State Estimation using Bacterial Foraging Algorithm Based Particle Filter for a Three Tank System

Mr. J. Joseph Ignatious, Research Scholar and Dr. S. Abraham Lincon

Department of Electronics and Instrumentation Engineering, Annamalai University, Annamalai Nagar, India.

ARTICLE INFO
Article history:
Received 25 April 2014
Received in revised form 8 May 2014
Accepted 20 May 2014
Available online 17 June 2014

Keywords:
Kalman Filter (KF), Particle filter (PF), Particle swarm optimization particle filter (PPF), Bacterial Foraging Particle filter (BPF).

ABSTRACT
Recursive state estimation of constrained nonlinear dynamical system has attracted the attention of many researchers in recent years. For nonlinear/non-Gaussian state estimation problems, Particle Filters have been widely used (Arlampalam et al. (2002)). As pointed out by Daum (2005), particle filters require a proposal distribution and the choice of proposal distribution is the key design issue. In this paper, a novel approach for generating the proposal distribution based on a Particle Swarm Optimization Particle Filter (PPF), Bacterial Foraging Particle Filter (BPF) has been proposed. The efficacy of the proposed state estimation algorithms using a particle filter is illustrated via a successful implementation on a simulated three tank system, involving constraints on estimated state variables, which involves constraints on the process noise (Rao et al. (2006)). The efficacy of the proposed state estimation scheme is demonstrated by conducting simulation studies on a Three Tank System. The simulation studies underline the crucial role played by the choice of proposal distribution in formulation of particle filters.

© 2014 AENSI Publisher All rights reserved

To Cite This Article: Mr. J. Joseph Ignatious and Dr. S. Abraham Lincon., State Estimation using Bacterial Foraging Algorithm Based Particle Filter for a Three Tank System. Aust. J. Basic & Appl. Sci., 8(10): 655-663, 2014

INTRODUCTION

The particle filter (PF) is a sequential Monte Carlo method which provides a general framework for numerical (simulation based) solution of the stochastic Bayes filter specified by (S. Arulampalam et al 2002). Particle filters are attractive because, with sufficiently large samples, they approach the optimal estimate of the posterior PDF. Their drawback is the computational cost, which fortunately, with ever faster computer hardware, is becoming less relevant. Particle filters have become enormously popular and con- sequently several books and tutorials (X. Shao et al 2010) have been devoted for them. A brief overview of particle filters is also given in chapter 2. The standard Bayes stochastic filter, typically implemented as a particle filter, despite its popularity and numerous applications in various fields of human endeavor (e.g., ecology, economics, robotics, navigation) is very restrictive. Here are some examples highlighting the limitations of the formulation of the Bayes filter (Doucette, A et al 2000) and its particle filter implementations. The formulation does not cover situations where the dynamic system switches (in some probabilistic manner) on and off. Switching is very common; for example: moving objects enter and leave a surveillance area; an outbreak of an epidemic occurs at some point of time and eventually disappears. The formulation above is also restricted to a single dynamic system or object. In many applications, however, multiple dynamic objects can appear and disappear during the observation period. Furthermore, the measurement model is very restrictive, because it assumes perfect detection (no false detections and no miss detections). The measurement model also assumes that the measurement functions Zk are precise (i.e., points in the measurement space). There are many applications where these two assumptions are not valid. The measurement function often features parameters which are not known precisely (e.g., sensor location, orientation, gain, environmental coefficients such as propagation loss, etc). Similarly, the measurements coming from some sources of information (e.g., natural language statements, sensors with bounded measurement errors) are better modeled by intervals (crisp or fuzzy), than by points. The measurements which are imprecise or which result from imprecise measurement functions are referred to as non-standard measurements. The remaining section of this paper is organized as follows: Section II reviews Particle Filter and Section III gives our proposed distribution based on a Particle Swarm Optimization Particle Filter (PPF), Bacterial Foraging Particle Filter (BPF), Section IV provides a performance study of Three Tank System for our proposed schemes. Section V concludes this paper.

Corresponding Author: Mr. J. Joseph Ignatious, Research Scholar, Department of Electronics and Instrumentation Engineering, Annamalai University, Annamalai Nagar, India.
E-mail: jjignatious@gmail.com
**Particle filter:**

Particle filters are used in non-linear problems where the interest is in detection of dynamic signals. The signals are described by a system of equations that model their evolution with time, and the equations usually have the form

\[ x_n = f_n(x_{n-1}, u_n) \]  
\[ z_n = g_n(x_n, u_n) \]

Where \( n \in \mathbb{N} \) is discrete time index, \( x_n \in \mathbb{R}^{d_x} \) is a signal and \( z_n \in \mathbb{R}^{d_z} \) is a vector of observations. The symbols \( u_n \in \mathbb{R}^{d_u} \) and \( v_n \in \mathbb{R}^{d_v} \) are belong to noise vectors, \( f_n : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \rightarrow \mathbb{R}^{d_x} \) is signal transition function, and \( g_n : \mathbb{R}^{d_x} \times \mathbb{R}^{d_v} \rightarrow \mathbb{R}^{d_z} \) is a measurement function. The analytical forms of \( f_n(.) \) and \( g_n(.) \) are assumed as known. The densities of \( u_n \) and \( v_n \) are parametric and known, but their parameters may be unknown, of \( u_n \) and \( v_n \) are independent of each other. The objectives are to estimate recursively the signal of \( x_n \), \( u_n \) from the observations of \( z_{1:n} \), where \( z_{1:n} = \{ z_1, z_2, \ldots, z_n \} \). The Bayesian inference process is achieved by

\[ p(x_n/z_{1:n}) \propto p(z_n/x_n) p(x_n/z_{1:n-1}) \]

Where the prior \( p(x_n/z_{1:n-1}) \) is the propagation of the previous posterior along the temporal axis. (\( x_n/z_{1:n-1} \)=\( p(x_n/z_{1:n-1})p(x_{n-1}/z_{1:n-1})dx_{n-1} \)

When the state transition and observation models are nonlinear and non-Gaussian, the above integration is intractable and one has to resort to numerical approximations such as particle filters. The basic idea of particle filter is to use a number of particles \( x^k, k=1,2,3,\ldots,K \) be samples drawn from a state space to approximate the posterior distribution. Thus the posterior can be formulated as \( p(x_n/z_{1:n}) \equiv \frac{1}{K} \sum_{k=1}^{K} \delta(x_n - x^k_n) \), where \( \delta(\cdot) \) is dirac function. Since it is usually impossible to sample from the true posterior, an easy-to-implement distribution, the so-called proposal distribution denoted by \( q(\cdot) \) is employed, hence

\[ w^k_n \propto q(x_n/x^k_{n-1}, z_{1:n}) \]

Following from Equation 5.

After the importance sampling step, a re-sampling step is adopted to ensure the efficiency of the particles’ evolution. To summarize, the detail process of particle filter is presented in Algorithm 1

**Algorithm 1:**

1. **Initialization:** for \( n = 1,2,\ldots,K \), sample \( x_0^{(n)} \sim p(x_0) \), \( w_0^{(n)} = 1/K \)
2. **For time steps** \( k = 1, 2, \ldots \)
3. **Importance Sampling:** for \( n = 1, \ldots, K \), draw samples from the importance proposal distribution as follows:

\[ x_n^{(k)} = q(x_n/x^k_{n-1}, z_{1:n}) \]

\[ w_n^{(k)} = w_{norm}^{(k)} \]

4. **Weight update:** evaluate the importance weights with Equation 5.

5. **Normalize the importance weights:**

\[ w_n = \frac{w_n^{(k)}}{\sum_{k=1}^{K} w_n^{(k)}} \]

6. **Output the statistics of the particles:** MMSE or MAP estimate.

7. **Resampling:** generate \( K \) new particles \( x_k, k=1,2,3,\ldots,K \), according to important weights

8. **Repeat steps 3 to 6**

The proposal distribution \( q(.) \) is critically important for a successful particle filter because it concerns putting the sampling particles in the useful areas where the posterior is significant. The state transition distribution \( p(x_n/x_{n-1}) \) is usually taken as the proposal distribution for its simplicity. However, this proposal distribution contains little information about the current observations, consequently resulting to an inefficient sampling.

As shown in Fig.1(a), when the transition model is situated in the tail of the observation distribution, then the weight of most particles are low, thereby leading to the sample impoverishment problem.
Fig. 1: An illustration of importance sampling (Left: sample from \( p(X_0, X_n) \)), right: after BPF iterations

Swarm Intelligence Based Particle Filter:

The proposal distribution in the non-linear and non-Gaussian cases it is difficult to sample from (Dasgupta, S et al 2009) \( p(x_n/x_{n-1}, z_n) \) and to evaluate \( p(z_n/x_{n-1}, z_n = \int p(z_n/x_n)p(x_n/x_{n-1}) \, dx_n \) since minimizing the variance of the importance of weights is optimal in case of importance proposal distribution it is impossible to use as proposal distribution. The selection of a suitable form of importance function to represent the true posterior density is a crucial step in the particle filter (X. Shao et al 2009). The first and the most important step in the development of PF is the selection of proposal density function. When we compare two popular choices of proposal densities, Particle Swarm Optimization (PSO) technique (Anguluri, R et al 2011), global optimization technique developed with the inspiration of social activities in flock of birds and school of fish, and is widely applied in various engineering problems due to its high computational efficiency. Compared with other population-based stochastic optimization methods, such as GA and ACO, the PSO (Latha, K et al 2012) has comparable or even superior search performance for many hard optimization problems, with faster and more stable convergence rates. It has been proved to be an effective optimum tool in estimation of processes. We can see that the PSO iterations can naturally take the observation into consideration, since the particles cooperate and evolve according to their fitness values which are updated by their corresponding observations. Inspired by this property of the PSO, we propose a Bacterial Foraging Algorithm based particle filter, in which the particles are firstly propagated by the state transition model, and then corporately evolve according to the PSO iterations. Bacterial Foraging (BF) algorithm (Rajinikanth, V et al 2012) is a new division of biologically inspired stochastic search technique based on mimicking the foraging (methods for locating, handling and ingesting food) behavior of Escherichia coli bacteria (Ali, A et al 2006). Due to its merits such as high computational efficiency, easy implementation and stable convergence, it is widely applied to solve a range of complex engineering optimization problems (Dasgupta, S et al 2009). To give a clear view, the flow chart of the swarm intelligence based Particle Filter (Rajinikanth, V et al 2012) and its Algorithm 2 is shown in Figure 2.

Algorithm 2:

1. Initialization: for \( n = 1, 2, \ldots, K \), sample \( x_0^{(n)} \sim p(x_0) \), \( w_0^{(n)} = 1/K \)

Step 1: Initialization

i. Number of bacteria \( (S) \) to be used for finding the minima.
ii. Number of parameters \( (N_p) \) to be optimized.
iii. Specifying the location of the initial set of bacterias.
iv. \( N_p \) is the number of chemo tactic steps taken by each bacterium before reproduction.
v. \( N_d \) is the maximum number of steps taken by each bacterium when it moves from low nutrient area to high nutrient area.
vi. \( N_r \) and \( N_e \) are the number of reproduction and elimination dispersal events.
vii. \( p_e \) is the probability of elimination and dispersal

viii. Random swim direction vector \( \Delta(i) \) and run length vector \( C(i) \).

Step 2: Iterative algorithm for optimization[15]

The algorithm begins with the calculation of objective value for the initial bacterial population inside the innermost chemo taxis loop. Any \( i^{th} \) bacterium at the \( j^{th} \) chemo tactic, \( k^{th} \) reproduction and \( l^{th} \) elimination stage is \( \theta(j, k, l) \) and its corresponding objective value is given by \( f(i, j, k, l) \). The algorithm works as follows:
1. Starting of the elimination-dispersal loop
2. Starting of the reproduction loop
3. Starting of the chemotaxis loop
   a) for \( i = 1, 2, \ldots, S \), \( J_i(i, j, k, l) \) is calculated
   b) \( J_1(i, j, k, l) \) is served as the \( J_{\text{last}} \) so as to compare with other \( J_1 \) values.
   c) Tumble: A random vector \( \Delta(i) \) is generated with each element \( \Delta(i) \), \( m = 1, 2, \ldots, M \), a random number on \([-1, 1]\).
   d) Move:
      \[
      \hat{\theta}(j+1,k,l) = \theta(j,k,l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta(i) \Delta(i)}}
      \]
      This results in a step size \( C(i) \) in the direction of tumble for \( i^\text{th} \) bacterium.
   e) Calculate \( J_i(i, j + 1, k, l) \)
   f) Swim
      Let \( n = 0 \) (counter for swim length)
      while \( n < N_s \)
      \[
      n = n + 1;
      \]
      if \( J_i(i, j + 1, k, l) < J_{\text{last}} \) then \( J_{\text{last}} = J_i(i, j + 1, k, l) \) and
      \[
      \hat{\theta}(j+1,k,l) = \theta(j,k,l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta(i) \Delta(i)}}
      \]
      this \( \hat{\theta}(j+1,k,l) \) is used to calculate new \( J_i(i, j + 1, k, l) \) else \( n = N_s \)
   g) Go to the next bacterium \( (i+1) \) till all the bacteria undergo chemotaxis.
4. If \( j < N_c \), go to step 3 and continue chemotaxis since the life of bacteria is not over else go to the reproduction stage.
5. Reproduction
   a) For the given \( k \) and \( l \), and for each \( i = 1, 2, \ldots, S \),
      \[
      J_{\text{health}} = \sum_{j=1}^{N_c} J_i(i, j, k, l)
      \]
      is assumed to be the health of \( i^\text{th} \) bacterium. The bacteria are stored according to ascending order of \( J_{\text{health}} \).
   b) The bacteria with the highest \( J_{\text{health}} \) values die and other bacteria with minimum values split and the copies that are made are placed at the same location as their parent.
6. If \( k < N_c \), go to step 2 to start the next generation in the chemotactic loop else go to step 7.
7. Elimination - dispersal: For \( i = 1, 2, \ldots, S \) a random number \( (\text{rand}) \) is generated and if \( \text{rand} \leq P_{\text{ed}} \), then that bacterium gets eliminated and dispersed to a new random location, else the bacterium remains at its original location.
8. If \( l < N_{\text{ed}} \) go to step 1 else stop.
   \[
   x_n^{(k)} - q(x_n/x_{n-1}^k, z_{1:n})
   \]
   Weight update: evaluate the importance weights
   **Step 3.** Normalize the importance weights:
   \[
   w_n^{(k)} = \frac{w_n^{(k)}}{\sum_{k=1}^{K} w_n^{(k)}}
   \]
   **Step 4.** Output the statistics of the particles: MMSE or MAP estimate.
   **Step 5.** Resampling: generate \( K \) new particles \( x^k, k=1,2,3,\ldots,K \), according to important weights
   **Step 6.** Repeat steps 1 to 5
Fig. 2: Flow chart for the proposed BPF algorithm.

For a population size $S$, the particles are randomly generated between the minimum and the maximum limits of the threshold values. The objective function values $J$ of the particles are evaluated and next step is updated. At the end of specified chemo tactic steps, the bacterium is evaluated and sorted in descending order of fitness. In the act of reproduction, the first half of bacteria is retained and duplicated while the other half is eliminated. Finally, bacteria are dispersed as per elimination and dispersal probability which helps fasten the process of optimization and output the optimal threshold values corresponding to the overall best bacterium (Ali, A et al 2010).

**System Description – Three Tank System:**

The three tank system considered for study is shown in figure 3. The inflow to tank 1 (fin 1) and inflow to tank 3 (fin 3) are considered to be the two inputs. The level of the tank 1 ($h_1$), level of the tank 2 ($h_2$) and level of the tank 3 ($h_3$) are the three outputs. The unmeasured inflow into the tank 2 (d2) is considered as a disturbance variable. The material balance for the three tank system yields the following equations. The steady state operating data of the three tank system is shown in Table 1.

$$\frac{dh_1}{dt} = \frac{\text{fin}_1 - C_1 \text{Sp} \sqrt{2g(h_1 - h_2)}}{A_T}$$  \hspace{1cm} (7)

$$\frac{dh_2}{dt} = \frac{C_1 \text{Sp} \sqrt{2g(h_1 - h_2)}}{A_T} - \frac{C_2 \text{Sp} \sqrt{2g(h_2 - h_3)}}{A_T}$$  \hspace{1cm} (8)

$$\frac{dh_3}{dt} = \frac{\text{fin}_3 + C_2 \text{Sp} \sqrt{2g(h_2 - h_3)}}{A_T} - \frac{C_3 \text{Sp} \sqrt{2gh_3}}{A_T}$$  \hspace{1cm} (9)

where $C_1, C_2, C_3 \rightarrow$ constant flow coefficient
The three tank system is a nonlinear system. Hence linearising it about a nominal operating point and finding the state model using Taylors series approximation we get

\[
A = \begin{bmatrix}
-0.00318407802 & 0.00318407802 & 0 \\
-0.00318407802 & -0.0522300767 & 0.0203892966 \\
0 & 0.0203892966 & -0.0363135901
\end{bmatrix}
\]  

(10)

\[
B = \begin{bmatrix}
64.935 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 64.935
\end{bmatrix}
\]  

(11)

\[
C = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]  

(12)

\[
D = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]  

(13)

with \( h_1, h_2, h_3 \) as states and outputs

The discretized model with \( ts = 15 \) seconds is given by
Simulation Results:

The state estimation for the three tank system was done using Kalman Filter, Particle Filter, Particle swarm optimization Particle Filter, Bacterial Foraging Particle filter. The state and bias of the three tank system are estimated using the following schemes.

Noise Magnitude

The nominal inflow to the tanks 1 and 3 are fin1, fin3 = 50 ml/sec.

The nominal heights of the tanks h1, h2, and h3 are

\[
h_1 = 0.3345 m
\]

\[
h_2 = 0.2835 m
\]

\[
h_3 = 0.2039 m
\]

Process noise

The process noise is selected as 0.5% of the nominal inflow value:

\[
w_1, w_3 = 0.25 \text{ ml/sec.i.e. } 1\sigma = 0.5\% \text{ of the nominal inflow}=0.25\text{ml/sec}
\]

Measurement noise

The measurement noise is selected as 0.25% of the nominal heights of the tanks i.e., 1\sigma = 0.25\% of the nominal heights of the tanks.

\[
v_1 = 8.3625 \times 10^{-3} m
\]

\[
v_2 = 7.0875 \times 10^{-3} m
\]

\[
v_3 = 5.0975 \times 10^{-3} m
\]

In real time the noises are random in nature which may vary from time to time. In order to get more precise estimation, ten different sets of random noise sequences are generated and the performance of the estimators are studied as the average value of the ten trials.

Figures 4 (a), 4(b), 4(c), 4(d) gives an illustration of the estimates generated from a single run of the different filters. Compared with other Nonlinear Filters, this algorithm is more robust to the outlier, where the observation is severely contaminated by the noise. Since the result of a single run is a random variable, the experiment is repeated with re-initialization to generate statistical averages such as mean and variance are shown in Table 2.

It is obvious that the average accuracy of our algorithm is better than KF, PF, PPF, and comparable to that of BPF. Meanwhile, we can see that BPF can achieve a much faster convergence rate than PSOPF. This is because the velocity part employed which carries little information, while the annealing part in our BPF iterations enhances the diversity of the particle set and its adaptive effect enables a fast convergence rate. In summary, the total performance of our algorithm prevails over that of other nonlinear filters.
Table 2: Performance Measurements of Nonlinear filter with Mean and Variance

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>KF</td>
<td>9.3030800005296281e-08</td>
<td>1.37735088799049e-07</td>
</tr>
<tr>
<td>PF</td>
<td>7.8030800005296277e-08</td>
<td>9.653991675170774e-08</td>
</tr>
<tr>
<td>PPF</td>
<td>1.7990080850865868e-07</td>
<td>9.63550101786649e-08</td>
</tr>
<tr>
<td>BPF</td>
<td>4.3030800005296277e-08</td>
<td>1.09012762610751e-07</td>
</tr>
</tbody>
</table>

![Fig. 4(a): Estimated h1, h2, h3 with KF](image1)

![Fig. 4(b): Estimated h1, h2, h3 with PF](image2)

![Fig. 4(c): Estimated h1, h2, h3 with PSO - KF](image3)

![Fig. 4(c): Estimated h1, h2, h3 with BFO-KF](image4)

It is obvious that the average accuracy of our algorithm is better than KF, PF, PPF, and comparable to that of BPF. Meanwhile, we can see that BPF can achieve a much faster convergence rate than PSOPF. This is because the velocity part employed which carries little information, while the annealing part in our BPF iterations enhances the diversity of the particle set and its adaptive effect enables a fast convergence rate. In summary, the total performance of our algorithm prevails over that of other nonlinear filters.

**Conclusion:**

In this paper, we propose a Bacterial foraging particle filter to overcome the sample impoverishment problem. Unlike the independent particles in the convectional particle filters, the particles in our algorithm cooperate each other and evolve according to the cognitive effect and social effect in analogy with the cooperative and social aspects of animal populations. We conduct a theoretical analysis in a Bayesian Filtering view, and find that this algorithm is essentially a convectional Particle Filter with a hierarchical importance sampling process which is guided by the bacterial foraging extracted from particle configuration.

**REFERENCES**


