92-16 I

Stabilizing neural controllers with application to control of an inverted pendulum*

Johan Suykens, Bart De Moor†

Katholieke Universiteit Leuven
Department of Electrical Engineering
ESAT-SISTA
Kardinaal Mercierlaan 94
B-3001 Leuven (Heverlee), Belgium
tel: 32/16/22 09 31 fax: 32/16/22 18 55
e-mail: suykens@esat.kuleuven.ac.be

ESAT-SISTA Report 16

March 24, 1992

*This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven, in the framework of a Concerted Action Project of the Flemish Community, entitled *Applicable Neural Networks*. The scientific responsibility is assumed by its authors.

†Research Associate of the Belgian National Fund for Scientific Research

Abstract

A design method for stabilizing nonlinear systems by feedforward neural networks is proposed. A switching between equilibrium points can be realized with guaranteed stabilization around the 'target' equilibrium point. As a test case control tasks for an inverted pendulum are considered. First it is shown that one neuron is sufficient for keeping the pole 'up' in a relatively large region around the equilibrium point. In this case all weights are imposed by a linear controller design method, like LQR or pole placement. One can use a multilayer neural network too, but then there remains additional freedom in the choice of the weights. This property of redundancy can be exploited in the swinging up and swinging down problem. Optimization criteria can be formulated with constraints on the weights that guarantee a stable 'target' equilibrium point.

Keywords: feedforward neural nets, nonlinear optimization, Linear Quadratic Regulator (LQR)

1 Introduction

In many important applications the control engineer is confronted with the following problem: given a multivariable nonlinear plant that may operate around several equilibrium points. Design a control strategy that not only stabilizes the plant around these equilibrium points, but that also is able to switch from one operating point to another.

In this paper we will propose a general design philosophy and then illustrate it on a par-

ticular example.

The general idea is as follows:

- First a nonlinear model is obtained by whatever modelling method that is available (nonlinear system identification, physical laws, bondgraphs, etc. ..)
- Next the equilibrium points are identified. Around each of these points the system is linearized and a linear stabilizing controller is calculated for each specific operation point (e.g. by poleplacement, H_2, H_∞, etc. ...)
- A general parametrized control law is proposed for the switching from one operating point to another. This control law is overparametrized but the parameter vector is constrained in the following sense: in the neighborhood of each operating point, the control law coincides with the linear stabilizing controller around that specific point. The additional freedom in the parameters is used to enforce the desired switching from one point to another.
- If there are more than two switching trajectories, one can repeat the whole strategy for each pair of operating points and design an appropriate switching circuitry that governs the switchings.

An interesting academic test example for this kind of problem is an inverted pendulum. Up till now, most of the literature on control of an inverted pendulum is concerned with the problem of keeping a pole 'up' [3]. Besides that problem we study in this paper the difficult control task of swinging the pole from 'down' to 'up' and to stabilize it there. A

classical approach to control the inverted pendulum is a combination of two controllers: a first controller has to swing up the pendulum and a second one keeps it around the equilibrium point. The proposed neural controller can combine these two phases. Our approach introduces constraints on the set of weights such that a closed-loop stability in the neighborhood of the 'target' equilibrium point is guaranteed. These constraints are the result of a linear controller design. The swinging up and swinging down problems are then solved by formulating optimization criteria that have the linear controller design results as constraints. Finally one single neural controller is constructed that can handle both the swinging up and down control tasks.

This paper is organized as follows:

Section 2 describes the general design method for stabilizing nonlinear systems by a feed-forward neural network. In section 3 an inverted pendulum is taken as a test case. Section 4 shows that one neuron is sufficient to keep the pole 'up'. Sections 5 and 6 exploit the redundancy of multilayer neural networks to solve the switching problems from 'down' to 'up' and 'up' to 'down' by formulating the appropriate optimization problems constrained by the linear controller design results. In section 8 one neural controller is proposed that combines the two switching tasks with guaranteed stabilization around the target equilibrium points.

2 Nonlinear parametrized static state feedback by feedforward neural networks

In this section a general outline is proposed of a method for stabilizing nonlinear systems by means of feedforward neural networks.

Given a single input nonlinear system

$$\dot{x} = f(x, u) \tag{1}$$

with state vector $x \in \mathbb{R}^n$, input $u \in \mathbb{R}$ and f a vector field defined on $\mathbb{R}^n$. Suppose the control task is to bring the state x from $x(0)$ to a target equilibrium point x_{eq}. Hence, let us introduce a nonlinear parametrized static state feedback law

$$u = g(x, w) \tag{2}$$

where $w \in \mathbb{R}^p$ is a parameter vector to be determined. In the case of a feedforward neural network these parameters become the connection weights. For a neural net with one hidden layer eqn.(2) could be given e.g. by

$$u = tanh(w^t.tanh(V.x)) \tag{3}$$

with the set of weights $\{w, V\}$ to be determined.

The design procedure is as follows:

1. Design a linear controller around the target equilibrium point x_{eq} based on the linearized model by e.g. LQR or pole placement. The nonlinear control law (2) becomes the linearized controller around x_{eq}.

2. The parameters (weights) serve to satisfy other performance criteria such as the transition from $x(0)$ to x_{eq}. This results into an optimization problem constrained by the linear controller design results.

Remark that in the case where the nonlinear system (1) has two or more equilibrium points a control task could be the transition between two of this points. We will solve this problem for the special case of an inverted pendulum.

3 Test case: inverted pendulum

A state-space model [3] of the inverted pendulum (Fig.1) may be presented in the form

$$\dot{x} = f(x) + b(x).u \tag{4}$$

with state $x \in \mathbb{R}^4$ and input $u \in \mathbb{R}$ and

$$f(x) = \begin{bmatrix} x_2 \\ \frac{\frac{4}{3}mlx_4^2 sinx_3 - \frac{mg}{2}sin(2x_3)}{\frac{4}{3}m_t - mcos^2 x_3} \\ x_4 \\ \frac{m_t g sinx_3 - \frac{ml}{2}x_4^2 sin(2x_3)}{l(\frac{4}{3}m_t - mcos^2 x_3)} \end{bmatrix}, b(x) = \begin{bmatrix} 0 \\ \frac{4}{3}\cdot\frac{1}{\frac{4}{3}m_t - mcos^2 x_3} \\ 0 \\ -\frac{cosx_3}{l(\frac{4}{3}m_t - mcos^2 x_3)} \end{bmatrix} \tag{5}$$

In this model friction is not taken into account. The states x_1, x_2, x_3, x_4 are position and velocity of the cart, angle of the pole with the vertical and rate of change of the angle. The input signal u is the force applied to the cart's center of mass. The symbols m, m_t, l, g mean respectively mass of the pole, total mass of cart and pole, half pole length and the acceleration due to gravity. The input signal u is constrained as $u_{min} \leq u \leq u_{max}$. Remark

that in the autonomous case $x = [0\,0\,0\,0]^t$ and $x = [0\,0\,\pi\,0]^t$ are equilibrium points and we call them respectively eq^+ and eq^-.

4 Stabilization with one neuron

In this section we will investigate the possibilities that one neuron can offer for stabilizing the inverted pendulum around eq^+. We will take the following model for a neuron

$$u = \alpha.tanh(w^t x) \tag{6}$$

where $w \in \mathbb{R}^4$ is the weight vector and α a positive constant. Remark that u is bounded by (6) as $-\alpha < u < \alpha$. In closed-loop (4) and (6) become

$$\dot{x} = f(x) + b(x).\alpha.tanh(w^t x) \tag{7}$$

Equilibrium points of (7) are given by $\dot{x} = 0$. The stability of the equilibrium point $x = 0$ is determined by the eigenvalues of the Jacobian matrix J(x) evaluated at $x = 0$

$$J(x) = f_x + b_x.\alpha.tanh(w^t x) + b(x).\alpha.(1 - tanh^2(w^t x)).w^t \tag{8}$$

$$J(0) = f_0 + b(0).\alpha.w^t \tag{9}$$

Here f_x and b_x denote matrices that contain the partial derivatives with respect to x_i; f_0 and $b(0)$ are given by

$$f_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{\frac{4}{3}m_t - m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m_t g}{l(\frac{4}{3}m_t - m)} & 0 \end{bmatrix}, b(0) = \begin{bmatrix} 0 \\ \frac{4}{3}.\frac{1}{\frac{4}{3}m_t - m} \\ 0 \\ -\frac{1}{l(\frac{4}{3}m_t - m)} \end{bmatrix} \tag{10}$$

The general idea now is that we apply an LQR based design [2] on the linearized model around $x = 0$ which will give us values for α and w, which are then applied in the nonlinear feedback law (6). We search for a linear static state feedback $u = -k_{lqr}^t.x$ that can stabilize the linearized system

$$\dot{x} = f_0.x + b(0).u \tag{11}$$

by minimizing the cost function

$$C_{lqr} = \int_0^\infty (x^t Q x + u^t R u).dt \tag{12}$$

where Q and R are given positive definite symmetric matrices. The solution to this problem is given by

$$k_{lqr}^t = R^{-1}.b(0)^t.P \tag{13}$$

where P is the stabilizing solution to the matrix algebraic Riccati equation

$$O = Pf_0 + f_0^t P - Pb(0)R^{-1}b(0)^t P + Q \tag{14}$$

From (9) and (13) it follows that

$$k_{lqr}^t = -\alpha.w^t \tag{15}$$

Hence the choice of α and w is determined by k_{lqr}. In Fig.2 simulation results are shown for parameters m, m_t, l equal to $0.1, 1.1$ and 0.5. Furthermore $\alpha = 10$, $Q = I_4$ and $R = 1$ and initial state $x(0) = [0\,0\,\frac{\pi}{6}\,0]^t$.

The weight vector w is equal to $[0.1000\ 0.2303\ 3.1894\ 0.8178]^t$. In all simulations a trapezoidal integration rule with constant step length equal to 0.05 was used. The results are compared with linear static state feedback (LQR) with $u = -k_{lqr}^t.x$. The region around the

equilibrium point where the LQR based design strategy can be applied, seems to be relatively large. Simulations indicate that an increasing value α (and from (15) a corresponding decrease in the elements of w) enlarges the stabilizing region around the equilibrium point. In Fig.3 state space trajectories are shown for 2^4 initial states $x(0) = [\pm 2 \pm 0.5 \pm \frac{\pi}{6} \pm 0.3]^t$.

5 Extension to multilayer neural networks

In the case of one neuron all weights are completely determined by the LQR design. Now we will see that for multilayer neural nets the LQR design imposes constraints on the set of weights and an additional freedom in the choice of weights is left.

We can model a neural net with one hidden layer as

$$u = \alpha.tanh(\sum_{i=1}^{n_2} w_i.tanh(\sum_{j=1}^{n_1} v_{ij}x_j)) \tag{16}$$

where the connections are represented by $w = [w_i]$ and $V = [v_{ij}]$; $n_1 = n$ the number of state variables (full state feedback), n_2 is the number of neurons in the hidden layer. An obvious extension to n_h hidden layers is

$$u = \alpha.tanh(\sum_{i=1}^{n_{n_h+1}} w_i.tanh(\sum_{j=1}^{n_{n_h}} v_{ij}^{(n_h)}...tanh(\sum_{k=1}^{n_1} v_{jk}^{(1)}x_k)...)) \tag{17}$$

Remark that one hidden layer is sufficient for approximating a general nonlinear mapping [4] [5]. In the case of one hidden layer a linearization in closed-loop yields

$$J(0) = f_0 + b(0).\alpha.w^t.V \tag{18}$$

A LQR design like in the 'one neuron case' results into

$$k_{lqr}^t = -\alpha.w^t.V \tag{19}$$

It is clear that w and V are not completely determined by this design. For the n_h hidden layer case this results into

$$k_{lqr}^t = -\alpha.w^t.V^{(n_h)}.V^{(n_h-1)}...V^{(2)}.V^{(1)} \tag{20}$$

Because of the degree of freedom in the choice of the connections we can combine the LQR design with some other optimization procedure. One can discretize the closed-loop system (4)(17) as

$$x_{k+1} = x_k + h.\phi(x_k, w, V^{(n_h)}, V^{(n_h-1)}...V^{(2)}.V^{(1)}) \tag{21}$$

where ϕ depends upon f and b and the discretization rule (Euler, trapezoidal, ...) and h is a constant step length. A quadratic cost function $C = \sum_{k=1}^{N} x_k^t x_k$ or $C = x_N^t x_N$ can be computed as

$$C(x_0, w, V^{(n_h)}, V^{(n_h-1)}...V^{(2)}.V^{(1)}) \tag{22}$$

This unconstrained problem can then be constrained by the LQR design (20), guaranteeing a stabilizing controller.

In practice, this optimization problem suffers from local minima and can only be solved by means of global optimization routines.

6 Control task of switching from 'down' to 'up'

For solving the control task of bringing the pole from 'down' ($eq^- = [0\,0\,\pi\,0]^t$) to 'up' ($eq^+ = [0\,0\,0\,0]^t$), we formulated the problem as in (22). A neural network with one hidden layer was proposed (see Fig.4). The neural network consists of one neuron in the output

layer and two neurons in the hidden layer ($n_1 = 4$, $n_2 = 2$). The control signal is given by

$$u = \alpha.tanh(w^t.tanh(V.x)) \tag{23}$$

The 10 parameters for the optimization problem are the elements of $w(2 \times 1)$ and $V(2 \times 4)$. We have chosen as penalty function the criterion

$$C = x_N^t x_N \tag{24}$$

which means terminal control. In practice this penalty function can be calculated immediately from the integration of the nonlinear differential equations (4) under the feedback law (23) starting with initial state eq^- and a certain parameter set. The parameters are constrained by 4 equations

$$k_{lqr}^t = -\alpha.w^t.V \tag{25}$$

We have chosen for an exhaustive search method combined with an elimination strategy for solving this optimization problem (of course other global optimization methods like e.g. genetic algorithms can be used for this problem). Reduction of the dimension of the search space can be done thanks to the constraints (25). We generated random weights (according to a Gaussian distribution with variance 2.25) that obey these constraints (25). This can be done by splitting k_{lqr} into two parts

$$-\frac{k_{lqr}^t}{\alpha} = s^t = [s_1^t\ s_2^t] = w^t.[V_1\ V_2] \tag{26}$$

with dimension of s_1 and s_2 equal to 2×1. Let us introduce a matrix T which satisfies the relation $V_2 = V_1.T$ and hence $s_1^t.T = s_2^t$. Two elements of T may be chosen freely (e.g.

elements t_{11} and t_{12}). The other elements are calculated from

$$t_{21} = \frac{s_{21} - s_{11}.t_{11}}{s_{12}}$$
$$t_{22} = \frac{s_{22} - s_{12}.t_{12}}{s_{12}}$$

Now choose V_1 according to the same Gaussian distribution and calculate V_2 from $V_2 = V_1.T$ and $w^t = s_2^t.V_2^{-1}$. In Fig.5 the simulation results are shown for initial state $x(0) = [0\,0\,\pi\,0]^t$. A trapezoidal integration rule was used with constant step length equal to 0.05 (200 steps). Fig.6 gives the evolution of the pole during the 'swinging up'. The optimal weights (after 100,000 simulations) are

$$w = \begin{bmatrix} -0.4452 \\ -1.0738 \end{bmatrix}, \; V = \begin{bmatrix} -0.9239 & -0.6029 & -7.4761 & -0.9926 \\ 0.2899 & 0.0355 & 0.1294 & -0.3500 \end{bmatrix} \tag{27}$$

Parameters are chosen as $m = 0.1, m_t = 1.1, l = 0.5, \alpha = 10, Q = I_4, R = 1$ and k_{lqr} is the same as in the 'one-neuron case' of section 4. The neural network performs this task only for the given initial state eq^-. For other initial states a new set of weights is to be determined. Nevertheless, with respect to the first initial state variable there exist some robustness: when $x_1(0)$ belongs approximately to the interval $[0, 2]$ the control task remains performed well. In Fig.7 results are given for initial state $x(0) = [2\,0\,\pi\,0]^t$. Experiments also show that for $\alpha = 10$, α may be increased to $\alpha^* = r.\alpha$ with $1 \leq r \leq 4$ (provided that the elements change into $V^* = \frac{V}{r}$ because of (19)).

In order to test the robustness against process and measurement noise we did some tests with additive white noise (normal Gaussian distributed) and measurement noise on state x, both with covariance matrix equal to $\sigma^2.I_4$. The neural controller performed well for

$\sigma < 0.15$.

7 Control task of switching from 'up' to down'

Just like in the swinging up problem the control task of bringing the pole from eq^+ to eq^- can be formulated as a nonlinear constrained optimization problem. Instead of terminal control (24) we used a quadratic regulation criterion

$$C = \sum_{k=1}^{N} x_k^t x_k \tag{28}$$

which provides some damping of the oscillations when going from eq^+ to eq^-. Remark that it is not needed to include terms related to u into expression (28) because u is restricted by the relation $-\alpha < u_k < \alpha$. Again a feedforward neural network consisting of three neurons with two neurons in the hidden layer was proposed (see Fig.4). The feedback law (23) now changes into

$$u = \alpha.tanh(w^t.tanh(V.(x - eq^-))) \tag{29}$$

The constraints to be placed on (28) follow from the linear controller design around eq^-. An LQR design is based on the linearized model around $eq^- = [0\,0\,\pi\,0]^t$ with

$$f_{eq^-} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{\frac{4}{3}m_t - m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{m_t g}{l(\frac{4}{3}m_t - m)} & 0 \end{bmatrix}, b(eq^-) = \begin{bmatrix} 0 \\ \frac{4}{3}.\frac{1}{\frac{4}{3}m_t - m} \\ 0 \\ \frac{1}{l(\frac{4}{3}m_t - m)} \end{bmatrix} \tag{30}$$

which results in a k_{lqr} according (12), (13) and (14). Constraints on the optimization criterion (28) are then given by

$$k_{lqr}^t = -\alpha.w^t.V \tag{31}$$

The same strategy of generating random weights (Gaussian normal distribution with variance 2.25) that obey these constrained was used. We took $Q = I_4$, $R = 0.01$, $\alpha = 10$. The result of the LQR design was $k_{lqr} = [10.0000\ 13.8413\ 34.1298\ 4.9217]^t$. For the simulations a trapezoidal integration rule with constant step length equal to 0.02 (300 steps) was used. As best result after 100,000 runs we obtained

$$w = \begin{bmatrix} 8.5034 \\ \\ 8.0219 \end{bmatrix}, \ V = \begin{bmatrix} -1.2689 & 0.3497 & 0.6719 & 0.1552 \\ \\ 1.2204 & -0.5432 & -1.1377 & -0.2259 \end{bmatrix} \tag{32}$$

In Fig.8 the corresponding state space trajectories and the control signal are given. In Fig.9 the evolution of the pole during the swinging down is given.

8 Neural controller realizing both 'down' to 'up' and 'up' to 'down' switching

In Fig.10 a neural controller is proposed that can perform the swinging up problem as well as the swinging down problem. The control signal u is equal to $\beta(r).u_1 + \beta(-r).u_2$ with

$$u_1 = \alpha.tanh(w_{up}^t.tanh(V_{up}.x)) \tag{33}$$

$$u_2 = \alpha.tanh(w_{down}^t.tanh(V_{down}.(x - eq^-))) \tag{34}$$

where the reference input r specifies the desired control task and u_1, u_2 correspond respectively to the expressions (23) and (29). The weights w_{up}, w_{down}, V_{up} and V_{down} are the optimal weights obtained in the previous sections of this text. The function $\beta(r)$ is the step function

$$\beta(r) = 1 \; if \; r > 1 \tag{35}$$

$$= 0 \; if \; r \leq 0 \tag{36}$$

If $r = 1$ that part of the neural network responsible for the switching up is activated and $u = u_1$. For $r = -1$ the switching down controller is selected and $u = u_2$.

9 Conclusion

In this paper we proposed a general method for stabilizing nonlinear systems by means of feedforward neural networks. This was applied to the special case of an inverted pendulum. In the 'one neuron case' the weights are completely determined by this procedure. In the multilayer case it delivers constraints on them. Due to this degree of freedom in the choice of the weights one can combine an LQR design with some other optimization method. That optimization may then lead to the solution of the 'swinging up' or 'swinging down' problem, while the LQR design guarantees stability around the target equilibrium point. The strength of the method lies in the fact that thanks to the nonlinear control, realized by a neural net, a trajectory passing trough the whole nonlinear region can be generated together with a guaranteed stabilization around the target equilibrium point. Finally a

relatively simple controller consisting of six neurons, capable of performing the swinging up and down problems for the inverted pendulum, was constructed.

References

[1] Miller W., Sutton R., Werbos P., *Neural networks for control*, Cambridge, MA: M.I.T. Press, 1990.

[2] Bryson A.E., Ho Y.C., *Applied optimal control*, Waltham, MA: Blaisdel, 1969.

[3] Barto A.G., Sutton R.S., Anderson C.W., *Neuronlike adaptive elements that can solve difficult learning control problems*, IEEE Trans. on Syst, Man and Cyb, Vol. SMC-13, No.5, Sept/Oct. 1983, pp 834-846.

[4] Hornik K. *Multilayer Feedforward Networks are universal approximators*, Neural Networks, Vol. 2, pp.359-366, 1989.

[5] Funahashi K.I., *On the Approximate Realization of continuous Mappings by Neural Networks*, Neural Networks, Vol.2, pp 183-192, 1989.

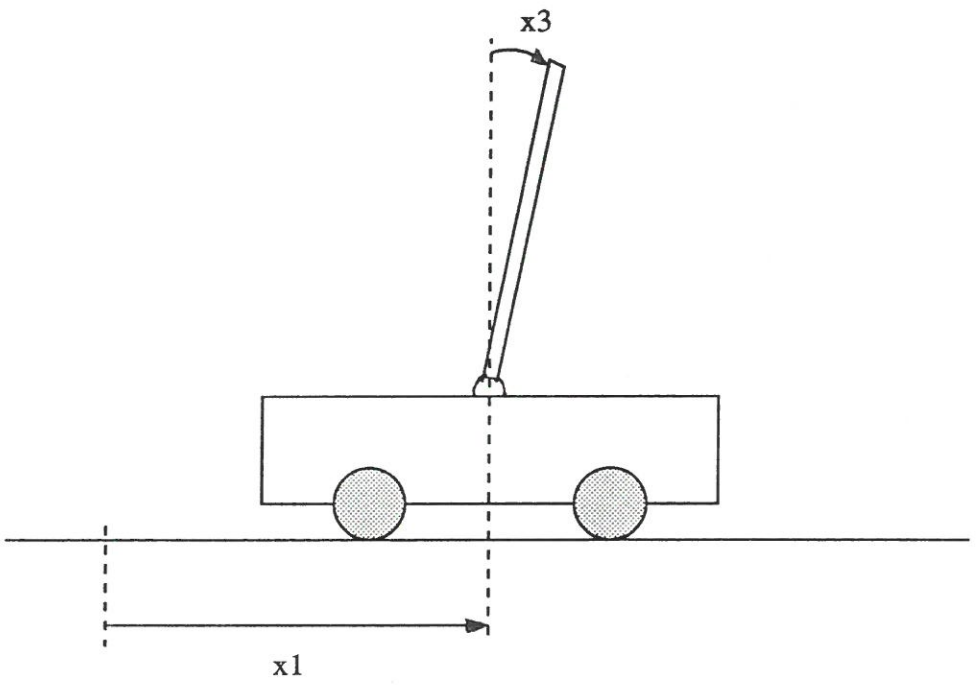

Figure 1: *Inverted Pendulum*

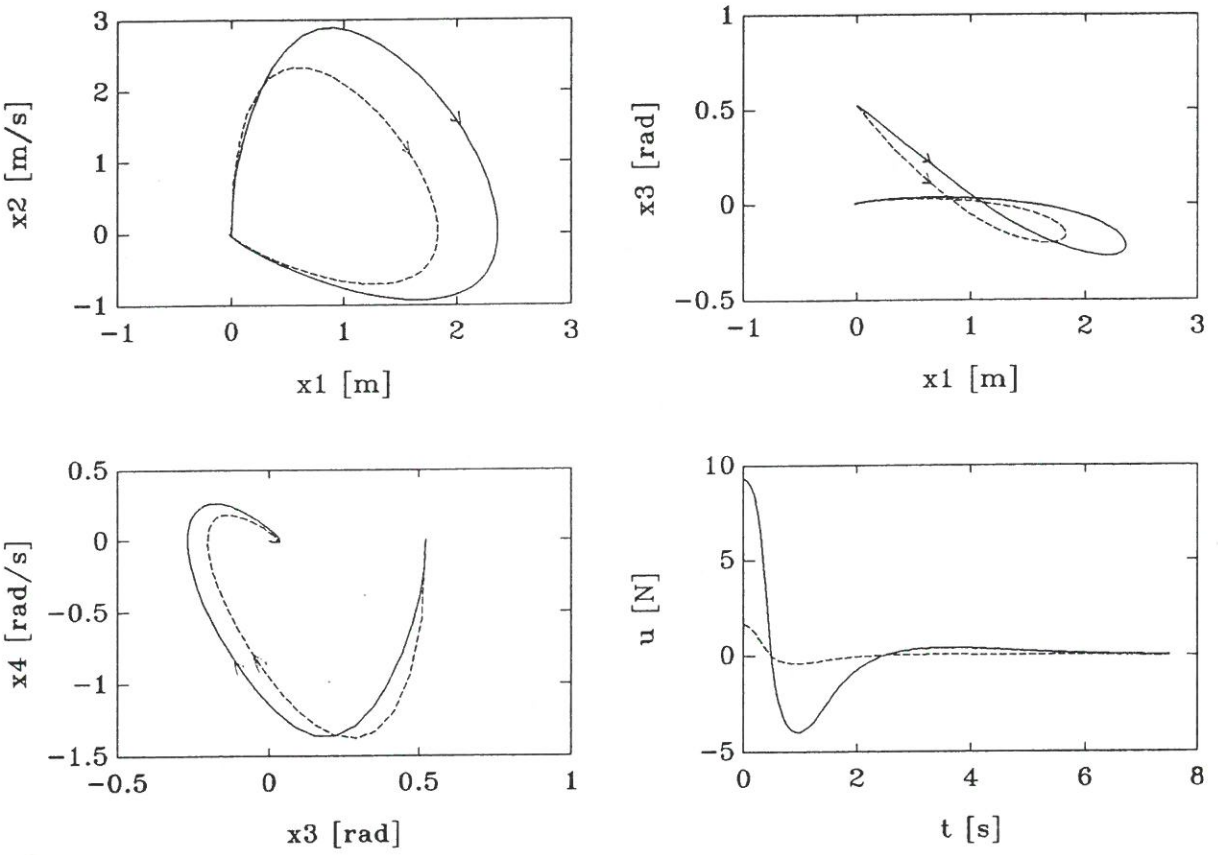

Figure 2: *Stabilization of inverted pendulum with one neuron based on LQR design (full line), compared with linear static state feedback (LQR) (dashed line) with initial state* $x(0) = [0\,0\,\frac{\pi}{6}\,0]^t$, *(*$m = 0.1, m_t = 1.1, l = 0.5, \alpha = 10, Q = I_4, R = 1$*)*

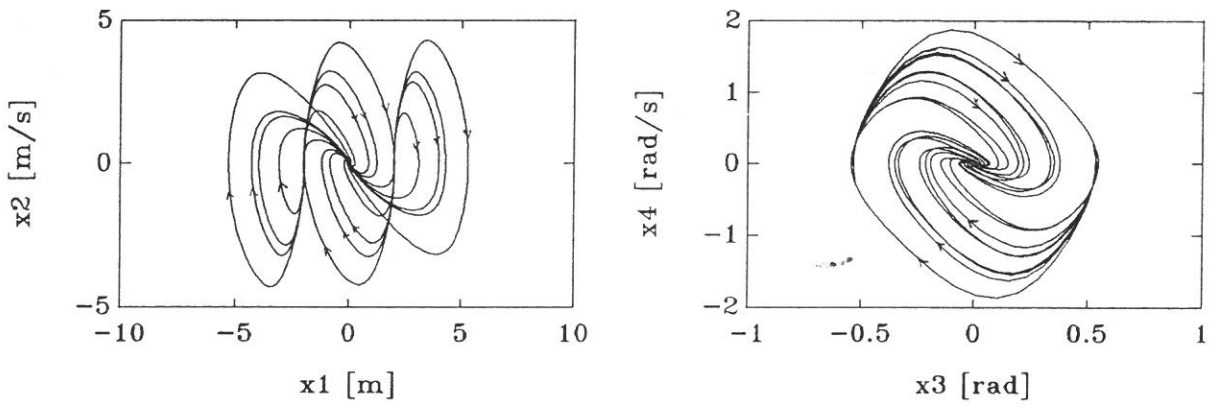

Figure 3: *Stabilization of inverted pendulum with one neuron based on LQR design for* 2^4 *initial states* $x(0) = [\pm 2 \pm 0.5 \pm \frac{\pi}{6} \pm 0.3]^t$ *(see Fig.2 for parameters).*

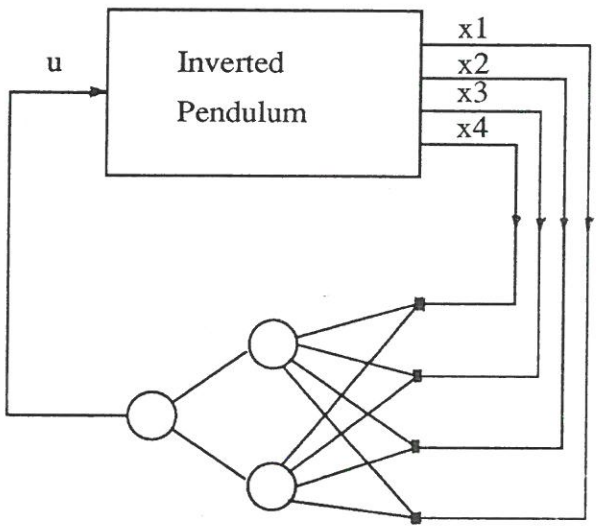

Figure 4: *Inverted pendulum controlled by a neural net with three neurons (one hidden layer with two neurons)*

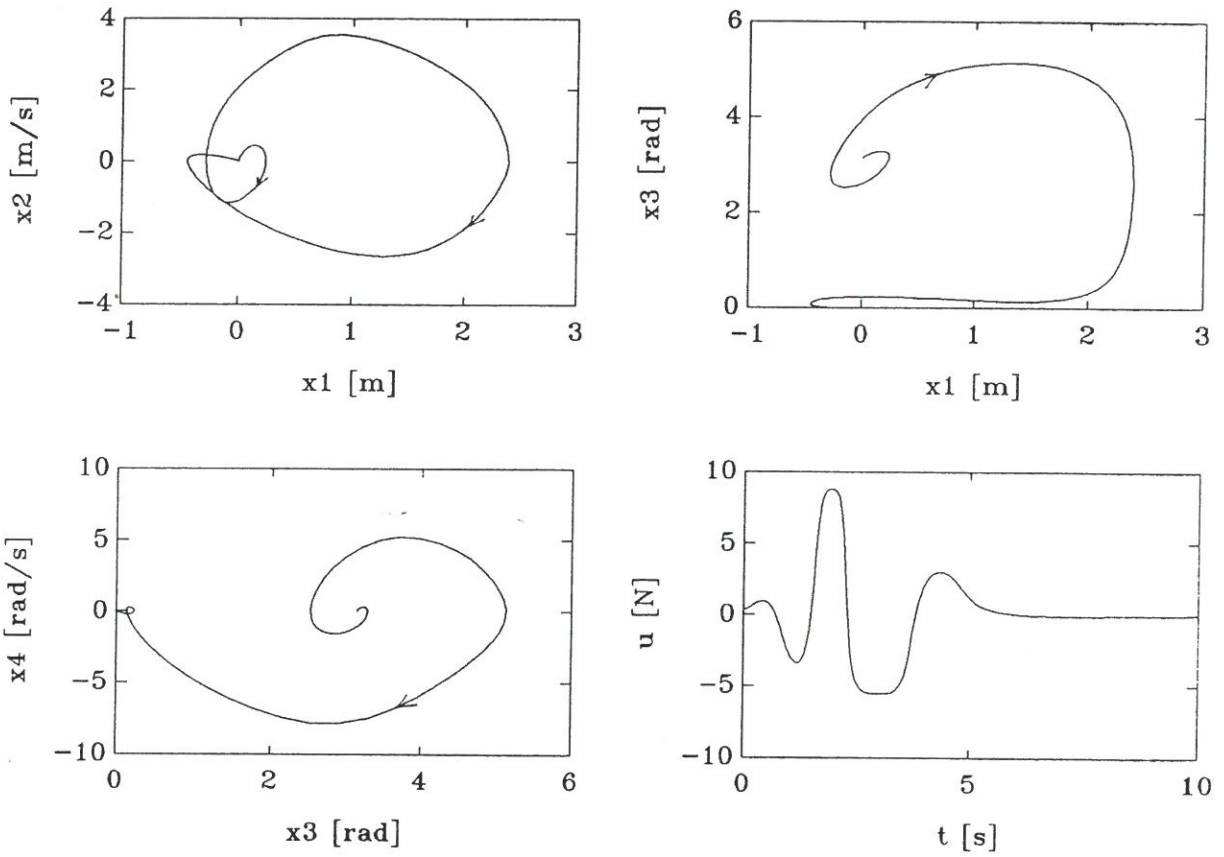

Figure 5: *Neural net with three neurons (one hidden layer with two neurons) that performs the swinging up problem (initial state* $x(0) = [0\,0\,\pi\,0]^t$ *; for parameters see Fig.2)*

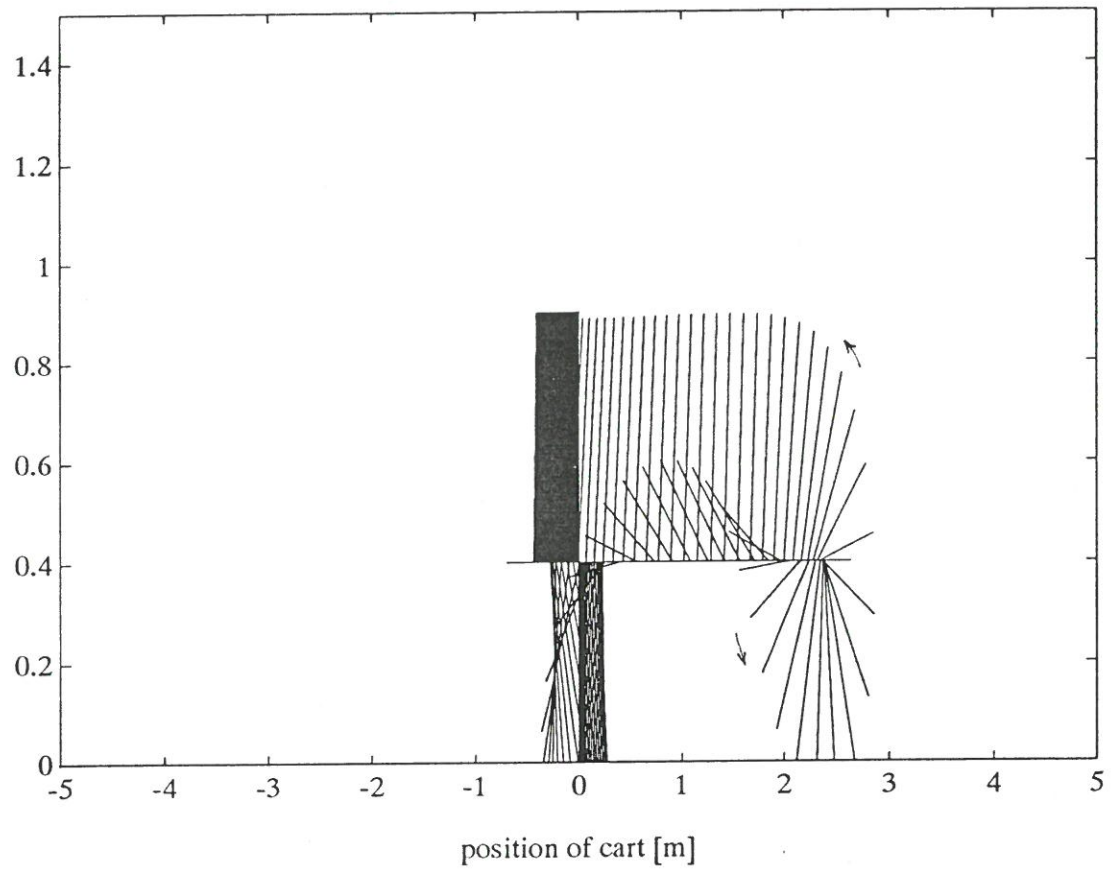

Figure 6: *Evolution of the pole during the 'swinging up' (initial state* $x(0) = [0\,0\,\pi\,0]^t$*)*

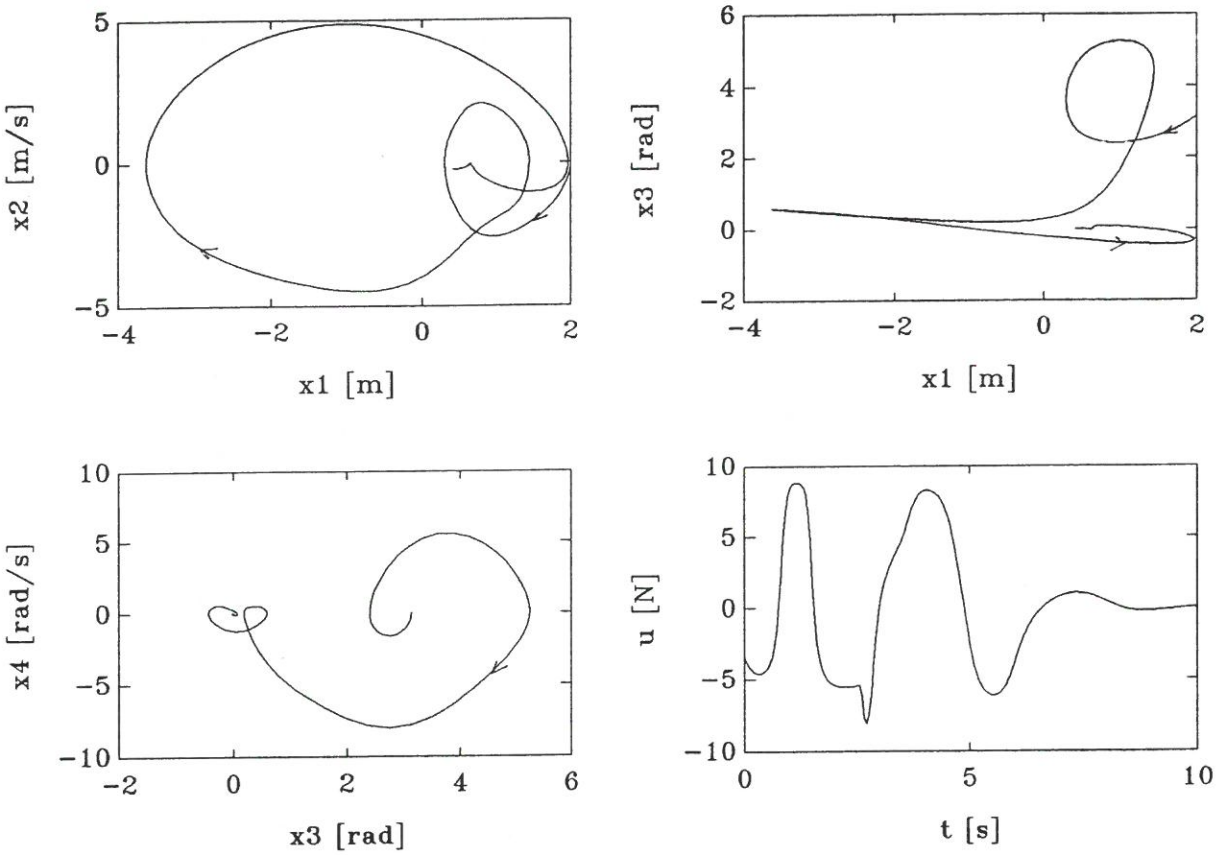

Figure 7: *Neural net with three neurons (one hidden layer with two neurons) that performs the swinging up problem (initial state* $x(0) = [2\,0\,\pi\,0]^t$ *; for parameters see Fig.2)*

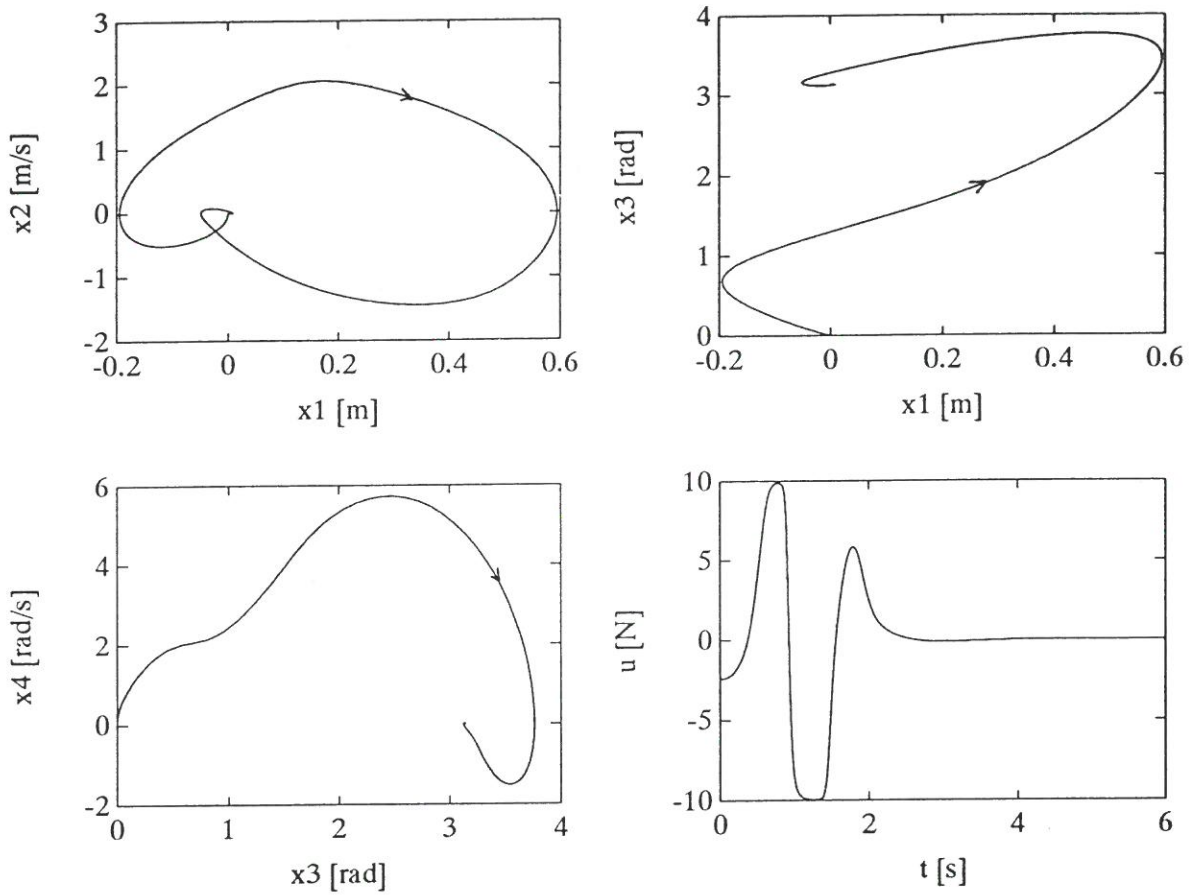

Figure 8: *Neural net with three neurons (one hidden layer with two neurons) that performs the swinging down problem (initial state* $x(0) = [0\,0\,0\,0]^t$ *; for parameters see Fig.2, but* $R = 0.01$*)*

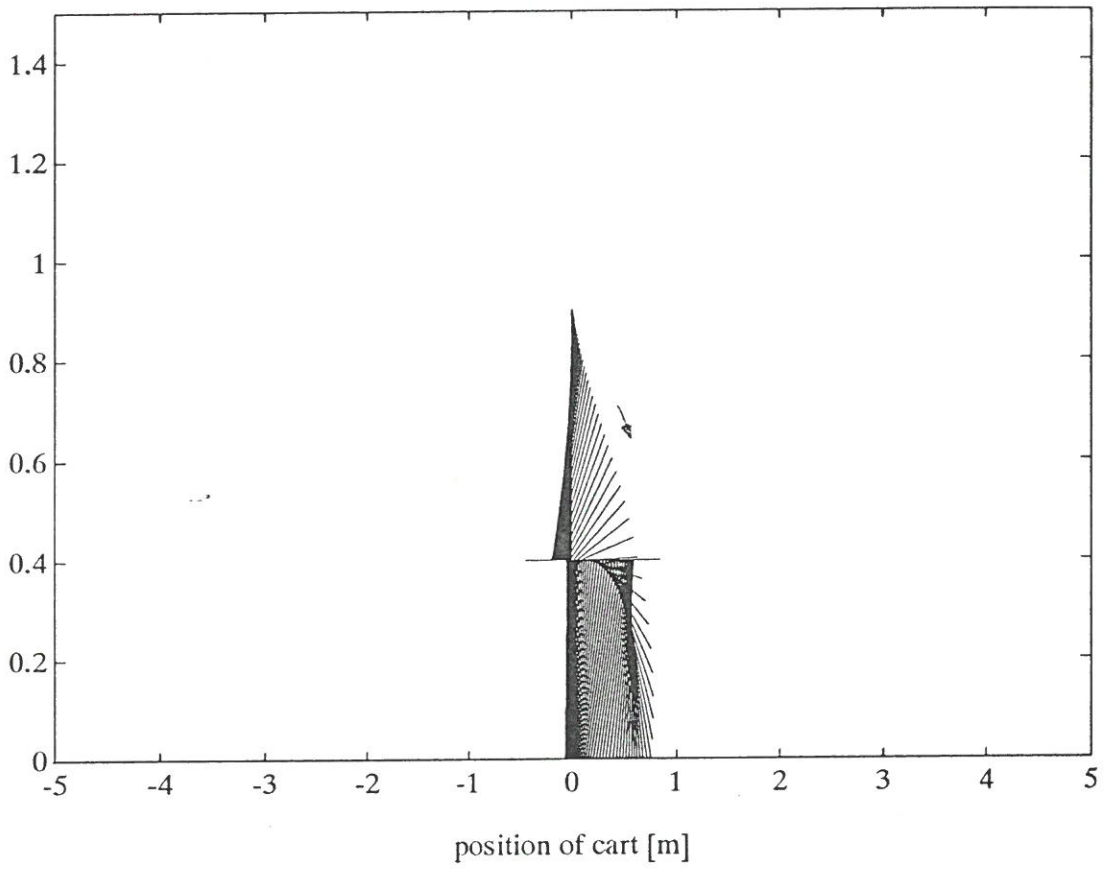

Figure 9: *Evolution of the pole during the 'swinging down' (initial state* $x(0) = [0\,0\,0\,0]^t$*)*

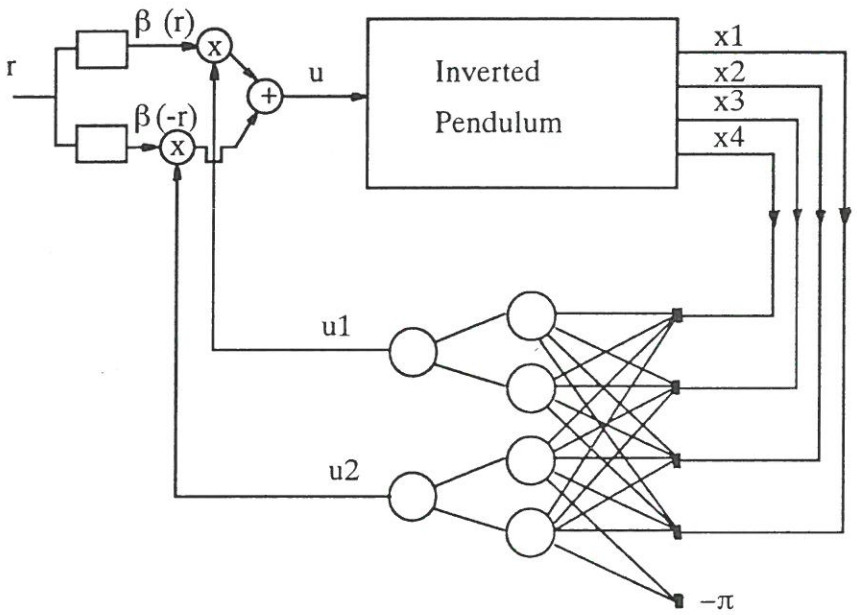

Figure 10: *Neural net with six neurons (one hidden layer with four neurons and two neurons in the output layer) that performs the swinging up and down control tasks. If* $r = 1$ *that part of the neural network responsible for the switching up is activated* ($u = u_1$). *For* $r = -1$ *the switching down controller is selected* ($u = u_2$).