Particle Filtering with Particle Swarm Optimization in Systems with Multiplicative Noise

A.D. Klamargias
Computational Intelligence Laboratory (CI Lab)
Department of Mathematics
University of Patras
GR–26110 Patras, Greece
adk@math.upatras.gr

K.E. Parsopoulos
Computational Intelligence Laboratory (CI Lab)
Department of Mathematics
University of Patras
GR–26110 Patras, Greece
kostasp@math.upatras.gr

Ph.D. Alevizos
Computational Intelligence Laboratory (CI Lab)
Department of Mathematics
University of Patras
GR–26110 Patras, Greece
philipos@math.upatras.gr

M.N. Vrahatis
Computational Intelligence Laboratory (CI Lab)
Department of Mathematics
University of Patras
GR–26110 Patras, Greece
vrahatis@math.upatras.gr

ABSTRACT

We propose a Particle Filter model that incorporates Particle Swarm Optimization for predicting systems with multiplicative noise. The proposed model employs a conventional multiobjective optimization approach to weight the likelihood and prior of the filter in order to alleviate the particle impoverishment problem. The resulting scheme is tested on a well–known test problem with multiplicative noise. Results are promising, especially in cases of high system and measurement noise levels.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; G.3 [**Probability and Statistics**]; I.6.8 [**Simulation and Modeling**]: Types of Simulation

General Terms

Algorithms

Keywords

Sequential Monte Carlo Simulation, Particle Filter, Particle Swarm Optimization

1. INTRODUCTION

Particle Filters (PFs) are popular models for estimating the state of a dynamical system as observations become available on–line [1, 3, 6]. PFs generate a set of random samples, which is propagated and updated recursively in order to approximate the state probability density function (pdf) of the system. Therefore, they are capable of addressing any nonlinearity and noise distribution. PFs rely on importance sampling, i.e., they use proposal distributions to approximate the posterior distribution satisfactorily. The most common choice of proposal distribution is the probabilistic model of the states evolution, i.e., the transition prior. This choice, however, fails if the likelihood is highly peaked compared to the prior or if it lies in the tail of the prior.

To overcome this problem several methods have been proposed in the literature. A popular approach is the Unscented Particle Filter (UPF) [16], which uses the Unscented Kalman Filter [7] approximation as the proposal distribution of the PF. This combination outperforms other existing filters but it comes at the cost of heavy computational burden. Recent approaches are based on optimization methods to avoid the resampling stage by biasing the prior sample towards regions of the state space with high likelihood [11, 15, 17]. An approach that employs Particle Swarm Optimization (PSO) for this purpose was recently proposed [14]. Although the produced filter with PSO outperformed the generic PF and UPF (in terms of computational load), the experiments were conducted only for one system with very small observation noise. Clearly, in such cases, the reliability of measurement plays a crucial role in the performance of the algorithm. Also, in all the aforementioned studies, additive noise was considered solely.

In this paper, we propose a PF model with PSO for systems with multiplicative noise. The proposed approach attempts to address the problem of biasing the sample significantly towards either the prior or the likelihood. This problem can be detrimental for the algorithm's performance in cases of highly noisy systems and measurements. The problem is tackled by considering a conventional multiobjective optimization problem where the likelihood and the prior distribution are aggregated into a single objective function. Then, modifying the prior sample such that the resulting

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

particles maximize the objective function, results in a new sample that assumes a more balanced distribution between the prior and the likelihood. The maximization of the corresponding objective function is performed using the PSO algorithm, which has been shown to be very efficient in a plethora of problems in science and engineering [5, 9, 10, 13]. The resulting scheme is tested on a well–known test problem with multiplicative noise. The reported results are promising, especially in cases of high system and measurement noise levels.

The paper is organized as follows: in Section 2, we briefly review PF and PSO, while, in Section 3, the proposed approach is presented. Section 4 is devoted to experimental results and the paper concludes in Section 5.

2. BACKGROUND INFORMATION

In the following subsections we describe the basic concepts of Particle Filtering (PF) as well as the PSO algorithm.

2.1 Particle Filtering

In this section we expose the basic concepts of PF, based on the introduction provided in [6]. We consider the general discrete–time, nonlinear, non–Gaussian state estimation problem. The signal (state) vector,

$$x_t \in \mathbb{R}^{n_x}, \qquad t \geqslant 0,$$

is assumed to evolve according to the system model

$$x_{t+1} = f(x_t, w_t),$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ is the (known) system transition function and $w_t \in \mathbb{R}^{n_w}$ is a zero mean, white–noise sequence of known pdf which is independent of past and current states. At discrete time moments, measurements

$$y_t \in \mathbb{R}^{n_y}, \qquad t \geqslant 1,$$

become available. These measurements are related to the state vector via the observation equation

$$y_t = h(x_t, e_t),$$

where $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_e} \to \mathbb{R}^{n_y}$ is the (known) measurement function and $e_t \in \mathbb{R}^{n_e}$ is a zero mean, white–noise sequence of known pdf, independent of past and current states and system noise.

In other words, the signal, $\{x_t\}_{t\geqslant 0}$, is an unobserved (hidden) Markov process of initial distribution $p(x_0)$ and prior distribution $p(x_{t+1} \mid x_t)$. The observations, $\{y_t\}_{t\geqslant 1}$, are conditionally independent given the process $\{x_t\}_{t\geqslant 0}$, and their marginal distribution is $p(y_t \mid x_t)$. This hidden Markov model can be shortly described by $p(x_0)$ and $p(x_{t+1} \mid x_t)$ for $t \geqslant 0$, and $p(y_t \mid x_t)$ for $t \geqslant 1$.

The main goal in PF is the recursive estimation in time of the filtering distribution $p(x_t \mid y_{1:t})$, i.e., the approximation of the pdf of the current state x_t, given all the measurements up to time t. This pdf may be obtained recursively in two stages: *prediction* and *update*.

Suppose that the required pdf, $p(x_t \mid y_{1:t})$, at time step t is available. Then using the system model it is possible to obtain the prior pdf of the state at time step $t+1$ (prediction):

$$p(x_{t+1} \mid y_{1:t}) = \int_{\mathbb{R}^{n_x}} p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t}) \, \mathrm{d}x_t. \qquad (1)$$

The transition pdf, $p(x_{t+1} \mid x_t)$, is given by

$$p(x_{t+1} \mid x_t) = p_{w_t}\left(f^{-1}(x_t, x_{t+1})\right) \left|\det(\mathrm{J}_{f^{-1}})\right|. \qquad (2)$$

Then, at time step $t+1$, a measurement, y_{t+1}, becomes available and may be used to update the prior via Bayes rule (update):

$$p(x_{t+1} \mid y_{1:t+1}) = \frac{p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid y_{1:t})}{p(y_{t+1} \mid y_{1:t})}. \qquad (3)$$

The likelihood function, $p(y_{t+1} \mid x_{t+1})$, is given by

$$p(y_{t+1} \mid x_{t+1}) = p_{e_{t+1}}\left(h^{-1}(x_{t+1}, y_{t+1})\right) \left|\det(\mathrm{J}_{h^{-1}})\right|. \qquad (4)$$

In Eqs. (2) and (4), $|\det(\mathrm{J})|$ denotes the absolute value of the Jacobian determinant. The normalizing denominator $p(y_{t+1} \mid y_{1:t})$ in Eq. (3) is given by

$$p(y_{t+1} \mid y_{1:t}) = \int_{\mathbb{R}^{n_x}} p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid y_{1:t}) \, \mathrm{d}x_{t+1},$$

and it is usually unknown. However, it is sufficient to evaluate:

$$p(x_{t+1} \mid y_{1:t+1}) \propto p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid y_{1:t}).$$

Let $\{x_t^{(i)}\}_{i=1,2,\ldots,N}$ be a set of random samples from the pdf $p(x_t \mid y_{1:t})$. PF constitutes an algorithm for propagating and updating these samples to obtain a set of values $\{x_{t+1}^{(i)}\}_{i=1,2,\ldots,N}$, distributed approximately as $p(x_{t+1} \mid y_{1:t+1})$. Thus, the filter is an approximation mechanism (simulation) of the relations in Eqs. (1) and (3). Let us now illustrate the prediction and update phase in the simulation.

Prediction: Each sample, $x_t^{(i)}$, is passed through the system model to obtain samples, $x_{t+1}^{(i)*}$, $i = 1, 2, \ldots, N$, from the prior at time step $t+1$, i.e., $x_{t+1}^{(i)*} = f(x_t^{(i)}, w_t^{(i)})$, where $w_t^{(i)}$ is a sample drawn from the pdf of the system noise $p(w_t)$.

Update: On receipt of the measurement y_{t+1}, evaluate the likelihood $p(y_{t+1} \mid x_{t+1}^{(i)*})$ of each prior sample and obtain a normalized weight $q_{t+1}^{(i)}$ for each sample, i.e.,

$$q_{t+1}^{(i)} = \frac{p(y_{t+1} \mid x_{t+1}^{(i)*})}{\sum_{j=1}^{N} p(y_{t+1} \mid x_{t+1}^{(j)*})}, \qquad i = 1, 2, \ldots, N.$$

Thus, a discrete distribution over $\{x_{t+1}^{(i)*}\}_{i=1,2,\ldots,N}$, with probability mass $q_{t+1}^{(i)}$ for each i, is defined. Next, resampling takes place N times from this (discrete) distribution to generate samples $\{x_{t+1}^{(i)}\}_{i=1,2,\ldots,N}$, so that for any $j = 1, 2, \ldots, N$, it holds that

$$\mathbb{P}(x_{t+1}^{(j)} = x_{t+1}^{(i)*}) = q_{t+1}^{(i)}, \qquad i = 1, 2, \ldots, N.$$

The above steps of prediction and update form a single iteration of the recursive algorithm, which is initialized using N samples, $\{x_0^{(i)}\}_{i=1,2,\ldots,N}$, drawn from the known initial pdf $p(x_0)$. The samples, $\{x_{t+1}^{(i)}\}_{i=1,2,\ldots,N}$, are approximately distributed as the required pdf $p(x_{t+1} \mid y_{1:t+1})$ [6].

2.2 Particle Swarm Optimization

PSO is a stochastic, population–based optimization algorithm. Since 1995 when it was originally proposed by Eberhart and Kennedy [4], PSO has been shown to be very efficient in a plethora of applications [5, 9, 13]. Its efficiency, simplicity as well as adaptability to different problems has rendered PSO as a very attractive approach for solving numerical optimization problems.

Let $S \subset \mathbb{R}^D$ be a D–dimensional search space and $f : S \to \mathbb{R}$ be the objective function under minimization. PSO employs a population, $\mathbb{S}$, called *swarm*, that consists of N search points called *particles*,

$$\mathbb{S} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}.$$

Each particle is a D–dimensional vector, $x^{(i)} \in S$, $i = 1, 2, \ldots, N$, and it moves with an adaptive velocity, $v^{(i)}$, in S. Also, it has a memory where it stores the best position it has ever visited, $b^{(i)} \in S$.

An iteration of the algorithm corresponds to an update of the positions of all particles. The update for each particle, $x^{(i)}$, is performed by computing the new velocity of the particle, taking into account both its own experience as well as the experience of some other particles that constitute its *neighborhood*, $NB^{(i)}$ [8]. The most common neighborhood scheme is the *ring topology*, where the particles are assumed to be organized on a ring, communicating with their immediate neighbors. Under this topology, a neighborhood of radius r of $x^{(i)}$ is defined as the set

$$NB_r^{(i)} = \{x^{(i-r)}, \ldots, x^{(i)}, \ldots, x^{(i+r)}\},$$

where $x^{(1)}$ follows immediately after $x^{(N)}$. We denote with g_i the index of the best particle in $NB^{(i)}$, i.e., the particle that has visited the best position in S in terms of its function value:

$$f(b^{(g_i)}) \leqslant f(b^{(j)}),$$

for all j such that $x_j \in NB^{(i)}$. Let t to be the iteration counter. Then, the swarm is updated using the equations [2],

$$\begin{aligned} v_{t+1}^{(i)} &= \chi \left[v_t^{(i)} + c_1 R_1 (b_t^{(i)} - x_t^{(i)}) + c_2 R_2 (b_t^{(g_i)} - x_t^{(i)}) \right], \\ x_{t+1}^{(i)} &= x_t^{(i)} + v_{t+1}^{(i)}, \end{aligned}$$

where $i = 1, 2, \ldots, N$ (all vector operations are performed componentwise). The parameter χ is called *the constriction coefficient* and it is used to constrain the magnitude of the velocities during the search. The positive constants c_1 and c_2 are referred to as the *cognitive* and *social* parameter, respectively; while R_1, R_2 are random vectors with components uniformly distributed in $[0, 1]$. Default values for χ, c_1 and c_2 are determined in the theoretical analysis of Clerc and Kennedy [2]. The best positions of the particles are updated at each iteration according to

$$b_{t+1}^{(i)} = \begin{cases} x_{t+1}^{(i)}, & \text{if } f(x_{t+1}^{(i)}) < f(b_t^{(i)}), \\ b_t^{(i)}, & \text{otherwise,} \end{cases} \qquad i = 1, 2, \ldots, N.$$

In the next section, we describe the proposed approach that incorporates PSO in PF.

3. THE PROPOSED APPROACH

The main disadvantage of PF is the *particle impoverishment* problem, which is a direct consequence of the resampling stage of the algorithm. This problem appears in cases where the region of the state space where the likelihood, $p(y_{t+1} \mid x_{t+1}^{(i)*})$, is significant is small, compared to the region where the prior, $p(x_{t+1} \mid y_{1:t})$, is significant. As a result, many of the samples, $x_{t+1}^{(i)*}$, $i = 1, 2, \ldots, N$, assume small weights, $q_{t+1}^{(i)}$, and they are not selected in the resampling stage. The effect is more intense if the narrow likelihood falls in a region of low prior density, i.e., in the tail(s) of the prior, where the samples $x_{t+1}^{(i)*}$ are few.

We address this problem by intervening at PF after the generation of the prior samples, $x_{t+1}^{(i)*}$, $i = 1, 2, \ldots, N$, in the prediction phase and before resampling. The aim is to move these samples towards regions of the state space where the likelihood is significant, without allowing them to go far away from the region of significant prior. Therefore, a multiobjective problem arises, with two competing objectives. The first objective consists of a function, F_1, that is maximized at regions of high likelihood, while the second objective, F_2, is maximized at regions of high prior. Assuming that the two objective functions are known, the simplest approach for solving the multiobjective problem is through a conventional weighted aggregation (CWA) approach that combines the two objectives into a single one, resulting in the maximization problem:

$$\max_{x \in S} \quad F(x) = \alpha_1 F_1(x) + \alpha_2 F_2(x), \tag{5}$$

where α_1 and α_2 are non–negative weights with $\alpha_1 + \alpha_2 = 1$, and S being the state (search) space of the prior samples. Thus, the generated prior samples, $x_{t+1}^{(i)*}$, $i = 1, 2, \ldots, N$, are considered as the initial swarm in PSO, and they are let to move in order to maximize the objective function $F(x)$. The resulting samples, after maximization, consist of the best positions ever achieved by the PSO particles during the search. These best positions are now considered as the sample that will undergo resampling from the PF. The rest of the procedure from this point is the same as for the plain PF. Therefore, at each iteration of the PF, an optimization phase intervenes between prediction and update.

The CWA approach is selected due to its straightforward applicability, simplicity, and its ability in tackling multiobjective problems with PSO satisfactorily [12, 13]. Moreover, CWA allows the user to modify the weights as desired in cases where special attention must be paid to either the system model (prior) or to the observation (likelihood). Obviously, the objective functions F_1 and F_2 are problem–dependent, since they are intimately related to the likelihood and prior of the system at hand. In the following paragraphs, we illustrate the derivation of these objective functions for the studied test problem.

We considered the following nonlinear model [6] with multiplicative noise:

$$x_{t+1} = s(x_t, w_t) = f(x_t)(1 + w_t), \tag{6}$$

$$y_t = m(x_t, e_t) = h(x_t)(1 + e_t), \tag{7}$$

where,

$$f(x_t) = 0.5x_t + \frac{25x_t}{1 + x_t^2} + 8\cos(1.2t),$$

and

$$h(x_t) = \frac{x_t^2}{20},$$

where w_t, e_t, are zero–mean Gaussian white noises with variances Q and R, respectively. In order to avoid singularities in computations, we will assume hereafter that f, h, as well as a sample point, x, cannot be exactly zero. According to

Table 1: The two objective functions.

$F_1\left(x_{t+1}^{(i),\mathrm{PSO}}\right)$	$=$	$\left\|\frac{1}{h\left(x_{t+1}^{(i)*}\right)}\right\| \frac{1}{\sqrt{2\pi R}} \exp\left\{-\frac{1}{2R}\left[\frac{y_{t+1}-h\left(x_{t+1}^{(i)*}\right)}{h\left(x_{t+1}^{(i)*}\right)}\right]^2\right\}$
$F_2\left(x_{t+1}^{(i),\mathrm{PSO}}\right)$	$=$	$\frac{1}{\sqrt{2\pi Q}} \exp\left\{-\frac{1}{2Q}\left[\frac{x_{t+1}^{(i),\mathrm{PSO}}-x_{t+1}^{(i)*}}{x_{t+1}^{(i)*}}\right]^2\right\}$

Eq. (6), it holds that

$$x_{t+1} = f(x_t) + f(x_t)w_t \;\Rightarrow$$
$$w_t = \frac{x_{t+1} - f(x_t)}{f(x_t)} = s^{-1}(x_t, x_{t+1}).$$

Similarly, from Eq. (7) we have

$$y_t = h(x_t) + h(x_t)e_t \;\Rightarrow$$
$$e_t = \frac{y_t - h(x_t)}{h(x_t)} = m^{-1}(x_t, y_t).$$

Thus, the transition pdf is given by

$$p(x_{t+1} \mid x_t) = p_{w_t}\left(s^{-1}(x_t, x_{t+1})\right)\left|\frac{\mathrm{d}}{\mathrm{d}x_{t+1}}s^{-1}\right|$$
$$= \left|\frac{1}{f(x_t)}\right| p_{w_t}\left(\frac{x_{t+1} - f(x_t)}{f(x_t)}\right),$$

and the likelihood is given by

$$p(y_t \mid x_t) = p_{e_t}\left(m^{-1}(x_t, y_t)\right)\left|\frac{\mathrm{d}}{\mathrm{d}y_t}m^{-1}\right|$$
$$= \left|\frac{1}{h(x_t)}\right| p_{e_t}\left(\frac{y_t - h(x_t)}{h(x_t)}\right),$$

with $p_{w_t}(\cdot), p_{e_t}(\cdot)$, being Gaussian pdfs. Therefore, the resulting objective functions that will be used in Eq. (5) are reported in Table 1, where $x_{t+1}^{(i),\mathrm{PSO}}$ is the i–th particle of the swarm in PSO, $x_{t+1}^{(i)*}$ is the corresponding sample point where $x_{t+1}^{(i),\mathrm{PSO}}$ was initialized at, and y_{t+1} is the observation. Note that the time index t refers to the PF system iteration and not to the PSO iteration.

4. EXPERIMENTAL ANALYSIS

The proposed approach was applied for the model defined by Eqs. (6) and (7) for different system and measurement noise levels,

$$Q, R \in \{0.01, 0.05, 0.1, 0.3, 0.5\},$$

that correspond to a disruption of 1%, 5%, 10%, 30% and 50% of the system or measurement value, respectively. The particle filter was used for 60 time steps, i.e., $t = 1, 2, \ldots, 60$. At each time step, an optimization phase was triggered after generating the sample from the prior. Two sample sizes were considered in our experiments, namely $N = 20$ and $N = 50$ particles. The sample was passed as initial swarm to the PSO, which performed 50 iterations and returned the best positions as the final sample. The PSO parameters were the typical $\chi = 0.729$, $c_1 = c_2 = 2.05$, in all cases, while the selected variant was the global one, in order to speed up the swarm's convergence. The objective function defined by Eq. (5) was used in the optimization phase for three different levels of α_1 (α_2 is simply computed as $1-\alpha_1$), namely 0.2 (more important prior), 0.5 (equal importance) and 0.8 (more important likelihood).

For each parameter level, 100 independent experiments were performed. At each experiment, the Root Mean Squared (RMS) error was computed for the 60 system time steps. These RMS errors were averaged over all experiments, and the mean number as well as the corresponding variance was recorded, both for the plain PF approach as well as for the proposed approach that will be denoted PSO–PF hereafter. Moreover, the Wilcoxon Rank Sum test was applied to compute the statistical significance between each PSO–PF case and the plain PF, with p–values under 0.05 being considered as statistically significant.

All the obtained results are reported in Tables 2 and 3, for the case of sample size $N = 20$ and $N = 50$, respectively. More specifically, the first two columns of the tables determine the corresponding system and measurement noise level. The rest of the columns contain the mean number and variance (in parenthesis) of the plain PF as well as the three PSO–PF approaches, along with their corresponding p–values. The cases where PSO–PF outperformed PF are either boldfaced, if the corresponding p–value was less or equal to 0.05, or emphasized, if the corresponding p–value was larger than 0.05. In the rest of the cases, plain PF outperformed PSO–PF (with or without statistical significance).

For the case of $N = 20$ particles and system noise level equal to 0.01, we observe in Table 2 that PF outperforms in almost all cases PSO–PF. However, the picture changes as system noise increases. For system noise level 0.05, PSO–PF outperforms marginally the plain PF, although without statistical significance in most cases. The differences between the different variants of PSO–PF with respect to the value of α_1 are negligible. Further increase in the system noise level, especially from 0.3 and higher, results in clear outperformance of the plain PF by all PSO–PF variants, with statistical significance in almost all cases. In these cases, the smaller values of α_1 exhibited the best results, i.e., the smallest mean averaged RMS, while, increasing the measurement noise level resulted (as expected) to an increase in error.

Similar observations can be made also for the case of $N = 50$ particles. However, as expected, the higher number of particles improved the performance of the plain PF. Thus, in cases of both high system and measurement noise level, the plain PF had no statistically significant differences from PSO–PF. Again, we can notice that $\alpha_1 = 0.2$ was, overall, the most promising setting. The obtained results indicate

Table 2: Results for the case of $N = 20$ particles.

syst. noise	meas. noise	PF	PSO–PF $\alpha_1 = 0.2$	p-value	$\alpha_1 = 0.5$	p-value	$\alpha_1 = 0.8$	p-value
0.01	0.01	1.720 (4.169)	2.164 (3.886)	3.5e-4	3.115 (6.389)	3.5e-8	3.008 (4.473)	1.3e-9
	0.05	2.739 (4.763)	2.560 (3.284)	*3.8e-1*	2.999 (3.473)	5.3e-2	3.622 (3.941)	3.9e-4
	0.1	2.191 (3.658)	2.917 (3.531)	1.7e-4	3.182 (4.503)	1.1e-6	3.386 (2.976)	2.8e-9
	0.3	2.590 (3.507)	3.682 (3.580)	8.0e-7	3.663 (2.729)	1.2e-7	4.113 (2.568)	4.5e-10
	0.5	2.598 (3.906)	3.715 (3.968)	1.3e-6	4.356 (3.245)	6.2e-12	4.366 (2.490)	3.6e-12
0.05	0.01	2.438 (4.554)	3.065 (4.420)	1.4e-3	2.826 (4.039)	6.3e-3	3.059 (3.038)	1.5e-3
	0.05	3.831 (5.667)	2.797 (3.240)	**1.0e-2**	3.537 (2.454)	*8.9e-1*	3.324 (2.438)	*4.1e-1*
	0.1	3.811 (4.902)	3.000 (2.299)	*7.7e-2*	3.167 (3.104)	*9.8e-2*	3.447 (2.026)	*7.8e-1*
	0.3	4.128 (3.928)	4.218 (2.979)	4.2e-1	3.984 (2.684)	*9.4e-1*	4.298 (1.728)	8.9e-2
	0.5	3.712 (3.117)	4.543 (3.475)	2.7e-4	4.403 (2.464)	5.9e-4	4.634 (2.029)	1.1e-5
0.1	0.01	2.674 (4.855)	3.179 (4.049)	3.5e-3	3.273 (4.059)	3.4e-3	3.823 (4.100)	1.2e-5
	0.05	4.033 (7.261)	3.249 (3.489)	*2.3e-1*	3.338 (2.556)	*4.8e-1*	3.799 (4.019)	*8.7e-1*
	0.1	4.447 (5.756)	3.498 (2.643)	**1.3e-2**	3.748 (2.691)	*9.1e-2*	3.964 (2.181)	*3.6e-1*
	0.3	5.159 (4.520)	4.609 (3.342)	*9.4e-2*	4.577 (2.926)	*6.2e-2*	5.254 (2.802)	4.8e-1
	0.5	5.294 (4.673)	4.927 (2.499)	*3.2e-1*	4.889 (2.792)	*2.5e-1*	5.061 (2.341)	*6.6e-1*
0.3	0.01	7.350 (11.255)	5.250 (7.008)	**4.2e-6**	5.219 (7.256)	**1.4e-6**	6.224 (6.367)	**5.1e-3**
	0.05	7.297 (10.408)	5.112 (6.191)	**8.9e-7**	4.934 (5.651)	**4.4e-8**	5.824 (7.388)	**5.9e-4**
	0.1	6.933 (9.041)	4.987 (5.070)	**2.8e-7**	5.244 (4.901)	**2.5e-5**	5.756 (5.004)	**2.9e-3**
	0.3	7.967 (6.530)	6.230 (5.416)	**7.0e-7**	6.078 (5.157)	**3.5e-8**	6.835 (4.659)	**8.7e-4**
	0.5	7.628 (7.151)	6.604 (3.545)	**9.4e-3**	6.961 (6.489)	**3.7e-2**	7.019 (5.373)	*1.1e-1*
0.5	0.01	9.537 (7.172)	6.532 (11.582)	**2.3e-11**	7.152 (9.116)	**5.1e-9**	8.203 (7.997)	**1.1e-3**
	0.05	8.434 (9.217)	6.866 (9.879)	**8.4e-5**	6.989 (8.242)	**2.1e-4**	7.404 (7.850)	**5.0e-3**
	0.1	8.628 (10.319)	7.364 (11.084)	**2.2e-3**	7.112 (5.715)	**7.7e-4**	7.477 (8.023)	**6.5e-3**
	0.3	8.923 (8.451)	8.096 (6.871)	**2.7e-2**	8.038 (5.859)	**3.1e-2**	8.635 (9.104)	*3.0e-1*
	0.5	9.135 (7.197)	8.255 (6.600)	**1.5e-2**	8.256 (6.868)	**1.1e-2**	8.540 (5.375)	*1.0e-1*

that PSO–PF can be considered as a promising alternative in cases of systems with multiplicative noise, especially in high noise levels and small sample sizes.

5. CONCLUSIONS

We proposed a PF model with PSO for systems with multiplicative noise. The proposed model employs a multiobjective optimization approach to weight the likelihood and the prior, which is maximized through PSO. The resulting samples are more balanced in terms of their prior and likelihood values.

The resulting scheme is tested on a well–known test problem, with multiplicative noise. Results are promising, especially in cases of high system and measurement noise levels and small sample sizes. Further work is needed to unveil the potential of the algorithm as well as the effect of the multiobjective function to the performance of the PF.

6. ACKNOWLEDGEMENTS

The work of K.E. Parsopoulos is supported by the State Scholarship Foundation of Greece (I.K.Y.).

7. REFERENCES

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50(2):174–188, 2002.

[2] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[3] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Cambridge University, Department of Engineering, 1998.

[4] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings Sixth Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

[5] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2006.

[6] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE–Proceedings–F*, 140(2):107–113, 1993.

[7] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proc. of Aerosense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, volume Multi Sensor Fusion, Tracking and Resource Management II, Orlando, Florida, 1997.

[8] J. Kennedy. Small worlds and mega–minds: Effects of neighborhood topology on particle swarm performance. In *Proc. IEEE Congr. Evol. Comput.*,

Table 3: Results for the case of $N = 50$ particles.

syst. noise	meas. noise	PF	PSO–PF $\alpha_1 = 0.2$	p-value	$\alpha_1 = 0.5$	p-value	$\alpha_1 = 0.8$	p-value
0.01	0.01	1.167 (2.402)	1.375 (1.673)	9.9e-9	1.472 (1.292)	6.0e-10	2.049 (2.572)	3.5e-15
	0.05	1.339 (1.860)	1.528 (1.061)	2.7e-7	1.671 (1.162)	3.5e-9	2.111 (1.666)	3.2e-12
	0.1	2.128 (3.584)	2.100 (2.220)	**1.3e-2**	2.065 (1.415)	**8.8e-4**	2.351 (1.866)	6.9e-5
	0.3	2.002 (2.819)	3.180 (2.664)	7.1e-12	3.056 (2.269)	7.2e-12	3.351 (1.760)	4.1e-14
	0.5	2.120 (2.663)	3.165 (2.044)	2.4e-9	3.629 (2.682)	1.0e-13	4.053 (1.861)	7.6e-16
0.05	0.01	1.583 (4.287)	1.519 (1.114)	**6.6e-7**	1.953 (1.773)	1.7e-9	2.274 (1.544)	8.9e-12
	0.05	1.922 (2.244)	1.792 (0.851)	**2.7e-2**	2.143 (0.965)	1.4e-5	2.439 (1.000)	1.0e-8
	0.1	2.835 (3.211)	2.233 (1.101)	*2.5e-1*	2.498 (0.910)	*5.4e-1*	2.953 (1.406)	8.2e-3
	0.3	3.512 (3.257)	3.374 (1.698)	*7.2e-1*	3.572 (1.557)	2.2e-1	4.068 (1.932)	2.5e-3
	0.5	3.190 (2.155)	3.793 (2.116)	2.9e-4	3.919 (1.734)	7.6e-6	4.274 (1.787)	6.8e-9
0.1	0.01	1.784 (3.646)	1.814 (1.248)	2.7e-4	2.202 (1.827)	3.3e-6	2.721 (1.777)	7.5e-9
	0.05	2.617 (4.566)	1.916 (0.668)	*5.6e-1*	2.287 (1.208)	**2.9e-2**	2.706 (1.597)	1.1e-3
	0.1	3.057 (3.351)	2.417 (1.293)	*7.5e-2*	2.805 (1.093)	*3.0e-1*	3.027 (1.565)	*6.8e-2*
	0.3	3.493 (2.561)	3.823 (1.789)	4.8e-3	3.962 (1.612)	7.1e-4	4.127 (1.507)	1.4e-5
	0.5	4.169 (3.698)	4.230 (2.242)	2.1e-1	4.383 (1.907)	1.5e-2	4.526 (1.369)	5.0e-4
0.3	0.01	6.087 (10.694)	3.644 (3.726)	**9.9e-9**	3.704 (3.452)	**1.3e-8**	4.375 (3.326)	**4.9e-5**
	0.05	5.086 (8.615)	3.487 (2.754)	**4.2e-4**	3.847 (2.352)	**1.8e-2**	4.316 (3.008)	**2.7e-1**
	0.1	5.481 (5.204)	4.194 (3.853)	**8.0e-6**	4.570 (3.011)	**4.0e-3**	4.431 (2.395)	**5.2e-4**
	0.3	5.644 (3.815)	5.808 (5.201)	8.2e-1	5.434 (3.697)	*4.3e-1*	5.360 (3.337)	*3.1e-1*
	0.5	5.740 (3.390)	6.000 (5.186)	5.5e-1	5.791 (2.854)	6.4e-1	6.173 (2.881)	3.0e-2
0.5	0.01	7.714 (11.181)	5.283 (7.459)	**5.6e-9**	5.733 (6.037)	**4.6e-6**	6.369 (6.873)	**3.3e-3**
	0.05	7.188 (10.632)	5.423 (5.209)	**3.2e-5**	5.392 (3.772)	**2.9e-5**	6.339 (3.981)	*1.0e-1*
	0.1	6.866 (7.794)	5.906 (6.741)	**1.5e-3**	6.173 (4.152)	*5.1e-2*	6.178 (4.494)	**4.8e-2**
	0.3	7.229 (6.211)	7.479 (8.126)	5.8e-1	7.226 (6.332)	*7.8e-1*	7.493 (5.251)	2.3e-1
	0.5	7.050 (4.247)	7.690 (6.517)	8.0e-2	7.460 (6.858)	4.4e-1	7.498 (5.052)	8.3e-2

pages 1931–1938, Washington, D.C., USA, 1999. IEEE Press.

[9] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[10] R. A. Krohling. Gaussian particle swarm and particle filter for nonlinear state estimation. In *Proceeding (481) Artificial Intelligence and Soft Computing*, 2005.

[11] N. M. Kwok, W. Zhou, G. Dissanayke, and G. Fang. Evolutionary particle filter: re-sampling from the genetic algorithm perspective. In *Proc. of the IEEE 2005 Int. Conf. on Intelligent Robots and Systems*, pages 2935–2940, 2005.

[12] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proc. of the ACM 2002 Symp. on Appl. Comp.*, pages 603–607, 2002.

[13] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2–3):235–306, 2002.

[14] G. Tong, Z. Fang, and X. Xu. A particle swarm optimized particle filter for nonlinear system state estimation. In *Proc. of the IEEE 2006 Cong. on Evolutionary Computation*, pages 438–442, 2006.

[15] K. Uosaki and T. Hatanaka. Nonlinear state estimation by evolution strategies based gaussian sum particle filter. In *Lecture Notes in Computer Science*, volume 3681, pages 635–642. Springer Berlin / Heidelberg, 2005.

[16] R. Van der Merwe, A. Doucet, N. De Freitas, and E. Wan. The unscented particle filter. Technical report, Cambridge University, Department of Engineering, 2000.

[17] Q. Wang, L. Xie, J. Liu, and Z. Xiang. Enhancing particle swarm optimization based particle filter tracker. In *Lecture Notes in Computer Science*, volume 4114, pages 1216–1221. Springer Berlin / Heidelberg, 2006.