Katholieke Universiteit Leuven

Departement Elektrotechniek ESAT-SISTA/TR 1993-03

Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points[1]

Johan A.K. Suykens, Bart L.R. De Moor, Joos P.L. Vandewalle[2]

Feb. 1993
Accepted for publication in *Neural Networks*

[1]This report is available by anonymous ftp from *gate.esat.kuleuven.ac.be* in the directory *pub/SISTA/suykens/reports/nnopt93.ps.Z*

[2]ESAT - Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, 3001 Leuven (Heverlee), Belgium, tel 32/16/22 09 31, fax 32/16/22 18 55, email: *Johan.Suykens@esat.kuleuven.ac.be*. 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.

Abstract

A design method for stabilization of nonlinear systems by feedforward and recurrent neural networks is proposed, applicable to transition between equilibrium points with local stabilization at the endpoint. Both static and dynamic state and output feedback stabilizing neural control are discussed. The link with linear controller design techniques is explained by linearizing the model around the target equilibrium point and incorporating the linear controller design results in the neural controller. The weights are learned off-line and are the solution to a nonlinear optimization problem through simulation of the system. The method is illustrated with the example of swinging up the pole of an inverted pendulum system with local stabilization at the upper equilibrium point, both by a feedforward and a recurrent neural network.

Key Words: Feedforward and recurrent neural networks, nonlinear optimization, static and dynamic output feedback, neural control.

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 can also switch the system from one operating point to another.
In this paper we will propose a general design strategy and then illustrate it on a particular 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 according to classical or modern linear control theory (e.g. by pole placement, PID, LQR, H_2, H_∞, etc. ...) (see e.g. Åström, Wittenmark (1984); Franklin *et al.* (1990); Maciejowski (1989) for an introduction to linear control theory).

- A general parametrized control law is proposed for the switching from one operating point to another, either by a feedforward or a recurrent neural network depending on the linear controller design. 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 equilibrium points, one can repeat the whole strategy for each pair of operating points and design an appropriate switching circuitry that governs the switchings.

An academic test example for this kind of problem is e.g. 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' (see e.g. Barto *et al.* (1983); Miller *et al.* (1990)). In this paper we study the more difficult control task of swinging the pole from 'down' to 'up' and to stabilize the pole locally at the upper equilibrium point. The proposed neural controller, parametrized by a feedforward or recurrent neural net, is able to perform this kind of transition. First linear controller design is done around the upper equilibrium point and then serves as constraint on the set of weights such that closed loop stability in the neighborhood of the 'target' equilibrium point is guaranteed locally. The switching between the equilibrium points is formulated as a nonlinear optimization problem with the linear controller design as constraint.

This paper is organized as follows:
Section 2 explains how the controller may be parametrized by feedforward and recurrent neural networks. In Section 3 a design method is presented where linear controller design results can be included in the neural controller. Section 4 treats the problem of transition between equilibrium points making use of the results of Section 3 and finally in Section 5 the methods are illustrated on the example of an inverted pendulum system.

2 Parametrization of controllers by feedforward and recurrent neural networks

In this section a general outline is proposed of a method for stabilization of nonlinear systems by means of feedforward and recurrent neural networks.
Given a single input nonlinear system

$$\left\{ \begin{array}{rcl} \dot{x} & = & f(x) + g(x)\, u \\ y & = & h(x) \end{array} \right. \tag{1}$$

with state vector $x \in \mathbb{R}^n$, input $u \in \mathbb{R}$, output vector $y \in \mathbb{R}^m$ and f, g, h vector fields defined on $\mathbb{R}^n$ ($f : \mathbb{R}^n \mapsto \mathbb{R}^n$, $g : \mathbb{R}^n \mapsto \mathbb{R}^n$, $h : \mathbb{R}^n \mapsto \mathbb{R}^m$). Furthermore we suppose that $f(0) = 0$, $h(0) = 0$. If this is not the case a change of coordinates can be applied. All techniques that will be discussed carry over to the multi-input and discrete time case too, but this will not be treated in the text.
Suppose the control task defined on (1) is to bring the state x from $x(0)$ to a target equilibrium point x_{eq}.
Therefore we consider both the case of parametrized static and dynamic output feedback. For *static* output feedback we have

$$u = \mu(y, \theta) \tag{2}$$

where $\theta \in \mathbb{R}^p$ is a parameter vector to be determined and $\mu : \mathbb{R}^m \mapsto \mathbb{R}$ is a continuous nonlinear mapping. This feedback law may be parametrized by a feedforward neural network with one hidden layer (see Fig.1) as

$$u = \alpha \tanh(w^t \tanh(V y)) \tag{3}$$

with interconnection weights $w \in \mathbb{R}^{n_h}$, $V \in \mathbb{R}^{n_h \times m}$ where n_h is the number of hidden neurons and the scalar $\alpha > 0$ is to be determined[1]. The parameter vector θ in (2) contains the elements of w,V and α. The motivation for such a parametrization stems from the fact that any continuous nonlinear function may be approximated by a multilayer neural network with one or more hidden layers (Cybenko (1989); Funahashi (1989); Hornik *et al.* (1989)).
In the *dynamic* output feedback case we have

$$\left\{ \begin{array}{rcl} \dot{z} & = & \nu(z, y, \theta_1) \\ u & = & \rho(z, y, \theta_2) \end{array} \right. \tag{4}$$

with $z \in \mathbb{R}^{n_z}$ the state of the controller, $\theta_1 \in \mathbb{R}^{p_1}$ and $\theta_2 \in \mathbb{R}^{p_2}$ parameter vectors to be determined and $\nu : \mathbb{R}^{n_z} \times \mathbb{R}^m \mapsto \mathbb{R}^{n_z}$, $\rho : \mathbb{R}^{n_z} \times \mathbb{R}^m \mapsto \mathbb{R}$ continuous nonlinear mappings.
This dynamic controller can be parametrized by a recurrent neural network (see Fig.1) as

$$\left\{ \begin{array}{rcl} \dot{z} & = & W_1 \tanh([V_{11}\ V_{12}].\begin{bmatrix} z \\ y \end{bmatrix}) \\ u & = & \alpha \tanh(w_2^t \tanh([V_{21}\ V_{22}].\begin{bmatrix} z \\ y \end{bmatrix})) \end{array} \right. \tag{5}$$

[1] As activation function we take the hyperbolic tangent function ($\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$), which is applied elementwise to a vector or a matrix.

with interconnection matrices $W_1 \in \mathbb{R}^{n_z \times n_{h_1}}$, $V_{11} \in \mathbb{R}^{n_{h_1} \times n_z}$, $V_{12} \in \mathbb{R}^{n_{h_1} \times m}$, $V_{21} \in \mathbb{R}^{n_{h_2} \times n_z}$, $V_{22} \in \mathbb{R}^{n_{h_2} \times m}$ and vector $w_2 \in \mathbb{R}^{n_{h_2}}$. The parameter vectors θ_1 and θ_2 in (4) contain respectively the elements of W_1, V_{11}, V_{12} and α, w_2, V_{21}, V_{22}.
The design procedure for the feedforward and recurrent neural net works basically as follows:

1. Design a linear controller around the target equilibrium point x_{eq} based on the linearized model. This controller may be static or dynamic depending on the particular system to be controlled. The nonlinear control law (2)(5) becomes then the linearized controller around x_{eq}.

2. The parameters (interconnection weights) serve to satisfy other performance criteria such as specifications on the transition from $x(0)$ to x_{eq}. This results into optimization problems constrained by the linear controller design results of step 1.

This procedure can be repeated between each pair of equilibrium points of the system or for other initial states.

3 Designing the neural controller

Incorporation of linear controller design results around the target equilibrium point will be outlined here for the case of static and dynamic output feedback.

3.1 Static output feedback using feedforward neural networks

3.1.1 General case

Suppose the plant model is given by (1) and the origin is the target equilibrium point ($f(0) = 0$, $h(0) = 0$). If this is not the case a change of coordinates can be applied in order to accomplish this. The static controller, parametrized by a feedforward neural networks with one hidden layer, is given by (3)

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

where $\alpha = |u|_{\max}$, which is a user defined maximum amplitude for the control signal. Another choice may be an output layer consisting of 'linear' neurons

$$u = w^t \tanh(Vy) \tag{7}$$

but (6) has the advantage of automatically restricting $|u|$ by α. Hence (6) is used in the sequel.
In closed loop we obtain

$$\dot{x} = f(x) + g(x)\, \alpha \tanh(w^t \tanh(Vh(x))) \tag{8}$$

In order to derive conditions for local stability around $x = 0$ the Jacobian matrix J is calculated and evaluated at $x = 0$:

$$\begin{array}{rcl} J(x) & = & f_x + g_x \alpha \tanh(w^t \tanh(Vh(x))) + g(x)\, \alpha \gamma_1 w^t \Gamma_2 V h_x \\ J(0) & = & f_0 + g(0)\, \alpha w^t V h_0 \end{array} \tag{9}$$

where f_x, g_x and h_x denote the matrices with partial derivatives $[\frac{\partial f_i}{\partial x_j}]$, $[\frac{\partial g_i}{\partial x_j}]$ $(i, j = 1, ..., n)$ and $[\frac{\partial h_i}{\partial x_j}]$ $(i = 1, ..., m, j = 1, ..., n)$. The matrices f_0 and h_0 represent the matrices f_x and h_x evaluated at 0. The scalar γ_1 and the diagonal matrix $\Gamma_2 = diag([\gamma_{2_1}, ..., \gamma_{2_{n_h}}])$ are given by

$$\begin{array}{rcll} \gamma_1 & = & 1 - \tanh^2(w^t \tanh(Vh(x))) & \\ \gamma_{2_i} & = & 1 - \tanh^2(v_i^t h(x)) & i = 1, ..., n_h \end{array} \tag{10}$$

where v_i^t denotes the i-th row of V. Local stability around $x = 0$ is guaranteed if the eigenvalues λ_i of $J(0)$ satisfy the condition

$$\Re\{\lambda_i(f_0 + g(0)\, \alpha w^t V\, h_0)\} < 0 \qquad i = 1, ..., n \tag{11}$$

It will be explained now how the condition (11) can be related to linear controller design results and how such results can be incorporated into the neural controller. Linear static output feedback $u = -k^t y$ applied to the linearized system around $x = 0$

$$\left\{ \begin{array}{rcl} \dot{x} & = & f_0\, x + g(0)\, u \\ y & = & h_0\, x \end{array} \right. \tag{12}$$

leads to the closed loop system

$$\dot{x} = (f_0 - g(0)\, k^t\, h_0)\, x \tag{13}$$

which is stable if

$$\Re\{\lambda_i(f_0 - g(0)\, k^t\, h_0)\} < 0 \qquad i = 1, ..., n \tag{14}$$

Identifying (11) with (14) gives the following set of constraints on the set of weights related to a certain linear controller design with static feedback gain k

$$k^t = -\alpha\, w^t V \tag{15}$$

Observe that the interconnection weights w and V are not fully determined by (15) which means that additional requirements could be achieved by the neural controller. This degree of freedom in the choice of the weights will be used further to enforce a transition between equilibrium points with local stabilization at the endpoint.

3.1.2 Full state feedback: example of LQR

In the case of full state feedback we have $h_0 = I_n$ if $y = x$ and the set of constraints on w, V can then be related e.g. to an LQR design (Franklin *et al.* (1990)). The linear state feedback law $u = -k_{lqr}^t x$ applied to (12) can be determined such that the cost function

$$C_{lqr} = \int_0^\infty (x^t Q x + u^t R u)\, dt \tag{16}$$

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

$$k_{lqr}^t = R^{-1} g(0)^t P \tag{17}$$

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

$$O = P f_0 + f_0^t P - P g(0) R^{-1} g(0)^t P + Q \tag{18}$$

and $k_{lqr}^t = -\alpha\, w^t V$.
Other techniques like H_∞ control (state feedback case) may be applied here.

3.1.3 Introduction of more hidden layers

It is a straightforward calculation to see that the results of one hidden layer (15) extend to feedforward neural networks with more than one hidden layer. We can summarize the conditions for local stability at the target equilibrium point as

1. *One neuron:*
$$k^t = -\alpha \, w^t \tag{19}$$

2. *One hidden layer:*
$$k^t = -\alpha \, w^t V \tag{20}$$

3. *General case of q hidden layers:*
$$k^t = -\alpha \, w^t V_1 V_2 ... V_q \tag{21}$$

Remark that in the case of one neuron all weights are determined by (19) for a given α and in the other cases (20)(21) a set of constraints is obtained from the local stability condition. This means that linear static state feedback can be placed at the same level as static state feedback with one neuron.

3.2 Dynamic output feedback using recurrent neural networks

We suppose again that the plant model is given by (1) with the same assumptions on f, g, h as in the static output feedback case. The dynamic controller is parametrized now by a recurrent neural network with one hidden layer (5)

$$\left\{ \begin{array}{lcl} \dot{z} & = & W_1 \tanh([V_{11} \; V_{12}] \begin{bmatrix} z \\ y \end{bmatrix}) \\ u & = & \alpha \tanh(w_2^t \tanh([V_{21} \; V_{22}] \begin{bmatrix} z \\ y \end{bmatrix})) \end{array} \right. \tag{22}$$

The closed loop system

$$\left\{ \begin{array}{lcl} \dot{z} & = & \phi(z, x) \\ \dot{x} & = & \psi(z, x) \end{array} \right. \tag{23}$$

yields

$$\left\{ \begin{array}{lcl} \dot{z} & = & W_1 \tanh([V_{11} \; V_{12}] \begin{bmatrix} z \\ h(x) \end{bmatrix}) \\ \dot{x} & = & f(x) + g(x) \, \alpha \tanh(w_2^t \tanh([V_{21} \; V_{22}] \begin{bmatrix} z \\ h(x) \end{bmatrix})) \end{array} \right. \tag{24}$$

Again $|u|_{\max} = \alpha$ because of (22). The closed loop system (24) will now be linearized around its target equilibrium $x = 0$, $z = 0$. The Jacobian matrix J is equal to

$$J(z, x) = \begin{bmatrix} \frac{\partial \phi}{\partial z} & \frac{\partial \phi}{\partial x} \\ \frac{\partial \psi}{\partial z} & \frac{\partial \psi}{\partial x} \end{bmatrix} = \begin{bmatrix} W_1 \Gamma_1 V_{11} & W_1 \Gamma_1 V_{12} h_x \\ g(x) \alpha \gamma_2 w_2^t \Gamma_3 V_{21} & f_x + g_x \alpha \tanh(.) + g(x) \alpha \gamma_2 w_2^t \Gamma_3 V_{22} h_x \end{bmatrix} \tag{25}$$

with diagonal matrices $\Gamma_1 = diag([\gamma_{1_1}, ..., \gamma_{1_{n_{h1}}}])$, $\Gamma_3 = diag([\gamma_{3_1}, ..., \gamma_{3_{n_{h2}}}])$ and γ_2 equal to

$$\begin{array}{rcll} \gamma_{1_i} & = & 1 - \tanh^2(v_{11_i}^t z + v_{12_i}^t h(x)) & i = 1, ..., n_{h_1} \\ \gamma_2 & = & 1 - \tanh^2(w_2^t \tanh(V_{21} z + V_{22} h(x)) & \\ \gamma_{3_i} & = & 1 - \tanh^2(v_{21_i}^t z + v_{22_i}^t h(x)) & i = 1, ..., n_{h_2} \end{array} \tag{26}$$

with $v_{11_i}^t$,$v_{12_i}^t$,$v_{21_i}^t$, $v_{22_i}^t$ respectively the i-th row of V_{11},V_{12}, V_{21},V_{22}. Local stability around $x = 0$, $z = 0$ is guaranteed if the eigenvalues λ_i of $J(0, 0)$ satisfy

$$\Re\{\lambda_i(J(0, 0))\} < 0 \qquad i = 1, ..., n + n_z \tag{27}$$

with

$$J(0, 0) = \left[\begin{array}{cc} W_1 V_{11} & W_1 V_{12}\, h_0 \\ g(0) \alpha\, w_2^t V_{21} & f_0 + g(0) \alpha\, w_2^t V_{22}\, h_0 \end{array} \right] \tag{28}$$

Like in the static output feedback case it will be shown how linear controller design results can be included in the neural controller. Linear dynamic output feedback

$$\left\{ \begin{array}{rcl} \dot{z} & = & E\, z + F\, y \\ u & = & G\, z + H\, y \end{array} \right. \tag{29}$$

applied on the linearized system around $x = 0$

$$\left\{ \begin{array}{rcl} \dot{x} & = & A\, x + B\, u \\ y & = & C\, x \end{array} \right. \tag{30}$$

with $A = f_0$, $B = g(0)$, $C = h_0$ leads to the closed loop system

$$\left\{ \begin{array}{rcl} \dot{z} & = & E\, z + FC\, x \\ \dot{x} & = & BG\, z + (A + BHC)\, x \end{array} \right. \tag{31}$$

which is stable if

$$\Re\{\lambda_i(\left[\begin{array}{cc} E & FC \\ BG & A + BHC \end{array} \right])\} < 0 \qquad i = 1, ..., n + n_z \tag{32}$$

A set of constraints on the interconnection weights is obtained by selecting for the linearized dynamic output feedback controller. Indeed by identifying (27)(28) with (32) one obtains

$$\left[\begin{array}{cc} E & F \\ G & H \end{array} \right] = \left[\begin{array}{cc} W_1 V_{11} & W_1 V_{12} \\ \alpha\, w_2^t V_{21} & \alpha\, w_2^t V_{22} \end{array} \right] \tag{33}$$

Again the weights are not fully determined by (33) and this additional degree of freedom can be used to enforce a transition between equilibrium points with local stabilization at the endpoint as will be illustrated for an inverted pendulum system in section 5. For the linear controller design on the linearized system all existing linear controller design methods can be applied, including e.g. *PID*, *LQG*, H_2, H_∞, μ-synthesis etc (Åström, Wittenmark (1984); Franklin *et al.* (1990); Maciejowski (1989)). The ideas can also be extended to parametrizations with more than one hidden layer like in the static output feedback case.

4 Transition between equilibrium points

Transition between equilibrium points will be discussed here for the static and dynamic output feedback case.

4.1 Static feedback

In order to impose a transition from a given initial state $x(0)$ to a target equilibrium point x_{eq} (we suppose $x_{eq} = 0$ here without loss of generality) to the system (1) the following optimization problem may be formulated

$$\min_{\alpha,w,V} \; C(\alpha, w, V, f, g, h, x(0), \eta, \zeta, T) \tag{34}$$

where the cost function C is defined as

$$C = \eta(x(T)) + \int_0^T \zeta(x(t))\, dt$$

with constraints

1. Local stability at $x = 0$ (15):
$$k^t = -\alpha\, w^t V \tag{35}$$

2. Closed loop system dynamics (8):
$$\dot{x} = f(x) + g(x)\, \alpha\, \tanh(w^t\, \tanh(V h(x))) \tag{36}$$

where e.g. $\eta(x(T)) = \|x(T)\|$ and $\zeta(x(t)) = x(t)^t x(t)$ (quadratic control) or $\zeta = 0$ (terminal control) and T is the time horizon (Bryson, Ho (1969)). In the case of m hidden neurons (V square and full rank) constraint (35) can be eliminated because $w^t = -\frac{k^t}{\alpha} V^{-1}$ so that we get the optimization problem

$$\min_{\alpha,V} \; C(\alpha, V, f, g, h, x(0), \eta, \zeta, T) \tag{37}$$

with constraint

$$\dot{x} = f(x) + g(x)\, \alpha\, \tanh(-\frac{k^t}{\alpha}\, V^{-1}\, \tanh(V h(x))) \tag{38}$$

Some remarks are made here with repect to the optimization problem (34)-(38)

Remarks:

1. The region around $x = 0$ where the linearization (9) is valid depends on w, V. This indicates that it will be needed to formulate an additional constraint on the norm of w and V to (38).

2. It is not guaranteed a priori that the linearized region will be entered. This question of controllability may depend e.g. on the choice of the horizon T.

3. The determination of the number of hidden neurons needed to perform a certain transition for a given system is in fact a trial and error procedure, but it will be illustrated with the example of an inverted pendulum that small size neural nets are capable of performing complex control tasks.

4. Robustness considerations are twofold: robustness with respect to the initial state and robustness inside the linearized region. For the linearized region it can be expected that the robustness is comparable to the corresponding linear controller on which the neural controller was designed. Omitting the constraint (35) would result in a controller that is less robust to perturbations or even locally unstable around the target equilibrium point. With respect to the initial state nothing can be said in general and this problem has to be investigated for the particular system under study.

Numerical solution to the optimization problem:

A practical solution to (37)(38) is to simulate the closed loop system (38) over a finite time T, by applying an appropriate integration rule and to calculate the cost function from this simulation result. In order to guarantee a sufficiently 'large' region for which the linearization (9) is valid we add the following constraints to (38)

$$\begin{array}{lcl} \|V\| & < & \beta_1 \\ \|w^t V\| & < & \beta_2 \end{array} \qquad (39)$$

where $\|.\|$ represents any induced norm and β_1, β_2 are user defined scalars. The last constraint $\|w^t V\|$ can be interpreted as a design rule for α because $-\|\frac{k^t}{\alpha}\| = \|w^t V\| < \beta_2$ so that $\alpha > \frac{\|k\|}{\beta_2}$. This leads to the constrained optimization problem

$$\min_V \; C(V, f, g, h, x(0), \eta, \zeta, T) \qquad \|V\| < \beta_1 \qquad (40)$$

where the costfunction C is calculated from the simulation result of the closed loop system dynamics (38).
In general many local optima may exist to this problem. For solving (40) one can use global optimization techniques (e.g. Monte Carlo method, Genetic Algorithms, ...) or local optimization algorithms (e.g. Sequential Quadratic Programming) by trying some different starting points. If a local optimization routine is used, the constraint in (40) is also needed to keep $\|V\|$ bounded, because it was observed that otherwise $\|V\| \to \infty$ for several starting points.

Remark: In the case where the matrix $V_{n_h \times m}$ is not square ($n_h \neq m$) the following elimination trick can be done to eliminate the constraint (35) out of (34):

- $n_h < m$: Define s_1, s_2, V_1, V_2 according to

  $$-\frac{1}{\alpha} k^t = [s_1^t \; s_2^t] = w^t [V_1 \; V_2] \qquad (41)$$

  with V_1 square ($n_h \times n_h$). This leads to a condition

  $$s_1^t T = s_2^t \qquad (42)$$

  with $T = V_1^{-1} V_2$ where one part of T may be chosen freely and the other one is calculated afterwards from (42).

- $n_h > m$: Define V_1, V_2 such that

  $$-\frac{1}{\alpha} k^t = w_1^t V_1 + w_2^t V_2 \qquad (43)$$

  with V_1 square ($m \times m$). The elements of w_2, V_1, V_2 may be chosen freely and w_1 can be calculated from (43).

In both cases the parameter vector to be optimized consists of all elements that my be chosen freely in (42)(43).

4.2 Dynamic output feedback

In the dynamic output feedback case, the following optimization problem can be formulated for a transition from a given initial state $x(0)$ to $x = 0,\ z = 0$ for the system (1)

$$\min_{\alpha, W_1, w_2, V_{11}, V_{12}, V_{21}, V_{22}} \quad C(\alpha, W_1, w_2, V_{11}, V_{12}, V_{21}, V_{22}, f, g, h, x(0), z(0), \eta, \zeta, T) \tag{44}$$

where the cost function C is defined as

$$C = \eta(x(T), z(T)) + \int_0^T \zeta(x(t), z(t))\, dt$$

with constraints

1. Local stability at $x = 0,\ z = 0$ (33):

$$\begin{array}{rcl} [E\ F] & = & W_1 [V_{11}\ V_{12}] \\ [G\ H] & = & \alpha\, w_2^t [V_{21}\ V_{22}] \end{array} \tag{45}$$

2. Closed loop system dynamics (24):

$$\left\{ \begin{array}{rcl} \dot{z} & = & W_1 \tanh([V_{11}\ V_{12}] \begin{bmatrix} z \\ h(x) \end{bmatrix}) \\ \dot{x} & = & f(x) + g(x)\, \alpha \tanh(w_2^t \tanh([V_{21}\ V_{22}] \begin{bmatrix} z \\ h(x) \end{bmatrix})) \end{array} \right. \tag{46}$$

If the matrices $[V_{11}\ V_{12}]$ and $[V_{21}\ V_{22}]$ are square and full rank, constraint (45) can be eliminated such that

$$\min_{\alpha, V_{11}, V_{12}, V_{21}, V_{22}} \quad C(\alpha, V_{11}, V_{12}, V_{21}, V_{22}, f, g, h, x(0), z(0), \eta, \zeta, T) \tag{47}$$

with constraint

$$\left\{ \begin{array}{rcl} \dot{z} & = & [E\ F]\, [V_{11}\ V_{12}]^{-1} \tanh([V_{11}\ V_{12}] \begin{bmatrix} z \\ h(x) \end{bmatrix}) \\ \dot{x} & = & f(x) + g(x)\, \alpha \tanh([G\ H]\, [V_{21}\ V_{22}]^{-1} \tanh([V_{21}\ V_{22}] \begin{bmatrix} z \\ h(x) \end{bmatrix})) \end{array} \right. \tag{48}$$

The same remarks can be made here as in the static output feedback case with respect to controllability, reachability of the linearized region and robustness.

Numerical solution to the optimization problem:

Again the system (46) is simulated over a finite time T, by applying an integration rule and next the cost function is calculated from this simulation result. A sufficiently 'large' region for which the linearization (25) is valid is guaranteed by

$$\begin{array}{lcl} \|[V_{11}\ V_{12}]\| & < & \beta_1 \\ \|[V_{21}\ V_{22}]\| & < & \beta_2 \\ \|w_2^t\,[V_{21}\ V_{22}]\| & < & \beta_3 \end{array} \tag{49}$$

The last constraint serves as a design rule for α because $[G\ H] = \alpha w_2^t\,[V_{21}\ V_{22}]$ such that $\alpha > \frac{\|[G\,H]\|}{\beta_3}$. This leads to the constrained optimization problem

$$\min_{V_{11},V_{12},V_{21},V_{22}} \quad C(V_{11}, V_{12}, V_{21}, V_{22}, f, g, h, x(0), z(0), \eta, \zeta, T) \tag{50}$$

with constraints

$$\begin{array}{lcl} \|[V_{11}\ V_{12}]\| & < & \beta_1 \\ \|[V_{21}\ V_{22}]\| & < & \beta_2 \end{array} \tag{51}$$

5 Example: inverted pendulum

We will now discuss an example of an inverted pendulum system for transition between equilibrium points with local stabilization at the target equilibrium point. A state space model for the inverted pendulum system (Barto *et al.*(1983)) (Fig.2) is given by

$$\left\{ \begin{array}{lcl} \dot{x} & = & f(x) + g(x)\,u \\ y & = & h(x) \end{array} \right. \tag{52}$$

with state $x \in \mathbb{R}^4$ and input $u \in \mathbb{R}$ and output $y \in \mathbb{R}^2$

$$f(x) = \left[\begin{array}{c} x_2 \\ \frac{\frac{4}{3} m l x_4^2 sin x_3 - \frac{mg}{2} sin(2x_3)}{\frac{4}{3} m_t - m cos^2 x_3} \\ x_4 \\ \frac{m_t g sin x_3 - \frac{ml}{2} x_4^2 sin(2x_3)}{l(\frac{4}{3} m_t - m cos^2 x_3)} \end{array} \right], \; g(x) = \left[\begin{array}{c} 0 \\ \frac{4}{3}\,\frac{1}{\frac{4}{3} m_t - m cos^2 x_3} \\ 0 \\ -\frac{cos x_3}{l(\frac{4}{3} m_t - m cos^2 x_3)} \end{array} \right], \; h(x) = \left[\begin{array}{c} x_1 \\ x_3 \end{array} \right] \tag{53}$$

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 by $|u| < \alpha$. 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^-. The linearized system around the target equilibrium point eq^+ is

$$f_0 = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{\frac{4}{3}m_t - m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m_t g}{l(\frac{4}{3}m_t - m)} & 0 \end{array} \right], \; g(0) = \left[\begin{array}{c} 0 \\ \frac{4}{3}\,\frac{1}{\frac{4}{3}m_t - m} \\ 0 \\ -\frac{1}{l(\frac{4}{3}m_t - m)} \end{array} \right], \; h_0 = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \tag{54}$$

Two neural controllers will be studied now: a full static state feedback neural controller (feedforward net) related to LQR design and a dynamic ouput feedback neural controller (recurrent net) based on LQG.

5.1 Feedforward neural network

We take full static state feedback (see Fig.3) here with a control signal parametrized as

$$u = \alpha \tanh(w^t \tanh(Vx)) \tag{55}$$

where α is the maximal force applied to the cart and $V \in \mathbb{R}^{4\times 4}$ and $w \in \mathbb{R}^4$. The design method for the neural controller can be outlined as follows

- **Control task:** The control task is a transition from eq^- to eq^+ with local stability at eq^+.

- **LQR design on the linearized system:**
An LQR controller (16)-(18) is calculated for $Q = I_4$ and $R = 0.01$, $m = 0.1\,\text{kg}$, $m_t = 1.1\,\text{kg}$, $l = 0.5\,\text{m}$ which gives

$$k_{lqr}^t = [-10.0000 \;\; -16.8140 \;\; -100.4101 \;\; -27.7187]$$

- **Choice of α:**
In (39) $\beta_2 = 15$ was taken and $\alpha = 10\,\text{N}$ such that $\alpha > \frac{\|k\|_2}{\beta_2}$. This choice of β_2 turns out not to be very critical.

- **Optimization problem:**
The optimization problem (40) was solved here with $\beta_1 = 2$, $x(0) = [0\;0\;\pi\;0]^t$, $T = 3\,\text{sec}$. The closed loop system is simulated by means of a trapezoidal integration rule (see Rice (1983)) with constant step length 0.03. This simulation was written in C software. The cost function was calculated from this simulation result with terminal control ($\zeta = 0$). A local optimization method of Sequential Quadratic Programming (SQP) was used (*constr* of Matlab) (Matlab User's Guide (1992)). In this method a Quadratic Programming subproblem is solved at each iteration. An estimate of the Hessian of the Lagrangian is updated at each iteration using the BFGS formula (see Fletcher (1987); Gill *et al.* (1981); Powell (1983)) A local optimal solution is

$$V = \begin{bmatrix} 0.8375 & -0.0987 & 0.8139 & 0.2730 \\ 0.3902 & 0.7257 & 0.9669 & -1.1032 \\ -0.0051 & 0.4888 & -0.2993 & 0.7534 \\ 0.2590 & 0.0862 & 0.1389 & -0.8283 \end{bmatrix}$$

In Fig.4 this solution is given in state space. Fig.5 illustrates the evolution of the pole during swinging up.

- **Robustness:** In Fig.6 the neural optimal controller was tested for some other initial states in the neighborhood of eq^-. Close to eq^+ the controller acts as a classical LQR controller with comparable robustness. In Fig.7 the behaviour of the neural controller for initial states in the neighborhood of eq^+ is given.

5.2 Recurrent neural network

The control signal is parametrized by a recurrent neural network (see Fig.8) as

$$\left\{ \begin{array}{rcl} \dot{z} & = & W_1 \tanh([V_{11}\ V_{12}] \left[\begin{array}{c} z \\ y \end{array} \right]) \\ u & = & \alpha \tanh(w_2^t \tanh([V_{21}\ V_{22}] \left[\begin{array}{c} z \\ y \end{array} \right])) \end{array} \right. \tag{56}$$

with α the maximal force applied to the cart and $W_1 \in \mathbb{R}^{4\times 6}$, $w_2 \in \mathbb{R}^6$, $V_{11} \in \mathbb{R}^{6\times 4}$, $V_{12} \in \mathbb{R}^{6\times 2}$, $V_{21} \in \mathbb{R}^{6\times 4}$, $V_{22} \in \mathbb{R}^{6\times 2}$ ($n_{h_1} = n_{h_2} = 6$) and $y = [x_1\ x_3]^t$. The neural controller design works as follows

- **Control task:** The control task is a transition from eq^- to eq^+ with local stability at eq^+.
- **Linear controller design:**
  Several linear controller design techniques may be applied here to (30) like H_2, H_∞, μ-synthesis etc (Åström, Wittenmark (1984); Franklin *et al.* (1990); Maciejowski (1989)). We take here an LQG design. The goal is then to minimize the quadratic cost function

  $$C_{lqg} = E\{\int_0^\infty [x^t\ u^t] \left[\begin{array}{cc} Q & N_c \\ N_c^t & R \end{array} \right] \left[\begin{array}{c} x \\ u \end{array} \right] dt\} \tag{57}$$

  given the plant model ($E\{.\}$ denotes the expectation operator)

  $$\left\{ \begin{array}{rcl} \dot{x} & = & Ax + Bu + \xi \\ y & = & Cx + \psi \end{array} \right. \tag{58}$$

  where ξ and ψ represent the process and measurement noise that is white and Gaussian with joint correlation function

  $$E\{\left[\begin{array}{c} \xi(t) \\ \psi(\tau) \end{array} \right] [\xi(t)\ \psi(\tau)]^t\} = \left[\begin{array}{cc} \Xi & N_f \\ N_f^t & \Psi \end{array} \right] \delta(t-\tau) \tag{59}$$

  where the following choice was made

  $$\left[\begin{array}{cc} Q & N_c \\ N_c^t & R \end{array} \right] = 0.01\ I_6, \ \left[\begin{array}{cc} \Xi & N_f \\ N_f^t & \Psi \end{array} \right] = I_5$$

  This gives

  $$\left[\begin{array}{c|c} E & F \\ \hline G & H \end{array} \right] = \left[\begin{array}{rrrr|rr} -1.7402 & 1.0000 & 0.2403 & 0 & 1.7402 & -0.2403 \\ -0.0675 & 2.2465 & 31.7702 & 7.9782 & 1.0431 & -1.3714 \\ 0.2403 & 0 & -8.0070 & 1.0000 & -0.2403 & 8.0070 \\ -0.4927 & -3.3698 & -62.4684 & -11.9672 & -0.9707 & 31.5852 \\ \hline -1.0000 & -2.3027 & -31.8945 & -8.1776 & 0 & 0 \end{array} \right]$$

- **Choice of α:**
  In (49) $\beta_3 = 4$ was taken and $\alpha = 10\,\mathrm{N}$ such that $\alpha > \frac{\|[G\ H]\|}{\beta_3}$. The choice of β_3 is not critical here.

- **Optimization problem:**
The optimization problem (50)(51) was solved with $\beta_1 = \beta_2 = 2$, $x(0) = [0\ 0\ \pi\ 0]^t$, $z(0) = 0$, $T = 10\,sec$. The other parameters were the same as in the static output feedback case and a trapezoidal integration rule with constant step length 0.01. The cost function C in (50) was calculated from the simulation result with terminal control ($\zeta = 0$). A local optimal solution obtained in Matlab is

$$\left[\begin{array}{c|c} V_{11} & V_{12} \\ \hline V_{21} & V_{22} \end{array}\right] = \left[\begin{array}{rrrr|rr} -0.1496 & -0.1356 & -0.0200 & 0.1870 & -0.1646 & 0.1179 \\ 0.1444 & 0.1255 & 0.0085 & -0.0312 & 0.0386 & 0.0693 \\ -0.0889 & 0.0090 & 0.0032 & 0.0507 & 0.1512 & 0.0328 \\ 0.1358 & -0.0653 & -0.1455 & -0.0025 & 0.1289 & -0.0117 \\ 0.1234 & 0.0129 & 0.1871 & 0.1086 & 0.0607 & -0.0696 \\ 0.0590 & -0.0615 & 0.0540 & 0.1158 & -0.0187 & 0.0479 \\ \hline -0.0462 & -0.0326 & 0.0962 & 0.0905 & -0.1174 & -0.0314 \\ -0.1275 & 0.0797 & -0.0140 & -0.0248 & -0.0925 & 0.0349 \\ -0.0985 & -0.0147 & -0.1009 & 0.1019 & 0.0907 & 0.0229 \\ 0.0566 & 0.0750 & -0.2760 & -0.0069 & -0.1212 & 0.0158 \\ -0.0249 & 0.1195 & 0.0659 & 0.0398 & 0.0042 & -0.0958 \\ 0.1224 & 0.0249 & -0.1407 & -0.0584 & -0.0188 & -0.1166 \end{array}\right]$$

In Fig.9 this solution is presented in state space. Fig.10 shows the evolution of the pole during swinging up.

- **Robustness:** The neural optimal controller was tested for some other initial states in the neighborhood of eq^- (see Fig.11) and seems very robust with respect to the choice of $x(0)$ and certainly far more robust for this inverted pendulum system than the feedforward neural net controller based on LQR. Close to eq^+ the controller acts as an LQG controller with comparable robustness. In Fig.12 the behaviour of the neural controller is given for initial states in the neighborhood of eq^+.

6 Conclusion

It was shown how transitions between equilibrium points can be realized by static and dynamic neural controllers, parametrized by feedforward or recurrent neural networks, with local stability at the target equilibrium point. All classical and modern linear controller design techniques like PID, LQR, pole placement, H_2, H_∞ etc. may be applied around the target equilibrium point and can be incorporated into the neural controller. Hence non local control can be obtained by means of neural networks without losing the results of well developed linear control theory.

References

Åström K.J., B. Wittenmark (1984). *Computer-controlled systems, theory and design*, Englewood Cliffs NJ: Prentice-Hall.

Barto A.G., R.S. Sutton, C.W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No.5, pp. 834-846.

Bryson A.E., Y.C. Ho (1969). *Applied optimal control*, Waltham, MA: Blaisdel.

Cybenko G. (1989). Approximations by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems,* 2, pp.183-192.

Fletcher R. (1987). *Practical methods of optimization*, second edition, Chichester and New York: John Wiley and Sons.

Franklin G.F., J.D. Powell, M.L. Workman (1990). *Digital control of dynamic systems*, Reading MA: Addison-Wesley.

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

Gill P.E., W. Murray, M.H. Wright (1981). *Practical Optimization*, London: Academic Press.

Hornik K., M. Stinchcombe, H. White (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, Vol.2, pp.359-366.

Maciejowski J.M. (1989). *Multivariable feedback design*, Addison-Wesley.

Matlab User's Guide (1992). *Optimization Toolbox User's Guide*, Version 3.5f, The MathWorks, Inc.

Miller W.T., R.S. Sutton, P.J. Werbos (1990). *Neural networks for control*, Cambridge, MA: M.I.T. Press.

Powell M.J.D. (1983). Variable metric methods for constrained optimization, In: A. Bachem, M. Grötschel and B. Korte (Eds.), *Mathematical programming: the state of the art*, pp.288-311, Berlin and New York: Springer-Verlag.

Rice J.R. (1983). *Numerical methods, software and analysis*, Mc Graw-Hill.

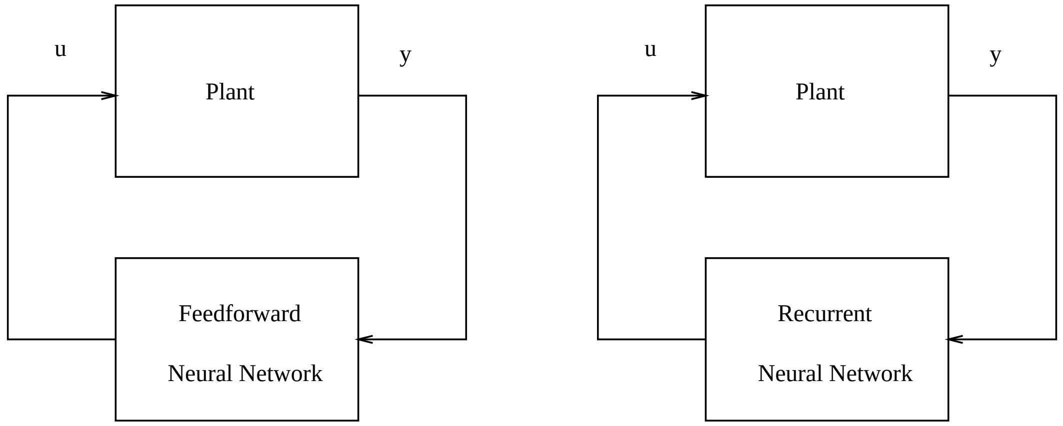

Figure 1: *Static and dynamic output feedback laws, parametrized respectively by feedforward or recurrent neural networks.*

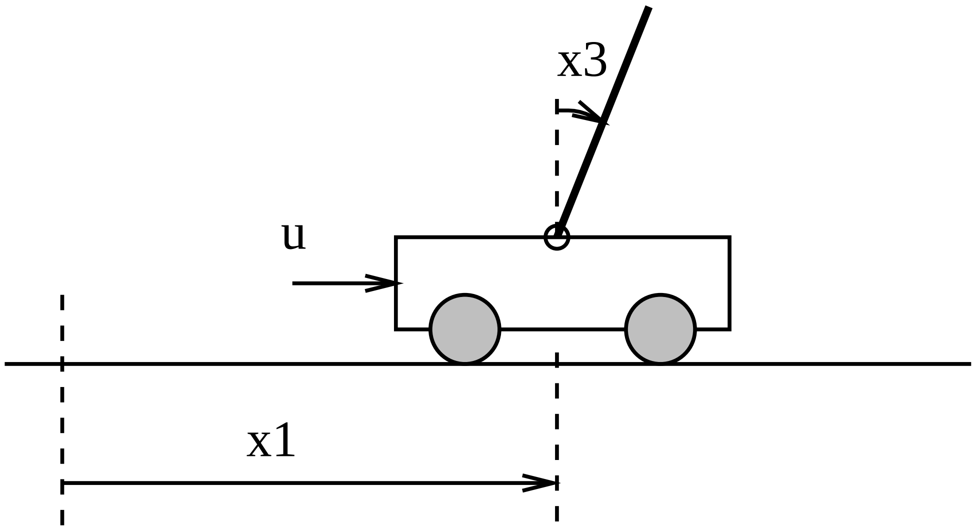

Figure 2: *Inverted pendulum system:* x_1 *is the position of the cart,* x_3 *is the angle of the pole,* u *is the force applied to the cart and* $x_2 = \dot{x}_1$*,* $x_4 = \dot{x}_3$*.*

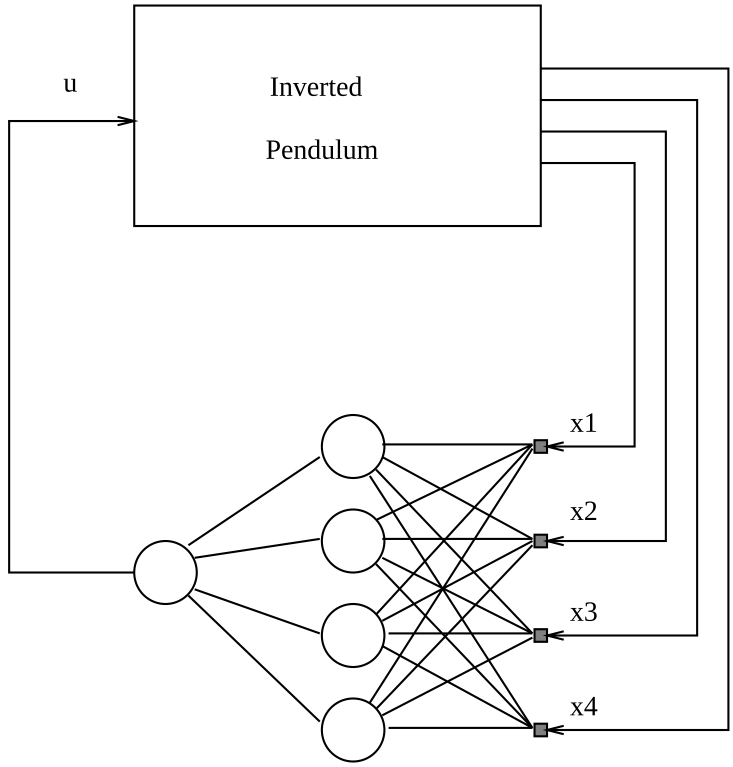

Figure 3: *Inverted pendulum controlled by a feedforward neural network with full state feedback for performing the swinging up problem.*

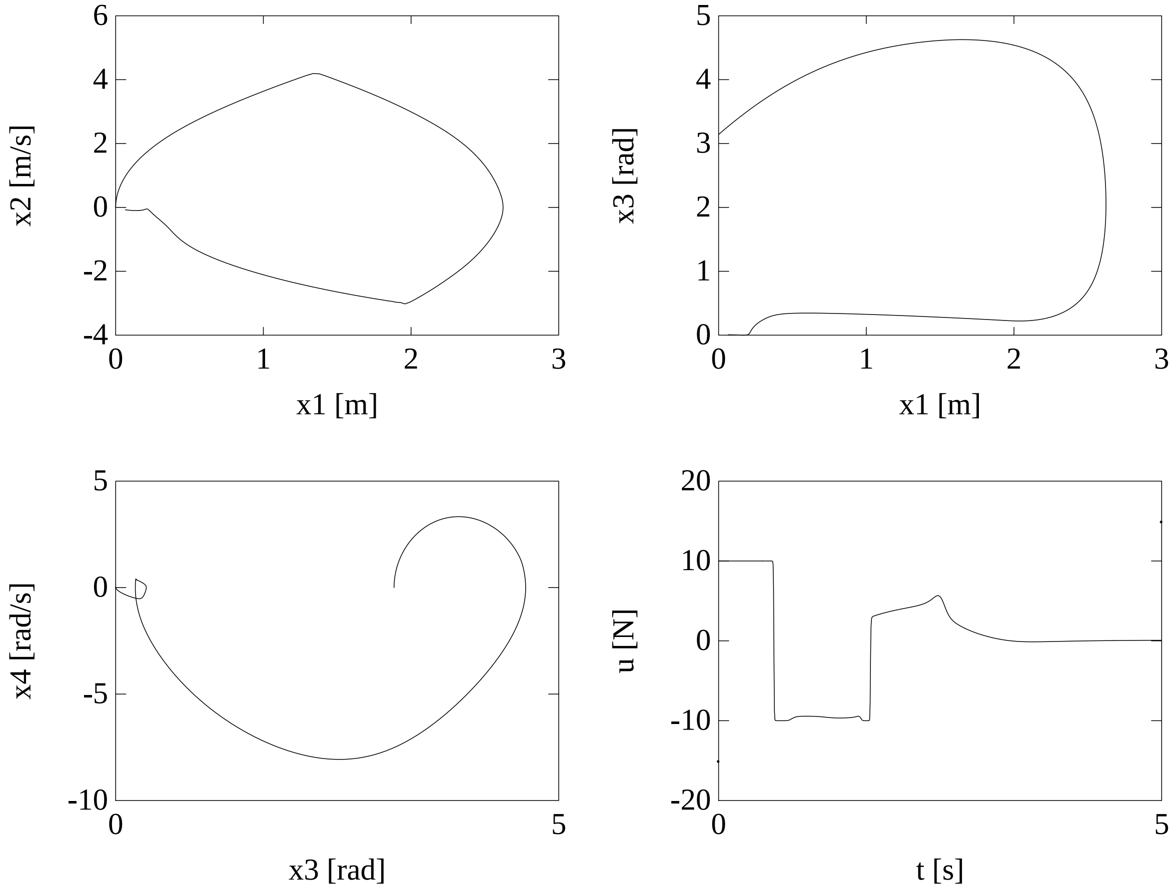

Figure 4: *Solution in state space to the swinging up problem of an inverted pendulum system by a feedforward neural network related to LQR design (initial state $x(0) = [0\ 0\ \pi\ 0]^t$ and target equilibrium point $x = 0$). This solution corresponds to an intuitive solution that human would come up with when ask to swing up and stabilize an inverted pendulum by manual control: first apply a maximal force to the right and to the left in order to swing up the pole and finally a smaller force to keep the pole at its upper equilibrium state.*

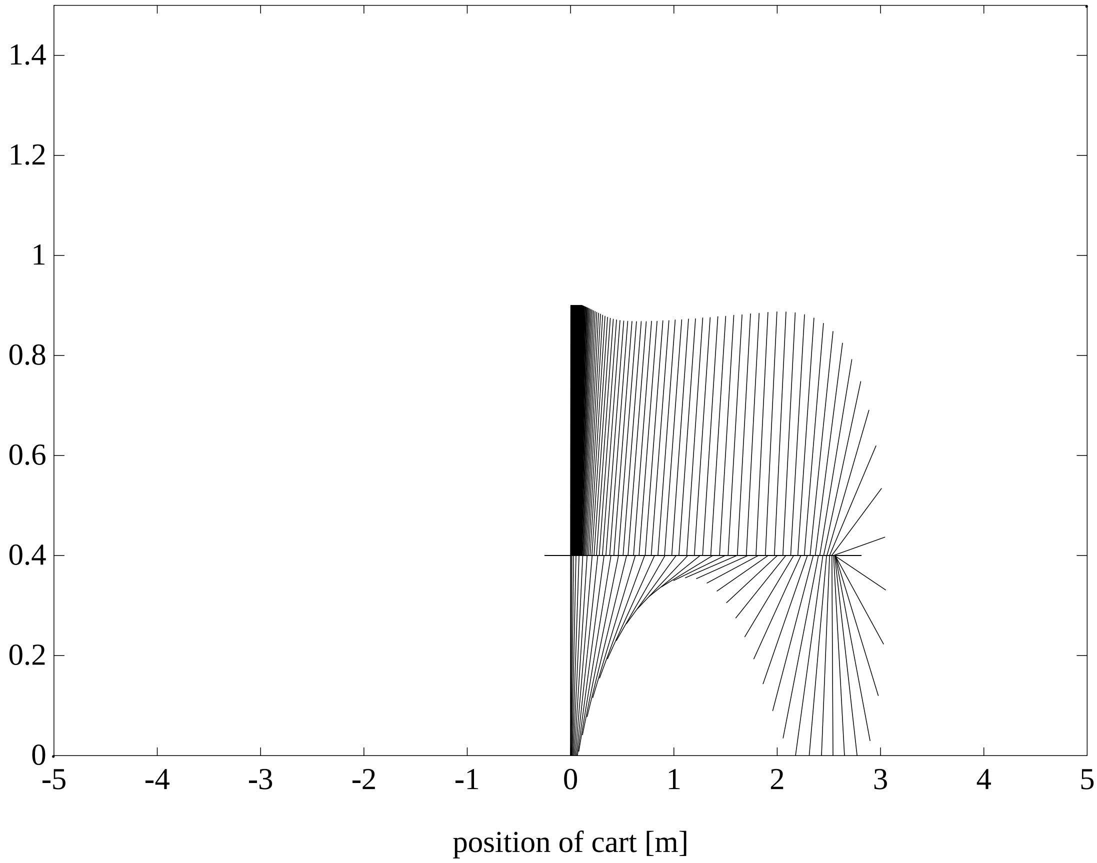

Figure 5: *Evolution of the pole during the transition from* $x(0) = [0\ 0\ \pi\ 0]^t$ *to the target equilibrium point* $x = 0$ *by feedforward neural net control of Fig.3-4.*

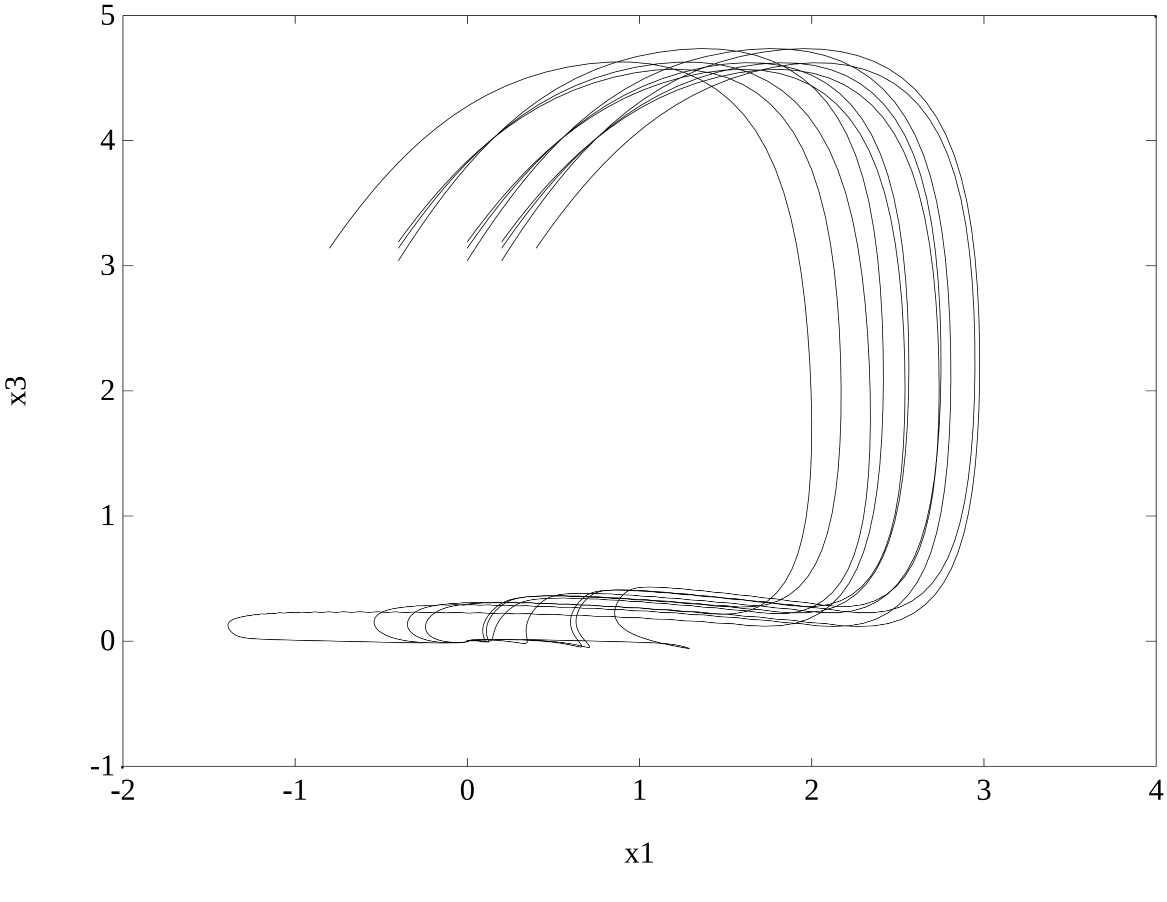

Figure 6: *Robustness of the feedforward neural controller of Fig.3-4 with respect to the initial state $x(0)$ lying in the neighborhood of eq^-.*

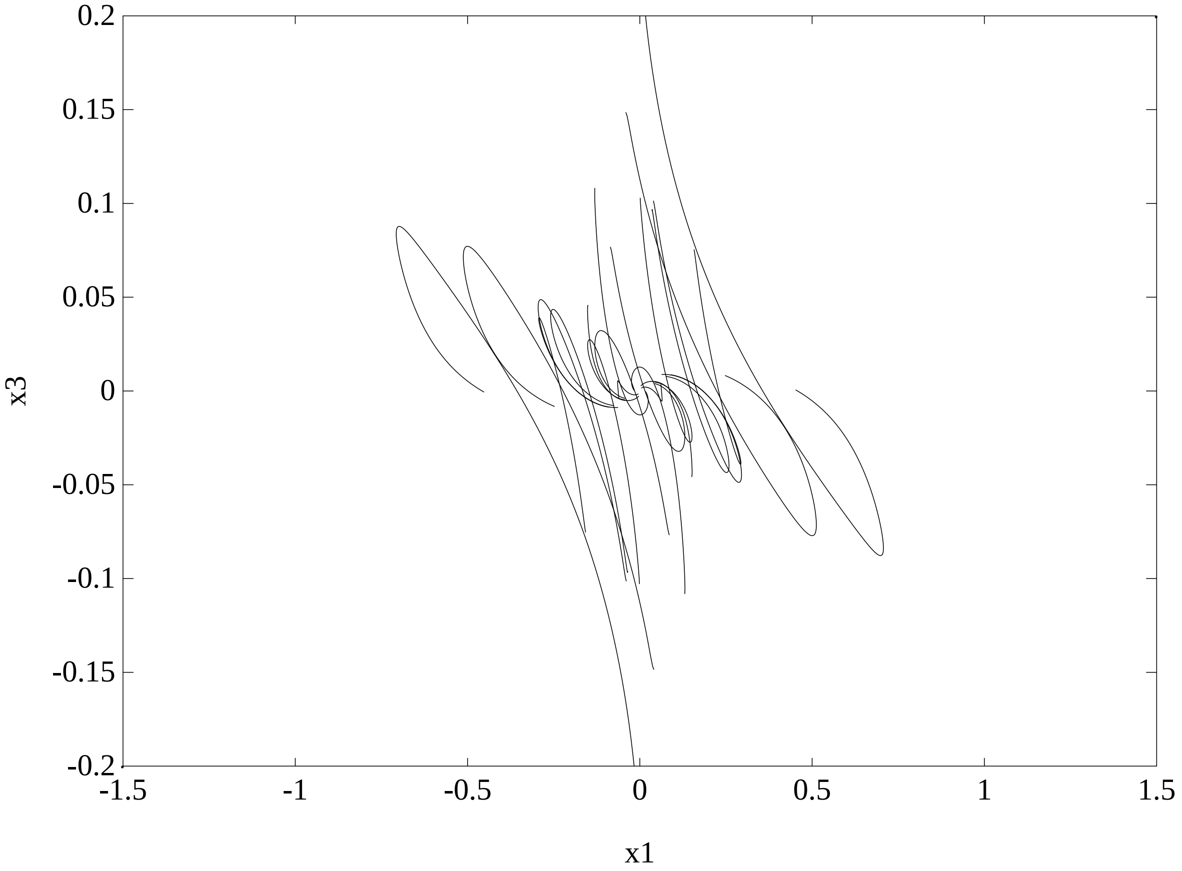

Figure 7: *Behaviour of the feedforward neural controlled system close to the target equilibrium point* eq^+*. All trajectories shown on the figure are going to the origin.*

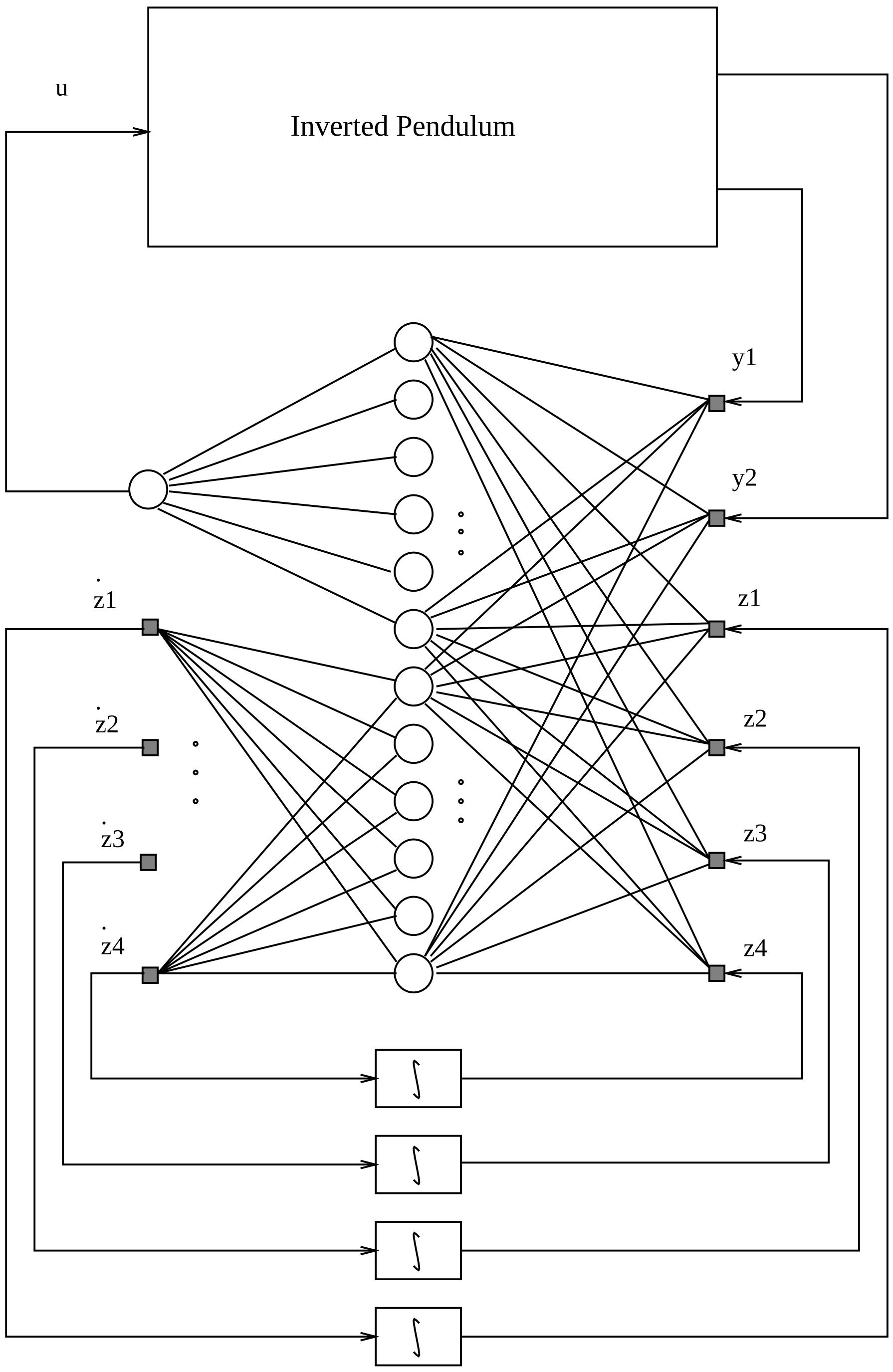

Figure 8: *Inverted pendulum controlled by a recurrent neural network (dynamic output feedback) for performing the swinging up problem. The outputs y_1 and y_2 are the position of the cart and the angle of the pole.*

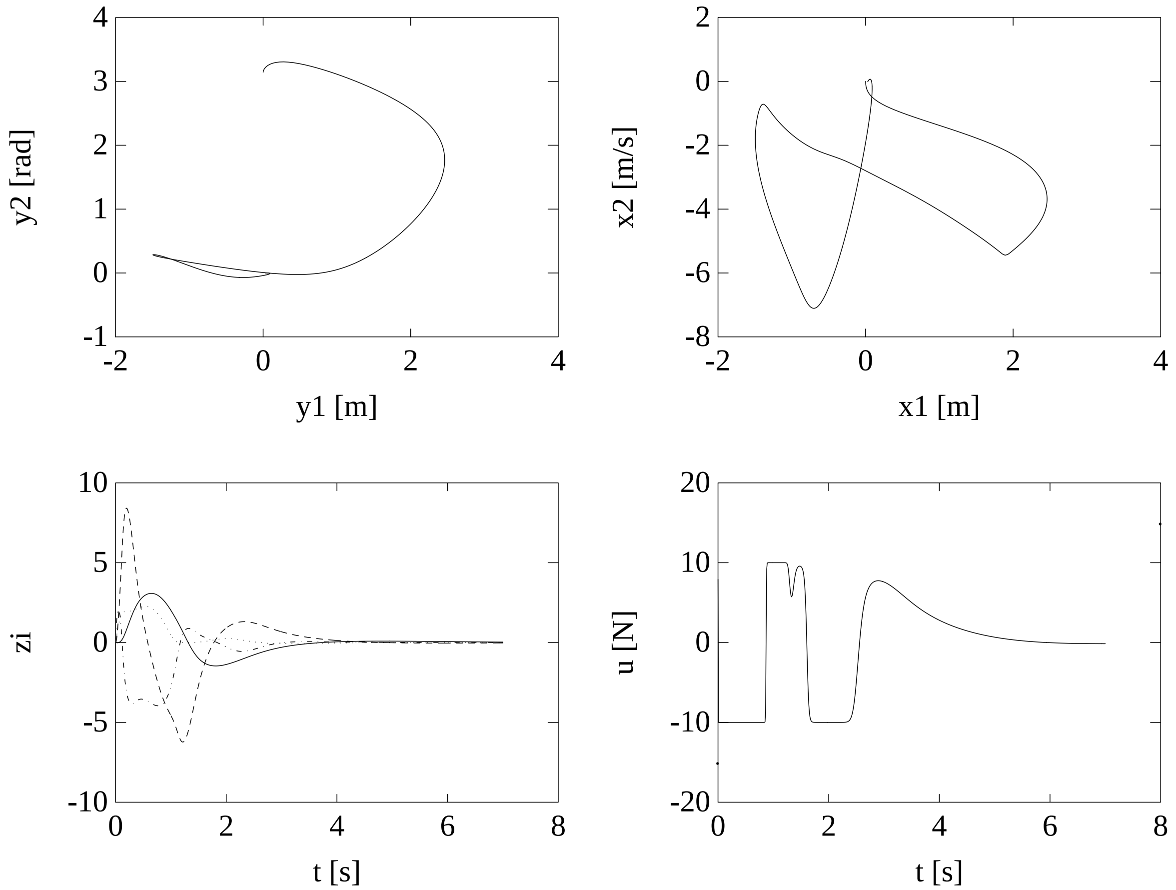

Figure 9: *Solution in state space to the swinging up problem of an inverted pendulum system by a recurrent neural network based on LQG design (initial state* $x(0) = [0\ 0\ \pi\ 0]^t$*,* $z = 0$ *and target equilibrium point* $x = 0$*,* $z = 0$*). Plot of* $z_i(t)$*:* z_1 *(solid line),* z_2 *(dashed line),* z_3 *(dotted line),* z_4 *(dashdot line).*

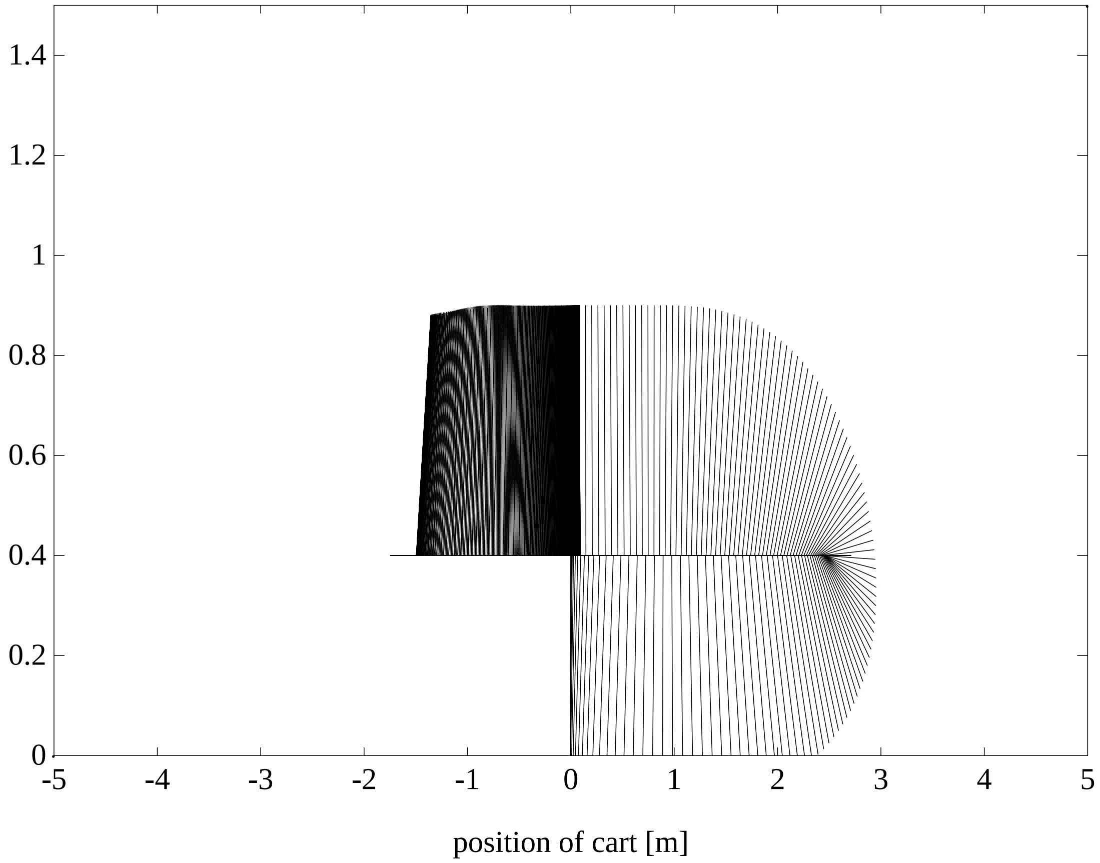

Figure 10: *Evolution of the pole during the transition from* $x(0) = [0\ 0\ \pi\ 0]^t, z = 0$ *to the target equilibrium point* $x = 0, z = 0$ *by recurrent neural net control of Fig.8-9.*

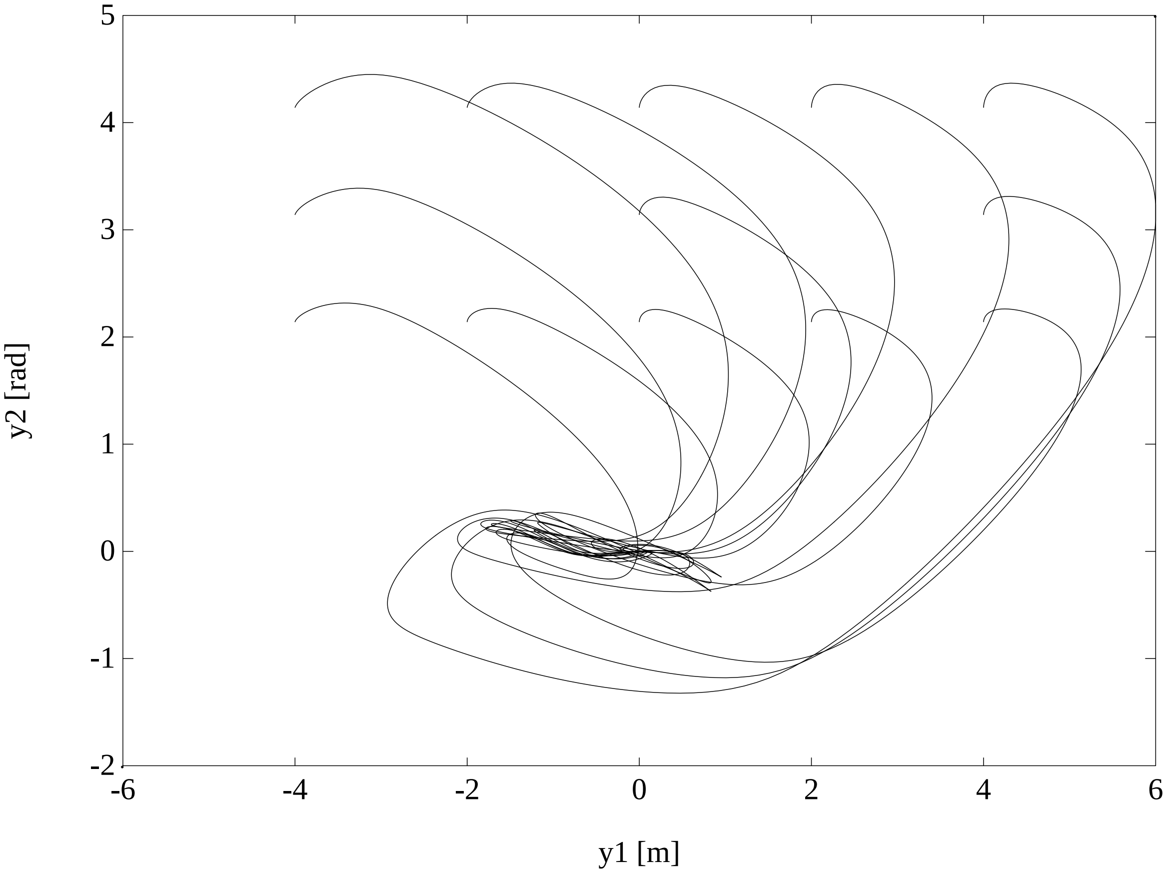

Figure 11: *Robustness of the recurrent neural controller of Fig.8-9 with respect to the initial state $x(0)$ lying in the neighborhood of eq^-: remark that this is superior with respect to the feedforward neural control of Fig.6, because of the existence of a larger region around $x = [0\,0\,\pi\,0]^t$ containing $x(0)$ such that the neural controller is still able to perform the control task of swinging up and locally stabilize at the endpoint.*

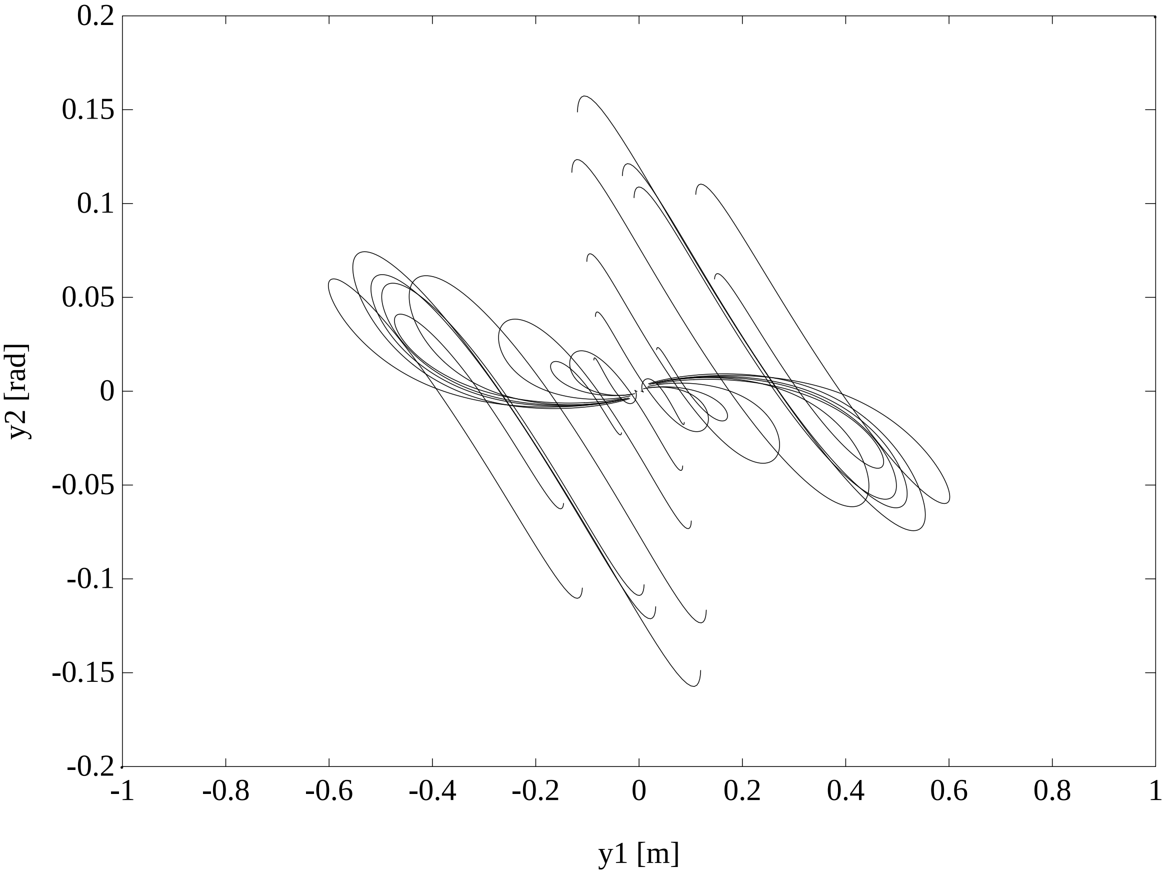

Figure 12: *Behaviour of the recurrent neural controlled system close to the target equilibrium point* eq^+. *All trajectories shown on the figure are going to the origin.*