Particle swarm optimization particle filter denoising algorithm with mutation operator

H X Chen¹,², L Fang¹, D L Fan¹, W J Huang¹, G Y Zhang¹, C H Cao¹, J M Huang¹ and L Yang¹

¹School of Mechanical and Electrical Engineering, Wuhan Institute of Technology, Wuhan 430205, China
E-mail: 449392297@qq.com

Abstract. In this paper, a novel particle filter (NPSO-PF) algorithm is proposed, which is called particle cluster optimization particle filter algorithm with mutation operator, which is used for real-time filtering and noise reduction of nonlinear vibration signals. Because of its introduction of mutation operator, this algorithm has no problem that particle swarm optimization (PSO) algorithm is easy to fall into local optimal value and the calculation accuracy is not high. At the same time, through the mutation operation, the distribution and diversity of particles in the sampling process are improved, and the particle filter (PF) algorithm is solved in which the particles are poor and the utilization rate is not high. The mutation control function makes the particle set optimization process in the early and late stages, and improves the convergence speed of the particle set, which greatly reduces the running time of the whole algorithm. Simulation experiments show that compared with PF algorithm and PSO-PF algorithm, the proposed new particle swarm optimization particle filter (NPSO-PF) algorithm has lower root mean square error, shorter running time, higher signal to noise ratio and more stable filtering performance. It is proved that the algorithm is suitable for real-time filtering and noise reduction processing of nonlinear signals.

1. Introduction

Denoising is an important part of signal processing. The effect of denoising directly affects the accuracy of subsequent analysis results. The denoising method for studying nonlinear signals is the focus of attention in recent years. Ananth et al [1] used a PSO based blur filter to remove high density image impulse noise from the signal. These algorithms can achieve the denoising of nonlinear signals. However, using the PSO algorithm structure, it is easy to fall into the local optimal solution. Processing of signals with multiple local extrema, the convergence speed and accuracy are not high, the convergence speed and accuracy are not high. In order to improve the defects of the PSO algorithm, Dai J et al introduced the rough set theory into the PSO algorithm in [2], and constructed a discrete PSO method to enhance the search ability. Kar et al [3] proposed a particle swarm optimization algorithm based on gravitational search algorithm (GSA), which is used for the optimization design of two common analog circuits and achieved good results.

The unique advantage of PF algorithm in dealing with parameter estimation and state filtering of nonlinear non-Gaussian time-varying systems, which makes the PF method attract attention at home and abroad in recent years. However, the conventional PF algorithm adopts the suboptimal importance function in the resampling process, which makes the conventional particle filtering method have the problems of low particle and low computational efficiency. The intelligent optimization algorithm is
integrated into particle filter, which becomes the hotspot of particle filter at present. The particle filter algorithm based on particle swarm optimization algorithm is one of the representatives of intelligent particle filter algorithm. In [4], Ramazan et al proposed a dual estimation method for joint parameters and state estimation based on edge particle filter and particle swarm optimization, which improves the performance of PF algorithm. However, the algorithm tends to be locally optimal in the middle and late stages, and the operation results are unstable and the denoising effect is poor. In order to improve the performance of the PSO-PF algorithm, Tao et al [5] introduced chaotic sequences into the PSO-PF algorithm, and the literature [6] and [7] combined the genetic resampling method with the PSO-PF algorithm. These measures all improve the filtering performance of the PSO-PF algorithm. However, as the structure of the algorithm becomes more complicated, the amount of calculation will increase, which is not suitable for real-time denoising processing.

Inspired by the above, this paper proposes a NPSO-PF algorithm for filtering noise reduction. The algorithm passes the genetic variation control function after sampling, so that the particle optimization process is divided into the front and the end, which accelerates the convergence of the particle set and improves the operation speed. At the same time, through the mutation exercise, the particle diversity is controlled, and the problem of low particle consumption and low utilization rate is avoided. The algorithm is used for noise reduction of nonlinear signals with the advantages of stability, speed and effectiveness. This paper first introduces the principles of PF, PSO and PSO-PF algorithms and analyzes their defects, and then constructs the NPSO-PF algorithm. In order to verify the filtering performance of the NPSO-PF algorithm, the state estimation experiment was carried out. Finally, the denoising simulation experiment was carried out to verify the filtering noise reduction of the NPSO-PF algorithm.

2. Conventional particle filter algorithm
The essence of the PF algorithm is to use the Monte Carlo method to realize the integral operation of the optimal Bayesian estimation. The sample particles are used to simulate the current state of a given system and determine the particle samples corresponding to the system in different states at different times. Thereby a posterior probability distribution approximated to the state of the system is obtained. The key is that the weight of the core particles can be constantly corrected and adjusted, and finally an accurate estimate is obtained [8]. The state equation and observation equation of nonlinear dynamic system are selected as follows:

\[
\begin{aligned}
  x_k &= f(x_{k-1}) + v_{k-1} \\
  y_k &= h(x_k) + n_k
\end{aligned}
\]  

Where \(x_k\) is the system state vector at \(k\) moment, \(y_k\) is the observation output. And the \(v_{k-1}\) is the system noise. The \(n_k\) is the observation noise. The \(f(\cdot)\) and \(h(\cdot)\) are non-linear functions respectively.

The implementation steps of particle filter are as follows:

Step 1: Initialization. At \(k=0\) moment, \(\{x_0^i, i=1,2,...,N\}\) is obtained by sampling according to \(P(x_0)\) distribution.

Step 2: Sampling
- For \(i=1,2,...,N\). Sample \(x_k^i\) from the proposal distribution \(q(x_k^i|x_{k-1}^i,z_k^i)\),
- Evaluate the importance weights

\[
\omega_k^i = \omega_{k-1}^i \frac{P(z_k|x_k^i)P(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i,z_k^i)}, i=1,2,...,N
\]  

- Normalize the weights

\[
\omega_k = \frac{\omega_k^i}{\sum_{i=1}^{N} \omega_k^i}
\]
\[
\tilde{w}_k^i = w_k^i \cdot \left( \sum_{i=1}^{N} w_k^i \right)^{-1}, i = 1, 2, 3, \ldots, N
\] (3)

Step 3: Resampling. The set \( \{x_0^i, i = 1, 2, \ldots, N\} \) of new particles at \( k \) time is obtained by resampling from set \( \{x_0^i, i = 1, 2, \ldots, N\} \) according to the importance weight \( \tilde{\omega}_k^i \), and the weight of particles is redistributed to \( \omega_k^i = \tilde{\omega}_k^i = 1/N \).

Step 4: Output. State estimation
\[
\bar{x}_k = \sum_{i=1}^{N} \omega_k^i \tilde{x}_k^i
\] (4)

3. PSO-PF algorithm

3.1. The principle of PSO algorithm
The PSO algorithm is a kind of optimization algorithm for simulating swarm intelligence behavior proposed by Kennedy and Eberhart in 1995. It is an effective global optimization algorithm by optimizing the search through the group intelligence guidance generated by the cooperation and competition of particles in the group [9]. Specifically, it can be expressed as: randomly initialize a particle group (the quantity is \( m \) ) in an N-dimensional search space. The particle \( i \) position is \( X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \), its velocity is \( V_i = (v_{i1}, v_{i2}, \ldots, v_{in}) \), its individual extreme value \( P_i = (p_{i1}, p_{i2}, \ldots, p_{in}) \), and the population's global extreme value is \( G_i = (g_{i1}, g_{i2}, \ldots, g_{in}) \). After determining the above two extremes, you can update the speed and position of the particle according to the following equation:

\[
\begin{align*}
V_{i}^{k+1} &= \omega \cdot V_{i}^{k} + c_1 \cdot \text{rand}() \cdot (P_{i}^{k} - X_{i}^{k}) + c_2 \cdot \text{rand}() \cdot (G_{i}^{k} - X_{i}^{k}) \\
X_{i}^{k+1} &= X_{i}^{k} + V_{i}^{k+1}
\end{align*}
\] (5)

Where: \( \text{rand}() \) is a random number between interval \((0,1)\), and \( c_1, c_2 \) are collectively referred to as a learning factor or an acceleration factor, generally \( c_1 = c_2 = 1.4962 \). \( \omega \) is the inertia coefficient.

3.2. PSO-PF algorithm
The PSO algorithm is integrated into the PF algorithm by first introducing the latest observations into the sampling process of the particle samples and defining the fitness function in the PSO algorithm as:

\[
\text{fit} = \exp \left\{ -\frac{1}{2R_i} \left( Y_{\text{new}} - Y_{\text{pred}} \right)^2 \right\}
\] (6)

Where \( R_i \) is the observed noise variance, \( Y_{\text{new}} \) is the latest observation, and \( Y_{\text{pred}} \) is the predicted observation.

During the sampling process, if the particle sets are distributed near the real state, the fitness of the particles will be high. Particles that are not in the real state will have a low fitness value. At this point, the PSO algorithm optimization algorithm is used to continuously update the velocity and position of each particle according to the optimal value and equation (5), so that the particles are constantly approaching the real state. Thereby, the omission of important particles is avoided, which not only ensures the diversity of the particles, but also improves the utilization rate of the effective particles.

4. PSO-PF algorithm with variation operator (NPSO-PF)
4.1. PSO algorithm with mutation operator

The conventional PSO algorithm is prone to fall into the local optimal value in the early iteration. In order to improve this defect, in view of the variation idea of the genetic mutation algorithm, the genetic algorithm operation operator [10] is introduced into the PSO algorithm to keep the diversity within the population. Furthermore, the algorithm avoids premature convergence to the local optimal value. The specific method is to introduce a mutation operator control function to control the mutation rate of the mutation operator. The mutation control function is shown in equation (7):

\[ y(\text{epoch}) = (1 - (\text{epoch}/\text{epoch}_{\text{max}})^{\alpha})^{\beta} \]  

(7)

Where \(\text{epoch}\) represents the current number of iterations, \(\text{epoch}_{\text{max}}\) represents the maximum number of iterations, \(\alpha, \beta\) represents the control coefficient.

According to equation (7), the control rate of the mutation operator is:

\[ u = m \cdot y(\text{epoch}) \]  

(8)

Where \(u\) is the mutation rate of the mutation operator, and \(m\) is the preset mutation rate, which is unchanged after setting.

According to the mutation rate, the number of particles undergoing the mutation operation is:

\[ M = \left\lfloor N \cdot u \right\rfloor \]  

(9)

\(M\) particles are randomly selected for mutation operation, and real numbers are used. Considering that particle \(k\) is selected for mutation operation, and element \(j\) of the \(X_k = (x_{k1}, x_{k2}, \ldots, x_{kd})\) is mutated, the operation strategy is:

\[ x_{kj} = x_{kj} + \text{rand} \cdot y(\text{epoch}) \]  

(10)

Where \(\text{rand} \in (-a, a)\) is a random number.

In this paper, the dynamic decreasing inertia coefficient is used, and the dynamic inertia coefficient is calculated as:

\[ w = w_{\text{min}} + (w_{\text{max}} - w_{\text{min}}) \cdot y(\text{epoch}) \]  

(11)

Where \(w_{\text{max}}\) represents the inertia coefficient and \(w_{\text{min}}\) represents the minimum inertia coefficient.

The mutation operator controls the operation of the PSO algorithm through the particle's mutation rate, dynamic inertia coefficient and mutation operation. The mutation rate is determined by equation (7), the dynamic inertia coefficient is determined by equation (11), and the mutation operation is performed by equations (8)-(10), which are all related to the number of iterations of the algorithm. In the early stages of algorithm iteration, the control variation rate and the dynamic inertia coefficient take a large value, and the variation operation of the particles is large, which maintains the diversity of the internal particles of the population, and makes the PSO algorithm search in a large range of the solution space. In the later stage of the algorithm, the control variation rate and the dynamic inertia coefficient take a small value, and the variation operation of the particles is very small. At this time, the particles inside the population can quickly become uniform, making the PSO algorithm within a small range of the solution space. Search. Thereby, the convergence speed and convergence precision of the algorithm are improved. The specific implementation steps of the NPSO algorithm are as follows:

Step 1: Initialize group size \(N\), learning factors \(C_1\) and \(C_2\), inertia coefficient \(w\), position \(X_i\) and velocity \(V_i\) of each particle, maximum iteration number \(\text{epoch}_{\text{max}}\), preset mutation rate \(m\).

Step 2: The control function \(y(\text{epoch})\) is calculated according to equation (7), and the inertia coefficient \(w\) is calculated according to equation (11).
Step 3: Calculate the mutation rate $y(\text{epoch})$ according to the equation (8), and randomly select the $M$ particles to perform the mutation operation according to the equation (10).

Step 4: For each particle, the fitness solution $f_{it}(i)$ is expressed according to equation (6) and compared with its individual optimal value $P_i$, if $f_{it}(i) > P_i$ then $P_i = f_{it}(i)$.

Step 5: Find the global optimal value $G_i$, execution procedures such as if $P_i > G_i$ then $P_i = G_i$.

Step 6: Update the velocity $V_i$ and the position $X_i$ of the particle according to equation (5).

Step 7: When the number of iterations or the optimal value of the particle group meets a certain threshold $\varepsilon$, the result is output, otherwise it returns to the second step.

4.2. NPSO-PF algorithm
The principle of the NPSO-PF algorithm is to optimize the sampling process of particle filter by using the fast and accurate convergence performance of the PSO algorithm with mutation operator. The PSO algorithm with mutation operator is used to calculate the fitness to move all particles to the optimal particle. Thereby improving the distribution of the particle set, so that the particle set is in the vicinity of the real state, thereby improving the precision and efficiency of the particle filtering, the specific steps are:

![Flowchart of NPSO-PF algorithm](image)

**Figure 1.** The flowchart of NPSO-PF algorithm.

Step 1: initialization. At time $k = 0$, $\{x_i^0, i = 1, 2, ..., N\}$ is obtained from the $P(x_0)$ distribution sample.

Step 2: NPSO optimizes the particle set distribution. Combine the NPSO implementation steps described in the Section 4.1 to optimize the particle swarm. When the number of iterations is reached or the optimal value of the particle swarm meets a certain threshold $\varepsilon$, the particle is output to obtain a new particle sample $\hat{x}_i^k$.

Step 3: Importance weight calculation. Sampling $\{\hat{x}_i^k, i = 1, 2, ..., N\} - q(x_k|x_{i-1}, y_k)$, calculating a new importance weight according to equation (2), and performing weight normalization according to equation (3).

Step 4: Resampling. Resampling from set $\{\hat{x}_i^k, i = 1, 2, ..., N\}$ based on new importance weights, the
process is consistent with traditional particle filtering algorithms.

Step 5: Output. The updated state estimate is obtained according to equation (4).

The general steps of filtering and noise reduction are shown in figure 1.

5. Test simulation and result analysis

5.1. Particle filter simulation test and comparison

The state equation of the dynamic non-linear system selected by the test is:

\[ x_k = 0.5 x_{k-1} + \frac{25 x_{k-1}}{1 + x_{k-1}^2} + 8 \cos(1.2(k-1)) + v_{k-1} \]  

(12)

The observation equation is:

\[ y_k = \frac{x_k^2}{20} + n_k \]  

(13)

Where \( v_{k-1} \) and \( n_k \) are zero mean Gaussian noise. The PF algorithm, PSO-PF algorithm and NPSO-PF algorithm are used to estimate and track the state of the nonlinear system respectively. To compare the performance of the three algorithms, N=50, N=500 and N=1000 respectively. Three different kinds of particle numbers are selected, taking the initial state probability density function as \( N(0,2) \), measuring noise variance and process noise variance are \( R=1\times10^{-3} \) and \( Q=1\times10^{-2} \), respectively, and the parameters of the PSO algorithm are the same as above. The Root of Mean Square Error (RMSE) is calculated as follows:

\[ RMSE = \left[ \frac{1}{T} \sum_{k=1}^{T} (x_k - \hat{x}_k)^2 \right]^{1/2} \]  

(14)

Where \( T \) is the number of samples, \( x_k \) is the true value of the sample, and \( \hat{x}_k \) is the average value of the sample.

In order to compare the overall performance of the three algorithms, the simulation experiment was carried out 30 times, and the average of 30 root mean square errors and program running time of the three algorithms under three different particle numbers were measured, as shown in tables 1 and 2:

| Table 1. RMSE of three algorithms. |
|-----------------------------------|
| Algorithm | Mean of Root Mean Square Errors |
|          | N=100  | N=500  | N=1000 |
| PF       | 0.2627 | 0.1101 | 0.0974 |
| PSO-PF   | 0.1547 | 0.0966 | 0.0800 |
| NPSO-PF  | 0.0797 | 0.0572 | 0.0561 |

| Table 2. Run time of three different algorithms. |
|-----------------------------------------------|
| Algorithm | Program Running Time (S) |
|          | N=100  | N=500  | N=1000 |
| PF       | 0.8608 | 0.8930 | 1.0748 |
| PSO-PF   | 0.7597 | 1.0519 | 1.1111 |
| NPSO-PF  | 0.7744 | 0.7904 | 0.9247 |
From the results of tables 1 and 2, the average estimation error and running time of the NSPO-PF algorithm are lower than the other two. With the increase of the number of particles, the effect is more obvious. The PSO-PF algorithm is only slightly better than the PF algorithm, and the performance is basically the same as the number of particles increases. In summary, the NPSO-PF algorithm has better and more stable estimation performance than the PSO-PF and PF algorithms.

5.2. Noise reduction experiment simulation

Using the above three methods to perform noise reduction simulation experiments on the following signals, select an original signal equation as follows:

\[
Y(t) = 0.01\cos(800\pi t - 5) e^{-0.5t} + 0.003\cos(1000\pi t - 5) e^{-0.5t} + 0.01\cos(1200\pi t - 5) e^{-0.5t} + 0.005\cos(1400\pi t - 5) e^{-0.5t} + 0.002\cos(1600\pi t - 5) e^{-0.5t}
\]

(15)

Add a random noise signal of the same size as the original signal:

\[
Z(t) = Y(t) + 0.005 R(t)
\]

(16)

In the above equation, the original signal equation (15) and the noisy signal equation (16) are used to replace the state equation (12) and the observation equation (13) in the upper section, respectively, to form a spatial state model among the three algorithms. The initial probability density function is \(N(0, 2)\), the initial state value is 1, the measurement noise variance \(R = 1 \times 10^{-4}\), the process noise variance \(Q = 1 \times 10^{-5}\), and the PSO algorithm related parameters are the same as above, and the three kinds of particles in the upper section are experimentally simulated respectively.

The signal-to-noise ratio is the ratio of the original signal to the pure noise signal. The larger the signal-to-noise ratio is, the smaller the noise is. Therefore, the signal-to-noise ratio is an important indicator to measure the noise reduction effect. In order to better compare the noise reduction performance of the three algorithms, the signal to noise ratio (SNR) analysis is performed on the above experimental results. Taking the 50 noise reduction experiments, the three algorithms analyze the signal-to-noise ratio mean values of the three particle numbers. The results are shown in table 3. The SNR calculation formula is:

\[
SNR = 10\log\left(\frac{\sum x^2}{\sum (z - x)^2}\right)
\]

(17)

| Particle Number | SNR before filtering | SNR after filtering | Improved SNR Value |
|-----------------|----------------------|---------------------|--------------------|
| N=100           | -0.3200              | 1.0314              | 1.3514             |
|                 |                      | 1.8626              | 2.1826             | 3.3532             |
| N=500           | -0.2256              | 4.2526              | 4.4782             |
|                 |                      | 4.3995              | 4.6251             | 6.2661             |
| N=1000          | -0.2427              | 4.3535              | 4.5962             |
|                 |                      | 4.4469              | 4.6896             | 7.0104             |

It can be seen from table 3 that the signal-to-noise ratio filtered by the NPSO-PF algorithm is higher than the other two algorithms under the test of the three particle numbers, and the signal-to-noise ratio after filtering by PSO-PF and PF algorithm is basically the same. It shows that the noise reduction performance of NPSO-PF algorithm is better than the other two algorithms, and the noise reduction performance of PSO-PF and PF algorithm is basically the same. The improved value of signal-to-noise ratio (SNR) of NPSO-PF algorithm is generally higher than that of the other two algorithms, and it continues to rise with the increase of the number of particles, while the PSO-PF and PSO algorithms are basically saturated. In summary, the NPSO-PF algorithm has better and more stable filtering noise reduction performance.
6. Conclusions
In this paper, the mutation operator is introduced into the PSO algorithm, which improves the structure of the PSO algorithm, optimizes the problem that the traditional PSO algorithm is easy to fall into the local optimal solution, and improves the performance of the PSO algorithm. Combining it with particle filter algorithm constitutes NPSO-PF algorithm, which not only has better estimation performance, but also has better effect in filtering noise reduction. The simulation results show that the NPSO-PF algorithm is suitable for real-time filtering and noise reduction of nonlinear signals.

Acknowledgment
The experimental data is provided by the Reliability Research Lab in the Department of Mechanical Engineering at the University of Alberta in Canada. This work was supported by the Special Major Project of the Ministry of Science and Technology of Hubei Province of China (Grant No. 2016AAA056), Major project of Hubei Provincial Department of Education (Z20101501), Research foundation for the returned Overseas Chinese Scholars of State Education Ministry (20091001) and the National Natural Science Foundation of China (Grant 51775390).

References
[1] Ananth C, Vivek T, Selvakumar S et al 2017 Impulse Noise Removal Using Improved Particle Swarm Optimization (Social Science Electronic Publishing Sarasota Florida USA)
[2] Dai J, Han H, Hu Q et al 2016 Discrete particle swarm optimization approach for cost sensitive attribute reduction knowl. based syst. 102 C 116-26
[3] Mallick S, Kar R, Mandal D et al 2015 Optimal sizing of CMOS analog circuits using gravitational search algorithm with particle swarm optimization Int. J. Machine Learning and Cybernetics 8 1
[4] Havangi and Ramazan 2018 Joint parameter and state estimation based on marginal particle filter and particle swarm optimization Circ. Syst. Signal Pr. 38(8) 3558-75
[5] Ershen W, Tao P, Pingping Q U et al 2016 Improved particle filter algorithm based on chaos particle swarm optimization J. Beijing University of Aeronautics & Astronautics 42(5) 885-90
[6] Xue H, Qifeng C, Tingting Z et al 2016 Improved particle swarm optimization based on re-sampling of particle filter and mutation J. Comput. Applications 36(4)1008-14
[7] Haibo L, Jingjing K E, Yi Z and Rao-Blackwellized 2016 Particle filter algorithm combined particle swarm optimization and genetic re-sampling Comput. Eng.42(11) 295-9
[8] Fang Z, Geng G F and Xu X H 2007 Particle swarm optimization particle filter J. Contr. Decision 22(3) 273-7
[9] Li Y, Bai B D and Zhang Y K 2010 Improved particle swarm optimization algorithm for Fuzzy multi-class SVM J. Systems Eng. and Electronics 21 509-13
[10] Yu R B, Zhao X P and Meng F L 2016 A PSO algorithm with mutation operator Ship Electronics Eng. 36 26-9