# Kalman Gain Calculations with a Neural Network

Ensar Gul
Marmara University
Dept. of Computer Engineering,
Goztepe Kampusu, Kadıkoy, Istanbul, Turkey
ensar.gul@marmara.edu.tr

## Abstract

**In Kalman filtering, a novel approach is presented to replace a part of the Kalman gain calculations with a neural network. The proposed algorithm avoids matrix inversion in the Kalman filter, hence eliminating the problems encountered**.

## 1 Introduction

Since the publication of R. Kalman`s well known paper [1] on the solution of discrete data linear filtering problem, thousands of scientific papers related to Kalman filtering and its applications were published. Very detailed and excellent materials on Kalman filter can be found in [2], [3], [4], [5],[6] and [7]. A discrete time process is modeled as

$$x_{k+1} = \Phi x_k + \Gamma w_n , \quad (1)$$

with a measurement vector *z*

$$z_{n+1} = Hx_{n+1} + v_{n+1} . \quad (2)$$

Where, the random variables $w_k$ and $v_k$ assumed to be independent each other represent the process and measurement noise respectively with covariance matrices $Q_k$ and $R_k$ respectively. $\Phi$ is an nxn state transition matrix, $\Gamma$ is an nxm excitation matrix. *H* is the mxn measurement matrix. Before the measurements arrival the Kalman filter updates state vector and covariance matrix of prediction as (time update equations):

$$\hat{x}_{k+1/k} = \Phi \hat{x}_k . \quad (3)$$

$$P_{k+1/k} = \Phi P_k \Phi^T + \Gamma Q_k \Gamma^T . \quad (4)$$

Kalman filter calculates new state vector using the Kalman gain and measurements and updates the covariance matrix of estimation as (measurement update equations).

$$B_{k+1} = R_{k+1} + HP_{k+1/k} H^T . \quad (5)$$

$$K_{k+1} = P_{k+1/k} H^T B^{-1} . \quad (6)$$

$$\hat{x}_{k+1} = \hat{x}_{k+1/k} + K_{k+1}(z_{k+1} - H\hat{x}_{k+1/k}) . \quad (7)$$

$$P_{k+1} = (I - K_{k+1} H) P_{k+1/k} . \quad (8)$$

If we look at the Kalman filtering equations above, the matrices are manipulated by multiplying, adding, subtracting and inverting them. We can use standard Gaussian elimination in order to invert the matrix *B* in the Kalman gain calculations, however, there are problems encountered in performing matrix inversion:

- singular or near singular matrices can not be inverted.
- errors accumulates in gauss elimination algorithm reducing accuracy

hence the overall system including matrix inversion becomes less robust .

In a linear system of equations *AX* = *C*, it can be shown that

$$\frac{\|\Delta X\|}{\|X + \Delta X\|} \le \|A\| \|A^{-1}\| \frac{\|\Delta C\|}{\|C\|} . \quad (9)$$

$$\frac{\|\Delta X\|}{\|X\|} \le \|A\| \|A^{-1}\| \frac{\|\Delta A\|}{\|A\|} . \quad (10)$$

The equations above show that the relative change in the norm of right hand side vector or the coefficient matrix can be amplified by as much as $\|A\| \|A^{-1}\|$ which is the condition number of the matrix. This number coupled with the machine epsilon, can quantify the accuracy of the solution of *AX* = *C*. The order of the relative error in the solution will be *Cond(A)* * $\varepsilon_{machine}$ . Hence *Cond(A)* * $\varepsilon_{machine}$ gives

us the number of significant digits at least correct by comparing it with $0.5x10^{-p}$ [8][9][10]. The condition number of matrix *B* and machine epsilon value will be used in the simulations to asses the accuracy of matrix inversion.

In order to prevent the effect of the problems mentioned above, the proposed simple algorithm replaces a part of the Kalman gain calculations with a neural network. This algorithm is presented in Section 2. In section 3 simulation results which compare the Kalman filter with the neural network aided Kalman filter are presented.

## 2 Neural Network Aided Kalman Gain Calculations

The innovation matrix *B* is an mxm matrix. We can construct a neural network taking the elements of *B* as inputs and calculating the inverse of *B* as outputs. Note that this can be achieved through training the network using a learning algorithm. The main advantage of this is that it avoids matrix inversion in Kalman filtering calculations. The neural network aided Kalman gain calculations is shown in Fig. 1.

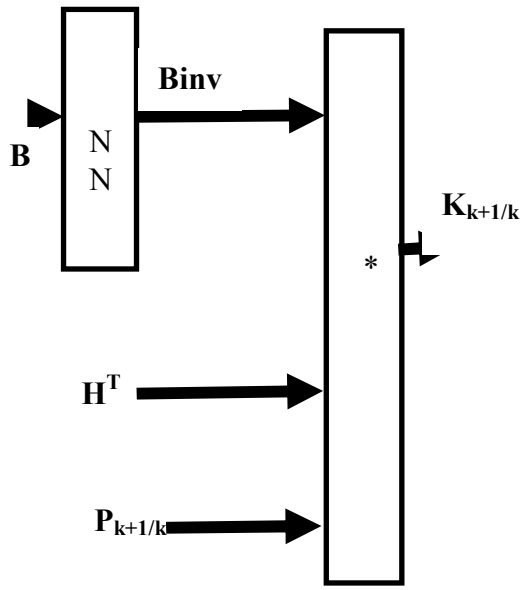


Fig 1 Kalman gain calculations with a neural network

Jang et al[11] proposed a modified Hopfield neural network for matrix inversion. Steriti at al [12] simulated this algorithm and showed that although neural network algorithm takes longer processing time, the results are less sensitive to error, round off and noise.

Fa-Long and Zeng [13] developed a two dimensional neural network for matrix inversion and showed by both simulations and analytically that the network is stable and can perform matrix inversion within a limited time period. Wang [14] also presented a recurrent neural network for matrix inversion which can be decomposed into n independent sub-networks and can be implemented by an analog circuit.

Al-Mudhaf and Esat [15] discuss the use of feed forward neural network (FNN) in matrix manipulations. They concentrate on training algorithms of FNN and speeding up the training process of the neural network. To the best knowledge of author of this article, there has been no study in the literature which uses NN matrix inversion in Kalman filtering calculations.

In this study, a back-propagation network is used to construct the neural network for matrix inversion. It is a 3-layer network including an input layer, a hidden layer and output layer with m-(m+2)-m processing elements respectively. There are two major parameters that have to be set up prior to the learning. These are learning coefficients and momentum coefficient. Sigmoid function is used to define the output values of the processing elements of the network. The detailed information about back propagation networks can be found in [16].

Back propagation networks are capable of mapping certain inputs to certain outputs using its internal dynamics featured by a learning algorithm. This can explain how the inverse of matrix is achieved using the matrix itself as an input. In this study the learning algorithm is based on a gradient descent rule.

Assuming that the state space where the filter will operate is known in advance. Training data were generated using Monte Carlo simulations. The input values generated through simulations scaled down to 0-1 interval in order to improve the generalization capability of the network. The following equations are used for this purpose

$$y_{scaled} = (y - y_{\min}) / (y_{\max} - y_{\min}), \qquad (11)$$

$$x_{scaled} = (x - x_{\min}) / (x_{\max} - x_{\min}), \qquad (12)$$

A well known training multi-layer perception (MLP) algorithm [17][18] [19][20] was used to train the network.

The Kalman gain is calculated using the equation (6) in the standard Kalman filter. In this equation the inversion of the matrix $B$ was replaced by the neural network. The rest of Kalman filter equations stay the same.

## 3. Simulation Results

Consider a simple target tracking problem to examine mean square errors in the estimates of the Kalman Filter and Neural Network aided Kalman Filter. Assuming that the target move in x-y plane with a constant velocity and random acceleration noise, the state vector, the state transition matrix, the excitation matrix and measurement matrix are given below respectively.

$$x_{k+1}^T = (x\ \dot{x}\ y\ \dot{y})_{k+1}, \tag{13}$$

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{14}$$

$$\Gamma = \begin{bmatrix} \Delta t^2/2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix}, \tag{15}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{16}$$

Here $\Delta t$ is the sampling interval which is set to 1 sec. The variance of range measurement noise is 0.00001km$^2$ and the variance angular measurement noise is 0.00004 rad$^2$. The speed of the target is 0.015km/sec. The acceleration noise variance is $qx$=$qy$=0.000005(km/s$^2$)$^2$. The inclination of the target with respect of the x axis is 45 degree. Simulations were performed for 100 scans. Figures 2 and 3 present comparison between estimation errors of the two filters for x and y position estimates. As we see from these figures the rms errors of x estimates in neural network aided Kalman filter are comparable to the standard Kalman filter. However, those of y estimates are grater than the Kalman filter. Note that the error scale is ten times smaller in magnitudes than error scale for x estimates.

Figure 4 shows how the determinant of the matrix B changes during the simulations for 100 scans. When the filter is initialized the determinant of $B$ is very close to zero which can cause matrix inversion algorithm failure. On the other hand, the condition numbers of the matrix which give better idea about the matrix $B$ is shown in Figure 5. The condition numbers are calculated using 2-norm. By examining this figure it can be seen that the matrix $B$ is not well conditioned. After the initial period, the condition numbers are settled around 1300. The value of $\varepsilon_{machine}$ for an embedded 32 bit processor with single precision floating point unit which has a mantissa of 24 bits can be calculated as $\varepsilon_{machine} = 2^{1-24} = 0.1192x10^{-6}$. Then

$$0.5x10^{-p} < cond(B) * \varepsilon_{machine}$$

$0.5\text{x}10^{-p} < 1300 * 0.1192\text{x}10^{-6} = 0.1549 * 10^{-3}$

hence p >=3

Here the results are given for a particular scenario. Simulations have been performed with different noise levels. In all of them the matrix $B$ is found to be not well conditioned. For three different values of range and measurements noise, the average values of the condition numbers of the matrix $B$ for 100 scans are shown in the Table 1. From this table it can be seen that as measurements noise increases the condition numbers of the matrix $B$ also increases. Therefore using neural network in the calculations of the matrix B prevents us dealing with ill conditioned matrices.

Table 1 The condition numbers of the matrix B

| Noise | Noise variance | average of condition numbers of the matrix B |
|---|---|---|
| Measurement noise (km$^2$)<br>Angular noise (rad$^2$) | 0.000001<br>0.000004 | 623 |
| Measurement noise (km$^2$)<br>Angular noise (rad$^2$) | 0.00001<br>0.00004 | 1279 |
| Measurement noise (km$^2$)<br>Angular noise (rad$^2$) | 0.0001<br>0.0004 | 1890 |

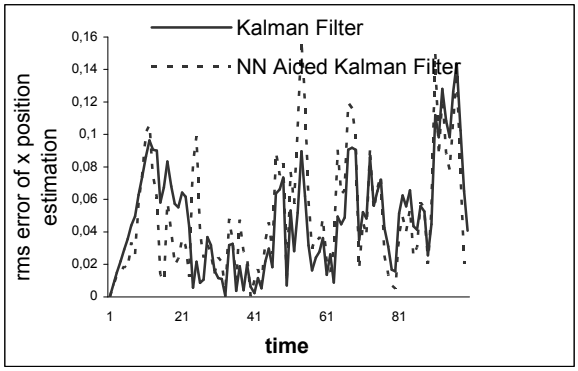


Fig 2 rms error of x position estimation

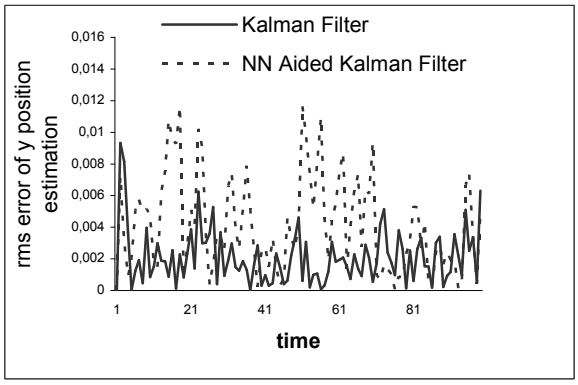


Fig 3 rms error of y position estimation

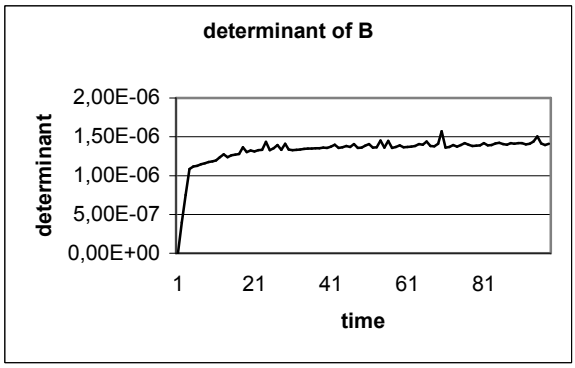


Fig 4 Determinant of the matrix *B*

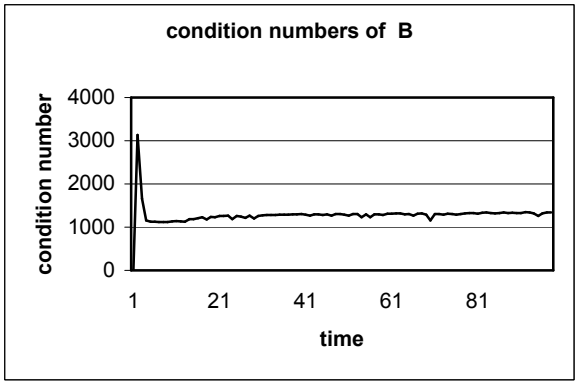


Fig 5 Conditions numbers of the matrix *B*

# 4. Conclusions

In this study a novel approach is presented in such a way that the neural network technology is utilized for performing matrix inversion in Kalman filtering algorithm. The approach overcomes matrix inversion problems. Note that traditional gauss elimination algorithm for matrix inversion in a Kalman filter may yield filter which is not robust enough due the loss of accuracy in the inversion process. The simulation results showed that for the chosen example the matrix *B* is not well conditioned. Simulations were performed using double precision arithmetic, however in the actual embedded system implementation it is assumed that only single precision is available. Neural networks are well known for their robust structure which may definitely produce more stable estimates. Moreover neural networks can estimate inverse values of any matrix regardless of condition numbers of the matrix which is one of the main difficulties of the standard Kalman filter.

Although generalization capability of neural network makes them applicable in a wide range of problem spaces, it should be stated the proposed approach is effectively applicable when the problem space is known, for example using the filter to estimate trajectories of the objects with limited life time.

Further investigations can be made by using different type of neural networks in Kalman gain calculations and comparing the results. Applying the proposed approach to nonlinear filters such as Unscented Kalman Filter [21] and Central Difference Kalman Filter [22] should be a subject of future work.

# Acknowledgement

The author would like to express his gratitude to Ercan Oztemel for his support in the construction of the NN.

# References


[1] Kalman, R. E. (1960) "A New Approach to Linear Filtering and Prediction Problems," Transaction of the ASME—Journal of Basic Engineering, pp. 35-45

[2] Sorenson, H. W. (1970) "Least-Squares estimation: from Gauss to Kalman", IEEE Spectrum, vol. 7, pp. 63-68.

[3] Gelb, A., (1974) *Applied Optimal Estimation*, MIT Press, Cambridge, MA.

[4] Maybeck, Peter S. (1979) *Stochastic Models, Estimation, and Control*, Volume 1, Academic Press, Inc.

[5] Lewis, R, (1986) *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.

[6] Grewal, Mohinder S., and Angus P. Andrews, (1993) *Kalman Filtering Theory and Practice*, Upper Saddle River, NJ USA, Prentice Hall.

[7] Bozic, S. M. (1994) *Digital and Kalman Filtering*. Edward Arnald Publications.

[8] http://numericalmethods.eng.usf.edu/mws/gen/04sle/

[9] Golub, Gene H., Charles F. Van Loan,(1996) *Matrix Computations*, The John Hopkins University Press, (3rd edition).

[10] A priori and a posteriori bounds in matrix computations, Encyclopaedia of Mathematics,(2001), http://eom.springer.de/A/a110010.htm.

[11] Jang, J., S. Lee, and S. Shin, (1988), "An optimization network for matrix inversion," in Neural Information Processing Systems, D.Z. Anderson, (Editor), American Institute of Physics, New York, pp. 397-401.

[12] Steriti, R.; J. Coleman, M. A. Fiddy, (1990) "A neural network based matrix inversion algorithm", IJCNN International Joint Conference on Neural Networks, vol. 1, pp:467 – 470.

[13] Fa-Long, L. and B. Zeng,(1992) "Neural Network Approach to Computing Matrix Inversion", Applied Mathematics and Computation, Vol. 47, pp. 109-120..

[14] Wang, J., (1993) "A Recurrent Neural Network for Real-Time Matrix Inversion", Applied Mathematics and Computation, Vol. 55, pp. 89-100.

[15] Al-Mudhaf, A., and I.I Esat, (2004), "Matrix Manipulations using Artificial Neural Networks", Transactions of SDPS, Vol. 8 No 3, pp. 113-122.

[16] Haykin S. (1998), *Neural Netwoks A Comprehensive Foundation*, Prentice Hall, (2nd edition).

[17] Rumelhart, D.E., GE. E. Hinton, and R. J. Williams (1988) 'Learning internal representations by error propagation' in Neurocomputing: foundations of research, pp 673-695, edited by James A. Anderson and E. Rosenfeld, MIT Press, Cambridge MA USA.

[18] Amin, H.; K. M. Curtis, B. R. Hayes Gill (1997), " Dynamically pruning output weights in an expanding multilayer perceptron neural network" 13th International Conference on Digital Signal Processing Proceedings, Volume 2, Issue , 2-4 pp:991 – 994.

[19] Chen, H. H., M. T. Manrym M. T. and H. Chandrasekaran (1999), "A neural network training algorithm utilizing multiple sets of linear equations," Neurocomputing, vol. 25, no. 1-3, pp. 55-72

[20] Delashmit, W. H. and M. T. Manry, (2005)., "Recent Developments in Multilayer Perceptron Neural Networks", Proceedings of the 7th annual Memphis Area Engineering and Science Conference (MAESC).

[21] Julier, S.J, and J.K. Uhlmann, (2004), "Unscented Filtering and Nonlinear Estimation", Proceedings of IEEE, vol 92, pp. 401-422.

[22] Norgaard, M, N. K. Poulsen, and O. Ravn,(2000) "New Developments in State Estimation for Nonlinear Systems," Automatica, vol. 36, pp.1627-1638.