Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (22): 3957-3965, 2009
ISSN 1812-5654

State Estimation for Mobile Robot Using Neural Networks

S. Chouraqui and M. Benyettou
Simulations and Modelling of Industrial Systems Laboratory,
Department of Computer Science, Faculty of Science,
University of Science and Technology of Oran, Mohammed Boudiaf USTO,
P.O. Box 1505, El Menaour, Oran, Algeria

Abstract: In this study, we describe a state estimation method using neural networks. The developed method has been applied to the problem of estimating the parameter of a six degree of freedom (DOF) PUMA 560 Robot arm manipulator. The results obtained were compared with the conventional Extended Kalman Filter, showing an improvement in performance and robustness. From the simulations performed, it can be said that the neural filter proposed for the studied application is more consisted and convergent than the EKF filter.

Key words: Kalman filter, neural networks, six degree of freedom, robot arm, nonlinear state, estimation

INTRODUCTION

Accurate on-line estimates of critical system states and parameters are needed in a variety of engineering applications, such as condition monitoring, fault diagnosis and process control. In these and many other applications it is required to estimate a system variable which is not easily accessible for measurement, using only measured system inputs and outputs.

Since, the development of the well-known Kalman Filter (KF) approach for state estimation in linear systems (Crassidis and Junkins, 2004; Kalman, 1960) this method has been widely studied in the literature and applied to many problems. The KF is a recursive estimation approach, where state estimates are obtained in two distinct steps. In the first step, a one-step-ahead state prediction is obtained, based on the latest state estimate. In the second step, the state estimate is refined by linearly combining the state prediction obtained in the first step, with the new output measurements. The KF has been extended to nonlinear systems. One such extension, called the extended Kalman filter (EKF) (Crassidis and Junkins, 2004), has experienced a lot of interest from researchers but it has found few industrial applications (Jonsson and Palsson, 1994). In the EKF approach, the state prediction is obtained in the first step using nonlinear state propagation based on the available model. In the second step, the newly received output measurements are combined with the state prediction, still in a linear fashion. As a result the state estimate is effectively a linear combination of all output measurements. In both the KF and the EKF, the model of the system used in the predictor is assumed to be perfectly known, along with the statistics of the process and sensor noise entering the system. These assumptions severely restrict the applicability of these filters in real-world problems and a more general nonlinear state estimation approach is definitely desirable. The EKF algorithm has been around for over 25 years, but it has only in the last a couple of years that rigorous analysis for this algorithm has appeared in the literature (Boutayeb *et al*., 1997; Reif *et al*., 1999).

Neural networks have been extensively investigated in the context of adaptive control and system identification (Billings *et al*., 1992; Haykin, 1999; Ljung and Sjoberg, 1992; Naraendra and Parthasarathy, 1990), but it was more recently that their use has been proposed in state estimation (Rochefort *et al*., 2005).

This study introduces a new view of estimating the position and velocity of a six degree of freedom robot arm manipulator using a nonlinear filter based on neural networks, the results obtained of the presented algorithm were compared with the EKF method.

DYNAMIC MODEL OF PUMA560 WITH SIX DEGREE OF FREEDOM

The PUMA 560 (Programmable Universal Machine for Assembly) depicted in Fig. 1 is a 6 DOF manipulator with revolute joints that has been widely used in industry

Corresponding Author: S. Chouraqui, Simulations and Modelling of Industrial Systems Laboratory, Department of Computer Science, Faculty of Science, University of Science and Technology of Oran, Mohammed Boudiaf USTO, P.O. Box 1505, El Menaour, Oran, Algeria

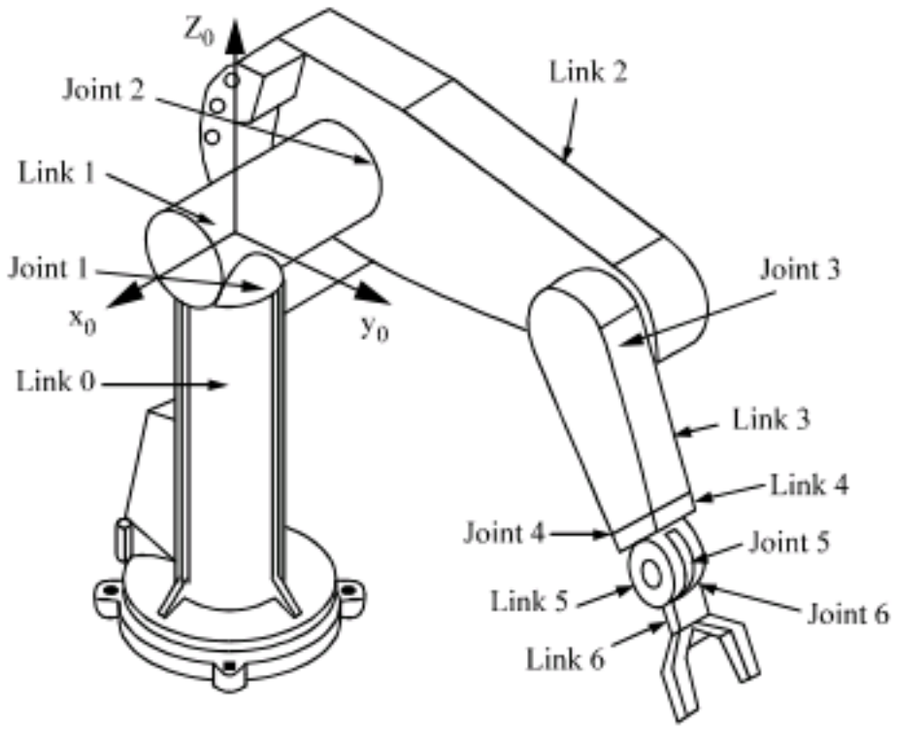

Fig. 1: Puma 560

and still is in academia. The PUMA uses DC servo motors as its actuators, has high resolution relative encoders for accurate positioning and potentiometers for calibration (Armstrong *et al*., 1986).

Manipulator dynamics is concerned with the equations of motion, the way in which the manipulator moves in response to torques applied by the actuators, or external forces. The history and mathematics of the dynamics of serial-link manipulators are well covered in the literature. The equations of motion for an n-axis manipulator are given by:

$$T = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) \quad (1)$$

where, q is the vector of generalized joint coordinates describing the pose of the manipulator, $\dot{q}$ is the vector of joint velocities, $\ddot{q}$ is the vector of joint accelerations, M is the symmetric joint-space inertia matrix, or manipulator inertia tensor, C describes Coriolis and centripetal effects-Centripetal torques are proportional to $\ddot{q}_i^2$, while the Coriolis torques are proportional to $\dot{q}_i\dot{q}_j$, F describes viscous and Coulomb friction and is not generally considered part of the rigid body dynamics.

It is straightforward to express $\ddot{q}$ from Eq. 1 as follows:

$$\ddot{q} = M^{-1}(q)\left(T - C(q,\dot{q})\dot{q} - F(\dot{q}) - G(q)\right) \quad (2)$$

For the six-degree-of-freedom model used here n = 6. The nonzero elements of the inertia matrix (in kgm^2) are given as:

$$M(1,1) = 2.75 + 1.38\cos(q_1^2) + 0.3(\sin(q_1 + q_2))^2 + 0.77\cos(q_1)\sin(q_2 + q_2) \quad (3)$$

$$M(1,2) = 0.69\sin(q_1) - 0.134\cos(q_1 + q_2) + 0.238\cos(q_1) \quad (3.1)$$

$$M(1,3) = -0.134\cos(q_1 + q_2) - 0.00397\sin(q_1 + q_2) \quad (3.2)$$

$$M(1,4) = M(1,5) = M(1,6) = 0 \quad (3.3)$$

$$M(2,1) = M(1,2) \quad (3.4)$$

$$M(2,2) = 6.79 + 0.744\sin(q_2) \quad (3.5)$$

$$M(2,3) = 0.333\sin(q_2) - 0.011\cos(q_2) \quad (3.6)$$

$$M(2,4) = M(2,5) = M(2,6) = 0 \quad (3.7)$$

$$M(3,1) = M(1,3) \quad (3.8)$$

$$M(3,2) = M(2,3) \quad (3.9)$$

$$M(3,3) = 1.16 \quad (3.10)$$

$$M(3,4) = -0.00125\ \sin(q_3)\sin(q4) \quad (3.11)$$

$$M(3,5) = -0.00125\cos(q_3)\cos(q4) \quad (3.12)$$

$$M(3,3) = 0 \quad (3.13)$$

$$M(4,1) = M(4,2) = 0 \quad (3.14)$$

$$M(4,3) = M(3,4) \quad (3.15)$$

$$M(4,4) = 0.2 \quad (3.16)$$

$$M(4,5) = M(4,6) = 0 \quad (3.17)$$

$$M(5,1) = M(5,2) = M(5,4) = M(5,6) = 0 \quad (3.18)$$

$$M(5,3) = M(3,5) \quad (3.19)$$

$$M(5,5) = 0.18 \quad (3.20)$$

The nonzero Christoffel (Coriolis) symbols (in Nms^2) are given as:

$$C(1,1) = 0.5\left(0.6\sin(q_1+q_2)\cos(q_1+q_2) - 2.76\sin(q_1)\cos(q_1) + 0.744\cos(2q_1+q_2)\right)\dot{q}_2 + 0.5\left(0.6\sin(q_1+q_2)\cos(q_1+q_2) + 0.744\cos(q_1)\cos(q_1+q_2)\right)\dot{q}_3 \quad (4)$$

$$C(1,2) = 0.5\left(0.6\sin(q_1+q_2)\cos(q_1+q_2) - 2.76\sin(q_1)\cos(q_1) + 0.744\cos(2q_1+q_2)\right)\dot{q}_1 + \begin{pmatrix} 0.69\cos(q_1) - 0.0238\sin(q_1) \\ +0.134\sin(q_1+q_2) \end{pmatrix}\dot{q}_2 + 0.5\left(0.268\sin(q_1+q_2) - 0.00397\cos(q_1+q_2)\right)\dot{q}_3 \quad (4.1)$$

$$C(1,3) = 0.5\left(0.6\sin(q_1+q_2)\cos(q_1+q_2) + 0.744\cos(q_1)\cos(q_1+q_2)\right)\dot{q}_1 + 0.5\left(0.268\sin(q_1+q_2) - 0.00397\cos(q_1+q_2)\right)\dot{q}_2 + \left(0.134\sin(q_1+q_2)\right)\dot{q}_2 - \left(0.00397\cos(q_1+q_2)\right)\dot{q}_3 \quad (4.2)$$

$$C(1,4) = C(1,5) = C(1,6) = 0 \quad (4.3)$$

$$C(2,1) = 0.5\left(2.76\sin(q_1)\cos(q_1) - 0.6\sin(q_1+q_2)\cos(q_1+q_2) - 0.744\cos(2q_1+q_2)\right)\dot{q}_1 + \begin{pmatrix} 0.69\cos(q_1) - 0.0238\sin(q_1) \\ +0.134\sin(q_1+q_2) \end{pmatrix}\dot{q}_2 + 0.5\left(0.00397\cos(q_1+q_2)\right)\dot{q}_3 \quad (4.4)$$

$$C(2,2) = \left(0.5 * 0.774\cos(q_2)\right)\dot{q}_3 \quad (4.5)$$

$$C(2,3) = \left(0.5 * 0.00397\cos(q_1+q_2)\right)\dot{q}_1 + \left(0.5 * 0.744\cos(q_2)\right)\dot{q}_2 + \left(0.372\cos(q_2) + 0.011\sin(q_2)\right)\dot{q}_3 \quad (4.6)$$

$$C(2,4) = C(2,5) = C(2,6) = 0 \quad (4.7)$$

$$C(3,1) = -0.5\left(0.6\sin(q_1+q_2)\cos(q_1+q_2) + 0.744\cos(q_1)\cos(q_1+q_2)\right)\dot{q}_1 - 0.5\left(0.00397\cos(q_1+q_2)\right)\dot{q}_2 \quad (4.8)$$

$$C(3,2) = \left(-0.5 * 0.00397\cos(q_1+q_2)\right)\dot{q}_1 - \left(0.5 * 0.744\cos(q_2)\right)\dot{q}_2 \quad (4.9)$$

$$C(3,3) = C(3,6) = 0 \quad (4.10)$$

$$C\,(3,4) = \left(-0.00125\cos(q_3)\cos(q_4)\right)\dot{q}_4 a \quad (4.11)$$

$$C(3,5) = \left(0.00125\cos(q_3)\cos(q_4)\right)\dot{q}_5 \quad (4.12)$$

$$C(4,1) = C(4,2) = C(4,3) = C(4,4) = C(4,6) = 0 \quad (4.13)$$

$$C(4,3) = \left(-0.00125\sin(q_3)\cos(q_4)\right)\dot{q}_5 \quad (4.14)$$

$$C(4,5) = \left(-0.00125\sin(q_3)\cos(q_4)\right)\dot{q}_3 \quad (4.15)$$

$$C(5,3) = \left(0.00125\sin(q_3)\cos(q_4)\right)\dot{q}_4 \quad (4.16)$$

$$C(5,4) = -C(4,5) \quad (4.17)$$

$$C(5,1) = C(5,2) = C(5,5) = C(5,6) = 0 \quad (4.18)$$

$$C(6,1) = C(6,2) = C(6,3) = C(6,4) = C(6,5) = C(6,6) = 0 \quad (4.19)$$

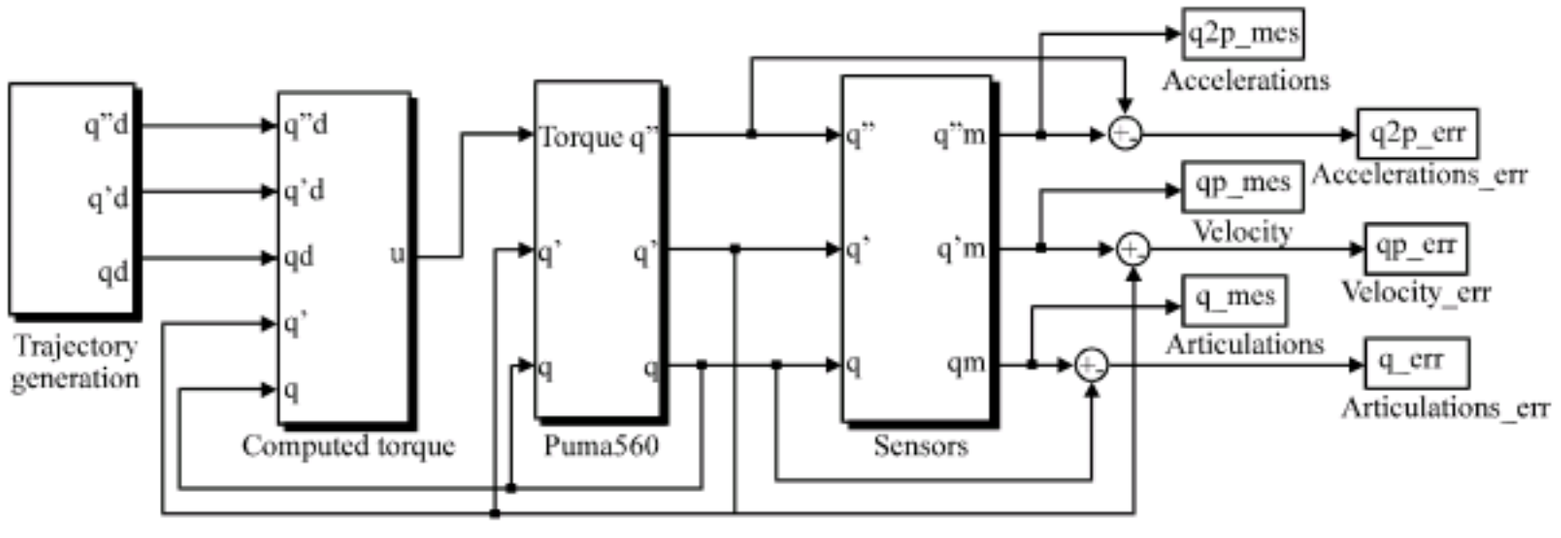

Fig. 2: Simulink simulation diagram of PUMA560

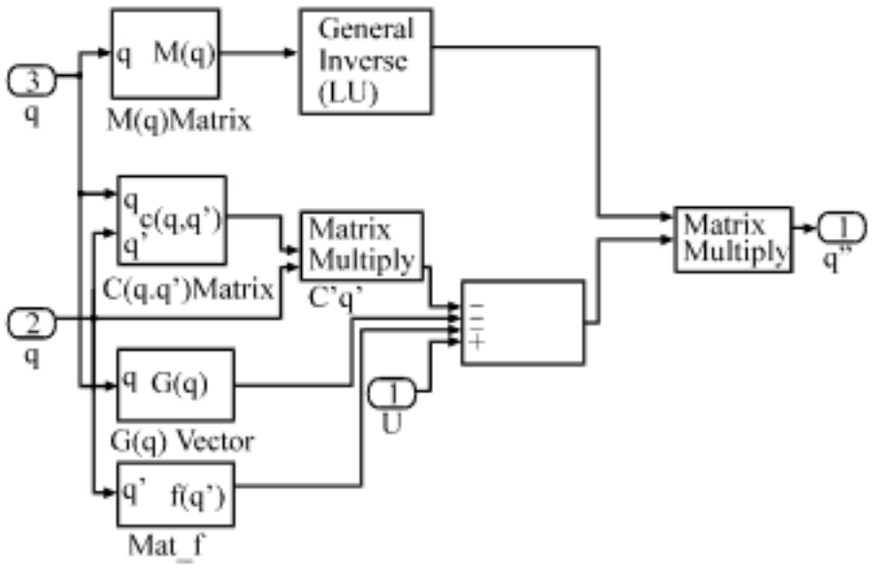

Fig. 3: Simulink diagram of PUMA560 dynamics

and nonzero gravity vector entries (in Nm) are given below:

$$F(1,1) = 0.04\dot{q}_1,\ F(2,1) = 0.04\dot{q}_2 \tag{5}$$

$$F(3,1) = 0.04\dot{q}_3,\ F(4,1) = 0.04\dot{q}_4 \tag{5.1}$$

$$F(5,1) = 0.04\dot{q}_5,\ F(6,1) = 0.04\dot{q}_6 \tag{5.2}$$

$$G(1,1) = G(1,6) = 0 \tag{5.3}$$

$$G(2,1) = -37,2\sin(q_1+q_2) - (8.4\sin(q_1+q_2)) + (1.02\sin(q_1)) \tag{5.4}$$

$$G(3,1) = (-8.4\sin(q_1+q_2)) + (0.25\cos(q_1+q_2)) \tag{5.5}$$

$$G(4,1) = 0.028\sin(q_1+q_2)\sin(q_3)\sin(q_4) \tag{5.6}$$

$$G(5,1) = -0.028(\cos(q_1+q_2)\sin(q_3)\sin(q_4) + \sin(q_1+q_2)\cos(q_3)\cos(q_4)) \tag{5.7}$$

Simulink based simulation model of puma 560 dynamics and complete simulation diagram is shown in Fig. 2 and 3.

KALMAN FILTER

The main steps of the extended Kalman filter (EKF) can be formulated as described by Haykin (1996). It assumes a formulation of the non-linear system with the input states u, the output states y and the non-measurable, internal states x. In the following formulation, w and v are the process and output noise vectors, respectively.

$$\begin{aligned}\dot{x}(t+1) &= f(x(t),u(t),w(t))\\ y(t) &= h(x(t),v(t))\end{aligned} \tag{6}$$

The state vector of interest in this estimation problem is:

$$x = \left[q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ \dot{q}_1\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ \dot{q}_5\ \dot{q}_6\right] \tag{7}$$

The nonlinear state-space model follows from Eq. 1 as:

$$\dot{x} = f(x) = \begin{pmatrix} x_7 \\ x8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ -M^{-1}(q_1)\left[C(q_1\cdot\dot{q}_1)\dot{q}_1 + F(\dot{q}_1) + G(q_1)\right] \\ -M^{-1}(q_2)\left[C(q_2\cdot\dot{q}_2)\dot{q}_2 + F(\dot{q}_2) + G(q_2)\right] \\ -M^{-1}(q_3)\left[C(q_3\cdot\dot{q}_3)\dot{q}_3 + F(\dot{q}_3) + G(q_3)\right] \\ -M^{-1}(q_4)\left[C(q_4\cdot\dot{q}_4)\dot{q}_4 + F(\dot{q}_4) + G(q_4)\right] \\ -M^{-1}(q_5)\left[C(q_5\cdot\dot{q}_5)\dot{q}_5 + F(\dot{q}_5) + G(q_5)\right] \\ -M^{-1}(q_6)\left[C(q_6\cdot\dot{q}_6)\dot{q}_6 + F(\dot{q}_6) + G(q_6)\right] \end{pmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ M^{-1}(q_1) \\ M^{-1}(q_2) \\ M^{-1}(q_3) \\ M^{-1}(q_4) \\ M^{-1}(q_5) \\ M^{-1}(q_6) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} \tag{8}$$

The Extended Kalman filter equations are given as follows:

$$\tilde{x}_{k+1} = x_k + \int_{t_k}^{t_{k+1}} f(x)dt \tag{9}$$

$$\tilde{P}_k = \Phi_{k-1} P_{k-1} \Phi_{k-1}^T + Q_k \tag{9.1}$$

$$K_k = \tilde{P}_k H_k^T \left(H_k \tilde{P}_k H_k^T + R_k\right)^{-1} \tag{9.2}$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k \left(y_k - h\left(\hat{x}_{k-1}\right)\right) \tag{9.3}$$

$$P_k = \left(I - K_k H_k\right)\tilde{P}_k \tag{9.4}$$

Matrix H_k is obtained by linearization of the function h (x) as follows:

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\tilde{x}_k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{for} \quad v=0 \tag{10}$$

In the same way the matrix Φ_k is obtained as:

$$\Phi_k = \left.\frac{\partial f}{\partial x}\right|_{x=\tilde{x}_k} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{11}} & \frac{\partial f_1}{\partial x_{12}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{11}} & \frac{\partial f_2}{\partial x_{12}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_{11}}{\partial x_1} & \frac{\partial f_{11}}{\partial x_2} & \cdots & \frac{\partial f_{11}}{\partial x_{11}} & \frac{\partial f_{11}}{\partial x_{12}} \\ \frac{\partial f_{12}}{\partial x_1} & \frac{\partial f_{12}}{\partial x_2} & \cdots & \frac{\partial f_{12}}{\partial x_{11}} & \frac{\partial f_{12}}{\partial x_{12}} \end{bmatrix} \tag{11}$$

NEURAL NETWORKS

Multilayer Perceptron (MLP): A Neural Network (NN) is a computational structure composed of a collection of artificial neurons. Each neuron is an operator that processes, with a nonlinear activation function ϕ, the weighted sum of its inputs to produce an output signal. The connections between the artificial neurons define the behavior of the net, identify its applicability and specify its training methods (Haykin, 1994). In a Multi-Layer Perceptron (MLP), the neurons are grouped in one or more layers with the output of each layer being the input to the next layer.

The training process consists of adjusting the neuron weights based on the expected output and some optimization rules. In a supervised scheme, the weights are adjusted interactively by comparing the output of the network with the desired value at each step. This scheme means that the training process teaches the net the expected output for each given input.

A feed forward MLP is a mapping function with n_0 input elements and n_L output elements. An MLP network is composed of L layers, with n_l (l = 1, 2… L) neurons in layer l. When the layer l is not 0 or L the layer is referred to as a hidden layer. To formalize this description, if x_i^l is the output of the neuron of layer l, w_{ij}^l, is the weight of the j^{th} input (coming from the j^{th} neuron of the preceding layer) and ϕ^l is the activation function, then:

$$x_i^l = \phi^l\left(\bar{x}_i^l + b_i^l\right) = \phi^l\left(\sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1} + b_i^l\right) \tag{12}$$

where, x_i^{l-1} is the output of the j^{th} neuron in the previous layer l-1 and b_i^l is the bias, introduced to allow the neuron to present a non-null output even in the presence of a null input (Stinchcombe and White, 1989). While an MLP can have any number of layers, L, for most applications a single hidden layer, or two total layers, is sufficient as is shown in Fig. 2. This system has p inputs, q hidden nodes and r outputs. In matrix form, the complete expression can be formulated as:

$$y = w_2\left[\phi\left(w_1 x + b_1\right)\right] + b_2 \tag{13}$$

Training algorithm: Training a neural net generally consists of applying a method to adjust or estimate the neuron weights. The training process minimizes the neural net output error through the application of an optimization method. All methods need to know how the net output varies with respect to the variation of a given neuron weight.

The back propagation algorithm is an algorithm for learning in feed forward networks using mean squared error and gradient descent (Rumelhart *et al*., 1986). The error that we choose to minimize for our training algorithm is:

$$j = \frac{1}{2}\sum_k e_k^2(n) \tag{14}$$

calculated at a given time step, n.

The calculated error at the output layer is:

$$e_k(n) = d_k(n) - y_k(n) \tag{15}$$

where d_k is the desired output of element k and y_k is the hidden layer output response specified by:

$$y_k(n) = \phi(v_k(n)) \tag{16}$$

In (6), v_k is the output layer activation level:

$$v_k(n) = \sum_j w_{kj}(n) y_j(n) \tag{17}$$

This continues through the hidden layer to the input layer:

$$y_j(n) = \phi(v_j(n)) \tag{18}$$

and

$$v_j = \sum_i w_{ji}(n) x_i \tag{19}$$

To calculate the gradient of the error function, we use the chain rule to get:

$$\begin{aligned}\frac{\partial J}{\partial w_{kj}} &= \frac{\partial J}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial w_{kj}} \\ &= e_k(n)(-1)\phi'(v_k)y_j(n)\end{aligned} \tag{20}$$

This derivative allows the weight update to be calculated, using a learning parameter, ς_1:

$$\Delta w_{kj} = -\eta_l \frac{\partial J}{\partial w_{kj}} \tag{21}$$

The resulting weight update is then:

$$\begin{aligned}w_{kj}(n+1) &= w_{kj}(n) + \Delta w_{kj} \\ &= w_{kj}(n) + \eta_l e_k(n)\phi'(v_k)y_j(n)\end{aligned} \tag{22}$$

To simplify the equations, we calculate an intermediate delta value,δ, for the output layer:

$$\delta_k \overset{\Delta}{=} e_k \phi'(v_k) \tag{23}$$

which gives the final weight update equation for the output layer:

$$w_{kj}(n+1) = w_{kj}(n) + \eta_l \delta_k y_j(n) \tag{24}$$

This operation is repeated for the hidden layer:

$$\begin{aligned}\frac{\partial J}{\partial w_{ji}} &= \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial v_i}\frac{\partial v_i}{\partial w_{ji}} \\ &= -\sum e_k \phi'(v_k) w_{kj} \phi'(v_j) y_i(n)\end{aligned} \tag{25}$$

and again defining an intermediate δ:

$$\delta_j \overset{\Delta}{=} -\frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial v_i} = \sum e_k \phi'(v_k) w_{kj} \phi'(v_j) \tag{26}$$

gives the final weight update:

$$w_{ji}(n+1) = w_{ji}(n) + \eta_l \delta_j y_i(n) \tag{27}$$

Activation function: The nonlinear neuron computes its activation level by adding all the weighted activations it receives. It then transforms this activation level into a response using an activation, or squashing, function ϕ. Several squashing functions can be used. The most common one is the logistic function:

$$\phi(x) = \log(x) = \frac{1}{1+e^{-x}} \tag{28}$$

This function maps the set of real numbers into the (0, 1) range. The derivative is easy to compute:

$$\phi'(x) = \phi(x)(1-\phi(x)) \tag{29}$$

which is useful for computing the back-propagation error. The hyperbolic tangent is also often used when the response range is the interval (-1, +1):

$$\phi(x) = \tan h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{30}$$

Its derivative is also easy to compute:

$$\phi'(x) = 1 - \phi^2(x) \tag{31}$$

There are many such activation functions that can be used. Choice of which to use in which layers is left to the designer of the network based on the dynamics of the system.

SIMULATION RESULTS

To illustrate the performance of the proposed estimator, an initial set of simulation results will be presented. In these simulations we have utilized the model of a Puma 560 type robot manipulator as presented in Fig. 1.

The networks considered in this paper for approximating the filter functional forms are the feedforward multilayer perceptron (FFMLP) illustrated in Fig. 4, the state filter utilizing neural networks assumes that a deterministic system model is available with the structural form of Eq. 3. The neural is chosen to be a three layer FFMLP network with eighteen nodes in the first layer, corresponding to six component of torque, six component of orientation q and six component of velocity qd. The twelve nodes in the third layer correspond to outputs q and qd. The number of units in the hidden layer is determined experimentally, from studying the network behavior during the training process taking into consideration some factors like convergence rate, error criteria, etc. In this regard, different configurations were tested and the best suitable configuration was selected based on the accuracy level required. The number of hidden units for the MLP is 40.

In this study, an attempt is made to develop and compare EKF with the developed neural state filters.

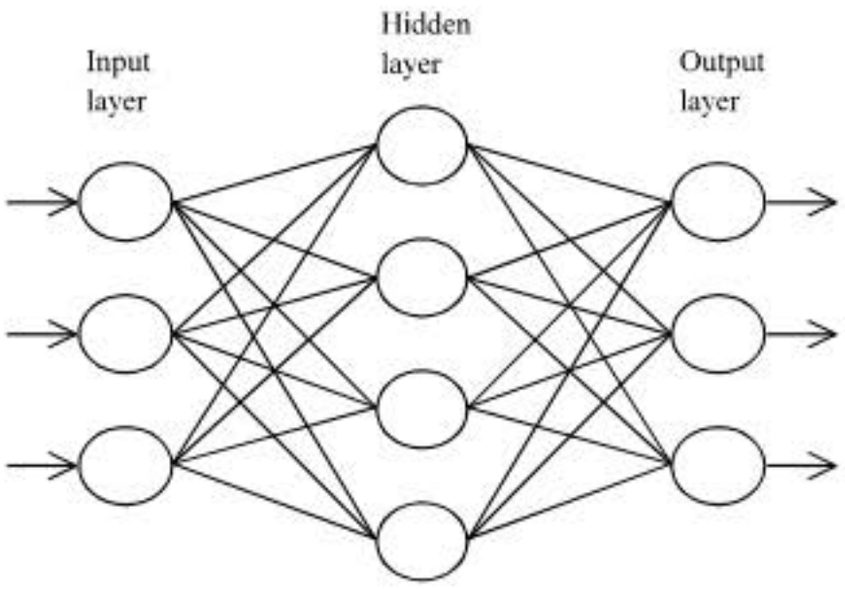

Fig. 4: Topology of MLP network

For implementation of the EKF, the covariance matrices of the process noise and measurement noise were tuned empirically to obtain the successful values.

$$Q = \begin{bmatrix} 0.001 & & & & & & & & & & & \\ & 0.01 & & & & & & & & & & \\ & & 0.001 & & & & & & & & & \\ & & & 0.01 & & & & & & & & \\ & & & & 0.001 & & & & & & & \\ & & & & & 0.01 & & & & & & \\ & & & & & & 0.001 & & & & & \\ & & & & & & & 0.01 & & & & \\ & & & & & & & & 0.001 & & & \\ & & & & & & & & & 0.01 & & \\ & & & & & & & & & & 0.001 & \\ & & & & & & & & & & & 0.01 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.001 & & & & & & \\ & 0.001 & & & & & \\ & & 0.001 & & & & \\ & & & 0.001 & & & \\ & & & & 0.001 & & \\ & & & & & 0.001 & \\ & & & & & & 0.001 \end{bmatrix}$$

The initial guess of the covariance matrix P_o was simply zero.

After simulating the predicting trajectories and the true trajectory are shown in Fig. 5 (Position estimation) and Fig. 6 (Velocity estimation) respectively. Moreover the Fig. 7 and 8 exhibit error comparison of algorithms.

From the results obtained it can be observed that the neural estimator is more accurate than the EKF. In Fig. 6 it is clear that the EKF is not robust to estimate the velocity of the arm manipulator.

As can be seen from Fig. 7 and 8, the estimation errors for neural filter proposed decrease fast after the first estimation. The average RMS error for the EKF was 0.0309, whereas the neural observer presents an average

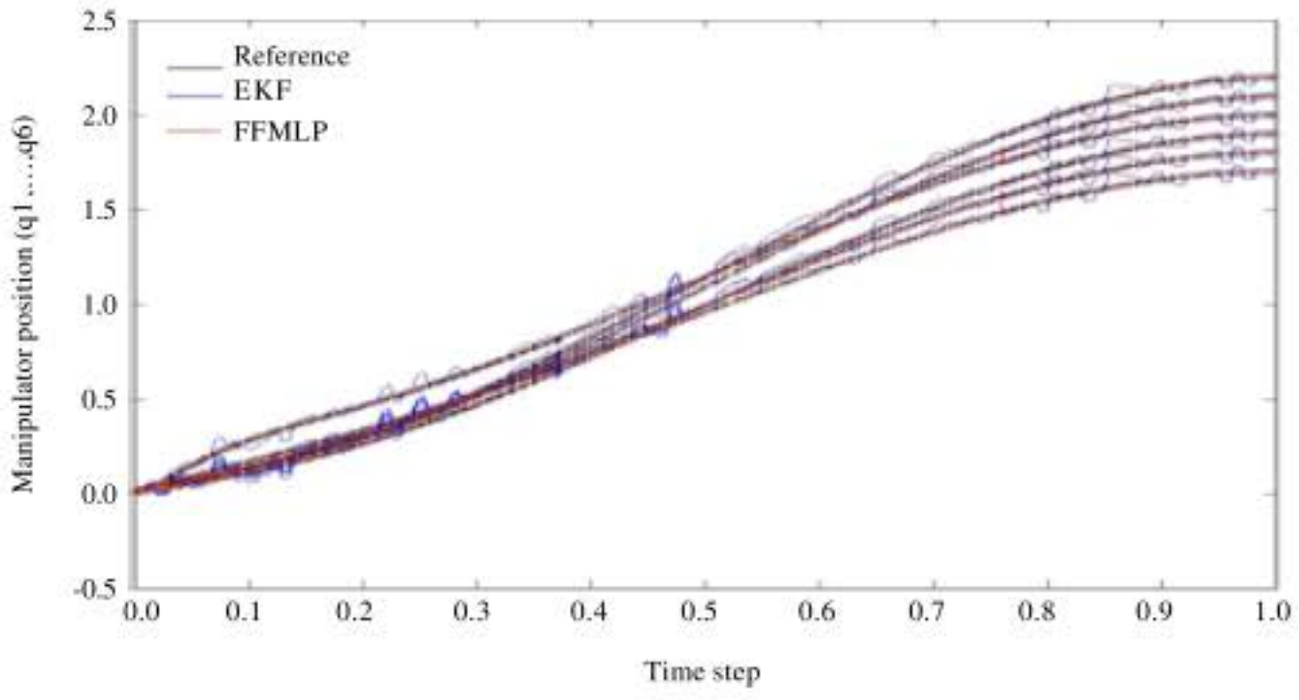

Fig. 5: Position estimation for six DOF PUMA 560 arm manipulato

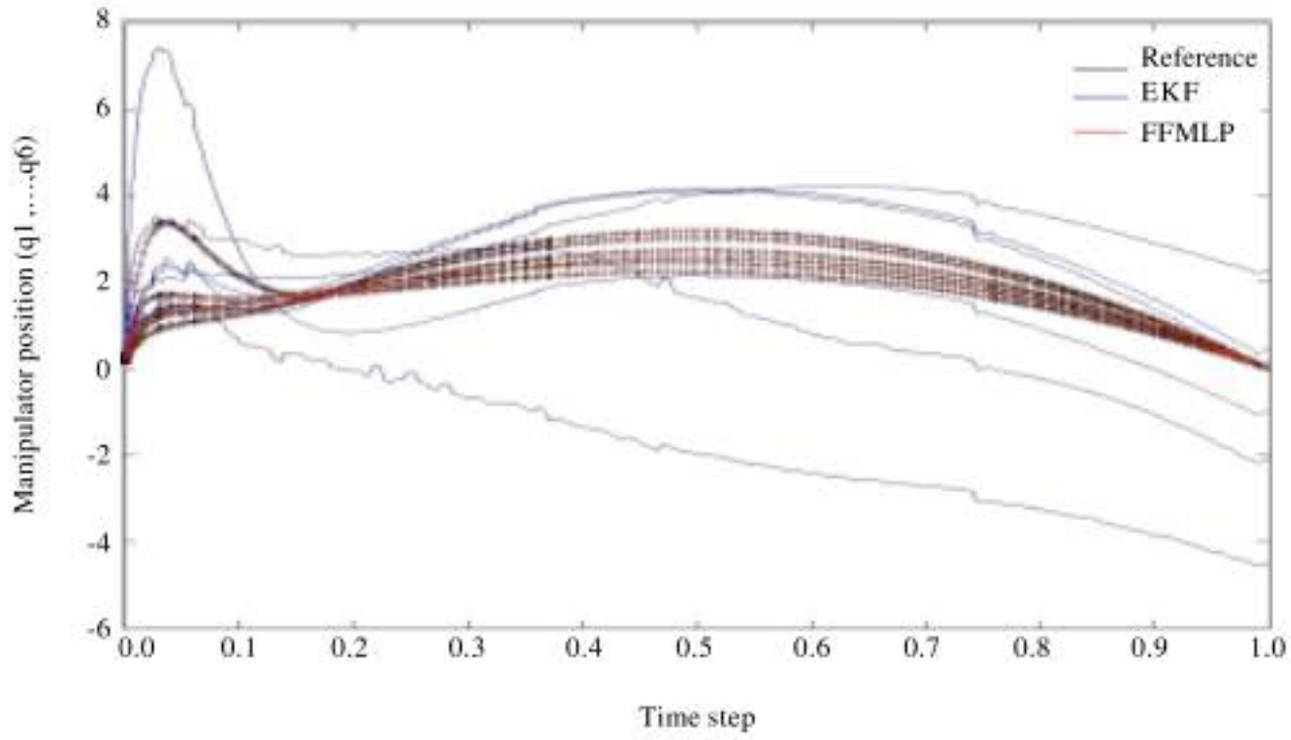

Fig. 6: Velocity estimation for six DOF PUMA 560 arm manipulator

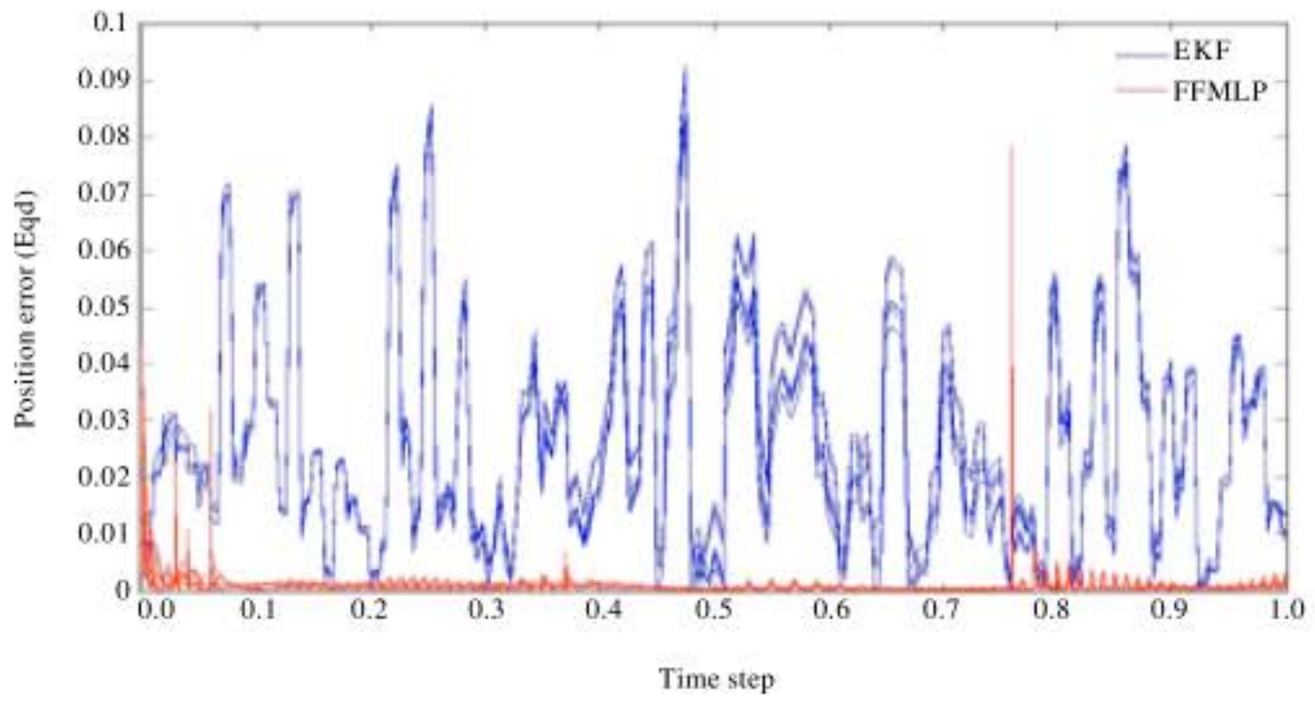

Fig. 7: Position estimation error for both methods

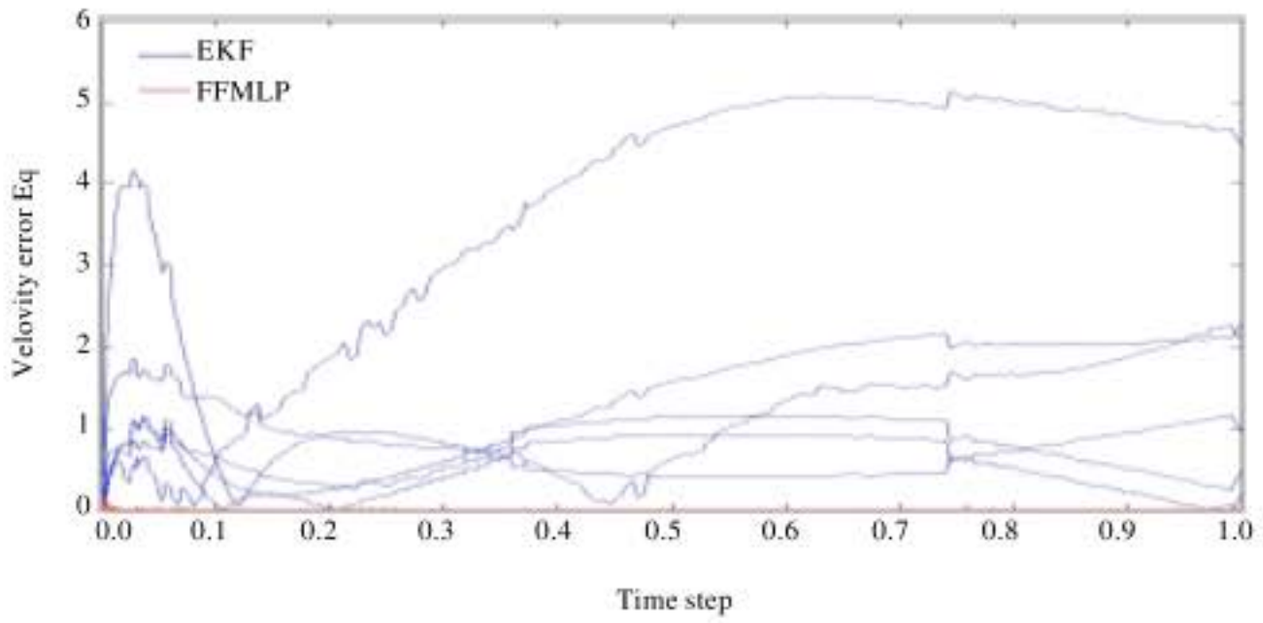

Fig. 8: Velocity estimation error for both methods

error of 0.0035. From simulations it can be said the neural filter proposed for the studied application is more consisted and convergent than the EKF filter.

In this study we present a new method for developing neural network-based state filters for use when system models are not available or they are inaccurate (Menon *et al*., 1998). The study demonstrates

the accuracy and effectiveness of the state filtering method on a real-world process state and parameter filtering problem.

DISCUSSION

More recently, neural networks have been proposed for use in state estimation problems for approximating nonlinear mappings between the states, inputs and outputs. No constraining assumptions are needed about the structure of the dynamic system investigated and the nature of the stochastic process and or measurement noises. This is unlike traditional state estimation methods, where explicit knowledge of the system dynamics and the stochastic processes affecting the system must be available or assumed.

In this study, the Neural Networks as a tool for state estimation has been used in robot arm manipulator state estimation and compared to the standard method for nonlinear state estimation the EKF. The state estimation methods have been compared using the same tuning parameters to make the comparative study as credible as possible.

The results show promise for using neural networks for developing state filters for use when system models are not available or they are inaccurate. The study demonstrates the accuracy and effectiveness of the adaptive state filtering method on a real-world process state and parameter filtering problem.

CONCLUSION

The extended Kalman filter has been widely used as a position estimation method for mobile robot applications. The system model is generally nonlinear in mobile robotics, so the classical extended Kalman filter for mobile robot position estimation suffers from a fundamental flaw. Linear approximation of nonlinear system equations with first order Taylor series induces linearization error. To overcome these drawbacks a new estimation method based on neural networks was proposed in this paper. The simulation results show an improvement of the new filter. It avoids linear approximation of nonlinear system equations and is free of linearization error. The filter is consistent and convergent. Comparing with EKF, it gives more conservative estimation result.

REFERENCES

Armstrong, B., O. Khatib and J. Burdick, 1986. The explicit dynamic model and inertial parameters of the puma 560 arm. IEEE Int. Conf. Robotics Automation, 3: 510-518.

Billings, S.A., H.B. Jamaluddin and S. Chen, 1992. Properties of neural networks with applications to modeling non-linear dynamical systems. Int. J. Control, 55: 193-224..

Boutayeb, M., H. Rafaralahy and M. Darouach, 1997. Convergence analysis of the extended kalman filter used as an observer for nonlinear deterministic discrete-time systems. IEEE Trans. Automatic Control, 42: 581-586.

Crassidis, J.L. and J.L. Junkins, 2004. Optimal Estimation OF Dynamic Systems. Chapman and Hall/Crcpress, USA,ISBN: 9781584883913.

Haykin, S., 1994. Neural Networks A Comprehensive Foundation. Macmillan College Publishing, New York, USA.

Haykin, S., 1996. Adaptive Filter Theory. 3rd Edn., Prentice Hall, New Jersey.

Haykin, S., 1999. Neural Networks: A Comprehensive Foundation. 2nd Edn., Prentice Hall, New Jersey.

Jonsson, G. and O.P. Palsson, 1994. An application of extended Kalman filtering to heat exchanger models. ASME J. Dyn. Syst. Meas. Contr., 116: 257-264.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Trans. ASME. J. Basic Eng., 82: 35-45.

Ljung, L. and T. Sjoberg, 1992. A system identification perspective on neural nets. Proceedings of the IEEE Workshop Neural Networks for Signal Processing, Aug. 31-Sept. 2, Helsingoer, Denmark, pp: 423-435.

Menon, S.K., A.G. Parlos and A.F. Atiya, 1998. Nonlinear state estimation in complex system using neural networks: part I algorithms, part-II applications. IEEE Transactions on Neural Networks, 1998.

Naraendra, K.S. and K. Parthasarathy, 1990. Identification and control of dynamic systems using neural networks. IEEE Trans. J. Neural. Networks, 1: 4-27.

Reif, K., S. Gunther, E. Yaz and R. Unberhauen, 1999. Stochastic stability of the discrete-time extended kalman filter. IEEE Trans. Automatic Control, 44: 714-728.

Rochefort, D., J. de Lafontaine and C.A. Brunet, 2005. A new satellite attitude state estimation algorithm using quaternion neural networks, Paper AIAA-2005-6447. Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, San Francisco, Aug. 15-18.

Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986. Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, Rumelhart, D.E. and J.L. McClelland (Eds.). MIT Press, Cambridge, ISBN: 0-262-68053-X, pp: 318-362.

Stinchcombe, M. and H. White, 1989. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. IEEE Int. Joint Confer. Neural Networks, 1: 613-617.