Jestr

Journal of Engineering Science and Technology Review 10 (4) (2017) 191- 198

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Neural Networks Trained with Levenberg-Marquardt-Iterated Extended Kalman Filter for Mobile Robot Trajectory Tracking

Ben Cherif Aissa* and Chouireb Fatima

Laboratory of Telecommunications Signals and Systems, Department of Electronics, University Amar Telidji, Laghouat, Algeria

Received 4 March 2017; Accepted 17 September 2017

Abstract

This paper proposes a neural network controller using a new efficient optimisation algorithm for learning that is the Levenberg-Marquart Iterated Extended Kalman filter LM-IEKF. The trained neural network is applied to control a wheeled mobile robot for trajectory tracking problem. The proposed algorithm is compared to the standard extended Kalman filter and the back-propagation algorithms. Simulation and experimental results using MATLAB 7.1 and National Instrumentation mobile robot (starter kit 2.0) respectively show that in terms of mean squared errors, the proposed algorithm is superior to the extended Kalman filter and back-propagation. This indicates that Levenberg-Marquart iterated extended Kalman filter based neural networks learning could be a good alternative in the artificial neural networks based applications for mobile robot trajectory tracking.

Keywords: Mobile robot control, Extended Kalman Filter, Levenberg-Marquardt-Iterated-EKF, neural networks controller.

1. Introduction

In recent years, intelligent mobile robots are the subject that has received large attention. It is a topic of great research concern arising from the possibility of real applications in many areas, such as manufacturing, aerospace, civil engineering, transportation, agriculture, military operations exploration, help for disabled, and medical surgery and in other areas of science and technology research [1]. These applications require mobile robots to have the ability to track a reference trajectory. Thus, the stable trajectory tracking control of mobile robots has attracted significant attention from researchers.

There are many recent studies that addressed the problem of mobile robot control, by suggesting kinematic-based controllers for trajectory tracking problems, such as linear feedback control [2], backstepping control [3][4], time-varying feedback control [5], sliding mode control [6][7], but these algorithms have problems with complex trajectories, uncertainty and unlimited velocities. In response to these complex control issues, a number of advanced controllers have recently been proposed, typically artificial neural networks (ANNs) [8][9][10].

The artificial neural network, in general, is a system of programs and data design that approximates the process of the human brain [11]. In over the last decade, neural networks have been used to solve the trajectory tracking control problem for a mobile robot. Several studies have proposed different architectures for neural networks control, but most of these researches have not focused on the neural network learning. The learning operation consists of finding the optimal synaptic weights and biases of the neural network, this can be solved with the common classical gradient descent used in the back-propagation training method. However, the gradient method usually behaves very slowly and is not assured to find the global minimum of the error function. These are the reasons for searching for the most effective methods of neural network training.

An effective tool for training neural networks in the last decades is the extended Kalman filter (EKF) [12][13], which has become popular as an algorithm for state and parameters estimation. This is because it is easy to implement and exhibit computationally efficient calculation which is especially useful for nonlinear systems and practical applications see [14]. There are many variables that affect EKF training algorithm performances. These variables are matrices that must be correctly initialized otherwise the EKF training algorithm can exhibit poor performance. These matrices are the estimation error covariance matrix (P), the measurement covariance matrix (R), and the additional process noise matrix (Q) [14]. the iterative version of EKF is the iterative extended Kalman filter (IEKF), which improves the linearization of the extended Kalman filter by recursively, this version is powerful than the standard EKF for neural network training [15].

Another algorithm has a better performance for training neural networks than EKF is Levenberg-Marquardt algorithm see [16]. The Levenberg-Marquardt method is a standard technique used to solve nonlinear least squares problems [17][18].

A modification based on an optimisation viewpoint is done by including the Levenberg-Marquardt algorithm in the iterated extended Kalman filter [19]. The Levenberg-Marquardt-iterated Kalman filter is made to include a diagonal damping matrix which could further speed up convergence, with results that exceed the performance of the IEKF state estimation in nonlinear systems. In the estimation

*E-mail address a.bencherif@lagh-univ.dz
ISSN: 1791-2377
doi:10.25103/jestr.104.23

of parameters of neural networks, LM-IEKF is better than the EKF and IEKF, with the significant advantage that it does not need the calculation of the Jacobian of the neural network.

This paper presents the principle of neural network training method based on LM-IEKF, which serves as a better alternative to the classical methods back-propagation and standard EKF, and proposes a mobile robot trajectory tracking control using the LM-IEKF based neural network training algorithm. The effectiveness and efficiency of the proposed control approach are proved by simulation results and experimental tests.

The following sections are organised as follows: In Section 2, we describe the mobile robot kinematic model, Section 3 describes the neural network controller design. In Section 4, we detail the neural network training methods, whereas Section 5 and Section 6 present simulation and experimental results respectively, and finally, the conclusion is presented in section 7.

2. Kinematic Model of Mobile Robot

In this work, we consider a trajectory tracking control problem of the mobile robot as shown in Fig.1.

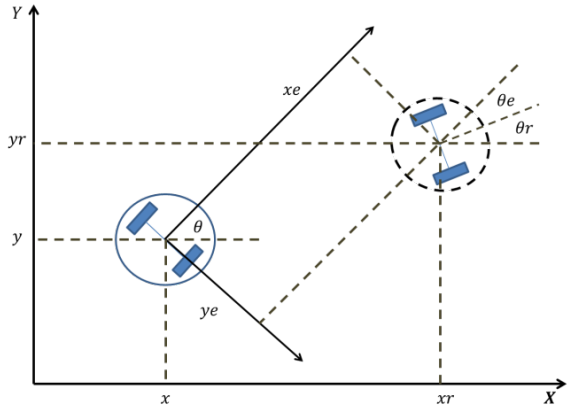

Fig. 1. The error coordinates of WMR

The kinematics model (or equation of motion) of a wheeled mobile robot is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\ cos(\theta) \\ v\ \sin(\theta) \\ w \end{bmatrix} \quad (1)$$

Where x and y are coordinates of the centre of mobile robot gear, θ is the angle that represents the orientation of the mobile robot, v and w are linear and angular velocities of the robot.

To consider a trajectory tracking problem, a reference trajectory should be generated as follow:

$$\dot{q}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v_r\ \cos(\theta_r) \\ v_r\ \sin(\theta_r) \\ w_r \end{bmatrix} \quad (2)$$

The error coordinates represented by the world coordinates are

$$q_r - q = \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \quad (3)$$

In the view of moving coordinates, the error coordinates are transformed into:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \quad (4)$$

3. Neural network controller design

In this section, we propose two neural network controllers for mobile robot trajectory tracking as shown in figure.2. We assume that each neural network controller under consideration consists of two inputs i.e. x_e and $\dot{x}_e$ for linear velocity, y_e and θ_e for angular velocity (figure.3).

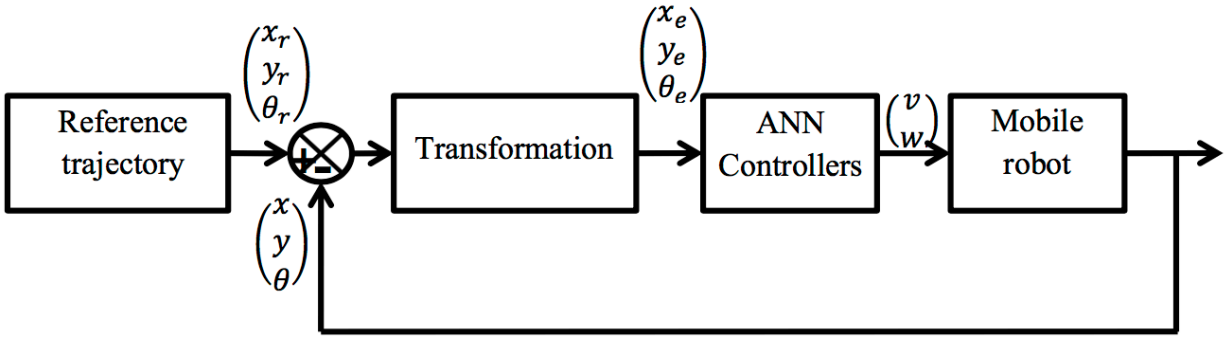

Fig. 2. Neural network controllers for Mobile Robot Trajectory Tracking

The proposed architecture of neural network controllers is composed of three layered static neural networks (Fig. 3). The input layer contains two neurones, the hidden layer has seven neurones and one neurone in the output layer.

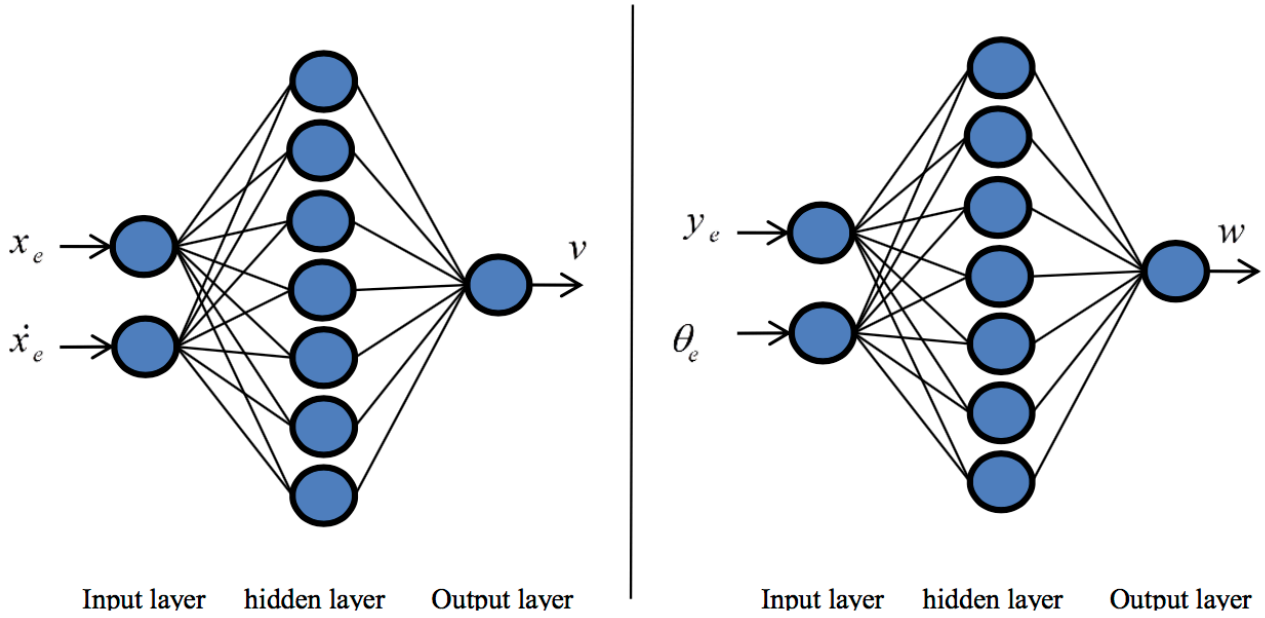

Fig. 3. The neural network's architecture

The activation function used for hidden layer in both neural controllers is tangent sigmoid function

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{5}$$

While pure linear function is employed in output layer

The neural networks with the training data sets are trained offline. During training, for each sample value, the error is calculated between the desired output and neural network output using the following equations.

$$out(j)=\sigma\left(\left(\sum_{i=1}^{m} w_{i,j}\ x_i\right)+b_j\right)$$

$$out(k)=\sum_{j=1}^{n} w_{j,k}\ out(j)+b_k \tag{6}$$

Where n is the number of neurones and $out(j)$ is The output of each neurone in hidden layer, $out(k)$ is The output of each neurone of output layer, $\sigma(x)$ is The activation function, m is The number of inputs, $w_{i,j}$ is The weight from input layer node i to hidden layer node j , $w_{j,k}$: The weight from hidden layer node j to output layer node, b_j are the Biases of node j of hidden layer, b_k are the Biases of a node k of output layer.

The error is then minimised using extended Kalman filter or Levenberg-Marquardt Iterated EKF algorithms. The algorithm minimises the error by updating the weights and biases of the neural networks.

4. Training neural networks controllers

Once the neural networks have been structured, they will be ready to be trained. To start this process the initial weights are chosen randomly. Then, the training or Learning begins using the Extended Kalman filter or Levenberg-Marquardt Iterated Extended Kalman filter.

4.1 Extended Kalman filter

The Kalman filter is an optimal estimator tool which is able to estimate both linear and non-linear systems [19]. The Kalman Filter can perform the estimation in the presence of noise in the system and sensors. When the systems are dynamic and non-linear, the use of the Extended Kalman Filters is applied through the linearization at each time step of the system. The discrete nonlinear system model is given by:

$$\begin{aligned} X_k &= f(X_{k-1})+\zeta_{k-1} \\ Y_k &= h(X_k)+v_k \end{aligned} \tag{7}$$

The first equation describes the state transition relationship, where X_k is the state vector and $\zeta_{k-1}\in\mathcal{R}^n$ is the unknown random noise.

The second equation represents the nonlinear output model where v_k is the white Gaussian measurement noise given by:

$$v_k=\aleph(0,R_k)\,.$$

The extended Kalman filter uses the Jacobian of the (nonlinear) functions appearing in the state transition equation, and the measurement equation, respectively

- ✓ the Jacobian matrix of partial derivatives of the system f with respect to X
- ✓

$$A_{[i,j]}=\frac{\partial f_{[i]}}{\partial X_j}\left(\hat{X}_{k-1},0\right) \tag{8}$$

- ✓ the Jacobian matrix of partial derivatives of measurement h with respect to X

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial X_j}\left(\hat{X}_k, 0\right) \tag{9}$$

The implementation of the extended Kalman algorithm is as follows:

➢ the Kalman filter time update equations (or prediction) are given by[20]:

$$\tilde{X}_k = f\left(\hat{X}_{k-1}\ , 0\right) \tag{10}$$

$$P_k^- = A_k P_{k-1} A_k^{\ T} + Q_{k-1} \tag{11}$$

Where $\tilde{X}_k$ represents the predicted state and P_k^- the covariance matrix of the prediction error.

➢ The update equations of the Kalman filter (or correction) are given by:

$$K_k = P_k^- H_k^{\ T}\left(H_k P_k^- H_k^{\ T} + R\right)^{-1}$$

$$\hat{X}_k = \hat{X}_k^- + K_k\left(Z_k - h\left(\hat{X}_k^-\ , 0\right)\right)$$

$$P_k = \left(I - K_k H_k\right) P_k^- \tag{12}$$

With K_k is the Kalman gain, $\hat{X}_k$ is the estimated state at time k, P_k is the covariance matrix of the estimation error and Z_k is the measurement.

For neural network training, the weights w of the network are the states the Kalman filter attempts to estimate using all observed data. The discrete nonlinear system for neural network training process is shown in the equations below[21]:

$$w_{k+1} = w_k + \xi_k \tag{13}$$

$$Z_{k+1} = h(w_{k+1}, u_k) + v_k \tag{14}$$

where $h(\bullet)$ is the function of the neural network, ζ_k and v_k are the system and measurement artificial noises. These noises are assumed to be white Gaussian noises with zero mean and covariance matrices Q and R (learning rate) respectively. Z_k is the output of the neural network, u_k is the input vector and w_k is the state vector which includes all parameters of the neural network. Its dimension *ns* is determined by the number of inputs *m*, hidden neurones *n* and outputs *ny* :

$$ns = \left(m\ \ n\right) + n + \left(n\ \ ny\right) + ny \tag{15}$$

The algorithm of Kalman filter for training the neural network is summarised[21][22] below:

State estimate propagation

$$\hat{w}_k^- = \hat{w}_{k-1} \tag{16}$$

$$P_k^- = A_k P_{k-1} A_k^{\ T} + Q_{k-1} \tag{17}$$

Kalman gains matrix

$$K_k = P_k^- H_k^{\ T}\left(H_k P_k^- H_k^{\ T} + R_k\right)^{-1} \tag{18}$$

State estimate update

$$\hat{w}_k = \hat{w}_k^- + K_k(Z_k - h(\hat{w}_k^-, 0)) \tag{19}$$

Error covariance update

$$P_k = (I - K_k H_k) P_k^- \tag{20}$$

4.2 Iterated extended Kalman filter (IEKF)

The purpose of the iterated Kalman filter update [23] is to repeatedly calculate the measurement Jacobian each time linearizing about the most recent estimate $\hat{X}_k$. On the other hand, the EKF, the measurement Jacobian is linearized about the predicted state estimate $\hat{X}_k$. The iteration is initialized by choosing

$$\hat{X}_{i=0} = \hat{X}_k \tag{21}$$

$$P_{i=0} = P_k \tag{22}$$

The implementation of iterated Extended Kalman filter algorithm is as follows [23][15]:

$$K_i = P_k^- H_i^{\ T}\left(H_i P_k^- H_i^{\ T} + R\right)^{-1} \tag{23}$$

$$\hat{X}_{i+1} = \hat{X}_i + K_i\left(Y_k - h\left(\hat{X}_i\ , 0\right) - H_i\left(\hat{X}_k^- - \hat{X}_i\right)\right) \tag{24}$$

$$P_i = \left(I - K_i H_i\right) P_k^- \tag{25}$$

$$\hat{X}_k = \hat{X}_{i+1} \tag{26}$$

$$P_k = P_i \tag{27}$$

With K_i is the Kalman gain for each iteration i>0, $\hat{X}_k$ is the estimated state at time k and P_k is the covariance matrix estimate.

For a single iteration, setting i = 0 in (23),(24) and (25) above, we obtain the conventional EKF update formulas in the past section

The algorithm of Iterated Extended Kalman filter for training the neural network is summarised below with consideration of state equations as in (13) and (14)

State estimate propagation

$$\hat{w}_k^- = \hat{w}_{k-1} \tag{28}$$

$$P_k^- = A_k P_{k-1} A_k^{\ T} + Q_{k-1} \tag{29}$$

Kalman gains matrix

$$K_i = P_k^- H_i^{\ T}\left(H_i P_k^- H_i^{\ T} + R\right)^{-1} \tag{30}$$

State estimate update

$$\hat{w}_{i+1} = \hat{w}_i + K_k(Z_k - h(\hat{w}_i, 0) - H_i(\hat{w}_k^- - \hat{w}_i)) \quad (31)$$

Error covariance update

$$P_i = \left(I - K_i H_i\right) P_k^- \quad (32)$$

$$\hat{w}_k = \hat{w}_{i+1} \quad (33)$$

$$P_k = P_i \quad (34)$$

4.3 Levenberg-Marquardt Iterated Kalman filter

4.3.1 Levenberg-Marquardt Algorithm

The LMA is used in many software applications for solving nonlinear least squares problems. However, as for many algorithms, the objective function of least squares problem is then formulated as

$$F(\rho) = \frac{1}{2}\sum_{i=1}^{n} r_i^2 \quad (35)$$

$$r_i(x) = f\left(x_i; \rho\right) - y_i \quad (36)$$

Where n is the total number of data considered, r_i is residual, and y_i is y component of the data at x_i and $\rho = [\rho_1 \rho_2 .... \rho_m]$ the model parameters vector and m are the total number of parameters.

The LMA finds only a local minimum, but not necessarily the global minimum. First Levenberg (1944) [24] suggested algorithm with the following search scheme to update model parameters vector: $\rho = [\rho_1 \rho_2 .... \rho_m]$ [17]

$$(J_k^T J_k + \mu I)\delta\rho_k = -J_k^T(r_k) \quad (37)$$

$$(J_k^T J_k + \mu I)\delta\rho_k = -J_k^T(y - \hat{y}) \quad (38)$$

Where $\delta\rho_k$ is the perturbation to the estimated parameters, J_k is the Jacobian matrix of residuals, and μ is a damping factor.

Marquardt proposed update by replacing the identity matrix I in the original equation of levenberg, with the diagonal of $J_k^T J_k$ resulting into Levenberg-Marquardt algorithm [25]

$$(J_k^T J_k + \mu diag(J_k^T J_k))\delta\rho_k = -J_k^T(y - \hat{y}) \quad (39)$$

$$\rho_{k+1} = \rho_k + \delta\rho_k \quad (40)$$

Where large values of the algorithmic parameter μ result in a gradient descent update and small values of μ result in a Gauss-Newton update. If we get worse results in an approximation, we increase the value of μ. But if the solution is improved, we decrease the value of μ, the Levenberg-Marquardt method approximates the Gauss-Newton method, and the solution typically accelerates to the local minimum.

4.3.2 Levenberg-Marquardt Iterated Extended Kalman filter

The LM-IEKF can be improved by replacing the diagonal damping with $\mu diag(J_k^T J_k)$ in (38) and by applying this to the following equations as in [19]

$$X_{k+1} = X_k - (J_k^T J_k)^{-1} J_k^T r_k \quad (41)$$

By doing so, larger steps are made in directions where the gradient is small, further speeding up convergence, results in

$$\hat{X}_{i+1} = \hat{X}_{k-1} + K_i \begin{pmatrix} \left(Z_k - h\left(\hat{X}_i, 0\right) - H_i \hat{X}_i\right) - \\ -\mu\left(I - K_i H_i\right) P_k^- B_i^{-1} \hat{X}_i \end{pmatrix} \quad (42)$$

$$P_k^- = \left(I - P_{k-1}\left(P_{k-1} + \frac{1}{\mu} B_i^{-1} \right)^{-1} \right) P_{k-1} \quad (43)$$

$$B_i = diag\left(J\ J^T\right) = diag\left(H_i R H_i^T + P_k^{-1}\right) \quad (44)$$

$$K_i = P_k^- H_i^T \left(H_i P_k^- H_i^T + R\right)^{-1} \quad (45)$$

$$\hat{X}_k = \hat{X}_{i+1} \quad (46)$$

$$P_k = P_i \quad (47)$$

There are two parameters in the LM-IEKF algorithm, which have to be set. The K_k^i is again to be found using the exact line search (41) because it influences the results. The selection of the damping parameter μ is rather difficult because it influences the step-length too. Some discussion on this topic can be found in [19].

4.3.3 Training neural networks with Levenberg-Marquardt Iterated EKF

The algorithm of the Levenberg-Marquardt iterated Kalman filter for training the neural network is summarised below:

State estimate propagation

$$\hat{w}_k^- = \hat{w}_{k-1} \quad (48)$$

$$P_k^- = \left(I - P_{k-1}\left(P_{k-1} + \frac{1}{\mu} B_i^{-1} \right)^{-1} \right) P_{k-1} \quad (49)$$

$$B_i = diag\left(J\ J^T\right) = diag\left(H_i R H_i^T + P_k^{-1}\right) \quad (50)$$

Kalman gains matrix

$$K_i = P_k^- H_i^T \left(H_i P_k^- H_i^T + R\right)^{-1} \quad (51)$$

State estimate update

$$\hat{w}_{i+1} = \hat{w}_{k-1} + K_i \begin{pmatrix} \left(Z_k - h\left(\hat{w}_i, 0\right) - H_i \hat{w}_i\right) - \\ -\mu\left(I - K_i H_i\right) P_k^- B_i^{-1} \hat{w}_i \end{pmatrix} \quad (52)$$

Error covariance update

$$P_i = \left(I - K_i H_i\right) P_k^- \quad (53)$$

$$\hat{w}_k = \hat{w}_{i+1} \quad (54)$$

$$P_k = P_i \quad (55)$$

5. Simulation results

This section presents numerical results for trajectory tracking control of a wheeled mobile robot using the trained neural networks. The simulation was done under MATLAB to demonstrate the effectiveness of the proposed training algorithms.

For the neural networks, we have 7 neurones in the hidden layer, 2 neurones in the input layer and a neurone in the output layer for each controller.

$n = 7$: The number of neurons in hidden layer
$m = 2$: The number of neurons in input layer
$ny = 1$: The number of neurons in output layer

The number of training parameters is then given by:

$$ns = \left(m \;\; n\right) + n + \left(n \;\; ny\right) + ny = 29$$

which is the total number of weights and biases to estimate.

The initial state and initial state and error covariance matrix are given by:

$$\hat{X}_0 = \left[\hat{w}_{i,j}^0; \hat{w}_{j,k}^0; \hat{b}_j^0; \hat{b}_k^0\right] = randn\left(ns,1\right)$$

$$P_0 = 1000 \; eye\left(ns,ns\right)$$

The noises covariance matrices are given as follow:

$$Q = 0.001 \; eye\left(ns,ns\right)$$

$$R = 1000$$

To obtain a good estimation of weights and biases with Levenberg-Marquardt Iterated EKF, we train the two neural networks for a different number of updating iterations. Table.1 resumes the results obtained. These results show that 7 iterations have small RMS errors; this number is fixed for all simulation and experimental results.

Table.1 LMIEKF RMS training errors

ANNs	3 iterations	5 iterations	7 iterations
ANN1	0.4118	0.3581	0.2530
ANN2	0.1133	0.1070	0.1048

In the training phase, figures 4 and 5 shows the results of the trained neural network for an angular velocity controller with LM-IEKF and EKF respectively, the RMS training errors for both velocity controllers are represented in the table. 2. These results show that the LM-IEKF has a good performance to better estimate the parameters of neural networks than EKF

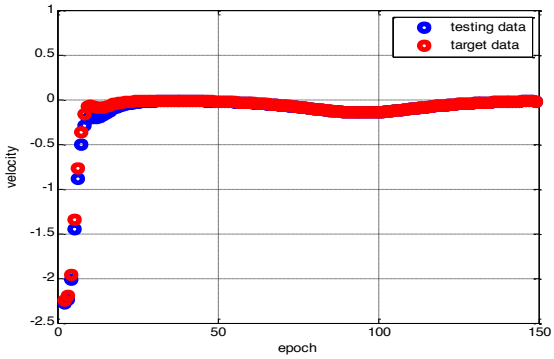

Fig. 4. Neural network training with LM-IEKF

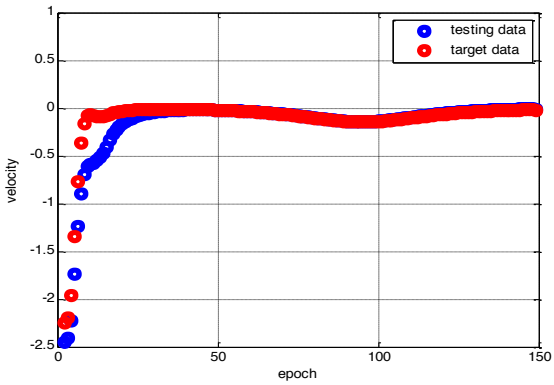

Fig. 5 Neural network training with EKF

Table. 2. Training RMSE

Training method	v	w
EKF	0.6914	0.2330
LM-IEKF	0.2530	0.1048

Figures 6-8 shows the mobile robot trajectory tracking for different reference trajectories: sinusoidal, lemniscate, circular respectively, where the green line represents the reference trajectory, the red and black are the real trajectory with a neural network trained with EKF and LM-IEKF respectively. These two methods are also compared with neural networks trained with LMBP represented with a magenta line.

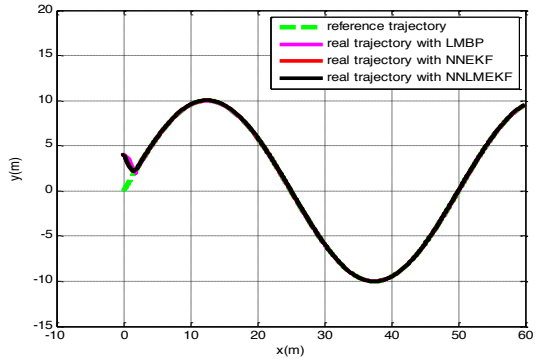

Fig. 6. Tracking response of a mobile robot with NN controllers for sinusoidal trajectory with initial position $X_0 = \left[0;4;0\right]$

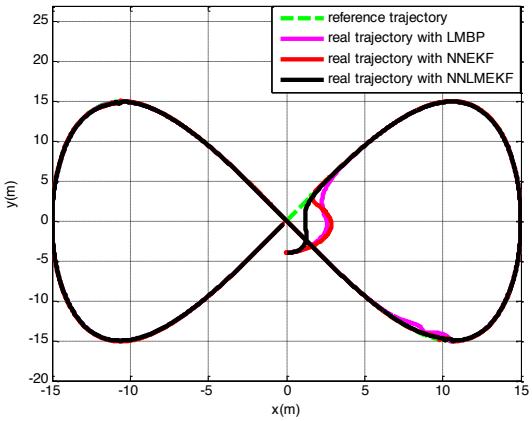

Fig. 7 Tracking response of a mobile robot with NN controllers for lemniscate trajectory $X_0 = [0; -4; \pi/4]$

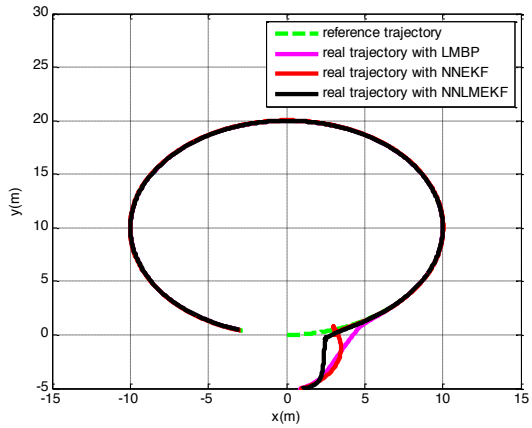

Fig. 8 Tracking response of a mobile robot with NN controllers for Circular trajectory $X_0 = [0; -5; 0]$

These results show that the robot has tracked effectively any reference trajectory. The comparative results of each method of training the neural network are shown in table 3, this table represents the RMS errors in x, y and theta for each trajectory and for each training method, the important remark is that the robot controlled using neural network trained with LM-IEKF has tracked the reference trajectory with the minimum RMSE than neural network control trained with EKF or LMBP.

Table. 3 RMS errors between reference and real trajectory

trajectory	RMS error	Methods		
		EKF	LM-IEKF	LMBP
Fig .6	RMSE in x	0.0755	0.0640	0.2469
	RMSE in y	0.3585	0.3567	0.3941
	RMSE in θ	0.2506	0.2491	0.2515
Fig .7	RMSE in x	0.1971	0.1523	0.2467
	RMSE in y	0.2924	0.2849	0.5705
	RMSE in θ	0.6574	0.5391	0.7866
Fig .8	RMSE in x	0.3791	0.2939	0.3944
	RMSE in y	0.3940	0.3670	0.6616
	RMSE in θ	0.3153	0.2136	0.2223

All simulation results show clearly that the LM-IEKF training algorithm outperforms the other algorithms in estimating optimal weights and biases of neural network controllers.

6. Experimental results

In this section, we use experimental results to compare the proposed Levenberg-Marquardt-IEKF method with the EKF method for neural network training. A practical photograph of the experimental equipment for the differential driving mobile robot system is depicted in Figure. 11. All the experimental results are carried out via LabVIEW 2013 in an i3 core personal computer (PC). The vehicle used in the experiments is the starter kit 2.0 mobile robot, manufactured by the National Instrumentation.

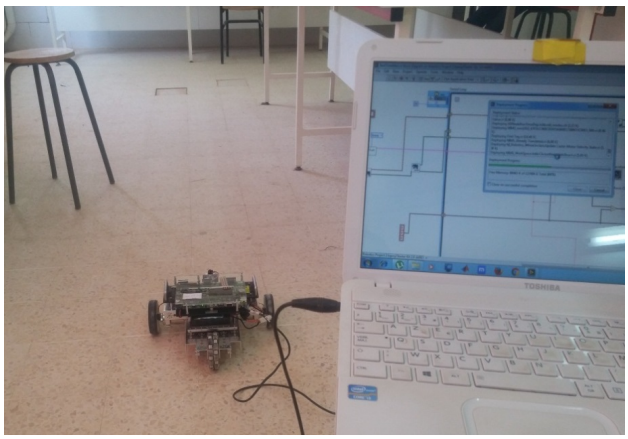

Fig. 11. NI starter kit 2.0 mobile robot

Figure11 shows the actual mobile robot following the desired path where the initial pose for the NI mobile robot starts at position (0, 0) meter and orientation π/2 radian and the desired path starts at position (1,1, π/2).

Figure12 shows that the trajectory of the real NI mobile robot controlled by neural networks trained with the LM-IEKF algorithm is more close to the reference trajectory than the trajectory obtained when the neural networks are trained with EKF algorithm.

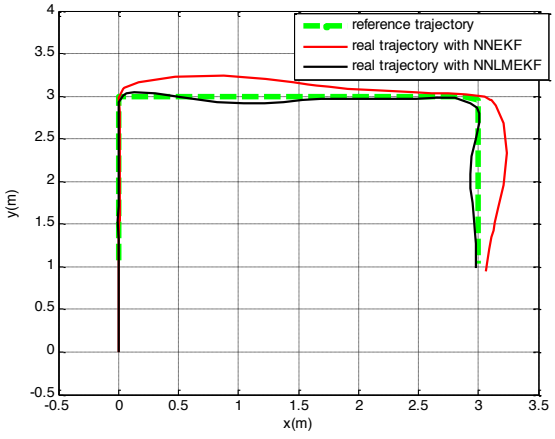

Fig. 12. Tracking response of NI mobile robot with an NN controller

Table 4 represent the experimental RMSE of coordinates (*x,y*) and the orientation. This result shows that the neural network trained with LM-IEKF has better performance to estimate the parameters of the neural network than the extended Kalman filter.

Table 4 Experimental RMS errors

method	RMSE in x	RMSE in y	RMSE in theta
EKF	0.2463	0.3912	0.5554
LM-IEKF	0.1828	0.2457	0.4371

7. Conclusion

In this study, we design a neural network feedback controller for unicycle-type nonholonomic mobile robots. The proposed controller consists of two neural networks, each one has two inputs (position errors of wheels) and one output corresponding to each velocity. They are trained off-line with a standard extended Kalman filter and Levenberg-Marquardt Iterated-EKF. We found that a neural network trained with the proposed Levenberg-Marquardt-Iterated EKF shows better results than the NN trained with the EKF algorithm. Simulation and experimental results demonstrate the efficiency and the effectiveness of neural networks controllers trained with the Marquardt-Iterated-EKF algorithm for the mobile robot trajectory tracking problem.

References

[1] Z. Hendzel and M. Trojnacki, "Neural Network Control of a Four-Wheeled Mobile Robot Subject to Wheel Slip," *Adv. Intell. Syst. Comput.*, vol. 317, pp. 473–482, 2015.

[2] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *Control Syst. Technol. IEEE Trans.*, vol. 10, no. 6, pp. 835–852, 2002.

[3] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: backstepping kinematics into dynamics," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, 1995, vol. 4, pp. 3805–3810.

[4] D. Chwa, "Tracking Control of Differential-Drive Wheeled Mobile Robots Using a Backstepping-Like Feedback Linearization," *Syst. Man Cybern. Part A Syst. Humans, IEEE Trans.*, vol. 40, no. 6, pp. 1285–1295, 2010.

[5] C. Samson, "Control of Chained Systems Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots," *IEEE Trans. Automat. Contr.*, vol. 40, no. 1, pp. 64–77, 1995.

[6] J.-M. Yang and J.-H. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *Robot. Autom. IEEE Trans.*, vol. 15, no. 3, pp. 578–587, 1999.

[7] D. Chwa, "Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates," *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 4, pp. 637–644, 2004.

[8] M. Peña-cabrera, H. Gómez, and V. Lomas, "Fuzzy Logic for omnidirectional mobile platform control based in FPGA and Bluetooth communication," vol. 13, no. 6, pp. 1907–1914, 2014.

[9] T. Dierks and S. Jagannathan, "Asymptotic Adaptive Neural Network Tracking Control of Nonholonomic Mobile Robot Formations," *J. Intell. Robot. Syst.*, vol. 56, no. 1–2, pp. 153–176, 2009.

[10] J. Ye, "Tracking control for nonholonomic mobile robots: Integrating the analog neural network into the backstepping technique," *Neurocomputing*, vol. 71, no. 16–18, pp. 3373–3378, 2008.

[11] L. C. Jain, M. Seera, C. P. Lim, and P. Balasubramaniam, "A review of online learning in supervised neural networks," *Neural Comput. Appl.*, vol. 25, no. 3–4, pp. 491–509, 2014.

[12] A. N. Chernodub, "Training Neural Networks for Classification Using the Extended Kalman Filter : A Comparative Study," vol. 23, no. 2, pp. 96–103, 2014.

[13] A. Saptoro, "Extended and unscented kalman filters for artificial neural network modelling of a nonlinear dynamical system," *Theor. Found. Chem. Eng.*, vol. 46, no. 3, pp. 274–278, 2012.

[14] F. Heimes, "Extended Kalman Filter Neural Network Training: Experimental Results and Algorithm Improvements," pp. 1639–1644, 1998.

[15] D. B. and E. W. S. Gannot, "Iterative and Sequential Kalman Filter-Based," *IEEE Trans. Speech Audio Process.*, vol. 6, no. 4, pp. 373–385, 1998.

[16] P. Deossa, J. Pati??o, J. Espinosa, and F. Valencia, "A comparison of Extended Kalman Filter and Levenberg-Marquardt methods for neural network training," *2011 IEEE 9th Lat. Am. Robot. Symp. IEEE Colomb. Conf. Autom. Control. LARC 2011 - Conf. Proc.*, 2011.

[17] Z. V. P. Murthy, "Nonlinear Regression: Levenberg- Marquardt Method," pp. 4–6, 2014.

[18] R. Toushmalani, Z. Parsa, and A. Esmaeili, "Comparison result of inversion of gravity data of a fault by Cuckoo Optimization and Levenberg-Marquardt methods," *Res. J. Pharm. Biol. Chem. Sci.*, vol. 5, no. 1, pp. 418–427, 2014.

[19] M. A. Skoglund, G. Hendeby, and D. Axehill, "Extended Kalman filter modifications based on an optimization view point," in *Information Fusion (Fusion), 2015 18th International Conference on*, 2015, pp. 1856–1861.

[20] J. J. Kappl, "Estimation Kalman Filtering," *Ieee Trans. Aerosp. Electron. Syst.*, no. 1, pp. 79–84, 1971.

[21] X. Wang and Y. Huang, "Convergence study in extended Kalman filter-based training of recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 22, no. 4, pp. 588–600, 2011.

[22] K.-W. Wong, C.-S. Leung, and S.-J. Chang, "Use of periodic and monotonic activation functions in multilayer feedforward neural networks trained by extended Kalman filter algorithm," *IEE Proc. - Vision, Image, Signal Process.*, vol. 149, no. 4, p. 217, 2002.

[23] B. M. Bell and F. W. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *IEEE Trans. Automat. Contr.*, vol. 38, no. 2, pp. 1991–1994, 1993.

[24] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Q. Appl. Math.*, vol. 2, pp. 164–168, 1944.

[25] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2. pp. 431–441, 1963.