19. Resampling – essence of particle filter

Piotr Kozierski[1)], Marcin Lis[2)], Andrzej Królikowski[3)], Adam Gulczyński[4)]

19.1. Introduction

Particle Filter (PF) based on the Monte Carlo method is sometimes called Sequential Monte Carlo method (SMC). However [Doucet, Johansen 2009] notes that SMC is the wider topic than PF. Particle prediction and particle smoothing are another algorithms belonging to SMC. In smoothing, to estimate value of state variable x_k measurements from further time steps Y_{k+K} are used. Whereas in prediction based on measurements to the current time Y_k the future value of state variable x_{k+K} is estimated. In filtering estimation of state variable value is based on measurements in the same moment Y_k. This chapter is dedicated to filtering.

Particle Filter history dates back to the mid-twentieth century, when Norbert Wiener proposed something similar to particle filter, but only in the 80`s computing power has enabled for further work in this direction [Simon 2006]. The breakthrough came in 1993, when Gordon, Salmond and Smith proposed in [Gordon et al. 1993] algorithm, which has been devoid the biggest flaw – degeneration (the algorithm is described in subchapter "Bootstrap Filter"). The uniqueness of this algorithm is due the use of resampling.

19.2. Sequential Bayesian Filter

The operation principle of PF is based on Bayes theorem

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} \tag{1}$$

i.e. on the theorem on conditional probability (probability of the event A, given the event B), where A and B are some random variables. To use Bayes theorem, state variables and measurements must be taken as random variables with concrete probability densities:

$$x_k \sim p(x_k \mid x_{k-1}) \tag{2}$$

$$y_k \sim p(y_k \mid x_k) \tag{3}$$

Expression (2) means that state variable is a random variable with conditional probability density function (PDF), where $p(x_k \mid x_{k-1})$ is the transition model. Measurement also is random variable with conditional PDF, where $p(y_k \mid x_k)$ means measurement model.

With these assumptions one can write Bayes theorem for state variables and measurements

$$p(X_k \mid Y_k) = \frac{p(Y_k \mid X_k)p(X_k)}{p(Y_k)} \tag{4}$$

where it was assumed that

$$X_k = \{x_1, x_2, ..., x_k\} \tag{5}$$

$$Y_k = \{y_1, y_2, ..., y_k\} \tag{6}$$

Equation (4) concerns joint density function, however in practice more often marginal density $p(x_k \mid Y_k)$ is estimated. It is also assumed that system model is a Hidden Markov Model (HMM) – it means that values of state variables depends only on values of state variables in previous time step and values of measurements depends only on values of state variables in current time step

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{7}$$

$$y_k = h_k(x_k, n_k) \tag{8}$$

where v_{k-1} is a process noise, whereas n_k is a measurement noise. It is assumed that PDF of all noises are known. System model (7-8) also is given. It should be noted that knowledge about system and measurements models (2-3) is equivalent to knowledge about expressions (7-8).

Using above assumptions one can obtain equation describing Sequential Bayesian Filter (SBF)

$$p(x_k \mid Y_k) = \frac{p(y_k \mid x_k) p(x_k \mid Y_{k-1})}{p(y_k \mid Y_{k-1})} \tag{9}$$

The main task of filter (9) is posterior PDF $p(x_k \mid Y_k)$ estimation. There are also likelihood $p(y_k \mid x_k)$, prior PDF $p(x_k \mid Y_{k-1})$ and evidence $p(y_k \mid Y_{k-1})$ (normalizing coefficient).

Prior can be written as integral

$$p(x_k \mid Y_{k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid Y_{k-1}) dx_{k-1} \tag{10}$$

in which $p(x_{k-1} \mid Y_{k-1})$ is a posterior from previous time step, and PDF $p(x_k \mid x_{k-1})$ is given by transition model (2). Since the evidence is a number, equation (9) can be written as

$$p(x_k \mid Y_k) \propto p(y_k \mid x_k) p(x_k \mid Y_{k-1}) \tag{11}$$

where symbol "$\propto$" means "directly proportional".

In Sequential Bayesian Filter (9) there are 2 main steps. First it must be calculated prior PDF (10) – this is prediction step. In second step there is calculated the posterior PDF (11) – this is update step.

Presented derivation is strongly condensed – full derivation can be found in [Candy 2009] and [Kozierski, Lis 2012].

19.3. Particle filter

Particle filter principle of operation is the same as SBF. Difference between them lies only in the posterior representation – in PF it is a set of particles, which are composed of values x_k^i and weights q_k^i. With this approach calculations can be separately performed for each sample. This provides opportunity to implement parallel computations and speeds up algorithms (see [Mountney et al. 2011, Sutharsan et al. 2012]).

Particle filter is used in many different areas, such as robotics (robot localization problem [Thrun 2002, Woo et al. 2006]), image processing (object tracking [Chang et al. 2005]) and identification (estimation of system parameters [Poyiadjis et al. 2005, Schön et al. 2011]).

PF algorithm based on Importance Sampling (IS) method. IS assums that there are two probability density functions:

- $f_{IS}(x)$ – from which should be draw, but this is difficult (or impossible),
- $g_{IS}(x)$ – from which it is easy to draw.

PDF $g_{IS}(x)$ is used to draw samples, and by assigning weights to the values, one can obtain PDF $f_{IS}(x)$ properties. Weights should be proportional to the ratio

$$w_{IS}^i \propto \frac{f_{IS}(x^i)}{g_{IS}(x^i)} \tag{12}$$

so that more often drawn values from $g_{IS}(x)$ have respectively less weights. Operation of method is shown in Fig.1.

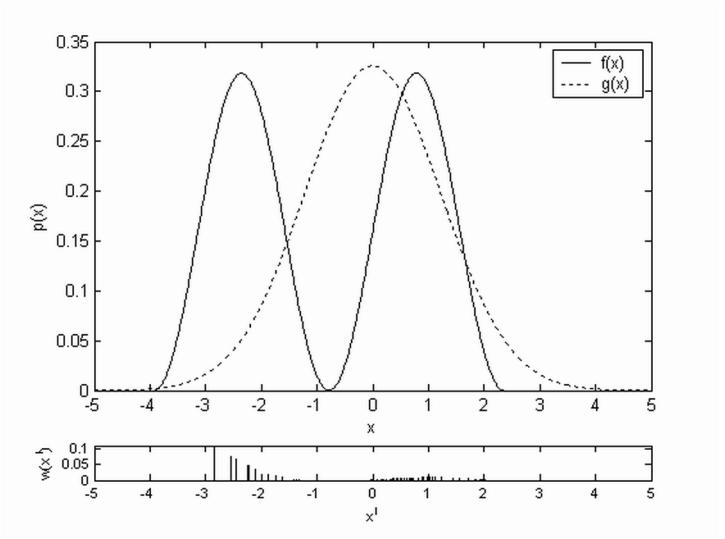

Fig.1. Principle of Importance Sampling method operation

19.3.1. Sequential Importance Sampling

Sequential Importance Sampling (SIS) method is basic variation of PF, but unfortunately resampling absence causes that it is completely useless.

Using the importance sampling method (12) and SBF (9) one can derived expression for particle weight values (derivation of expression was abandoned, however it can be found in many articles, e.g. [Arulampalam et al. 2002, Kozierski, Lis 2012]).

$$q_k^i \propto q_{k-1}^i \frac{p\left(y_k \mid x_k^i\right) p\left(x_k^i \mid x_{k-1}^i\right)}{g\left(x_k^i \mid x_{k-1}^i, y_k\right)} \tag{13}$$

where q_k^i is weight of i-th particle in time step k, $g(\cdot)$ is a PDF used to draw (as in IS method). The algorithm of the SIS method is presented below.

Algorithm 1 (SIS method)

1. Draw N particles from initial PDF $x_0^i \sim p(x_0)$, set initial weights $q_0^i = \frac{1}{N}$, set time step $k = 1$.
2. Draw N particles from proposed importance density $x_k^i \sim g\left(x_k \mid x_{k-1}^i, y_k\right)$.
3. Calculate weights according to formula

$$\tilde{q}_k^i = q_{k-1}^i \frac{p\left(y_k \mid x_k^i\right) p\left(x_k^i \mid x_{k-1}^i\right)}{g\left(x_k^i \mid x_{k-1}^i, y_k\right)} \tag{14}$$

4. Normalize weights

$$q_k^i = \frac{\tilde{q}_k^i}{\sum_{j=1}^{N} \tilde{q}_k^j} \tag{15}$$

5. Increase time step $k = k + 1$, go to 2nd algorithm step.

Note that importance density $g\left(x_k^i \mid x_{k-1}^i, y_k\right)$ may be dependent on state variable value from previous step x_{k-1}^i, and also on measurement value y_k. However, it does not mean that this PDF must depend on both of these values. Importance density may depend only on one of these values (this case was used in Bootstrap Filter), and can also be completely independent (but then algorithm performance will be much worse).

The biggest disadvantage of SIS is that after few time steps degeneration occurs. In this case, all particle weights, except one, have values close to zero – see Fig.2.

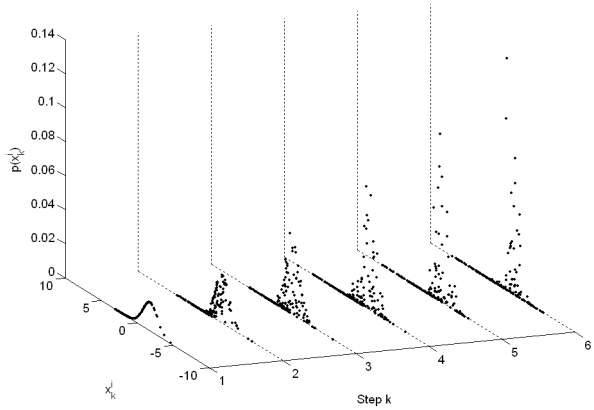

Fig.2. SIS degeneration – first 6 steps evaluation of algorithm 1

The main problem results from expression (14) in which weights depends on weights from previous step. If particle weight is equal to zero, in all subsequent steps it will also be zero, so the particle will be useless. To continue to use it, the "reset" is required, e.g. using resampling.

19.3.2. Sequential Importance Resampling

Sequential Importance Resampling method (SIR) has been extended, in comparison to the SIS, by only two steps – one is check that if resampling is necessary, second is execution of resampling. SIR principle of operation is shown in Algorithm 2.

Algorithm 2 (general SIR method)

1. Draw N particles from initial PDF $x_0^i \sim p(x_0)$, set initial weights equal to $q_0^i = \frac{1}{N}$, set time step $k = 1$.
2. Draw N particles from proposed importance density $x_k^i \sim g\left(x_k \mid x_{k-1}^i, y_k\right)$.
3. Calculate particle weights according to equation (14).
4. Normalize weights according to (15).
5. Check condition of resampling – if not satisfied go to step 7 of algorithm.
6. Do resampling.
7. Increase time step $k = k + 1$, go to step 2 of algorithm.

More information about resampling and condition of resampling will be presented later in this chapter.

The task of particle filter is posterior estimation. However, using PF as observer, more needed is state variable estimation – usually obtained by calculating expected value of posterior PDF [Zieliński 2007]

$$\hat{x}_k = E[x_k] = \sum_{i=1}^{N} x_k^i \cdot p(x_k^i) = \sum_{i=1}^{N} x_k^i \cdot q_k^i \tag{16}$$

It should be found between the steps 6 and 7 of Algorithm 2.

It was assumed that algorithms with resampling are included in family of SIR methods, and algorithms without resampling – family of SIS methods. Therefore, the vast majority of PF based on SIR method, although operation principle of some PF algorithms may differ from Algorithm 2. Examples of such filters are Auxiliary PF [Pitt, Shephard 1999], Rao-Blackwellised PF [Doucet et al. 2000, Handeby et al. 2010], Distributed PF [Bashi et al. 2003], Gaussian PF [Kotecha, Djurić 2003], Unscented PF [Merwe et al. 2000], Linearized PF [Candy 2009], Multiple Model PF [Doucet et al. 2001] and many others.

19.3.3. Bootstrap Filter

Bootstrap Filter (BF) is one of the standard SIR varieties. Was proposed in 1993 by Gordon, Salmond and Smith [Gordon et al. 1993]. Transition model is proposed as importance density in this method

$$g(x_k \mid x_{k-1}, y_k) = p(x_k \mid x_{k-1}) \tag{17}$$

This choice is the best, if one takes into account the minimization of conditional weight variance [Brzozowska-Rup, Dawidowicz 2009]. It also causes simplification formula (14), which takes the form

$$\tilde{q}_k^i = q_{k-1}^i \cdot p(y_k \mid x_k^i) \tag{18}$$

However assumption (17) also causes negative effects, because transition model is independent from measurements, which may result in lack of resistance to outliers [Brzozowska-Rup, Dawidowicz 2009].

The second assumption in BF algorithm is abandonment of resampling condition (step 5 of Algorithm 2). Additionally, after each resampling all weights of new particles are set to $\frac{1}{N}$, and since all weights are equal, equation (18) may be written as

$$\tilde{q}_k^i = p(y_k \mid x_k^i) \tag{19}$$

BF algorithm is shown below.

Algorithm 3 (Bootstrap Filter)

1. Draw N particles from initial PDF $x_0^i \sim p(x_0)$, set initial weight values equal to $q_0^i = \frac{1}{N}$, set initial time step $k = 1$.
2. Draw N new particles from proposed importance density $x_k^i \sim p(x_k \mid x_{k-1}^i)$.
3. Calculate particle weights in accordance with (19).
4. Normalize weights (15).
5. Resampling
6. Increase time step $k = k + 1$, go to step 2 of algorithm.

19.4. Resampling

As has been shown, PF needs resampling to work properly. Degeneration does not occur, because particles with low weights are replaced with others (with higher weights). Thrun rightly pointed out that the resampling can be compared to probabilistic implementation of Darwin`s theory of evolution by natural selection [Woo et al. 2006].

But resampling also introduces few disadvantages, e.g. after this step of algorithm particles are not independent any more. The second problem is related to the computation time – resampling is the most complex step in algorithm (all other steps are linearly dependent on the number of particles). Thus, the attention will be given to computational complexity of individual resampling methods.

The purpose of resampling is random selection of new set of particles based on the current posterior – the greater particle weight is, the more likely that this particle value will be drawn. All new particles are assigned a weight $q_k^i = \frac{1}{N}$.

To determine whether resampling is required, one should find Effective Sample Size (ESS). However accurate value can not be calculate, so expression for estimated value is used [Doucet et al. 2000]

$$\hat{N}_{ESS} = \frac{1}{\sum_{i=1}^{N} \left(q_k^i\right)^2} \tag{20}$$

ESS can be interpreted as minimum particles number needed to submit posterior PDF. Therefore if a lot of particle weights are close to zero, then $\hat{N}_{ESS}$ value decreases (particles with low weights can be omitted). $\hat{N}_{ESS}$ ranges from 1 (one particle has weight equal to 1, and all others particle weights are equal to 0) to N (after resampling, when all weights are equal).

It should be taken a certain threshold value N_T. If ESS is less than N_T, then resampling is required. Typically the threshold is set at half of particles number $N_T = \frac{N}{2}$ [Doucet, Johansen 2009].

19.4.1. Multinomial resampling

In multinomial resampling N particles should be drawn from posterior PDF assuming that chance to choice the value x_k^i is exactly q_k^i (weights are normalized, and so their sum is 1). This algorithm is simple and most often cited in the literature.

Algorithm 4 (multinomial resampling)

1. Prepare discrete cumulative distribution function (CDF) $S_k^{1:N}$ based on particle weights, so that $S_k^1 = q_k^1$ and $S_k^N = 1$.
2. For $i = 1,...N$ perform steps 3-5.
   3. Draw value from uniform distribution $d \sim U(0,1)$, set variable value $j = 1$.
   4. As long as $S_k^j < d$ increment variable $j = j + 1$.
   5. Remember drawn value $\breve{x}_k^i = x_k^j$.
6. The old set of samples replace by saved values $x_k = \breve{x}_k$, set new weights for $i = 1,...,N: \; q_k^i = \frac{1}{N}$.

Algorithm 4 is simple to implement, but unfortunately computational complexity is $O(N^2)$. This notation applies "asymptotic upper bound" and means that in the worst case algorithm execution time is proportional to the square of particles number [Cormen et al. 2004] (double the number of particles will result in a fourfold increase in computation

time). Quadratic computational complexity is not a good result, and it is caused by linear search in step 4 of Algoritm 4. This step can be replaced by binary search, which will speed up resampling and reduce the complexity to $O(N\,lg\,N)$. Multinomial resampling algorithm with binary search is shown below.

Algorithm 5 (multinomial resampling with binary search)

1. Prepare discrete CDF $S_k^{1:N}$ based on particle weights, so that $S_k^1 = q_k^1$ and $S_k^N = 1$.
2. For $i = 1,...N$ perform steps 3-7.
   3. Draw value from uniform distribution $d \sim U(0,1)$, set variable values $j = \left[\frac{N}{2}\right]^{round}$ and $r = \frac{N}{4}$.
   4. As long as $\left(S_k^j < d\right)$ or $\left(\left(j > 1\right) AND \left(S_k^j > d\right) AND \left(S_k^{j-1} \geq d\right)\right)$ perform steps 5-6
      5. If $S_k^j < d$ then $j = \left[j + r\right]^{round}$, and if not $j = \left[j - r\right]^{round}$.
      6. If $r > 1$ then $r = \frac{r}{2}$, and if not $r = 1$.
   7. Remember drawn value $\breve{x}_k^i = x_k^j$.
8. The old set of samples replace by saved values $x_k = \breve{x}_k$, set new weights for $i = 1,...,N:\ q_k^i = \frac{1}{N}$.

Steps 3-6 are proposed implementation of binary search without recurrence function. Variable r means "move" which will be executed by variable j and every iteration this "move" is reduced by half. After the loop 4-6 variable j satisfies inequality $S_k^{j-1} < d \leq S_k^j$.

In [Launay et al. 2012] authors proposed to first draw all random values $d_{1:N}$, then sort them, and having sorted values $d_{1:N}^{sort}$ one can select N new values. However the sort complexity is $O(N\,lg\,N)$, therefore the approach taken in [Launay et al. 2012] has the same computational complexity as the Algorithm 5.

19.4.2. Systematic resampling

In this resampling method it is assumed that uniform distribution is divided into N equal parts, and from each part there is drawn only 1 random value. This allows each successively drawn value to be greater than the previous one, and it is possible to implement resampling with linear complexity $O(N)$. Operation of resampling presents Algorithm 6.

Algorithm 6 (systematic resampling)

1. Prepare discrete CDF $S_k^{1:N}$ based on particle weights, so that $S_k^1 = q_k^1$ and $S_k^N = 1$.
2. Set initial variable value $j = 1$.
3. For $i = 1,...N$ perform steps 4-6.
   4. Draw random value from partial uniform distribution $d \sim U\left(\frac{i-1}{N}, \frac{i}{N}\right)$.

5. As long as $S_k^j < d$ increase variable $j = j + 1$.
6. Remember drawn value $\breve{x}_k^i = x_k^j$.

7. The old set of samples replace by saved values $x_k = \breve{x}_k$, set new weights for $i = 1,...,N:\ q_k^i = \frac{1}{N}$.

19.4.3. Residual resampling

Residual resampling is also called remainder resampling [Douc et al. 2005]. In this method there is assumed that for all particles which weights are greater than $\frac{1}{N}$, new particles are arbitrarily "drawn" – the greater weight, the more copies of particle. But this is the way to select only a part of particles, and the rest needs to be drawn using previous methods (multinomial resampling with binary search was used). Resampling principle of operation is shown in Algorithm 7.

Algorithm 7 (residual resampling)

1. Set value of variable $N_r = N$.
2. For $i = 1,...N$ perform steps 3-5.
3. Calculate $m^i = \lfloor N \cdot q_k^i \rfloor$ and $N_r = N_r - m^i$.
4. for $j = 1,...m^i$ add particle to the "drawn" $\hat{x}_k = \left[\hat{x}_k, x_k^i\right]$.
5. If $m^i > 0$, then $\breve{q}_k^i = N \cdot q_k^i - m^i$, and if not $\breve{q}_k^i = q_k^i$.
6. Normalize weights $\bar{q}_k^i = \breve{q}_k^i \Big/ \sum \breve{q}_k^j$
7. Prepare discrete CDF $S_k^{1:N}$ based on particle weights (compute in step 6), so that $S_k^1 = \bar{q}_k^1$ and $S_k^N = 1$.
8. For $i = 1,...N_r$, using $S_k^{1:N}$ perform resampling (Algorithm 5) and add drawn particles to $\hat{x}_k$.
9. The old set of samples replace by saved values $x_k = \hat{x}_k$, set new weights for $i = 1,...N:\ q_k^i = \frac{1}{N}$.

One can see that the greater weight values (the smaller ESS value), the better for algorithm, because there will be fewer particles to draw with multinomial resampling. Algorithm 5 was used, so Algorithm also has computational complexity $O(N\ lg\ N)$.

19.4.4. Evolutive resampling

The new approach to resampling in this chapter is proposed and assumed that resampling applies only to selected particles with low or zero weight. One wants to combine advantages of normal resampling (reset particles with low weight) and high speed method (reducing the number of particles which must be resampled). Operation principle of this resampling is presented in Algorithm 8.

Algorithm 8 (evolutive resampling)

1. Prepare discrete CDF $S_k^{1:N}$ based on particle weights, so that $S_k^1 = q_k^1$ and $S_k^N = 1$, set threshold Q_T, set initial value of variable $p_{om} = 0$.

2. For $i = 1,...N$ check if the weight is fewer then threshold Q_T, and if yes, then increase variable $p_{om} = p_{om} + 1$ and add index number to array $r_{es} = [r_{es}, i]$.
3. Set variable $j = 1$.
4. For $i = 1,...p_{om}$ perform steps 5-7.
   5. Draw random value from partial uniform distribution $d \sim U\left(\frac{i-1}{p_{om}}, \frac{i}{p_{om}}\right)$.
   6. As long as $S_k^j < d$ increment variable $j = j + 1$.
   7. Remember drawn value $x_k^{r_{es}^i} = x_k^j$, set new weight $q_k^{r_{es}^i} = \frac{1}{N}$.
8. Normalize weights q_k.

One can see that all these steps are linearly dependent on the particles number, so computational complexity is $O(N)$. Proposed name refers to the theory of evolution by natural selection, because only particles with small weights are subjected to resampling, therefore particles with the worst adaptation.

19.4.5. Notes

In the literature one can find several other proposals for resampling, for example, Local Monte Carlo Resampling [Liu, Chen 1998], Stratified Resampling [Douc et al. 2005], Metropolis Resampling and Rejection Resampling [Murray et al. 2013]. Sometimes it is also used so-called MCMC step (Markov Chain Monte Carlo step), which is performed after resampling [Launay et al. 2012].

Further acceleration of the Algorithm 8 is possible, for example by omitting weights normalization in step 8. Note that the estimated value (16) must be normalized (divided by the sum of weights). In the next step, it would not be noticeable, because weights have to be normalized after the calculation (15). This saves time required for $N - 1$ divisions (in every time step).

19.5. Simulation results

System used for simulation is described by equations

$$x_k = 0.8x_{k-1} + \frac{e^{0.1x_{k-1}}}{0.1x_{k-1}^2} \cdot v_{k-1} \tag{21a}$$

$$y_k = x_k + n_k \tag{21b}$$

where v and n are random variables normally distributed with variances equal respectively to 0.2 and 0.1. Each simulation was consisted of $M = 10000$ time steps.

Simulation results are shown in Tab.1 and Tab.2. Tracking performance was evaluated based on Mean Square Error (MSE). Computation time t and resamplings number R_{es} also were measured. In the case of evolutive resampling R_{es} was calculated as the number of all “small resamplings” (single particle resamplings) divided by N.

Bootstrap Filter (first row in Tab.1) and SIR (all other results) algorithms were used in simulations. System model was proposed as importance density in SIR.

19.6. Conclusions and summary

Comparing results in Tab.1, one can see that in all cases the fastest is systematic resampling. MSE and R_{es} are almost identical for different resamplings, thus one can

conclude that MSE and R_{es} are independent of resampling type, and dependent only on number of particles N and threshold N_T.

Looking for the best value of N_T one can see, that for $N_T = 0.25N$ results are almost the same as for resampling in each time step. However, it should be noted that although the resampling was performed fewer times, the calculation time is the same – to

Tab.1. Simulation results for multinomial, systematic and residual resamplings with different particle number N and threshold N_T

		Resampling					
		N = 500			N = 200		
		Multinomial	Systematic	Residual	Multinomial	Systematic	Residual
CONDITION FOR RESAMPLING	each iteration	MSE = 0.0915 t = 11.38s	MSE = 0.0913 t = 6.14s	MSE = 0.0913 t = 8.01s	MSE = 0.1197 t = 4.63s	MSE = 0.1204 t = 2.74s	MSE = 0.1245 t = 3.69s
	N_T = 0.7N	MSE = 0.0930 t = 11.50s (R_{es} = 9691)	MSE = 0.0914 t = 6.36s (R_{es} = 9692)	MSE = 0.0913 t = 8.26s (R_{es} = 9693)	MSE = 0.1197 t = 4.60s (R_{es} = 9690)	MSE = 0.1329 t = 2.87s (R_{es} = 9690)	MSE = 0.1295 t = 3.83s (R_{es} = 9687)
	N_T = 0.5N	MSE = 0.0916 t = 11.27s (R_{es} = 9316)	MSE = 0.0917 t = 6.25s (R_{es} = 9314)	MSE = 0.0914 t = 7.99s (R_{es} = 9318)	MSE = 0.1301 t = 4.62s (R_{es} = 9312)	MSE = 0.1205 t = 2.82s (R_{es} = 9322)	MSE = 0.1272 t = 3.73s (R_{es} = 9316)
	N_T = 0.25N	MSE = 0.0925 t = 10.48s (R_{es} = 7989)	MSE = 0.0915 t = 6.10s (R_{es} = 7984)	MSE = 0.0923 t = 7.40s (R_{es} = 7991)	MSE = 0.1299 t = 4.35s (R_{es} = 7979)	MSE = 0.1239 t = 2.79s (R_{es} = 7979)	MSE = 1.301 t = 3.50s (R_{es} = 7980)
	N_T = 0.1N	MSE = 0.0965 t = 9.07s (R_{es} = 5776)	MSE = 0.0940 t = 5.83s (R_{es} = 5768)	MSE = 0.0944 t = 6.58s (R_{es} = 5768)	MSE = 0.1591 t = 6.25s (R_{es} = 5766)	MSE = 0.1703 t = 2.67s (R_{es} = 5762)	MSE = 0.1503 t = 3.13s (R_{es} = 5769)
	N_T = 0.05N	MSE = 0.0999 t = 8.37s (R_{es} = 4815)	MSE = 0.0999 t = 5.70s (R_{es} = 4818)	MSE = 0.0992 t = 6.28s (R_{es} = 4811)	MSE = 0.1999 t = 8.13s (R_{es} = 4813)	MSE = 0.2131 t = 2.66s (R_{es} = 4813)	MSE = 0.1999 t = 3.04s (R_{es} = 4819)
	N_T = 0.02N	MSE = 0.1181 t = 7.71s (R_{es} = 3948)	MSE = 0.1185 t = 5.57s (R_{es} = 3947)	MSE = 0.1197 t = 6.01s (R_{es} = 3951)	MSE = 0.3726 t = 7.81s (R_{es} = 3878)	MSE = 0.3402 t = 2.59s (R_{es} = 3878)	MSE = 0.3428 t = 2.94s (R_{es} = 3874)

Tab.2. Simulation results for evolutive resampling with different particle number N and threshold Q_T

	Q_T = 1e-4	Q_T = 1e-5	Q_T = 1e-10	Q_T = 1e-20	Q_T = 1e-50	Q_T = 1e-100	Q_T = 0
N=500	MSE = 0.0958 t = 6.27s (R_{es} = 6913)	MSE = 0.0967 t = 6.24s (R_{es} = 6430)	MSE = 0.0989 t = 6.12s (R_{es} = 5224)	MSE = 0.1024 t = 6.03s (R_{es} = 4147)	MSE = 0.1113 t = 5.99s (R_{es} = 2864)	MSE = 0.1225 t = 5.87s (R_{es} = 2105)	MSE = 0.1508 t = 6.03s (R_{es} = 1227)
N=200	MSE = 0.1117 t = 2.93s (R_{es} = 6692)	MSE = 0.1076 t = 2.93s (R_{es} = 6274)	MSE = 0.1106 t = 2.87s (R_{es} = 5149)	MSE = 0.1188 t = 2.82s (R_{es} = 4101)	MSE = 0.1402 t = 2.83s (R_{es} = 2836)	MSE = 0.1635 t = 2.80s (R_{es} = 2087)	MSE = 0.2385 t = 2.84s (R_{es} = 1220)

calculate $\hat{N}_{ESS}$ additional time is required. Further, comparing results for $N = 500$ and $N_T = 0.02N$ with results for $N = 200$ and resampling in each iteration, one can conclude that it is better to reduce particles number N, but do resampling in each iteration, than lower the threshold N_T.

It is also noted that number of resamplings R_{es} is independent of particles number N – in the future it will be checked if it is satisfied also for smaller value N.

For $N = 200$ and low threshold N_T values one can see that computation time for multinomial resampling increase – during calculations the case when all weights are equal

to zero occurs. In this case drawing from importance density is repeated, resulting in the best quality estimation.

Based on these observations one can clearly say that systematic resampling is better then other, therefore evolutive resampling is compared only to systematic resampling.

Based on results in Tab.2 and results in Tab.1 for systematic resampling, one can see that generally evolutive resampling is worse than systematic ($N = 500$). However for $N = 200$ evolutive obtain similar results as systematic. One can suppose that for a smaller number of particles evolutive will be better than systematic.

Future research will be focused on testing others PF algorithms and on improving proposed in this chapter approach – evolutive resampling.

References

Arulampalam S., Maskell S., Gordon N., Clapp T. 2002. A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking, IEEE Proceedings on Signal Processing, Vol.50, No.2, pp.174-188.

Bashi A., Jilkov V., Li R., Chen H. 2003. Distributed Implementation of Particle Filters, Proceedings of the Sixth International Conference of Information Fusion.

Brzozowska-Rup K., Dawidowicz A.L. 2009. Metoda filtru cząsteczkowego, Matematyka Stosowana: matematyka dla społeczeństwa, T. 10/51, pp.69-107.

Candy J.V. 2009. Bayesian signal processing, WILEY, New Jersey, pp.36-44, 237-298.

Chang C., Ansari R., Khokhar A. 2005. Multiple Object Tracking with Kernel Particle Filter, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, Vol. 1, pp.566-573.

Cormen T., Leiserson C., Rivest R., Stein C. 2004. Wprowadzenie do algorytmów, Wydawnictwa Naukowo-Techniczne, Warszawa, pp.40-59.

Douc R., Cappe O., Moulines E. 2005. Comparison of Resampling Schemes for Particle Filtering, Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, September, pp.64-69.

Doucet A., Freitas N., Gordon N. 2001. Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, pp.225-246.

Doucet A., Freitas N., Murphy K., Russell S. 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks, Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence, pp.176-183.

Doucet A., Godsill S., Andrieu C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing, 10, pp.197-208.

Doucet A., Johansen A.M. 2009. A Tutorial on Particle Filtering and Smoothing: Fifteen years later, handbook of Nonlinear Filtering 12, pp.656-704.

Gordon N.J., Salmond N.J., Smith A.F.M. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proceedings-F, Vol.140, No.2, pp.107-113.

Handeby G., Karlsson R., Gustafsson F. 2010. The Rao-Blackwellized Particle Filter: A Filter Bank Implementation, EURASIP Journal on Advances in Signal Processing, Article ID 724087, pp.10.

Kotecha J.H., Djurić P.M. 2003. Gaussian Particle Filtering, IEEE Trans Signal Processing, Vol.51, No.10, pp.2592-2601.

Kozierski P., Lis M. 2012. Filtr cząsteczkowy w problemie śledzenia – wprowadzenie, Studia z Automatyki i Informatyki, Tom 37, pp.79-94.

Launay T., Philippe A., Lamarche S. 2012. On particle filters applied to electricity load forecasting, arXiv preprint, arXiv:1210.0770.

Liu J.S., Chen R. 1998. Sequential Monte Carlo Methods for Dynamic Systems, Journal of the American Statistical Association, September, Vol. 93, No. 443, pp.1032-1044.
Merwe R., Doucet A., Freitas N., Wan E. 2000. The Unscented Particle Filter, Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000.
Mountney J., Obeid I., Silage D. 2011. Modular Particle Filtering FPGA Hardware Architecture for Brain Machine Interfaces, Conf Proc IEEE Eng Med Biol Soc., pp.4617-4620.
Murray L., Lee A., Jacob P. 2013 Rethinking resampling in the particle filter on graphics processing units, arXiv preprint, arXiv:1301.4019.
Pitt M., Shephard N. 1999. Filtering via simulation: auxiliary particle filters, Journal of the American Statistical association, Vol.94, No.446, pp.590-599.
Poyiadjis G., Doucet A., Singh S. 2005. Maximum likelihood parameter estimation in general state-space models using particle methods, Proceedings of the American Statistical Association.
Schön T.B., Wills A., Ninness B. 2011. System identification of nonlinear state-space models, Automatica 47, pp.39-49.
Simon D. 2006. Optimal State Estimation, WILEY--INTERSCIENCE, New Jersey, pp.461-484.
Sutharsan S., Kirubarajan T., Lang T., McDonald M. 2012. An Optimization-Based Parallel Particle Filter for Multitarget Tracking, IEEE Transactions on Aerospace and Electronic Systems, Vol.48, No.2, pp.1601-1618.
Thrun S., 2002. Particle Filters in Robotics, Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI).
Thrun S., Burgard W., Fox D. 2005. Probabilistic robotics, MIT Press, Cambridge, MA, pp.67-90.
Woo J., Kim Y-J., Lee J., Lim M-T. 2006. Localization of Mobile Robot using Particle Filter, SICE-ICASE International Joint Conference, pp.3031-3034.
Zieliński T. 2007. Cyfrowe przetwarzanie sygnałów Od teorii do zastosowań, Wydawnictwa Komunikacji i Łączności, Warszawa, pp.1-38.

Affiliation: Poznan University of Technology, Faculty of Electrical Engineering,
Institute of Control and Information Engineering[1,3)].
Institute of Electrical Engineering and Electronics[2,4)].

Scientific supervisor:
Prof. Assoc. Eng. Andrzej Królikowski[1)],
Prof. Assoc. Eng. Ryszard Porada[2,4)].

Mailing address:
piotr.kozierski@gmail.com[1)],
mail.dla.studenta@gmail.com[2)],
Andrzej.Krolikowski@cie.put.poznan.pl[3)],
ag.zeis@gmail.com[4)].