Journal of ELECTRICAL ENGINEERING, VOL. 55, NO. 5-6, 2004, 150–155

USING FORWARD AND INVERSE NEURAL MODELS FOR SOLVING OPTIMAL TRACKING PROBLEM OF NON–LINEAR SYSTEM

Anna Jadlovská *

This paper presents the use of the theory of optimal control and linear-quadratic approach to the deterministic controller design for non-linear dynamical system - robot. In the first stage the optimal target trajectories are off-line determined that serve as the required inputs for the feedback tracking control of the robot, whereby the feedback is determined from the linearized robot model. The results of this classical approach are used to design an optimal tracking neuro-controller (OTNC) with quadratic cost function. The proposed feedforward control scheme with neural models will be demonstrated on a typical non-linear system, a two-link robot. Computer simulations of the control algorithm are made by the simulation language Matlab and the presented results are analyzed.

K e y w o r d s: programmed optimal control, two-bounded problem, sensitive function, linear-quadratic control, tracking problem, Riccati equations, forward neural model, inverse neural model

1 INTRODUCTION

There exist large number of papers dedicated to the control problems of the robots. We know that the robots are characterized by a complex non-linear dynamical structure with unmodelled dynamics and unstructured uncertainties. These features make the designing of controllers for the robots a difficult task in the framework of non-adaptive and adaptive control of the non-linear dynamic systems.

The goal of this paper is to present an engineering approach to the control of an industrial robot using the results of the theory of optimal control and linear-quadratic approach to the design of a deterministic controller applying two-stage control synthesis, [1, 6].

In the first stage the optimal target trajectories are determined from a non-linear model of the robot solving the two-bounded problem that serve as the required nominal inputs for the feedback tracking control of the robot. In the second stage the feedback is determined from a linearized model of the robot along nominal trajectories using LQ approach.

The results of this classical approach of nominal trajectories (programmed optimal control) and matrix of gains from Riccati equations (feedback optimal control) are used to design an optimal tracking neuro-controller (OTNC) for the robot with quadratic cost function.

The OTNC is made of two controllers: the feedforward neuro-controller (FFNC) and feedback neuro-controller (FBNC). The feedforward neuro-controller (FFNC) generates steady-state control inputs to keep the system output to a given reference value and is trained as inverse neural model by back-propagation algorithm. The feedback neural controller (FBNC) generates the transient control input to stabilize error dynamics along the optimal trajectories while minimizing the quadratic function. The FBNC is trained as forward neural model by back-propagation algorithm.

The control methodology in this paper is useful as a control method, where the nominal control inputs and target trajectories of the motion of the robot are first off-line computed by an algorithm of sensitive functions and then a controller is designed on-line by an algorithm of LQ approach.

The proposed feedforward control scheme with neural models will be demonstrated on a typical non-linear system two-link robot.

2 DYNAMIC MODEL OF ROBOT

We shall consider the robot with the kinematics structure by Fig. 1, [7], the dynamic model of which is described in the state space:

$$\begin{aligned} \dot{x}_1 &= x_2\,, \quad \dot{x}_2 = \frac{m_b}{m_r} x_1 x_4^2 + \frac{K_1}{m_r} u_1\,, \\ \dot{x}_3 &= x_4\,, \quad \dot{x}_4 = -\frac{2m_b x_1 x_2 x_4}{I_{23} + m_b x_1^2} + \frac{K_2}{I_{23} + m_b x_1^2} u_2 \end{aligned} \tag{1}$$

where

$m_b = m_z + m_1\,,\ m_r = m_{rr} + m_b + m_2\,,$

$m_z = 35$ kg is the mass of the weight,

$m_1 = 52$ kg is the mass of the grasp head and part of the arm,

$m_{rr} = 62.5$ kg is reduced mass of the gear,

$m_2 = 78$ kg is the mass of the servomotors of the arm,

$K_1 = 281.4$ Nm, $K_2 = 291$ Nm are the constants of the operational values,

* Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, E-mail: Anna.Jadlovska@tuke.sk

ISSN 1335-3632

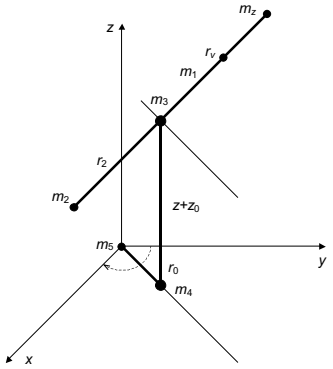

Fig. 1. The kinematics scheme of the robot

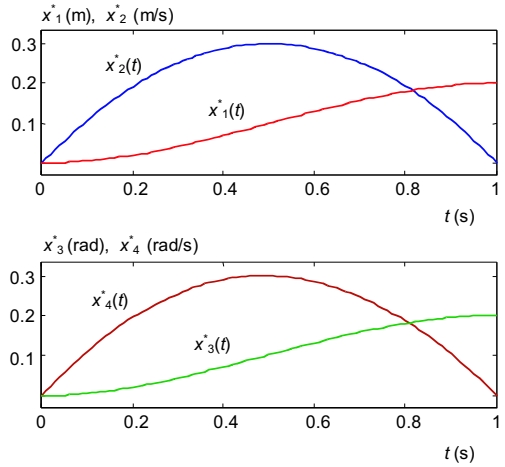

Fig. 2. Optimal state variables $x_1^*(t) = r(t)$, $x_2^*(t) = \dot{r}(t)$, $x_3^*(t) = \varphi(t)$, $x_4^*(t) = \dot{\varphi}(t)$ for $m_Z = 35$ kg ($x_{ref}(t) = x^*(t)$)

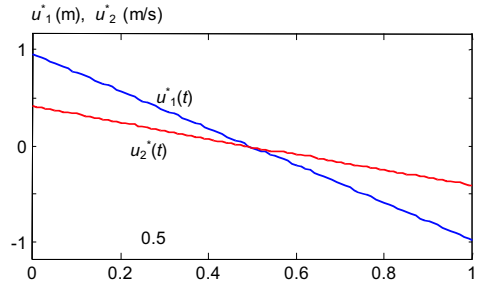

Fig. 3. Optimal control inputs $u_1^*(t)$, $u_2^*(t)$ for $m_Z = 35$ kg

$I_{23} = I_r + (m_2 + m_3 + m_4)r_0^2$,
$I_r = 82.5\,\mathrm{kgm}^2$ is reduced torque of the inertia of the electric servo-motor and the gear box,
$m_3 = 90$ kg, $m_4 = 125$ kg, $r_0 = 250$ mm.
The elements of the state vector are:S

$$x_1 = r\,[\mathrm{m}]\,; \qquad x_2 = \dot{r}\,[\mathrm{m\,s^{-1}}]\,;$$
$$x_3 = \varphi\,[\mathrm{rad}]\,; \qquad x_4 = \dot{\varphi}\,[\mathrm{rad\,s^{-1}}]\,;$$

where r is translation of the arm,
φ is rotation of the arm $(0, 2\pi)$.

3 OPTIMAL CONTROL OF ROBOT

The dynamic of the robot (1) is modelled by a vector differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)\,;\ \boldsymbol{x}_0(t) = \boldsymbol{x}_0 \tag{2}$$

where
$\boldsymbol{x}(t)$ is the state vector of the system, n-dimensional vector with components $x_1(t), x_2(t), \ldots, x_n(t)$;
$\boldsymbol{u}(t)$ is the control vector of the system, m-dimensional vector with components $u_1(t), \ldots, u_m(t)$;
$\boldsymbol{x}_0(t) = \boldsymbol{x}_0$ is the initial state vector;
$\boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$ is the non-linear vector function of the system with components
$f_1\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big), f_2\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big), \ldots, f_n\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$.

For the mechanical system (robot) which is described by (1) and the initial state $\boldsymbol{x}_0$, we find optimal control $\boldsymbol{u}^*(t)$ and target trajectories of the motion $\boldsymbol{x}^*(t)$, which drives dynamical system (2) from the initial state $\boldsymbol{x}_0(t) = \boldsymbol{x}_0$ to a given finite state $\boldsymbol{x}_T$ for all $t \in \langle t_0, T\rangle$ such that the cost function

$$J = \frac{1}{2}\int_{t_0}^{T} \boldsymbol{u}^{\top}(t)\,\boldsymbol{u}(t)\,\mathrm{d}t \tag{3}$$

is minimized.

We can use Pontryagin maximum principle to transfer this problem to the solution of the two-point boundary problem of the ordinary differential equations

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\,[\boldsymbol{x}(t), \boldsymbol{p}(t)] \\ \dot{\boldsymbol{p}}(t) &= \boldsymbol{g}\,[\boldsymbol{x}(t), \boldsymbol{p}(t)] \end{aligned} \tag{4}$$

where $\boldsymbol{p}(t)$ is the costate vector and the boundary conditions are:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \quad \text{and} \quad \boldsymbol{x}(T) = \boldsymbol{x}_T\,. \tag{5}$$

The problem is to find the initial conditions of the costate vector $\boldsymbol{p}(t)$. Determination of the correct initial conditions for costate vector $\boldsymbol{p}(t_0) = \boldsymbol{p}_0$ requires an iterative process. We used the sensitive functions method in the algorithm of programme optimal control and the detailed description of the solution for this problem is in [4].

The result solving of the problem of programme optimal control are found optimal controls inputs $\boldsymbol{u}_1^*(t)$, $\boldsymbol{u}_2^*(t)$ and target trajectories of the motion of the robot: $x_1^*(t) = r(t)$, $x_2^*(t) = v(t)$, $x_3^*(t) = \varphi(t)$, $x_4^*(t) = \omega(t)$ by simulations in language Matlab by Control and Symbolic Toolboxes.

The boundary conditions are $\boldsymbol{x}(0) = \boldsymbol{0}$, $\boldsymbol{x}(T) = [0.2, 0, 0.2, 0]$ and the cost function is

$$J = \frac{1}{2}\int_0^1 \big(\boldsymbol{u}_1^2 + \boldsymbol{u}_2^2\big)\,\mathrm{d}t\,. \tag{6}$$

Simulation results of the implementation of the algorithm programme optimal control (POC) are in Fig. 2 and Fig. 3.

In real industrial robots which are characterized by complex non-linear dynamical structures with unmodelled dynamics and unstructured uncertainties, their parameters are not constant (the deterministic mathematical model of the robot does not correspond to the real physical model. We must find the control input $\boldsymbol{u}(t)$ which keeps the real state of the system $x(t)$ near its target trajectories $x^*(t)$ for all $t \in \langle t_0, T\rangle$. It is clear that the real plant input $\boldsymbol{u}(t)$ must be different from the nominal input $\boldsymbol{u}^*(t)$ computed by optimal control using the sensitivity functions.

So, we define the following quantities:

1. The state perturbation vector $\delta\boldsymbol{x}(t)$:

$$\delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}^*(t)\,, \tag{7}$$

where $\boldsymbol{x}(t)$ is the real state of the system, $\boldsymbol{x}^*(t)$ is nominal state of the system — optimal trajectory computed by sensitivity function.

2. The control correction vector $\delta\boldsymbol{u}(t)$:

$$\delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}^*(t)\,, \tag{8}$$

where $\boldsymbol{u}(t)$ is the real control input, $\boldsymbol{u}^*(t)$ is the nominal control input.

We can imagine that the control correction vector $\delta\boldsymbol{u}(t)$ is generated by a deterministic controller the input of which is the state perturbation vector $\delta\boldsymbol{x}(t)$. Thus, in this deterministic case we must use the control to take care for errors that are primarily associated with errors in modelling.

3.1 Linearized Perturbation Model

We assume, that the deterministic model of our nonlinear system (robot) is described by eqn. (2) as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

Similarly, for nominal control input $\boldsymbol{u}^*(t)$, the nominal state $\boldsymbol{x}^*(t)$ are related by

$$\boldsymbol{x}^*(t) = \boldsymbol{f}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))\,. \tag{9}$$

Expanding $\boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$ along $\boldsymbol{x}^*(t)$, $\boldsymbol{u}^*(t)$ in a Taylor series expansion we obtain

$$\boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) = \boldsymbol{f}\big(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\big) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_0 \delta\boldsymbol{x}(t) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_0 \delta\boldsymbol{u}(t) + \alpha_0\big(\delta\boldsymbol{x}(t), \delta\boldsymbol{u}(t)\big) \tag{10}$$

where $\alpha_0(\delta\boldsymbol{x}(t), \delta\boldsymbol{u}(t))$ denote the higher order terms in the Taylor series expansion.

From the preceding we readily deduce that

$$\delta\dot{\boldsymbol{x}}(t) = \mathbf{A}_0(t)\delta\boldsymbol{x}(t) + \mathbf{B}_0(t)\delta\boldsymbol{u}(t) + \alpha_0\big(\delta\boldsymbol{x}(t), \delta\boldsymbol{u}(t)\big) \tag{11}$$

where

$$\mathbf{A}_0(t) \hat{=} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_0 \hat{=} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}^*(t)\\ \boldsymbol{u}^*(t)}} \tag{12}$$

is an $n \times n$ time-varying matrix which is obtained by evaluating the elements of the Jacobian matrix $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ along the known (precomputed) time functions $\boldsymbol{x}^*(t)$ and $\boldsymbol{u}^*(t)$.

$$\mathbf{B}_0(t) \hat{=} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_0 \hat{=} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}^*(t)\\ \boldsymbol{u}^*(t)}} \tag{13}$$

is an $n \times m$ time-varying matrix which is obtained by evaluating the elements of the Jacobian matrix $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ along the known (precomputed) time functions $\boldsymbol{x}^*(t)$ and $\boldsymbol{u}^*(t)$.

Equation (11) including the higher order terms represents the exact relationship between $\delta\boldsymbol{x}(t)$ and $\delta\boldsymbol{u}(t)$.

Linearized perturbation model is obtained by setting the higher order terms equal to zero in (11). We obtain:

$$\delta\dot{\boldsymbol{x}}(t) = \mathbf{A}_0(t)\,\delta\boldsymbol{x}(t) + \mathbf{B}_0(t)\delta\boldsymbol{u}(t) \tag{14}$$

which is a standard description of a linear time-varying system.

3.2 Linear-quadratic Approach to the Deterministic Controller Design

For given linear deterministic time-varying system (14) and given a fixed time interval of interest $t \in \langle t_0, T\rangle$ we shall find the control perturbation vector $\delta\boldsymbol{u}(t)$, $t \in \langle t_0, T\rangle$ such that the following deterministic quadratic cost functional is minimized:

$$\mathbf{J}_{TR} = \delta\boldsymbol{x}^\top(T)\mathbf{F}_0\delta\boldsymbol{x}(T) + \int_{t_0}^{T} \left[\delta\boldsymbol{x}^\top(t)\mathbf{Q}(t)\delta\boldsymbol{x}(t) + \delta\boldsymbol{u}^\top(t)\mathbf{R}(t)\delta\boldsymbol{u}(t)\right] \mathrm{d}t \tag{15}$$

where

$$\mathbf{F}_0 = \mathbf{F}^\top \geq \mathbf{0}, \qquad (n \times n \text{ matrix}), \tag{16}$$

$$\mathbf{Q}(t) = \mathbf{Q}^\top(t) \geq \mathbf{0}, \text{ for all } t \in \langle t_0, T\rangle \qquad (n \times n \text{ matrix}), \tag{17}$$

$$\mathbf{R}(t) = \mathbf{R}^\top(t) > \mathbf{0}, \text{ for all } t \in \langle t_0, T\rangle \qquad (m \times n \text{ matrix}). \tag{18}$$

The optimal control perturbation vector $\delta\boldsymbol{u}(t)$ is related to the state perturbation vector $\delta\boldsymbol{x}(t)$ by means of the linear time-varying feedback relationship

$$\delta\boldsymbol{u}(t) = -\mathbf{G}_{LQ}(t)\delta\boldsymbol{x}(t) \tag{19}$$

where $\mathbf{G}_{LQ}(t)$ is an $m \times n$ time-varying control gain matrix.

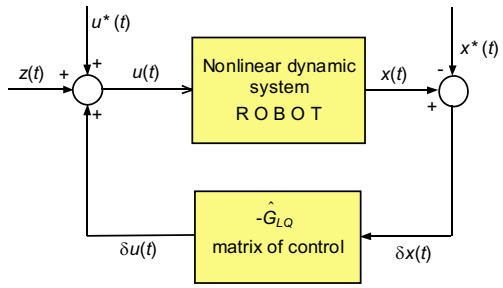

Fig. 4. The structure of deterministic feedback control system with disturbance $z(t)$.

The value of $\mathbf{G}_{LQ}(t)$ is given by

$$\mathbf{G}_{LQ}(t) = \mathbf{R}^{-1}\mathbf{B}_0^{\top}(t)\mathbf{K}(t) \tag{20}$$

where $n \times n$ matrix $\mathbf{K}(t)$ is the solution of the Riccati matrix differential equation

$$\dot{\mathbf{K}}(t) = -\mathbf{K}(t)\mathbf{A}_0(t) - \mathbf{A}_0^{\top}(t)\mathbf{K}(t) - \mathbf{Q}(t) + \mathbf{K}(t)\mathbf{B}_0(t)\mathbf{R}^{-1}(t)\mathbf{B}_0^{\top}(t)\mathbf{K}(t) \tag{21}$$

with boundary condition at the terminal time T:

$$\mathbf{K}(T) = \mathbf{F}_0\,. \tag{22}$$

The solution of the deterministic linear-quadratic problem provides us with a deterministic feedback design that attempts to null out deviations of the true state $\boldsymbol{x}(t)$ from its ideal response $\boldsymbol{x}^*(t)$, [1, 4, 5]. Figure 4 shows the structure of this of deterministic feedback control system for general time-varying case.

From a practical viewpoint this deterministic design is appealing because the control gain matrix $\mathbf{G}_{LQ}(t)$ can be precomputed completely. We put in line the steps that must be followed:

- Modelling
  Step 1: We consider a non-linear deterministic model of the robot $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)$.
  Step 2: The engineer determines the time functions $\boldsymbol{u}^*(t)$ and $\boldsymbol{x}^*(t)$ for all $t \in \langle t_0, T\rangle$, for example by selecting the cost functional (4) and solving programme optimal control problem by an algorithm using the method of sensitivity functions.
  Step 3: The engineer selects the weighting matrices $\mathbf{F}_0$, $\mathbf{Q}(t)$, $\mathbf{R}(t)$ in the quadratic criterion (15).
- Off-line computations
  Step 4: From the equations selected in Steps 1 and 2 we compute the matrices $\mathbf{A}_0(t)$ and $\mathbf{B}_0(t)$ according to (12) and (13); at this step the linear system (14) is determined.
  Step 5: The matrices $\mathbf{A}_0(t)$, $\mathbf{B}_0(t)$, $\mathbf{Q}(t)$, $\mathbf{R}(t)$ are coefficients of the Riccati equation (21) and $\mathbf{F}_0(t)$ is the boundary condition (22). Numerically integrate the Riccati equation (21) by function **ode45** in language Matlab we obtain the matrix $\mathbf{K}(t)$ and $\mathbf{G}_{LQ}(t)$ by (20).
- On-line computations:
  Step 6: We measure the true state $\boldsymbol{x}(t)$; subtract $\boldsymbol{x}^*(t)$ precomputed in Step 2 from $\boldsymbol{x}(t)$ to find $\delta\boldsymbol{x}(t)$.
  Step 7: We compute on-line $\delta\boldsymbol{u}(t)$ by

  $$\delta\boldsymbol{u}(t) = -\mathbf{G}_{LQ}(t)\delta\mathbf{x}(t)\,;$$

  only a matrix vector multiplication is required in real time; since $\mathbf{G}_{LQ}(t)$ has been precomputed in Step 5.
  Step 8: We compute the true control input $\boldsymbol{u}(t)$ by

  $$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + \delta\boldsymbol{u}(t)\,.$$

  $\boldsymbol{u}^*(t)$ was precomputed in Step 2.

Selection of the weighting matrices in the quadratic criterion (15) is not a simple matter. Usually they are selected by the designer on the basis of engineering experience. In most practical applications $\mathbf{F}_0$, $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ are selected to be diagonal.

4 SOLVING TRACKING PROBLEM USING NEURAL MODELS

The results of the classical approach written in the third section will be used to design an optimal trajectory tracking neuro-controller (OTNC) for non-linear dynamic system — robot with a quadratic cost function, [2, 6 and 8].

The OTNC is made of two controllers: feedforward neuro controller (FFNC) and feedback neuro-controller (FBNC). The FFNC controls the steady-state output of the plant and is trained as *inverse neural model* (Fig. 5) by the well-known back propagation error algorithm (BPEA). The training set for the FFNC is computed solving two-bounded problem using sensitive functions, (Fig. 2, Fig. 3). The FBNC controls the transient output of the system and is trained as *forward neural model* by BPEA, (Fig. 6). The FBNC generates the transient control input $\delta\boldsymbol{u}(k)$ to stabilize error dynamics along the optimal trajectories while minimizing the cost function (15). The training patterns are obtained by solving nonlinear Riccati differential equations, [6]. The architecture for the optimal tracking control problem using neural networks **NN1** (forward neural model as feedback controller) and **NN2** (inverse neural model as a feedforward controller) is in Fig. 7.

5 EXAMPLE

To test the problem of tracking we considered a dynamic model of robot (1), Steps 1 and 2 in Section 3.
Step 3: The weighting matrices $\mathbf{F}_0$, $\mathbf{Q}$ and $\mathbf{R}$ are diagonal, for example:

$$\begin{aligned} q_{ii} &= 1\,, && \text{for } i = 1, \ldots, 4\,, \\ r_{ii} &= 0.05 : 0.1 && \text{for } i = 1, 2\,. \end{aligned} \tag{23}$$

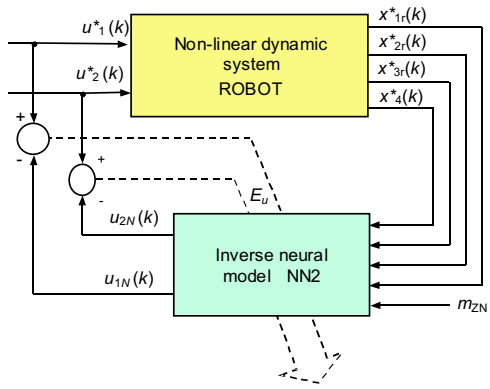

Fig. 5. Training neural network NN2 as inverse model of non-linear system.

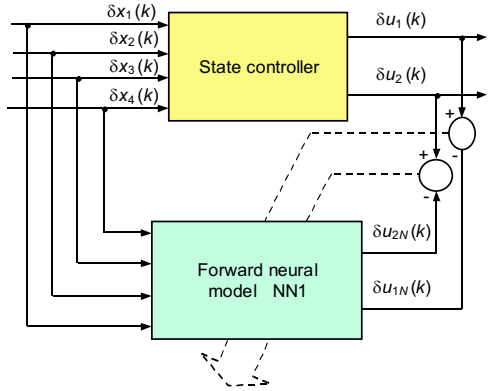

Fig. 6. Training neural network NN1 as forward model of the non-linear system.

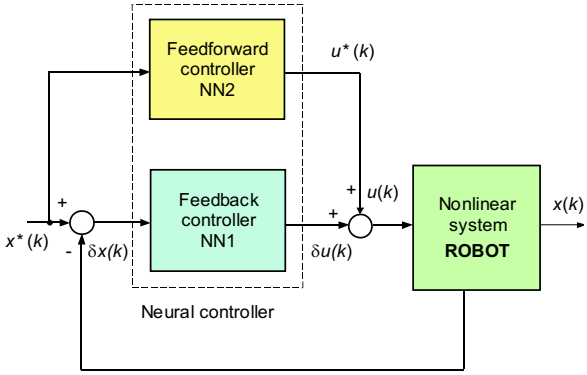

Fig. 7. The control structure for problem tracking target trajectories by neural networks NN1 and NN2.

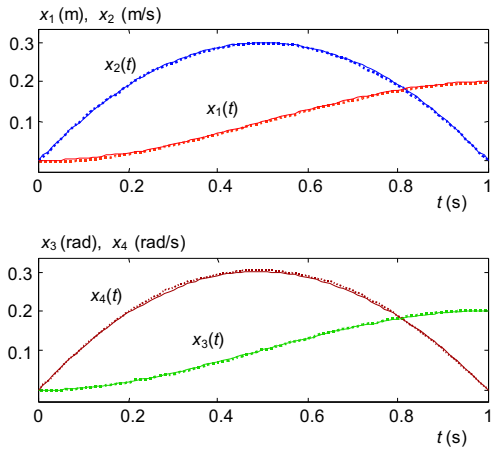

Fig. 8. Tracking optimal state trajectories $x_1^*(t), x_2^*(t), x_3^*(t), x_4^*(t)$ for time-variant case classical approach — LQ design, the real state of the system is denoted by dotted line

Step 4: Computing the matrices $\mathbf{A}_0(t)$, $\mathbf{B}_0(t)$

$$\mathbf{A}_0(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ A_{21} & 0 & 0 & A_{24} \\ 0 & 0 & 0 & 1 \\ A_{41} & A_{42} & 0 & A_{44} \end{bmatrix}_{\substack{\mathbf{x}^*(t) \\ \mathbf{u}^*(t)}} \tag{24}$$

where

$$A_{21} = \frac{m_b}{m_r} x_4^2(t) \quad A_{24} = \frac{2m_b}{m_r} x_1(t) x_4(t)$$

$$A_{41} = -\frac{2m_b x_1(t)\big(u_2(t)K_2 - 2m_b x_1(t) x_2(t) x_4(t)\big)}{H^2} - \frac{2m_b x_2(t) x_4(t)}{H}, \quad H = I_{23} + m_b x_1^2(t), \tag{25}$$

$$A_{42} = -\frac{2m_b x_1(t) x_4(t)}{H}, \quad A_{44} = -\frac{2m_b x_1(t) x_2(t)}{H},$$

$$\mathbf{B}_0 = \begin{bmatrix} 0 & 0 \\ B_{21} & 0 \\ 0 & 0 \\ 0 & B_{42} \end{bmatrix}_{\substack{\mathbf{x}^*(t) \\ \mathbf{u}^*(t)}} \tag{26}$$

where

$$B_{21} = \frac{K_1}{m_r}, \quad B_{42} = \frac{K_2}{I_{23} + m_b x_1^2(t)}. \tag{27}$$

Step 5: Numerically integrate the Riccati equation (21) by function **ode45** in simulation language Matlab we obtain the matrix $\mathbf{K}(t)$ and $\mathbf{G}_{LQ}(t)$ by (20)

Step 6: We compute the regulate error:

$$\delta x_i(t) = x_i(t) - x_i^*(t) \qquad \text{for } i = 1, \ldots, 4.$$

By **step 7** and **step 8** we compute control input:

$$\begin{aligned} \boldsymbol{u}_1(t) &= \boldsymbol{u}_1^* + \delta \boldsymbol{u}_1(t), \\ \boldsymbol{u}_2(t) &= \boldsymbol{u}_2^* + \delta \boldsymbol{u}_2(t). \end{aligned} \tag{28}$$

Results of simulations of classical approach LQ design are in Fig. 8 and Fig. 9 and the results of simulations using neural networks NN1, NN2 for solving tracking problem are in Fig. 10 and Fig. 11.

In Figs. 8 and 9 we can see tracking optimal trajectories x*(t) for time-variant case by classical approach of LQ design. In Figs. 10 and 11 we can see the tracking by the optimal tracking neural controller composed from two off-line trained neural networks NN1 and NN2.

For both cases we consider a defect in the mass of the weight m_z from 35 kg to 40 kg and white noise as perturbation input of the non-linear system.

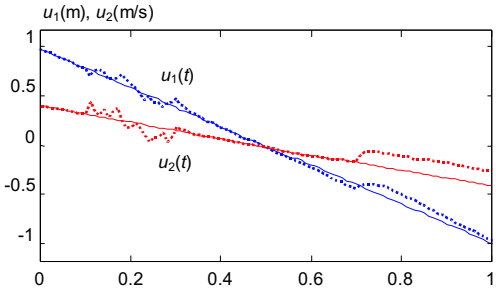

Fig. 9. The control inputs $\boldsymbol{u}_1(t), \boldsymbol{u}_2(t)$ classical approach — LQ design, the real control inputs are denoted dotted line

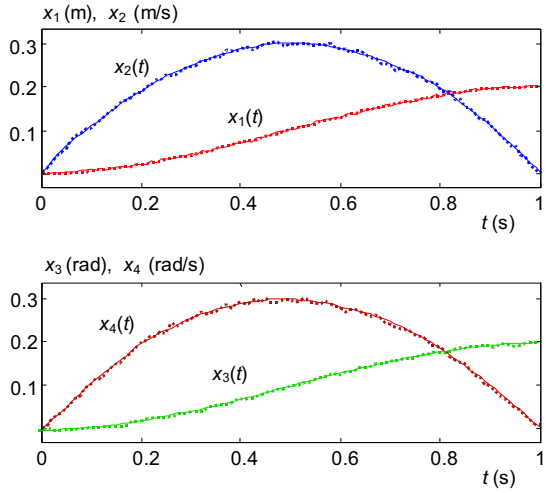

Fig. 10. Tracking optimal trajectories $x_1^*(t), x_2^*(t), x_3^*(t), x_4^*(t)$ by neuro-controller OTNC, the real state of the system is denoted dotted line

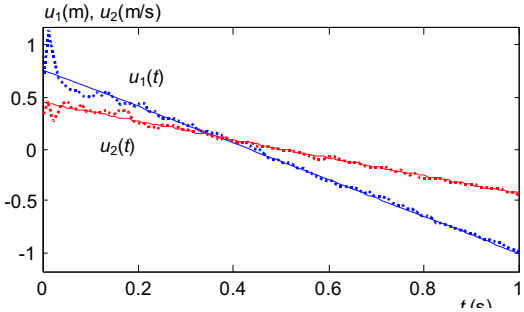

Fig. 11. The control inputs $u_1(t), u_2(t)$ — neuro controller OTNC, the real state of the system is denoted dotted line

6 CONCLUSION

From the results we can see that the proposed methodology is useful as a control method, where the results from the classical approach programme optimal control (POC) and feedback optimal LQ control (FOC) can be used to design an optimal-tracking neuro-controller in feedforward control structure. The use of neural networks with learning ability helps the controller design to be rather flexible and robust, especially when the dynamics of the system is complex and non-linear and with time-variable parameters. In this article we show that neural networks offer promising possibilities for providing better solutions to robot tracking problems, primarily because of their excellent capability to learn any complex mapping from training examples.

Acknowledgements

The paper was worked out as a part of the solution of the scientific project *Optimization of the Process Control Based on the Use of Informatics Methods*, number 1/0373/03.

References

[1] ATHANS, M.—FALB, P. L.: Optimal Control, McGraw-Hill, New York, 1966.

[2] BEHERA, L.—GOPAL, M.—CHAUDHURY, S.: On Adaptive Trajectory Tracking of a Robot Manipulator Using Inversion of its Neural Emulator, IEEE Transaction on Neural Networks **7** No. 6 (1996), 1401–1414.

[3] HRUBINA, K.—JADLOVSKÁ, A.: Optimal Control and Approximation of Variational Inequalities, KYBERNETES, the International Journal of Systems & Cybernetics, MBC University Press of England **31** No. 9/10 (2002), 1401–1408.

[4] JADLOVSKÁ, A.—SARNOVSKÝ, J.: Numeric Solution Optimal Control Problems Using Sensitive Functions, AT&P Journal **3** No. 4 (1996), 35–38.

[5] JADLOVSKÁ, A.—SARNOVSKÝ, J.: Using Dynamic Neural Models in Feedforward Control Structure, Automatizace **46** No. 4 (2003), 236–241.

[6] JADLOVSKÁ, A.: Modelling and Control Dynamic Processes Using Neural Networks, FEI TU Košice, Informatech 2003,. (in Slovak)

[7] NEUMAN, P.: Optimal Control of Industrial Robot, Automatic Control Journal No. 6 (1980).

[8] OZAKI, T.—SUZUKI, T.—OKUMA, S.—UCHIKAWA, Y.: Trajectory Control of Robotic Manipulators Using Neural Networks, IEEE Transactions on Industrial Electronics **38** No. 3 (1991).

Received 26 January 2004

Anna Jadlovská (MSc, PhD), was born in Banská Bystrica, Slovakia, on October 29, 1960. She received MSc degree in technical cybernetics from the Faculty of Electrical Engineering of the Technical University Košice in 1984 and PhD degree in Automation and Control in 2001 from the same university. Her PhD thesis dealt with modelling and control of non-linear processes using neural networks. Since 1994 she has worked at the Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics at TU in Košice as an assistant professor. Her main research and teaching activities include problems adaptive and optimal control — especially predictive control with constrains for non-linear processes with time-variant parameters using neural networks (Intelligent Control Design). She is author of articles and contributions published in journals and international conference proceedings. She is also co-author of three monographs.