Verification of Particle Filtering Based Framework Implemented in MATLAB®

Miroslav Krupa
Department of Control and Instrumentation
Brno University of Technology, Faculty of Electrical Engineering and Communication
Brno, Czech Republic
miroslav.krupa@phd.feec.vutbr.cz

***Abstract*— Particle Filtering (PF) technique is functional approach for tracking application, navigation problem and in non-linear non-Gaussian system estimation problem. Technical diagnosis and prognosis is another promising area where PF is getting into foreground especially for Remaining Useful Life (RUL) estimation and potential fault prediction. The main reason is that PF could handle different types of noises and can handle measurement uncertainties, which are typically connected to problem of real degradation estimation. Even PF has different kind of filtering method and re-sampling algorithms there is still only limited number of real comparisons available in the state-of-the art literature, especially from the point of root mean square error and computational demandingness. Main aim of this paper is to present different type of PF implementation and compare those on simple system simulation. At the same time an object oriented MATLAB Toolbox for Particle Filtering developed by Scientific Systems Company Inc. and University of North Carolina is introduced. This toolbox covers the gap in limited number of real, configurable and robust PF implementation available to broader pool of scientists and engineers.**

***Keywords* — particle filtering framewok, particle filtering toolbox, technical prognosis, root mean square error, resampling technique, computational profiler**

I. Introduction

Fault or fault indicator prediction is a process with high uncertainty and it has been proven that Bayesian estimation techniques provides framework which can deal with such uncertainties [7], consequently it is not surprising that those techniques are finding application domains in machinery fault diagnosis and prognosis of the remaining useful life of a failing component/subsystem.

Bayesian estimation with particle filters is alternative to Kalman filters for estimating the posteriori in Bayesian framework model not limited by either linearity or Gauss noise assumption. They are also known as sequential Monte Carlo simulation methods and are particularly useful for situations where the posterior distribution is multivariate and or non-standard [5]. Particle filters are methods based on point mass (or "particle") representations of probability densities, which can be applied to any state-space model and which generalize the traditional Kalman filtering methods. Several variants of the particle filter such as Sequential Importance Re-sampling (SIR), Adaptive Sample Importance Re-sampling (ASIR) and Regularized Particle Filter (RPF) exists within a generic framework of the sequential importance sampling (SIS) algorithm [1]. In other wording particle filtering is a technique for implementing recursive Bayesian filter by Monte Carlo sampling. The main idea is to represent the posterior density by a set of random particles with associated weights. Estimate is computed based on these samples and weights. Particle filters use weighted set of samples (particles) for approximating the filtering distributions. See summarized list of PF advantages and disadvantages as per [1], [2] and [3].

PF Advantages:

- Applicable even to non-linear systems
- Adaptive focusing on probable regions of state-space
- Working for non-Gaussian noise
- Ability to represent arbitrary densities
- The framework allows for including multiple models (tracking maneuvering targets)

PF Disadvantages:

- Difficult to determine optimal number of particles
- High computational complexity (depending on number of system states)
- Number of particles increase with increasing model dimension
- Potential problems: degeneracy and loss of diversity
- The choice of importance density is the most important

PF usage for purpose of trajectory estimate was demonstrated in [2]. Battery degradation tracking and remaining useful life prediction based on particle filtering method has been demonstrated in [4]. There are many other areas, where we can find deployment of this powerful and robust approach.

II. Particle Filtering Algorithms

A dynamic system could be described as state sequence represented by Markov random process. State equation (is quite commonly available but equations are collected from [1], [7]:

$$\boldsymbol{x_k} = f_x(\boldsymbol{x_{k-1}}, \omega_k) \tag{1}$$

Where x_k is state vector at time instant k, f_x is state transition function, ω_k is process noise with known distribution. Observation equation:

$$\boldsymbol{y_k} = h_x(\boldsymbol{x_k}, \boldsymbol{v_k}) \qquad (2)$$

Where y_k is observations at time instant k, h_x is observation function, v_k is observation noise with known distribution. The main Bayesian framework objective is to estimate unknown state $\mathbf{x}_k$, based on a sequence of observations $\mathbf{y}_k$, k=0,1. In other wording find posterior distribution:

$$p(\boldsymbol{x_{0:k}}|\boldsymbol{y_{1:k}}) \qquad (3)$$

The importance normalized weights $w_k^{(i)}$ are approximations to the relative posterior probabilities of the particles such that:

$$\int f(\boldsymbol{x}_k) * p(\boldsymbol{x}_k|y_{0:k})dx_k \approx \sum_{i=1}^{P} w_k^i * f\left(\boldsymbol{x}_k^i\right),$$
$$\sum_{i=1}^{P} w_k^i = 1 \qquad (4)$$

A common problem with the SIS particle filter is the degeneracy phenomenon, where after a few iterations, all but one particle will have negligible weight [1]. There is a way how suitable measure degeneracy of weights. An effective sample size approach has been introduced in [3] an estimated effective sample size $\widehat{N_{eff}}$ is defined as:

$$\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{P}(w_k^{(i)})^2} \qquad (5)$$

Where $\widehat{N_{eff}} \le P$ indicates severe particles degeneracy. Re-sampling is used to avoid whenever severe particle degeneracy is indicated by $\widehat{N_{eff}} \le P$, the main goal as mentioned above is avoiding cases in which all but one of the importance weights are close to zero, in other wording the basic idea of is to eliminate particles that have small weights and to focus on particles with large weights [7]. See next sections with description of different particle algorithms implementation and its sequence steps.

A. Generic Particle Filter

Generic particle filter represents implementation, which includes the observations into the proposal density. This is basis implementation of SIS filter [1].

```
$[\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N] = GPF[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k]$    (6)
  • FOR i = 1 : N
      o Draw: $x_k^{(i)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$
      o Calculate: $w_k^{(i)} = p(\boldsymbol{u}_k|\boldsymbol{x}_{k-1})$
  • END FOR
  • Calculate total weight: $W = \sum_{i=1}^N w_k^i$
  • FOR i = 1 : N
      o Normalize: $w_k^{(i)} = \frac{w_k^{(i)}}{W}$
  • END FOR
  • Calculate $\widehat{N_{eff}}$
  • IF $\widehat{N_{eff}} < N_{threshold}$
      o RESAMPLE using algorithm
      o $[\{\boldsymbol{x}_k^i, w_k^i, -\}_{i=1}^N] = RESAMPLE[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{u}_k]$
  • END IF
```

B. Sample Importance Resampling (SIR) Filter

As the importance sampling density for the SIR filter is independent on measurement, the state space is explored without any knowledge of the observations [1]. Therefore, this filter can be inefficient and is sensitive to outliers. Furthermore, as re-sampling is applied to each iteration, this can result in rapid loss of diversity in particles. However, the SIR method does have the advantage that the importance weights are easily evaluated and that the importance density can be easily sampled [3] .

```
$[\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N] = SIR[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k]$    (7)
  • FOR i = 1 : N
      o Draw: $x_k^{(i)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$
      o Calculate: $w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1})$
  • END FOR
  • Calculate total weight: $W = \sum_{i=1}^N w_k^i$
  • FOR i = 1 : N
      o Normalize: $w_k^{(i)} = \frac{w_k^{(i)}}{W}$
  • END FOR
  • RESAMPLE using algorithm
      o $[\{\boldsymbol{x}_k^i, w_k^i, -\}_{i=1}^N] = RESAMPLE[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k]$
```

C. Regularized Particle Filter

Regularized particle filter (RPF) implementation focuses on avoiding the problem when all particles will collapse to a single point within a specified time period [3]. The RPF is identical to the SIR filter, except for the re-sampling stage.

```
$[\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N] = RPF[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k]$    (8)
  • FOR i = 1 : $N_s$
      o Draw: $x_k^{(i)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$
      o Calculate: $w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1})$
  • END FOR
  • Calculate total weight: $W = \sum_{i=1}^{N_s} w_k^i$
  • FOR i = 1 : $N_s$
      o Normalize: $w_k^{(i)} = \frac{w_k^{(i)}}{W}$
  • END FOR
  • Calculate $\widehat{N_{eff}}$
  • IF $\widehat{N_{eff}} < N_{threshold}$
      o Calculate Empirical covariance matrix $S_k$ of $\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N$
      o Compute $\boldsymbol{D}_k\boldsymbol{D}_k^T = S_k$
      o Resample using $\left[\{\boldsymbol{x}_k^i, \boldsymbol{w}_k^i, -\}_{i=1}^N\right] = RESAMPLE\left[\{\boldsymbol{x}_{k-1}^i, \boldsymbol{w}_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k\right]$
      o FOR i = 1 : $N_s$
          ▪ Draw $\varepsilon^i \sim K$ from Epachnikov Kernel
          ▪ $\boldsymbol{x}_k^{i*} = \boldsymbol{x}_k^i +$
```

```
                $h_{OPT}.\boldsymbol{D}_k.\varepsilon^i$
        o  END FOR
    •  END IF
```

D. Auxiliary Sampling Importance Resampling

ASIR filter is a variant of the standard SIR filter. This filter can be derived from the SIS framework by introducing an importance density $q(\boldsymbol{x}_k, i|\boldsymbol{y}_{1:k})$, which samples the pair $\{\boldsymbol{x}_k^i, i^j\}_{j=1}^N$, where i^j refers to the index of the particle at $k-1$. Algorithm steps are defined as per [1]:

```
    $[\{\boldsymbol{x}_k^i, w_k^i\}_{i=1}^N] = ASIR[\{\boldsymbol{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^N, \boldsymbol{y}_k]$
    •  FOR i = 1 : $N_s$
        o  Calculate: $u_k^i$
        o  Calculate:
           $w_k^{(i)} = q(i|\boldsymbol{y}_{1:k}).\alpha.p(\boldsymbol{y}_k|u_k^i).w_{k-1}^i$
    •  END FOR
    •  Calculate total weight: $W = \sum_{i=1}^{N_s} w_k^i$
    •  FOR i = 1 : $N_s$
        o  Normalize: $w_k^{(i)} = \frac{w_k^{(i)}}{W}$
    •  END FOR
    •  Resample using $\left[\{\boldsymbol{x}_k^i, \boldsymbol{w}_k^i, -\}_{i=1}^N\right] =$
       RESAMPLE $\left[\{\boldsymbol{x}_{k-1}^i, \boldsymbol{w}_{k-1}^i\}_{i=1}^N, \boldsymbol{u}_k\right]$
    •  FOR i = 1 : $N_s$
        o  Draw: $x_k^{(i)} \sim q(\boldsymbol{x}_k|i^j, \boldsymbol{y}_k) = p(\boldsymbol{x}_k|x_{k-1}^{i^j})$
        o  Assign weigth: $w_k^{(i)} = \frac{p(y_k|x_k^j)}{p(y_k|u_k^{ij})}$
    •  END FOR
    •  Calculate total weight: $W = \sum_{i=1}^{N_s} w_k^i$
    •  FOR i = 1 : $N_s$
        o  Normalize: $w_k^{(i)} = \frac{w_k^{(i)}}{W}$
    •  END FOR
                                                          (9)
```

E. Extended Kalman Filter

Even it is not particle filter we should mention Extended Kalman filter (EKF) as an option for non-linear solution. EKF is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. EKF is considered as default estimation technique and is quite well known. See more detailed description in [8]

III. IMPLEMENTATION AND VERIFICATION

In previous section detailed background PF. Mathworks do not offer any standard toolbox for particle filters, even there is System Identification Toolbox with Kalman filters and ARMA models. There is only limited number of real examples of PF implementation but no solid and mature approach. After detailed research the PFLib - An Object Oriented MATLAB Toolbox for Particle Filtering developed by [4] was selected for all simulation and system verification. This toolbox is product of Scientific Systems Company Inc. and University of North Carolina at Chapel Hill, developed under a United States Army Small Business Technology Transfer (STTR) project called “Advanced Computational Algorithms for Nonlinear Filtering for Real Time Environment". Toolbox could be used under GNU License, which is followed in this paper. The implementation of a filtering algorithm and its use are separated by an object oriented approach. Major algorithms and options have been implemented by authors [4] followingly:

- Simple Particle Filter
- Auxiliary Particle Filter
- Regularized Particle Filter,
- Extended Kalman Filter,

Resampling schemes includes - None, Simple (Multinomial), Residual and Systematic. Other available parameter choices include sampling, frequency, number of particles, and specifation of Jacobians for EKF needs.

A. Comparison of PF algorithms

As a part of particle filter verification and familiarization with new filtering technique a simple one dimensional system model has been selected in a form:

$$x_k = 0.5 * x_{k-1} + \frac{25 * x_{k-1}}{1 + {x_{k-1}}^2}$$

$$y_k = \frac{{x_k}^2}{20} \qquad (1)$$

Simulation has been performed with following variances with Gauss Noise Distribution:

- Different Filter Types (Simple, Auxiliary, Regularized, EKF)
- Different parameters ω process noise distribution mean, variance
- Different Re-sampling
- Different number of particles

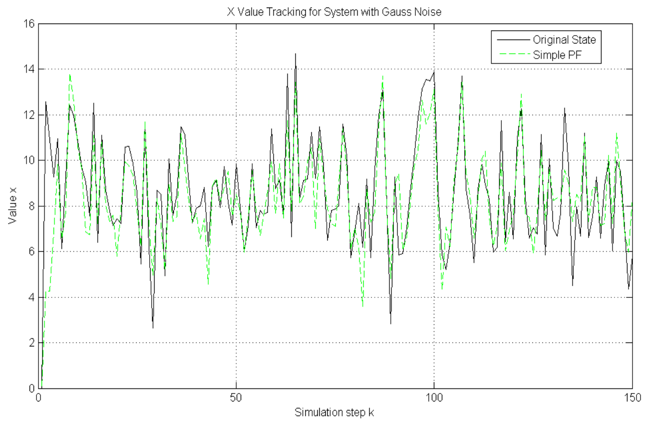

Figure 1. Filtering of a system represented by Equation 2 for simple SIR, N_s=60, Residual re-sampling, Resample every 2 samples

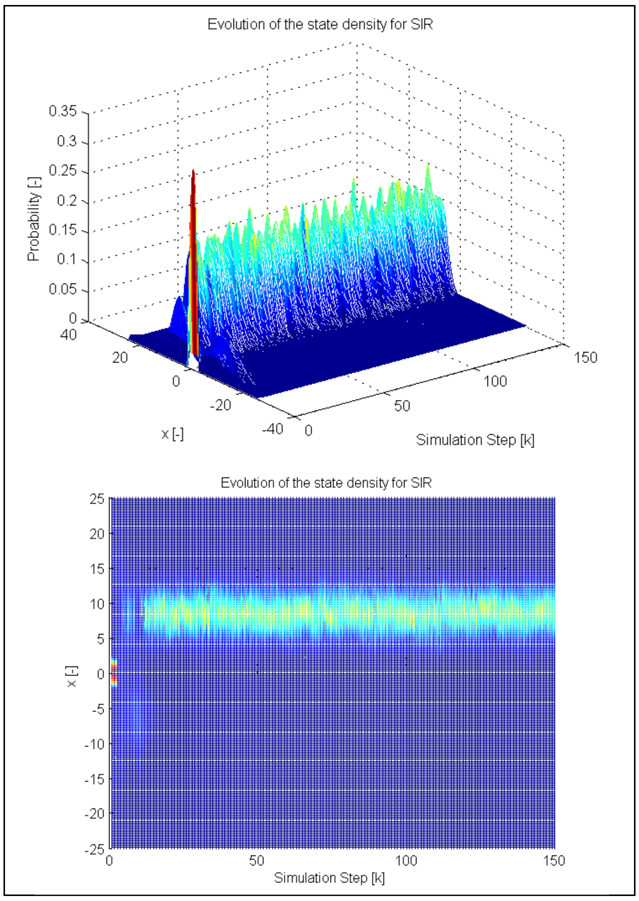

Figure 2. Evolution of State Density Represented by Particles for SIR filter, Ns=60, Residual re-sampling, Resample every 2 samples

We can see in previous figures simulation results and evolution of particles state desnity during simulation. Method of comparing RMSE has been chosen to numerically compare each PF methods and it ability to track internal states. See next figure 3, with RMSE evolution during simulation.

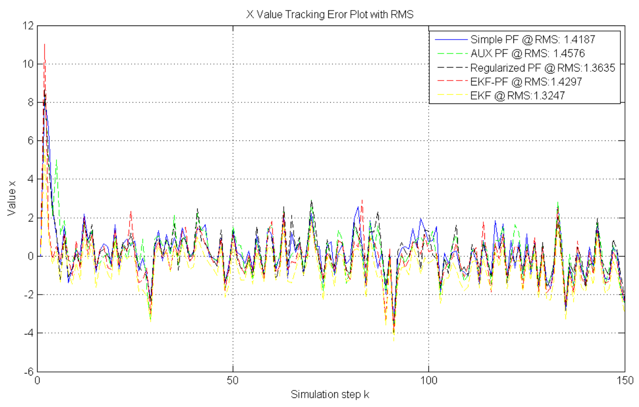

Figure 3. RMSE of Estimation for Each PF algorithms one particular run

In next table you can see comparison of root mean square error (RMSE) for different number of particles (EKF is mentioned just for completeness). RMSE or sometimes called as a Root Mean Square Deviation (RMSD) is quite common metric for comparison estimators and is usually defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_{1,i} - x_{2,i})^2}{N}} \qquad \text{(III-2)}$$

Where $x_{1,i}$ is simulated system state and $x_{2,i}$ is estimated value.

TABLE I. COMPARISON OF DIFFERENT IMPLEMENTATION OF PARTICLE FILTERS

Filter Type / Number of Particles	Simple PF RMSE[-]	AUX PF RMSE[-]	Regularized PF RMSE[-]	EKF RMSE[-]
	Residual resampling every 3steps			
N_s=15	1.9613	1.8417	1.5772	1.3247
N_s=30	1.4558	1.4431	1.3601	
N_s=60	1.9238	1.4217	1.3415	
N_s=120	1.7281	1.7168	1.4394	
Average	**1.7673**	**1.6058**	**1.4296**	**1.3247**
	Residual resampling every 1 step			
N_s=15	1.7119	2.4440	1.2842	1.3247
N_s=30	1.4777	1.8479	1.4589	
N_s=60	1.5525	10.695	1.9160	1.3247
N_s=120	1.7372	1.9304	1.7282	
Average	**1.6198**	**4.2293**	**1.5968**	**1.3247**
	Simple resampling every 3steps			
N_s=15	2.4283	2.6377	1.2887	1.3247
N_s=30	1.6694	1.2549	1.7117	
N_s=60	1.7376	1.3906	1.8065	
N_s=120	1.8542	1.9901	1.6934	
	1.9224	**1.8183**	**1.6251**	**1.3247**
	Systematic resampling every 1step			
N_s=15	1.2458	1.3018	1.5780	1.3247
N_s=30	1.3729	1.5634	2.1945	
N_s=60	1.5984	1.3867	1.3793	
N_s=120	1.6572	1.0777	1.3531	
	1.4686	**1.3324**	**1.6262**	**1.3247**

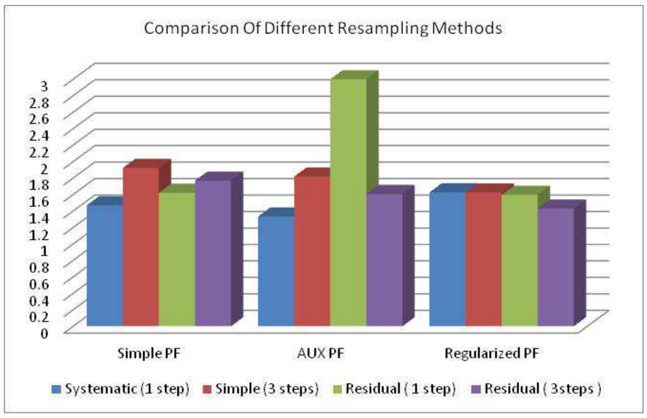

Figure 4. Comparison of RMSE for different resampling methods

MATLAB software provides the Profiler functionality, which enables to evaluate execution time of each algorithms including its sub-functions. This tool is quite helpful in algorithms development and optimizing. It provides information not only about execution time but even the number of calls. See results of profiling for different particle filters

TABLE II. COMPARISON OF EXECUTION TIME OF DIFFERENT PARTICLE FILTER ALGORITHMS (MEASURED TIME FOR 150 STEPS)

Filter Type	Simple PF Exec Time[ms]	AUX PF Exec Time[ms]	Regularized PF Exec Time[ms]	EKF Exec Time[ms]
Number of Particles				
	Residual resampling every 3steps			
N_s=15	1.3	2.8	1.4	0.1
N_s=30	2.8	5.7	2.6	0.1
N_s=60	5.7	9.7	5.5	0.1
N_s=120	11.0	19.5	11.3	0.1
Avg per part.	**0.093**	**0.168**	**0.092**	**0.002**
	Residual resampling every 1 step			
N_s=15	1.6	2.7	1.6	0.1
N_s=30	2.5	5.9	2.3	0.1
N_s=60	7.9	8.3	7.5	0.0
N_s=120	10.8	20.2	10.7	0.2
Avg per part.	**0.102**	**0.164**	**0.098**	**0.002**
	Simple resampling every 3steps			
N_s=15	1.9	2.4	1.4	0.1
N_s=30	2.6	5.4	2.6	0.1
N_s=60	5.3	10.0	4.8	0.1
N_s=120	11.5	19.6	11.5	0.1
Avg per part.	**0.094**	**0.166**	**0.090**	**0.002**
	Systematic resampling every 1 steps			
N_s=15	1.7	2.2	2.0	0.1
N_s=30	3.0	5.2	2.6	0.3
N_s=60	4.3	12.8	4.3	0.1
N_s=120	11.4	19.4	11.7	0.1
Avg per part.	**0.090**	**0.176**	**0.091**	**0.003**

As we can see from results in Table II the most time consuming execution is in case of implemented AUX algorithm. The rest of the execution seems to be proportionally dependent on number of particles as expected in theory.

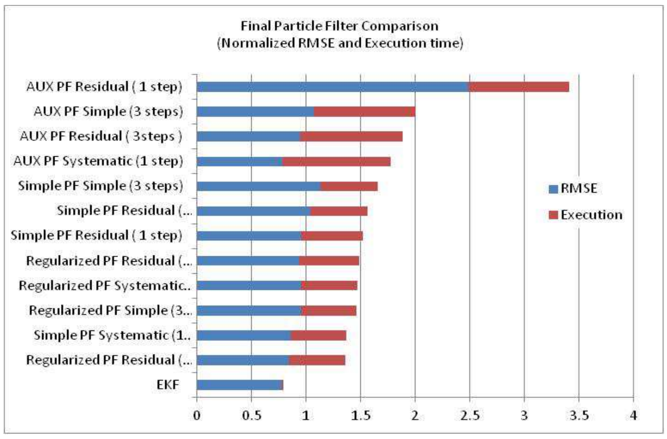

Figure 5. Final Particle Comparison considering time execution and RMSE

Just for completeness a testing for Non-Gaussian distribution was performed. Gamma distribution (a = 2, b = 1) was used for simulation of internal states and the results were similar to previous utilizing Gaussian PDF. From embedded point of view there is a room for algorithms optimization. Non Particle approach EKF is still promising and is considered as a default approach. What has to be mentioned in support of promising PF is fact that initialization portion of the internal state estimates takes several steps, which cause worse RMSE. It has been verified than in longer simulation run the results nearly overcomes EKF.

IV. CONCLUSION

Different types of particle filters are presented including detailed description of algorithms. Simple, Regularized and Auxiliary particle filters are tested together with well known and established Extended Kalman Filter. Based on the results it is indicated that EKF still overcomes PF from execution time point of view in case of Gaussian noise. An object oriented MATLAB Toolbox for Particle Filtering has been verified and it is proven that implementation of particle filtering toolbox is ready for use and could be utilized not only for prognostics approaches. In future work possibility of RUL estimate based on particle filtering technique will be verified and next level of PF comparison tests will be performed with more complex multidimensional non-linear systems .

ACKNOWLEDGMENT

This paper was supported by Honeywell, spol s r.o. – HTS CZ o.z., Aerospace division, Condition Based Maintenance group. Author declares that the pieces of information herein published are solely the result of his personal research and do not reflect the views, attitudes and intentions of Honeywell.

I would like to thank to my consultants – associated professors Ludvik Bejcek and Petr Benes from Brno University of Technology for their valuable comments and support. This paper was prepared as a part of the author's PhD thesis related to Technical Prognostics Methods Applicable to Embedded Systems.

REFERENCES

[1] ARULAMPALAM, M.S.; MASKELL, S.; GORDON, N.; CLAPP, T.; A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing, Volume: 50, Issue: 2.* February 2002. Page(s): 174 – 188, ISSN : 1053-587X.

[2] BAAR-SHALOM, Y.; LI RONG X.; KIRUBARAJAN T.; Estimation with Applications to Tracking and Navigation. Theory Algorithms and Software. John Willey & Sons, NJ. 2001. ISBN 0-471-41655-X.

[3] BERGMAN, N.; *Recursive Bayesian estimation: Navigation and tracking applications*, Ph.D. dissertation, Linköping Univ., Linköping, Sweden,1999, Dissertation No: 579, Pages

[4] CHENA, L.; LEEB, CH.; BUDHIRAJAB, A.; MEHRAA, R. K.PFLib - An Object Oriented MATLAB Toolbox for Particle Filtering. User Manual. <http://www.stat.colostate.edu/~chihoon/paper-6567-25-revised.pdf> [Retrieved 4/22/.2012]

[5] DOUC, R.; CAPPE, O.; MOULINES, E.; Comparison of Resampling Schemes for Particle Filtering. *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (2005)*, Pages 64-69, ISSN : 1845-5921 Print ISBN: 953-184-089-XCAPPE

[6] SAHA, B., GOEBEL, K.; CHRISTOPHERSON, J.;Comparison of Prognostic Algorithms for Estimating Remaining Useful Life of Batteries, Transactions of the Institute of Measurement and Control vol. 31 no. 3-4, June/August 2009. Pages 293-308. ISSN**:** 0142-3312

[7] VACHTSEVANOS, G.; LEWIS, F.; ROEMER, M.; HESS, A.; WU, B.; Intelligent Fault Diagnosis and Prognosis for Engineering Systems. John Wiley & Sons, Inc. New Jersey, 2006. Pages 434. ISBN:978-0-471-72999-0.

[8] WELCH, G; BISHOP, G.; An Introduction To The Kalman Filter. Online Publication. <http://clubs.enscachan.fr/krobot/old/data/positionnement/kalman.pdf> [retrieved: 4/22/2012]