Applied Soft Computing 49 (2016) 474–484

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Simplex filter: A novel heuristic filter for nonlinear systems state estimation

Hadi Nobahari*, Seid Miad Zandavi, Hamed Mohammadkarimi

Sharif University of Technology, 1458889694 Tehran, Iran

ARTICLE INFO

Article history:
Received 2 March 2015
Received in revised form 7 July 2016
Accepted 4 August 2016
Available online 18 August 2016

Keywords:
State estimation
Nonlinear system
Nelder-Mead simplex algorithm
Simplex filter
Heuristic filter

ABSTRACT

This paper introduces a new filter for nonlinear systems state estimation. The new filter formulates the state estimation problem as a stochastic dynamic optimization problem and utilizes a new stochastic method based on simplex technique to find and track the best estimation. The vertices of the simplex search the state space dynamically in a similar scheme to the optimization algorithm, known as Nelder-Mead simplex. The parameters of the proposed filter are tuned, using an information visualization technique to identify the optimal region of the parameters space. The visualization is performed using the concept of parallel coordinates. The proposed filter is applied to estimate the state of some nonlinear dynamic systems with noisy measurement and its performance is compared with other filters.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

In many engineering applications, it is needed to estimate the states of a dynamic system. Many systems are affected by stochastic inputs and model uncertainties. A state estimation problem is defined as follows: given the mathematical model of a dynamic system, it is desired to estimate the time-varying states using noisy measurements. Estimation problems are often categorized as prediction, filtering and smoothing, depending on the intended objectives and the available observations [1]. Here, the focus is on filtering. Filtering is the problem of estimating the current states of a dynamic system using a set of observations. Filters are usually categorized to recursive and batch filters [1,2]. In a batch filter, e.g. least square filter, the complete history of measurments is used to estimate unkown states. A recursive filter, in comparsion, has the ability to receive and process measurments sequentially. Recursive filters consist of two essential stages: prediction and update [2]. Kalman Filter (KF) [2], Extend Kalman Filter (EKF) [3], Unscented Kalman Filter (UKF) [4] and Particle Filter (PF) [5] are examples of recursive filters.

Recursive filters can also be categorized to linear and nonlinear filters. In a linear filter, such as KF, both system and measurement models are linear. A classical nonlinear filter is EKF, the equations of which are the same as KF by linearizing the nonlinear equations of the system based on first-order Taylor expansion at the predicted points. There have also been other attempts to propagate filtering densities, an example of which is Gaussian Sum Filter (GSF) [6], where the posterior distributions are approximated as finite Gaussian Mixture (GM). The computational requirement of GSF can be significantly greater than that of EKF because GSF is obtained as the convex combination of several KFs. The performance of GSF degrades when it is applied to highly nonlinear systems [6]. Other methods evaluate the required densities over grids [7–11], but they are computationally expensive especially for high dimensions.

Analytical approximation and states sampling are two common approaches in nonlinear filtering. In the first approach, the nonlinear functions of the mathematical model are linearized and then a linear filter such as KF is used. EKF is a filter that works based on analytical approximation. Unlike to EKF, UKF is a sample based filter [2], and approximates a Gaussian distribution by a set of deterministically chosen samples [4].

Sample based filters are categorized to mathematical and heuristic approaches [12]. UKF is a mathematical sample based filter, since it uses a deterministic sampling process, the general estimation mathematics and the mathematical operators such as unscented transform. In comparison to UKF, there are sample based filters that utilize heuristic algorithms to sample the particles and to improve the position of them. These filters are called heuristic filters [12].

PF is a sample based filter and is an example of heuristic filters [12]. It works based on point mass (or particle) representation

* Corresponding author.
E-mail addresses: nobahari@sharif.edu (H. Nobahari), sm.zandavi@yahoo.com (S.M. Zandavi), mohammadkarim@ae.sharif.edu (H. Mohammadkarimi).

http://dx.doi.org/10.1016/j.asoc.2016.08.008
1568-4946/© 2016 Elsevier B.V. All rights reserved.

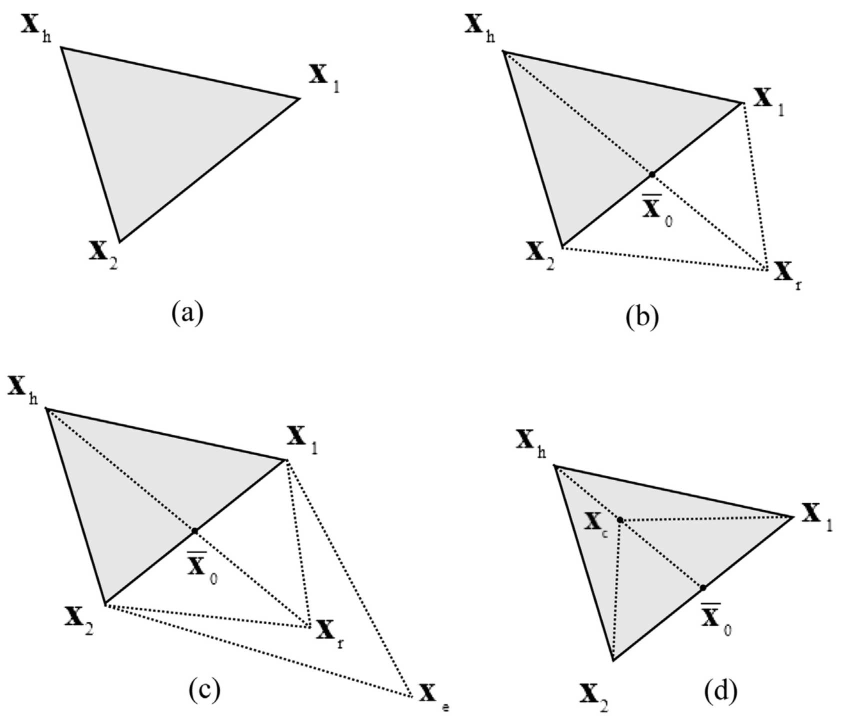

Fig. 1. Available moves in the simplex method: (a) initial simplex; (b) reflection; (c) expansion; (d) contraction;.

of the probability densities [13]. Unlike UKF, PF represents the required posterior Probability Density Function (PDF) by a set of random samples instead of deterministic ones. Also, it uses a resampling process to reduce the degeneracy of particles. The standard resampling process copies the important particles and discards insignificant ones based on their fitness. This strategy suffers from the gradual loss of diversity among the particles, known as sample impoverishment. Researchers have proposed different resampling strategies such as Binary Search [14], systematic resampling [15] and residual resampling [13]. Some heuristic optimization algorithms have also been inserted to PF to improve its performance, the examples of which are Genetic Algorithm (GA) [16,17], Simulated Annealing [18], Particle Swarm Optimization [19] and Ant Colony Optimization (ACO) [20–22]. Continuous Ant Colony Filter (CACF) [12] is another heuristic filter. In this filter, Continuous Ant Colony System (CACS) [23] is utilized for state estimation. Ant colony estimator, proposed in [24] utilizes real-coded ant colony optimization (ACO_R) [25]. Adaptive particle swarm filter and adaptive differential evolution filter, proposed in [26] utilize particle swarm optimization and differential evolution [27], respectively. In [28], particle swarm optimization, Kalman filter and maximum likelihood methods have been incorporated to estimate the states and the parameters of a discrete-time microstructure (DTMS) model, in order to enhance the flexibility and adaptability of the DTMS model. In [29], weighted least square method [30] has been hybridized with firefly algorithm [31] to reach more reliable and accurate state estimation of distribution networks. Recently, Hamiltonian cycle theory has also been utilized as a state estimation algorithm in distribution networks [32].

In [33], Nelder-Mead simplex algorithm has been added to Monte Carlo particle filter to decrease the parameter estimation error in a parameter estimation problem. The self-organizing state space model, proposed in [34], is used to augment the states and parameters of the problem and Nelder-Mead simplex algorithm seeks the initial distribution of the parameters by maximizing the likelihood of the self-organizing state space.

In the present work, the state estimation problem is modeled as a stochastic dynamic optimization problem, and a new estimation algorithm, based on simplex optimization, is proposed to solve this problem. It is the first time that an algorithm based on Nelder-Mead simplex is proposed as a filter for nonlinear systems state estimation. The proposed filter, called Simplex Filter (SF), uses a simplex with stochastic moving vertices to find and track the best estimation. The parameters of the proposed filter are tuned using a systematic approach to visualize the proper region of the parameters space.

The organization of this paper is as follows: Mathematical model of the estimation problem is represented in Section 2. Nelder-Mead simplex is described in Section 3. The novel filter, proposed in this paper, is introduced in Section 4. Implementation of the filter, tuning of the parameters and the numerical results are provided in Section 5. Finally, a conclusion is made in Section 6.

2. Problem formulation

The problem is to estimate the states of a discrete nonlinear dynamic system in a continuous search space. The model is written as follows:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \quad (1)$$

where k is time step, $\mathbf{f}_k$ is the system model, $\mathbf{x}_{k-1}$ is the state vector and $\mathbf{w}_{k-1}$ is the process noise corresponding to system uncertainties. Also, the measurement model is considered as

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \quad (2)$$

where $\mathbf{h}_k$ is the measurement model and $\mathbf{v}_k$ is the measurements noise.

3. Simplex method

Simplex method is a heuristic optimization algorithm. It is a direct search method and does not use the derivatives of the objective function. Simplex is a geometrical object produced by n + 1 points ($\mathbf{x}_0, \ldots, \mathbf{x}_n$) in an n-dimensional space [35,36]. For example, the simplex in two-dimensional space is a triangle. The basic idea in simplex method is to compare the value of the objective function at n + 1 vertices of a simplex and move the simplex gradually toward the optimum point through an iterative process. Initially, the following equation can be used to generate the vertices of a regular simplex of size a (an equilateral triangle in two-dimensional space), within the n-dimensional space [35]:

$$\mathbf{x}_j = \mathbf{x}_0 + p\mathbf{x}_j + \sum_{\substack{s=1 \\ s\neq j}}^{n} q\mathbf{u}_s \tag{3}$$

where $\mathbf{x}_0$ is the initial base point and $\mathbf{u}_s$ is the unit vector along the coordinate axis s. Also, p and q are defined as

$$p = \frac{a}{n\sqrt{2}}(\sqrt{n+1} + n - 1) \tag{4}$$

$$q = \frac{a}{n\sqrt{2}}(\sqrt{n+1} - 1) \tag{5}$$

Through a sequence of elementary geometric transformations (reflection, contraction and expansion), the simplex moves (see Fig. 1) toward the optimum point. After each transformation, the current worst vertex is replaced by a better one. Therefore, simplex gradually moves toward the optimum point.

The reflected, expanded and contracted points, denoted by x_r, $\mathbf{x}_e$ and $\mathbf{x}_c$, are formulated as

$$\mathbf{x}_r = (1+\alpha)\bar{\mathbf{x}}_0 - \alpha\,\mathbf{x}_h \qquad \alpha > 0 \tag{6}$$

$$\mathbf{x}_e = \gamma\,\mathbf{x}_r + (1-\gamma)\bar{\mathbf{x}}_0 \qquad \gamma > 1 \tag{7}$$

$$\mathbf{x}_c = \beta\,\mathbf{x}_h + (1-\beta)\bar{\mathbf{x}}_0 \qquad 0 \le \beta \le 1 \tag{8}$$

Here, $\bar{\mathbf{x}}_0$ is the centroid of all vertices $\mathbf{x}_j$ except j = h; where h is the index of the worst vertex. The parameters α, γ and β are called reflection, expansion and contraction coefficients, respectively.

Reflection is the operation by which the worst vertex, called high, is reflected with respect to the centroid $\bar{\mathbf{x}}_0$. If the reflected point is better than all other points, the method expands the simplex in the reflection direction; otherwise, if it is at least better than the worst, the algorithm performs again the reflection with the new worst point [35]. The contraction is performed when the worst point is at least as good as the reflected point.

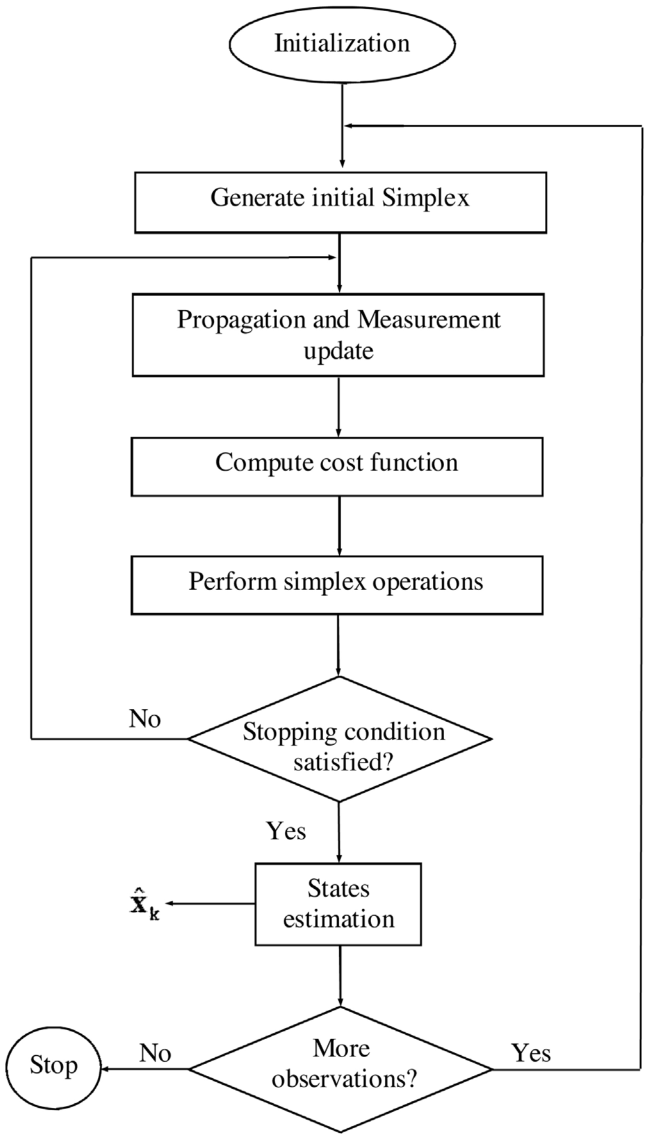

Fig. 2. Flowchart of simplex filter.

4. Simplex filter

In this section, the new heuristic filter, called "Simplex Filter" (SF), is introduced as a tool for nonlinear systems state estimation. First, the general structure of the proposed filter is introduced. Then, the formulation of the algorithm is presented in detail.

4.1. General setting out of the algorithm

The block diagram of SF is shown in Fig. 2. The parameters of SF are set during the initialization. SF has two loops. The outer loop generates an initial simplex every time a new measurement in entered. The inner loop iterates to find the best estimation of the current states, corresponding to the entered measurement. To do this, the inner loop, first propagates the vertices of the simplex. Then, for each vertex, the corresponding output is calculated based on the measurement model. The calculated outputs are compared with the real measurement and each vertex is assigned a cost. The inner loop uses reflection, expansion and contraction operations to reshape and move the simplex toward the maximum likelihood regions of the state space and is terminated when the maximum number of iterations ($i_{\max}$) is reached. At last, the state estimation is made using the centroid of the final simplex. Table 1 represents pseudo-code of SF.

4.2. Formulation of the algorithm

Simplex filter has six parameters that must be set before the execution of the algorithm. These parameters are listed in Table 1 and they are introduced in the following.

The outer loop generates an initial simplex every time a new measurement in entered. For this purpose, an initial guess ($\mathbf{x}_0$) is generated randomly with uniform distribution in search space. The initial simplex is then generated using Eqs. (3)–(5) where the initial simplex size (a) is generated as follows:

$$a = \mathrm{rand}(a_{\min}, a_{\max}) \tag{9}$$

where $a_{\min}$ and $a_{\max}$ are lower and upper bounds of a. These parameters are tuned in Section 5.1.

During the i-th iteration of the inner loop, the position of the j-th vertex at time $k-1$, defined by $\mathbf{x}_{k-1}^{i,j}$, is propagated as follows:

$$\mathbf{x}_k^{i,j} = \mathbf{f}_k(\mathbf{x}_{k-1}^{i,j}, \mathbf{w}_{k-1}^{i,j}) \tag{10}$$

Table 1
Pseudo-code of simplex filter.

```
Initialization
    Set $a_{\min}$, $a_{\max}$, $\alpha_{\max}$, $\gamma_{\max}$, $\beta_{\max}$ and $i_{\max}$
Repeat
    $\mathbf{x}_0$ = random
    Generate initial simplex
    $i$ = 1
    Repeat
        Propagate vertices of the simplex
        Measurement update for each vertex
        Compute cost function for each vertex
        Update simplex
            Reflection
            Contraction
            Expansion
    $i = i + 1$
Until $i \geq i_{\max}$
Until measurement is stopped
```

Then, the output, corresponding to $\mathbf{x}_k^{i,j}$, denoted by $\mathbf{z}_k^{i,j}$, is calculated as

$$\mathbf{z}_k^{i,j} = \mathbf{h}_k(\mathbf{x}_k^{i,j}) \tag{11}$$

Vertices of the simplex are evaluated, based on the likelihood of their position within the state space. The evaluation function is defined as the error between the calculated output, $\mathbf{z}_k^{i,j}$, and the real measurement, $\tilde{\mathbf{z}}_k$. Different evaluation functions are used to evaluate the vertices the example of which is

$$f_k^{i,j} = |\mathbf{z}_k^{i,j} - \tilde{\mathbf{z}}_k| \tag{12}$$

Table 2
Functions, utilized to evaluate the vertices of simplex.

Evaluation Function	f_1	f_2	f_3	f_4
Math definition	$\lvert z-\tilde{z}\rvert$	$(z-\tilde{z})^2$	$\exp[-\frac{1}{2R}(z-\tilde{z})^2]$	$-\frac{1}{2R}\exp(z-\tilde{z})^2$

Other functions are listed in Table 2. These functions are examined in Section 5.2.

During any iteration of the inner loop, the simplex will be updated using reflection, expansion and contraction operators. The parameters of these operators are changed randomly as follows:

$$\alpha = \alpha_{\max}\ \text{rand}\,(0,1) \tag{13}$$

$$\gamma = 1 + (\gamma_{\max} - 1)\ \ \text{rand}(0,1) \tag{14}$$

$$\beta = \beta_{\max}\ \text{rand}\ (0,1) \tag{15}$$

where $\alpha_{\max}$, $\gamma_{\max}$ and $\beta_{\max}$ are SF parameters that are tuned in Section 5.1.

The inner loop stops when the maximum number of iterations ($i_{\max}$) is reached. The outer loop stops when the measurements are finished.

After termination of the inner loop, the center of average of the vertices is calculated and passed as the state estimation.

$$\hat{\mathbf{x}}_k = \frac{1}{n+1}\sum_{j=1}^{n+1}\mathbf{x}_k^{i_{\max},j} \tag{16}$$

5. Implementation of the algorithm

In this section, the performance of SF is evaluated using the benchmark problems, listed in Table 3. The noise properties of these problems are given in Table 4. Different functions are examined to evaluate the vertices of the simplex. For each evaluation function, the parameters of SF are tuned and the algorithm is applied to all benchmarks. The results are compared to KF, UKF, Generic Particle Filter (PF), Continuous Ant Colony Filter (CACF) [12], Sequential Importance Sampling Filter (SIS) [39], Evolutionary Particle Filter (EPF) [17] and Evolution Strategies based Particle filter (ESP) [38].

5.1. Tuning of the parameters

SF has six parameters that must be tuned to achieve a good performance. Different methods exist for tuning the parameters of an algorithm. Ref [40] has introduced popular methods: one factor at a time design, factorial design and random sampling. In this paper, random sampling is carried out to tune the parameters of SF within the intervals, given in Table 5.

The intervals of the parameters space can be determined, regarding the definition of each parameter and the problem at hand. These intervals are taken widely enough to commonly be used for different problems. The maximum size of the initial simplex, $a_{\max}$, must be large enough to have enough exploration during the initial

Table 3
Nonlinear estimation problems, used to investigate the performance of SF.

No. (Ref)	Propagation Model	Measurement Model	Simulation time (sec)
1 ([20])	$x(t+1) = 1 + \sin(4\times10^{-2}\pi t) + 0.5x(t) + w_1(t)$	$z(t) = \begin{cases} \frac{x(t)^2}{5} + v_1(t) & t \leq 30 \\ -2 + \frac{x(t)}{2} + v_1(t) & t > 30 \end{cases}$	60
2 ([37])	$x(t+1) = 1 + \sin(4\times10^{-2}\pi t) + 0.5x(t) + w_2(t)$	$z(t) = \begin{cases} \frac{x(t)2}{5} + v_1(t) & t \leq 30 \\ -2 + \frac{x(t)}{2} + v_1(t) & t > 30 \end{cases}$	60
3 ([17])	$x(t) = x(t-1) + 12\frac{x(t-1)}{1+x(t-1)^2} + 7\cos(1.2(t-1)) + w_3(t)$	$z(t) = \frac{x(t)^2}{20} + v_3(t)$	100
4 ([38])	$x(t) = \frac{x(t-1)}{2} + 25\frac{x(t-1)}{1+x(t-1)^2} + 8\cos(1.2t) + w_4(t)$	$z(t) = \frac{x(t)^2}{20} + v_4(t)$	100

Table 4
Noise model of the problems introduced in Table 2 .

Problem Number	Dynamic system noise (w_i)		Measurement noise (v_i)	
1	Gamma distribution	$k=7$ $\theta=2$	White noise	$E=0$ $\sigma^2=10^{-5}$
2	Gamma distribution	$k=3$ $\theta=2$	White noise	$E=0$ $\sigma^2=10^{-5}$
3	White noise	$E=0$ $\sigma^2=4$	White noise	$E=0$ $\sigma^2=4$
4	White noise	$E=0$ $\sigma^2=10$	White noise	$E=0$ $\sigma^2=1$

Table 5
Search domain of SF parameters.

Parameter	min	max
$a_{\max}$	5	20
$a_{\min}$	0	$a_{\max}$
$\alpha_{\max}$	0	10
$\gamma_{\max}$	1	10
$\beta_{\max}$	0	1
$i_{\max}$	20	120

Table 6
Normalized cost functions, utilized to tune the parameters of SF.

Problem Number	Cost
1	$c_1=\frac{\text{Mean of RMSE}}{10}+\text{Standard Deviation of RMSE}$
2	$c_2=\frac{\text{Mean of RMSE}}{10}+\text{Standard Deviation of RMSE}$
3	$c_3=\left(\text{Mean of RMSE}+\frac{\text{Standard Deviation of RMSE}}{2}\right)/20$
4	$c_4=\frac{\text{Mean of RMSE}}{10}+\text{S tan dard Devation of RMSE}$

iterations and it must be small enough with respect to the size of the state space to provide precise estimations. Logically, $a_{\min}$ must be greater than zero and less than $a_{\max}$. The maximum reflection coefficient, $\alpha_{\max}$, can be less or greater than one. Therefore, the permitted interval is taken as [0,10]. The expansion coefficient, $\gamma_{\max}$, according to its definition, is taken greater than one. Therefore, the permitted interval is taken as [1,10]; The contraction coefficient, $\beta_{\max}$, is defined to be less than one. Therefore, it is taken between zero and one. Finally, the tuning interval for the maximum number of iterations, $i_{\max}$, is taken by trial and error as [20,120]. This parameter must be large enough to obtain good performance for different problems and be small enough to decrease the computational cost.

Flowchart of the tuning algorithm is shown in Fig. 3. In this method, the performance of SF is evaluated for m sets of parameters, generated randomly using uniform distribution. For each set, all benchmarks, introduced in Table 3, are solved. Each problem is solved 30 times and the mean and the standard deviation of Root Mean Square Error (RMSE) is calculated. Here, RMSE is defined as follows:

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t-x_t)^2}{n}} \tag{17}$$

Then, a cumulative cost is calculated for each problem using the normalized functions (c_i), defined in Table 6. Then, for each parameter set, the average cost of all problems is obtained as follows:

$$\bar{c}=\frac{1}{4}\sum_{i=1}^{4}c_i \tag{18}$$

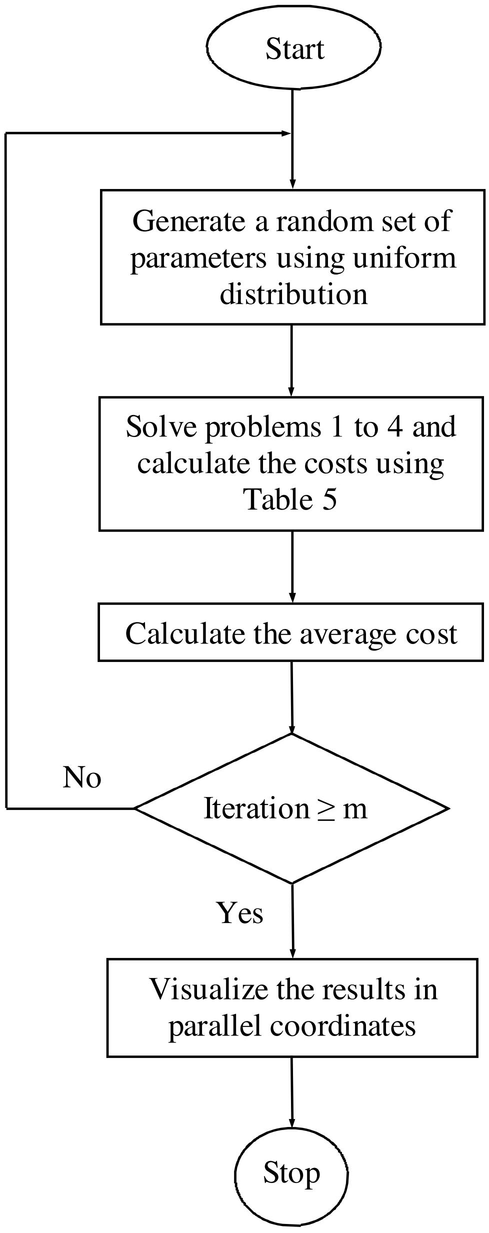
Start
Generate a random set of parameters using uniform distribution
Solve problems 1 to 4 and calculate the costs using Table 5
Calculate the average cost
Iteration ≥ m
No
Yes
Visualize the results in parallel coordinates
Stop

Fig. 3. Flowchart of the tuning algorithm.

In order to visualize the optimal values of the parameters in parallel coordinates, these parameters must be normalized, too. The normalized parameters are defined as follow:

$$\bar{a}_{\max}=(a_{\max}-5)/15 \tag{19}$$

Table 7
Range of the parameters, obtained for each evaluation function when m = 100.

Evaluation Function	Parameter(min, max)					
	$\bar{a}_{max}$	$\bar{a}_{min}$	$\bar{\alpha}_{max}$	$\bar{\gamma}_{max}$	$\bar{\beta}_{max}$	$\bar{i}_{max}$
f_1	(0.078, 0.967)	(0.043, 0.458)	(0.238, 0.925)	(0.098, 0.956)	(0.037, 0.957)	(0.135, 0.829)
f_2	(0.099, 0.835)	(0.033, 0.681)	(0.140, 0.834)	(0.011, 0.985)	(0.364, 0.897)	(0.035, 0.852)
f_3	(0.036, 0.931)	(0.021, 0.747)	(0.029, 0.970)	(0.041, 0.813)	(0.050, 0.842)	(0.158, 0.914)
f_4	(0.036, 0.912)	(0.0185, 0.408)	(0.012, 0.951)	(0.017, 0.892)	(0.029, 0.837)	(0.235, 0.7963)

$$\bar{a}_{min} = a_{min}/a_{max} \tag{20}$$

$$\bar{\alpha}_{max} = \alpha_{max}/10 \tag{21}$$

$$\bar{\gamma}_{max} = (\gamma_{max} - 1)/9 \tag{22}$$

$$\bar{\beta}_{max} = \beta_{max} \tag{23}$$

$$\bar{i}_{max} = (i_{max} - 20)/100 \tag{24}$$

Fig. 4 shows in parallel coordinates, the results, obtained for function f_1 after evaluation of 100 random sets of the normalized parameters (i.e. m = 100). The axes present the normalized parameters and the average cost function ($\bar{c}$), defined in Eq. (18). In Fig. 5, ten top sets of SF parameters out of 100 sets, are visualized. Based on this figure, the proper bounds of the parameters are obtained for function f_1.

The same procedure is performed for other evaluation functions, defined in Table 2. In Table 7, the proper bounds of SF parameters, obtained for each alternative function, are presented. The tuned value of SF parameters are then calculated using the weighted average of ten top sets of the parameters. For example, α is fixed as follows:

$$\alpha = \frac{\sum_{i=1}^{10} (11 - r_i)\alpha_i}{\sum_{i=1}^{10} (11 - r_i)} \tag{25}$$

where α_i is the reflection coefficient of the i-th set of parameters and r_i is the rank of that set among ten top sets. Finally, the tuned parameters, obtained for m = 100, are presented in Table 8.

To investigate the effect of m on the performance of the tuning procedure, $\bar{c}$ is again calculated using the tuned set of parameters, obtained for m = 50, 100, 200 and 500. The variation of $\bar{c}$ versus m is illustrated in Fig. 6. This figure shows that 100 random sets of parameters are enough for a good tuning of the parameters.

Table 8
Tuned parameters of simplex filter for m = 100.

Evaluation Function	Parameter					
	$\bar{a}_{max}$	$\bar{a}_{min}$	$\bar{\alpha}_{max}$	$\bar{\gamma}_{max}$	$\bar{\beta}_{max}$	$\bar{i}_{max}$
f_1	0.4365	0.3389	0.4237	0.5678	0.4273	0.4178
f_2	0.5084	0.2912	0.4511	0.5701	0.7330	0.4975
f_3	0.4025	0.2190	0.1293	0.4300	0.5817	0.6725
f_4	0.3599	0.1783	0.0504	0.4231	0.5494	0.6509

5.2. Numerical results

In Table 9, the performance of SF is compared with some other filters. Dash signs correspond to the cases for which there is no reported value available in the literature. In this table, Root Mean Square Error (RMSE) and Mean Square Error (MSE) of different filters, obtained for different benchmark problems, are compared with those of SF. To find the best evaluation function from the list of candidates, $f_1, \ldots, f_4$, proposed in Table 2, the results of SF, obtained for $f_1, \ldots, f_4$, are also represented in Table 9 and compared to each other. This comparison shows that f_1 and f_2 have superior performance than f_3 and f_4, and f_1 is the best. In other word, the absolute of the innovation function (i.e. f_1) is the best option to evaluate the performance of each vertex as compared to other options of Table 2 and it is a little better than its square (i.e. f_2). Table 9 also shows that SF has superior results than other filters in all mentioned problems.

The average execution time is measured and given in Table 10, using a specific computer, characterized by Inter(R) Core(TM) i3 CPU M370 at 2.40 GHz. Results are compared with those, reported in [19] and [20]. Note that, the specifications of the computer machines, used in [19] and [20] are unknown. Therefore, one cannot compare the execution time of different references. However, the presented results show that SF has acceptable execution time. Also, SF shows similar execution times for the examined problems; while Generic PF, for example, shows much more execution time in problem 2 than problem 1.

Figs. 7–10 illustrate the performance of SF in solving the benchmark problems, given in Table 3. These figures represent a sample

Table 9
Comparison of Simplex Filter with other filters. Dash signs represent unavailable data.

Filter	Problem Number (Statistical Performance Measure, No. of Run)			
	1 (RMS Error, 30)	2 (MSE, 30)	3 (MSE, 10)	4 (MSE, 10)
EKF [20,21]	0.9809	0.3783	–	–
UKF [20,21]	0.68237	0.2969	–	–
Generic PF [20,21]	0.7792	0.2205	–	–
PF + ACO [20]	0.28153	–	–	–
CACF [12]	0.6513	–		–
PF + IEKF [37]	–	0.0495	–	–
PF + PSO [19]	–	0.074377	–	–
EPF [17]	–	–	1.1050	–
ESP[38]	–	–	–	61.21
SIS [14]	–	–	–	133.44
Simplex Filter (f_1)	0.1139	0.0118	0.9768	5.8246
Simplex Filter (f_2)	0.1198	0.01208	1.0868	5.9286
Simplex Filter (f_3)	75.5234	83.8429	6.1265×10^7	3.1274×10^7
Simplex Filter (f_4)	65.3802	89.7736	1.1215×10^3	1.1292×10^3

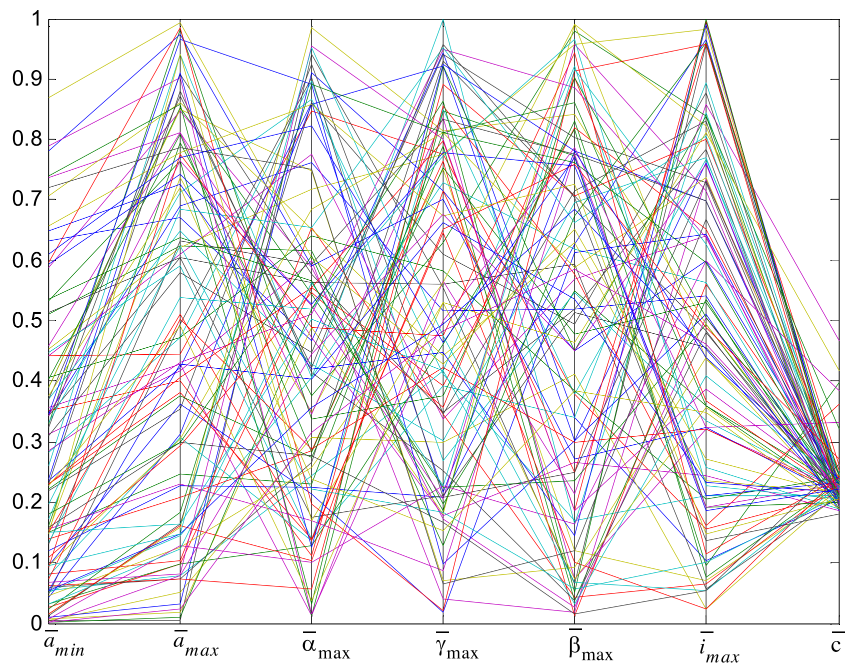

Fig. 4. Parallel coordinates plot of 100 random settings for function f_1.

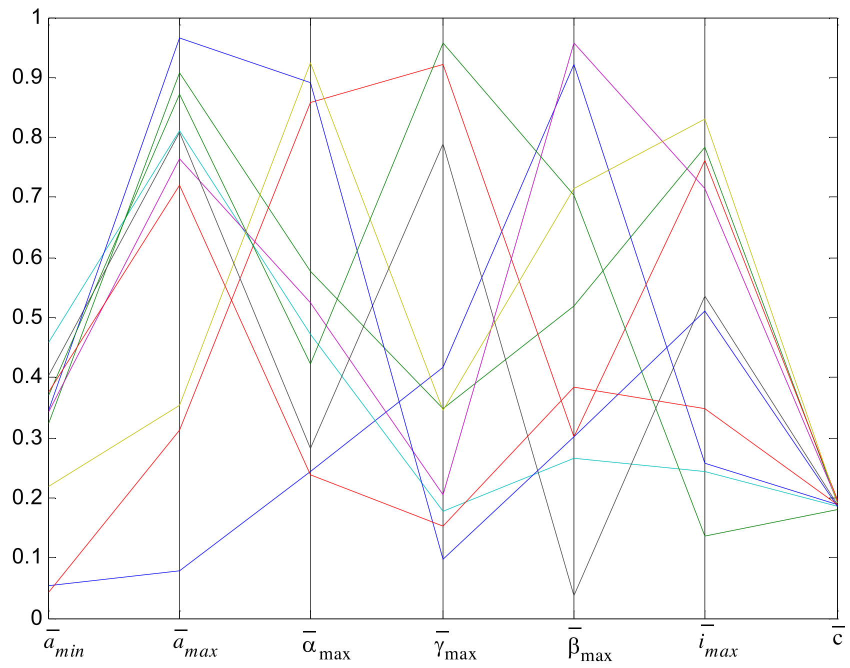

Fig. 5. Parallel coordinates plot of 10 top out of 100 settings for function f_1.

output of SF and show that the proposed filter can track the true signal accurately. These figures also show the best estimations, obtained for $i_{\max} = 1$, 20 and 60. In other words, it shows the fast convergence rate of SF. It also shows that the parameter tuning method has done its best and $i_{\max} = 60$ is sufficient. This result is verified again, later.

The time history of the estimation error in solving problems 1–4 is shown in Fig. 11. The errors are very small as compared to the estimated and true states, shown in Figs. 7–10. It means that the proposed filter can accurately estimate the states. Also, Fig. 12 shows the convergence of SF, when a new measurement is entered. This figure again shows that $i_{\max}$ has properly been tuned.

In the following, SF is applied to state estimation of a vehicle with five states. The tracking model of the vehicle is formulated as follow [41]:

$$\mathbf{x}_k = \begin{bmatrix} 1 & \dfrac{\sin(\omega_k T)}{\omega_k} & 0 & -\dfrac{1-\cos(\omega_k T)}{\omega_k} & 0 \\ 0 & \cos(\omega_k T) & 0 & -\sin(\omega_k T) & 0 \\ 0 & \dfrac{1-\cos(\omega_k T)}{\omega_k} & 1 & \dfrac{\sin(\omega_k T)}{\omega_k} & 0 \\ 0 & \sin(\omega_k T) & 0 & \cos(\omega_k T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \\ 0 & 0 \end{bmatrix} \mathbf{w}_k \quad (26)$$

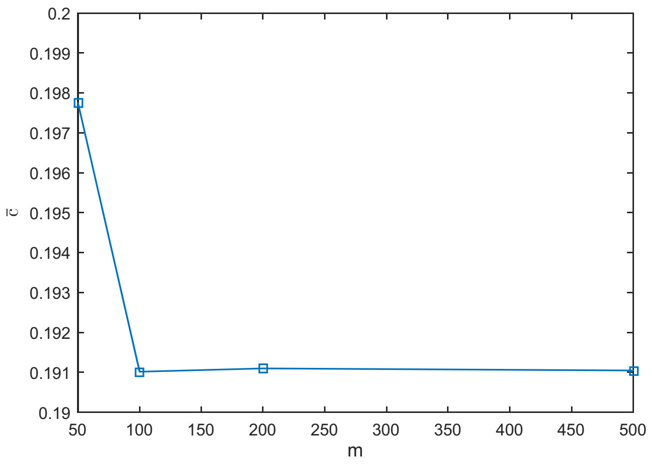

Fig. 6. Performance of the tuning algorithm versus the number of random sets (m).

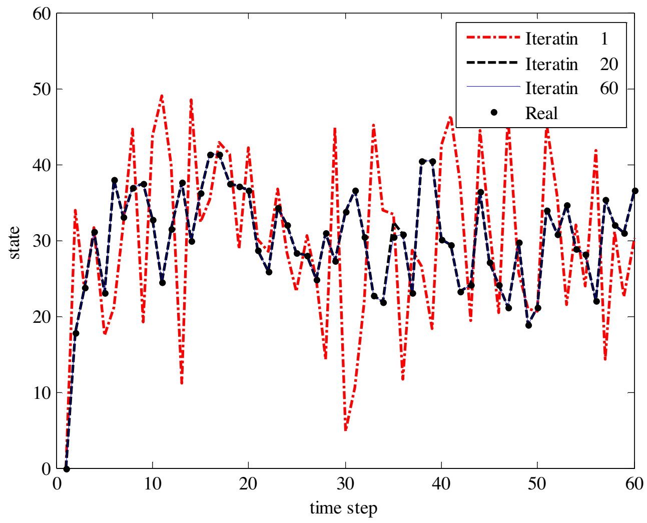

Fig. 7. Time history of the estimated and true states for problem 1.

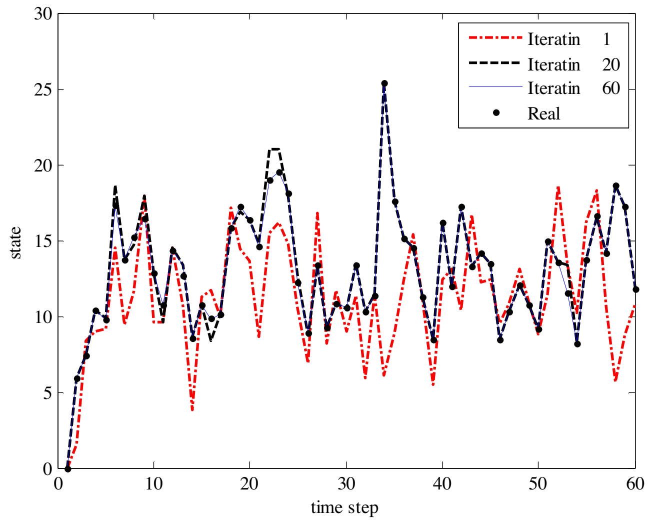

Fig. 8. Time history of the estimated and true states for problem 2.

Table 10
Average execution time of SF compared with other filters (computer machines, used in [19] and [20], are unknown).

Filter	Problem Number			
	1	2	3	4
EKF [20]	0.53321	–	–	–
UKF [20]	0.92803	–	–	–
Generic PF [20]	0.89604	4.047	–	–
PF + ACO [20]	3.1854	–	–	–
PF + PSO [19]	–	4.672	–	–
Simplex Filter	0.6895	0.7390	1.0644	1.0245

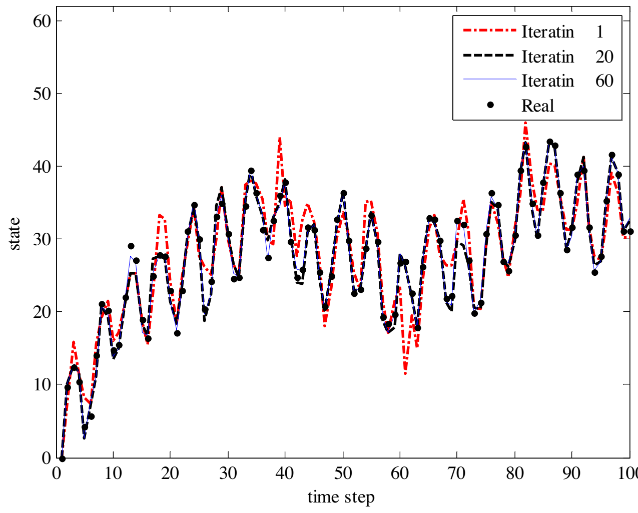

Fig. 9. Time history of the estimated and true states for problem 3.

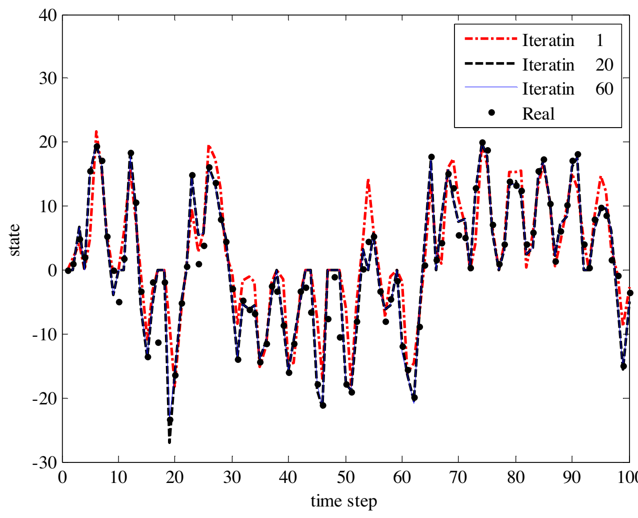

Fig. 10. Time history of the estimated and true states for problem 4.

$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_{k-1} + \mathbf{v}_k \tag{27}$$

where $\mathbf{x}_k = [\, x_k \quad \dot{x}_k \quad y_k \quad \dot{y}_k \quad \omega_k \,]^{\mathrm{T}}$ is the state vector, x_k is the longitudinal distance, y_k is the lateral distance, ω_k is turn rate of the vehicle, k is the time index, T is the sample time and $\mathbf{z}_k$ is the measurement vector. Two other vectors $\mathbf{w}_k = N(0, \mathbf{Q})$ and $\mathbf{v}_k = N(0, \mathbf{R})$ are mutually independent white noises, where $\mathbf{Q} = \mathrm{diag}(1^2, 1^2)$ and $\mathbf{R} = \mathrm{diag}(0.1^2, \ 0.1^2)$. The initial states are set as $\mathbf{x}_0 = [\, 0 \quad 0.2 \quad 0 \quad 0.2 \quad 0 \,]$. It is assumed that measurements are

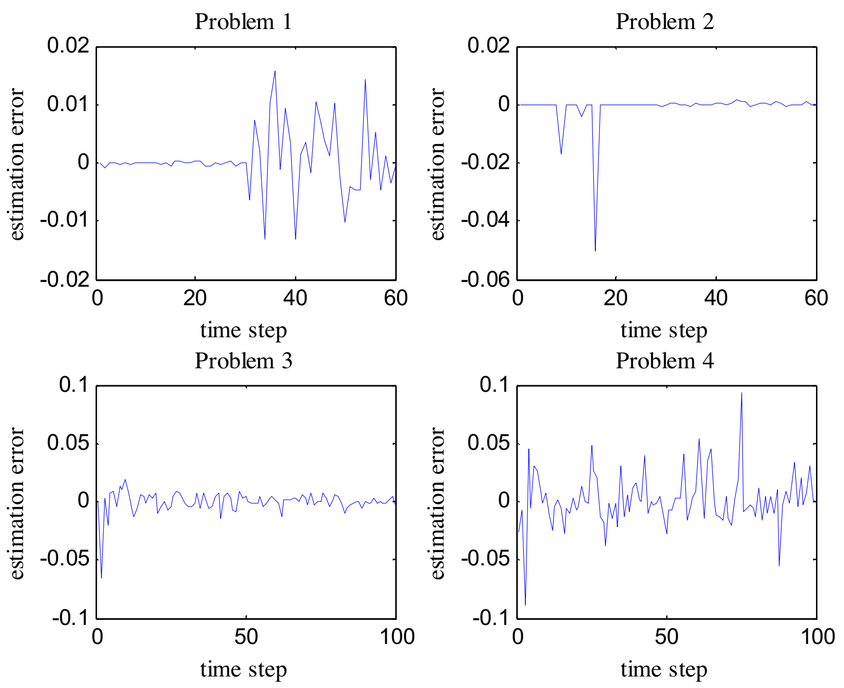

Fig. 11. Time history of the estimation error in different problems.

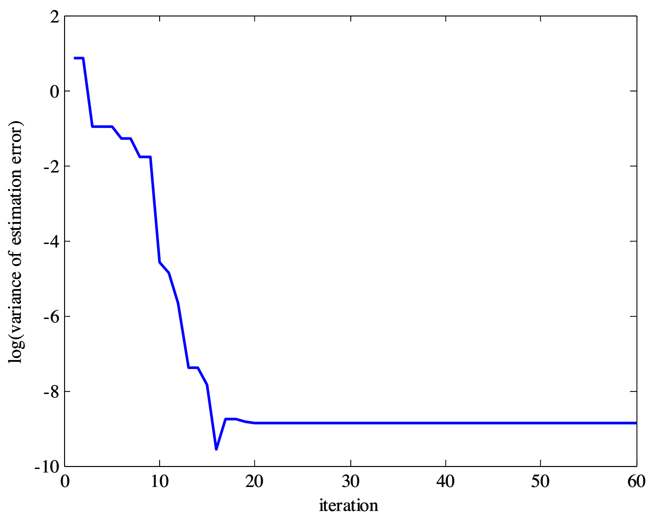

Fig. 12. Variance of the inner loop versus the iteration number.

Table 11
Average mean square error in vehicle tracking problem.

	SIR [17]	APF [17]	BPF [17]	RPF [17]	EPF [17]	Simplex Filter
x_k	4.0779	4.5527	4.8034	1.9520	0.7413	0.6553
y_k	11.2739	10.9387	8.9493	3.4262	1.4480	1.2813

missing for one step at $k = 25$ and the vehicle is moving fast during this period [41]. Fig. 13 represents the time history of true and estimated states and Fig. 14 represents the estimation errors. These figures show that SF can accurately estimate the states of the vehicle. Simulations are repeated 10 times and the average mean square error is reported in Table 11 and compared with Sampling Importance Resampling (SIR) [14], Auxiliary Particle Filter (APF) [42], Boostrap Particle Filter (BPF) [14], Regularized Particle Filter (RPF) [13] and Evolutionary Particle Filter (EPF) [17]. According to the results, taken from [17], SF shows the best performance among the aforementioned filters.

To investigate the effect of sample time on the performance of SF, the mean square error of x_k and y_k, obtained for different values of sample time, is analyzed. Fig. 15 shows that the accuracy of estimation is acceptable when the sample time is small enough, and it deteriorates by increasing the value of sample time. As expected, the designer must select sample time to be small enough to obtain acceptable performance.

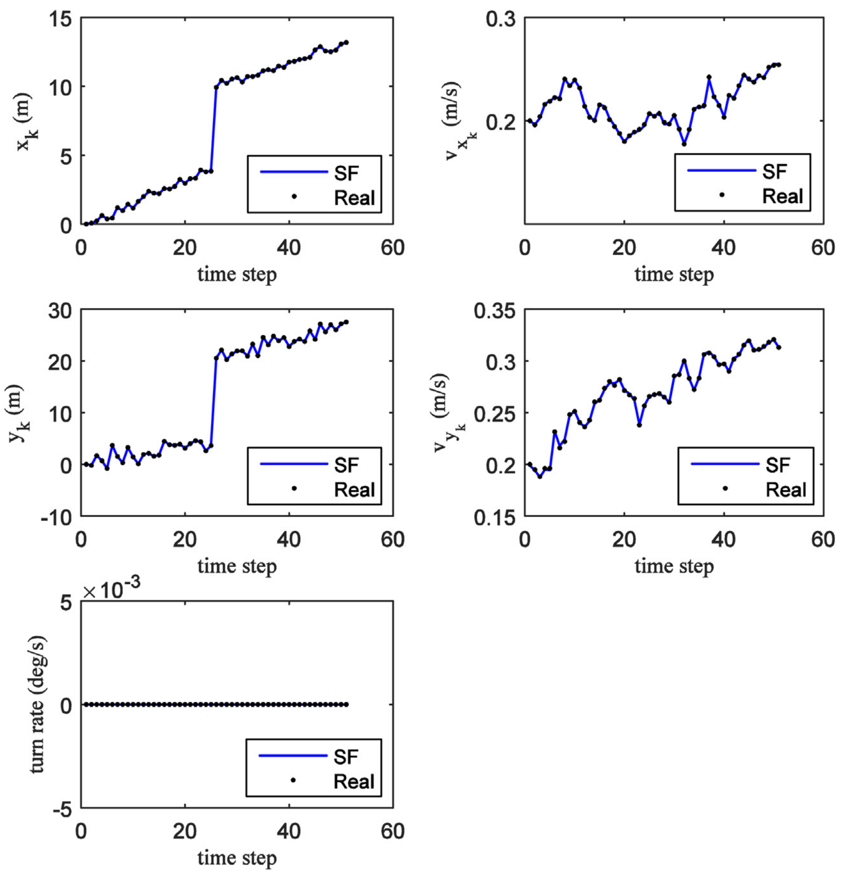

Fig. 13. Time history of the estimated and true states in vehicle tracking problem.

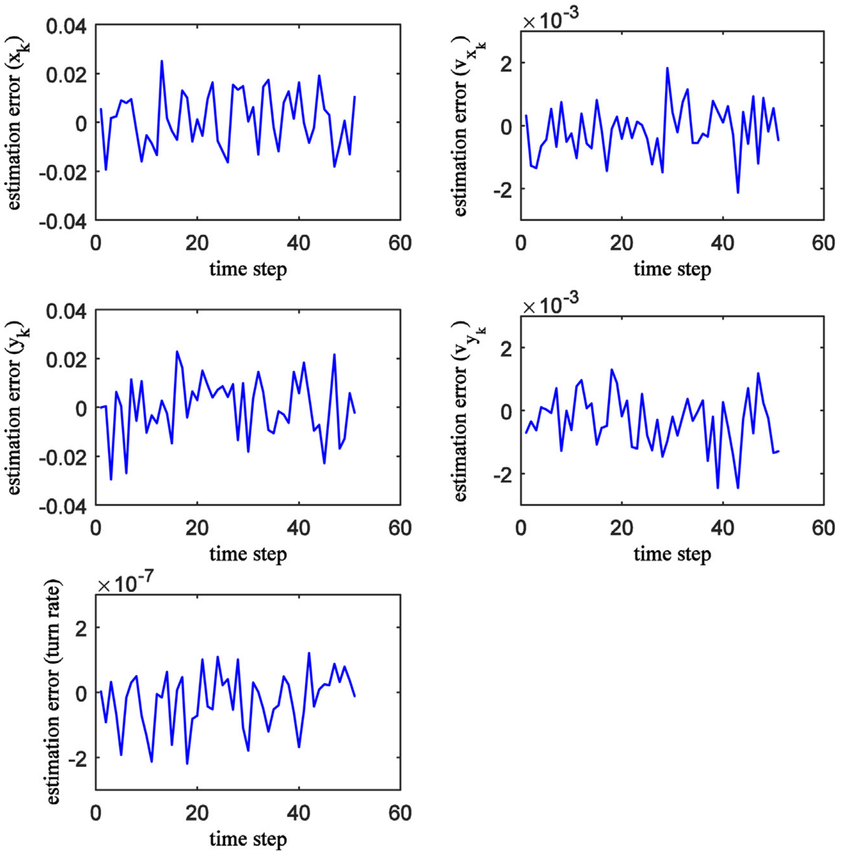

Fig. 14. Time history of the estimation errors in vehicle tracking problem.

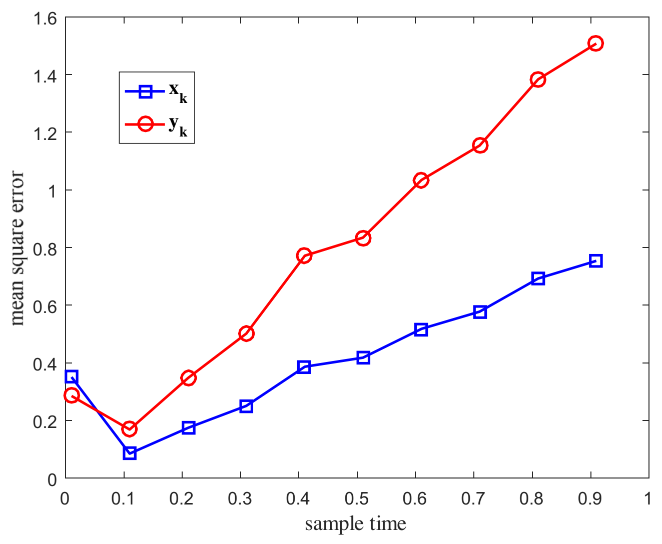

Fig. 15. Sensitivity of mean square error to sample time in vehicle tracking problem.

6. Conclusion

In this paper, a new heuristic filter was proposed for nonlinear systems state estimation. The proposed filter formulates the state estimation problem as a stochastic dynamic optimization problem. A new stochastic method, based on Nelder-Mead simplex, was proposed to solve this problem. The proposed scheme introduces a new filter, called simplex filter. This filter uses reflection, contraction and expansion operators to search the state space dynamically, in a scheme similar to the simplex algorithm. The parameters of the new filter were tuned utilizing a sample based visualization technique. This technique visualizes the proper ranges of the parameters on a series of parallel coordinates. The performance of the new filter was evaluated using a series of benchmarks and the results were compared with other filters. Numerical results show that simplex filter has a great performance in solving nonlinear state estimation problems.

Acknowledgements

The authors gratefully acknowledge financial supports of the research office at Sharif University of technology.

References

[1] J. Siouris, An Engineering Approach to Optimal Control and Estimation Theory, Air Force Institute of Technology, New York, 1995.
[2] R.E. Kalman, A new approach to linear filtering and prediction problem, Trans. ASME J. Basic Eng. 82 (Series D) (1960) 34–45.
[3] A.H. Jazwinski, Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.
[4] S.J. Julier, J.K. Uhlmann, A new extension of the Kalman filter to nonlinear system, AeroSense 11th International Symposium Aerospace Defense Sensing, Simulation and Control (1997) 182–193.
[5] J. Carpenter, P. Clifford, P. Fernhead, An improved particle filters for non-linear problem, IEE, Proceedings, Radar Sonar and Navigation 146 (1997) 2–7.
[6] D.L. Alspach, H.W. Sorenson, Nonlinear Bayesian estimation using Gaussian sum approximation, IEEE Trans. Autom. Control 17 (1972) 439–448.
[7] H.W. Sorenson, Recursive estimation for nonlinear dynamic systems, Bayesian analysis of time series and dynamic models, 94 (1988) 127–165.
[8] R.S. Bucy, Bayes theorem and digital realization for nonlinear filters (one dimensional discrete time nonlinear filters synthesis using Bayes theorem and digital techniques), J. Astronaut. Sci. 17 (1969) 80–94.
[9] G. Kitagawa, Non-Gaussian state-space modeling of non-stationary time series, J. Am. Stat. Assoc. 82 (400) (1987) 1032–1063.
[10] S.C. Kramer, H.W. Sorenson, Recursive Bayesian estimation using piece-wise constant approximations, Automatica 24 (6) (1988) 789–801.
[11] A. Pole, M. West, Efficient Numerical Integration in Dynamic Models, Department of Statistics, University of Warwick, 1988.
[12] H. Nobahari, A. Sharifi, A Novel Heuristic Filter Based on Ant Colony Optimization for Non-Linear Systems State Estimation, Computational Intelligence and Intelligent Systems, Springer, Berlin, Heidelberg, 2012, pp. 20–29.
[13] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174–188.
[14] N.J. Gordon, D.J. Salmond, A.F.M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proc. F (Radar and Signal Process.) 140 (2) (1993) 107–113.
[15] A. Doucet, D. Freitas, N.J. Gordon, Sequential Monte Carlo Methods in Practice, Springer, 2001.
[16] T. Higuchi, Monte Carlo filter using the genetic algorithm operators, J. Stat. Comput. Simul. 59 (1) (1997) 1–23.
[17] S. Park, J.P. Hwang, E. Kim, H. Kang, A new evolutionary particle filter for the prevention of sample impoverishment, IEEE Trans. Evol. Comput. 13 (4) (2009) 801–809.
[18] T.C. Clapp, Statistical Methods for the Processing of Communication Data, University of Cambridge, Cambridge, 2001.
[19] G. Tong, Z. Fang, X. Xu, A particle swarm optimized particle filter for nonlinear system state estimation, IEEE Congr. Evol. Comput. (CEC) (2006) 438–442.
[20] J. Zhong, Y. Fung, M. Dai, A biologically inspired improvement strategy for particle filter: ant colony optimization assisted particle filter, Int. J. Control Autom. Syst. 8 (3) (2010) 519–526.
[21] Z. Hao, X. Zhang, P. Yu, H. Li, Video object tracing based on particle filter with ant colony optimization, IEEE Advanced Computer Control (ICACC) 2nd International Conference on 3 (2010) 232–236.
[22] Y. Yu, X. Zheng, Particle filter with ant colony optimization for frequency offset estimation in OFDM systems with unknown noise distribution, Signal Process. 91 (5) (2011) 1339–1342.
[23] S.H. Pourtakdoust, H. Nobahari, An extension of ant colony system to continuous optimization problems, in: Ant Colony Optimization and Swarm Intelligence, Springer, Berlin, Heidelberg, 2004, pp. 294–301.
[24] S.M. Kalami Heris, H. Khaloozadeh, Ant colony estimator: an intelligent particle filter based on ACO_R, Eng. Appl. Artif. Intell. 28 (2014) 78–85.
[25] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, Eur. J. Oper. Res. 185 (2008) 1155–1173.
[26] M. Kiani, S.H. Pourtakdoust, State estimation of nonlinear dynamic systems using weighted variance-based adaptive particle swarm optimization, Appl. Soft Comput. 34 (2015) 1–17.
[27] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[28] Y. Qin, H. Peng, Y. Xi, W. Xie, Y. Sun, X. Chen, An adaptive modeling and asset allocation approach to financial markets based on discrete microstructure model, Appl. Soft Comput. 43 (2016) 390–405.
[29] R. Khorshidi, F. Shabaninia, T. Niknam, A new smart approach for state estimation of distribution grids considering renewable energy sources, Energy 94 (2016) 29–37.
[30] A. Bose, K.A. Clements, Real-time modeling of power networks, Proc. IEEE 75 (12) (1987) 1607–1622.
[31] X.S. Yang, Firefly algorithm, Nat. Inspir. Metaheuristic Algorithms 20 (2008) 79–90.
[32] J.B. Leite, J.R.S. Mantovani, Distribution system state estimation using the Hamiltonian cycle theory, smart grid, IEEE Trans. 7 (1) (2016) 366–375.
[33] K. Yano, A self-organizing state space model and simplex initial distribution search, Comput. Stat. 23 (2) (2008) 197–216.
[34] G. Kitagawa, A self-organizing state- space model, J. Am. Stat. Assoc. (1998) 1203–1215.
[35] S.S. Rao, Engineering Optimization Theory and Practice, Third edition, WileyInterscience Publication, 1996.
[36] R. Chelouah, P. Siarry, Genetic and Nelder-Mead algorithm hybridized for a more accurate global optimization of continuous multiminima function, Eur. J. Oper. Res. 148 (2) (2003) 335–348.
[37] L. Liang-qun, J. Hong-bing, L. Jun-hui, The iterated extended Kalman particle filter, IEEE Int. Symp. Commun. Inf. Technol. 2 (2005) 1213–1216.
[38] K. Uosaki, Y. Kimura, T. Hatanaka, Nonlinear state estimation by evolution strategies based particle filters, IEEE Congr. Evol. Comput. 3 (2003) 2102–2109.
[39] J.S. Liu, R. Chen, Monte Carlo methods for dynamic systems, J. Am. Stat. Assoc. 93 (443) (1998) 1032–1044.
[40] N. Franken, Visual exploration of algorithm parameter space, IEEE Congr. Evol. Comput. (CEC) (2009) 389–398.
[41] Y.S. Kim, K.S. Hong, An IMM algorithm for tracking maneuvering vehicles in an adaptive cruise control environment, Int. J. Control Autom. Syst. 2 (3) (2004) 310–318.
[42] M. Pitt, N. Shephard, Auxiliary particle filters, J. Am. Stat. Assoc. 94 (446) (1999) 590–599.