MULTILAYER PERCEPTRON FUNCTIONAL ADAPTIVE CONTROL FOR TRAJECTORY TRACKING OF WHEELED MOBILE ROBOTS

Marvin K. Bugeja† and Simon G. Fabri‡
Department of Electrical Power and Control Engineering, University of Malta
Msida MSD06, Malta

Keywords: Wheeled mobile robots, trajectory tracking, adaptive control, neural networks, stochastic estimation.

Abstract: Sigmoidal multilayer perceptron neural networks are proposed to effect functional adaptive control for handling the trajectory tracking problem in a nonholonomic wheeled mobile robot. The scheme is developed in discrete time and the multilayer perceptron neural networks are used for the estimation of the robot's nonlinear kinematic functions, which are assumed to be unknown. On-line weight tuning is achieved by employing the extended Kalman filter algorithm based on a specifically formulated multiple-input, multiple-output, stochastic model for the trajectory error dynamics of the mobile base. The estimated functions are then used on a certainty equivalence basis in the control law proposed in (Corradini et al., 2003) for trajectory tracking. The performance of the system is analyzed and compared by simulation.

1 INTRODUCTION

The last fifteen years have witnessed extensive research activity on the use of artificial neural networks for adaptive estimation and control of nonlinear systems. Since the classical pioneering paper by Narendra and Parthasarathy (Narendra and Parthasarathy, 1990), numerous neural network-based algorithms and stability proofs have been published, covering different types of neural networks and various system conditions (Fabri and Kadirkamanathan, 2001; Lewis et al., 1999; Ge et al., 1999).

Robotic systems, being inherently nonlinear and subject to a relatively high degree of uncertainty, represent an important area where neural networks hold much promise for effecting adaptive control. In fact, artificial neural networks have been used extensively for the nonlinear estimation problem in motion control of mobile robots (Fierro and Lewis, 1997; Fierro and Lewis, 1998; Corradini et al., 2003). Mobile robots are non-linear, multiple-input, multiple-output (MIMO) systems. In practical environments they exhibit a high degree of uncertainty in the model parameters and are subject to unmeasurable external disturbances. These characteristics necessitate the use of robust adaptive controllers, possibly incorporating on-line estimation of the nonlinear functions in the process model.

In particular, the trajectory tracking problem for nonholonomic wheeled mobile robots has become one of the main challenges within the automatic control research community over the last decade (Fierro and Lewis, 1995; Fierro and Lewis, 1998; Corradini and Orlando, 2001; Asensio and Montano, 2002; Oriolo et al., 2002; Corradini et al., 2003; Grech and Fabri, 2004). This research activity is not only justified by the theoretical challenges posed by this particular task, but also by a vast array of potential practical applications in the field of mobile robotics (Corradini and Orlando, 2001; Ding and Cooper, 2005).

However, algorithms employing neural networks with an on-line weight tuning facility, underpinning the adaptation feature, are more sparse (Fierro and Lewis, 1997; Fierro and Lewis, 1998). Moreover, the robot models commonly encountered in related literature are considered to be deterministic and process noise and external disturbances are often ignored.

In order to address these issues, this paper proposes the use of on-line function estimation, taking into account process noise and external disturbances, by using a specifically formulated MIMO, stochastic model for the trajectory error dynamics of a mobile robot. A multilayer percpetron (MLP) neural network is used to learn these nonlinear dynamics. The proposed stochastic estimation technique used for parameter adjustment of the MLP network is an extension of Fabri

K. Bugeja M. and G. Fabri S. (2005).
MULTILAYER PERCEPTRON FUNCTIONAL ADAPTIVE CONTROL FOR TRAJECTORY TRACKING OF WHEELED MOBILE ROBOTS.
In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics - Robotics and Automation*, pages 66-72
DOI: 10.5220/0001182200660072

and Kadirkamanathan's (Fabri and Kadirkamanathan, 1998) method for a general nonlinear class of single-input, single-output (SISO) systems, to underactuated MIMO robotic systems. The estimated functions are used in the trajectory tracking control law in (Corradini et al., 2003), to make the robot follow a given reference trajectory.

The paper is organized as follows. The stochastic discrete-time error dynamic model for the robot is developed in Section 2. This is then utilized in developing an on-line, stochastic estimator based on MLP neural networks in Section 3. The control law in (Corradini et al., 2003) is revisited and incorporated with the proposed recursive weight tuning algorithm, based on extended Kalman filtering in Section 4. Section 5 presents simulation results and is followed by a conclusion in Section 6.

2 PROBLEM STATEMENT

This paper considers a unicycle wheeled mobile robot. The robot state is expressed as a three dimensional vector $\mathbf{q}$, known as *the pose*, such that $\mathbf{q} = [x \;\; y \;\; \theta]^T$ where (x, y) is the axle midpoint coordinate and θ is the robot orientation referred to a fixed frame (Corradini et al., 2003). Similarly, a given reference trajectory can be described by a reference derivative vector $\dot{\mathbf{q}}_r$, composed of the reference pose velocities $\dot{x}_r$, $\dot{y}_r$ and $\dot{\theta}_r$ such that

$$\dot{\mathbf{q}}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v_r \cos(\theta_r) \\ v_r \sin(\theta_r) \\ \omega_r \end{bmatrix}, \qquad (1)$$

where v_r and ω_r are the linear and angular reference velocities respectively. Commonly, the tracking error vector $\mathbf{e} = [e_1 \;\; e_2 \;\; e_3]^T$ is defined as

$$\mathbf{e} := \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}. \qquad (2)$$

The aim is to make $\mathbf{e}$ converge to zero so that $\mathbf{q}$ converges to $\mathbf{q}_r$. The resulting tracking error equation modelled in continuous time is given by

$$\dot{\mathbf{e}} = \begin{bmatrix} v_r \cos(e_3) \\ v_r \sin(e_3) \\ \omega_r \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \mathbf{u}, \qquad (3)$$

where $\mathbf{u}$ is the velocity control input vector, given by $\mathbf{u} = [v \;\; \omega]^T$, where v and ω are the linear and angular input velocities of the robot, respectively. These are related to the velocities of the robot wheels.

Discretizing the error dynamics in (3) by using a first order explicit forward Euler approximation with sampling interval T seconds and assuming that the control input vector $\mathbf{u}$ remains constant over each sampling interval, the following discrete time dynamic model for the trajectory error vector is obtained:

$$\mathbf{e}_k = \mathbf{f}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1}, \qquad (4)$$

where the subscript integer k denotes that the corresponding variable is evaluated at time instant kT seconds, and assuming that the sampling interval is chosen low enough for the Euler approximation to hold. Vector $\mathbf{f}_{k-1}$ and matrix $\mathbf{G}_{k-1}$ are defined as:

$$\mathbf{f}_{k-1} := \begin{bmatrix} e_{1_{k-1}} + T v_{r_{k-1}} \cos(e_{3_{k-1}}) \\ e_{2_{k-1}} + T v_{r_{k-1}} \sin(e_{3_{k-1}}) \\ e_{3_{k-1}} + T \omega_{r_{k-1}} \end{bmatrix} \qquad (5)$$

$$\mathbf{G}_{k-1} := \begin{bmatrix} -T & T e_{2_{k-1}} \\ 0 & -T e_{1_{k-1}} \\ 0 & -T \end{bmatrix}. \qquad (6)$$

Notation-wise we will occasionally drop the time index subscript where this is obvious. Introducing a vector of discrete random variables $\boldsymbol{\varepsilon}_k$, the error dynamic model in (4) is converted to the following nonlinear, stochastic, MIMO, error dynamic model:

$$\mathbf{e}_k = \mathbf{f}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \boldsymbol{\varepsilon}_k, \qquad (7)$$

where $\boldsymbol{\varepsilon}_k$ is assumed to be an independent, zero-mean, white, Gaussian-distributed process, with covariance matrix $\mathbf{R}$. This stochastic term represents the process noise and external disturbances.

The error models in (3) and (7) are based only on the kinematic model of the mobile base. Control schemes based on a full dynamic model would capture better the behaviour of the robot (Fierro and Lewis, 1997; Corradini et al., 2003) because these take into account dynamic terms such as mass, viscous damping and inertia. However, such controllers will have more complex control laws when compared with those based upon kinematic modelling only. Moreover this complexity is increased when considering that in practice, the values of the dynamic process parameters are uncertain or unknown and may even be time varying. Such complexity might hinder the practical implementation of these dynamic control schemes. However, the experimental results presented in (Corradini et al., 2003) indicate that by combining neural networks for estimation of uncertainty in the *kinematic* model and a kinematic controller, one gets superior performance in comparison to the stand-alone kinematic control case. The estimation of uncertainty introduced in the simpler kinematic model will compensate somewhat for the ignored dynamic parameters, that would have otherwise been included in a full dynamic model control scheme with all its associated complexity.

The work proposed in this paper supports the same philosophy, but differs from the work presented in (Corradini et al., 2003) in two main aspects. Primarily, the estimation process developed in this paper is on-line, enriching the controller with the adaptive feature. On-line weight tuning algorithms do not require any preliminary off-line training (Fierro and Lewis, 1998). Consequently, on-line estimators are superior in handling real-time applications with possible unforeseen variations in the process parameters; a typical scenario in mobile robotics. Secondly, we consider process noise and external disturbances in our model. These disturbances are modelled as random processes, as shown in (7). In this manner the resulting control scheme is more robust in the presence of varying process parameters, noise and external disturbances.

3 SIGMOIDAL MLP ESTIMATOR

Theoretically speaking, the vector of discrete functions $\mathbf{f}_{k-1}$ and the matrix of discrete functions $\mathbf{G}_{k-1}$, composing the trajectory error model in (7) are both known, assuming that the pose vector $\mathbf{q}$ is available at each sampling interval. However, as proposed in Section 2, neural networks will be employed for the estimation of these functions recursively, as if they were totally unknown. This approach is justified since in a real robot these functions depend on the robot's unmodelled dynamics, time-varying parameters and unmeasurable disturbances.

In this paper two sigmoidal MLP networks, denoted by $\mathbf{NN_f}$ and $\mathbf{NN_G}$, for the estimation of $\mathbf{f}_{k-1}$ and $\mathbf{G}_{k-1}$ respectively, are employed. Unlike Gaussian radial basis function (RBF) neural networks (Poggio and Girosi, 1990; Haykin, 1999), which are also popular in the field of neuro-adaptive control, MLPs do not posses the possible advantage of linearity in the parameters being estimated. For this reason tuning algorithms for MLPs tend to be more complex and usually result in sub-optimal estimation. On the other hand, unlike the Gaussian basis functions in RBFs, the sigmoidal activation functions used in MLPs are not localized, implying that typically MLP networks require less neurons than RBF networks for the same degree of accuracy. This in turn means that MLP weight tuning algorithms need to handle relatively less parameters, reducing computational intensity, which ultimately speeds up the overall control loop. Naturally this is imperative in real time digital control systems. These details become more pronounced when dealing with high dimensional order systems, as in the case of WMRs, where the number of neurons increases exponentially with the number of systems states. This is often referred to as *the curse of dimensionality* (Bellman, 1961).

Since the MLP parameters do not appear linearly in the output equation, as will become clearer later in the section, nonlinear stochastic estimation techniques have to be employed. In this paper the extended Kalman filter (EKF) (Maybeck, 1979), is used for the recursive estimation of the MLP networks parameters. This, along with a set of mathematical operations, is the heart of the stochastic online weight tuning algorithm proposed next in the following paragraphs.

Consider the following:

- $\mathbf{x}_{\mathbf{f}_{k-1}}$ and $\mathbf{x}_{\mathbf{G}_{k-1}}$ denote the two neural network input vectors of $\mathbf{NN_f}$ and $\mathbf{NN_G}$ respectively. A constant, serving as a neuron bias input, is included in each of these two input vectors complete defined as

$$\mathbf{x}_{\mathbf{f}_{k-1}} := \begin{bmatrix} \mathbf{e}_{k-1}^T & v_{r_{k-1}} & \omega_{r_{k-1}} & 1 \end{bmatrix}^T \tag{8}$$

$$\mathbf{x}_{\mathbf{G}_{k-1}} := \begin{bmatrix} e_{1_{k-1}} & e_{2_{k-1}} & 1 \end{bmatrix}^T . \tag{9}$$

- $\boldsymbol{\phi}_\mathbf{f}$ and $\boldsymbol{\phi}_\mathbf{G}$ are the sigmoidal activation function vectors, representing the outputs of the corresponding hidden layer, whose ith element is given by

$$\phi_{\mathbf{f}_i} = 1/\left\{1 + \exp(-\mathbf{s}_{\mathbf{f}_i}^T \mathbf{x}_\mathbf{f})\right\} \tag{10}$$

$$\phi_{\mathbf{G}_i} = 1/\left\{1 + \exp(-\mathbf{s}_{\mathbf{G}_i}^T \mathbf{x}_\mathbf{G})\right\}, \tag{11}$$

  where $\mathbf{s}_{\mathbf{f}_i}$ and $\mathbf{s}_{\mathbf{G}_i}$ are the parameter vectors of the ith neuron in the hidden layer, characterizing the shape of that particular sigmoidal activation function. These sigmoidal parameter vectors form part of the overall weight vector to be estimated, meaning that each activation function is shaped recursively as part of the tuning process. This detail is responsible for the undesired nonlinear appearance of the final weight vector in the output equation.

- $L_\mathbf{f}$ and $L_\mathbf{G}$ denote the number of sigmoidal activation functions in $\mathbf{NN_f}$ and $\mathbf{NN_G}$ respectively.

- Let the activation function parameter vectors for $\mathbf{NN_f}$ and $\mathbf{NN_G}$ be grouped in two individual vectors $\hat{\mathbf{a}}_\mathbf{f}$ and $\hat{\mathbf{a}}_\mathbf{G}$ respectively, such that

$$\hat{\mathbf{a}}_\mathbf{f} := \begin{bmatrix} \hat{\mathbf{s}}_{\mathbf{f}_1}^T & \cdots & \hat{\mathbf{s}}_{\mathbf{f}_{L_\mathbf{f}}}^T \end{bmatrix}^T \tag{12}$$

$$\hat{\mathbf{a}}_\mathbf{G} := \begin{bmatrix} \hat{\mathbf{s}}_{\mathbf{G}_1}^T & \cdots & \hat{\mathbf{s}}_{\mathbf{G}_{L_\mathbf{G}}}^T \end{bmatrix}^T , \tag{13}$$

  where the symbol $\hat{}$ indicates that the particular parameter vector is undergoing estimation.

- The employed multilayer feedforward neural network structure yields the following input-output relations for the two neural networks:

$$\hat{\mathbf{f}}_{k-1} = \begin{bmatrix} \hat{\mathbf{w}}_{\mathbf{1}_k}^T \boldsymbol{\phi}_{\mathbf{f}_{k-1}} \\ \hat{\mathbf{w}}_{\mathbf{2}_k}^T \boldsymbol{\phi}_{\mathbf{f}_{k-1}} \\ \hat{\mathbf{w}}_{\mathbf{3}_k}^T \boldsymbol{\phi}_{\mathbf{f}_{k-1}} \end{bmatrix}, \tag{14}$$

$$\hat{\mathbf{G}}_{k-1} = \begin{bmatrix} \hat{\mathbf{w}}_{\mathbf{11}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} & \hat{\mathbf{w}}_{\mathbf{12}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} \\ \hat{\mathbf{w}}_{\mathbf{21}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} & \hat{\mathbf{w}}_{\mathbf{22}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} \\ \hat{\mathbf{w}}_{\mathbf{31}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} & \hat{\mathbf{w}}_{\mathbf{32}_k}^T \boldsymbol{\phi}_{\mathbf{G}_{k-1}} \end{bmatrix}, \quad (15)$$

where $\hat{\mathbf{f}}_{k-1}$ and $\hat{\mathbf{G}}_{k-1}$ denote the output approximates of $\mathbf{NN_f}$ and $\mathbf{NN_G}$ respectively. Moreover, $\hat{\mathbf{w}}_{\mathbf{i}_k}$ represents the weight vector of the connection between the sigmoidal activation functions and the ith output element of $\mathbf{NN_f}$ and similarly $\hat{\mathbf{w}}_{\mathbf{ij}_k}$ represents the weight vector of the connection between the sigmoidal activation functions and the output element of $\mathbf{NN}_G$ that corresponds to the (ij)th term of $\hat{\mathbf{G}}_{k-1}$. Notation-wise $\boldsymbol{\phi}_{\mathbf{f}_{k-1}}$ implies that the activation function is evaluated for $\mathbf{x}_{\mathbf{f}_{k-1}}$ and $\hat{\mathbf{a}}_{\mathbf{f}_k}$. The same notation applies for $\boldsymbol{\phi}_{\mathbf{G}_{k-1}}$.

- Let us group all the parameters requiring estimation in two vectors $\hat{\mathbf{v}}_{\mathbf{f}_k}$ and $\hat{\mathbf{v}}_{\mathbf{G}_k}$, corresponding to $\mathbf{NN_f}$ and $\mathbf{NN_G}$ respectively, such that

$$\hat{\mathbf{v}}_{\mathbf{f}_k} := \begin{bmatrix} \hat{\mathbf{w}}_{\mathbf{1}_k}^T & \hat{\mathbf{w}}_{\mathbf{2}_k}^T & \hat{\mathbf{w}}_{\mathbf{3}_k}^T & \hat{\mathbf{a}}_{\mathbf{f}_k}^T \end{bmatrix}^T \quad (16)$$

$$\hat{\mathbf{v}}_{\mathbf{G}_k} := \begin{bmatrix} \hat{\mathbf{w}}_{\mathbf{11}_k}^T \hat{\mathbf{w}}_{\mathbf{12}_k}^T \cdots \hat{\mathbf{w}}_{\mathbf{31}_k}^T \hat{\mathbf{w}}_{\mathbf{32}_k}^T \hat{\mathbf{a}}_{\mathbf{G}_k}^T \end{bmatrix}^T. \quad (17)$$

- Let $\mathbf{z}_k$ be the complete overall weight vector defined as

$$\hat{\mathbf{z}}_k := \begin{bmatrix} \hat{\mathbf{v}}_{\mathbf{f}_k}^T \vdots \hat{\mathbf{v}}_{\mathbf{G}_k}^T \end{bmatrix}^T. \quad (18)$$

- Differentiating $\hat{\mathbf{f}}_{k-1}$ and $\hat{\mathbf{G}}_{k-1}\mathbf{u}_{k-1}$ with respect to $\hat{\mathbf{z}}_\mathbf{k}$, yields the following closed form expressions:

$$\boldsymbol{\nabla}_{\mathbf{f}_k} := \frac{\partial(\hat{\mathbf{f}}_{k-1})}{\partial(\hat{\mathbf{z}}_\mathbf{k})} = \begin{bmatrix} \boldsymbol{\nabla}_{\mathbf{f1}_k} \vdots \boldsymbol{\nabla}_{\mathbf{f2}_k} \end{bmatrix} \quad (19)$$

$$\boldsymbol{\nabla}_{\boldsymbol{\Gamma}_k} := \frac{\partial(\hat{\mathbf{G}}_{k-1}\mathbf{u}_{k-1})}{\partial(\hat{\mathbf{z}}_\mathbf{k})} = \begin{bmatrix} \boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{1}_k} \vdots \boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{2}_k} \end{bmatrix} \quad (20)$$

where $\boldsymbol{\nabla}_{\mathbf{f1}_k}$,$\boldsymbol{\nabla}_{\mathbf{f2}_k}$,$\boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{1}_k}$ and $\boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{2}_k}$ are defined as follows:

$$\boldsymbol{\nabla}_{\mathbf{f1}_k} := \begin{bmatrix} \boldsymbol{\phi}_{\mathbf{f}_{k-1}}^T & \mathbf{0}_\mathbf{f}^T & \mathbf{0}_\mathbf{f}^T \\ \mathbf{0}_\mathbf{f}^T & \boldsymbol{\phi}_{\mathbf{f}_{k-1}}^T & \mathbf{0}_\mathbf{f}^T \\ \mathbf{0}_\mathbf{f}^T & \mathbf{0}_\mathbf{f}^T & \boldsymbol{\phi}_{\mathbf{f}_{k-1}}^T \end{bmatrix}, \quad (21)$$

where $\mathbf{0_f}$ denotes a zero vector having the same length as $\boldsymbol{\phi}_{\mathbf{f}_{k-1}}$.

$$\boldsymbol{\nabla}_{\mathbf{f2}_k} := \begin{bmatrix} \cdots \hat{w}_{1,i}(\phi_{\mathbf{f}_i} - {\phi_{\mathbf{f}_i}}^2)\mathbf{x}_\mathbf{f}^T \cdots \\ \cdots \hat{w}_{2,i}(\phi_{\mathbf{f}_i} - {\phi_{\mathbf{f}_i}}^2)\mathbf{x}_\mathbf{f}^T \cdots \\ \cdots \hat{w}_{3,i}(\phi_{\mathbf{f}_i} - {\phi_{\mathbf{f}_i}}^2)\mathbf{x}_\mathbf{f}^T \cdots \end{bmatrix}, \quad (22)$$

where $i = 1, \ \ldots \ , L_\mathbf{f}$ and $\hat{w}_{j,i}$ denotes the ith element of the jth output weight vector, $\hat{\mathbf{w}}_{\mathbf{j}_k}$. Note also that in this equation, both $\phi_{\mathbf{f}_i}$ and $\mathbf{x_f}$ correspond to time instant $(k-1)$.

$$\boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{1}_k} := \begin{bmatrix} \gamma_v & \gamma_\omega & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T \\ \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \gamma_v & \gamma_\omega & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T \\ \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \mathbf{0}_\mathbf{G}^T & \gamma_v & \gamma_\omega \end{bmatrix}, \quad (23)$$

where γ_v and γ_ω are defined as $\boldsymbol{\phi}_{\mathbf{G}_{k-1}}^T v_{k-1}$ and $\boldsymbol{\phi}_{\mathbf{G}_{k-1}}^T \omega_{k-1}$ respectively and $\mathbf{0_G}$ denotes a zero vector having the same length as $\boldsymbol{\phi}_{\mathbf{G}_{k-1}}$.

$$\boldsymbol{\nabla}_{\boldsymbol{\Gamma}\mathbf{2}_k} := \begin{bmatrix} \cdots \sigma_{1,i}(\phi_{\mathbf{G}_i} - {\phi_{\mathbf{G}_i}}^2)\mathbf{x}_\mathbf{G}^T \cdots \\ \cdots \sigma_{2,i}(\phi_{\mathbf{G}_i} - {\phi_{\mathbf{G}_i}}^2)\mathbf{x}_\mathbf{G}^T \cdots \\ \cdots \sigma_{3,i}(\phi_{\mathbf{G}_i} - {\phi_{\mathbf{G}_i}}^2)\mathbf{x}_\mathbf{G}^T \cdots \end{bmatrix} \quad (24)$$

where $i = 1, \ \ldots \ , L_\mathbf{G}$ and $\sigma_{j,i}$ is defined as $\hat{w}_{j1,i}v_{k-1} + \hat{w}_{j2,i}\omega_{k-1}$ where $\hat{w}_{jn,i}$ denotes the ith element of the (jn)th output weight vector, $\hat{\mathbf{w}}_{\mathbf{jn}_k}$. Note also that in this equation, both $\phi_{\mathbf{G}_i}$ and $\mathbf{x_G}$ correspond to time instant $(k-1)$.

Assuming that, within the space of interest, the neural network approximation errors are negligibly small when the complete weight vector $\hat{\mathbf{z}}_k$ is equal to some unknown optimal weight vector $\mathbf{z}_k^*$. This is justified in the light of the *Universal Approximation Theorem* of neural networks (Haykin, 1999). It follows that the trajectory error model in (7) can be rewritten, using the formulated neural networks as approximations for $\mathbf{f}_{k-1}$ and $\mathbf{G}_{k-1}$. The resulting model is given by

$$\mathbf{e}_k = \mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{z}_k^*\right) + \varepsilon_k, \quad (25)$$

where

$$\mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{z}_k^*\right) := \hat{\mathbf{f}}_{k-1}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{v}_{\mathbf{f}_k}^*\right) + \hat{\mathbf{G}}_{k-1}\left(\mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{v}_{\mathbf{G}_k}^*\right)\mathbf{u}_{k-1}, \quad (26)$$

with $\mathbf{v}_{\mathbf{f}_k}^*$ and $\mathbf{v}_{\mathbf{G}_k}^*$ representing the optimal versions of $\hat{\mathbf{v}}_{\mathbf{f}_k}$ and $\hat{\mathbf{v}}_{\mathbf{G}_k}$ respectively. Finally, let $\boldsymbol{\nabla}_{\mathbf{h}_k}$ denote the Jacobian matrix of $\mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{z}_k^*\right)$ with respect to the weight vector $\mathbf{z}_k^*$ evaluated at $\mathbf{z}_k^* = \hat{\mathbf{z}}_k$, then it is clear that

$$\begin{aligned} \boldsymbol{\nabla}_{\mathbf{h}_k} &:= \left.\frac{\partial\left(\mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{z}_k^*\right)\right)}{\partial(\mathbf{z}_k^*)}\right|_{\mathbf{z}_k^* = \hat{\mathbf{z}}_k} \\ &= \begin{bmatrix} \boldsymbol{\nabla}_{\mathbf{f}_k} \vdots \boldsymbol{\nabla}_{\boldsymbol{\Gamma}_k} \end{bmatrix}, \end{aligned} \quad (27)$$

where $\boldsymbol{\nabla}_{\mathbf{f}_k}$ and $\boldsymbol{\nabla}_{\boldsymbol{\Gamma}_k}$ are defined in (19) and (20) respectively. Equation (26) indicates that the unknown weight vector $\mathbf{z}_k^*$, does not appear linearly in the output equation in (25). For this reason a nonlinear stochastic estimator is employed in the recursive weight tuning process detailed in the next section.

4 ADAPTIVE CONTROL SCHEME

Having derived the stochastic model (25) for the trajectory error dynamics that is dependent on the MLPs

through $\hat{\mathbf{f}}_{k-1}$, $\hat{\mathbf{G}}_{k-1}$ and the optimal weight vector $\mathbf{z}_k^*$, it is straightforward to write down the following nonlinear state-space equations which are used next within the EKF algorithm (predictive mode) to enable the recursive estimation of the neural networks weights:

$$\begin{aligned} \mathbf{z}_{k+1}^* &= \mathbf{z}_k^* \\ \mathbf{e}_k &= \mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{z}_k^*\right) + \varepsilon_k, \end{aligned} \quad (28)$$

It is assumed that:

- The initial parameter vector $\mathbf{z}_0^*$ has a Gaussian distribution with some mean $\bar{\mathbf{z}}$ and covariance $\mathbf{V}$.
- The process noise vector ε_k is Gaussian, zero-mean and uncorrelated in time (i.e. white).
- ε_k and $\mathbf{z}_0^*$ are independent, implying that they are also uncorrelated.
- The conditional density of the parameter vector $\mathbf{z}_{k+1}^*$ is approximately Gaussian. However, it should be emphasized that the latter is only an approximation introduced by the EKF, since the nonlinearity of the stochastic model will not conserve the Gaussian nature in $\mathbf{z}_{k+1}^*$.

From EKF theory (Maybeck, 1979), it follows that the following recursive weight adjustment rules can be employed for the estimation of the weight vector:

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{K}_k \boldsymbol{\nabla}_{\mathbf{h}_k} \mathbf{P}_k \quad (29)$$

$$\hat{\mathbf{z}}_{k+1} = \hat{\mathbf{z}}_k + \mathbf{K}_k \mathbf{i}_k, \quad (30)$$

where $\mathbf{K}_k$ is the Kalman gain matrix given by

$$\mathbf{K}_k = \mathbf{P}_k \boldsymbol{\nabla}_{\mathbf{h}_k}^T \left[\boldsymbol{\nabla}_{\mathbf{h}_k} \mathbf{P}_k \boldsymbol{\nabla}_{\mathbf{h}_k}^T + \mathbf{R}\right]^{-1} \quad (31)$$

and $\mathbf{i}_k$ is the innovations vector given by

$$\mathbf{i}_k = \mathbf{e}_k - \mathbf{h}\left(\mathbf{x}_{\mathbf{f}_{k-1}}, \mathbf{x}_{\mathbf{G}_{k-1}}, \mathbf{u}_{k-1}, \hat{\mathbf{z}}_k\right). \quad (32)$$

$\mathbf{P}_{k+1}$ denotes the prediction covariance matrix and represents a measure of the accuracy of the estimate $\hat{\mathbf{z}}_{k+1}$. The initial conditions for the weight vector and the prediction covariance are set to some desired values $\bar{\mathbf{z}}$ and $\mathbf{V}$ respectively.

For each estimate of $\hat{\mathbf{z}}_k$, the corresponding estimates $\hat{\mathbf{f}}_{k-1}$ and $\hat{\mathbf{G}}_{k-1}$ are computed using the formulated relations given by (8) to (18). These estimates are then used recursively in the control law adopted from (Corradini et al., 2003), making the overall control scheme adaptive, since the process model used for control is now being estimated in real-time and requires no preliminary off-line training.

Note that the extra computational burden that comes along with the EKF, in comparison to other non-stochastic techniques such as back-propagation, pays off in several ways. Primarily, it renders the recursive weight tuning algorithm stochastic, since the uncertainty in the process model is taken into account by the EKF algorithm. Secondly, the conditional probability density of the error vector $\mathbf{e}_k$ is being updated in real time. The resulting information, most importantly the covariance matrix $\mathbf{P}_k$, is essential in the design of stochastic control laws (Fabri and Kadirkamanathan, 2001), which are part of our plan for future work. On the other hand, it is widely known that the assumption of local linearity inherent in the EKF, may lead to convergence problems in highly nonlinear applications. For this reason future investigation of other nonlinear estimation techniques, such as the Unscented Kalman Filter (UKF) (Julier and Uhlmann, 1997) is desirable.

The discrete error model utilized in (Corradini et al., 2003) is effectively given by

$$\begin{aligned} \mathbf{e}_k &= \begin{bmatrix} e_{1_{k-1}} + h_2(e_{3_{k-1}}, v_{r_{k-1}}) \\ e_{2_{k-1}} + h_4(e_{3_{k-1}}, v_{r_{k-1}}) \\ e_{3_{k-1}} + h_5(\omega_{r_{k-1}}) \end{bmatrix} \\ &\quad + \begin{bmatrix} -T & h_1(e_{2_{k-1}}) \\ 0 & h_3(e_{1_{k-1}}) \\ 0 & -T \end{bmatrix} \mathbf{u}_{k-1}, \end{aligned} \quad (33)$$

where h_1 to h_5 represent nonlinear functions, appropriately defined in (Corradini et al., 2003). Comparing (33) with the stochastic model in (7) and ignoring the stochastic term, the following relations become evident:

$$\begin{aligned} h_1(e_{2_{k-1}}) &= g_{12_{k-1}} \\ h_2(e_{3_{k-1}}, v_{r_{k-1}}) &= f_{1_{k-1}} - e_{1_{k-1}} \\ h_5(\omega_{r_{k-1}}) &= f_{3_{k-1}} - e_{3_{k-1}}, \end{aligned} \quad (34)$$

where f_i and g_{ij} denote the ith and (ij)th element in $\mathbf{f}_{k-1}$ and $\mathbf{G}_{k-1}$ respectively. Hence the estimates $\hat{\mathbf{f}}_{k-1}$ and $\hat{\mathbf{G}}_{k-1}$ provided by the neural networks, can be used to estimate h_1, h_2 and h_5 recursively according to (34). These are then used in the control law proposed in (Corradini et al., 2003), given by

$$\begin{aligned} v_k &= \frac{1}{T}\{k_1 e_{1_k} + h_1(e_{2_k})\omega_k + h_2(e_{3_k}, v_{r_k})\} \\ \omega_k &= \frac{1}{T}\{h_5(\omega_{r_k}) + k_2 e_{2_k} + k_3 e_{3_k}\}. \end{aligned} \quad (35)$$

The design parameters k_1, k_2 and k_3 can be selected according to the following set of relations:

$$\begin{aligned} k_1 &= 1 - \lambda_1 \\ k_2 &= 2 - \lambda_2 - \lambda_3 \\ k_3 &= \frac{1}{T v_{r_k}}(1 - \lambda_2 - \lambda_3 + \lambda_2\lambda_3), \end{aligned} \quad (36)$$

with λ_1, λ_2 and λ_3 being the eigenvalues of the resulting closed loop error discrete dynamics, that must be placed inside the unity circle to guarantee local asymptotic stability as suggested in (Corradini et al., 2003).

5 SIMULATION RESULTS

The proposed on-line adaptive control scheme was verified and compared to the non-adaptive scheme proposed in (Corradini et al., 2003) by simulations. The mobile robot was simulated through a discrete-time model whose error dynamics were given in (7). The covariance $\mathbf{R}$ of the noise vector $\boldsymbol{\varepsilon}_k$ was set to $1 \times 10^{-6}\mathbf{I_3}$, where $\mathbf{I_i}$ denotes an $(i \times i)$ identity matrix. Neural network pre-training is not used and, to demonstrate further the adaptive feature of the proposed controller, the model used for simulations is abruptly modified by replacing T by $1.5T$ in (7) in the middle of the simulation, specifically at 12.5 seconds. This has the effect of modifying the robot model considerably without altering its kinematic structure. The selected reference trajectory was generated recursively (one time step ahead) using:

$$\begin{aligned} x_r(t) &= 2\cos(0.25t), \\ y_r(t) &= 2\sin(0.5t), \\ \theta_r(t) &= \arctan\left(\dot{y_r}/\dot{x_r}\right), \end{aligned} \tag{37}$$

which define a figure-of-eight path in the x-y plane.

Neural networks $\mathbf{NN_f}$ and $\mathbf{NN_G}$ were structured with 18 and 1 hidden unit neurons respectively. The EKF initial covariance $\mathbf{P}_0$ was set to $\mathbf{5I}_\alpha$, where α is the total number of weights. The initial weight values were randomly generated from a normal distribution with zero mean. The controller design parameters were set according to (36) with the eigenvalues placed inside the unity circle. Several simulation trials were conducted, each indicating good tracking and repeatable performance for the proposed algorithm. A number of results from a particular 25 second duration simulation, with a time step of 10 ms, are depicted in Figure 1.

Referring to Figure 1, plots (a) to (g) were generated using the proposed adaptive controller while plot (h) corresponds to the non-adaptive controller proposed in (Corradini et al., 2003) under the same specified simulation conditions and assuming that the original robot functions are known. Primarily one should note that the pose errors shown in plots (a), (b) and (c) remain well bounded about zero during the whole trajectory, indicating that the proposed scheme has learned the nonlinear functions and yields stable control throughout the simulation, including the period following the previously mentioned model variation at time 12.5 seconds. This variation is overcome with no more than a slight error transient visible in (b). This transient dies out quickly once the controller adapts to the recently modified model. By contrast, plot (h) reveals that the non-adaptive method proposed in (Corradini et al., 2003) goes unstable just after this model variation.

The complete trajectory path in the $x - y$ plane is superimposed on the reference trajectory in plot (e). The two are hardly distinguishable due to the good tracking performance obtained. The relatively high deviation in the initial part of the trajectory, magnified in plot (f), corresponds to the learning phase of the neural networks, which require no preliminary off-line training.

The neural network weights remain well bounded at reasonable values. Plot (d) depicts time plots for three particular weights, selected arbitrarily. For these particular simulations, the absolute maximum and minimum weight values over the two trajectories were 8.2 and -9.4 respectively, with the majority of the weights centred about zero. This indicates that the proposed algorithm is also practically realizable as no infinitely high neural network weights are required. As a result, the control velocities v and ω remained well bounded with decent magnitudes.

Plot (g) depicts the mean of the diagonal of the covariance matrix $\mathbf{P}_k$ in time. This is used to indicate the uncertainty in the estimated weights, as generated recursively by the EKF. As expected, the uncertainty decreases with time, indicating that the EKF is stable and the neural network estimations are continuously improving. The slope of this curve is related to the learning rate of the system.

6 CONCLUSIONS

In this paper a neural network adaptive control scheme for the trajectory tracking problem of mobile robots is proposed. The resulting algorithm requires no preliminary information about the process non-linear functions and uses MLP neural networks, trained online in consideration of the process uncertainties and external disturbances by using the EKF. The designed scheme was tested repeatedly by simulation for several noise conditions and sudden model variations, modeling the uncertainty and time-varying parameters encountered in practical environments. In contrast to the non-adaptive scheme proposed in (Corradini et al., 2003), the robot exhibited good tracking performance in each case.

Future research will include the development of stochastic non-linear control laws (Fabri and Kadirkamanathan, 2001), which would take into account the neural network approximation errors recursively through the readily available covariance matrix $\mathbf{P}_k$. Such a controller would then be amalgamated with the estimation algorithm developed in this paper, replacing the current heuristic certainty equivalence control law. This is anticipated to give better transient performance due to its stochastic features. Stability proofs in the ambience of stochastic control are very rare due to the complexity of random processes, but is still part of our agenda for future work.

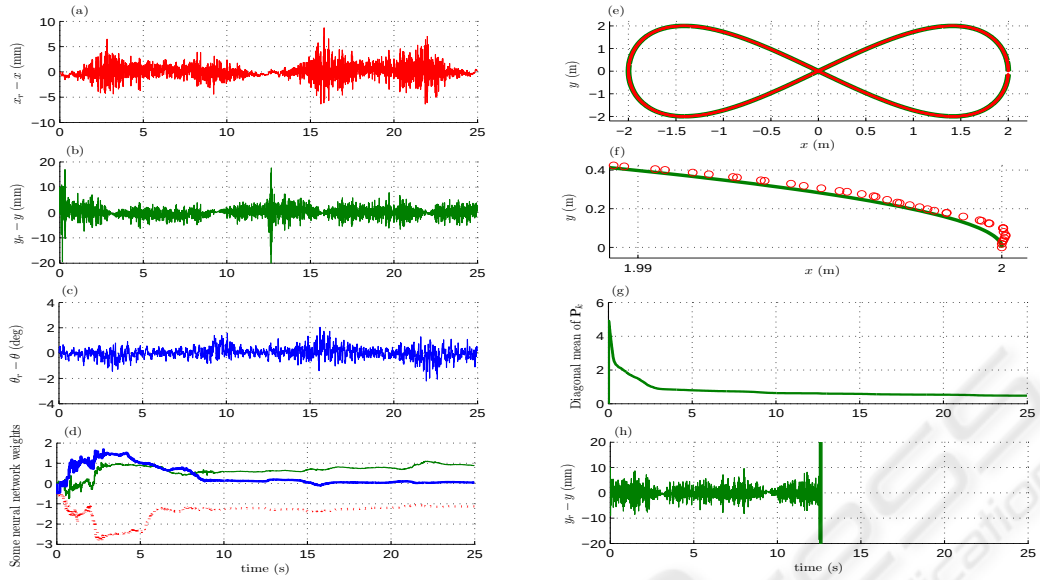

Figure 1: Proposed algorithm: (a) error in x, (b) in y, (c) in θ, (d) some weights, (e) complete reference (—) and actual (○) trajectories, (f) initial phase of (e), (g) diagonal mean of $\mathbf{P}_k$. Non-adaptive algorithm: (h) error in y.

REFERENCES

Asensio, J. R. and Montano, L. (2002). A kinematic and dynamic model-based motion controller for mobile robots. In *Proc. 15th IFAC World Congress on Automatic Control (b'02)*, Barcelona, Spain.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, NJ.

Corradini, M. L., Ippoliti, G., and Longhi, S. (2003). Neural networks based control of mobile robots: Development and experimental validation. *Journal of Robotic Systems*, 20(10):587–600.

Corradini, M. L. and Orlando, G. (2001). Robust tracking control of mobile robots in the presence of uncertainties in the dynamic model. *Journal of Robotic Systems*, 18(6):317–323.

Ding, D. and Cooper, R. A. (2005). Electric-powered wheelchairs. *IEEE Control Systems Magazine*, 25(2):22–34.

Fabri, S. G. and Kadirkamanathan, V. (1998). Dual adaptive control of nonlinear stochastic systems using neural networks. *Automatica*, 34(2):245–253.

Fabri, S. G. and Kadirkamanathan, V. (2001). *Functional Adaptive Control: An Intelligent Systems Approach*. Springer-Verlag, London, UK.

Fierro, R. and Lewis, F. L. (1995). Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics. In *Proc. 34th Conference on Decision and Control (CDC'95)*, pages 3805–3810, New Orleans, LA.

Fierro, R. and Lewis, F. L. (1997). Robust practical point stabilization of a nonholonomic mobile robot using neural networks. *Journal of Intelligent and Robotic Systems*, 20:295–317.

Fierro, R. and Lewis, F. L. (1998). Control of a nonholonomic mobile robot using neural networks. *IEEE Trans. Neural Networks*, 9(4):589–600.

Ge, S. S., Lee, T. H., and Harris, C. J. (1999). *Adaptive Neural Network Control of Robotic Manipulators*, volume 19 of *World Scientific Series in Robotics and Intelligent Systems*. World Scientific Publishing Co Pte Ltd, USA.

Grech, R. and Fabri, S. G. (2004). Trajectory tracking of a differentially driven wheeled mobile robot in the presence of obstacles. In *Proc. 12th Mediterranean Conference on Control and Automation (MED'04)*, Izmir, Turkey.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*, chapter 5. Prentice Hall, London, UK.

Julier, S. J. and Uhlmann, J. K. (1997). A new extention of the Kalman filter to nonlinear systems. In *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*.

Lewis, F. L., Jagannathan, S., and Yesildirek, A. (1999). *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor and Francis, Padstow, UK.

Maybeck, P. S. (1979). *Stochastic Models, Estimation and Control*, volume 141-1 of *Mathematics in Science and Engineering*. Academic Press Inc., London, UK.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, 1(1):4–27.

Oriolo, G., Luca, A. D., and Vendittelli, M. (2002). WMR control via dynamic feedback linearization: Design, implementation and experimental validation. *IEEE Trans. Contr. Syst. Technol.*, 10(6):835–852.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proc. IEEE*, 78(9):1481–1497.