Web Based Educational Tool for Neural Network Robot Control

J. Čas, D. Hercog and R. Šafarič, member of IAOE
University of Maribor, Faculty of electrical engineering and computer science, Maribor, Slovenia

Abstract— **This paper describes the application for teleoperations of the SCARA robot via the internet. The SCARA robot is used by students of mehatronics at the University of Maribor as a remote educational tool. The developed software consists of two parts i.e. the continuous neural network sliding mode controller (CNNSMC) and the graphical user interface (GUI). Application is based on two well-known commercially available software packages i.e. MATLAB/Simulink and LabVIEW. Matlab/Simulink and the DSP2 Library for Simulink are used for control algorithm development, simulation and executable code generation. While this code is executing on the DSP-2 Roby controller and through the analog and digital I/O lines drives the real process, LabVIEW virtual instrument (VI), running on the PC, is used as a user front end. LabVIEW VI provides the ability for on-line parameter tuning, signal monitoring, on-line analysis and via *Remote Panels* technology also teleoperation. The main advantage of a CNNSMC is the exploitation of its self-learning capability. When friction or an unexpected impediment occurs for example, the user of a remote application has no information about any changed robot dynamic and thus is unable to dispatch it manually. This is not a control problem anymore because, when a CNNSMC is used, any approximation of changed robot dynamic is estimated independently of the remote's user.**

Index Terms—**LabVIEW, Matlab/Simulink, Neural network control, remote educational tool, robotics,**

I. INTRODUCTION

An inverse dynamic model of a robot arm is needed to design a robot controller but modelling dynamic systems using mathematical equations is often difficult. This fact has involved soft computing techniques, which are used for estimating inverse dynamic and/or the regulation of dynamic systems, such as robot mechanisms. The dynamics of a robot is relatively unknown because of the friction and other influences from within the robot environment and, in the case of remote-controlled robot applications, this problem is even greater. Users of remote-controlled robot application are unable to physically remove the disturbing causes of a robot's dynamics and, what is even more obstructive, the disturbance is completely unknown. The consequence is that these conventional procedures for eliminating disturbance, which are used in non-teleoperation robot applications, can not be used in teleoperations. When using the CNNSMC as a control algorithm however, unknown robot dynamics is not a control problem anymore, because the approximation of changed robot dynamics is computed independently of the remote user. The presented application's additional advantage is the possibility for the on-line tuning parameters of CNNSMC with numerical controls on GUI, which improves the performance of a robot's controller and, additionally, makes application more appropriate as an educational tool for students of mehatronics. Students are able to learn how the parameters of CNNSMC have an affect on its performance because of the possibility of on-line tuning parameters and observing the responses of regulation using graphs and numerical indicators. They can improve their knowledge of the neural network control algorithm and, what is more important, with the use of internet connections experiments can be executed on a real SCARA robot from somewhere outside a laboratory (e.g.: home).

The developed software, which consists of both CNNSMC and GUI, is being executed on a DSP2 Roby robotic controller card. This DSP2 Roby system is connected via a serial RS-232 bus to a lab server, where the LabVIEW program is running. On the other hand, the DSP2 Roby robotic controller card is connected to robot servo electronics. Servo electronics is designed for generating the required current for the DC driving motors of robot axes and for measuring the position of robot axes with combination of incremental encoders, by which a sinus signal is generated. Hardware implementation is shown in Figure 1.

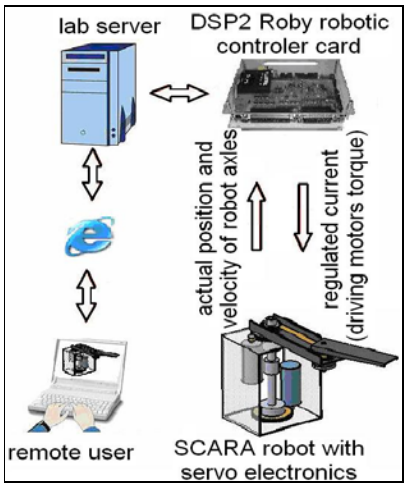

lab server
DSP2 Roby robotic controler card
actual position and velocity of robot axles
regulated current (driving motors torque)
remote user
SCARA robot with servo electronics

Figure 1: Hardware implementation of application

Section II, describes the mechanical and electrical characteristics of the SCARA robot that is used for experiment. In section III describes the software equipment that is used during the development and implementation of web-based educational tools. Section IV describes the synthesis of CNNSMC for a centralized neural network sliding-mode control law. The use of this application for educational purposes is described in section V. Section VI gives some conclusions.

II. DESCRIPTION OF 2. D.O.F SCARA ROBOT

The SCARA robot has two degrees of freedom. Rotation of the first robot link around the first robot axis is presented as the first degree of freedom, and rotation of the second robot link around the second robot axis is presented as the second. Each robot link is driven by a DC motor, and a gear-box with a transmission ratio 173/19. By using gear-boxes, any nonlinear inertia influences of the presented robot mechanism are decreased, but not completely. Additional dynamic nonlinearities are brought to the system as friction, which is proportionally large (about 20% of maximal torque) in the used SCARA robot.

Robot links are driven by a DC-motor with the commercial name ESCAP 28D11-219P. When a nominal voltage of 12[V] is reached, the nominal current is 1.5[A], nominal velocity is 5800[rpm] and the nominal torque is 28.4[*mNm*]. The used DC-motors are equipped with incremental encoders with an output sinus signal. The sinus signal from the incremental encoder is transmitted to 400 pulses per joint rotation by using servo electronics. Pulses from the incremental encoder are used for information on the position of a robot arm.

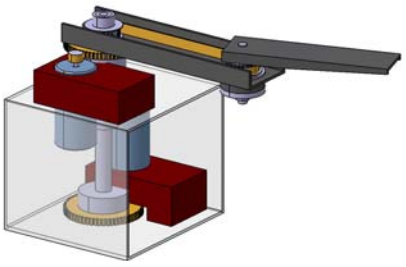


Figure 2: SCARA robot

A dynamic model of the SCARA robot mechanism, with two degrees of freedom is described by the Lagrange equation of motion, as follows:

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

where T is a vector of the drive torque on the robot's joints, M is an inertial matrix, h is a torque vector due to the centrifugal forces, centripetal forces, and Coriollis forces, F is a torque vector due to friction forces and T_n stands for a torque vector due to unknown disturbances.

$$\theta = [\theta_1, \theta_2]^T \in \Re^m,$$

$\dot{\theta}$, and $\ddot{\theta}$ are vectors of real positions, velocities and accelerations of the robot joints.

This well known mathematical note of a robot mechanism dynamics (1) can also be expressed with an n-dimensional state-space system of equations with regard to the control value u

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

New terms are defined as:

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

where d is an unknown disturbance, B is an actual input matrix, $\tilde{B}$ is an estimated input matrix, u is a control vector, x is a state space vector of mechanism, and t stands for time.

III. SOFTWARE COMPONENTS

The DSP-2 Roby system (Fig. 3) is composed of a DSP-2 controller [1] and a DSP-2 add-on robotic board. The key components of the DSP-2 controller are the floating point digital signal processor (DSP), used for control algorithm execution, and the Xilinx FPGA, which implements peripheral interfaces. The DSP-2 Roby system contains all the necessary peripheral for 4-axes robot control i.e. this system has 16 digital inputs, 8 digital outputs, 4 analog inputs/outputs, and 4 incremental encoder interfaces.

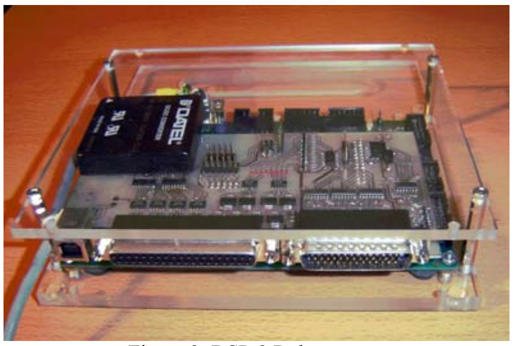


Figure 3: DSP-2 Roby system

The "DSP-2 Library for Simulink" [2] is a Simulink add-on toolbox, which provides rapid control prototyping (RCP) support for DSP-2 system (DSP-2 controller and DSP-2 Roby system). The DSP-2 library contains a set of device driver's blocks for all available I/O ports of the DSP-2 system, including blocks for analog I/O, digital I/O, incremental encoder and blocks for communication between the PC and DSP-2 system. Simulink, Real-Time Workshop and the DSP-2 Library for Simulink enable developers to model applications in the Simulink block-diagram environment and, after successful simulation, quickly verify the designed algorithm with the DSP-2 controller or DSP-2 Roby system on a real mechatronic device.

The LabVIEW virtual instrument named "ComVIEW" has been used for serving fly data visualization and parameter tuning tasks for the DSP-2 system (DSP-2 system is connected to the PC using RS-232 serial connection). When the DSP-2 target is selected in the Simulink model, a LabVIEW virtual instrument is automatically created from the ComVIEW template VI during binary code generation.

A ComVIEW template contains an empty front panel and a fully functional block diagram. The block diagram implements functions for VI initialization, executable code download to the DSP-2 system, functions for transmitting and receiving messages between the PC and the DSP-2 system, and other functions. During VI creation, numerical controls and indicators are automatically added to the VI front panel template, where the number of controls and indicators depends on the number of DSP-2 communication blocks used in the Simulink model. Links between DSP signals and VI front panel objects are established programmatically using the DSP Connection Manager window. This window appears on the PC immediately after the downloadable binary code starts executing on the DSP-2 target. All information about the DSP-2 input signals, DSP-2 parameters and DSP-2 output signals, as well as all information about VI scalar controls and the indicators of the front panel window, appear in the afore mentioned window. Using mouse clicks, the user can create links between the VI front panel indicators and the DSP-2 output signals, and links between the VI front panel controls and the DSP-2 input signals or DSP-2 parameters. When these links are set, a communication link is established between the VI running on the PC and the code being executed on a DSP-2 controller. Whenever the controls on the VI front panel are changed, LabVIEW automatically downloads them via RS-232 serial connection, to the DSP-2 controller. At the same time, all arrived DSP-2 output signals are read from the PC serial port and displayed on an appropriate numerical indicator. In addition, ComVIEW provides scope capabilities. In the scope mode, a small portion of code running on the DSP-2 controller handles data acquisition and storage management. The selected DSP-2 signals are, firstly, captured and then stored in the temporary controller's memory. After that, the captured data is transferred to the PC.

"Remote Panels" is a LabVIEW add-on toolkit developed by National Instruments that enables the viewing and controlling of LabVIEW VI's over the Internet. Using this toolkit, the LabVIEW VI can be published on the internet with no additional programming. Afterwards, the virtual instrument can be remotely observed or controlled by using standard web browser. The remote user can fully access the user interface that appears on the web browser and, consecutively, has complete control of the remote application. Other users can point their web browser to the same URL to view a remote experiment. To avoid confusion, only one client can control the application at a time but this control can easily be passed among the various clients at run-time.

A DSP-2 Roby system, connected to a lab PC through the serial port implements a control algorithm, developed using Simulink, and through the analog and digital I/O signals, drives a two-axes SCARA robot (Fig. 2). ComVIEW VI and the LabVIEW server are run on the same lab PC for the purpose of enabling teleoperation of the described system. ComVIEW VI performs communication between the lab PC and the DSP-2 system and online DSP-2 signals monitoring and parameters tuning, while the LabVIEW server enables remote operation of the ComVIEW VI. Remote users, connected to the server through the internet, must have a 'LabVIEW Run-Time Engine' installed on their personal computers, in order to perform remote experiments.

The remote user can also send teleoperational results via an email. The e-mail attachment contains experimental results in a format appropriate for further offline analysis in MATLAB.

IV. SLIDING-MODE CONTROLLER WITH NEURAL NETWORK ESTIMATION

The main advantages of the sliding mode control are the robustness to parameter uncertainty, loading disturbance, and fast dynamics response. However, these properties are valid on the sliding surface under the conditions of modelling imprecision absence, external disturbances, and switching time delays. The control law of the sliding mode technique contains a discontinuous component and may, on the sliding surface, excite those high frequency dynamics neglected in modelling, due to high control activity. Practical sliding mode control implementations exhibit high frequency oscillations in the plant output, called chattering. This causes harmful effects such as torque pulsation in robot electrical drives and, consequently, imprecise positional control of a robot tip. Various methods have been proposed to eliminate chattering. The most commonly cited approach is smooth approximation of the switching element by saturation, so that a narrow boundary layer is introduced near the sliding surface. In [3] authors suggested that the control law is constituted by two components: a continuous law derived from the plant model using the Lyapunov theory, and a saturation law. The first component is required to slide-down on the sliding surface, while the saturation law controls the unknown parts of the robot dynamics.

A number of researchers have suggested methods for alleviating the chattering effects and improve precision in the sliding mode position control of robot mechanisms. In [4] and in [5], the authors developed perturbation estimation schemes, others suggested supplementing the control input with a predictive correction term [6]. Recently, soft computing techniques fuzzy logic [7], neural network [8] and genetic algorithm [9] have been used to estimate the robot non-linear dynamics, thus needing only the continuous law derived from the robot dynamics using the Lyapunov theory.

Our approach uses a neural network as an estimator for a part or, even complete, robot dynamics. We decided to use a neural network because of its high convergence speed in robot dynamics estimation, robust sliding-mode control scheme, and as little preliminary knowledge on the estimated mechanism dynamics model as possible. In our case, only nominal (average) values were used for inertia matrix parameters. Differences between actual inertia matrix parameters and nominal inertia matrix parameters represent structured uncertainties. Torque terms due to Coriollis forces and friction forces were neglected and they represent non-structured uncertainties. If we did not have a robust control scheme, the robot behavior would be unpredictable during the first few moments of learning.

a) Design of a continuous neural network sliding-mode controller (CNNSMC)

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

Our goal is to prove the function stability $\sigma(x,t)=0$ (equation 4) for the robot system (equation 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 x_r will equal zero and will be stable for all disturbances. Function $\sigma(x,t)=0$ will be stable if the Lyapunov function *V*>0 and the first Lyapunov time derivative of function is $\dot{V}<0$. The selected Lyapunov function *V* (equation 5) is always greater than zero for whichever selected vector x_r, *x,* and matrix *G.* However, it is not always possible to select the negative first derivative of the Lyapunov time-dependent function $\dot{V}$ (equation 6) for every x_r, *x,* and *G*. According to the following definition:

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

where 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's stability (equation 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:

$$V=\sigma^T\cdot\sigma/2 \tag{5}$$

The following is derived from equation 5:

$$\dot{V}=\sigma^T\cdot\dot{\sigma} \tag{6}$$

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

$$\dot{V}=-\sigma^T\cdot D\cdot\sigma, \tag{7}$$

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

If the definition 7 and the derivative of Lyapunov's function 6 are made equal, the result is:

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

Equation 8 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 (equation 9):

$$D.\sigma+\dot{\sigma}=0. \tag{9}$$

If equation 4 is differentiated and the 2 is inserted into the recently calculated derivative, we obtain the following result:

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

After the equation 10 has been inserted into the implementation condition of control law 9, the result is as follows:

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

Since the term $(f+\Delta B\cdot u+d)$ is unknown and not measurable it is, therefore, approximated with the neural network $N=\left[o_1\cdots o_i\right]^T$ (see Fig. 4) by changing equation 11 into:

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

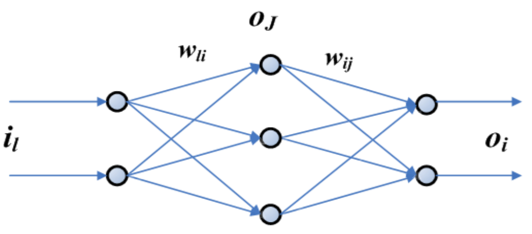


Figure 4: Neural network

Since the term $(f+\Delta B\cdot u+d)$ is unknown and non-measurable, a classic supervised weight learning of the neural network cannot be used. Therefore, a so-called 'on-line estimator' has been developed (Fig. 5), estimating a learning signal (that is the difference between the target and the output of a neural network).

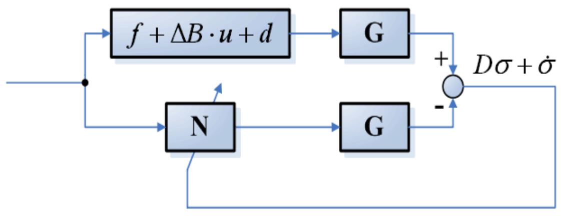


Figure 5: On-line estimator

The result after equation 4 has been differentiated, is as follows:

$$\dot{x}=G^{-1}.\dot{\sigma}+\dot{x}_r. \tag{13}$$

After the terms 12 and 13 have been inserted into the basic equation of mechanism dynamics, the result is as follows:

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

where is substituted $Z = f + \Delta B \cdot u + d$.

By using the derivative of Lyapunov's function and the equation 14, the condition, where the system controlled by the developed control law (equation 12) remains stable, is assured:

$$\dot{V}=\sigma^T.\dot{\sigma}=\sigma^T.G.(f+\Delta B u+d-N)-\sigma^T.D.\sigma<0. \qquad (15)$$

The next condition (Equation 16) has been developed from equations 14 and 15. To make $\dot{V} < 0$ and consequently $\sigma \rightarrow 0$ possible, the condition expressed in equation 16 has to be fulfilled during the time during in which the neural network is approximating the unknown part of robot dynamics ($f + \Delta B \cdot u + d$).

$$|G.(f+\Delta B.u+d)-G.N| = |D.\sigma+\dot{\sigma}| < |D.\sigma|. \qquad (16)$$

To learn about the output layer of neural network with two layers, a modified BPG rule has been used:

$$net_i = \sum_j w_{ij}.o_j + b_i, o_i = g(net_i), \qquad (17)$$

$$g(net) = 1-2/(1+e^{-net}), \Delta w_{ij} = \varepsilon.\partial E / \partial w_{ij}, \qquad (18)$$

$$\begin{aligned} E &= (D.\sigma + \dot{\sigma})^T.(D.\sigma + \dot{\sigma})/2 = \\ &= (G(Z-N)^T).(G.(Z-N))/2, \end{aligned} \qquad (19)$$

$$\begin{aligned} &\Delta w_{ij} = \varepsilon.\partial E/\partial net_i.\partial net_i/\partial w_{ij} = \varepsilon.\partial E/\partial net_i.o_j = \\ &= \varepsilon.\partial E/\partial o_i/\partial o_i/\partial net_i.o_j \Rightarrow \\ &\Delta w_{ij} = \varepsilon.\partial E/\partial o_i.g'(net_i)o_j, \end{aligned} \qquad (20)$$

where:

$$\begin{aligned} &\partial E/\partial o_i = \partial/\partial o_i\left[(G.Z-G.N)^T.(G.Z-G.N)\right]/2 = \\ &= \partial/\partial o_i\left[(G.N)^T.(G.Z-G.N)\right] \Rightarrow \\ &\partial E/\partial o_i = \partial/\partial o_i\left[(G.N)\right]^T.(D\sigma+\dot{\sigma}) \end{aligned} \qquad (21)$$

To learn the hidden layer of a neural network the traditionally back propagation rule is used.

b) Centralized control law for two 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 SCARA robot mechanism with two degrees of freedom, which is shown in the figure 2, will be derived.

$$T = M.\dot{\theta} + h + G_f + T_n, \qquad (22)$$

where *T*, *h*, G_f , and T_n (See equation 1) are column vectors of the 2 by 1 dimension, *M* is the matrix of the 2 by 2 dimension, and $\theta = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^T$ is the column vector of the 2 by 1 dimension of two axes of the SCARA robot. The previous equation can also be rewritten in the following form:

$$\dot{x} = f(x,t) + B(x,t).u + d(x,t), \qquad (23)$$

where:

$$\begin{aligned} x &= \begin{bmatrix}\theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2\end{bmatrix}^T, \\ \dot{x} &= \begin{bmatrix}\dot{\theta}_1 & \dot{\theta}_2 & \ddot{\theta}_1 & \ddot{\theta}_2\end{bmatrix}^T, \end{aligned} \qquad (24)$$

$$\begin{aligned} f &= \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \hat{M}^{-1}\left[\hat{h} + \hat{G}_f + \hat{F} + \hat{T}_n\right] \end{bmatrix}, \\ \widetilde{B} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \multicolumn{2}{c}{\hat{M}^{-1}} \end{bmatrix} \end{aligned} \qquad (25)$$

and where $\hat{M}$, $\hat{h}$, $\hat{G}_f$ *in* $\hat{T}_n$ are estimated values *M*, *h*, G_f , and T_n .

Only nominal or average parameters are selected for the matrix $\hat{M}$. This means that all 4 parameters of the matrix $\hat{M}$ are constant while the robot hand is moving. This is, of course, only a rough simplification of how things really look; for it is a common fact that the parameters of the matrix $\hat{M}$ vary according to individual axis movements in a robot's working space. Because of this, the unknown variable part ΔB exists and is estimated by the neural network (see equation 14). The dimension of vector *f* is 4 by 1 and the dimension of the matrix $\widetilde{B}$ is 4 by 2.

The control law *u* of the 2 by 1 dimension is illustrated in the following equation:

$$u = -(G\widetilde{B})^{-1}.\left[G.(N - \dot{x}_r) + D\sigma\right], \qquad (26)$$

where:

$$\begin{aligned} G &= \begin{bmatrix} K_{P1} & 0 & K_{V1} & 0 \\ 0 & K_{P2} & 0 & K_{V2} \end{bmatrix}, \\ D &= \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}, \end{aligned} \qquad (27)$$

$$x_r = \begin{bmatrix}\theta_{1r} & \theta_{2r} & \dot{\theta}_{1r} & \dot{\theta}_{2r}\end{bmatrix}^T$$
$$\dot{x}_r = \begin{bmatrix}\dot{\theta}_{1r} & \dot{\theta}_{2r} & \ddot{\theta}_{1r} & \ddot{\theta}_{2r}\end{bmatrix}^T \quad (28)$$

and

$$\sigma = G(x - x_r)\,. \quad (29)$$

Coefficients of the matrices *G* in *D* should be selected in such a way that they enable the fastest convergence of neural network algorithm possible. The column vector *N* is of the 4 by 1 dimension and represents the outputs of the neural network o_i with i=1...4.

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

$$\begin{aligned}\Delta w_{1j} &= \varepsilon_J.\begin{bmatrix}K_{P1} & 0\end{bmatrix}.(D\sigma+\dot{\sigma}).g'(net_1).o_j\\ \Delta w_{2j} &= \varepsilon_J.\begin{bmatrix}0 & K_{P2}\end{bmatrix}.(D\sigma+\dot{\sigma}).g'(net_2).o_j\\ \Delta w_{3j} &= \varepsilon_J.\begin{bmatrix}K_{V1} & 0\end{bmatrix}.(D\sigma+\dot{\sigma}).g'(net_3).o_j\\ \Delta w_{4j} &= \varepsilon_J.\begin{bmatrix}0 & K_{V2}\end{bmatrix}.(D\sigma+\dot{\sigma}).g'(net_4).o_j\end{aligned} \quad (30)$$

where:

$$net_i = \sum_j w_{ij}.o_j + b_i \quad (31)$$

and where *j*=1...20, *i*=1...6, *l*=1...4, and *g*'(*) is the first derivative of the sigmoid function (equation 18).

The neural network consists of 6 inputs; these are: two actual positions, two actual velocities and two differences between the reference and the actual position. All of them lie in the joint space of the robot mechanism.

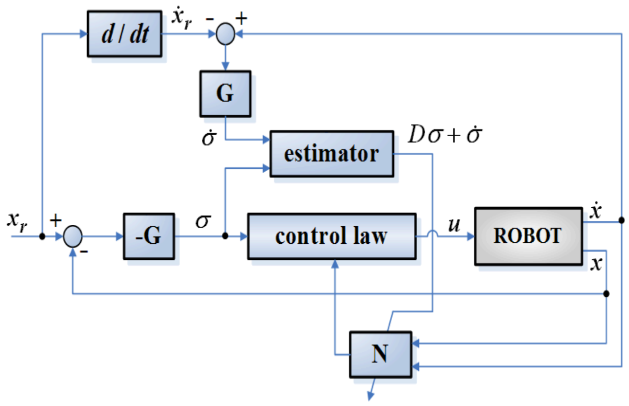


Figure 6: CNNSMC's control scheme

Web based educational tool is designed for educational purposes for students of mechatronics at the University of Maribor. The influence of parameters on the quality of SCARA robot position control with CNNSMC can be studied. Executing control tests from a student's home is another advantage of its application and the only predefined conditions are reliable internet connection and installed Lab View's free library *LV Run Time Engine*.

When a student or any other user intends to take a control over a web-based educational tool, the internet address of the web-based application is written in to the internet browser program. The falling menu appears using the right-mouse click. In this menu the option *Request control of VI* must be selected.

After control is established, the dialog window appears. The user name and user email address must be written there, because the results of the experiment are sent via an email (fig.7).

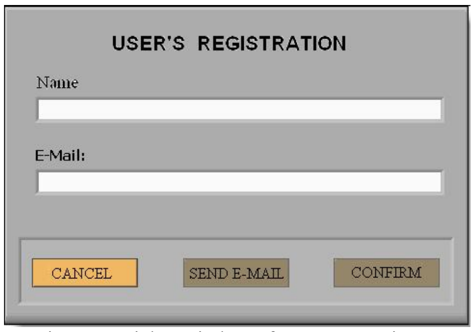


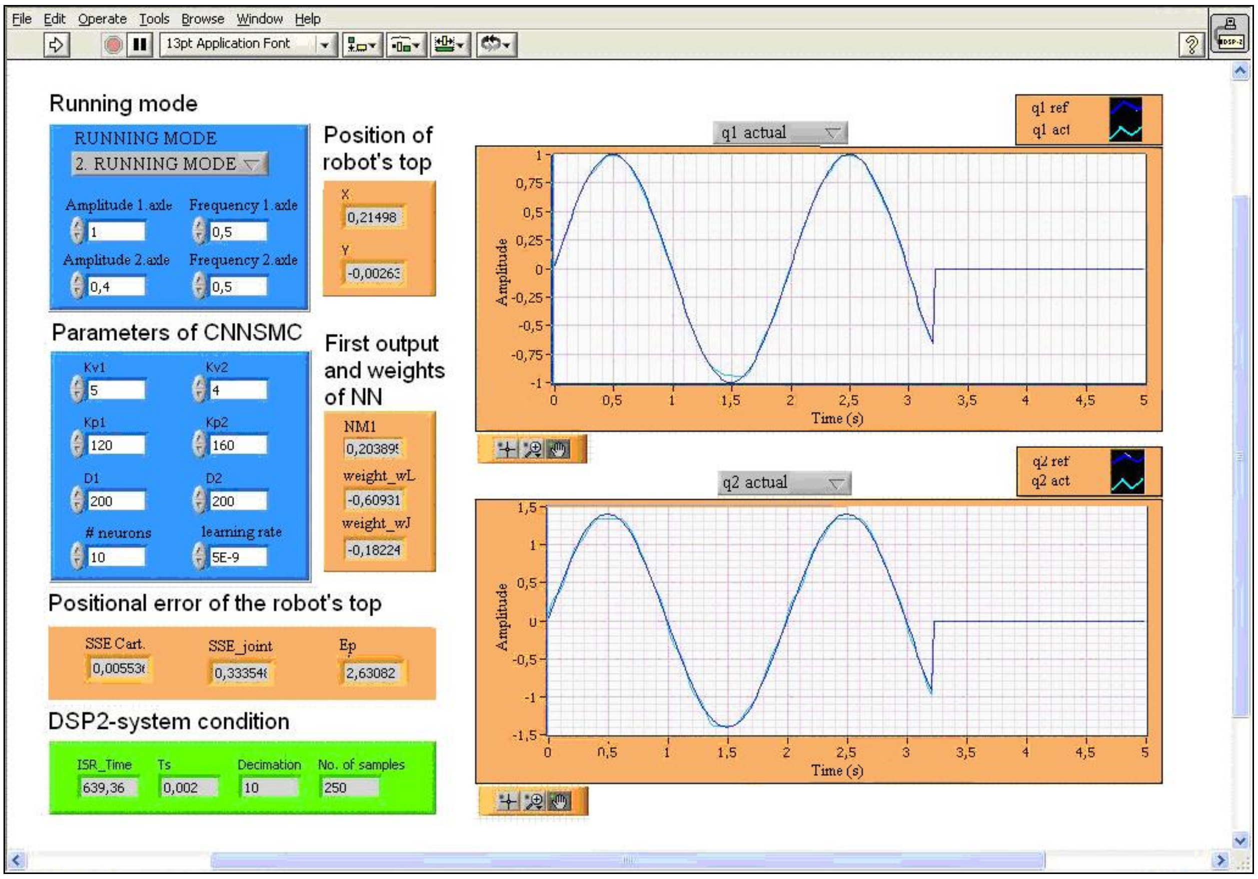
Figure 7: Dialog window of remote experiment

When the button POSLJI (send) on the control panel is chosen, the results are sent to a defined email address. The results are attached in folder "rezultati.zip", which consists of:

- **"rezultati.rtf"**: the front panel and block diagram of the used virtual instrument is attached in this folder.
- **"rezultati.dat"**: the text-data of all measured signals is contained in this folder (in column form). The first column is time data, and the other columns contain the measured data. This format can easily be imported to Matlab, where it can be observed in graph form.

When the initial procedure of a web-based educational tool is completed by the user, the front panel of the virtual instrument appears (fig. 8). By using 'switch' on the top left rectangle of the control panel, it is possible to choose from three different running modes of the SCARA robot. The holding robot in the fixed-point is the first running mode, rotating robot links with sinus reference in a joint-space is the second running mode (fig. 8), and defining the top coordinates of the robot arm in the Cartesian space by mouse clicking on the working area, which is presented as the grid on control panel, is the third running mode.

When the second running mode of the robot mechanism is chosen, the user is able to change the rotational frequency of the SCARA robot's axes from zero to half-radian per second in joint space and the amplitude of rotation from zero to *pi* radians and two graphs appear on the control panel of the virtual instrument. The referenced and actual position values of the first robot axis are shown on the upper graph, and the referenced and actual position values of the second axis are shown on the bottom graph. In addition, a positional error of robot's top, a sum square error in the joint space and the Cartesian space, values of some weights of neural network and the first output from neural network, can also be shown on both graphs.


File Edit Operate Tools Browse Window Help

13pt Application Font

Running mode

RUNNING MODE: 2. RUNNING MODE

Amplitude 1.axle: 1 | Frequency 1.axle: 0,5

Amplitude 2.axle: 0,4 | Frequency 2.axle: 0,5

Position of robot's top

X: 0,21498

Y: -0,00263

Parameters of CNNSMC

Kv1: 5 | Kv2: 4

Kp1: 120 | Kp2: 160

D1: 200 | D2: 200

neurons: 10 | learning rate: 5E-9

First output and weights of NN

NM1: 0,20389

weight_wL: -0,60931

weight_wJ: -0,18224

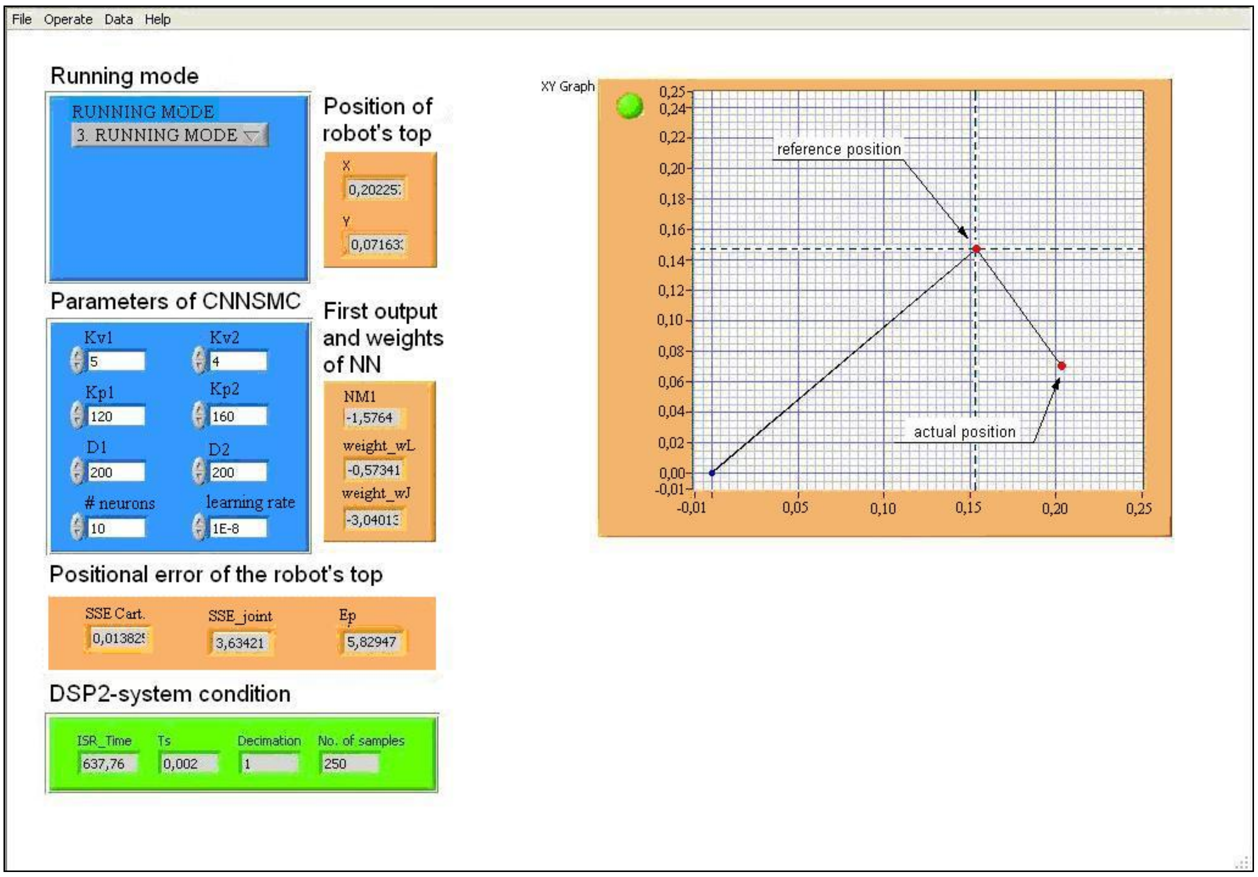
Positional error of the robot's top

SSE Cart.: 0,00553 | SSE_joint: 0,33354 | Ep: 2,63082

DSP2-system condition

ISR_Time: 639,36 | Ts: 0,002 | Decimation: 10 | No. of samples: 250

Figure 8: Front panel of VI (second running mode)


File Operate Data Help

Running mode

RUNNING MODE: 3. RUNNING MODE

Position of robot's top

X: 0,20225

Y: 0,07163

Parameters of CNNSMC

Kv1: 5 | Kv2: 4

Kp1: 120 | Kp2: 160

D1: 200 | D2: 200

neurons: 10 | learning rate: 1E-8

First output and weights of NN

NM1: -1,5764

weight_wL: -0,57341

weight_wJ: -3,04013

Positional error of the robot's top

SSE Cart.: 0,01382 | SSE_joint: 3,63421 | Ep: 5,82947

DSP2-system condition

ISR_Time: 637,76 | Ts: 0,002 | Decimation: 1 | No. of samples: 250

Figure 9: Front panel of VI (third running mode)

When the third running mode of the robot mechanism is chosen, the reference points of the robot's top can be defined by mouse clicking on the X-Y graph on control panel, by which the working area of the robot on the x-y plane is presented. If the selected reference point is not on the control panel, a red bulb signals it and the robot does not move. In the third running mode, only the grid that represent the x-y plane of the robot is shown on the control panel, while the two graphs from the second running mode are not represented.

In the middle of the control panel, within the small rectangles, the numerical values of the robot's position in the joint space, the values of some neural network's weights, the value of the first output from the neural network, the values of sum square errors in the joint space and Cartesian space, and the positional error value of the robot's top in Cartesian space, can be observed in the same control panel.

By using the numerical controls on the left middle rectangle on the control panel, the user can change the values of CNNSMC parameters at predefined intervals. The performance of CNNSMC changes when changing these parameters. The student's work is to find the best parameters, which would ensures that the CNNSMC's performance is optimal. This is shown as an accurate and fast regulation of a robot, without chattering. Predefined intervals of parameters are shown in Table 1.

Parameter	Interval
K_{V1}	2 ÷ 20
K_{V2}	2 ÷ 20
K_{P1}	20 ÷ 200
K_{P2}	20 ÷ 200
D_1	20 ÷ 400
D_2	20 ÷ 400
$no.neurons \quad (h.layer)$	2 ÷ 20
η	1e-8÷1e-11

TABLE 1: INTERVALS OF PARAMETERES

The bottom left box shows the DSP2 Roby card condition. The fixed-step time and the computing time of algorithm in one step are shown.

VI. CONCLUSION

This web-based educational tool for neural network robot control is designed to provide students of mechatronics useful control techniques in telerobotics. When the robot dynamic is changed, this information is unknown to the user and because of this he is unable to dispatch it manually. This is not a problem for control however, because when CNNSMC is used, the approximation of changed robot dynamics is computed independently of the remote's user. The presented application's additional advantage is the possibility on-line tuning parameters using CNNSMC, with the numerical controls of GUI, thus improving the performance of the robot's control and also making application more appropriate as an educational tool for students of mehatronics. With the possibility of on-line tuning parameters and observing responses of control using graphs and numerical indicators, students are able to learn how the parameters of CNNSMC affect performance. They can improve their knowledge of a neural-network based control algorithm.

One additional option, not previously mentioned, is using a Windows remote desktop connection with the laboratory server. Students are able to load their own code for CNNSMC and run it, but a self-developed code must be tested beforehand. Testing of self-developed code can be done on the inverse model of the SCARA robot, which was designed in Matlab/Simulink during this project. As can be seen, additional work on developing this web-based application for neural network robot control is possible in the future.

REFERENCES

[1] DSP-2 web page: www.ro.feri.uni-mb.si/projekti/dsp2

[2] D. Hercog, M. Čurkovič, G. Edelbaher, E. Urlep, "Programming of the DSP2 board with the MATLAB/Simulink," Proceedings IEEE ICIT 2003, December 2003, pp. 709-713;

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

[4] H. Elmali and N. Olgac (1996): Implementation of sliding mode control with perturbation estimation. *IEEE Trans. On Con. Sys. Technology* , Vol.4, No.1, pp. 79-85, 1996

[5] B. Curk, and K. Jezernik (2001): Sliding mode control with perturbation estimation: application on DD robot mechanism. *Robotica*, Vol. 19, pp. 641-648, 2001.

[6] O. Kaynak and A. Denker (1993): Discrete time sliding mode control in the presence of system uncertainty. *Int. J. Control*, Vol. 11, No. 7, pp. 665- 678, 1993.

[7] Rojko A., Jezernik K., (2004) Sliding-mode motion controller with adaptive fuzzy disturbance estimation. *IEEE trans. ind. electron*. Vol. 51, No. 5, pp. 963-971

[8] Šafarič R., K. Jezernik, M. Pec (1998): Neural network control for direct-drive robot mechanisms. *Engineering application of artificial intelligence*, Vol. 11, No. 6, pp. 735-745, 1998.

[9] Fujisawa, S., M. Obika,. T. Yamamoto, K. Kawada, O. Sueda (2004): Speed control of 3-mass system with sliding mode control and CMAC.*IEEE International Conference on Systems, Man and Cybernetics*, 2004 Vol. 5, pp.4400 – 4407

AUTHORS

J.Čas is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: jure.cas@uni-mb.si).

D. Hercog is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: darko.hercog@uni-mb.si).

R. Šafarič is with the University of Maribor, Faculty of electrical engineering and computer science, Institute for robotics, Smetanova 17, 2000 Maribor, Slovenia (e-mail: riko.safaric@uni-mb.si).

Manuscript received March 20, 2007.