Evolving Neurocontrollers for Balancing an Inverted Pendulum [*]

Frank Pasemann
Max-Planck-Institute for Mathematics in the Sciences
D-04103 Leipzig, Germany
email: f.pasemann@mis.mpg.de

Abstract

The paper introduces an evolutionary algorithm that is tailored to generate recurrent neural networks functioning as nonlinear controllers. Network size and architecture as well as network parameters like weights and bias terms are developed simultaneously. There is no quantization of inputs, outputs or internal parameters. Different kinds of evolved networks are presented that solve the pole-balancing problem, i.e. balancing an inverted pendulum. Especially controllers solving the problem for reduced phase space information (only angle and cart position) use a recurrent connectivity structure. Evolved controllers of "minimal" size still have a very good benchmark performance.

[*]appeared in *Network: Computation in Neural Systems*, **9**, 1998, pp. 495–511.

1 Introduction

There is a threefold motivation for the investigations reported in this paper. The first one comes from the observation that already small recurrent neural networks can display a rich spectrum of dynamical phenomena. Their dynamical properties depend on parameters like synaptic weights and bias terms, but also on the placement of excitatory and inhibitory connections along closed signal loops. For the case of discrete dynamics and graded neurons compare for instance results in [6], [9], [23], [24], [25], [32]. Recurrences involving excitatory as well as inhibitory connections are of course characteristic also for biological brains. This suggests the hypothesis that higher level information processing or "cognitive" abilities of biological as well as artificial brains are based on the complex dynamical properties of interacting dynamical neuromodules.

Although this seems to be an appealing hypothesis, to date there is no general learning rule able to implement a specific dynamical property into a recurrent network such that it contributes to "cognitive" functions. It is even not at all clear what kind of recurrent structure to choose for a specific task because in general there is no unique structure-function-relationship. This motivated the development of an evolutionary algorithm which generates the structure and optimizes the parameters of neuromodules at the same time. In section 2 we present an algorithm satisfying these conditions; that is, initially neither the number of neurons nor the architecture is specified. Network size and topology as well as network parameters like weights and bias terms are generated simultaneously. Thus, the algorithm not only optimizes parameters but is mainly used for the development of appropriate neural structures.

The third motivation comes from the growing interest in the evolution of recurrent neural networks which can serve as "artificial brains" for autonomous agents [5], [11], [14], [17], [31]. These evolved neural systems acquire their "cognitive" abilities by acting in a sensory-motor loop [18]. On an abstract level the simplest such system – receiving signals from sensors and providing behaviour-relevant signals for effectors – is a neurocontroller which stabilizes an unstable system.

We therefore have chosen the pole-balancing problem as a first test for our evolutionary algorithm. Balancing an inverted pendulum that is mounted on a cart provides a well known class of control problems and often serves as a benchmark problem for trainable controllers [16]. The objective of our investigations was to evolve neural controllers, that not only balance the pole, but at the same time can center the cart and avoid boundaries of the given interval in which the cart can move. To make the problem even more sophisticated we also used reduced phase space information (only cart position and pole angle) as inputs for the controller, expecting recurrent neuromodules to evolve. Furthermore, neurocontrollers with different types of neurons, using *tanh* or the strictly positive function $(1 + e^{-x})^{-1}$ as transfer functions, were evolved to study the dependence of the controller architecture with respect to different "constitutions of effectors". The results obtained are given in section 3.

Besides conventional control techniques, there have been many successful applications of neural networks to the pole balancing problem e.g. [2], [4], [8], [10], [12], [16], [19], [28], [30], [33], [34]. Using graded neurons for the controllers, in contrast to many other results, our approach does not make use of quantization either of the physical phase space variables or of internal network parameters, such as weights and bias terms or output values. Compared with other neural network solutions, the neural structures obtained by our evolutionary algorithm are very small in size and show a comparable or even better performance. Section 4 provides some performance details of the evolutionary algorithm, and in section 5 there is a discussion of the results.

2 The evolutionary algorithm

The combined application of neural network techniques and genetic algorithms turned out to be a very effective tool for solving an interesting class of problems (for a review see e.g. [1], [7], [29], [35]), especially in situations where there is no good guess for an appropriate network architecture or where recurrent dynamic networks should be used for tasks like generation of temporal sequences, recognition, storage and reproduction of temporal patterns, or control problems that require memory to compute derivatives or integrals.

The algorithm described below is inspired by a biological theory of co-evolution and is not based on genetic algorithms. It uses standard additive neuron models with sigmoidal transfer functions and sets no constraints on the number of neurons and the architecture of a network. It develops network topology and parameters like weights and bias terms simultaneously. Using a behaviour based approach to neural systems, the algorithm was originally designed to study the appearance of complex dynamics and the corresponding structure-function relationship in artificial sensory-motor systems for autonomous robots or software agents. For the solution of extended problems (more complex environments or sensory-motor systems) the synthesis of evolved neuromodules forming larger neural systems can be achieved by evolving the coupling structure between modules. This is done in the spirit of co-evolution of interacting species. Because of this background, the evolving process is called "*evolution of neural systems by stochastic synthesis*" or the *ENS*^{3}-algorithm. We suggest that this kind of evolutionary computation is better suited for evolving neural networks than genetic algorithms.

To start the *ENS*^{3}-algorithm, we first have to decide on the type of neurons to use for the network. We prefer to have the same type of neurons for output and internal units; the input units are used here as buffers as in feedforward networks. The number of input and output units is chosen according to the definition of the problem given in terms of incoming sensor and outgoing motor signals. Nothing else is determined, neither the number of internal units nor their connectivity, that is, self-connections and every kind of recurrence are allowed, as well as excitatory and inhibitory connections, but backward connections to input units are prohibited.

To evolve the desired neuromodule we consider a population $p(t)$ of $n(t)$ neuromodules undergoing a variation-evaluation-selection loop, i.e.

$$p(t+1) = \mathbf{S} \cdot \mathbf{E} \cdot \mathbf{V}\, p(t) \quad . \tag{1}$$

The three major steps of this variation-evaluation-selection loop are given as follows:

The *variation operators* $\mathbf{V}$ include insertion and deletion of neurons, insertion and deletion of connections, and alterations of bias and weight terms. The associated operators acting on the ith member of the population p are denoted by N_i^+, N_i^-, C_i^+, C_i^-, N_i^* and C_i^* respectively. A variation pass is then described by

$$\mathbf{V}: \quad p(t) \mapsto \prod_{i=1}^{n(t)} C_i^* C_i^+ C_i^- N_i^* N_i^+ N_i^- \; p(t) \quad .$$

The operators $N_i^{*,+,-}$, $C_i^{*,+,-}$ are of stochastic character. The chance that they will execute their respective action is determined by fixed per-neuron and per-connection probabilities. In a more complex version, the variation operator $\mathbf{V}$ may also induce the exchange of entire subnetworks between members of the population p.

In the next step, the performance of each individual neuromodule of a population is evaluated. Thus the *evaluation operator*

$$\mathbf{E}: \quad p(t) \mapsto (p(t), e(t))$$

is defined problem-specifically. In its simplest form, the performances $e_i(t)$ of the $n(t)$ networks in the population $p(t)$ are mutually independent. As an example, for a classification task the performance of each member of the population could be based on an error function [13], like those utilized by the backpropagation learning algorithm. The evaluation operator usually will be deterministic; more sophisticated versions may also account for network size and past performance. Furthermore, interactions between members of the population can be defined via an artificial sensory-motor loop opening up the potential for co-evolution dynamics.

Differential survival of the varied members of the population is defined by the selection operator. Possible definitions range from (a) probabilistic survival according to evaluation results to (b) winner-takes-all selection. The *selection operator* is given by

$$\mathbf{S}: \quad (p(t), e(t)) \mapsto \prod_{i=1}^{n(t)} R_i^{\nu(e_i(t))} \; p(t) \quad .$$

Here R_i is the reproduction operator for the ith network in the population p. Consequently, $\nu(e_i(t))$ copies of each such network are passed to the new population. In case (a), applied in the following, these integer numbers ν are stochastic variables drawn e.g. from Poisson distributions with mean values larger than 1 if $e_i(t) > \sum_{i=1}^{n(t)} e_i(t)/n(t)$ and mean values smaller than 1 for the other networks in the population p. Case (b) is defined by $\nu(e_i(t)) = n(t)$ if $e_i(t) = \max_i e_i(t)$ and $\nu(e_i(t)) = 0$ otherwise. The number of module copies passed to the next population depends on its performance.

The evolution of the population p is then generated by repeated application of the mapping $p(t) \mapsto \mathbf{SEV}\ p(t)$ on the initial population $p(0)$. In consequence of the selection process the average performance of the population will in general either stay the same or increase. So after repeated passes through the variation-evaluation-selection loop, individual neuromodules that solve the considered problem have built up in the population.

3 Evolving pole-balancing controllers

The cart-pole system, to be controlled by the neurocontroller, is given by an inverted pendulum mounted on a cart. The cart can move in a bounded interval. We want to evolve controllers that satisfy three objectives simultaneously: balancing the pole, avoiding the interval boundaries, and centering the cart.

We use the standard benchmark values for this problem defined for instance in [3]; that is, the cart position x is bounded by the interval $-2.4 < x < 2.4$ $[m]$, the pole angle θ by $-12 < \theta < 12$ $[°]$. The force F applied to the cart, providing the controlling signal, varies continuously between $-10 < F < 10$ $[N]$. We do not use friction terms like e.g. [4] because, as stated in [16], they are too small to generate interesting effects, and on the other hand they may cause the stopping of the cart in cases where controllers would allow for instance small oscillations. For simulations of the cart-pole system we use the following equations

$$\ddot{\theta} = \frac{m\, g\, sin\,\theta - cos\,\theta\,(F + m_p\, l\, \dot{\theta}^2\, sin\,\theta)}{l\,(\frac{4}{3}m - m_p\, cos^2\,\theta)}\ ,$$

$$\ddot{x} = \frac{F + m_p\, l\,(\,\dot{\theta}^2\, sin\,\theta - \ddot{\theta}\, cos\,\theta\,)}{m}\ ,$$

where $g = 9.81$ $[m/s^2]$ denotes gravitational acceleration, $m_c = 1.0$ [kg] and $m_p = 0.1$ [kg] mass of cart and pole, respectively, $m := (m_c + m_p)$, $l = 0.5$ [m] half of pole length, and F denotes the force applied to the cart. The cart-pole dynamics is computed by using Euler discretization with time step $\Delta = 0.01$ [s].

For the neural controller we will use the standard additive neuron model with sigmoidal transfer function σ, i.e. the discrete dynamics of a neurocontroller with

n units is given by

$$a_i(t+1) := \theta_i + \sum_{j=1}^{n} w_{ij}\sigma(a_j(t)) \quad , \quad i = 1, \ldots, n \; , \tag{2}$$

where a_i denotes the activity, $o_i = \sigma(a_i)$ the output, and θ_i the bias term of neuron i; w_{ij} denotes the weight from neuron j to neuron i. All neuron states are updated simultaneously with the states of the cart-pole system. Representing a network by its connectivity matrix w we later also use the notation $w_{i0} := \theta_i$ describing the bias term as weighted output of a bias unit 0 with constant output 1.

A failure signal is given if $|x| > 2.4$ or $|\theta| > 12°$ or balancing time t exceeds a given time t_{max}. The fitness function f for the evaluation of an individual module takes into account costs for each neuron and for each connection (to obtain networks of minimal size), and the balancing time until failure. Furthermore, the applied force integrated over the balancing time can enter the fitness function. This will optimize the applied force to balance the pole, that is, in general this will minimize oscillations of the cart and/or the pole. Thus, the fitness function for an evaluated module has the general form

$$f := P - cost_n \cdot N_n - cost_s \cdot N_s - cost_F \cdot I_F \tag{3}$$

where P denotes the output performance of a module given by

$$P := \sum_{i=1}^{n} \Delta \cdot (0.5 \cdot (1 - \frac{15 \cdot |\theta(i)|}{\pi}) + 0.5 \cdot (1 - \frac{|x(i)|}{2.4})) \quad ,$$

with n the maximal number of iterations; i.e. the maximal balancing time is $t_{max} = n \cdot \Delta$. The constants $cost_n$, $cost_s$, and $cost_F$ describe the costs of a neuron, a synapse, and the applied force, respectively; N_n and N_s denote the number of neurons and of synapses in the module, and the integrated force term I_F is given by

$$I_F = \frac{1}{t_{max}} \sum_{i=1}^{n} |F(i)| \cdot \Delta \quad .$$

Evolved neurocontrollers having full access to phase space information of the cart-pole system can be very simple, as is well known [33]. Here their four continuous input signals are proportional to cart position x and velocity $\dot{x}$, pole angle θ and angular velocity $\dot{\theta}$. Therefore we will only briefly discuss the evolved solutions. In fact, the *ENS³*-algorithm came up with the simplest possible architecture, thus demonstrating that it is able to generate minimal neural structures, at least for this simple control problem.

Controllers using only reduced phase space information, i.e. getting only two input signals proportional to cart position x and pole angle θ, respectively, have to solve a more difficult problem. In this situation it has to be expected, that neurocontrollers make use of a recurrent connectivity structures, because the derivatives $\dot{x}$ and $\dot{\theta}$ now have to be computed. So we will mainly concentrate on the evolution of two-input controllers.

We tested the performance of all evolved configurations on 40 x 40 benchmark initial conditions (x_0, θ_0, $\dot{x_0} = \dot{\theta_0} = 0.0$) represented in the following by squares, as for example in figure 2a; initial conditions, for which the controller balances the pole longer than 120 seconds, are coded in black; the others are white. Outer squares represent initial conditions that will already produce a failure signal. The second type of diagram, as for instance figure 2b, shows the behaviour of the controlled system (displaying x, θ, and F as functions of time) starting from extreme initial conditions. If not stated otherwise, they are given by $x_0 = -2.0$[m], $\theta_0 = -0.18$[rad], $\dot{x_0} = \dot{\theta_0} = 0.0$. The individual methods of handling the control problem become apparent from these diagrams.

3.1 The t-class and s-class controllers

In the following evolved control modules will be represented by their weight matrices; that is, we denote the weight matrix of configuration k by w^k, the weight vector of its neuron i by $w_i^k = (w_{i0}^k, w_{i1}^k, \dots, w_{in}^k)$ with $w_{i0}^k = \theta_i$ denoting the *bias term* of unit i. Furthermore, we will present and discuss two classes of neurocontrollers; one, called *t-class*, uses additive units with anti-symmetric transfer function $\sigma(x) = tanh(x)$, the other one, the *s-class*, uses the strictly positive transfer function $\sigma(x) = (1 + e^{-x})^{-1}$. The first class of controllers needs only one output neuron providing a force $F = 10 \cdot tanh(a_i)$[N], where a_i denotes the activity of the unique output unit i. The s-class needs two output units, i and $i + 1$, giving a force $F = 10 \cdot (\sigma(a_i) - \sigma(a_{i+1}))$[N]. Using the relation

$$tanh(x) = (\sigma(2x) - \sigma(-2x)) \quad , \quad x \in \mathbf{R} \, , \tag{4}$$

a t-class controller may be converted to an equivalent s-class controller by changing bias and weight term according to (4); but s-class controllers – giving output signals of the form $(\sigma(x) - \sigma(y))$ with $x \neq y$ – in general can not be converted to t-class controllers. Anyway, here we want to study whether there will evolve *specific* architectures for each class of controllers. This is due to the above mentioned hypothesis, that structure and function of “cognitive” neural systems - acting in a sensory-motor loop - will depend on placement and constitution of sensors as well as of effectors. Thus, the different types of neurons are chosen to represent different “boundary conditions” for the evolution of control modules. Although the t-class and s-class controllers operate on the same range of input values their evolved architectures – presented in the following – are *not* equivalent in the above sense.

3.2 Four-input-modules

We will first consider controllers having access to the full phase space information of the cart-pole system. Here the four input units of controllers receive the input signals:

$$in_1 := x/2.4\ ,\ in_2 := 15 \cdot \theta/\pi\ ,\ in_3 := \dot{x}/2.4\ ,\ in_4 := 15 \cdot \dot{\theta}/\pi\,.$$

3.2.1 A minimal t-class controller

This class of controllers uses its output unit 5 to provide a force F given by

$$F(t) = 10 \cdot tanh(a_5(t))\ .$$

For initial conditions confined to the benchmark domain, it is well known (see e.g. [33]) that there exist neural network solutions using only the output unit and no internal neurons. In contrast to classical network solutions, which had binary output units $(-1, 1)$ providing a pulsed force to the cart (bang-bang control), here a continuous force is applied. The evolved controller shown in figure 1a demonstrates that our evolutionary algorithm is able to generate such minimal solutions. One of them has weight vector

$$w^1 = w_5^1 = (\,0.0, 10.11, 15.1, 19.97, 2.99\,)\ . \qquad (5)$$

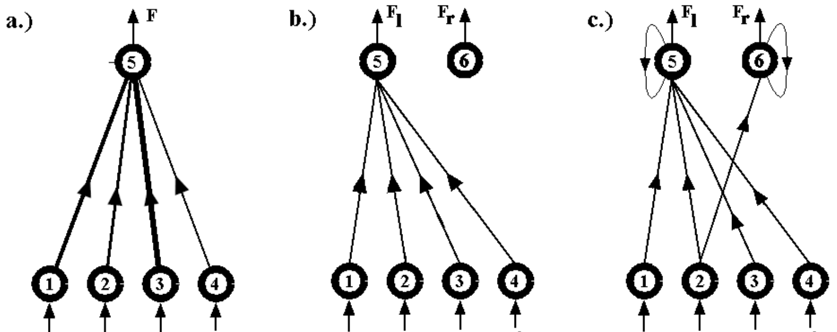
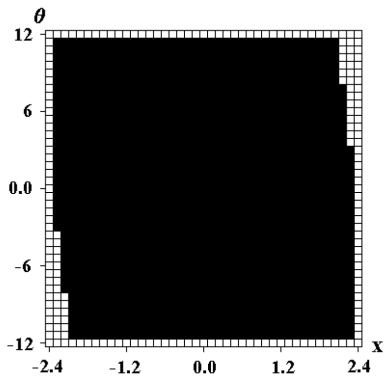


Figure 1: Minimal controllers with four inputs; a.) a t-class controller (w^1), b.) and c.) s-class controllers (w^2 and w^3).

We observe from figure 2a, that the minimal module w^1 avoids the ends of the interval very effectively; furthermore it centers the cart after not more than 10 seconds and balances the pole without oscillations, as can be seen from diagrams in figure 2b. Balancing time is infinitely long when starting on black initial conditions.

a) 

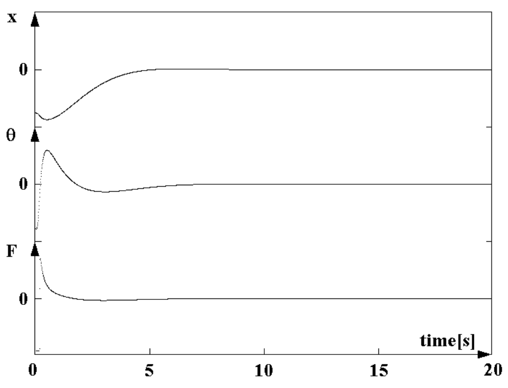
b) 

Figure 2: Controller w^1: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta = -0.18$.

Although the module was evolved with initial conditions x_0 and θ_0 inside the benchmark domain, we observed that it performs well also on initial conditions far outside this domain. Other evolved t-class modules, using for instance one hidden neuron, performed equally well. Some of them utilized also internal oscillators, which came into action only if the physical system reached certain critical phase space domains. If there was no optimizing condition for the applied force (i.e. $cost_F = 0$), almost all solutions solved the balancing problem with a continuously oscillating force and cart.

3.2.2 Two s-class controllers

This class of controllers gets the same input values as controller w^1 but uses output units 5 and 6 with transfer function $\sigma(x) = (1 + e^{-x})^{-1}$ to provide a force

$$F(t) = 10 \cdot (\sigma(a_5(t)) - \sigma(a_6(t))) \ .$$

An s-class controller w equivalent to w^1 will have each output unit connected with all four inputs and $w_5 = -w_6 = 2 \cdot w^1_5$. Thus, the following two evolved s-class controllers w^2 and w^3 are not equivalent to w^1. They are given by (6) and (7) with architectures shown in figures 1b and 1c.

$$w^2 = \begin{pmatrix} w^2_5 \\ w^2_6 \end{pmatrix} = \begin{pmatrix} -0.22 & 8.85 & 11.99 & 17.54 & 3.02 & 0.0 & 0.0 \\ -0.22 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}, \tag{6}$$

$$w^3 = \begin{pmatrix} w^3_5 \\ w^3_6 \end{pmatrix} = \begin{pmatrix} -2.94 & 10.26 & 38.0 & 34.27 & 8.42 & -3.94 & 0.0 \\ -3.35 & 0.0 & -4.1 & 0.0 & 0.0 & 0.0 & 5.58 \end{pmatrix}. \tag{7}$$

Both controllers will balance the pole and center the cart in less than 10 seconds starting from a large domain of benchmark initial conditions (compare figures 2a and 3a). There is no swing around zero of the cart even starting from extreme initial conditions, and also the pole does not oscillate (compare figures 3b and 4b).

Controller w^2 from figure 1b reproduces the architecture of t-class controller w^1, but is less effective in controlling the critical (lower-left, upper-right) corners of the benchmark domain, because it can apply only half of the allowed force. The constant output of neuron 6 is $\sigma(-0.22) = 0.445$, i.e. the force can vary only between $-0.45 < F < 4.55$. The controller w^3 uses the self-connections of output units for more effective control of the critical corners of the benchmark domain (figure 4a) and slows down cart and pole much faster (compare figure 4b).

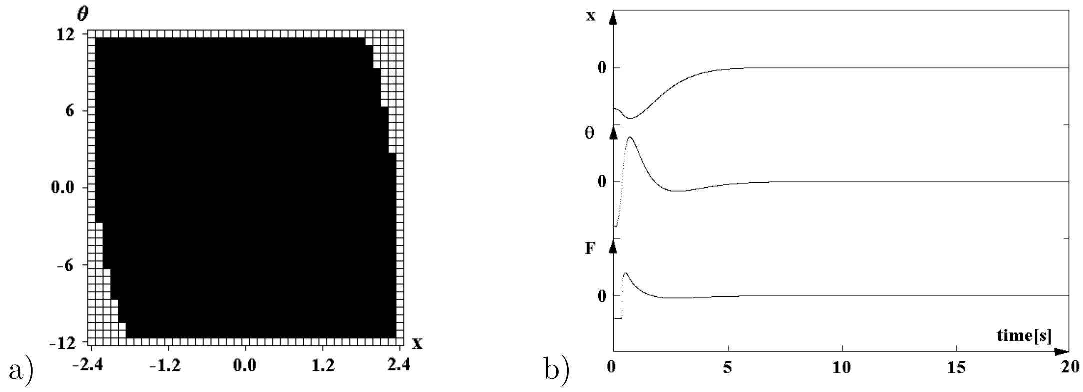


Figure 3: Controller w^2: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -1.9$, $\theta_0 = -0.18$.

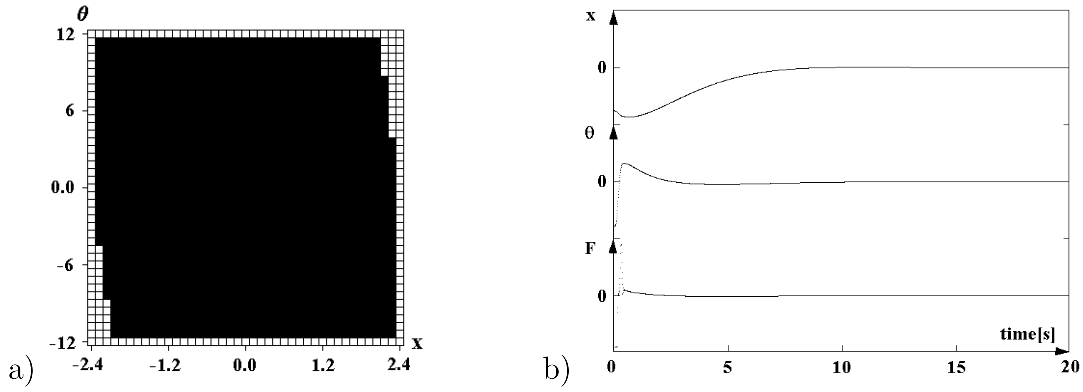


Figure 4: Controller w^3: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

3.3 Two-input-modules

Reducing the inputs to the control module to only the cart position and the pole angle makes the problem for the controller more sophisticated. It now has to compute the derivatives $\dot{x}$ and $\dot{\theta}$, and therefore modules with recurrent connections should be expected. As inputs we choose again for both types of controllers

$$in_1 := x/2.4 \quad , \quad in_2 := 15 \cdot \theta/\pi \ .$$

3.3.1 The t-class controllers

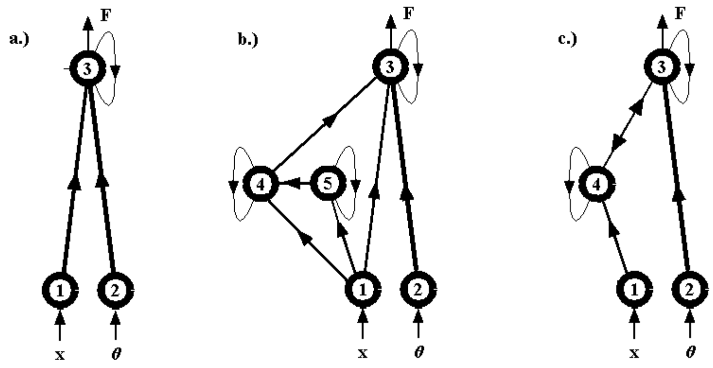


Figure 5: Three examples (w^4, w^5, w^6) of evolved t-class neurocontrollers.

Among many other networks, including even larger ones, the evolutionary algorithm came up with the three architectures depicted in figure 5. Their configurations are given by the following weight matrices:

$$w^4 = w^4_3 = (\, 0.0 \quad 0.15 \quad 4.59 \quad -0.95 \,) \ , \tag{8}$$

$$w^5 = \begin{pmatrix} w^5_3 \\ w^5_4 \\ w^5_5 \end{pmatrix} = \begin{pmatrix} 0.0 & -0.1 & 7.56 & -0.82 & -0.37 & 0.0 \\ 0.0 & -17.31 & 0.0 & 0.0 & 0.75 & -10.93 \\ 0.0 & -0.08 & 0.0 & 0.0 & 0.0 & 0.95 \end{pmatrix} \ , \tag{9}$$

$$w^6 = \begin{pmatrix} w^6_3 \\ w^6_4 \end{pmatrix} = \begin{pmatrix} 0.0 & 0.0 & 3.16 & 0.32 & -0.27 \\ 0.0 & -9.35 & 0.0 & 10.81 & 3.56 \end{pmatrix} \ . \tag{10}$$

As can be seen from figures 6a, 7, 9, they all balance the pole longer than 120 seconds on a large (x, θ)-domain of initial conditions with $\dot{x}_0 = \dot{\theta}_0 = 0$.

The simplest evolved t-class solution w^4 (figure 5a) given by (8) uses no hidden neuron, but only the output neuron with an inhibitory self-connection. As figure 6a demonstrates, it does balancing and wall avoiding for a large domain of initial conditions; but it does not center the cart, i.e. the cart keeps oscillating around zero with an apparently constant amplitude corresponding to its initial position, as can be seen from figure 6b.

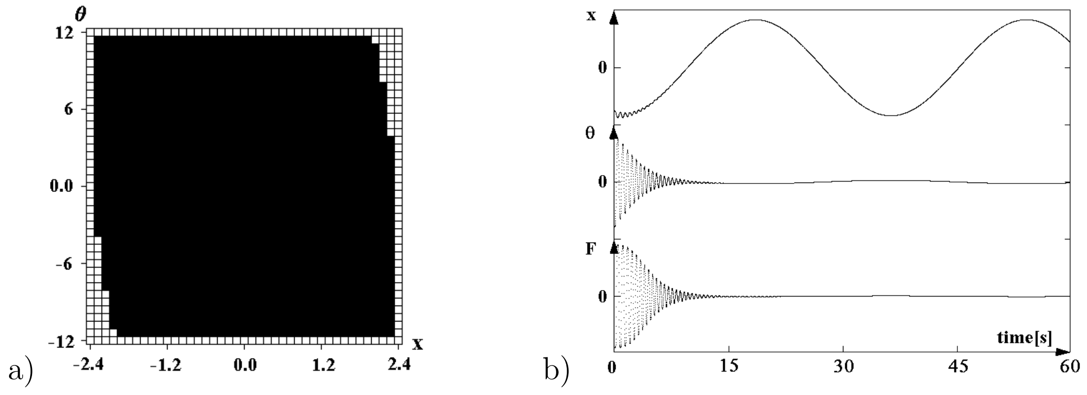


Figure 6: Controller w^4: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -1.9$, $\theta_0 = -0.18$.

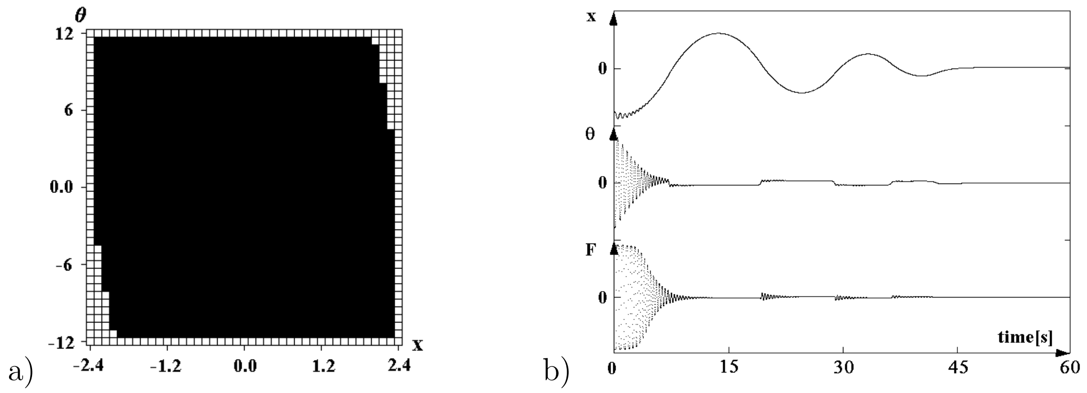


Figure 7: Controller w^5: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

The second solution w^5 (figure 5b) given by (9) is interesting not only because it has an optimal performance (compare figures 7a and 7b) but also because it represents a solution with two coupled modules. It centers the cart with only a few damped oscillations and balances the pole with zero oscillations in less than 60 seconds from almost all benchmark initial conditions; figure 7b gives an example.

To verify that this solution is in fact a modularized network with neurons 1, 2 and 3 reproducing the balancing module of figure 5a, and neurons 1, 4 and 5 functioning as a cart centering network, we evolved controllers that only had to center the cart starting from any position in the interval $|x| < 2.4$. Using two inputs (x and $\dot{x}$) simple solutions evolved that brought the cart into the central position without any swing around zero. Using only the cart position x as input, the cart still oscillated around zero, but now with a damped oscillation, bringing

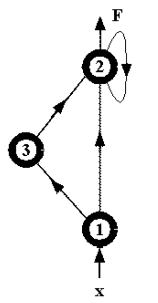


Figure 8: A control module centering the cart using only cart position x as input.

the cart finally to rest at zero. Such an evolved solution w^{cart} was given by

$$w^{cart} = \begin{pmatrix} w_2^{cart} \\ w_3^{cart} \end{pmatrix} = \begin{pmatrix} -0.68 & -19.98 & -0.72 & -5.97 \\ -0.12 & -3.15 & 0.0 & 0.0 \end{pmatrix} \quad , \tag{11}$$

and is shown in figure 8: It uses a delay line (w_{31}^{cart} and w_{23}^{cart}) to satisfy the task; the self-connection of output unit 2 simply smoothes the movement of the cart. Thus, the evolved controller w^5 can be understood in terms of two submodules; the cart centering module w^{cart} acts on the pole balancing module w^4 via the connection w_{34}^4. From this simple example, we may already deduce that interacting modules may retain their architecture, but will re-adapt their internal parameters (here synaptic weights and bias terms) to achieve an effective resulting action.

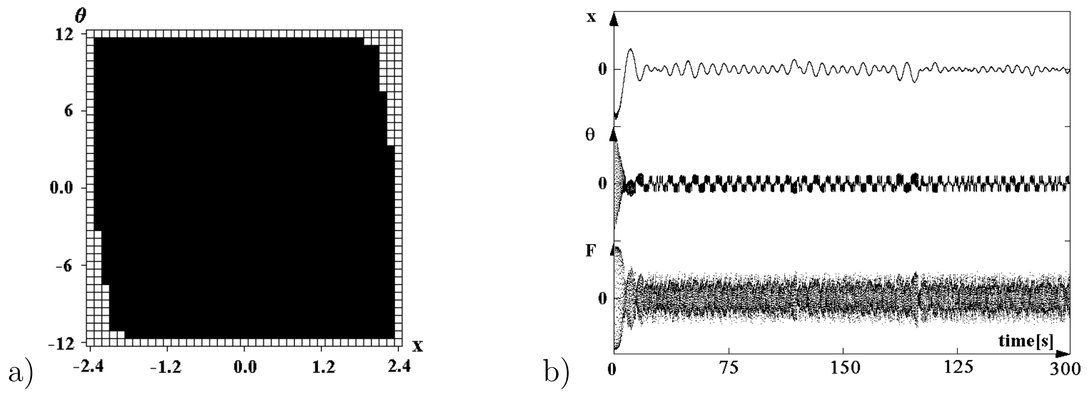


Figure 9: Controller w^6: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

The third solution w^6 (figure 5c) given by (10) uses two recurrent loops (one self-connection w_{33}^6 and the loop (w_{34}^6, w_{43}^6)) to solve the problem in an unconventional way: After bringing the cart position and pole angle near zero from almost all initial conditions (compare figure 9a), it starts to generate apparently "erratic" control signals, which are nevertheless able to prevent the pole from falling and to keep the cart off the interval boundaries. This can be read from figure 9b, where a five minute recording of cart position x, pole angle θ, and force F is displayed. Analyzing the dynamics of the 2-module composed of neurons 3 and 4 for stationary inputs, we observe quasi-periodic attractors for values I_3, I_4 around zero. Thus, for small x and θ the irregular behaviour of the controlled systems may result from coupling its eigenmodes back to a quasi-periodic control system.

3.3.2 Three s-class solutions

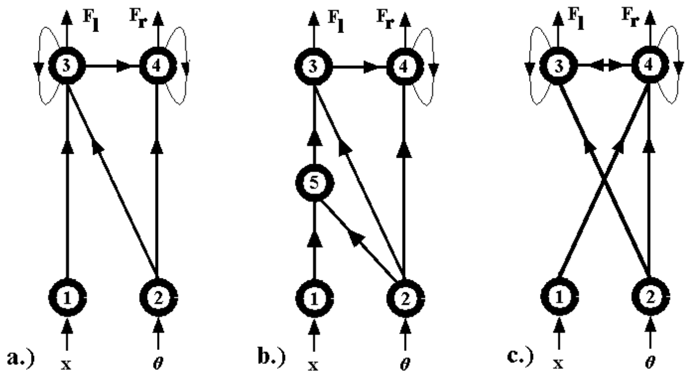


Figure 10: Three examples (w^7, w^8, w^9) of evolved s-class neurocontrollers.

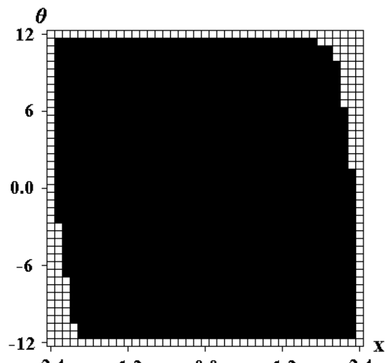
The three neurocontrollers shown in figure 10 are examples of evolved "minimal" configurations for s-class controllers. Their configurations are given by the following weight vectors:

$$w^7 = \begin{pmatrix} w_3^7 \\ w_4^7 \end{pmatrix} = \begin{pmatrix} 2.72 & 21.3 & 49.62 & 3.83 & 0.0 \\ -2.95 & 0.0 & -10.55 & 10.5 & -4.14 \end{pmatrix}, \qquad (12)$$

$$w^8 = \begin{pmatrix} w_3^8 \\ w_4^8 \\ w_5^8 \end{pmatrix} = \begin{pmatrix} -11.72 & 0.0 & 30.11 & 0.0 & 0.0 & 25.84 \\ -4.38 & 0.0 & -13.82 & 16.17 & -7.11 & 0.0 \\ -3.74 & 37.69 & 35.11 & 0.0 & 0.0 & 0.0 \end{pmatrix}, \qquad (13)$$

$$w^9 = \begin{pmatrix} w_3^9 \\ w_4^9 \end{pmatrix} = \begin{pmatrix} -2.74 & 0.0 & 23.85 & -9.79 & 14.69 \\ -2.29 & -10.05 & -35.6 & -10.71 & 12.08 \end{pmatrix}. \qquad (14)$$

All three configurations are not equivalent to any of the t-class controllers w^4, w^5 and w^6, but again they solve the problem for a large domain of benchmark initial conditions (compare figures 11a, 12a, and 13a). But none of them brings cart and pole simultaneously to rest at zero. Instead, with a successful control the final state is characterized by more or less small oscillations around the zero positions (compare figures 11b, 12b, and 13b).

a) 

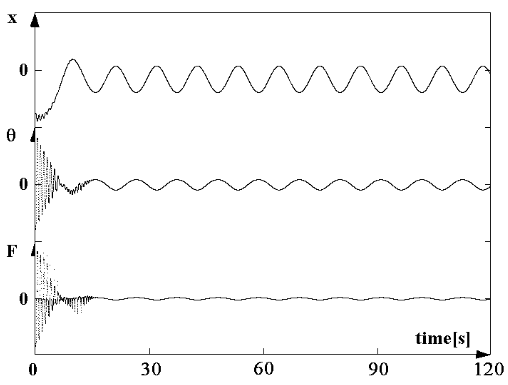
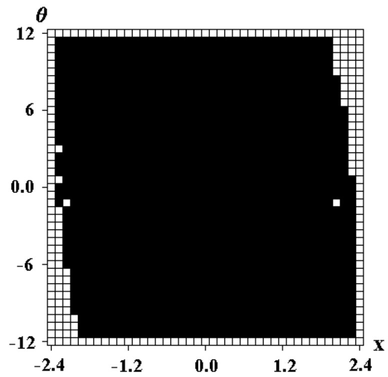
b) 

Figure 11: Controller w^7: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

a) 

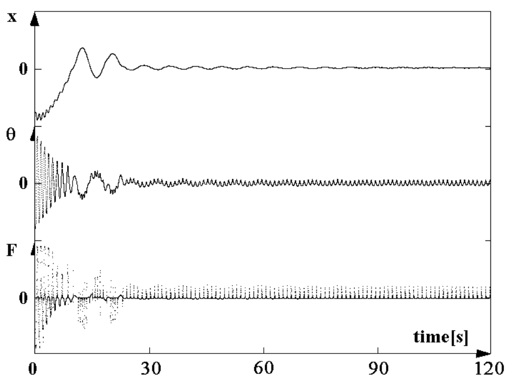
b) 

Figure 12: Controller w^8: a.) and b.) as in figure 11.

Although the three controllers have almost the same benchmark performance, they use different "techniques" to solve the problem. For instance controller w^7 (12) damps the cart oscillations around the origin to an amplitude that is constant after some time. It is endowed with a switchable oscillator realized by the inhibitory self-connection of neuron 4 [22], but this turns out not to be essential here. Self-inhibition with values just above the critical value $w^7_{44} = -4.0$ damps pole oscillations as well and gives the same benchmark results. Correspondingly, the excitatory self-connection w^7_{33} of neuron 3 keeps the amplitude of cart oscillations at a constant value; simulations show, that for $w^7_{33} = 0$ the oscillation amplitude of the cart will slowly grow again after first being damped to a lower value. In contrast to w^7, controller w^8 (13) makes explicit use of its oscillator given by unit 4: it brings the cart almost at rest at zero, but keeps the pole oscillating forever as can be seen from figure 12. It furthermore uses delay lines for the pole angle signals. Controller w^9 (14) is the one with the best benchmark performance (see figures 13a,b). It uses higher periods and even chaotic dynamics

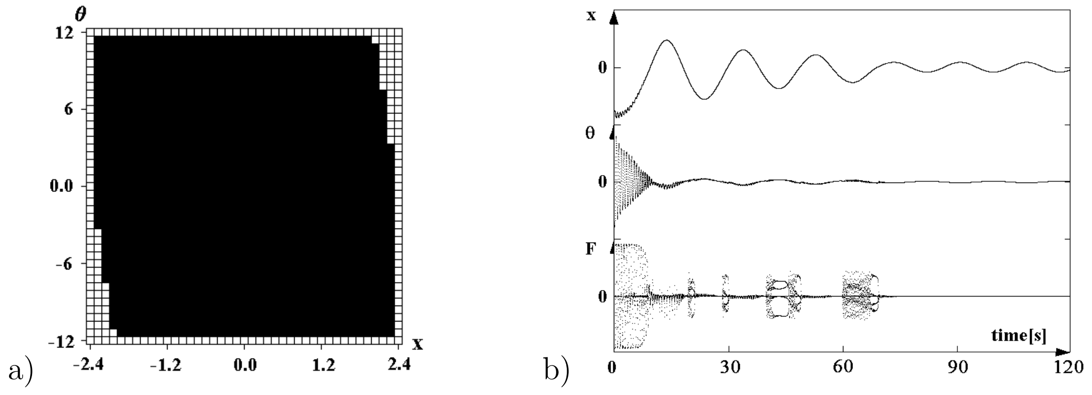


Figure 13: Controller w^9: a.) and b.) as in figure 11.

to stabilize the system in a comparatively short time (around 60 seconds for extreme initial conditions, compare figure 13b). In fact, output units 3 and 4 form a "chaotic neuromodule" [21], and simulations for this configuration show that the controller really uses also the chaotic domain of the module, that is, for specific (stationary) control inputs the dynamics of the output module is determined by a chaotic attractor.

4 Performance of the ENS^3-algorithm

Because the *ENS*^{3}-algorithm is primarily designed to study theoretical aspects of modularized recurrent networks, we were not concerned about statistics or computation time. For classical problems like pattern classification and the like, evolution will not outperform algorithms like backpropagation. This became clear, for instance, when solving the parity-n problems with the *ENS*^{3}-algorithm as reported in [13]. The advantage of this algorithm, however, is that it also generates different unconventional (that is, not strictly layered feedforward) solutions to function approximation, which are interesting to study for theoretical reasons.

While the algorithm is still under development, it is not easy to make final statements on its performance. Up to now, the required computing time depends strongly on the parameter settings - their optimal values in the context of a given problem are not known from the beginning - and on the design of an appropriate fitness function. Parameters of the algorithm are for instance the probabilities for insertion and deletion of neurons and connections, and for alteration of bias and weight terms; furthermore, the costs for neurons, connections and the applied force, the steepness of the selection function, the average population size, the maximal time to solve the problem (stop criterion), and the like.

Starting without any internal neurons or connections a typical run behaves as in the following example where a four-input t-controller is evolved (figure 14).

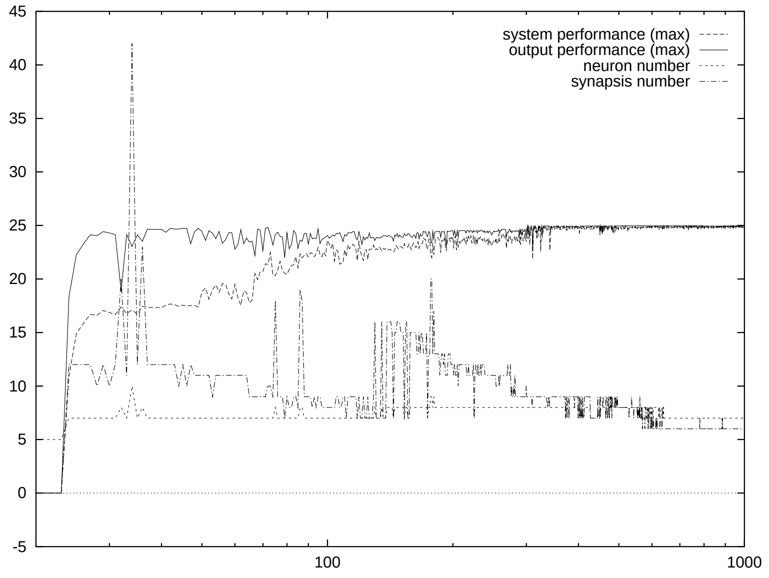


Figure 14: Performance of the *ENS*^{3}-algorithm evolving a four-input t-controller; generation numbers are given on a log scale starting with generation 20. Shown are the performances and the number of neurons and synapses of the best individual in a generation.

The average population size for this run was 50, the maximal balancing time was 20 seconds, and it took about 40 minutes to evolve the 1000 generations on a SUN SPARCstation4. For the interpretation of figure 14 one should know that every generation has to work on different initial conditions for the cart-pole system which are randomly chosen from the benchmark domain. In some generation – here it is generation 24 – there appears at least one individual able to balance the pole. In the following generations different individuals are selected for different initial condition; in general they are of different size and have different connectivity. Typically there is an abundance of neurons and connections - controlled by their costs - until individual modules reach the optimal *output performance* - here around generation 50 (maximum=25). After this phase mainly the *system performance* is optimized by reducing the number of neurons and connections, and by minimizing the expended force; for instance by switching off internal oscillators. Finally – starting around generation 650 – mainly variations of weights and bias terms are then optimizing the system performance on larger regions of the benchmark domain. The resulting four-input t-controller had two internal neurons and six connections; its benchmark performance was comparable to that of controller w^1 shown in figure 2.

Of course, the evolution of two-input modules down to the minimal controllers presented in this paper takes much longer. Most of the time we used 10.000 generations and obtained good results with population sizes between 30 and 50 individuals. Enlarging the size up to 500 showed no better effects.

5 Discussion of results

We have demonstrated that the *ENS^3*-algorithm presented in section 2 can be applied successfully to control problems like balancing an inverted pendulum. It is difficult to compare our results with those from the literature because some authors use different parameters for the cart-pole system or do not demand cart-centering and wall-avoiding. Furthermore, only a few authors present data like figures 2 - 4, 6 - 8 and 11 - 13 of this paper. Most articles discuss four-input controllers with one output neuron using anti-symmetric sigmoid functions or the *sign*-function ("bang-bang control") as transfer functions. In general they train networks of the feedforward type and often use phase space quantization techniques [2], [4], [8], [10], [12], [16], [28], [30], [33], [34]. All of these networks seemingly do not outperform the controller w^1. Even if the simple architecture of controller w^1 is presented (e.g. [16]), the controller results from pure parameter optimization, whereas w^1 is the final product of an evolutionary process which has quite large networks as intermediate steps. Also recurrent networks are discussed, at least for the case of two-input controllers [28], [34]. But again, in these cases the fully recurrent architectures are fixed from the beginning and parameters are optimized by learning rules. For optimization also genetic algorithms are used; for instance in [12], [19] and [34].

Because neither the number of neurons nor the type of connectivity structure is fixed in advance, size and structure of neurocontrollers crucially depend on fitness functions like the one given by equation (3). If costs $cost_n$ for neurons and $cost_s$ for connections are set to low values, then also large networks with more internal neurons (up to ≈ 16) and higher connectivity (up to ≈ 60 synapses) still having a good performance were observed. If there is no cost term $cost_F$ in equation (3) minimizing the force applied to the cart solutions tend to use internal oscillators, keeping the pole balanced by permanent oscillations. These oscillators are realized for instance by inhibitory self-connections [22] or loops of two or three neurons [23]. We also found solutions making use of switched oscillators [22], which come into action only if the physical system enters critical phase space domains.

Furthermore, it is observed that the evolved controllers are quite robust with respects to moderate input noise, moderate variations of the discretization time step Δ, and to the introduction of friction terms (compare e.g. solutions in [26], [27]). On the other hand, varying weight parameters of a controller can have a critical effect; for instance eliminating an internal oscillator - by moving e.g. the strength of a self-connection past a critical value - may reduce the performance of the module drastically. Often also the values for bias terms are critical. For instance, bias terms of output neurons will determine, of course, where on the interval the cart is coming to rest (or around which position it is oscillating); so exact centering corresponds in general to specific bias terms of output neurons.

With respect to module dynamics one finds controllers with "genuine" internal dynamics, i.e. one which is immanent in their structure (e.g. oscillations as for w^8). Here it is functional in the sense that keeping the cart in fast oscillations will balance the pole. But controller w^9 demonstrates that even chaotic dynamics may be used to stabilize the system. In other controllers (e.g. w^6) internal dynamics not caused by the network structure is observed; it seems to be induced by the back-coupling of "motor" actions to the "sensors" in the sensory-motor loop.

The fact that every two-input controller used an inhibitory self-connection or a 2-loop with one inhibitory connection, confirms the effectiveness of recurrent networks as well as the crucial role played by inhibitory connections for an improved neural processing. Moreover, the evolved controller w^5, which has the best benchmark performance of all our examples, suggests that modularity of dynamic recurrent networks is in fact a desirable - if not "natural" - design principle for neurocontrollers. It demonstrates that functionally different modules can co-operate or compete to generate a desired behaviour.

Acknowledgments

The author would like to thank Ulf Dieckmann and Ulrich Steinmetz for many valuable discussions improving the ENS^3-algorithm.

References

[1] Albrecht R F, Reeves C R and Steele N C ed 1993 *Artificial Neural Nets and Genetic Algorithms, Proceedings of the International Conference in Innsbruck, Austria 1993* (Wien: Springer Verlag)

[2] Anderson C W 1989 Learning to control an inverted pendulum using neural networks *IEEE Control Systems Magazine* **9** 31

[3] Anderson C W and Miller W T 1990 Challenging Control Problems *Neural Networks for Control* ed W T Miller, R S Sutton and P J Werbos (Cambridge, MA: MIT Press)

[4] Barto A G, Sutton R S and Anderson C W 1983 Neuronlike adaptive elements that solve difficult learning control problems *IEEE Transactions on Systems, Man, Cybernetics* **13** 834

[5] Beer R D and Gallagher J C 1992 Evolving dynamical neural networks for adaptive behavior *Adaptive Behavior* **1** 91

[6] Blum E K and Wang X 1992 Stability of fixed points and periodic orbits and bifurcations in analog neural networks *Neural Networks* **5** 577

[7] Branke, J 1995 Evolutionary algorithms for neural network design and training - a review *Proceedings 1st Nordic Workshop on Genetic Algorithms and its Applications (1NWGA)* ed J T Alander (Vaasa, Finland: University Proceedings)

[8] Bapi R S, D'Cruz B and Bugmann G 1997 Neuro-resistive grid approach to trainable controllers: A pole balancing example *Neural Computing and Applications* **5** 33

[9] Chapeau-Blondeau F and Chauvet G 1992 Stable, oscillatory, and chaotic regimes in the dynamics of small neural networks with delay *Neural Networks* **5** 735

[10] Chen V C and Pao Y-H 1989 Learning control with neural networks *Proceedings of the International Conference on Robotics and Automation* Scottsdale, AZ 14 -19 May 1989 (Washington, DC: IEEE Comp Soc Press)

[11] Cliff D, Harway I and Husbands P 1993 Explorations in evolutionary robotics *Adaptive Behavior* **2** 73–110.

[12] Dasgupta D and McGregor D R 1993 Evolving neurocontrollers for pole balancing *ICANN'93 Proceedings of the International Conference on Artificial Neural Networks* ed S Gielen and B Kappen (Berlin: Springer-Verlag) p 834

[13] Dieckmann U 1995 Coevolution as an autonomous learning strategy for neuromodules *Supercomputing in Brain Research - From Tomography to Neural Networks* ed H Herrmann, E Pöppel and D Wolf (Singapore: World Scientific) p 427

[14] Floreano D and Mondada F 1996 Evolution of plastic neurocontrollers for situated agents *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* ed P Maes, M Mataric, J A Meyer, J Pollack, H Roitblat and S Wilson (Cambridge, MA: MIT Press-Bradford Books)

[15] Floreano D 1997 Ago ergo sum *Evolving Conciousness* ed G. Mulhauser (Amsterdam: J. Benjamins)

[16] Geva S and Sitte J 1993 A cartpole experiment benchmark for trainable controllers *IEEE Control Systems Magazin* **13** 40

[17] Husbands P, Harvey I, Cliff D and Miller G 1997 Artificial evolution: A new path for artificial intelligence? *Brain and Cognition* **34** 130

[18] Mallot H A 1997 Behavior-oriented approaches to cognition: Theoretical perspectives *Theory in Biosciences* **116** 196

[19] Maričič B 1991 Genetically programmed neural network solving pole-balancing problem *ICANN'91 Proceedings of the International Conference on Artificial Neural Networks* Espoo Finland 24-28 June 1991 ed T Kohonen, K Mäkisara, O Simula and J Kangas (Amsterdam: Elsevier Science Publishers) pp 1273–1276.

[20] Nolfi S, Floreano D, Miglino O and Mondada F 1994 How to evolve autonomous robots: Different approaches in evolutionary robotics *Proceedings of the IV International Workshop on Artificial Life* ed R A Brooks and P Maes (Cambridge, MA: MIT Press)

[21] Pasemann F and Nelle E 1993 Elements of non-convergent neurodynamics *Dynamical Systems - Theory and Applications* ed S I Andersson, A E Andersson and U Ottoson (Singapore: World Scientific)

[22] Pasemann F 1993 Dynamics of a single model neuron *International Journal of Bifurcation and Chaos* **2** 271

[23] Pasemann F 1995 Characteristics of periodic attractors in neural ring networks *Neural Networks* **8** 421

[24] Pasemann F 1995 Neuromodules: A dynamical systems approach to brain modeling *Supercomputing in Brain Research - From Tomography to Neural Networks* ed H Herrmann, E Pöppel and D Wolf (Singapore: World Scientific) p 331

[25] Pasemann F 1998 Structure and Dynamics of recurrent neuromodules *Theory in Biosciences* **117** 1

[26] Pasemann F and Dieckmann U 1997 Evolved Neurocontrollers for pole-balancing *Biological and Artificial Computation: From Neuroscience to Technology - Proceedings IWANN'97* ed J Mira, R Moreno-Diaz and J Cabestany LNCS 1240 (Berlin: Springer-Verlag) p 1279

[27] Pasemann F 1997 Pole-balancing with different evolved neurocontrollers *Artificial Neural Networks - Proceedings ICANN'97* ed W Gerstner, A Germond, A, M Hasler and J D Nicoud LNCS 1327 (Springer-Verlag) p 823

[28] Puskorius G V and Feldkamp L A 1993 Roles of recurrence in neural control architectures *Science of Artificial Neural Networks* Conference Orlando 13 -16 April 1993 Proceedings of the SPIE VOL 1966 p 60

[29] Schaffer J D, Whitley D and Eshelman L J 1992 Combination of genetic algorithms and neural networks: A survey of the state of the art *International Workshop on Combinations of Genetic Algorithms and Neural Networks - Proceedings COGANN-92* ed L D Whitley and J D Schaffer (Los Alamitos, CA: IEEE Computer Society Press)

[30] Selfridge O G, Sutton R S and Barto A G 1985 Training and tracking in robotics *International Joint Conference on Artificial Intelligence - Proceedings IJCAI-85* (Los Angeles, CA) p 670

[31] Spissens P and Torreele J 1992 Massively parallel evolution of recurrent networks: An approach to temporal processing *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life* ed F J Varela and P Bourgine (Cambridge, MA: MIT Press) p 70

[32] Tino P, Horne B G and Giles C L 1995 Fixed points in two-neuron discrete time recurrent networks: Stability and bifurcation considerations *Tech. Rep. UMIACS-TR-95-51 and CS-TR-3461* (College Park, MD: University of Maryland, Institute for Advanced Computer Studies)

[33] Widrow B 1987 The original adaptive neural net broom-balancer *Proc. IEEE Intern. Symp. Circuits and Systems* p 351

[34] Wieland A P 1991 Evolving neural network controllers for unstable systems *International Joint Conference on Neural Networks - Proceedings IJCNN* (Seattle: IEEE Service Center) p II-667

[35] Yao X 1993 A review of evolutionary artificial neural networks *International Journal of Intelligent Systems* **8** 539

Appendix 1: Dynamics of controller w^9 (not published)

In the folling we want to consider the dynamics of the chaotic controller w^9, i.e. we consider the output units of w^9 as a separate 2-module with bias terms $\theta_3 = -2.74$ and $\theta_4 = -2.29$ receiving stationary inputs I_3 and I_4. (Recall that in the controller w^9, inputs can vary as follows: $-23.85 < I_3 < 23.85$ and $-45.65 < I_4 < 45.65$.) That there is in fact a complex dynamical behavior for smaller input values can be read first from the iso-periodic plot displayed in Fig. 14: besides a larger domain corresponding to fixed point attractors (white area), there are two broad strips of period-2 attractors for I_3 around -7.5, $I_4 > 0$, and I_3 around $+7.5$, $I_4 < 0$. At the corners of the period-2 domains, we observe period doubling bifurcations to chaotic attractors. The asymmetry (corners) of the period two domains is the result of starting always with the same initial condition. So we can already anticipate that at the corners a fixed point attractor will coexist with a chaotic attractor (e.g. at $(I_3, I_4) = (-11.0, -3.5)$). Between the two period-2 strips, near the origin we find attractors of higher periods as well as quasiperiodic attractors.

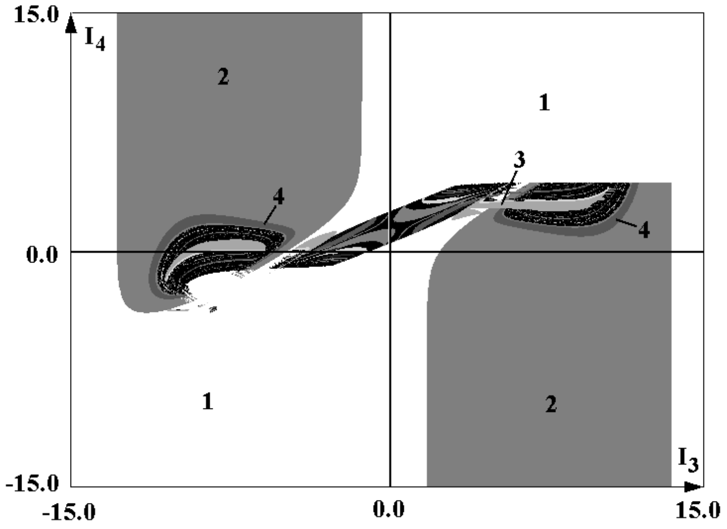


Figure 15: Iso-periodic plot for the output module of controller w^9. Fixed point attractors are coded white, others as shown (see text).

The dynamic complexity of this module becomes even more apparent in Fig. 15, where a bifurcation sequence for I_3 as control parameter is given. The input $I_4 = 0.0$ is fixed and $\overline{o}$ denotes the averaged module output, i.e. $\overline{o} = 1/2 \cdot (o_3 + o_4)$. Starting with a fixed point attractor for $I_3 = -15.0$, it follows a period doubling route to chaos, starting at $I_3 \approx -12.8$ and ending for $I_3 \approx -9.4$. It follows a new sequence of period doubling bifurcations starting from a period-3 attractor and ending at $I_3 \approx -3.8$. Then an interval with periodic and quasi-periodic attractors

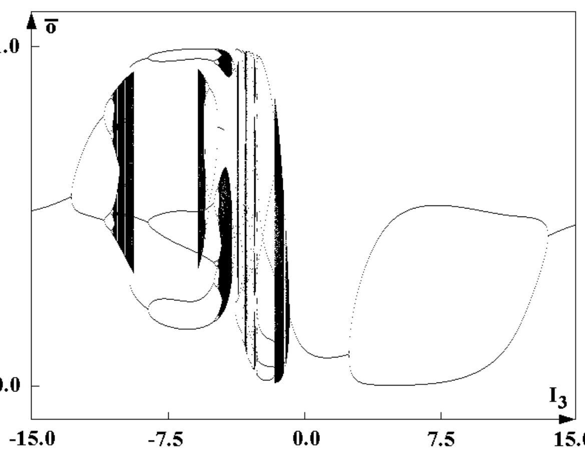


Figure 16: A bifurcation sequence for I_3 with $I_4 = 0.0$ fixed.

follows, finally ending ($I_3 \approx -0.76$) in a fixed point attractor domain, which is interrupted by the intervall $2.5 < I_3 < 13.5$ corresponding to period-2 attractors. What can be also read from Fig. 15 is that around $I_3 = -5.5$ a (backward) period doubling route to chaos coexists mainly with the period-6 attractors of the (forward) period doubling route to chaos. This is visualized by overlaying multiple passes with different initial conditions.

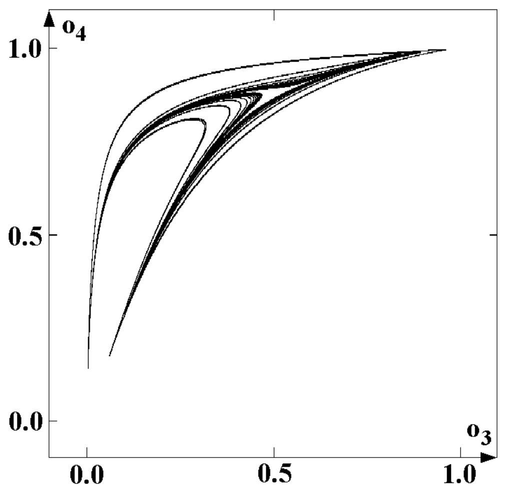


Figure 17: A chaotic attractor for $(I_3, I_4) = (-5.2, -1.0)$.

Acting in the controller w^9, inputs to the module are coming from input neurons 1 and 2 and will cover the domain shown in Fig. 14. Although the controller dynamics in general will not end up on an attractor, the appearance of different periods in the control signal (force F) can be observed also in Fig. 13b.

Appendix 2: Characterization of controllers

(not published in *Network: Computation in Neural Systems*)

Return maps of their force signal

An interesting observation is that the individual action of controllers can be made clearly visible by plotting the first return map of their force signal. This becomes apparent in the following figures where $F(t+1)$ is plotted over $F(t)$ for controllers w^1, and w^5 to w^9. The displayed signals follow always from the same initial condition $x_0 = -2.0$, $\theta = -0.18$.

Figure 17a displays the force signal of an optimal controller as w^1: after a few strong signals it moves down the main diagonal. The periodic signals of 2-input controllers move mainly on elliptic curves to the center. A typical picture is that of controller w^5 as displayed in figure 17b. The "chaotic" behavior of controller w^6 results in the compact cloud of points covering the central region as in figure 18.a. The return maps of 2-input s-class controllers look very differently. The signals of w^7 finally keep oscillating with a small amplitude along the main diagonal (figure 18.b). The final state of controller w^8 is characterized by a force signal that follows a deformed 8-shaped curve around the origin (figure 19.a). Typical for controller w^9 is that the force very often goes down to zero before the final state of small oscillations along the main diagonal is reached (figure 19.b).

Controlled path of the cart-pole system

A different way to characterize the action of the different controllers is to plot the controlled path of the pole-balancing system in (x, θ)-space. Examples for the controllers w^1, and w^5 – w^9 are shown in the figures 21–23 where the extremal initial condition for all examples is given by $x = -2.0$, $\theta = -0.18$. An optimal control - as realized by controller w^1 - gives a path like the one displayed in figure 21a. Controller w^5 finally keeps the pole in vertical position and the car near zero position. The path passes through a characteristic "hysteresis" loop before coming to a halt (figure 21b). The "chaotic" controller w^6 at first damps the oscillations and then keeps the path of the system mainly in the "S"-shaped black domain of figure 22a where it oscillates erraticly for all times.

The s-controllers show a quite different performance. As figures 11b and 13b already indicated the final situation of the cart-pole system, when controlled by w^7 and w^9, corresponds to small anti-phase oscillations of cart and pole, which are smaller for w^9. This corresponds to a path in (x, θ)-space which ends up oscillating along the slightly inclined line in figures 22b and 23b, correspondingly. Controller w^8 also damps oscillations again around an inclined line such that the path finally stays in a small region around the origin (figure 23a).

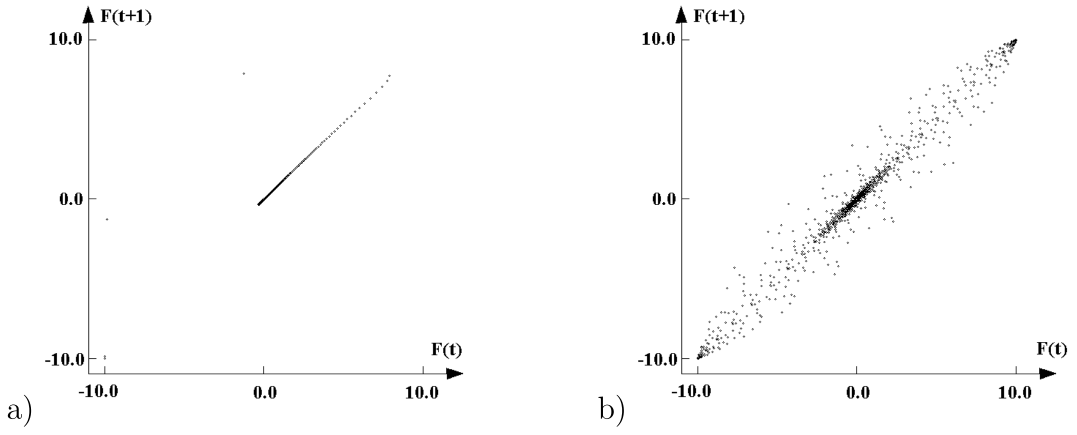


Figure 18: First return map for the force signal of controller a.) w^1 and b.) w^5.

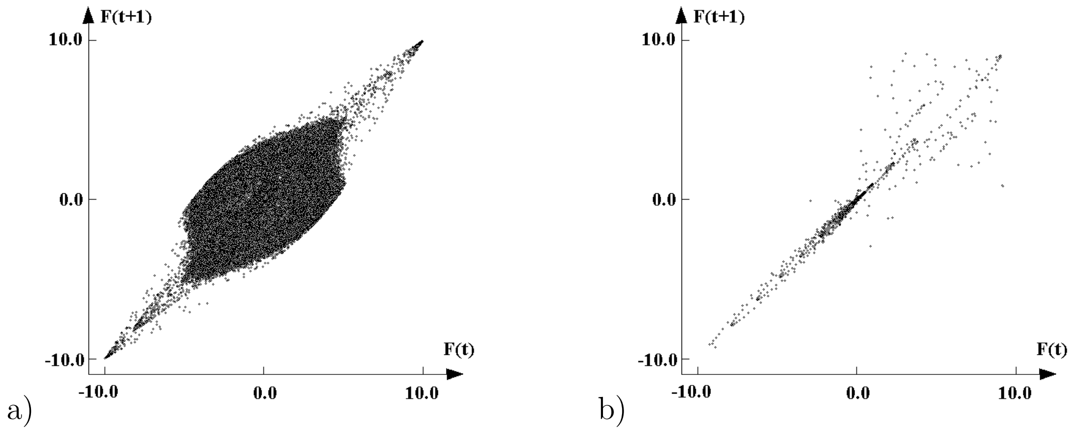


Figure 19: First return map for the force signal of controller a.) w^6 and b.) w^7.

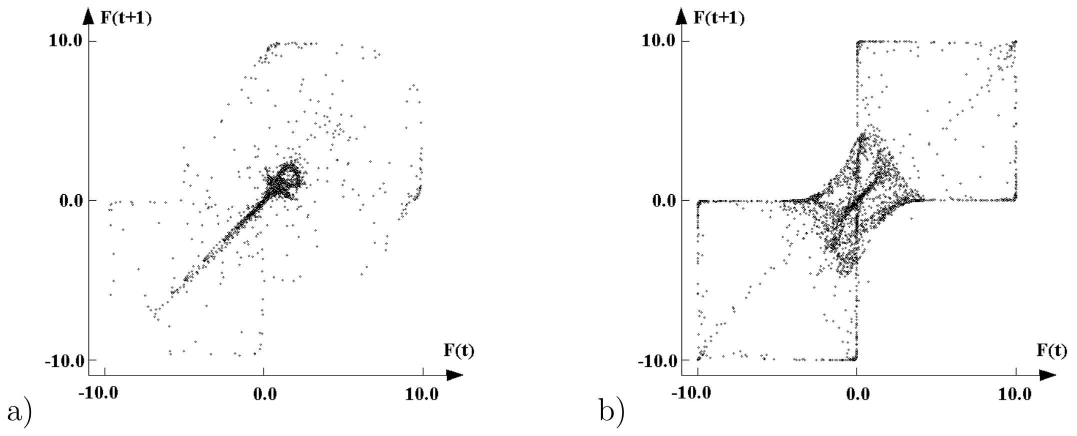


Figure 20: First return map for the force signal of controller a.) w^8 and b.) w^9.

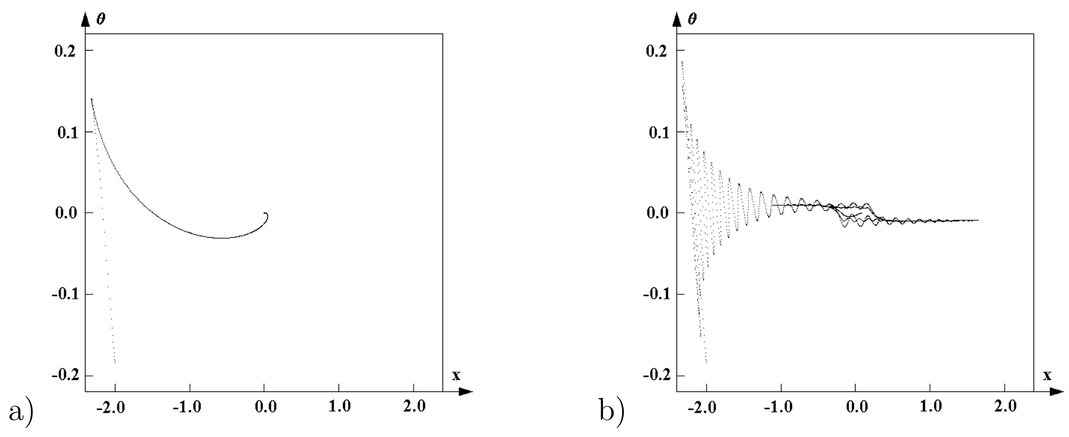


Figure 21: Path in (x, θ)-space for initial condition $x = -2.0$, $\theta = -0.18$ for pole under under control of a.) w^1 and b.) w^5.

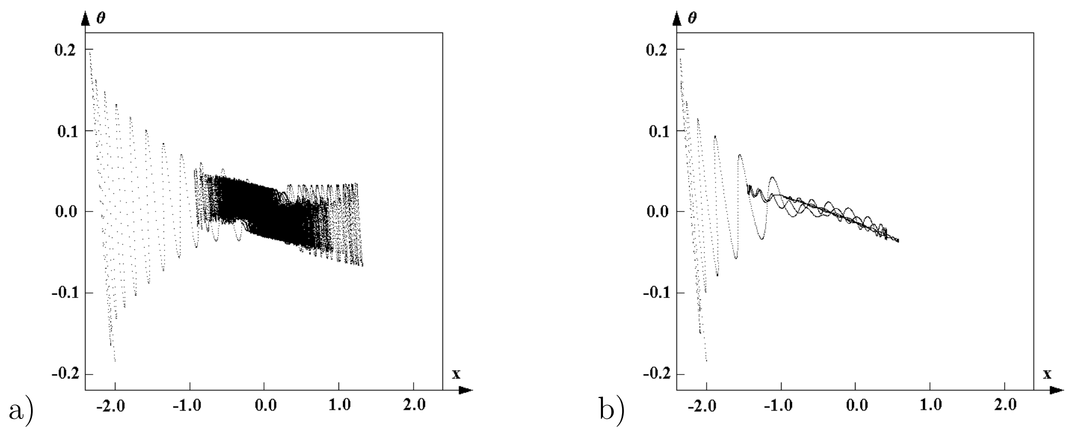


Figure 22: Path in (x, θ)-space for initial condition $x = -2.0$, $\theta = -0.18$ for pole under under control of a.) w^6 and b.) w^7.

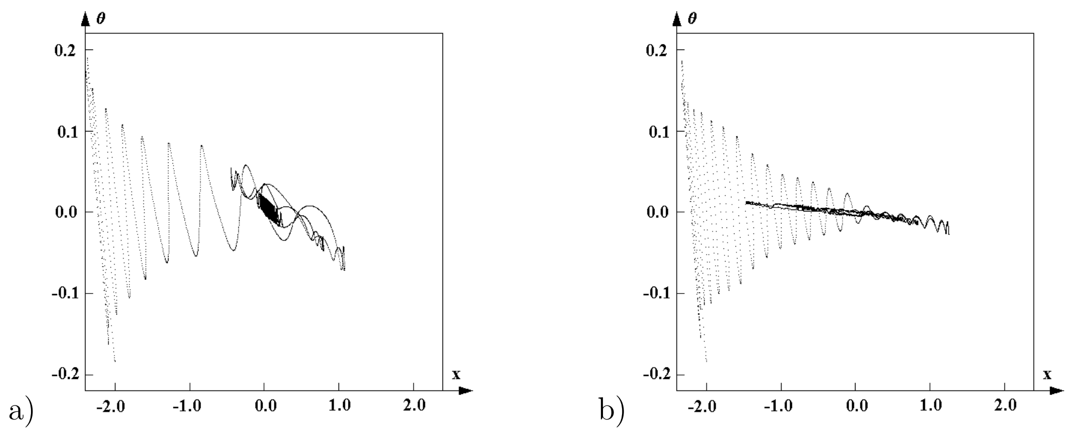


Figure 23: Path in (x, θ)-space for initial condition $x = -2.0$, $\theta = -0.18$ for pole under under control of a.) w^8 and b.) w^9.