Strojniški vestnik - Journal of Mechanical Engineering 58(2012)2, 93-101
DOI: 10.5545/sv-jme.2011.098
Paper received: 2011-05-05 , Paper accepted: 2011-11-07

Neural-Network Estimation of the Variable Plant for Adaptive Sliding-Mode Controller

Suzana Uran – Riko Šafarič*

University of Maribor, Faculty of Electrical Engineering and Computer Science, Slovenia

The Lyapunov based theoretical development of a neural-network sliding-mode based estimation of highly non-linear and variable robot plant for a direct-drive robot controller is shown in the paper. Derived adaptive control law was tested for four types of robot neural-network sliding-mode controllers: centralized, simplified centralized, decentralized and simplified decentralized, which were verified on a real laboratory direct-drive 3 D.O.F. PUMA like mechanism. Centralized and decentralized control approaches estimate only a part of the variable robot dynamic model (torque model due to friction, Coriolis, centripetal and centrifugal forces) and use only the part of a dynamic plant model (the so called estimated inertia matrix M). Both simplified methods do not need any plant model parameter for an accurate estimation of the direct-drive robot plant, but need some more time to learn dynamic model parameters. All four types of the neural network continuous sliding-mode controllers were successfully tested for algorithm's adaptation capability for sudden changes in the manipulator dynamics (load).
Keywords: sliding-mode adaptive controller, neural-network, robot

0 INTRODUCTION

Control techniques based on soft computing methods (neural network, fuzzy logic, genetic algorithm, particle swarm algorithm, fractal theory etc. or their combinations) [1] and especially neural network control techniques have been proven to be useful to control highly nonlinear robot arm control plants for more than two decades [2] to [8]. Therefore, the neural network control methods have been used with high interest in mobile robotics [9] to [11], especially for the dynamic and kinematic control in recent years. The research of the kinematic and dynamic neural network control of robot arm has also been evolving for two decades.

A number of publications dealing with the topic of the robot arm trajectory tracking neural network controller based on the computed torque method [12] to [15], etc. have been published. In sources [12] and [14], there was an attempt to replace the estimated model of the real mechanism (the vector **h** due to Coriolis forces and the inertia matrix **M**) with two neural networks. The disadvantage of this method is that it requires generalized learning [12] in addition to specialized learning or a time-consuming convergence of neural network learning [14] if generalized learning is not implemented. In order to speed up the convergence without generalized learning, the source [13] retained the complete compensator based on the computed torque method and added a neural network approximating an unstructured uncertainty, which would not be compensated by the computed torque method itself (friction torque) and would introduce an error into the control system if used with this method. The disadvantage of the method described in the source [13] is that the parameters of the inertia matrix **M** and vector **h** (torques due to Coriolis, centrifugal and centripetal forces) have to be known.

The sources [16] to [19] successfully deal with the neural network control based on an estimation of kinematics and dynamics of geared robot arms. The source [15] tried to adapt neural network controller based on the computed torque method also to high nonlinear direct drive (DD) robot arm dynamics and obtained good results in the tracking experiments, while the steady-state test and the sudden load change test had not been reported. The published paper [20] resolved the steady state problem. In order to diminish the drawbacks of all the above mentioned methods, a sliding-mode neural network controller was chosen as a robust control scheme [21], where only nominal (average) values of inertia matrix parameters were used, while the differences between actual inertia matrix parameters and nominal inertia matrix parameters torque terms due to Coriollis forces, gravitational forces and friction forces (structured uncertainties) were estimated by neural network. This was done due to the fear that the robot behaviour would be unpredictable during the first few moments of neural network learning. This method was successfully upgraded and used also for visual positioning control of robot mechanism [22] where a special four-layer neural network structure made possible to estimate the complete robot dynamics and kinematics. The next two reports [23] and [24] had shown that neural network based control approach could be effectively used also for direct driven piezo electric actuated micro robot mechanisms.

The theoretical development of a full and simplified centralized and decentralized neural-

*Corr. Author's Address: Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia, riko.safaric@uni-mb.si

network controller based on the theory of continuous sliding-mode control for DD robot arm mechanism is shown in the paper. Derived equations, based on Lyapunov theory, of the adaptive neural network controller were verified on a real laboratory direct-drive 3. D.O.F PUMA like mechanism. The newly developed neural network continuous sliding-mode centralized and decentralized controllers, as full and simplified sub-methods were successfully tested for adaptation capability of the algorithm for sudden load changes in the manipulator dynamics. All the mentioned tests were made on a real laboratory 3 D.O.F. DD robot mechanisms.

The main idea presented in the paper is to give the neural network controller only a part of robot dynamic, which does not include the coupling effect between axes, so the structual and unstructual uncertainties increase. It is shown in the paper that a neural network as a part of control law is able to learn the missing part of the robot dynamics which should be included in the control law during the learning procedure. Four methods, which have more or less robot dynamic, included in the neural network control law are presented and compared.

1 SYNTESIS OF CONTINUOUS NEURAL-NETWORK SLIDING-MODE CONTROLLER

A well known mathematical note of robot mechanism dynamics, Eq. (1), is transformed into an *n*-dimensional state-space system of equations with regard to the control value *u,* Eq. (2), because the Lyapunov theory for searching the control law can only be used in the following way.

$$\mathbf{T} = \mathbf{M}(\boldsymbol{\theta}) \cdot \ddot{\boldsymbol{\theta}} + \mathbf{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G_f}(\boldsymbol{\theta}) + \mathbf{F}(\dot{\boldsymbol{\theta}}) + \mathbf{T_n}, \qquad (1)$$

where **T** is a torque vector, **M** is an inertial matrix, **h** is a torque vector due to centrifugal forces, centripetal forces, and Coriollis forces, **F** is a torque vector due to frictional forces, $\mathbf{G_f}$ is a torque vector due to forces of gravity, $\mathbf{T_n}$ is a torque vector due to unknown disturbances, $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$ are vectors of real positions, velocity, and accelerations of the robot mechanism. The Eq. (2) presents a non-linear state-space system as a description of Eq. (1) and it is needed for the control law development by the Lyapunov theory.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{B}(\mathbf{x}, t) \cdot \mathbf{u} + \mathbf{d}(\mathbf{x}, t), \qquad (2)$$

where:

$$\mathbf{x} \in \mathrm{R}^n, \mathbf{u} \in \mathrm{R}^m, \mathbf{B}(\mathbf{x}, t) = \tilde{\mathbf{B}}(\mathbf{x}, t) + \Delta\mathbf{B}(\mathbf{x}, t), \qquad (3)$$

and **d** is an unknown disturbance, **B** is an actual input matrix, $\tilde{\mathbf{B}}$ is an estimated input matrix, **u** is a control vector, **x** is a state space vector of mechanism, and *t* stands for time. Our goal is to prove the function stability $\sigma(\mathbf{x}, t) = 0$ (Eq. (4)) for the robot system (Eq. (2)). This means that after transient time, defined with parameters of the matrix **G**, the difference between the actual and the desired vector of state space variables **x** and will equal zero and will be stable for all disturbances. Function $\sigma(\mathbf{x}, t) = 0$ will be stable if the Lyapunov function $V > 0$ and the first Lyapunov time derivative of function $\dot{V} < 0$. The selected Lyapunov function *V* (Eq. (5)) is always greater than zero for whichever selected vector $\mathbf{x_r}$, **x** and matrix **G**. However, it is not always possible to get the negative first derivative of the Lyapunov time-dependent function $\dot{V}$ (Eq. (6)) for every $\mathbf{x_r}$, **x** and **G**. According to the following equation:

$$\sigma(\mathbf{x}, t) = \mathbf{G}(\mathbf{x}(t) - \mathbf{x_r}(t) = \sigma = \mathbf{G}(\mathbf{x} - \mathbf{x_r}), \qquad (4)$$

where $\mathbf{x_r}$ is a vector of the desired state space variable and **G** is the matrix defining the control of system dynamics, we cannot prove the robot system stability (Eq. (2)). Nevertheless, we can look for suitable conditions for control law *u*, where the robot system will be stable. This is done in the following way.

For the simplest Lyapunov function *V* to determine the control law **u**, the following equation has been selected:

$$V = \sigma^{\mathbf{T}} \cdot \sigma / 2 . \qquad (5)$$

The following is derived from the Eq. (5):

$$\dot{V} = \sigma^{\mathbf{T}} \cdot \dot{\sigma} . \qquad (6)$$

Owing to the fact that $\dot{V}$ is not always less than zero for all $\mathbf{x_r}$, **x** and **G**, the first desired Lyapunov negative time function derivative has been defined as:

$$\dot{V} = -\sigma^{\mathbf{T}} \cdot \mathbf{D} \cdot \sigma, \qquad (7)$$

where **D** is a diagonal matrix with positive diagonal elements.

If the Eq. (7) and the derivative of Lyapunov's Eq. (6) are made equal, the result is:

$$\sigma^{\mathbf{T}}(\mathbf{D}\sigma + \dot{\sigma}) = 0. \qquad (8)$$

The Eq. (8) is valid if both or at least one of multiplicators equals zero. Since the first multiplicand, the term $\sigma^{\mathbf{T}}$, does not equal zero during the transient response, the control law can be calculated on the basis of the second multiplicand (Eq. (9)):

$$\mathbf{D} \cdot \sigma + \dot{\sigma} = 0. \qquad (9)$$

If Eq. (4) is differentiated and the Eq. (2) is inserted into the recently calculated derivative, we get the following result:

$$\dot{\sigma} = \mathbf{G}(f + \tilde{\mathbf{B}}\mathbf{u} + \Delta\mathbf{B}\mathbf{u} + \mathbf{d} - \mathbf{x}_r). \quad (10)$$

After Eq. (10) has been inserted into the implementation condition of control law Eq. (9), the result is as follows:

$$\mathbf{u} = -(\mathbf{G}\tilde{\mathbf{B}})^{-1}\left[\mathbf{G}(\mathbf{f} + \Delta\mathbf{B}\mathbf{u} + \mathbf{d} - \dot{\mathbf{x}}_r) + \mathbf{D}\dot{\sigma}\right]. \quad (11)$$

Since the term ($\mathbf{f} + \Delta\mathbf{B}\cdot\mathbf{u} + \mathbf{d}$) is unknown and not measurable, it is, therefore, approximated with the neural network $\mathrm{N} = [o_1 \ldots o_i]^{\mathrm{T}}$ (see Fig. 1) by changing the Eq. (11) into:

$$\mathbf{u} = -(\mathbf{G}\tilde{\mathbf{B}})^{-1}\left[\mathbf{G}(\mathbf{N} - \dot{\mathbf{x}}_r) + \mathbf{D}\sigma\right]. \quad (12)$$

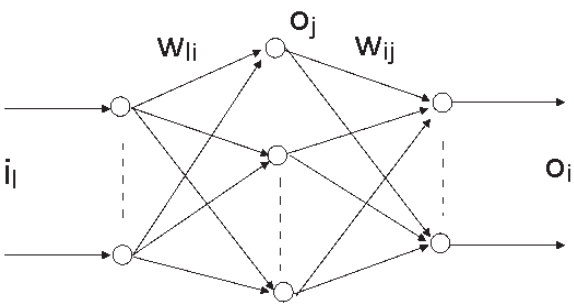

Fig. 1. *Neural network*

Since the term ($\mathbf{f} + \Delta\mathbf{B}\cdot\mathbf{u} + \mathbf{d}$) is unknown and not measurable, a classic supervised weight learning of neural network cannot be used. Therefore, a so-called on-line neural network estimator has been developed (Fig. 2), estimating a learning signal (that is the difference between the target and the output of a neural network).

The result after Eq. (4) has been differentiated is the following:

$$\dot{\mathbf{x}} = \mathbf{G}^{-1}\cdot\dot{\sigma} + \dot{\mathbf{x}}_r. \quad (13)$$

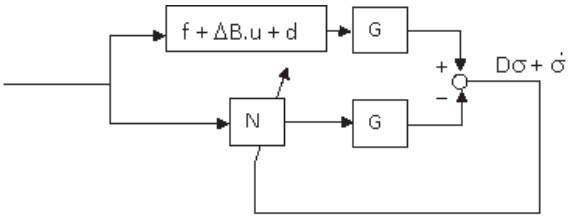

Fig. 2. *Neural network on-line estimator*

After Eqs. (12) and (13) have been inserted into the basic equation of mechanism dynamics (Eq. (2)), the result is as follows:

$$\dot{\sigma} + \mathbf{D}\sigma = \mathbf{G}(\mathbf{f} + \Delta\mathbf{B}\mathbf{u} + \mathbf{d}) - \mathbf{G}\mathbf{N} = \mathbf{G}(\mathbf{Z} - \mathbf{N}), \quad (14)$$

where we have substituted $\mathbf{Z} = (\mathbf{f} + \Delta\mathbf{B}\cdot\mathbf{u} + \mathbf{d})$. To learn the weights of a neural network hidden layer the traditionally back-propagation rule [25] is used.

1.1 Centralized Control Law for Three Degrees of Freedom Mechanism

In the previous section, the control law for a general robot mechanism with *n*-degrees of freedom has been derived; in this section, detailed equations of control law for a direct drive robot mechanism with three degrees of freedom, which is shown in Fig. 4, will be derived.

$$\mathbf{T} = \widehat{\mathbf{M}}\ddot{\boldsymbol{\theta}} + \hat{\mathbf{h}} + \widehat{\mathbf{G}}_{\mathbf{f}} + \mathbf{T}_{\mathbf{n}}, \quad (15)$$

where $\mathbf{T}$, $\hat{\mathbf{h}}$, $\widehat{\mathbf{G}}_f$, and $\mathbf{T_n}$ (see Eq. (1)) are column vectors of the 3×1 dimension, $\widehat{\mathbf{M}}$ is the matrix of the 3×3 dimension, and $\boldsymbol{\theta} = [\theta_1\ \theta_1\ \theta_1]^{\mathrm{T}}$ is the column vector of the 3×1 dimension of all three axes of the robot and where $\widehat{\mathbf{M}}$, $\hat{\mathbf{h}}$ and $\widehat{\mathbf{G}}_f$ are estimated and simplificated values of real $\mathbf{M}$, $\mathbf{h}$ and $\mathbf{G_f}$ (see Eq. (1)). Only nominal or average parameters of the matrix $\widehat{\mathbf{M}}$ have been selected. This means that all 9 parameters of the matrix $\widehat{\mathbf{M}}$ are constant while the robot hand is moving. This is, of course, only a rough simplification of how things really look like; for it is a common fact that the parameters of matrix $\widehat{\mathbf{M}}$ vary according to individual axis movements in robot's working space. The previous equation can also be rewritten in the following form (see also Eq. (3)):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + \tilde{\mathbf{B}}(\mathbf{x},t)\mathbf{u} + \mathbf{d}(\mathbf{x},t), \quad (16)$$

where:

$$\begin{aligned}\mathbf{x} &= \left[\theta_1 \quad \theta_2 \quad \theta_3 \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3\right]^T,\\ \dot{\mathbf{x}} &= \left[\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \ddot{\theta}_1 \quad \ddot{\theta}_2 \quad \ddot{\theta}_3\right]^T,\end{aligned} \quad (17)$$

$$\mathbf{f} = -\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\\ \widehat{\mathbf{M}}^{-1}\left[\hat{\mathbf{h}} + \widehat{\mathbf{G}}_{\mathbf{f}}\right]\end{bmatrix},\quad \tilde{\mathbf{B}} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ & \widehat{\mathbf{M}}^{-1} & \end{bmatrix}, \quad (18)$$

and where $\mathbf{u}$ is calculated vector of T done by control law (Eq. (19)).

Because of that the unknown variable part $\Delta\mathbf{B}$ exists and is estimated by the neural network (see

Eq. (14)). The dimension of vector **f** is 6×1 and the dimension of the matrix $\tilde{\mathbf{B}}$ is 6×3. The control law **u** of the 3×1 dimension is illustrated in the following equation:

$$\mathbf{u} = -(\mathbf{G}\tilde{\mathbf{B}})^{-1}\left[\mathbf{G}(\mathbf{N} - \dot{\mathbf{x}}_\mathbf{r}) + \mathbf{D}\sigma\right], \tag{19}$$

where:

$$\mathbf{G} = \begin{bmatrix} K_{p1} & 0 & 0 & K_{v1} & 0 & 0 \\ 0 & K_{p2} & 0 & 0 & K_{v2} & 0 \\ 0 & 0 & K_{p3} & 0 & 0 & K_{v3} \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}, \tag{20}$$

$$\mathbf{x_r} = \begin{bmatrix} \theta_{1r} & \theta_{2r} & \theta_{3r} & \dot{\theta}_{1r} & \dot{\theta}_{2r} & \dot{\theta}_{3r} \end{bmatrix}^T, \quad \dot{\mathbf{x}}_\mathbf{r} = \begin{bmatrix} \dot{\theta}_{1r} & \dot{\theta}_{2r} & \dot{\theta}_{3r} & \ddot{\theta}_{1r} & \ddot{\theta}_{2r} & \ddot{\theta}_{3r} \end{bmatrix}^T, \tag{21}$$

and

$$\sigma = \mathbf{G}\,(\mathbf{x} - \mathbf{x_r}). \tag{22}$$

Coefficients of the matrices **G** in **D** are selected in such a way that they enable the fastest convergence of neural network algorithm possible. The column vector **N** is of the 6×1 dimension and represents the outputs of the neural network o_i with $i = 1, ..., 6$.

The learning procedure for all the weights of an output layer is:

$$\begin{aligned} \Delta w_{1j} &= \mu_J \begin{bmatrix} K_{p1} & 0 & 0 \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_1)o_j, \\ \Delta w_{2j} &= \mu_J \begin{bmatrix} 0 & K_{p2} & 0 \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_2)o_j, \\ \Delta w_{3j} &= \mu_J \begin{bmatrix} 0 & 0 & K_{p3} \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_3)o_j, \\ \Delta w_{4j} &= \mu_J \begin{bmatrix} K_{v1} & 0 & 0 \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_4)o_j, \\ \Delta w_{5j} &= \mu_J \begin{bmatrix} 0 & K_{v2} & 0 \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_5)o_j, \\ \Delta w_{6j} &= \mu_J \begin{bmatrix} 0 & 0 & K_{v3} \end{bmatrix}(\mathbf{D}\sigma + \dot{\sigma})g'(net_6).o_j, \end{aligned} \tag{23}$$

$$net_i = \sum_j w_{ij}o_j + b_i\,, \tag{24}$$

where $j = 1,..., 60$, $i = 1, ..., 6$, $l = 1, ..., 9$, and *g'(*)* is the first derivative of the sigmoid function [25].

The neural network of centralized neural network sliding mode controller (CNNSMC) consists of 9 inputs; these are: three actual positions, three actual velocities, and three differences between the desired and the actual position. All of them lie in the joint space of the robot mechanism. The scheme of CNNSMC is shown in Fig. 3.

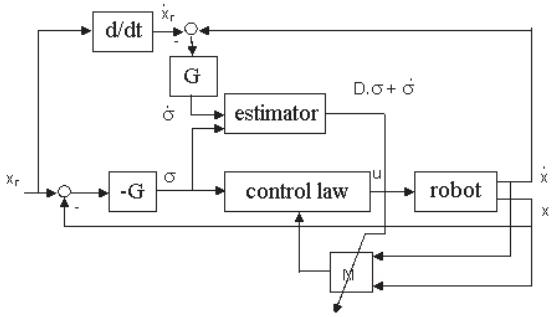

Fig. 3. *CNNSMC control scheme*

1.2 Decentralized Control Law for Three Degrees of Freedom Mechanism

Eq. (1) is simplified for the first single axis (θ1) of DD robot mechanism (Eq. (25)):

$$\mathbf{T}_1 = \hat{\mathbf{J}}_1\boldsymbol{\theta}_1 + \hat{\mathbf{h}}_1 + \hat{\mathbf{g}}_1 + t_{n1}, \tag{25}$$

where scalars T_1, $\hat{\mathbf{h}}_1$, $\hat{\mathbf{g}}_1$ and t_{n1} are torques needed to move the single axis. $\hat{\mathbf{h}}_1$ is torque due to Coriollis centripetal and centrifugal forces, $\hat{\mathbf{g}}_1$ is a torque due to gravitational forces and t_{n1} is unknown torque disturbance. $\hat{\mathbf{J}}_1$ is an average, constant and rough approximation of the single axis inertia parameter. Scalars $\hat{\mathbf{J}}_1$, $\hat{\mathbf{h}}_1$, $\hat{\mathbf{g}}_1$ are estimated and simplificated values of real **M**, **h** and $\mathbf{G_f}$ (see Eq. (1)).

Eq. (25) has been transformed as follows for the first single axis:

$$\dot{\mathbf{x}}_1 = \mathbf{f}_1 + \Delta\mathbf{B}_1\mathbf{u}_1 + \tilde{\mathbf{B}}_1\mathbf{u}_1 + \mathbf{d}_1, \tag{26}$$

where $\Delta\mathbf{B}_1 = \mathbf{B}_1 - \tilde{\mathbf{B}}_1$ and where:

$$\mathbf{x}_1 = \begin{bmatrix} \theta_1, \dot{\theta}_1 \end{bmatrix}^T, \quad \dot{\mathbf{x}}_1 = \begin{bmatrix} \dot{\theta}_1, \ddot{\theta}_1 \end{bmatrix}^T, \tag{27}$$

$$\mathbf{f}_1 = \begin{bmatrix} \dot{\theta}_1 \\ -\hat{\mathbf{J}}_1^{-1}\left[\hat{\mathbf{h}}_1 + \hat{\mathbf{g}}_1\right] \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{B}}_1 = \begin{bmatrix} 0 \\ \hat{\mathbf{J}}_1^{-1} \end{bmatrix}. \tag{28}$$

The dimensions of the vectors $\mathbf{f}_1$, $\mathbf{B}_1$, $\tilde{\mathbf{B}}_1$ are 2×1. The control law $\mathbf{u}_1$ is described by the following equation:

$$\mathbf{u}_1 = -(\mathbf{G}_1\tilde{\mathbf{B}}_1)^{-1}\left[\mathbf{G}_1(\mathbf{N}_1 - \dot{\mathbf{x}}_{\mathbf{r}1}) + \mathbf{d}_1\sigma_1\right], \tag{29}$$

where:

$$\mathbf{G}_1 = [K_{p1}\ \ K_{v1}]\,. \tag{30}$$

The coefficients K_{p1}, K_{v1} and constant $\mathbf{d}_1$ are selected in such a manner that the most rapid convergence of the neural network ($\mathbf{N}_1$ is an output of

the neural network for the first single axis) learning algorithm is made possible.

$$\mathbf{x}_{\mathbf{r}1} = \left[\theta_{r1}, \dot{\theta}_{r1}\right]^T \quad \text{and} \quad \dot{\mathbf{x}}_{\mathbf{r}1} = \left[\dot{\theta}_{r1}, \ddot{\theta}_{r1}\right]^T, \tag{31}$$

where scalars $\boldsymbol{\theta}_{r1}$, $\dot{\boldsymbol{\theta}}_{r1}$ and $\ddot{\boldsymbol{\theta}}_{r1}$ are reference joint position, speed and acceleration of the single axis, respectively.

$$\sigma_1 = \mathbf{G}_1(\mathbf{x}_1 - \mathbf{x}_{\mathbf{r}1}) \text{ and } \dot{\sigma}_1 = \mathbf{G}_1(\dot{\mathbf{x}}_1 - \dot{\mathbf{x}}_{r1}). \tag{32}$$

The column vector $\mathbf{N}_1$ is of the 2×1 dimension and represents the outputs of the neural network o_i (i = a, b). The variable of control law $\mathbf{u}_1$ is a scalar.

The learning procedure for all weights of the neural network output layer is:

$$\begin{aligned} \Delta w_{1aj} &= \eta_{a1} K_{p1}(d_1\sigma_1 + \dot{\sigma}_1) g'(net_a) o_j, \\ \Delta w_{1bj} &= \eta_{b1} K_{v1}(d_1\sigma_1 + \dot{\sigma}_1) g'(net_b) o_j, \end{aligned} \tag{33}$$

$$net_i = \sum_j w_{ij} o_j + b_i, \tag{34}$$

where j = 1, ..., 5 (a number of neurons in the hidden layer), i = 2 (indexes: *a, b* - the number of neurons in the output layer), l = 3 (three neural network inputs were used: an actual position, an actual velocity and a difference between desired and actual positions in the joint space) and *g'(*)* is the first derivative of sigmoidal function. Fig. 3 also shows the scheme of the single axis controller.

The remaining two D.O.F. single axis controllers are the same as the described one in this subsection. Every single axis controller has the equal number of inputs and outputs of the neural network, the equal on-line estimator, the same control law etc. The difference between equations developed in current subsection and equations needed for the second and third axis is that all indexes "1" in Eqs. (25) to (34) are changed from "1" to "2" for the second axis and from "1" to "3" for the third axis. In fact, there are three equal control laws: $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$; the only differences between the above mentioned control laws for all three axes are different values for parameters d_k, K_{pk}, K_{vk} and of course different inputs $\boldsymbol{\theta}_k$, $\dot{\boldsymbol{\theta}}_k$ and $\ddot{\boldsymbol{\theta}}_k$, where k = 1, 2, 3.

1.3 Simplified Centralized and Decentralized Control Laws

Centralized and decentralized control approaches estimate a part of the variable robot dynamic model (torque model due to friction, Coriolis, centripetal and centrifugal forces) and use only the part of a dynamic plant model – the so called estimated inertia matrix **M** (see Eqs. (1) to (3), (11) and (12)). If Eq. (3) is rewritten as:

$$\mathbf{B}(\mathbf{x}, t) = \mathbf{C} + \Delta\mathbf{B}(\mathbf{x}, t), \tag{35}$$

where **C** is a matrix, which includes the simple unity diagonal matrix **I** instead of $\widehat{\mathbf{M}}$, for CNNSMC, so Eq. (18) is changed to:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ & \mathbf{I}^{-1} & \end{bmatrix}, \tag{36}$$

where **I** is a unity diagonal matrix of 3×3 dimension. Consequently, Eq. (19) is also changed to:

$$\mathbf{u} = -(\mathbf{GC})^{-1}\left[\mathbf{G}(\mathbf{N} - \dot{\mathbf{x}}_{\mathbf{r}}) + \mathbf{D}\sigma\right], \tag{37}$$

while matrices **G**, **D** and vectors **N**, $\dot{\mathbf{x}}_{\mathbf{r}}$ and σ are not changed in comparison to the control law of CNNSMC (see Eqs. (16) to (24)). Eq. (37) represents the control law for a simplified centralized neural-network sliding-mode controller (SCNNSMC).

The equation development for the case of simplified decentralized neural-network sliding-mode controller (SDNNSMC) is similar as for the case of the SCNNSMC. Here, Eq. (28) is changed to:

$$\mathbf{C}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{38}$$

Therefore, the control law for the SDNNSMC is rewritten from Eq. (29) as:

$$\mathbf{u}_1 = -(\mathbf{G}_1\mathbf{C}_1)^{-1}\left[\mathbf{G}_1(\mathbf{N}_1 - \dot{\mathbf{x}}_{\mathbf{r}1}) + \mathbf{d}_1\sigma_1\right], \tag{39}$$

where vectors $\mathbf{G}_1$, $\dot{\mathbf{x}}_{\mathbf{r}1}$, $\mathbf{N}_1$ and scalar variables $\mathbf{d}_1$ and σ_1 are not changed in comparison to the control law of DNNSMC (see Eqs. (25) to (34)).

As it is seen from Eqs. (35) to (39), both simplified control laws do not need any plant model parameters for accurate estimation of the direct-drive robot mechanism dynamics.

2 APLICATION ON A REAL MECHANISM

The scheme of a direct-drive three degrees of freedom mechanism is illustrated in Fig. 4, while in Fig. 5 the photo of the robot mechanism is shown. The Dynaserv's AC-motors with maximum torque of 220, 160 and 60 Nm, and the nominal angular velocity 1 to

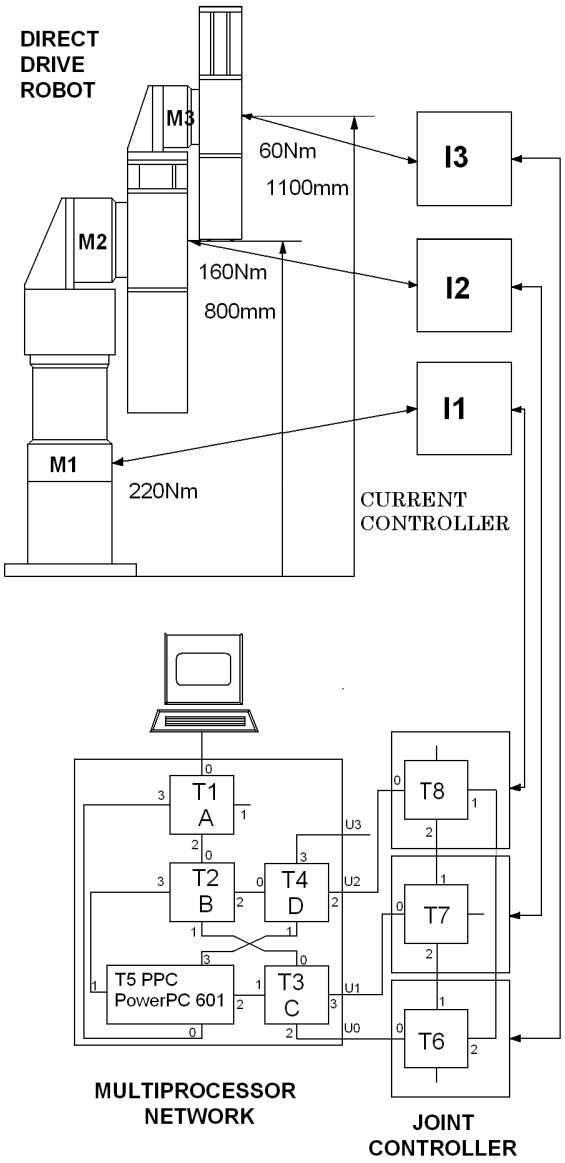

Fig. 4. *The robot system*

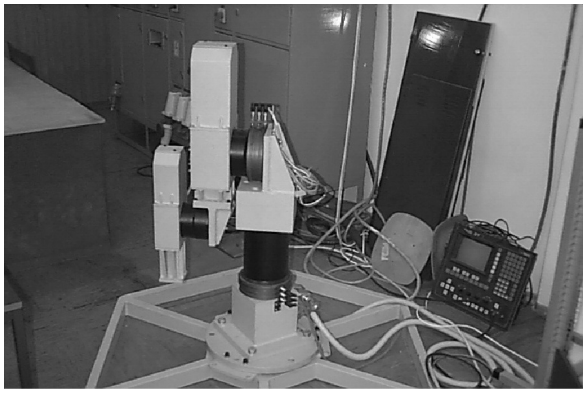

Fig. 5. *The real lab robot mechanism*

2 rotations per second were used. The mechanism is made of aluminium, which is fixed on the AC-motors. A robot wrist can be added to the top of the third axe of the robot.

Since the robot is expected to perform manipulation tasks, the complete system has been tested with all four developed controllers described in previous section to perform the following tasks:

- performance of the PTP movement with the static error less than 0.1 mm, and
- robustness when the top of the robot is disturbed with sudden load changes.

To satisfy the above mentioned demands, a robot controller with the sufficient computed power had to be developed. For a parallel execution of algorithms a transputer network of 8 transputers, one PowerPC, and one ordinary personal computer have been used; all have the possibility of working in parallel.

A robot computer controller is described in a source [21]. The sampling time T_s = 2 ms is the execution time of all algorithms needed for the robot computer control.

The position error of the robot tip (Eq. (40)) or a trajectory tracking error in the task space has been used to measure the quality of all four robot controller performances:

$$e = \left[(X_{di} - X_i)^2 + (Y_{di} - Y_i)^2 + (Z_{di} - Z_i)^2 \right]^{1/2}, \quad (40)$$

where X_{di}, Y_{di}, Z_{di} are reference trajectories in the i^{th} sampling time in the task space and X_i, Y_i, Z_i are the actual trajectories in the i^{th} sampling time in the task space.

2.1 The Test of Sudden Load Changes

The position error of a robot tip in a stationary position for the centralized neural-network sliding-mode controller (CNNSMC) is shown in Fig. 6, when sudden load changes occurred (approximately 80% of the maximal torque on the robot tip). The initial weights of neural network were randomly chosen between −1 and +1 learning rate η = 1e^{-8}, d_1 = 20, d_2 = d_3 = 30, K_{p1} = K_{p2} = K_{p3} = 100 and K_{v1} = K_{v2} = K_{v3} = 60.

The position error of the robot tip in stationary position for decentralized neural-network sliding-mode controller (DNNSMC) is shown in Fig. 7 when the same sudden load changes occurred as in previous test. The initial weights of DNNSMC were randomly chosen between -1 and +1, learning rate $\eta_{1a,b}$ = 4e^{-7}, $\eta_{2a,b}$ = 6e^{-6}, $\eta_{3a,b}$ = 6e^{-6}, d_1 = 15, d_2 = 23, d_3 = 20,

K_{p1} = 115, K_{p2} = 150, K_{p3} = 180 and K_{v1} = 25, K_{v2} = 40, K_{v3} = 20.

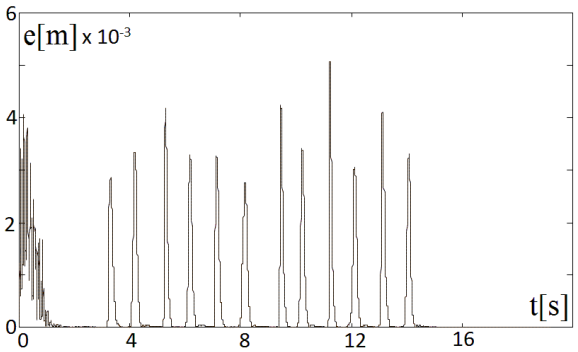

Fig. 6. *Robot tip's position error for CNNSMC during load changes in the stationary position test*

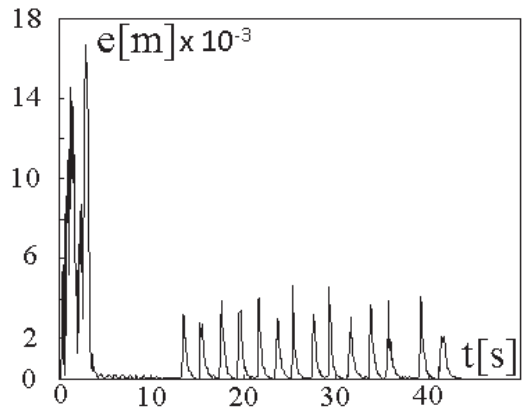

Fig. 7. *Robot tip's position error for DNNSMC during load changes in the stationary position test*

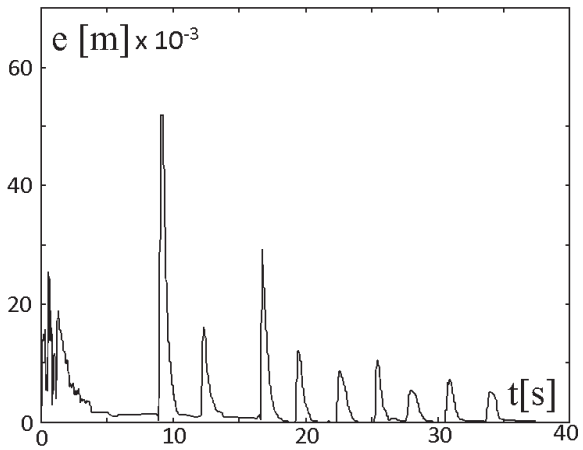

Fig. 8. *Robot tip's position error for SCNNSMC during load changes in the stationary position test*

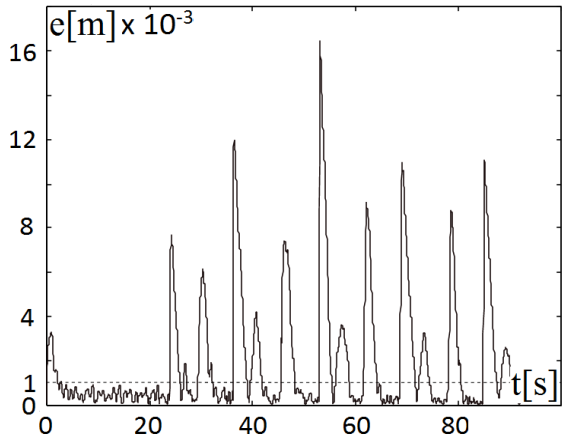

Fig. 9. *Position error of the robot tip for SDNNSMC during load changes in the stationary position test*

The results are almost the same for the test of sudden load changes in the case of CNNSMC and DNNSMC. The difference is found in the first few seconds of Figs. 6 and 7 (PTP movement). The point to point robot tip movement is executed during this time (all three axis start in the same positions and finish in position 1 rd). It could be observed that the dynamic error is higher and the set-up time is longer in a case of DNNSMC. The sudden load changes of robot tip position for SCNNSMC and SDNNSMC are presented in Figs. 8 and 9.

2.2 Summary of Results

The quality of the presented DNNSMC is practically the same as for CNNSMC. The disadvantage of DNNSMC against CNNSMC is that the CNNSMC has a shorter set-up time and a smaller dynamic error during the PTP movement (see Figs. 6 and 7). The advantage of DNNSMC against CNNSMC is that DNNSMC has three completely separated control law equations with three remarkably smaller neural networks (each neural network has 5 neurons in the hidden layer, two outputs and three inputs). Therefore, a learning procedure of the neural network can be made for each axis separately which is remarkably easier than in a case of one neural network of CNNSMC with nine inputs, eighty neurons in the hidden layer and six outputs. Due to this reason the robot control computer hardware could run more axes at the same sampling time in the case of DNNSMC than in the case of CNNSMC. The average execution time for CNNSMC was 1.75 ms, while the average execution time for all three DNNSMCs was 1.05 ms. This sampling time also includes the complete direct

and inverse kinematics, interpolators, etc., for the robot controller.

If a comparison between both simplified (SCNNSMC and SDNNSMC) methods against "full" methods (CNNSMC and DNNSMC) is made, the next observation can be seen: both simplified methods need a remarkably greater set-up time than the other two. In case of the initial PTP movement the set-up time is approximately 5 s in case of SCNNSMC against 2 s for the CNNSMC. Also, the set-up time for the transient responses to load changes is smaller in the case of "full" methods against simplified methods. The set-up time for the transient responses to load changes is 1 s for CNNSMC, 2 s for DNNSMC and SCNNSMC and 5 s for SDNNSMC.

The observation of peak values of a position error of PTP movement and during the load changes is also important for the quality comparison between the mentioned four methods (see Figs. 6 to 9). It can be seen that the best results, the smallest peak values of the position error of the robot tip for PTP movement is observed for CNNSMC (4 mm) and DNNSMC (16 mm) while the peak values of position error during load changes (disturbances) have almost the same values (4 to 5 mm) for CNNSMC and DNNSMC. The peak values of position error for PTP movement is higher for both simplified methods: SCNNSMC (32 mm) and SDNNSMC (18 mm), while an interesting effect can be observed when the position error of the robot tip during the load changes is measured. In the case of SCNNSMC the peak position error continuously decreases from the first load change (66 mm) to next load changes and it is only 5 mm in the end of experiment, which is practically the same result as for CNNSMC and DNNSMC.

This means that in the case of SCNNSM there is a longer learning period because the neural network sliding-mode estimator has to learn complete robot dynamic and not only a part as in the case of "full" methods. The worst results, which means the highest peak position error of the orbot tip was measured in the case of SDNNSMC (10 to 17 mm).

The steady-state position error of the robot tip was the smallest in the case of CNNSMC after the PTP movement and after load changes (practically zero) and almost zero in the case of DNNSMC, while in the case of SCNNSMC decreasing value of steady state position error is measured from the beginning of the experiment to the end of experiment. The worst result, the highest value of the steady-state position error is measured in the case of SDNNSMC where the steady state error is constantly between 0.5 to 1 mm.

3 CONCLUSIONS

This paper has presented the experimental development and laboratory implementation of four: centralized, decentralized, simplified centralized and simplified decentralized neural network continuous sliding-mode controllers for manipulation tasks for the real direct-drive 3 D.O.F. PUMA like robot manipulator. The neural network sliding-mode structure of the controller has been used to estimate and compensate structured (the inertia matrix) and unstructured (torques due to Coriollis forces, gravitational forces, friction forces, etc.) uncertainties of the robot manipulator. The adaptive and self-improving capability of the neural network controllers to the unstructured effects (sudden load changes) has been shown for all four neural network controllers.

4 REFERENCES

[1] Melin, P., Castillo, O. (2001). A new method for adaptive model-based control of non-linear dynamic plants using a neuro-fuzzy-fractal approach. *Soft Computing,* vol. 5, p. 171-177, DOI:10.1007/s005000000069.

[2] Narendra, K.S., Parthasarathy, K. (1990). Identification and control of dynamic systems using neural networks. *IEEE Transactions on Neural Networks*, vol. 1, p. 4-27, DOI:10.1109/72.80202.

[3] Bekey, G.A., Goldberg, K.Y. (Eds.) (1993). *Neural networks in robotics.* Kluwer, Boston.

[4] Miller III, T.W., Sutton, R.S., Werbos, P.J. (eds.) (1996). *Neural Networks for Control.* The MIT Press, Massachusetts.

[5] Ge, S.S., Lee, T.H., Harris, C.J. (1998). *Adaptive neural network control of robotic manipulators.* World Scientific, Singapore.

[6] Lewis, F.L., Jagannathan, L.S., Yeşildirek, A. (1999). *Neural network control of robot manipulators and nonlinear systems.* Taylor & Francis, London, Philadelphia.

[7] Ge, S.S., Hang, C.C., Lee, T.H., Zhang, T. (2002). *Stable adaptive neural network control.* Kluwer Academic, Boston.

[8] Jagannathan, S. (2006). *Neural network control of nonlinear discrete-time systems.* Taylor & Francis, London, New York.

[9] Weitzenfeld, A. (2008). A prey catching and predator avoidance neural-schema architecture for single and multiple robots. *Journal of Intelligent & Robotic Systems,* vol. 51, no. 2, p. 203-233, DOI:10.1007/s10846-007-9183-4.

[10] Dierks, T., Jagannathan, S. (2009). Asymptotic adaptive neural network tracking control of nonholonomic mobile robot formations. *Journal of Intelligent and*

Robotic Systems, vol. 56, p. 153-176, DOI:10.1007/s10846-009-9336-8.

[11] Bugeja, M., Simon, K., Fabri, G. (2011). Dual adaptive neurocontrol of mobile robots using the unscented transform: Monte Carlo and experimental validation. *Computational Intelligence,* vol. 34, p. 237-250, DOI:10.1007/978-3-642-20206-3_16.

[12] Ozaki, T., Suzuki, T., Furuhashi, T., Okuma, S., Uchikawa, Y. (1991). Trajectory control of robotic manipulator using neural networks. *IEEE Transaction on Industrial Electronics*, vol. 38, p. 195-202, DOI:10.1109/41.87587.

[13] Ishiguro, A., Furuhashi, T., Okuma, S., Uchikawa, Y. (1992). A neural network compensator for uncertainties of robotics manipulators. *Transactions on Industrial Electronics,* vol. 39, no. 6, p. 555-570.

[14] Šafarič, R., Jezernik, K. (1994). Trajectory tracking neural network controller for a robot mechanism and Lyapunov theory of stability. *Conference Proceedings IROS,* p. 626-633.

[15] Wu, C., Huang, C. (1996). Back-propagation neural networks for identification and control of a direct drive robot. *Journal of Intelligent & Robotic Systems,* vol. 1, p. 45-64, DOI:10.1007/BF00309655.

[16] Jiang, Z., Ishita, T. (2008). A neural network controller for trajectory control of industrial robot manipulators. *Journal of Computers*, vol. 3, no. 8, p. 93-104, DOI:10.4304/jcp.3.8.1-8.

[17] Parikh, P., Sarah, J., Lam, S. (2009). Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy. *The International Journal of Advanced Manufacturing Technology,* vol. 40, p. 595-606, DOI:10.1007/s00170-007-1360-x.

[18] Khoogar, A.R., Tehrani, A.K., Tajdari, M. (2010). A dual neural network for kinematic control of redundant manipulators using input pattern switching. *Journal of Intelligent & Robotic Systems,* vol. 60, p. 1-13.

[19] Ghasemi, A., Eghtesad, M., Farid, M. (2010). Neural network solution for forward kinematics problem of cable robots. *Journal of Intelligent & Robotic Systems,* vol. 60, p. 201-215, DOI:10.1007/s10846-010-9421-z.

[20] Arisariyawong, S., Charoenseang, S. (2001). Reducing steady-state errors of a direct drive robot using neurofuzzy control. *Second Asian Symposium on Industrial Automation and Robotics BITEC,* p. 212-216.

[21] Šafarič, R., Jezernik, K., Pec, M. (1998). Neural network control for direct-drive robot mechanisms. *Engineering application of artificial intelligence,* vol. 11, no. 6, p. 735-745.

[22] Klobučar, R., Čas, J., Šafarič, R., Brezočnik, M. (2008). Uncalibrated visual servo control with neural network. *Strojniški vestnik – Journal of Mechanical Engineering*, vol. 54, no. 9, p. 619-627.

[23] Čas, J., Škorc, G., Šafarič, R. (2010). Neural network position control of XY piezo actuator stage by visual feedback. *Neural Computing & Applications*, vol. 19, no. 7, p. 1043-1055, DOI:10.1007/s00521-010-0339-y.

[24] Čas, J., Škorc, G., Šafarič, R. (2010). Improved micropositioning of 2 DOF stage by using the neural network compensation of plant nonlinearities. *Strojniški vestnik – Journal of Mechanical Engineering*, vol. 56, no. 10, p. 599-608.

[25] Cichocki, A., Unbehauen, R. (1993): *Neural Netwok for Optimal Signal processing.* John Wiley&Sons, Chichester.