ISSN 0005–1144
ATKAAF 48(3–4), 169–181 (2007)

Andreja Rojko, Riko Šafarič, Karel Jezernik

Applied Soft Computing Robot Motion Control

UDK 004.896:681.513.3
IFAC 4.7;2.2

Preliminary communication

This paper considers the problem of the robot motion control in the presence of the major uncertainties such as it is varying load. Efficiency of one conventional and two soft computing model based control algorithms is investigated and compared trough the results of application on a direct drive robot. First control algorithm is a conventional computed torque based on the Lagrangian dynamic equations. Second method is a computed torque alike control with an adaptive fuzzy logic system that replaces Lagrangian model, and third is a continuous sliding mode control with an artificial neural network instead of the dynamic model. Both soft computing methods give excellent results, while inefficiency of the computed torque control confirms the disadvantages of the conventional model based motion control approaches.

Key words: adaptive fuzzy logic system, computed torque control, neural network, robot control, sliding mode

1 INTRODUCTION

Up to date the robot manipulators are employed in many demanding technological processes and therefore the robot motion control design still represents a challenging task. There are many sorts of the control algorithms to choose between and some of them require exact dynamical model of the system; those are so called model based control algorithms. The conventional model based motion control algorithm is well known computed--torque method (CT), [14], and many other controllers were obtained by the modification of this original feedback-linearization based algorithm. Also other control methods, as for example variable structure control [12] require dynamic model of the mechanism. For good efficiency of those and others model based control approaches, the realistic mathematical model based on physical differential equations of the second order is required; that is the model which with its structure and parameters matches the structure and parameters of the real robot dynamics with all influential inertia, gravitation, friction, Coriollis and centrifugal effects. But even if the good model is available, also stable conditions must be guaranteed for the model based control to be efficient. Those conditions include invariable, exactly known robot load and stable environment factors especially a temperature, because variation in the temperature can significantly influence the friction parameters. However varying and unknown load is present in the most robot applications. This problem limits the applicability of the conventional model based control to very few practical applications or to the robots with substantial gear reduction ratios.

Lately soft computing approaches based on adaptive fuzzy logic systems (FLS) or artificial neural networks (NN) are used a lot to replace dynamic model in the model based control algorithms. Both mathematical structures have ability to approximate any nonlinear function on a compact set and adapt its parameters by learning. However their complexity and therefore high computational demands are probably the reason why in the literature only a few real time applications of adaptive FLSs and NNs in the robot motion control can be found and the most works to the date present only extensive simulation results [1, 6, 13, 17].

In this paper the application results of three model based control algorithms are compared. Two of them are based on the soft computing models. All algorithms are applied to motion control of direct drive robot, so it was possible to directly compare controllers' efficiency as well as other important properties like pretentiousness of the parameter setting and computational requirements. The efficiency was tested with the movements where the robot load is varying. Varying load is the biggest

possible uncertainty and nonlinearity that can appear in the robot dynamics and therefore a very good indicator of the quality of the control algorithms.

This paper is organized as follows. In section 2 direct drive robot manipulator is described and robot motion control problem is stated. Section 3 describes CT control and gives implementation results. Section 4 describes design of CT alike adaptive FLS. Application results are shown and discussed. Section 5 describes design of a continuous sliding-mode controller with NN. Section 6 gives comparison of the methods by considering different aspects. Section 7 gives some conclusions.

2 DIRECT DRIVE ROBOT

The test plant was in-house designed and built direct-drive (DD) robot manipulator Puma configuration with three degrees of freedom shown in Figure 1. Robot dynamic model and basic construction plan are available at the web page [11]. Detailed description of the robot and controller is given in [16]. The mechanical construction of the DD robot mechanism is simplified by the use of Yokogawa Dynaserv direct drive AC-motors (maximal torques are 220 Nm, 160 Nm and 60 Nm). Here expensive gear-boxes are not needed and consequently no back-lash error is presented. On the other hand, the dynamic model equations for DD robot mechanism reveal especially high non-linearity, because the influences of the mechanism on the AC-motors are direct and not reduced by the high gear ratios. Actual positions of robot joints are measured by the resolvers. Joint velocities are not measured, but calculated from the positions. Robot controller is transputer based multiprocessor system and enables low sampling times and parallel execution of the controller's task.

The dynamic model of a rigid DD robot mechanism, with m degrees of freedom described with the Lagrange equation of motion is as follows:

$$T = M(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + G_f(\theta) + F(\dot{\theta}) + T_n. \quad (1)$$

Here T is a vector of the drive torques on robot's joints, M is inertia matrix, h is a torque vector due to the centrifugal forces and Coriollis forces, G_f is a torque vector due to forces of the gravity, F is a torque vector due to the friction forces, and T_n stands for a torque vector due to the other, unknown disturbances. $\theta = [\theta_1,\theta_1,...\theta_m]^T \in \Re^m$, $\dot{\theta} \in \Re^m$ and $\ddot{\theta} \in \Re^m$ are vectors of actual positions, velocities and accelerations. This well known mathematical note of robot mechanism dynamics (1) can be also expressed with an n-dimensional state-space system of equations with regard to the control value u as:

$$\dot{x} = f(x,t) + B(x,t)u + d(x,t). \quad (2)$$

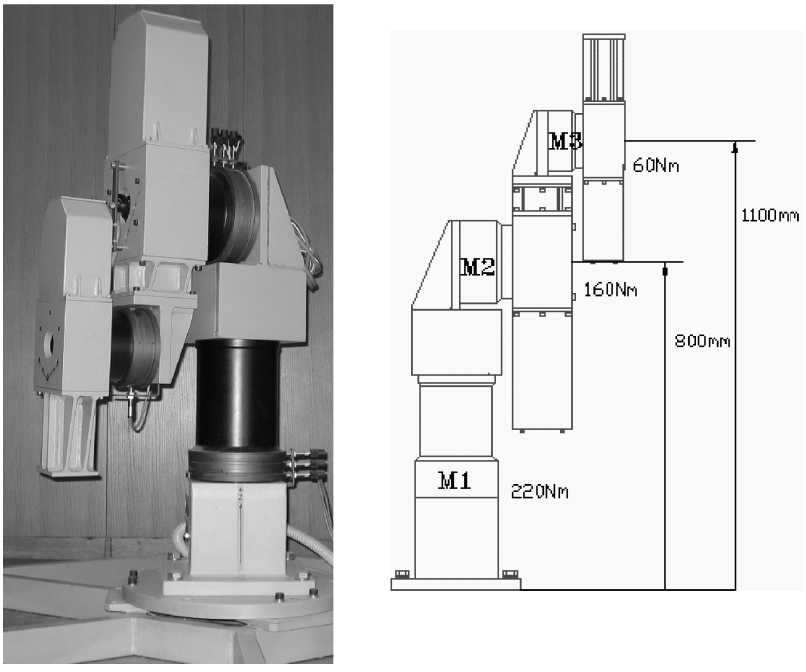

Fig. 1 Direct drive robot

t stands for time, $x \in \Re^n$ is a state space vector, $u \in \Re^m$ is a control vector, $d(x,t)$ is unknown disturbance, B is an actual input matrix, defined as:

$$B(x,t) = \tilde{B}(x,t) + \Delta B(x,t), \quad (3)$$

where $\tilde{B}(x,t)$ is estimated input matrix and $\Delta B(x,t)$ is discrepancy between estimated and actual input matrix.

Next the problem of the motion control must be stated. For the robot described with dynamic equations (1) or (2) this problem can be specified as the problem of finding the control torques T in (1) so that the equilibrium point $e = 0$ defined with (4) is globally asymptotically stable.

$$e = \left[e_\theta^T(t), \dot{e}_\theta^T(t) \right]^T \in \Re^{2m}. \quad (4)$$

In (4) the error vector components are:

$$\dot{e}_\theta = \left[\dot{\theta}_{r,1}(t) - \dot{\theta}_1(t), .., \dot{\theta}_{r,m}(t) - \dot{\theta}_m(t) \right]. \quad (5)$$

The reference trajectory is a smooth function prescribed with the position $\theta_r(t) \in \Re^m$, velocity $\dot{\theta}_r(t) \in \Re^m$ and acceleration vectors $\ddot{\theta}_r(t) \in \Re^m$.

3 COMPUTED TORQUE CONTROL

The conventional CT control [14] is a straightforward control scheme where feedback linearization is achieved by applying the complete robot dynamic model. The driving torques are calculated as:

$$T = M(\theta)\ddot{\theta}^C + h(\theta,\dot{\theta}) + G_f(\theta) + F(\dot{\theta}) + T_n, \quad (6)$$

where $\ddot{\theta}^C$ is calculated acceleration, defined as:

$$\ddot{\theta}^C = K_p e_\theta + K_v \dot{e}_\theta + \ddot{\theta}_r. \quad (7)$$

If the dynamic model is known fairly accurately, the closed loop error dynamics becomes:

$$\ddot{e}_\theta + K_p e_\theta + K_v \dot{e}_\theta = 0. \quad (8)$$

K_p and K_v are $m \times m$ diagonal matrixes of the velocity and position gains and are usually selected for critical damping $D = 1$ (9). $\omega_{n,k}$ is natural frequency of k-th joint.

$$K_{p,k} = \omega_{n,k}^2, \quad K_{v,k} = 2D\omega_{n,k}, \quad k = 1 \ldots m. \quad (9)$$

A. Experimental results

In experiment applied CT control law is given with (6), (7) and applied dynamical model is from [11]. Dynamic model was completed with the measured friction that includes joint velocity dependent viscous and Coulomb friction in form:

$$T_{n,k} = b_k \cdot \dot{\theta}_k + L_{f,k} \cdot sign(\dot{\theta}_k). \quad (10)$$

b_k are viscous friction coefficients, $L_{f,k}$ are Coulomb friction coefficient with the following values

$b_{(k = 1,2,3)}$ = [1.04, 4.05, 0.35] Nms,

$L_{f,(k = 1,2,3)}$ = [3.38, 3.30, 2.06] Nm.

Applied gains were

$Kp_{(k = 1,2,3)}$ = [100, 300, 900] and

$Kp_{(k = 1,2,3)}$ = [20, 35, 60].

Position gains were set to high values in order to achieve faster response for the joints with lower masses and to lower values for the heaviest first robot joint. Velocity gains were selected for the critical damping, equation (9). Robot tip's position error is used to measure the quality of the motion control:

$$e = \left[(X_{ri} - X_i)^2 + (Y_{ri} - Y_i)^2 + (Z_{ri} - Z_i)^2 \right]^{1/2}. \quad (11)$$

Here X_{ri}, Y_{ri}, Z_{ri} 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 task space.

Test reference trajectory was point-to-point movement same for all joints, Figure 2. The load with mass 4.7 kg has been attached and released few times, when the robot joints were already at the standstill. Robot tip's position error is shown in Figure 3. Very high tracking and positioning error is present from which it can be concluded that the accuracy of the mechanism's dynamic model is not sufficient, although a lot of effort was put into deriving of the accurate Lagrangian dynamic model. Error at the standstill shows that even gravity is not sufficiently compensated. But also when exact model would be available, the problem of varying

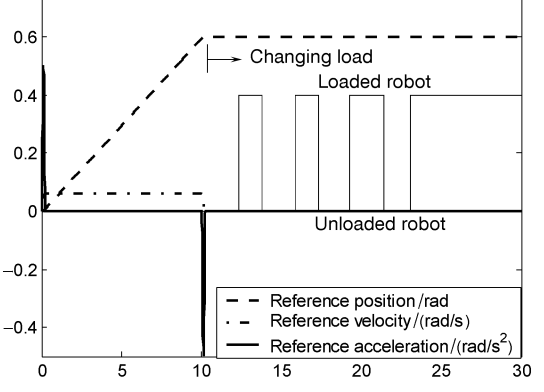

Fig. 2 All controllers: Reference point-to point trajectory

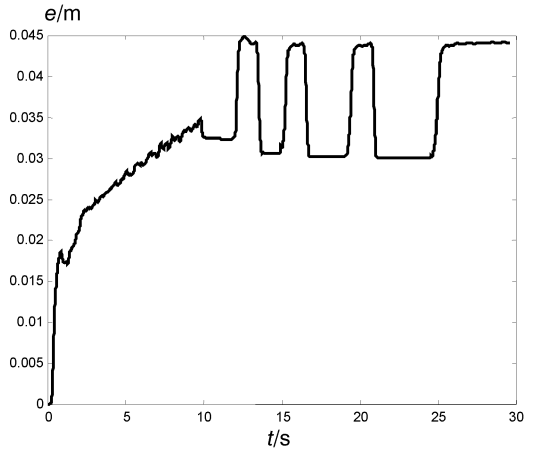

Fig. 3 Computed torque control: Robot tip's position error during load change

load would still exist. Because the model parameters in CT control are fixed, it is impossible to compensate for the load variations. Accordingly, Figure 3 shows that in the case of the additional load, the robot tip's position error is increased and stays high until the load is dropped.

4 MODIFIED COMPUTED TORQUE CONTROL WITH FUZZY LOGIC SYSTEM

Next, design of CT alike control, where dynamic model is replaced by an adaptive FLS, is described. Simplest strategy for developing FLS, which can be considered, is view on the robot as powered by the number of independent drives that are controlled separately, as a set of simple single input-single output systems. The mutual interactions among the robot joints that appear due to varying configurations during the robot motion are treated as the disturbances. Decentralization of the control algorithm helps to keep the system's complexity manageable, which is strong argument for choosing this approach. Especially in the case of adaptive FLS, that can easily become very complex and computational extensive, this is most welcomed. Those arguments are strong enough to make this worth an attempt, although the highly coupled DD robot dynamics stirs up doubts about the suitability of the decentralized approach. To facilitate the derivation of this control scheme, (1) is rewritten for the k-th robot joint, $k = 1...m$, as

$$T_k = \bar{M}_{kk}(\theta)\ddot{\theta}_k + \Delta M_{kk}(\theta)\ddot{\theta}_k + \sum_{j=1, j\neq k}^{m} M_{kj}(\theta)\ddot{\theta}_j + \\ + \sum_{j=1}^{m}\sum_{l=1}^{m} h_{jl,k}(\theta)\dot{\theta}_j\dot{\theta}_l + G_f(\theta) + F_k(\dot{\theta}) + T_{nk} \tag{12}$$

where $\bar{M}_{kk}$ is the constant part of inertia matrix, $\Delta M_{kk}(\theta)$ is the variable part of the inertia, and $M_{kj}(\theta)$ are the coupling inertias. Let us denote the whole dynamics of the robot joints (12) with exception of the constant part of the inertias as the disturbance term $w_k(\theta,\dot{\theta},\ddot{\theta})$:

$$w_k(\theta,\dot{\theta},\ddot{\theta}) = \Delta M_{kk}(\theta)\ddot{\theta}_k + \sum_{j=1, j\neq k}^{m} M_{kj}(\theta)\ddot{\theta}_j + \\ + \sum_{j=1}^{m}\sum_{l=1}^{m} h_{jl,k}(\theta)\dot{\theta}_j\dot{\theta}_l + G_f(\theta) + F_k(\dot{\theta}) + T_{nk}. \tag{13}$$

By using (13) in (12), the model of the k-th robot's joint with the joint drive torque as an input can be rewritten as:

$$T_k = \bar{M}_{kk}(\theta)\ddot{\theta}_k + w_k(\theta,\dot{\theta},\ddot{\theta}). \tag{14}$$

Applying CT control approach (6), (7) to the robot with joint dynamics (14), gives:

$$T_k = \bar{M}_{kk}\ddot{\theta}_k^c + \hat{w}_k(\theta,\dot{\theta},\ddot{\theta}). \tag{15}$$

$\hat{w}_k$ is estimation of disturbance term w_k (13) and will be estimated by adaptive FLS. The error dynamics of robot joint may be now written in the following form, [3]:

$$\ddot{e}_{\theta,k} + K_{p,k}e_{\theta,k} + K_{v,k}\dot{e}_{\theta,k} = \\ = \bar{M}_{kk}^{-1}\left[\ \hat{w}_k(\theta,\dot{\theta},\ddot{\theta}) - w_k(\theta,\dot{\theta},\ddot{\theta})\ \right]. \tag{16}$$

This implies that if following is fulfilled:

$$\lim_t\left[\ \hat{w}_k(\theta,\dot{\theta},\ddot{\theta}) - w_k(\theta,\dot{\theta},\ddot{\theta})\ \right] = 0, \tag{17}$$

the control (15) decouples and linearizes the system, as all nonlinear terms are completely cancelled and consequently good performance can be expected.

A. Design of decentralized FLS model

Because decentralized control approach has been chosen, also adaptive FLS for estimation of the dynamics should be decentralized. Additionally decentralized FLS is easier to design, more transparent and less complex then centralized system. Applying complete decentralized approach to control (15) gives:

$$T_k = \bar{M}_{kk}\ddot{\theta}_k^c + \hat{w}_k(\theta_k,\dot{\theta}_k,\ddot{\theta}_k). \tag{18}$$

The discrepancy from (15) is that new decentralized disturbance term $\hat{w}_k(\theta_k,\dot{\theta}_k,\ddot{\theta}_k)$ is introduced. Now it depends only on the position, the

velocity and the acceleration of the single joint, while information about states of the other robot joints is not included. In the sequel of this subsection the design of the corresponding FLS is described.

First the inputs in FLS have to be chosen. The obvious choice is three element vector $x_k = \left[\theta_k, \dot{\theta}_k, \ddot{\theta}_{r,k}\right]$ that includes the actual position, velocity and desired, reference acceleration of the k-th robot joint. Reference acceleration is used instead of the unknown actual acceleration. The fuzzy rule base designed for the k-th robot joint now consist of IF-THEN rules R_k^l with the following general form:

$$\text{IF} \quad \theta_k = X_k^{\theta,l} \quad \text{AND} \quad \dot{\theta}_k = X_k^{\dot{\theta},l} \quad \text{AND} \quad \ddot{\theta}_k^d = X_k^{\ddot{\theta}_r,l} \quad \text{THEN} \quad \hat{w}_k = \bar{y}_k^l. \tag{19}$$

Superscript l refers to the l-th rule $l = 1...M$. $X_k^{\theta,l}, X_k^{\dot{\theta},l}, X_k^{\ddot{\theta}_r,l}$ are input fuzzy sets and $\bar{y}_k^l$ are the positions of the output singleton fuzzy sets. The following structure of the FLS is applied: singleton output membership functions (MF), singleton fuzzifier, product-operation rule of fuzzy implication and center of average defuzzifier. For the sake of making the FLS nonlinear in its nature, bell shaped function form was chosen for the input MFs. The output of the resulting FLS is given by (20) and (21).

$$\hat{w}_k = \frac{\sum_{l=1}^{M} \bar{y}_k^l \prod_{i=1}^{n} \mu_{R_k^{i,l}}(x_k^i)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \mu_{R_k^{i,l}}(x_k^i)}, \tag{20}$$

$$\mu_{R_k^{i,l}}(x_k^i) = \frac{1}{1 + \left| \frac{x_k^i - \bar{x}_k^{i,l}}{\sigma_k^{i,l}} \right|^{2 \cdot b_k^{i,l}}}, \tag{21}$$

where $\bar{x}_k^{i,l}$ are the centers of the input MFs, $\sigma_k^{i,l}$ determine the width of the bell MFs and $b_k^{i,l}$ the slope of each MF. x_k^i refers to the i-th element of the vector of the input variables.

After choosing FLS's structure, corresponding parameters have to be assigned to it. First set of the parameters are input MFs' parameters, which are parameters concerning conditional (IF) part of the rules. This includes: MFs' number, MFs' positions $\bar{x}_k^{i,l}$ and parameters that determine their shape $(\sigma_k^{i,l}, b_k^{i,l})$. The interval that MFs should occupy must include all possible values that might appear in the inputs and it is conditioned by the robot construction. Consequently the parameters concerning conditional (IF) part of the rules can be fixed. This topic will be also addressed later in the section with experimental results.

The next step is identification of the parameters belonging to consequent (THEN) part of the rules, $\bar{y}_k^l$. Those parameters are chosen to be adaptive, so that the variations in the robot dynamics (changing load, friction parameters) can be estimated. In order to derive the adaptation law, well known property of the robot dynamics is applied: dynamic model of the rigid mechanism can be described as the linear combination of the parameter vector and nonlinear functions called base functions, [3]. FLS (20) can be written in that form after introducing the parameter vector $\hat{P}_k = \left[\bar{y}_k^1, \ldots, \bar{y}_k^M\right]^T$, with all adaptive parameters and vector of the nonlinear functions $\xi_k(x_k) = \left[\xi_k^1(x_k), \ldots, \xi_k^l(x_k), \ldots, \xi_k^M(x_k)\right]^T$ defined as:

$$\xi_k^l = \left(\frac{\prod_{i=1}^{n} \mu_{R_k^{i,l}}(x_k^i)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \mu_{R_k^{i,l}}(x_k^i)} \right). \tag{22}$$

By using (22), the FLS (20) can be written in the parameter vector-regressor form, often used in the conventional theory of the system identification [8] as:

$$\hat{w}_k = \sum_{l=1}^{M} \bar{y}_k^l \cdot \varphi_k^l(x_k) = \hat{P}_k^T \cdot \xi_k(x_k). \tag{23}$$

Rewritten joint error dynamics (16) is:

$$\ddot{e}_{\theta,k} + K_{v,k}\dot{e}_{\theta,k} + K_{p,k}e_{\theta,k} = = \bar{M}_{kk}^{-1}(\hat{P}_k^T - P_k^T)\xi_k(x_k) = \bar{M}_{kk}^{-1}\tilde{P}^T\xi_k(x_k), \tag{24}$$

where $\tilde{P} = \hat{P} - P$ is the parameter vector error. By introducing vector

$$v = [0,1]^T \text{ and matrix } \Lambda = \begin{bmatrix} 0 & 1 \\ -K_{p,k} & -K_{v,k} \end{bmatrix}$$

in the error equation (24), following is obtained:

$$\dot{e} = \Lambda \cdot e + \bar{M}_{kk}^{-1} \cdot \mathbf{v} \cdot \tilde{P}^{\boldsymbol{T}} \cdot \boldsymbol{\xi}_k(\mathbf{x}_k). \tag{25}$$

This error equation falls into the class of 'Error model 2', as defined by [9]. Suggested adaptive law is:

$$\dot{\hat{P}} = -\pm f_e \boldsymbol{\xi}(\mathbf{x}) \tag{26}$$

where a new term for linear combination of the position and the velocity error is introduced as:

$$fe_k = a_{1,k} \cdot e_k + a_{2,k} \cdot \dot{e}_k. \tag{27}$$

The adaptation law (27) guarantees the global asymptotic stability if following conditions (28) and (29) are satisfied:

$$a_{1,k} \cdot K_{p,k} > 0 \tag{28}$$

$$a_{2,k} > \frac{a_{1,k}}{K_{v,k}}. \tag{29}$$

$a_{1,k}$ and $a_{2,k}$ are positive parameters, α is adaptation rate. At this point one additional topic concerning FLS's design must be considered. It is known that FLSs often suffer from the problem of the complexity because its size is an exponential function of the number of its inputs [18]. Suggestion from [10] for improving the interpretability and lowering the complexity is to use three sets of the rules, where each set has different number of the inputs.

For part of the rules only the position and the acceleration are used in input vector

$$\mathbf{x}_{k,1.SET} = \left[q_k, \ddot{q}_k^d \right].$$

Those rules are local dynamic models for estimation of the varying part of the torque which is caused by the inertia and is mathematically a product of the position dependent inertia and acceleration of the robot joint. Second set of the rules has as input position and velocity,

$$\mathbf{x}_{k,2.SET} = [q_k, \dot{q}_k]$$

and it is designed to estimate Coriollis, centrifugal forces and velocity dependent friction. Third set of rules has as inputs all input vector's elements

$$\mathbf{x}_{k,3.SET} = \mathbf{x}_k$$

and compensates the rest of the robot's dynamics.

B. Experimental results

Applied control scheme is shown in Figure 4. CT alike control law is given with (18), FLS with (20), (21) and adaptation law with (26), (27). In previous subsection described rule optimization is applied. Table 1. shows implemented rule base. Positions of MFs were set by considering following; robot has joint limit switches at the positions and maximum joint's velocity is limited to 6 rad/s. Maximum possible acceleration depends on the robot load, but most used values are under 50 rad/s^2. Accordingly three MFs were implemented, two with the open sides, so that the whole interval of

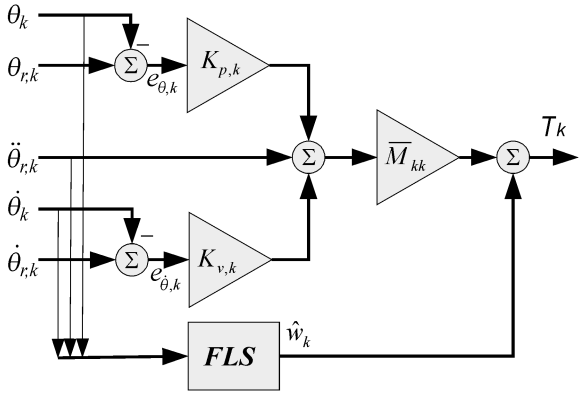

Fig. 4 CT alike control with FLS: Control scheme

Table 1 **CT alike control with FLS: Rule base**

	Rule	Position	Velocity	Acceleration
1. subsystem	R^1	negative	–	negative
	R^2	zero	–	zero
	R^3	positive	–	positive
2. subsystem	R^4	negative	negative	–
	R^5	zero	zero	–
	R^6	positive	positive	–
3. subsystem	R^7	positive	zero	zero
	R^8	negative	zero	zero
	R^9	zero	negative	positive
	R^{10}	zero	zero	zero
	R^{11}	zero	zero	positive
	R^{12}	zero	negative	negative
Not applied on 1. joint	R^{13}	positive	zero	positive
	R^{14}	negative	negative	negative
	R^{15}	positive	positive	positive

possible input values is covered. MFs are shown in Figure 5. MFs for the position and velocity are equally distributed. The initial values of the adaptive parameters, respectively the positions of the singleton output MFs, were set to very small random values. With using small initial values, the FLS outputs can also have only small values. Therefore the possibility of the instability due to randomly chosen initial values in the first moments is avoided. Another possibility would be to carefully consider the rules and determine proper initial values of each belonging adaptive parameter from known dynamic model. However this would cost a lot of work. Further the variable robot load would not be considered. Position gains were set to $Kp_{(k=1,2,3)} = [1000, 2400, 1200]$, velocity gains to $Kv_{(k=1,2,3)} = [64, 98, 70]$ and parameters of average inertia matrix to $\overline{M} = \text{diag}[3.5, 2.5, 0{,}13]$ kgm^2. The controller parameters satisfy the stability condition (28), (29). The initial values of the average inertia matrix were calculated by using known dy-

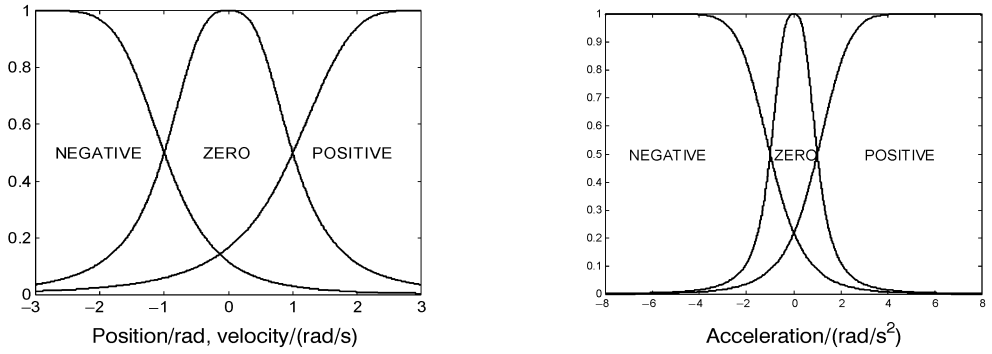

Fig. 5 CT alike control with FLS: Membership functions

namic model [11] and point-to-point movement same for all robot joints. Those values were in practice increased, because better results without loosing the stability were obtained. Position and velocity gains were set by increasing gains applied in the conventional CT method to the point where the position error was small but no problems with the instability appeared. At the procedure number of different reference movements were used. Adaptation parameters were set to $a_{k=1} = [44, 0.7]$ for the first robot joint, to $a_{k=2} = [50, 1]$ for the second and to $a_{k=1} = [7, 0.2]$ for the third robot joint and for all joints $\alpha_{k=1,2,3} = 1$ were used. Small values of the second parameter of a_k for each joint were chosen, therefore the velocity error has lower influence on the adaptation then the position error. This was necessary, because the velocity signals are contaminated with a lot of noise that could cause problems at the parameter adaptation stability. The reference trajectory is shown in Figure 2. Again variable load of 4.7 kg is applied. Figure 6 shows the robot's tips position error, equation (11). The top tracking error during the movement at the acceleration phase is less then 3 mm. There is less then 0.1 mm steady state position error and about 0.3 mm tracking error in the part of movement with the zero acceleration and constant velocity.

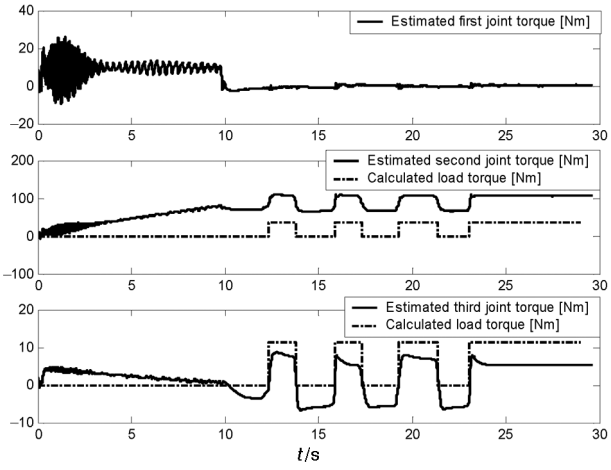

Fig. 7 CT alike control with FLS: Estimated total joint torques and calculated load torques

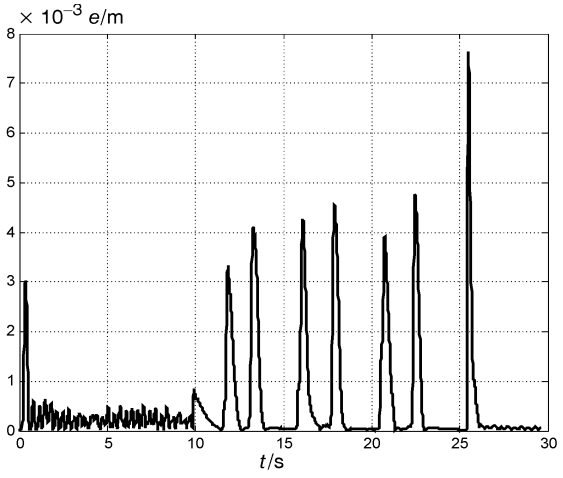

Fig. 6 CT alike control with FLS: Robot tip's position error during load changes

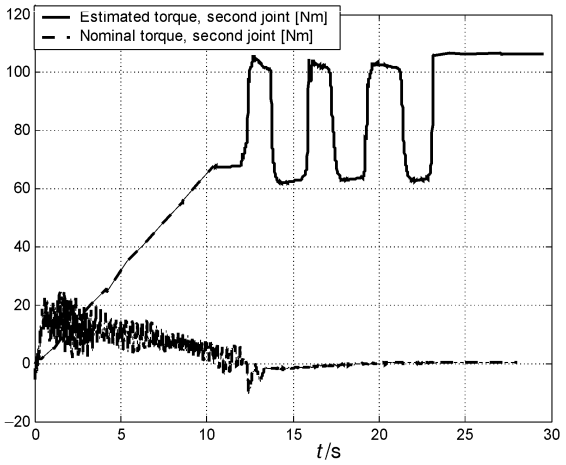

Fig. 8 CT alike control with FLS: Estimated torque $\hat{w}_2$ and nominal torque $\bar{M}_{22}\ddot{\theta}_2^c$ for second robot joint

When the load was attached and released, the increase in the position error can be observed, but it falls rapidly to the initial value without the stability problems. Note that the load changes were performed manually, so there were different forces strokes present each time and the resulting peak position error is also different at every change. Figure 7 shows torques applied at the each robot joint. At the second and the third joint that under influence of gravitation the load changes causes step-like fluctuations. The figure also shows from the Lagrangian model [11] calculated load torques. Figure 8 shows nominal and by the FLS estimated part of the torque for the second robot joint. Most of the torque is estimated by the FLS, which filters out the signal noise, so that its output signal is smooth. In the nominal torque signal oscillations caused by the measurement noise are present.

5 SLIDING MODE CONTROLLER WITH NEURAL NETWORK ESTIMATION

The main advantages of the sliding mode (SM) control are the robustness to the parameter uncertainty, to the load disturbance and fast dynamics response [12, 15]. However, these properties are valid on the sliding surface when no modeling imprecision, external disturbances and switching time delays are present. Practical SM control implementations exhibit high frequency oscillations in the plant output, called chattering. This may excite high frequency dynamics on the sliding surface that was neglected in the dynamic model. A number of methods have been suggested to alleviate the chattering effects and improve precision in the SM control of robot mechanism. In [4] and [5], the authors developed perturbation estimation schemes, others suggested to supplement the control input with a predictive correction term [7]. Here a two layer NN for estimation of most of the robot dynamics, with exception of average inertia matrix, is applied. Two layer NN is a proper choice to estimate dynamics, as it was shown that it possess good approximation abilities the same as adaptive FLS, [18].

A. Design of continuous neural network sliding-mode controller

For mechanism with dynamics (1) that is transformed into an n-dimensional state-space system (2) and (3) switching function (30) is chosen. The goal is to design the control law with the help of Lyapunov theory for the robot system (1), (2).

$$\sigma(x,t) = G \cdot [x(t) - x_r(t)]. \tag{30}$$

This means that after transient time, defined with elements of the matrix G, the difference between the actual and the desired vector of state, space variables x and reference state space variables x_r should equal zero and will be stable for all disturbances. Function $\sigma(x,t) = 0$ will be stable if the Lyapunov function is positive $V > 0$ and its first time derivative is negative $\dot{V} < 0$. The simplest V for guaranteeing $V > 0$ for whichever chosen x_r, x, G is (31) and its derivative is (32).

$$V = \sigma^T \cdot \frac{\sigma}{2} \tag{31}$$

$$\dot{V} = \sigma^T \cdot \dot{\sigma}. \tag{32}$$

Next the suitable conditions for control law u, where the robot system will be stable, have to be determined. This is done as follows. Because $\dot{V}$ is not always less than zero for all x_r, x, G, the first desired Lyapunov function time derivative has been defined as:

$$\dot{V} = -\sigma^T \cdot D \cdot \sigma \tag{33}$$

where D is a diagonal matrix with positive elements. Further, if the definition (33) and the derivative of Lyapunov function (32) are made equal, the result is:

$$\sigma^T \cdot (D \cdot \sigma + \dot{\sigma}) = 0. \tag{34}$$

The equation (34) is valid if both or at least one of the multiplicators equals zero. Since the first multiplicator, the term σ^T, does not equal zero during the transient response, the control law can be calculated on the basis of the second multiplicator, which is:

$$D \cdot \sigma + \dot{\sigma} = 0. \tag{35}$$

Further if (30) is differentiated and (2), (3) are inserted into the calculated derivative, the following is obtained:

$$\dot{\sigma} = G \cdot (f + \tilde{B} \cdot u + \Delta B \cdot u + d - \dot{x}_r). \tag{36}$$

Next (36) is inserted into the implementation condition of the control law (35) to obtain:

$$u(t) = -(G \cdot \tilde{B})^{-1} \cdot \left[G \cdot (f + \Delta B \cdot u + d - \dot{x}_r) + D \cdot \sigma \right]. \tag{37}$$

Since the term $(f + \Delta B \cdot u + d)$ is unknown and not measurable, it is approximated by the neural network N:

$$u(t) = -(G\tilde{B})^{-1} \cdot [G \cdot (N(t) - \dot{x}_r(t)) + D\sigma(t)]. \tag{38}$$

A conventional supervised learning of NN cannot be used, since the target values are not known. Therefore an on-line estimator has been developed for estimating of the learning signal which is the difference between the target and the output of NN. Next the exact stability condition will be derived. From differentiated equation (30) the derivative of the state vector can be expressed as:

$$\dot{x} = G^{-1} \cdot \dot{\sigma} + \dot{x}_r. \tag{39}$$

After (38) and (39) have been inserted into the basic equation of the mechanism dynamics (2), the result is:

$$\dot{\sigma} + D \cdot \sigma = G \cdot (f + \Delta B \cdot u + d) - G \cdot N = G(Z - N), \tag{40}$$

where it was substituted $Z = f + \Delta B \cdot u + d$. By using the derivative of Lyapunov function and the equation (40), the condition, where the system controlled with the derived control law (37) remains stable, is:

$$\dot{V} = \sigma^T \cdot \dot{\sigma} =$$

$$= \sigma^T \cdot G \cdot (f + \Delta B \cdot u + d - N) - \sigma^T \cdot D \cdot \sigma < 0. \tag{41}$$

The next condition, equation (42), has been derived from (30) and (41). To make $\dot{V} < 0$ and consequently $\sigma \to 0$ possible, the condition (42) has to be fulfilled in time during which NN is approximating the unknown part of robot dynamics ($f + \Delta B \cdot u + d$).

$$\left| G \cdot (f + \Delta B \cdot u + d) - G \cdot N \right| = \left| D \cdot \sigma + \dot{\sigma} \right| < \left| D \cdot \sigma \right|. \tag{42}$$

NN with two layers shown in Figure 9 is applied. To teach output layer, a modified backpropagation (BPG) rule has been used and for the hidden layer traditional BPG rule is used [2]. The hidden layer output is calculated as (43) and the output from an outer layer, that is an output from the NN, is calculated as (44).

$$net_j = \sum_l w_{jl} \cdot v_l, \quad o_j = g(net_j), \tag{43}$$

$$net_i = \sum_j w_{ij} \cdot o_j, \quad N_i = o_i = g(net_i), \tag{44}$$

j is number of the neurons in the hidden layer, i is number of the neurons in the output layer, v is NN's input vector, l is number of the NN's inputs, w_{jl} is matrix of the hidden layer weights, w_{ij} is matrix of the output layer weights, o_j is hidden layer output, o_i is NN's output. Transfer function of the hidden layer is nonlinear sigmoid function (45) and transfer function of the output layer is linear (46).

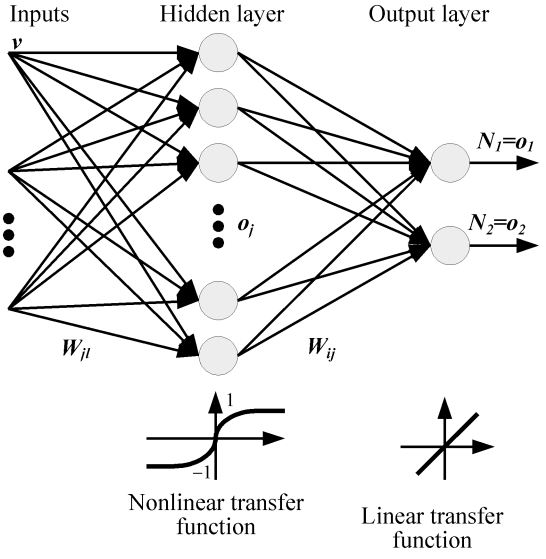

Fig. 9 SM control with NN: Neural network with two layers

$$g(net_j) = 1 - \frac{2}{1 + e^{-net_j}} \tag{45}$$

$$g(net_i) = net_i. \tag{46}$$

Change of the weights for the output layer is calculated as:

$$\Delta w_{ij} = \frac{\varepsilon \cdot \partial E}{\partial w_{ij}}, \tag{47}$$

where the error E is chosen to fulfill (35):

$$E = \frac{(D \cdot \sigma + \dot{\sigma})^T \cdot (D \cdot \sigma + \dot{\sigma})}{2} =$$

$$= \frac{[G \cdot (Z - N)^T] \cdot [G \cdot (Z - N)]}{2} \tag{48}$$

The change of weights can be now expressed as (49).

$$\Delta w_{ij} = \varepsilon \cdot \partial E / \partial net_i \cdot \partial net_i / \partial w_{ij} = \varepsilon \cdot \partial E / \partial net_i \cdot o_j =$$

$$= \varepsilon \cdot \partial E / \partial o_i / \partial o_i / \partial net_i \cdot o_j \Rightarrow$$

$$\Delta w_{ij} = \varepsilon \cdot \partial E / \partial o_i \cdot g'(net_i) \cdot o_j, \tag{49}$$

where from (46) $g'(net_i) = 1$ follows. The derivative $\partial E / \partial o_i$ from (49) can be calculated as:

$$\partial E / \partial o_i = \partial / \partial o_i \left[(G \cdot Z - G \cdot N)^T \cdot (G \cdot Z - G \cdot N) \right] / 2$$

$$= \partial / \partial o_i \left[(G \cdot N)^T \cdot (G \cdot Z - G \cdot N) \right] \Rightarrow$$

$$\partial E / \partial o_i = \partial / \partial o_i \left[(G \cdot N) \right]^T \cdot (D\sigma + \dot{\sigma}). \tag{50}$$

B. Decentralized control law for three degrees of freedom mechanism

Expression for robot dynamics, equation (14) is written to facilitate the derivation of decentralized control law as:

$$T_k = \bar{M}_{kk}(\theta)\ddot{\theta}_k + w_k(\theta_k, \dot{\theta}_k, \ddot{\theta}_k). \tag{51}$$

The equation (51) for the k-th joint can be written as:

$$\dot{x}_k = f_k + \Delta B_k \cdot u_k + \tilde{B}_k \cdot u_k + d_k, \tag{52}$$

where following expression are used:

$$x_k = \left[\theta_k, \dot{\theta}_k\right]^T, \ \dot{x}_k = \left[\dot{\theta}_k, \ddot{\theta}_k\right]^T, \tag{53}$$

$$f_k = \begin{bmatrix} \dot{\theta}_k \\ -\hat{\bar{M}}_{kk}^{-1} \cdot \hat{w}_k \quad (\theta_k, \dot{\theta}_k, \ddot{\theta}_k) \end{bmatrix}$$
$$\text{and} \quad \tilde{B}_k = \begin{bmatrix} 0 \\ -\hat{\bar{M}}_{kk}^{-1} \end{bmatrix}. \tag{54}$$

The dimensions of the vectors $f_k, B_k, \tilde{B}_k$ are 2 by 1. The control law u_k is described by the following equation:

$$u_k(t) = \\ = -(G_k \cdot \tilde{B}_k)^{-1} \cdot \left[G_k \cdot (N_k(t) - \dot{x}_{r,k}(t)) + d_k \sigma_k(t)\right]. \tag{55}$$

Here $G_k = \left[K_{p,k} \quad K_{v,k}\right]$ is vector of gains and $x_{r,k} = \left[\theta_{r,k}, \dot{\theta}_{r,k}\right]^T$ and $\dot{x}_{r,k} = \left[\dot{\theta}_{r,k}, \ddot{\theta}_{r,k}\right]^T$ are reference vectors. N_k with dimension 2 by 1 represents the NN's output. Switching function and its time derivative are:

$$\sigma_k(t) = G_k\left[x_k(t) - x_{r,k}(t)\right], \\ \dot{\sigma}_k = G_k\left[\dot{x}_k(t) - \dot{x}_{r,k}(t)\right]. \tag{56}$$

The learning procedure for NN's output layer weights according to (49) and (50) is given with (57). First equation gives learning law for the weights of the first neuron in the output layer (i = = 1) and second for the weights of the second neuron (i = 2), $g'(net_i)$ = 1.

$$\Delta w_{k1j} = \eta_{1,k} \cdot K_{p,k} \cdot (d_k \sigma_k + \dot{\sigma}_k) \cdot g'(net_1) \cdot o_j \\ \Delta w_{k2j} = \eta_{2,k} \cdot K_{v,k} \cdot (d_k \sigma_k + \dot{\sigma}_k) \cdot g'(net_2) \cdot o_j. \tag{57}$$

C. Experimental results

Applied decentralized SM control law with NN is given with (55) and (56), NN with (43)–(46) and the learning law with (57). NN structure is shown in Figure 9. The same as for the previously described experiments, point-to-point movement, Figure 2. The load with mass 4.7 kg has been attached and released few times. Best results were achieved with j = 5 of neurons in the hidden layer. When using more or less of neurons, deterioration of the tracking accuracy appeared. l = 3 neural network inputs were used: actual position, actual velocity, and difference between desired and actual positions in the joint space. Initial weights were randomly chosen between −1 and +1. Learning rates were

$\eta_{1a,b} = 4e - 7,$
$\eta_{2a,b} = \eta_{3a,b} = 6e - 6,$
$d_{(k=1,2,3)} = [15, 23, 20].$

Position and velocity gains were
$Kp_{(k=1,2,3)} = [115, 150, 180]$ and
$Kv_{(k=1,2,3)} = [25, 40, 20].$

All parameters were set experimentally, beginning with the low initial values that were increased to achieve the best results. When too high learning rates or gains were applied the system became unstable, since noise was affecting the learning process. It should be also noted, that mathematically the stability of the BPG learning algorithm can not be always guaranteed, especially in the first moments of learning. However in the case of the applied SM control algorithm (55), the basic SM part keeps the system stable if NN output is not very large, although no NN convergence is

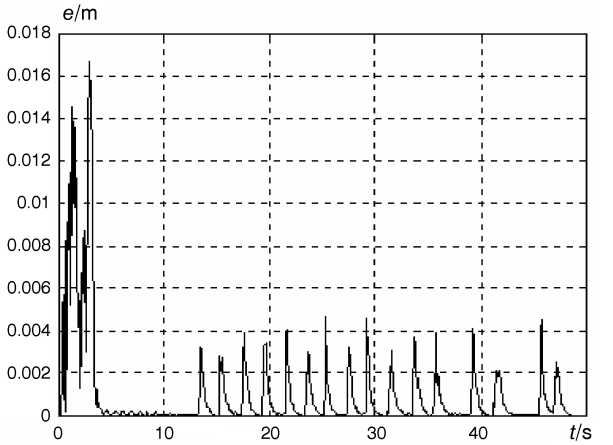

Fig. 10 SM control with NN: robot tip's position error during load changes

achieved. Therefore the NN output was limited to the lower and upper bounds estimated from the Lagrangian model [11]. When performing the experiments, there was no case of the unstable learning. The robot tip's position error is shown in Figure 10. Between the movement the top error is 16.5 mm while steady state position error is less then 0.1 mm. The load changes are efficiently compensated.

6 COMPARISON OF THE CONTROL METHODS

Performance of all designed control algorithms was verified by the application on DD robot. Comparison of the techniques by considering different aspects is given in this section.

A. Tracking and positioning error

Conventional CT control results in very high tracking and positioning error of about 35 mm; when the robot is loaded with unmodelled load, this error is even higher. For CT alike control with FLS the top tracking error at the acceleration phase is less that 3 mm and there is less then 0.1 mm steady state positioning error. SM with NN results in the top tracking error of 16.5 mm while steady state position error is also less then 0.1 mm.

B. Robustness

Robustness was tested by applying the step changes in the robot load, which require significant and rapid change in the motors' output torques and therefore presents outermost disturbance that can appear under the normal robot functioning conditions. In the case of both soft computing methods smooth transient response and fast convergence of the position errors after each load change was observed (Figure 6, Figure 10), since both schemes have adaptive parameters. In the case of CT control the load changes are not compensated and therefore cause the significant increase in the position error, however the system remains stable.

Another important aspect when considering robustness is sensitivity toward the measurement noise. This is the problem in the following cases: too high position and velocity gains (all three methods) and too high learning/adaptation rates (both soft computing methods). The problem can be omitted by the proper choice of the parameters, where the position and velocity gains are not too high. However the precedence concerning sensitivity to the measurement noise is evident in the case of control with FLS. FLS with input MFs filters out the noise and since the most of the control output is output from FLS (Figure 8) this lowers the noise sensitivity.

C. Computational requirements

Overall computational load for both soft computing methods is highly dependent on number of the MFs and rules, respectively on the number of neurons and NN's inputs. Therefore the comparison is given for here described controllers' structures.

CT control calculation requires at each sampling time: 21 additions/subtractions, 163 multiplications, 8 basic trigonometric functions, 9 if/else statements.

CT alike control with FLS calculation requires: 204 additions/subtractions, 390 multiplications, 30 if/else statements, 39 calculation of random numbers.

SM with NN calculation requires at each sampling time: 414 additions/subtractions, 474 multiplications, 6 calculations of exponential function, 90 random numbers.

D. Other design and implementation issues

Another important factor is time required for the design and the implementation. From here presented techniques CT control design is on overall most time consuming, since centers of the gravity and the inertia tensors for robot joints must be calculated, which is followed by the derivation of the complete dynamic model. When comparing, the design of the control with FLS system and SM with NN both require less time.

However the pretentiousness of the parameter setting is the lowest for CT control since only position and velocity gains have to be set and the required procedure is easy and straightforward as described in the Section 3. CT alike control with FLS is in this respect much more demanding, since beside position and velocity gains also initial values of the adaptive parameters, adaptation rates and average inertia have to be chosen. Similar is also pretentious of the parameters setting for SM control with NN.

Next redesign issue is considered. For soft computing methods sometimes the change of structure is required if the efficiency is not sufficient. For SM with NN this means different inputs, or number of the inputs, and/or different number of the neurons. The change of the number of NN's layers is usually not an option, since NN with two layers is already an universal aproximator. However with good programming, those two variations can be manageable by changing only some program vari-

ables. In the case of CT control with FLS, the change of the structure means the change of the number of MFs and rules. For this the complete design of the MFs and the rule base must be repeated.

Another aspect that should be considered is transparency of the controllers. CT control has the best transparency since its structure and parameters are physically conditioned. Also CT control with FLS subsystems remains transparent because it can be mostly recognized which dynamic effect is compensated by which rule. Therefore, if for example steady state error appears, the equations in CT control or rules in FLS that compensate gravitation should be redesigned. However SM control with NN can be considered as »black box« because the meaning of the each weight in NN can not be recognized.

7 CONCLUSION

The conventional model based CT control, CT control alike control with adaptive FLS and SM control with NN design and implementation results are presented and discussed in the paper.

CT control requires exact dynamic model, including the structure and the parameters. The design is pretentious and time consuming. Very poor results were obtained in the control of the DD robot, since the exact dynamic model could not be obtained. Unmodelled disturbances as changing load are not compensated. The method is not suitable for DD mechanisms, or other highly nonlinear mechanism.

CT alike control with FLS requires reasonable design and implementation time and yields good results even in the presence of the disturbances. It is less sensitive to the measurement noise. Algorithm is transparent, calculation requirements are acceptable. It can be also implemented to the control of the other mechanisms, where the structure of the dynamics is known, but the parameters are vague. Estimation of the average inertia is required.

SM control with NN presents good solution for cases where nor the parameters, nor the structure of the dynamics is known well enough. Only estimation of the lower and the upper bounds of the disturbances and estimation of the average inertia matrix is required. Good results were obtained for DD robot motion control. Design time and computational requirements are acceptable.

REFERENCES

[1] B. Bukkems, D. Kostić, B. De Jager, M. Steinbuch, **Learning-based Identification and Iterative Learning Control of Direct Drive Robots.** IEEE Transaction on Control Sys. Tec., Vol. 13, No. 4, pp. 537–549, 2005.

[2] A. Cichocki, R. Unbehauen, **Neural Network for Optimal Signal processing.** John Wiley and Sons, New York, 1993.

[3] J. Craig, **Adaptive Control of Mechanical Manipulators.** Addison-Wesley Publishing Company, 1988.

[4] B. Curk, K. Jezernik, **Sliding Mode Control with Perturbation Estimation: Application on DD Robot Mechanism.** Robotica, Vol. 19, pp. 641–648, 2001.

[5] H. Elmali, N. Olgac, **Implementation of Sliding Mode Control with Perturbation Estimation.** IEEE Trans. On Con. Sys. Technology, Vol. 4, No. 1, pp. 79–85, 1996.

[6] G. Fen, **A Survey on Analysis and Design of Model Based Fuzzy Control Systems.** IEEE Trans. On fuzzy sys., Vol. 14, No. 5, pp. 676–697, 2006.

[7] O. Kaynak, A. Denker, **Discrete Time Sliding Mode Control in the Presence of System Uncertainty.** Int. J. Control, Vol. 11, No. 7, pp. 665–678, 1993.

[8] L. Ljung, **System Identification: Theory for the Users.** Prentice Hall, New Jersey, 1987.

[9] K. S. Narendra, A. M. Annaswamy, **Stable Adaptive Systems.** Englewood Cliffs, Prentice-Hall International, 1998.

[10] A. Rojko, K. Jezernik, **Sliding-mode Motion Controller with Adaptive Fuzzy Disturbance Estimation.** IEEE trans. ind. electron., Vol. 51, No. 5, pp. 963–971, 2004.

[11] A. Rojko, S. Uran, J. Harnik, **Model of Direct Drive Robot.** (http://www.ro.feri.uni-mb.si/projekti/4_dd_drive.html, 2004; last accessed 08.11.2007).

[12] V. I. Utkin, **Sliding Modes in Control Optimization.** Springer Verlag, 1981.

[13] R. Safaric, K. Jezernik, M. Pec. **Neural Network Control for Direct-drive Robot Mechanisms.** Engineering application of artificial intelligence, Vol. 11, No. 6, pp. 735–745, 1998.

[14] J. Slotine, W. Li, **Applied Nonlinear Control.** Englewood Cliffs, NJ: Prentice-Hall, 1991.

[15] A. Šabanović, N. Šabanović, Ç. Onal, **Sliding Modes in Motion Control Systems.** Automatika, Vol. 46, 2005.

[16] M. Terbuc, K. Jezernik, **Design of DD Robot System.** V: 27th International Symposium on Industrial Robots, Milan. Robotics towards 2000: symposium proceedings., pp. 465–470, 1996.

[17] S. Treesatyyapun, **Adaptive Controller with Stable Fuzzy Rules Emulated Structure and its Application.** Engineering applications of Artificial Intelligence, Vol. 18, pp. 603–615, 2005.

[18] L. X. Wang, **Adaptive Fuzzy Systems and Control.** Englewood Cliffs, NJ: Prentice Hall, New Jersey, 1994.

Primjena tehnika mekog računalstva za upravljanje gibanjem robota. U članku se obrađuje problem upravljanja gibanjem robota uz djelovanje velikih neodređenosti kao što je promjenljivost tereta. Istražene su, i uspoređene na robotu s izravnim pogonima, jedna klasična metoda upravljanja i dvije metode zasnovane na tehnikama mekog računalstva. Prva je metoda upravljanja klasična metoda proračunavanja upravljačkog momenta pogonskih motora na osnovi Lagrangeovih jednadžbi koje opisuju dinamiku robota. Druga je metoda upravljanja slična klasičnoj metodi, ali se umjesto Lagrangeovih dinamičkih jednadžbi koristi adaptivni neizraziti sustav, a treća je metoda upravljanja zasnovana na kliznim režimima s primjenom neuronske mreže umjesto dinamičkog modela robota. Obje metode upravljanja zasnovane na tehnikama mekog računalstva dale su izvrsne rezultate u svim slučajevima, dok klasična metoda upravljanja nije dala dobre rezultate uz djelovanje neodređenosti u sustavu.

Ključne riječi: adaptivni neizraziti sustav, upravljanje proračunavanjem upravljačkog momenta, neuronske mreže, upravljanje robotom, klizni režim upravljanja

AUTHORS' ADDRESSES

Andreja Rojko
andreja.rojko@uni-mb.si

Riko Šafarič
riko.safaric@uni-mb.si

Karel Jezernik
karel.jezernik@uni-mb.si

University of Maribor
Faculty of Electrical Engineering and Computer Science,
Smetanova 17, 2000 Maribor, Slovenia

Received: 2007-09-19