indexes of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Max Error (ME) are given in this paper, and the results are shown in Table 4. It can be further seen from the error results that the errors of EKF are larger than that of PF because it is not suitable for non-Gaussian noise environments. The RMSE of IPSO-PF decreases by 0.7% compared with EKF and decreases by 0.41% compared with PF, and the error is controlled within 1%, while the other three algorithms are more than 1%, which proves that the proposed algorithm has good estimation accuracy.

Table 4. Errors of DST Driving Condition.

Method	RMSE/%	MAE/%	ME/%
EKF	1.09	1.74	1.99
PF	0.80	0.76	1.37
PSO-PF	0.56	0.47	1.47
IPSO-PF	0.39	0.33	0.99

Theoretically, the computational complexity of the IPSO-PF algorithm increases compared to the PF algorithm due to the inclusion of the PSO algorithm, but it is not much different from that of the PSO-PF algorithm. We ran all the algorithms on the MATLAB 2016b platform of an AMD I7-6500U processor with 3.20 GHz and 16 GB memory and calculated the computational complexity. The EKF, PF, PSO-PF, and IPSO-PF algorithms take 0.89 s, 6.32 s, 8.16 s, and 8.73 s, respectively. Although the EKF algorithm has the lowest complexity, it has poor estimation accuracy in non-Gaussian scenarios. The complexity of the proposed algorithm is almost the same as that of the PSO-PF algorithm and slightly inferior to the PF algorithm, but with a significant improvement in accuracy. Therefore, it further demonstrates the superiority of the IPSO-PF algorithm.

In order to verify the generalizability of the proposed algorithm under different driving conditions, the FUDS, US06, and BJDST data are used for SOC estimation, and then compare the estimation results with the PF and PSO-PF algorithm. The experimental results and errors are shown in Figure 18. As is shown in Figure 18a,c,e, the PF algorithm, the PSO-PF algorithm, and the IPSO-PF algorithm can effectively estimate the SOC, and the estimation results of the IPSO-PF algorithm are closer to the true value. It can be seen from Figure 18b,d,f that the estimation errors of the IPSO-PF algorithm are significantly lower and fluctuate less under different driving conditions compared with the EKF, PF, and PSO-PF algorithms. The three error evaluation indexes are shown in Tables 5–7. According to the results, the ME of the IPSO-PF algorithm is limited to 0.9%, and the MAE is limited to 0.28%, which is less than the ME and MAE of the PF algorithm and the PSO-PF algorithm. The experimental results show that the proposed algorithm effectively solves the problem of estimation error increase due to particle degradation and has good estimation performance. The RMSEs of the IPSO-PF algorithm in FUDS, US06, and BJDST conditions are 0.25%, 0.34%, and 0.33%, respectively, which are at least 55% lower than those of the EKF algorithm, 45% lower than those of the PF algorithm and 22% lower than those of the PSO-PF algorithm, effectively improving the estimation accuracy of the SOC, indicating that the IPSO-PF algorithm has good generalizability and superiority.

Table 5. Errors of FUDS Driving Condition.

Method	RMSE/%	MAE/%	ME/%
EKF	1.13	0.94	2.65
PF	0.71	0.67	1.23
PSO-PF	0.64	0.52	1.54
IPSO-PF	0.25	0.21	0.68

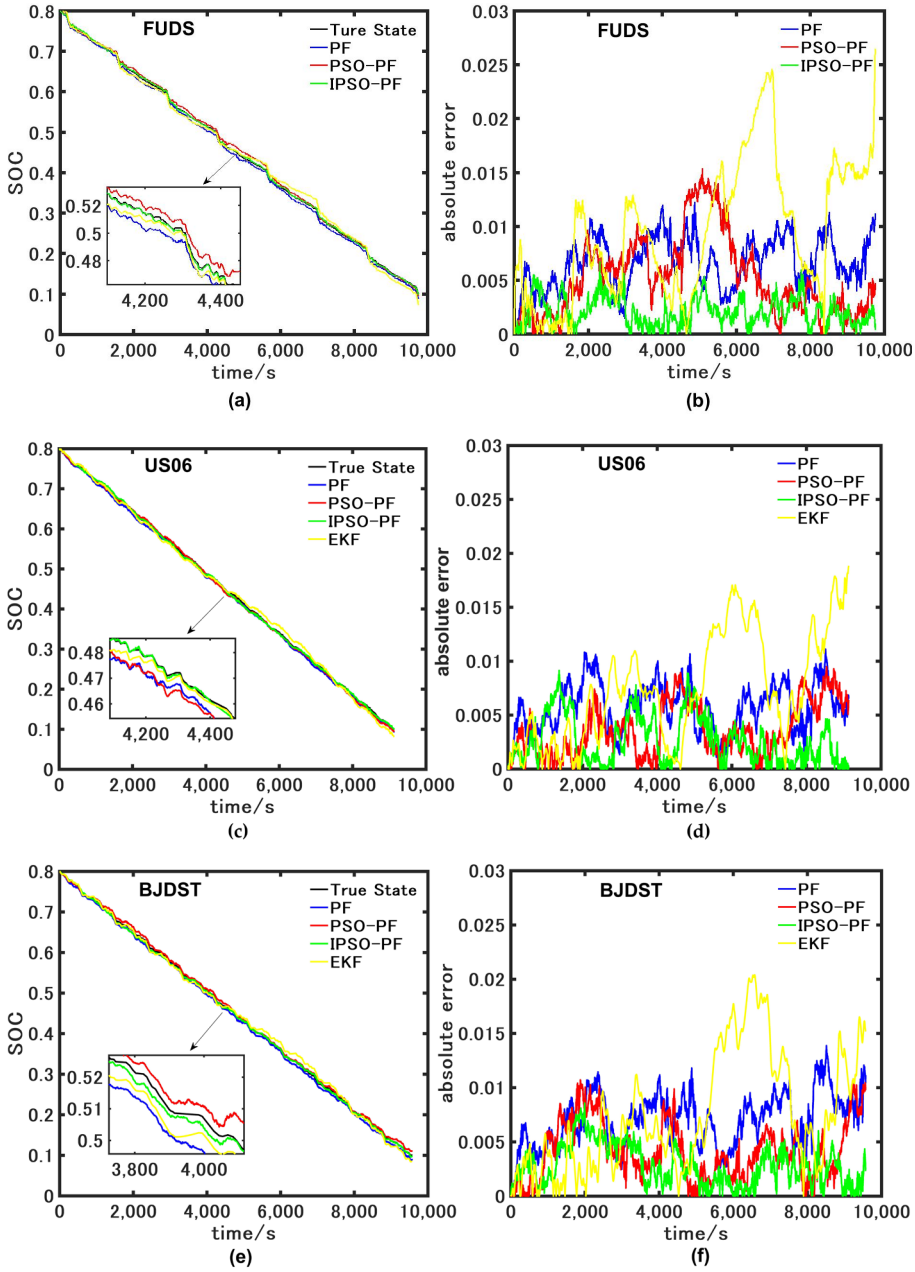

Figure 18. SOC Estimation Results and Errors of Different Driving Conditions.

Table 6. Errors of US06 Driving Condition.

Method	RMSE/%	MAE/%	ME/%
EKF	0.89	0.73	1.88
PF	0.62	0.57	1.11
PSO-PF	0.44	0.36	0.95
IPSO-PF	0.34	0.26	0.90

Table 7. Errors of BJDST Driving Condition.

Method	RMSE/%	MAE/%	ME/%
EKF	0.92	0.75	2.04
PF	0.74	0.70	1.39
PSO-PF	0.49	0.41	1.07
IPSO-PF	0.33	0.28	0.86

The experimental results show that the PF algorithm has an inaccurate posterior estimation accuracy due to particle weight degradation. In contrast, the IPSO-PF algorithm improves the estimation accuracy and shows good generalization by driving all particles closer to the true state.

5. Conclusions

Considering the problem that the PF algorithm adopts the normal distribution type likelihood function in the updating stage, which leads to particle degradation caused by the large difference of weights between particles, this paper proposes an IPSO-PF algorithm. The algorithm introduces the particle swarm optimization algorithm into the sampling process before the update of the PF algorithm, and through the self-variation and mutual learning of the particles, the particles with smaller weights tend to the true state by learning, which suppresses the weight degradation. The particles with larger weights avoid falling into the local optimum by self-variation, which promotes global development. Finally, it makes all particles closer to the true state distribution and improves the estimation performance of the PF algorithm.

The experimental results show that the algorithm proposed in this paper has good estimation accuracy under four actual driving conditions, with the root mean square error within 0.4% and the maximum error within 0.9%, and all of them are smaller than the PF algorithm, and PSO-PF algorithm, which fully verifies the superiority and generalization of the algorithm proposed in this paper.

This paper does not consider the convergence speed of a wrong initial SOC, and the experiments are performed at 25 °C. Therefore, we will make improvements to the convergence of the algorithm and increase the experiments under different temperature conditions to increase the reliability of the algorithm in the future.

Author Contributions: Conceptualization, X.L. and J.W.; methodology, X.L.; software and validation and formal analysis, X.F.; resources, X.L.; data curation and writing—original draft preparation, X.F.; writing—review and editing, X.L., X.F., L.W. and J.W.; visualization, X.F.; supervision, L.W. and J.W.; project administration, X.L.; funding acquisition, X.L. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant numbers 61803138 and 61903114, the Natural Science Foundation of Anhui Provincial under grant number 2008085QF301 and Central Universities Basic Scientific Research Business Fund Special Funds under grant number JZ2021HGTB0076.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting reported results can be found at: https://calce.umd.edu/data.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

SOC	state of charge
PF	particle filter
PSO	particle swarm optimization
IPSO	improved particle swarm optimization
KF	Kalman filter
EKF	extended Kalman filter
UKF	unscented Kalman filter
RMSE	root mean squared error
MAE	mean absolute error
ME	max error

OCV	open circuit voltage
DST	dynamic street test
FUDS	federal urban driving schedule
US06	intense Driving Schedule
BJDST	Beijing dynamic stress test

References

1. Chu, S.; Majumda, A. Opportunities and challenges for a sustainable energy future. *Nature* **2012**, *488*, 294–303. [CrossRef] [PubMed]
2. Yang, S.; Ma, C. SOC Estimation Algorithm Based on Improved PNGV Model. *Automot. Eng.* **2015**, *37*, 582–586+598.
3. Gregory, L.P. Review and Some Perspectives on Different Methods to Estimate State of Charge of Lithium-Ion Batteries. *J. Automot. Saf. Energy* **2019**, *10*, 249–272.
4. Yang, N.; Zhang, X.; Li, G. State of charge estimation for pulse discharge of a LiFePO4 battery by a revised Ah counting. *Electrochim. Acta* **2015**, *151*, 63–71. [CrossRef]
5. Lu, R.; Wang, T.; Feng, F.; Zhu, C. SOC Estimation Based on the Model of Ni-MH Battery Dynamic Hysteresis Characteristic. *World Electr. Veh. J.* **2010**, *4*, 259–265. [CrossRef]
6. Huang, S.-J.; Huang, B.-G.; Pai, F.-S. An approach to measurements of electrical characteristics of lithium-ion battery with open-circuit voltage function. *IET Power Electron* **2012**, *5*, 1968–1975. [CrossRef]
7. Meng, J.; Stroe, D.-I.; Ricco, M.; Luo, G.; Swierczynski, M.; Teodorescu, R. A Novel Multiple Correction Approach for Fast Open Circuit Voltage Prediction of Lithium-ion Battery. *IEEE Trans. Energy Convers.* **2018**, *34*, 1115–1123. [CrossRef]
8. Lee, S.; Kim, J.; Lee, J.; Cho, B.H. State-of-charge and capacity estimation of lithium-ion battery using a new open-circuit voltage versus state-of-charge. *J. Power Sources* **2008**, *185*, 1367–1373. [CrossRef]
9. Guo, Y.; Zhao, Z.; Huang, L. SOC estimation of Lithium battery based on improved BP neural network. *Energy Procedia* **2017**, *105*, 4153–4158. [CrossRef]
10. Guo, N.; Fang, Y.; Tian, Z.; Cao, S. Research on SOC fuzzy weighted algorithm based on GA-BP neural network and ampere integral method. *J. Eng.* **2019**, *19*, 576–580. [CrossRef]
11. Cheng, B.; Bai, Z.; Cao, B. State of charge estimation based on evolutionary neural network. *Energy Convers. Manag.* **2008**, *49*, 2788–2794.
12. Liu, X.; Li, K.; Wu, J.; He, Y.; Liu, X. State of Charge Estimation for Traction Battery Based on EKF-SVM Algorithm. *Automot. Eng.* **2020**, *42*, 1522–1528+1544.
13. Zhang, L.; Li, K.; Du, D.; Zhu, C.; Zheng, M. A sparse least squares support vector machine used for SOC estimation of Li-ion Batteries. *IFAC-Pap.* **2019**, *52*, 256–261.
14. Qiu, Y.; Liu, X.; Chen, W.; Wei, D.; Duan, Z.M. Vanadium redox battery SOC estimation based on RLS and EKF Algorithm. *Control. Decis.* **2018**, *33*, 37–44.
15. Tan, F.; Zhao, J.; Wang, Q. A Novel Robust UKF Algorithm for SOC Estimation of Traction Battery. *Automot. Eng.* **2019**, *41*, 944–952.
16. Liu, X.; Li, H.; Wei, Z.; He, Y.; Zeng, G. CKF estimation Li-ion battery SOC based on Drift-Ah integral method. *Control. Decis.* **2019**, *34*, 535–541.
17. Li, W.; Luo, M.; Tan, Y.; Cui, X. Online Parameters Identification and State of Charge Estimation for Lithium-Ion Battery Using Adaptive Cubature Kalman Filter. *World Electr. Veh. J.* **2021**, *12*, 123. [CrossRef]
18. Tang, A.; Gong, P.; Li, J.; Zhang, K.; Zhou, Y.; Zhang, Z. A State-of-Charge Estimation Method Based on Multi-Algorithm Fusion. *World Electr. Veh. J.* **2022**, *13*, 70. [CrossRef]
19. Ye, M.; Guo, H.; Xiong, R.; Yang, R. Model-based State-of-charge Estimation Approach of the Lithium-ion Battery Using an Improved Adaptive Particle Filter. *Energy Procedia* **2016**, *103*, 394–399. [CrossRef]
20. Zuo, J.; Zhang, Y.-Z.; Liang, Y. Particle Filter Based on Adaptive Part Resampling. *Acta Autom. Sin.* **2012**, *38*, 647–652. [CrossRef]
21. Wang, F.; Lu, M.; Zhao, Q.; Yuan, Z. Particle Filtering Algorithm. *Chin. J. Comput.* **2014**, *37*, 1679–1694.
22. Wu, X.; Huang, G.; Gao, J. Multiple-model probability hypothesis density filter for multi-target tracking without the statistic. *Control. Decis.* **2014**, *29*, 475–480.
23. Li, X.; Li, Y.; Shang, J.; Dai, Q. Performance Analysis of Underwater Acoustic Signal for Non-Gaussian System Using Particle Filter Algorithm. *Fire Control. Command. Control.* **2014**, *39*, 34–37+41.
24. Shao, S.; Bi, J.; Yang, F.; Guan, W. On-line estimation of state-of-charge of Li-ion batteries in the electric vehicle using the resampling particle filter. *Transp. Res. Part D Transp. Environ.* **2013**, *32*, 207–217. [CrossRef]
25. Liu, X.; Chen, Z.; Zhang, C.; Wu, J. A novel temperature-compensated model for power Li-ion batteries with dual-particle-filter state of charge estimation. *Appl. Energy* **2014**, *123*, 263–272. [CrossRef]
26. Zhao, Y.; Zhou, X.; Liu, Y. SOC Estimation for Li-Ion Battery Based on Extended Kaman Particle Filter. *China Mech. Eng.* **2015**, *26*, 394–397.
27. Wu, T.; Liu, K.; Du, X. SOC estimation of lithium-ion battery based on UKPF algorithm. *Chin. J. Power Sources* **2021**, *45*, 602–605+625.

28. Bi, J.; Zhang, D.; Chang, H.-T.; Shao, S. Estimation for SOC of PEV Battery Based on Artificial Immune Particle Filter. *J. Transp. Syst. Eng. Inf. Technol.* **2015**, *15*, 103–108.
29. Ahmed, M.S.; Raihan, S.A.; Balasingam, B. A scaling approach for improved state of charge representation in rechargeable batteries. *Appl. Energy* **2020**, *267*, 114880. [CrossRef]
30. Mao, Q.; Zhu, Q.; Xu, Z.J.; Xu, S.F. Parameter Identification of Battery Model Based on the Particle Swarm Optimization. *Electr. Eng.* **2021**, *12*, 156–157.
31. Zhang, Y.; Wang, L.; Zhou, H.; Zhao, H. Design and Applications of Particle Swarm Optimization Based on Competitive Learning. *Comput. Meas. Control.* **2021**, *29*, 182–189.
32. Zheng, F.; Xing, Y.; Jiang, J.; Sun, B.; Kim, J.; Pecht, M. Influence of different open circuit voltage tests on state of charge online estimation for lithium-ion batteries. *Appl. Energy* **2016**, *183*, 513–525. [CrossRef]