The Influence of the Prediction Order onto the Accuracy of a BPNN Tracking Filter

Mimi D. Daneva[1]

Abstract – **In this paper an investigation of the accuracy of a BPNN tracking filter with respect to the model (prediction) order is presented. The performances of some BPNN filters of different order are evaluated and compared with standard recursive Kalman filter. Simulated and recorded real radar data are used.**

Keywords – **Radar data processing, neural networks.**

I. Introduction

The basic operations for radar data processing (RDP) are measurement formation, correlation and association of the measurements to the existent target trajectories, and track filtering and prediction. During the measurement formation stage, it is accepted to work in inertial coordinate system, so the measured kinematic parameters of the aircraft are naturally obtained in polar coordinates, and next, if is necessary, they are transformed in other coordinate system, more convenient for future work [1]. Different methods for track filtering and prediction are used for estimation of the current and future kinematic parameters (position, velocity, and acceleration) of the aircraft. For tracking on multiple targets, the achievement of high accuracy of the tracking filter is very important quality of the algorithm for filtering and prediction. This accuracy influences onto the quality of performance of the algorithm for measurement-to-track classification (data association) and plays a key role for correct classification decisions making. Two main types of tracking algorithms are used in practice: probabilistic and heuristic [2]. The first, such as recursive Kalman filter (KF), fixed coefficients filters, etc., are based on the probabilistical theory. The algorithms from the second group use simple heuristic rules or score functions. The neural approach for RDP belongs to the second group of algorithms. Its main advantages are the naturally embedded principle of parallel processing, an independence of the input data's statistics and the mathematical model of the observed dynamic system.

The kinematic model of non-maneuvering aircraft is second-ordered (with nearly constant velocity), and is defined by the equations [1,2]

$$\mathbf{X}(k+1) = \mathbf{\Phi X}(k) + \mathbf{\Gamma w}(k) \qquad (1)$$

$$\mathbf{Z}(k) = \mathbf{HX}(k)+ , \qquad (2)$$

where $\mathbf{X}(k) = \left[\mathbf{X}_1^T(k)\ \mathbf{X}_2^T(k)\ \mathbf{X}_3^T(k)\right]$ is the state vector of the dynamic system (the aircraft) with components the state vectors $\mathbf{X}_i = [\eta_i, \dot{\eta}_i]$ for coordinate i, $i = 1, 2, 3$. Each $\mathbf{X}_i$ contains position η_i and velocity $\dot{\eta}_i$. The symbol i marks the one of the target coordinates, and is used for notational simplicity. The discrete time interval is noted by k. The matrices $\mathbf{w}$ and $\mathbf{v}$ (both of dimension three) are mutually uncorrelated random-valued processes, each with zero mean, known variance and uncorrelated with $\mathbf{X}(0)$. The first, $\mathbf{w}(k)$, gives the random velocity's changes, and the second, $\mathbf{v}(k)$, models the radar measurement errors. The noise covariance matrices and the system matrices are, as follow [1]

$$\mathbf{Q} = [q_{ij}] = E\left\{\mathbf{w}(k)\mathbf{w}^T(k)\right\};$$
$$\mathbf{R} = [r_{ij}] = E\left\{\mathbf{v}(k)\mathbf{v}^T(k)\right\};$$
$$\mathbf{\Phi} = diag[\mathbf{\Phi}_2\ \mathbf{\Phi}_2\ \mathbf{\Phi}_2]; \mathbf{\Gamma} = diag[\mathbf{\Gamma}_2\ \mathbf{\Gamma}_2\ \mathbf{\Gamma}_2];$$
$$\mathbf{H} = diag[\mathbf{H}_2\ \mathbf{H}_2\ \mathbf{H}_2],$$

where the superscript T denotes the transpose operator and

$$\mathbf{\Phi}_2 = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix};\ \mathbf{\Gamma}_2 = \begin{bmatrix} T^2/2 \\ T \end{bmatrix};\ \mathbf{H}_2 = [1, 0],$$

where T is the radar sampling interval.

In this paper a comparative analysis of the accuracy of several tracking filters for non-maneuvering aircrafts using back-propagation neural network (BPNN) with different architecture and different model (prediction) order for one-step-ahead prediction is presented. The performances of these filters are evaluated and compared with standard recursive KF for 50 Monte Carlo (MC) runs with equal input data in polar coordinates for all the filters at each run. An illustrative example using real radar data is shown. The experimental results show that the tracking error and the quality of BPNN training depend on the order of the filter and the best results are obtained when more available information about the tracking history is used.

II. BPNN for Time Series Prediction

A BPNN with static structure can be used as nonlinear predictor of a stationary time series [3,4]. In this case the elements of the input vector $\mathbf{X}_{inp}$ are the past input data samples, i.e.

$$\mathbf{X}_{inp} = [x_{inp}(k-1), x_{inp}(k-2), ..., x_{inp}(k-p)]^T, \qquad (3)$$

where p is the prediction order, and the BPNN's output vector $\mathbf{Y}_p(k)$ produces the estimate $\hat{\mathbf{X}}_{inp}$ of the input vector for one-step-ahead prediction as

$$\mathbf{Y}_p(k) = \hat{\mathbf{X}}_{inp} \qquad (4)$$

[1]Mimi D. Daneva is with the Faculty of Communications and Communicational Technologies, 8 Kl. Ohridski str., 1000 Sofia, Bulgaria, e-mail: mimidan@tu-sofia.acad.bg

The actual values of the elements of the input vector are used as elements of the desired target vector $\mathbf{T}_{inp}$. The input vector is applied to the input neurons of the BPNN. The hidden neurons produce the weighted sum of their output signals, transformed by the sigmoid functions. By summing the weighted output signals of all the hidden units, the output neurons form the estimate of the time series. The prediction error [3,4]

$$e(k) = x_{inp}(k) - \hat{x}_{inp}(k) \tag{5}$$

for iteration k is propagates in backward manner in the neural structure using the error back-propagation algorithm.

III. Design and Training of BPNN Tracking Filters Using Different Prediction Order

Two BPNN architectures with one and two hidden layers and different number of neurons in them are designed, trained, and compared for this investigation. The input and output layers contain equal number of neurons, $n_{inp} = n_{out} = 3$. The numbers of neurons in the first and the second hidden layer are denoted by n_{hid_1} and n_{hid_2}, respectively. They are chosen in a heuristic way to achieve the optimal network architecture for the concrete case [3,5]. The input and the hidden units have bipolar sigmoid functions with biases. The output units are linear.

The input data for the BPNN tracking filter are the polar coordinates range ρ, azimuth θ, and altitude h of the multiple targets, which form the measurement vector $\mathbf{Z}(k)$ at each radar scan k. They are preprocessed by a normalization procedure, so that they have zero mean and unity variance to ensure better convergence of the training algorithm. Thus, the input vector is formulated as [3]

$$\mathbf{X}_{inp} = [\mathbf{Z}^{\bullet}(k-1),\ \mathbf{Z}^{\bullet}(k-2), \ldots \mathbf{Z}^{\bullet}(k-p)] \tag{6}$$

where $\mathbf{Z}^{\bullet}(k-1)$ is the vector of the normalized measurements for all targets. They formed the training set with length Q (Q targets). After the data processing with BPNN, the original variables are recovered by inverse normalization procedure The maximum prediction order $p_{\max} = 3$ is chosen in view of track deletion criteria [1], the absence of measurements for a given track for 3 consecutive radar scans.

The Nguyen-Widrow hidden weights initialization procedure [5] was used for the BPNN filter. It prevents them from premature saturation during the first few iterations of the training algorithm. The BPNN training is performed by the back-propagation algorithm (BPA). The standard BPA is based on the method of steepest descent, but a number of modifications and variations of the standard BPA, such as Newton's and quasi-Newton methods, conjugate gradient methods, etc. may also be used [2-4]. The equations of the standard batch mode BPA for BPNN with two hidden layers are, as follow [3,4].

Forward computations: Computing the net internal activity levels, the output signals of the neurons in the layers and the error signals by the equations

$$\upsilon_j^{(l)}(k) = \sum_{i=1}^{i_{\max}} w_{ji}^{(l)}(k)\, y_i^{(l-1)}(k) \tag{7}$$

$$y_j^{(l)}(k) = \Psi\left(\upsilon_j^{(l)}(k)\right),\ l = 1, 2, 3 \tag{8}$$

$$e_j(k) = d_j(k) - o_j(k) \tag{9}$$

Backward computations: Computing the synaptic weight corrections and the local error gradients of the neurons in the layers as

$$\begin{aligned}\triangle w_{ji}^{(3)}(k) &= \mu\delta_j^{(3)}(k)\,\tilde{x}_i^{(3)}(k) = \\ &= \mu\delta_j^{(3)}(k)\, y_j^{(2)}(k) = \mu\delta_j^{(3)}(k)\, o_j(k)\end{aligned} \tag{10}$$

$$\delta_i^{(3)}(k) = e_i(k)\,\dot{\Psi}_i^{(3)}\left(\upsilon_i^{(3)}(k)\right) \tag{11}$$

for the output layer

$$\Delta w_{ji}^{(2)}(k) = \mu\delta_j^{(2)}(k)\,\tilde{x}_i^{(2)}(k) = \mu\delta_j^{(2)}(k)\, y_j^{(1)}(k) \tag{12}$$

$$\delta_i^{(2)}(k) = \dot{\Psi}_i^{(2)}\left(\upsilon_i^{(2)}(k)\right)\sum_{n_{out}} \delta_i^{(3)}(k)\, w_{ji}^{(3)}(k) \tag{13}$$

for the second hidden layer, and

$$\begin{aligned}\Delta w_{ji}^{(1)}(k) &= \mu\delta_j^{(1)}(k)\,\tilde{x}_i^{(1)}(k) = \\ &= \mu\delta_j^{(1)}(k)\, y_j^{(0)}(k) = \mu\delta_j^{(1)}(k)\,\tilde{x}_i^{(0)}\end{aligned} \tag{14}$$

$$\delta_i^{(1)}(k) = \dot{\Psi}_i^{(1)}\left(\upsilon_i^{(1)}(k)\right)\sum_{n_{hid_2}} \delta_i^{(2)}(k)\, w_{ij}^{(2)}(k) \tag{15}$$

for the first hidden layer, where μ is the learning rate, and $\tilde{x}_i^{(l)}$ is the input signal for neuron i in layer l. The synaptic weights are computed by iterations as

$$\begin{aligned}w_{ji}^{(l)}(k+1) &= w_{ji}^{(l)}(k) + \mu\delta_j^{(l)}(k)\, y_i^{(l-1)}(k), \\ &\qquad l = 1, 2, 3\end{aligned} \tag{16}$$

In the case of BPNN with single hidden layer $(l = 1, 2)$ the corresponding values of the adjustable network parameters are computed by analogy [3,4]. The net performance function

$$E = \frac{1}{Q}\sum_{q=1}^{Q} \xi_q = \frac{1}{2}\sum_{q=1}^{Q}\sum_{j=1}^{n_{out}} (d_{jq} - y_{jq})^2 \tag{17}$$

for batch mode training is minimized as the weight changes are accumulated over all training examples before the weights are actually changed [3-5].

By local approximation of the error surface in the neighborhood of the operating point $w_{ji}^{(l)}$ using Taylor series, Eq. (16) can be written in matrix form as [4]

$$\begin{aligned}\mathbf{W}(k+1) &= \mathbf{W}(k) + \mathbf{\Delta W}(k+1) = \\ &= \mathbf{W}(k) - \overline{\mathbf{H}}^{-1}(k)\,\mathbf{g}(k)\end{aligned} \tag{18}$$

where $\mathbf{g} = \nabla_w \xi_q$ is the vector of local error gradient, and $\overline{\mathbf{H}} = \nabla_w^2 \xi_q$ is the Hessian matrix.

The BPNN is trained by the algorithm of Marquardt-Levenberg, which has the fastest and guaranteed convergence compared with the other BP algorithms [4]. It approximates the Hessian matrix by

$$\overline{\mathbf{H}} \approx \left[\nabla_w^2 \xi_q(\mathbf{W}(k)) + \bar{\nu}\mathbf{I}\right] \tag{19}$$

where $\bar{\nu}$ is the parameter of Levenberg, and $\mathbf{I}$ is identity matrix. The training stop when the global minimum of the net performance function is found or the maximum number of epochs or the maximum learning rate is reached. The track filtering and prediction continues till the moment when the track deletion criteria, the absence of measurements about this track for three consecutive radar scans, is satisfied.

IV. Experimental Results

Simulated and real life data for 6 tracks of non-manoeuvering targets, chosen in random way, are used for the computer modelling. The real life data are recorded from Monopulse Secondary Surveillance Radar CMSSR-401 [6] with sampling time T=10 s. The real (live) tracks with length of 140 consecutive radar scans for the experiments are shown in Fig. 1. They are used as prototypes to model the tracks for the MC simulations, as follow. After polar to Cartesian transform to equalize the dimensions of the coordinates, the measurements from the first and the last radar scan, are connected with straight line and spaced with scan time T to form the trajectory of the non-manoeuvering target. The track T_1 is modelled using the first and the last point of the section with constant altitude and the first and the last point of the section with varying altitude. Next each track is corrupted with Gaussian noises, added directly to each coordinate to model the measurement errors according to the target kinematic model with Eqs. (1) and (2). The cumulative distribution function of the noises is verificated by chi-square test with significant level $\alpha = 0.05$. Next the tracks are transformed back in polar coordinates and then filtered by the algorithm under the test. The standard deviations of the radar measurement errors are 0.05 nautical miles (NM); $0.07°$ and 100 feet for ρ, θ, and h, respectively [6]. The acceleration's standard deviation for KF

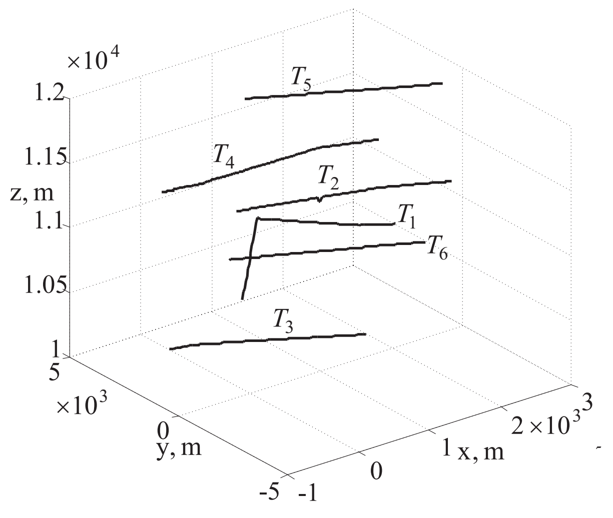

Fig. 1. Live track used for the experiments

Table 1. MC simulations results: BPNN training performance and computational costs

p	1		2		3	
$n_{hid\,1}$	12	12	12	12	12	12
$n_{hid\,2}$	-	6	-	6	-	6
$\bar{k}^T$	1	1	1	1	1	1
$\bar{E}^T_{final} \times 10^{-06}$	0,545	0,796	0,093	0,636	0,080	0,578
$\bar{t}_{CPU}$, s	0,49	0,66	0,55	0,92	0,67	1,14
$\bar{n}_{Mflops}$	1,101	4,640	2,486	8,422	4,755	1,196

Table 2. MC simulations results: BPNN and recursive Kalman filter's averaged tracking errors

Case	BPNN						KF
p	1		2		3		
$n_{hid\,1}$	12	12	12	12	12	12	p=1
$n_{hid\,2}$	-	6	-	6	-	6	
$\bar{\ae}^{T_1}_{\rho}$, NM	0,25	0,24	0,19	0,20	0,24	0,23	1,10
$\bar{\ae}^{T_2}_{\rho}$, NM	0,21	0,18	0,19	0,20	0,24	0,20	1,14
$\bar{\ae}^{T_3}_{\rho}$, NM	0,32	0,22	0,29	0,22	0,32	0,24	1,50
$\bar{\ae}^{T_4}_{\rho}$, NM	0,37	0,19	0,30	0,23	0,32	0,24	1,52
$\bar{\ae}^{T_5}_{\rho}$, NM	0,27	0,18	0,23	0,15	0,25	0,18	1,23
$\bar{\ae}^{T_6}_{\rho}$, NM	0,27	0,18	0,23	0,15	0,26	0,17	1,24
$\bar{\ae}^{T_1}_{\theta}$, rad	0,012	0,011	0,011	0,009	0,013	0,011	0,017
$\bar{\ae}^{T_2}_{\theta}$, rad	0,013	0,008	0,010	0,007	0,011	0,008	0,012
$\bar{\ae}^{T_3}_{\theta}$, rad	0,014	0,010	0,013	0,011	0,014	0,011	0,008
$\bar{\ae}^{T_4}_{\theta}$, rad	0,012	0,008	0,009	0,008	0,011	0,007	0,008
$\bar{\ae}^{T_5}_{\theta}$, rad	0,013	0,010	0,012	0,008	0,012	0,010	0,009
$\bar{\ae}^{T_6}_{\theta}$, rad	0,012	0,011	0,013	0,010	0,012	0,010	0,017
$\bar{\ae}^{T_1}_{h}$, ft	20,35	12,94	18,28	13,20	16,78	12,80	39,52
$\bar{\ae}^{T_2}_{h}$, ft	22,02	15,30	20,01	14,69	17,16	14,04	39,52

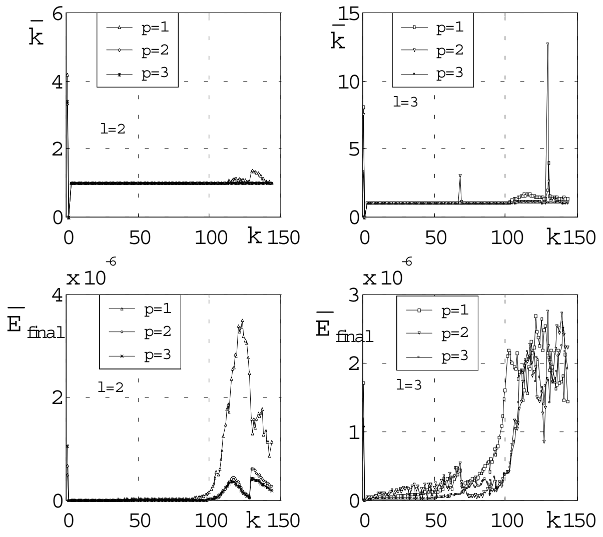

Fig. 2. MC results: BPNN training performance comparison

is 2 gm/s^2 [1]. The performances of BPNN tracking filters with prediction order p are compared with those of standard recursive KF. The tracking error $\zeta$of each filter is defined by the difference between the measurement vector and the estimated state vector [1]. The root mean square (*rms*) values

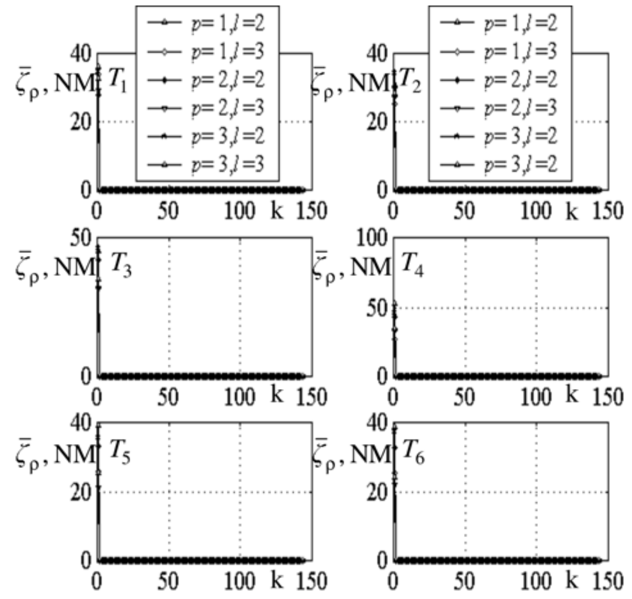

Fig. 3. MC results: BPNN tracking errors for range

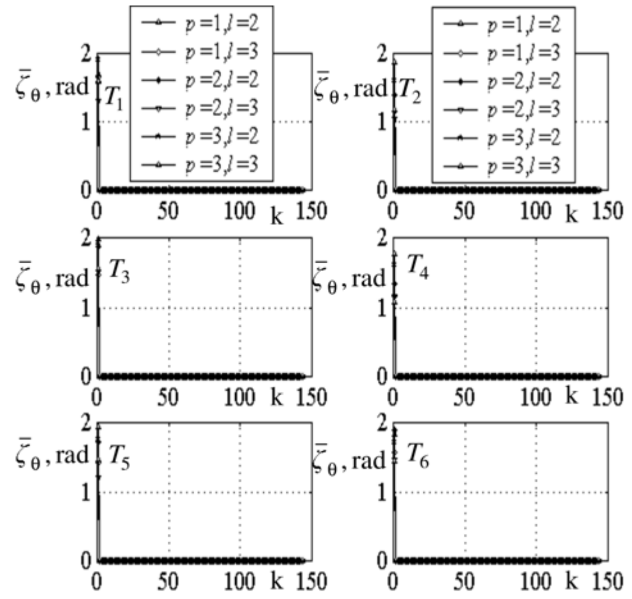

Fig. 4. MC results: BPNN tracking errors for azimuth

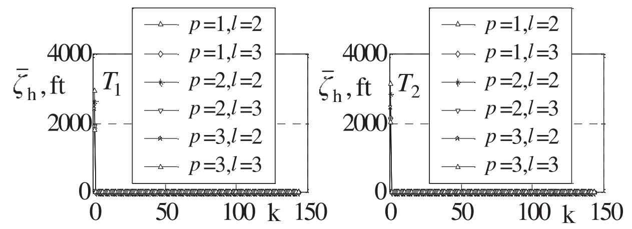

Fig. 5. MC results: BPNN tracking errors for altitude

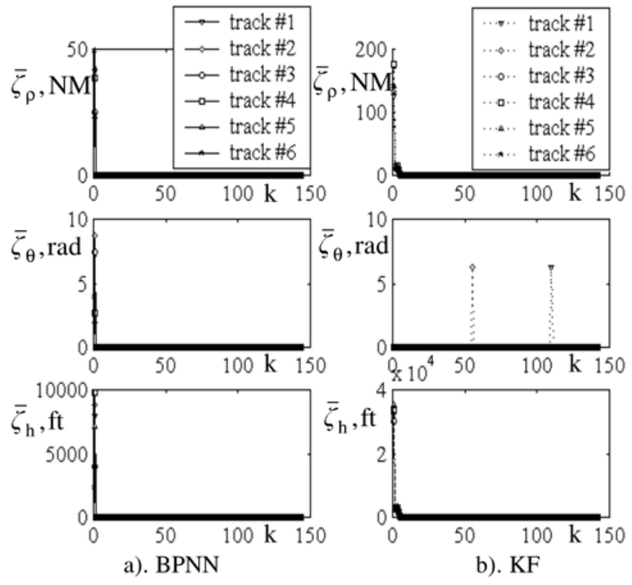

Fig. 6. BPNN and KF's tracking errors for 6 real tracks

of ζ are averaged over all MC runs to form the *rms* error at each time point. All results are obtained by Intel Celeron 500 PPGA with SDRAM 128 MB.

The results from 50 MC runs of the BPNN tracking filter with prediction order p, compared with those of KF for tracks $T_1 - T_6$are presented in Table 1 and Table 2. The averaged BPNN's parameters for the MC runs are averaged number of epochs $\bar{k}$, the averaged net performance function at the end of training $\bar{E}_{final}$, and the *rms* value of $\bar{\zeta}$ for each coordinate, averaged for the MC runs and for the whole track. The results for h coordinate only for the track with variable altitude and for one of the rest tracks are presented here, because the results for the tracks with $h = const$ are very similar. The computational complexity of each filter is estimated by the averaged parameters CPU time and number of Mflops, used for data processing of all training examples (all measurements for all the targets for scan k). The BPNN training performances are plotted in Fig. 2; the tracking errors during MC runs for ρ, θ, and h for all tracks are plotted in Figs. 3 to 5. It is seen that the relationships among the BPNN training parameters and the tracking error with respect to the prediction order are very similar. When more information about the tracking history of the targets is used, the net performance function at the end of training is more close to the global minimum. The network training time and flops increase dramatically as the number of hidden nodes in the layers increases as well as p increases. The best performance of the BPNN filter is with p=3 and BPNN with 12 hidden units in a single hidden layer, the training is stabile and requires almost constant number of epochs. An illustrative example of BPNN tracking performance for this case, compared with the KF for the real tracks is shown in Fig. 6. The BPNN tracking error has lower values than that of KF in all the cases. It is clear that the highest tracking error appears when the BPNN is initialized, and next this error decreases rapidly and keeps almost constant in time. The BPNN initial errors are different for each track, due to the different initial weights. The error due to recovering of the original variables after the inverse normalization does not affect to the accuracy of the BPNN algorithms.

V. Conclusion

This paper has presented an investigation of several BPNN filters with respect to the used prediction order and tracking accuracy. They perform state estimation using non-linear optimization, and do not require prior statistical information on the noise and the target kinematic model. The optimal BPNN architecture and prediction order is chosen by experimental way. It ensures the highest accuracy than the other considered cases, and consistently produces smaller tracking errors than the standard recursive Kalman filter for non-manoeuvering aircrafts. It will improve the performance of the measurement-to-track association logic algorithm in future research.

References

[1] S. Blackman, *Multiple Target Tracking with Radar Applications*, Norwood, Artech House, 1986.

[2] Y. Bar-Shalom (editor), *Multiple-Target Tracking*, vol. I, 1990, and vol. II, 1992, Dedham, Artech House.

[3] S. Haykin, *Neural Networks*, New York, Macmillan College Publishing Company, 1994.

[4] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, Stuttgart, John Wiley & Sons & B. G. Teubner, 1993.

[5] C. G. Looney, *Pattern Recognition Using Neural Networks*, New York, Oxford University Press, 1997.

[6] *Monopulse Secondary Surveillance Radar System Description*, Technical Report, Cardion Inc., *Report no. 131-162A.*