Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Evolving neurocontrollers for balancing an inverted pendulum

by

Frank Pasemann

Preprint-Nr.: 35 1997

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

November 12, 1997

Abstract

The paper introduces an evolutionary algorithm, that is tailored to generate 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, with good benchmark performance. Controllers solving the problem for reduced phase space information (only two inputs) use a recurrent connectivity structure and are very small in size. The typical behavior of controllers is characterized by the first return map of their control signals.

*submitted for publication.

1 Introduction

Neural networks with recurrent connectivity structure are an interesting subject to study, because of their rich dynamical behavior spectrum. Viewed as discrete-time parametrized dynamical systems, simulations reveal that small networks are already able to display dynamical features like periodic and chaotic attractors, various bifurcation sequences, hysteresis effects, synchronization and other more general coherence effects [12], [14], [15]. These properties depend on parameters like synaptic weights and bias terms, but also on the placement of excitatory and inhibitory connections along closed signal loops.

On the other hand, biological brains can be considered as modularized neural systems with highly recurrent connectivity; that is, there exist many closed signal loops, composed of excitatory and inhibitory synapses, between functionally different groups of neurons (modules) as well as between individual neurons. This suggests that higher level information processing or cognitive abilities of biological brains are based on the complex dynamical properties of interacting dynamical neural modules. Correspondingly, the idea underlying the investigations presented, is that also artificial neural systems with at least low level cognitive abilities should be developed as modular systems using the dynamical features of recurrent neural structures. By low level cognitive systems we refer to neural systems that can serve as "artificial brains" for "artificial bodies" in natural or artificial environments.

Although, to us, this seems to be an appealing hypothesis, many unsolved problems have to be considered. How can the complex dynamics of a recurrent neural system contribute to functions like perception, memory, prediction, planning, etc.? What kind of module architecture to choose? How to place inhibitory and excitatory connections? How should modules with specific functions interact to provide a desired performance of the composed system? Up to now there is no way to construct versatile dynamical modules, to design an effective interaction of appropriate dynamical modules, or to implement dynamical attractors relevant for a specific cognitive task with the help of a learning algorithm. Furthermore, there is no unique correspondence between the size and structure of a neural system and the specific function it can and has to display. This is one aspect one can learn from the results presented in this paper.

Because of the lack of design principles for dynamical neuromodules and modular neural systems, we think that evolutionary algorithms, which do not restrict the size or the connectivity structure of a system, are at the moment the only way out of this dilemma. 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. To apply this kind of evolutionary process, which is not based on genetic algorithms, one has to work of course in the context of systems acting in a sensori-motor loop, like living beings or autonomous agents (robots or softbots). The simplest such systems – receiving signals from sensors and providing behavior-relevant signals for a "motor" system – are control systems.

As one of the first tests for our evolutionary algorithm we therefore have chosen the pole balancing problem. 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 [9]. The objective of the investigations reported here 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. The results obtained for different input configurations and neuron models, using different transfer functions like *tanh* or the strictly positive sigmoide $(1 + e^{-x})^{-1}$, are described 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], [6], [19], [21]. Using continuous 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 gives a discussion of our results. Appendix 1 analyses the dynamics of the chaotic output layer of one of the evolved controllers, and appendix 2 describes the typical behavior of controllers in terms of the first return map of their control signals.

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], [5], [18], [22]), 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 coevolution 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 behavior 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 sensori-motor systems for autonomous robots or software agents. For the solution of extended problems (more complex environments or sensori-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 coevolution 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 here are used 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 no backward connections to input units are allowed.

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 [8], 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 sensori-motor loop opening up the potential for coevolutionary 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 can not only balance the pole but 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]; taht 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 [9], 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

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

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

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 the neurocontroller is given by

$$a_i(t+1) := \sum_{j=1}^{n} w_{ij}\sigma(a_j(t)) + \theta_i$$

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.

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, taht 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

$$f := P - cost_n \cdot N_n - cost_s \cdot N_s - cost_F \cdot I_F \qquad (2)$$

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

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

It turns out that simulations with populations of 30 - 50 individuals are optimal. During intermediate states of the evolutionary process, the fittest modules may become quite large - more than 40 neurons and 100 synapses - and network size and architecture are often varied. Later in the evolutionary process there appear smaller modules with equally good or even better performance.

In the following we present and discuss different kinds of evolved neurocontrollers. There are two classes of controllers; 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].

Evolved neurocontrollers having full access to phase space information of the cart-pole system can be very simple, as is well known since the paper of [20]. 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 *ENS^3*-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 2-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 Fig. 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 Fig. 2b, shows the behavior 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.

In the following, the 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 $w_i^k = (w_{i0}^k, w_{i1}^k, \ldots, w_{in}^k)$ with w_{i0}^k denoting the bias term of unit i.

3.1 4-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:

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

3.1.1 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. [20]) 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 Fig. 1a demonstrates that our evolutionary algorithm is able to generate such minimal solutions. One of them has weight vector

$$w^1 = w^1_5 = (\, 0.0, 10.11, 15.1, 19.97, 2.99 \,) \ . \tag{3}$$

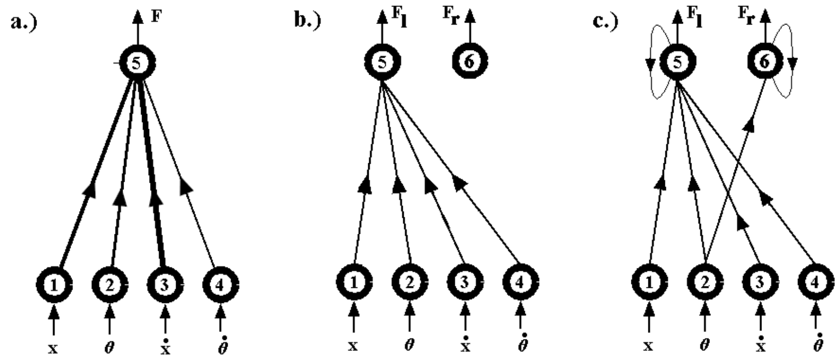


Figure 1: Minimal controllers with four inputs; a.) a t-class controller (w^1), b.) and c.) s-class controllers (w^2 and w^3).

We observe from Fig. 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 Fig. 2b. Balancing time is infinitely long when starting on black initial conditions.

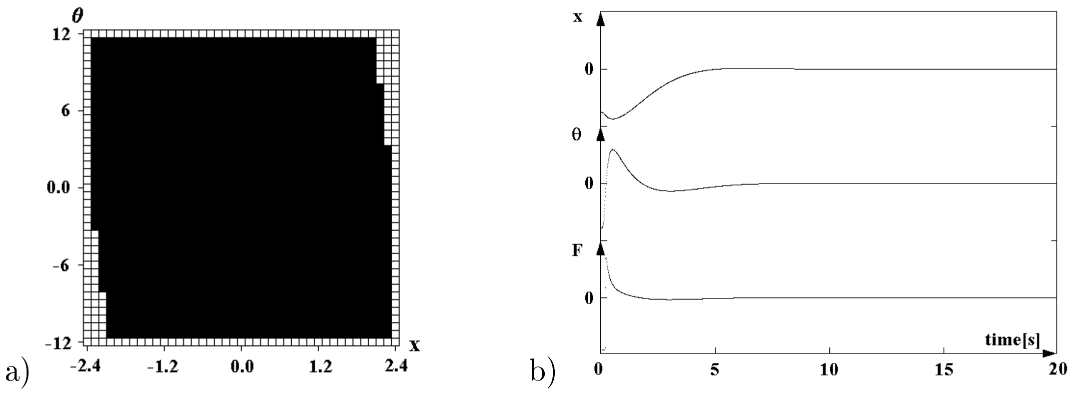


Figure 2: Controller w^1: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta = -0.18$.

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.

3.1.2 Two s-class controllers

This class of controllers uses output units 5 and 6 with transfer function $\sigma(x) = (1 + e^{-x})^{-1}$ to provide a force

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

Using the relations

$$tanh(x) = (2\sigma(2x) - 1) = (\sigma(2x) - \sigma(-2x))$$

an architecture from one class may be converted to an equivalent one in the other class, but here we want to study wether there will evolve specific architectures for each class of controllers. Two such evolved s-class neurocontrollers (w^2 and w^3) are given by (4) and (5) with architectures shown in Fig. 1b and 1c.

$$w^2 = \begin{pmatrix} w_5^2 \\ w_6^2 \end{pmatrix} = \begin{pmatrix} -0.22 & 8.85 & 11.99 & 17.54 & 3.02 & 0.0 & 0.0 \\ -0.22 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \ . \qquad (4)$$

$$w^3 == \begin{pmatrix} w_5^3 \\ w_6^3 \end{pmatrix} = \begin{pmatrix} -2.94 & 10.26 & 38.0 & 34.27 & 8.42 & -3.94 & 0.0 \\ -3.35 & 0.0 & -4.1 & 0.0 & 0.0 & 0.0 & 5.58 \end{pmatrix} \ . \qquad (5)$$

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 Fig. 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 Fig. 3b and 4b).

Controller w^2 from Fig. 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 (Fig. 4a) and slows down cart and pole much faster (compare Fig. 4b).

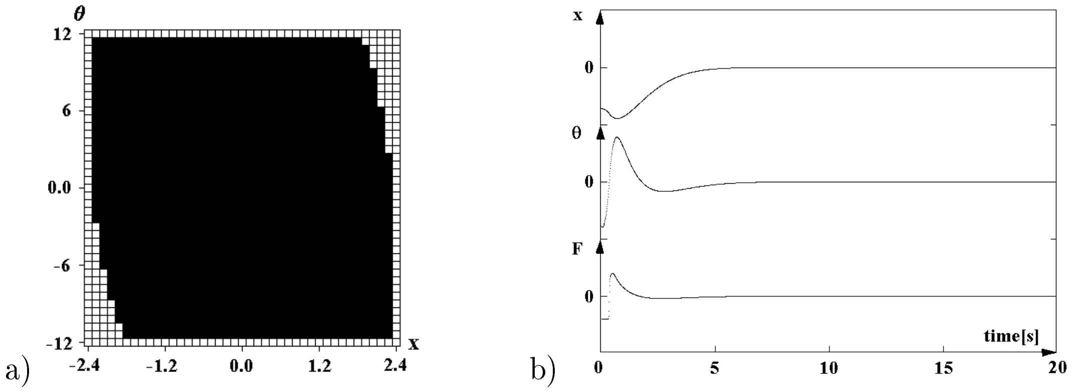


Figure 3: Controller w^2: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -1.9$, $\theta_0 = -0.18$.

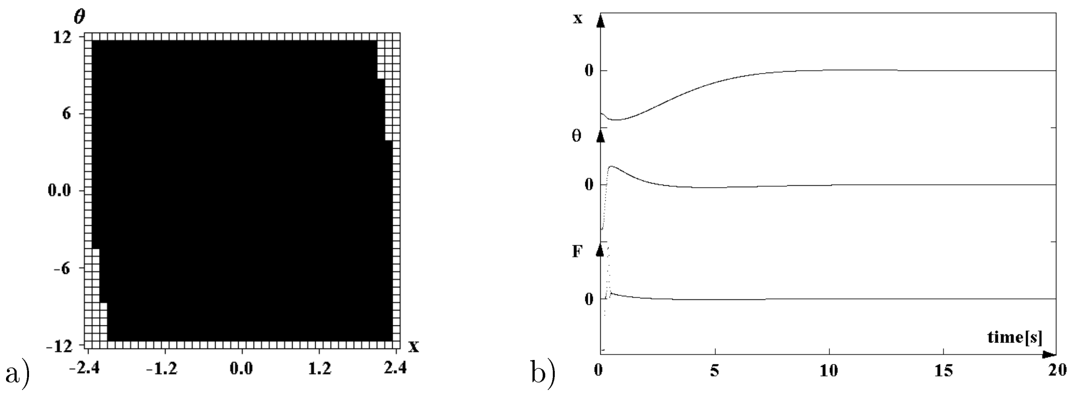


Figure 4: Controller w^3: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

3.2 2-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

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

3.2.1 t-class controllers

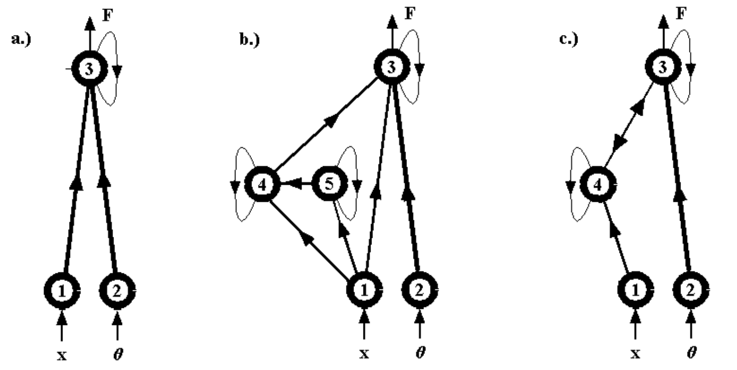


Figure 5: Three examples (w^4, w^5, w^6) of evolved t-class neurocontrollers.

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

$$w^4 = w^4_3 = (\,0.0 \quad 0.15 \quad 4.59 \quad -0.95\,) \ ; \tag{6}$$

$$w^5 = \begin{pmatrix} w^5_3 \\ w^5_4 \\ w^5_5 \end{pmatrix} = \begin{pmatrix} 0.0 & -0.1 & 7.56 & -0.82 & -0.37 & 0.0 \\ 0.0 & -17.31 & 0.0 & 0.0 & 0.75 & -10.93 \\ 0.0 & -0.08 & 0.0 & 0.0 & 0.0 & 0.95 \end{pmatrix} , \tag{7}$$

$$w^6 = \begin{pmatrix} w^6_3 \\ w^6_4 \end{pmatrix} = \begin{pmatrix} 0.0 & 0.0 & 3.16 & 0.32 & -0.27 \\ 0.0 & -9.35 & 0.0 & 10.81 & 3.56 \end{pmatrix} . \tag{8}$$

As can be seen from Figs. 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 (Fig. 5a) given by (6) uses no hidden neuron, but only the output neuron with an inhibitory self-connection. As Fig. 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 Fig. 6b.

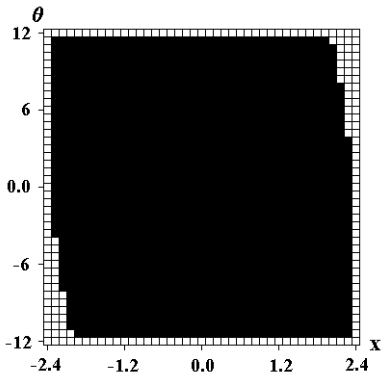
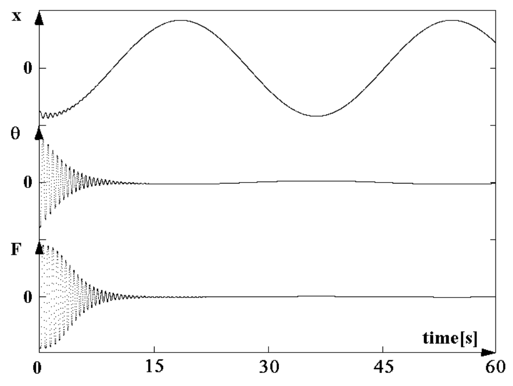
a)  b) 

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 $x_0 = -1.9$, $\theta_0 = -0.18$.

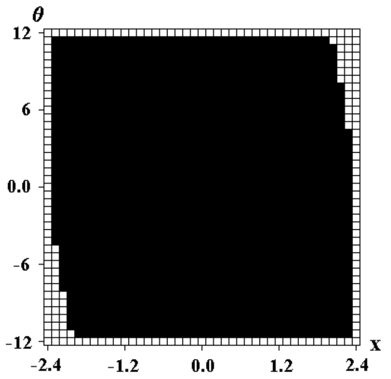
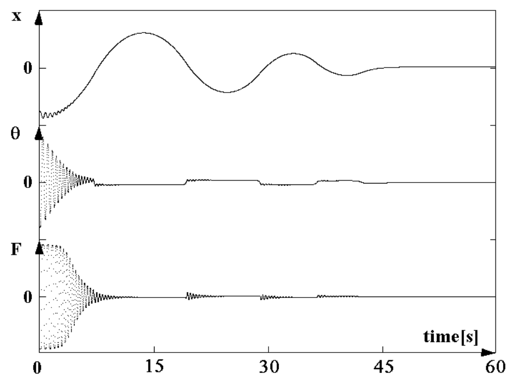
a)  b) 

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 (Fig.5b) given by (7) is interesting not only because it has an optimal performance (compare Figs. 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; Fig. 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 Fig. 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 taht 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

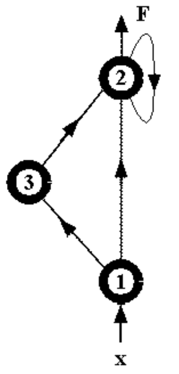


Figure 8: A control module centering the cart using only cart position x as input.

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

$$w^{cart} = \begin{pmatrix} w_2^{cart} \\ w_3^{cart} \end{pmatrix} = \begin{pmatrix} -0.68 & -19.98 & -5.97 & 0.0 \\ -0.12 & -3.15 & 0.0 & 0.0 \end{pmatrix} \quad ; \qquad (9)$$

and is shown in Fig. 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.

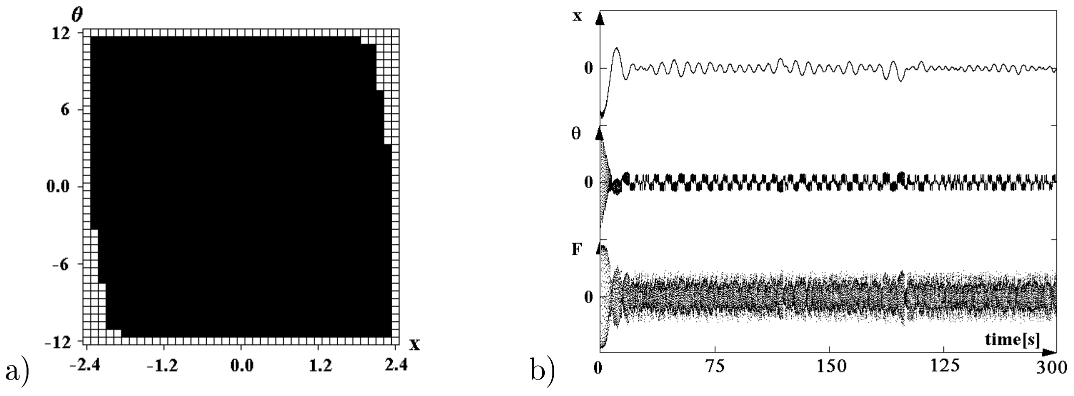


Figure 9: Controller w^6: a.) Benchmark initial conditions (black) for which balancing time is larger than 300[s]. b.) Cart position x, pole angle θ and force F as functions of time, starting at $x_0 = -2.0$, $\theta_0 = -0.18$.

The third solution w^6 (Fig. 5c) given by (8) 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 Fig. 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 Fig. 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 behavior of the controlled systems may result from coupling its eigenmodes back to a quasi-periodic control system.

3.2.2 Three s-class solutions

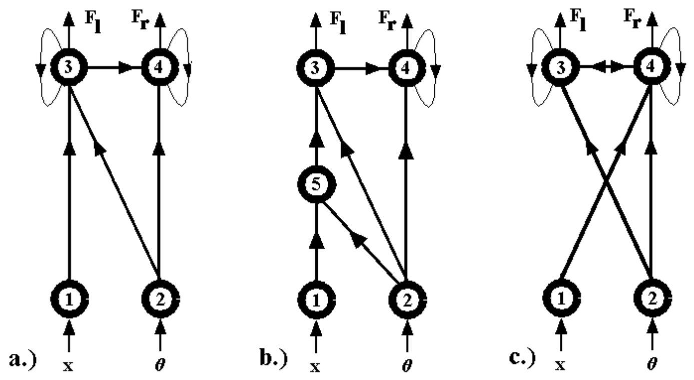


Figure 10: Three examples (w^7, w^8, w^9) of evolved s-class neurocontrollers.

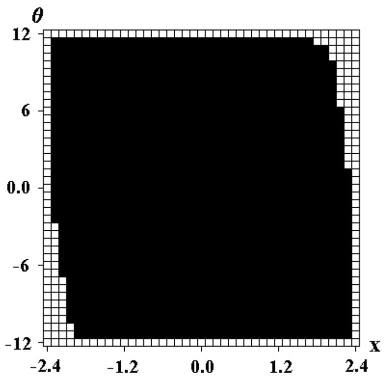
The three neurocontrollers shown in Fig. 10 are examples of evolved "minimal" configurations for s-class controllers. Their configurations are given by the following weight vectors:

$$w^7 = \begin{pmatrix} w_3^7 \\ w_4^7 \end{pmatrix} = \begin{pmatrix} 2.72 & 21.3 & 49.62 & 3.83 & 0.0 \\ -2.95 & 0.0 & -10.55 & 10.5 & -4.14 \end{pmatrix} \quad ; \tag{10}$$

$$w^8 = \begin{pmatrix} w_3^8 \\ w_4^8 \\ w_5^8 \end{pmatrix} = \begin{pmatrix} -11.72 & 0.0 & 30.11 & 0.0 & 0.0 & 25.84 \\ -4.38 & 0.0 & -13.82 & 16.17 & -7.11 & 0.0 \\ -3.74 & 37.69 & 35.11 & 0.0 & 0.0 & 0.0 \end{pmatrix} \quad ; \tag{11}$$

$$w^9 = \begin{pmatrix} w_3^8 \\ w_4^8 \end{pmatrix} = \begin{pmatrix} -2.74 & 0.0 & 23.85 & -9.79 & 14.69 \\ -2.29 & -10.05 & -35.6 & -10.71 & 12.08 \end{pmatrix} \quad . \tag{12}$$

Again they solve the problem for a large domain of benchmark initial conditions (compare Figs. 11a, 12a, and 13a). But none of them bring 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 Figs. 11b, 12b, and 13b).

a) 

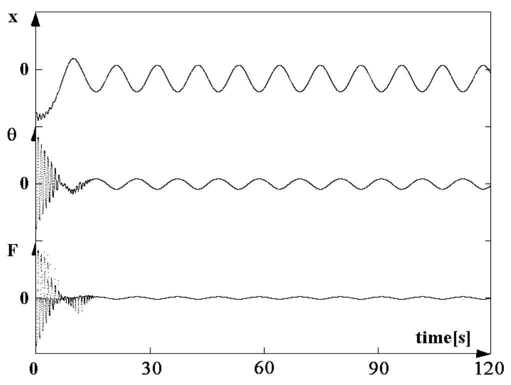
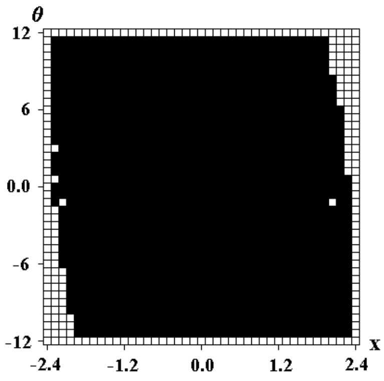
b) 

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) 

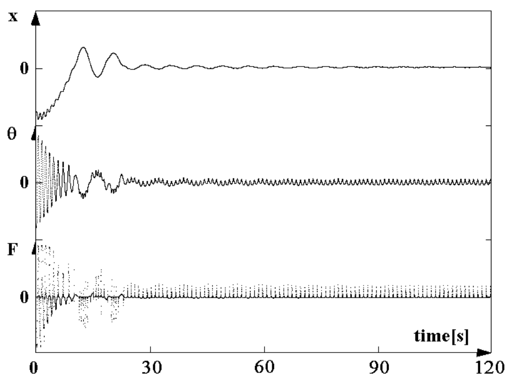
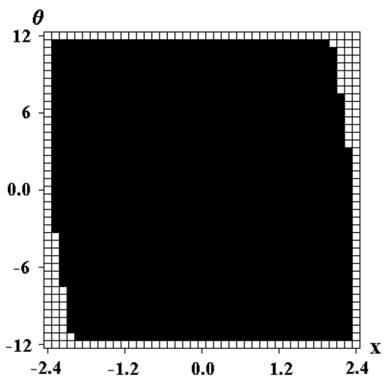
b) 

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

Although the three controllers have almost the same benchmark performance, they use different "techniques" to solve the problem. For instance controller w^7 (10) 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 [13], but this turns out not to be essential here. Self-inhibition with values just above the critical value $w_{44} = -4.0$ damps pole oscillations as well and gives the same benchmark results. Correspondingly, the excitatory self-connection w_{33} of neuron 3 keeps the amplitude of cart oscillations at a constant value; simulations show, that for $w_{33} = 0$ 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 (11) 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 Fig. 12. It furthermore uses delay lines for the pole angle signals. Controller w^9 (12) is the one with the best benchmark performance (see Fig. 13a,b). It uses higher periods and even chaotic dynamics

a) 

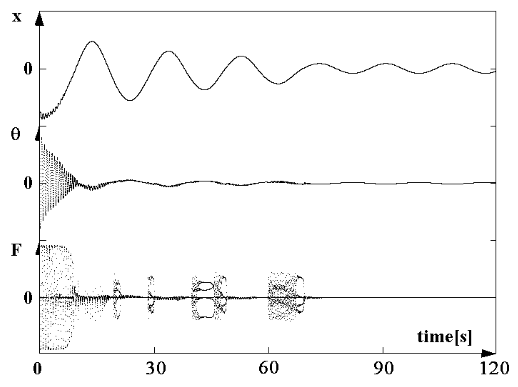
b) 

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

to stabilize the system in a comparatively short time (around 60 seconds for extreme initial conditions, compare Fig. 13b). In fact, output units 3 and 4 form a "chaotic neuromodule" [12], 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 (compare appendix 1 of this paper).

4 Discussion

We have demonstrated that the *ENS^3*-algorithm presented in Sec. 2 can be applied successfully to control problems such as balancing an inverted pendulum. The evolved solutions are remarkably effective, that is, they function with low computational effort when compared with solutions of classical control techniques, and they are small in size with low connectivity when compared with other neural network solutions. Especially for the case where controllers get only reduced phase space information (Sec. 3.2), the recurrent connectivity of the evolved control modules keeps networks small in size and remarkably effective. Since many published results on neurocontrollers for this problem do not present data for cart centering and wall avoiding like Figs. 2-4, 6-8, and 11-13, we can only suspect that for instance the control modules w^5 and w^9 can compete with those solutions.

The results presented here also demonstrate that there is no need for phase space quantization - used in many classical approaches - to handle this control problem. In contrast to solutions generated by genetic algorithms (GA), all network parameters like synaptic weights and bias terms are continuous; this might be an advantage for getting "minimal" solutions without losing effectiveness. 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 (2). If costs for neurons and connections (i.e. $cost_n$ and

$cost_s$ in (2)) 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 minimizing the force applied to the cart ($cost_F$) in (2), solutions tend to use internal oscillators, keeping the pole balanced by permanent oscillations. These oscillators are realized for instance by inhibitory self-connections [13] or loops of two or three neurons [14]. We also observed solutions making use of switched oscillators, which come into action only if the physical system enters critical phase space domains.

In addition, the evolved controllers are quite robust in many respects. A moderate noise on the input signals did not effect their performance noticeably. Varying the discretization time step Δ between 0.01 and 0.03 does not effect the principle functioning of most of the controllers. Although evolved for time steps $\Delta = 0.01$, controller w^5 (7), for example, still operates with good performance for time steps up to $\Delta = 0.05$. Comparing the controllers of this paper with controllers evolved for discretization time steps $\Delta = 0.02$ as in [17] we observe that the same architecture will do the job with slightly different weights: compare for instance controller w^9 of this paper with controller w^6 of [17]. But also controllers with the same architecture but very different weights can solve the problem (compare e.g. w^5 ($\Delta = 0.01$) of this paper with w^2 ($\Delta = 0.02$) of [17]). Likewise, the application of standard friction terms [4] to cart and pole also does not make a large difference for many controllers: evolved for the frictionless case, most of the time they have an even better performance on the benchmark domain. Controllers evolved for the problem with friction make use of the same architectures, sometimes with only slightly different weights (compare e.g. w^1 and w^4 of this paper with controllers given in [16]).

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.

Even though the evolved networks studied in this paper only had to solve a simple problem, we believe that there is nevertheless something to learn for a theory of neural processing. The fact that every 2-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 in neural processing tasks. Even the few evolved neurocontrollers described here show that there is no simple structure-function relationship: one and the same task can be solved by many different kinds of network configurations (architectures as well as parameter settings) using different "techniques" (e.g. delay lines, internal oscillators or no oscillators at all). We also saw that the realized network structures and network functions depend on the "survival conditions", that is, on the fitness function given by (2). With examples from sections 3.2.1 and 3.2.2 we can also deduce that different kinds of neurons will give rise to specific architectures.

Moreover, the evolved controller w^5, which has the best benchmark performance of examples presented in this paper, 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 cooperate or compete to generate a desired behavior. This is also in the spirit of evolutionary robotics, where one has to generate "brains" or nervous systems for robots, which have to operate on different noise sensor inputs and have to coordinate different motor actions to behave successfully in an interesting environment (compare e.g. [11], [10]).

There is one more observation one can make: there are controllers with a "genuine" internal dynamics, i.e. one which is immanent in the structure (for example oscillations, as for w^8), which here is functional in the sense that keeping the cart in fast oscillations will balance the pole. Controller w^9 demonstrates that even chaotic dynamics may be used to stabilize the system. In other controllers an internal dynamics is observed (e.g. w^6) that is not caused by the network structure and that seems to be induced by the back-coupling of "motor" actions to the "sensors" in the sensori-motor loop. In some controllers also an internal dynamics can be observed that is coherent with the external dynamics: for instance neuron activities "representing" the (oscillatory) cart position or pole angle.

Because the ENS^3-algorithm was originally designed to study theoretical aspects of modularized recurrent networks, we were not concerned about statistics on computation time. Perhaps for classical problems like pattern classification, evolution will not outperform algorithms like backpropagation. This became clear, for instance, when solving the parity-n problems with the evolutionary algorithm as reported in [8]. The advantage of the ENS^3-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.

Besides the first results presented here, further simulations on more difficult problems, such as for instance balancing a rotating pendulum or a ball on a beam, indicate that the ENS^3-algorithm may already be successfully applied to many challenging control applications. But the ENS^3-algorithm can be optimized further: for instance the evaluation of an individual network, given by the operator **E** in the variation-evaluation-selection cycle, may be replaced by an evaluation-learning cycle, if an appropriate learning procedure is at hand.

Acknowledgment

I am grateful to Ulf Dieckmann for many inspiring discussions.

References

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

[2] Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, **9**, 31 - 37.

[3] Anderson, C. W. and Miller W. T. (1990). Challenging Control Problems. In W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*, 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 - 846.

[5] Branke, J. (1995). Evolutionary algorithms for neural network design and training. In *Proceedings 1st Nordic Workshop on Genetic Algorithms and its Applications*, Vaasa, Finland.

[6] 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 - 44.

[7] Dasgupta, D., and McGregor, D. R. (1993). Evolving neurocontrollers for pole balancing. In S. Gielen and B. Kappen (eds.), *ICANN'93 Proceedings of the International Conference on Artificial Neural Networks*, Amsterdam 13.-16. Sept. 1993. Berlin: Springer-Verlag, 1993, pp. 834-837.

[8] Dieckmann, U. (1995), Coevolution as an autonomous learning strategy for neuromodules, in: Herrmann, H., Pöppel, E., and Wolf, D. (eds.), *Supercomputing in Brain Research - From Tomography to Neural Networks*, Singapore: World Scientific, (pp. 331–347).

[9] Geva, S., and Sitte, J. (1993), A cartpole experiment benchmark for trainable controllers, *IEEE Control Systems Magazin*, **13**, 40-51.

[10] Husbands, P., Harvey, I., Cliff, D., and Miller, G. (1997), Artificial evolution: A new path for artificial intelligence?, *Brain and Cognition*, **34**, pp. 130-159.

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

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

[13] Pasemann, F. (1993), Dynamics of a single model neuron, International Journal of Bifurcation and Chaos, 2, 271-278.

[14] Pasemann, F. (1995), Characteristics of periodic attractors in neural ring networks, Neural Networks, 8, 421-429.

[15] Pasemann, F. (1995), Neuromodules: A dynamical systems approach to brain modelling, in: Herrmann, H. J., Wolf, D. E. and Pppel, E. (Eds.), Supercomputing in Brain Research: From Tomography to Neural Networks, World Scientific, Singapore, pp. 331-348.

[16] Pasemann, F., and Dieckmann, U. (1997). Evolved Neurocontrollers for pole-balancing. In: Proceedings IWANN'97, Lanzarote, Spain, June 4-6, 1997, Lecture Notes in Computer Science. Berlin: Springer-Verlag.

[17] Pasemann, F. (1997), Pole-balancing with different evolved neurocontrollers, in: Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D. (eds.), *Artificial Neural Networks - ICANN'97*, October 7-10, Lausanne, Switzerland, Proceedings, LNCS 1327, Springer-Verlag, pp. 823 - 829.

[18] 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. In: *Proceedings International Workshop on combinations of genetic algorithms and neural networks (COGANN-92)*, Los Alamitos, CA, IEEE Computer Society Press.

[19] Selfridge, O. G., Sutton, R. S., and Barto, A. G. (1985). Training and tracking in robotics. In *Proceedings International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA, pp: 670 - 672.

[20] Widrow, B. (1987), The original adaptive neural net broom-balancer. *Proc. IEEE Intern. Symp. Circuits and Systems*, 351 - 357.

[21] Wieland, A. P. (1991). Evolving neural network controllers for unstable systems. In: *International Joint Conference on Neural Networks*, Seattle, USA, July1991. Proccedings Vol.**2**. Seattle: IEEE Service Center, 1991.

[22] Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, **8**, 539 - 567.

Appendix 1: Dynamics of controller w^9

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.

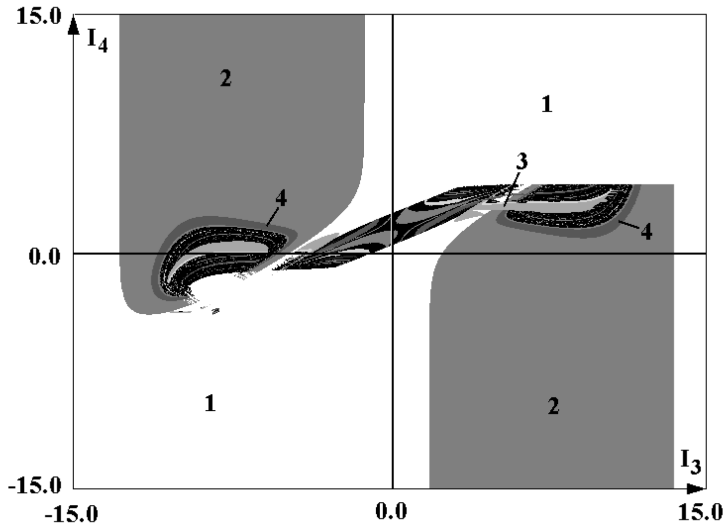


Figure 14: Iso-periodic plot for the output module of controller w^9. Fixed point attractors are coded white, others as shown (see text).

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 $\bar{o}$ denotes the averaged module output, i.e. $\bar{o} = 1/2 \cdot (o_3 + o_4)$. 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

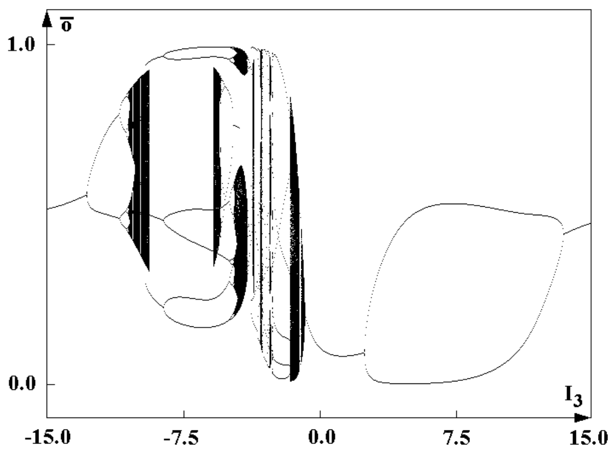


Figure 15: A bifurcation sequence for I_3 with $I_4 = 0.0$ fixed.

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.

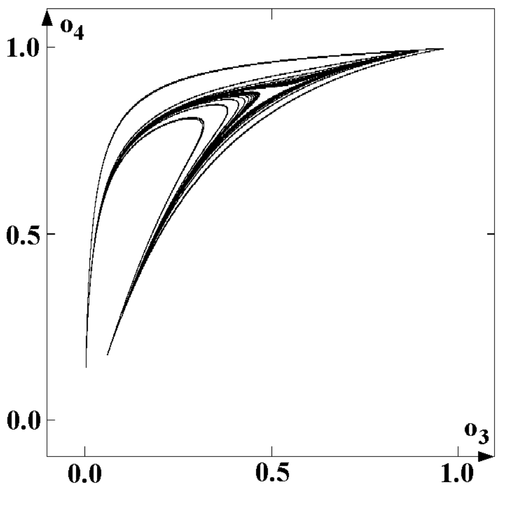


Figure 16: A chaotic attractor for $(I_3, I_4) = (-5.2, -1.0)$.

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: Characterizing controllers by 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).

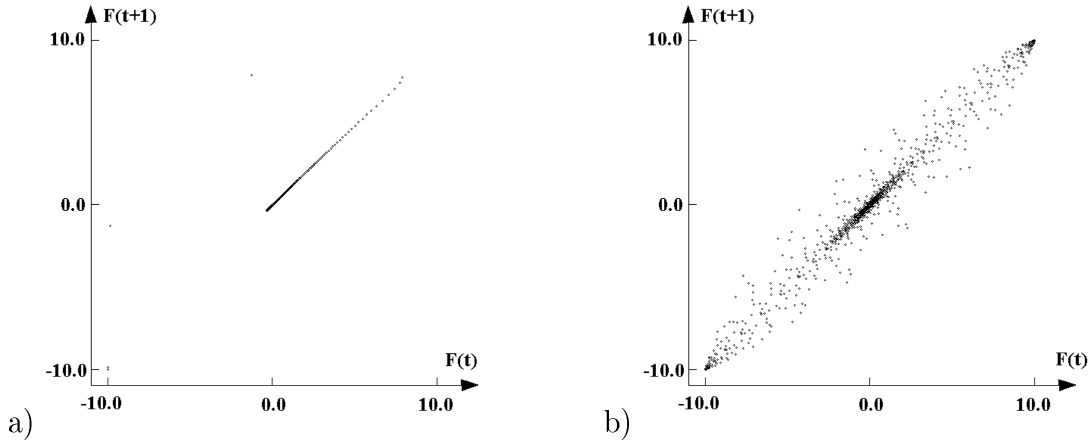


Figure 17: First return map for the force signal of controller a.) w^1 and b.) w^5.

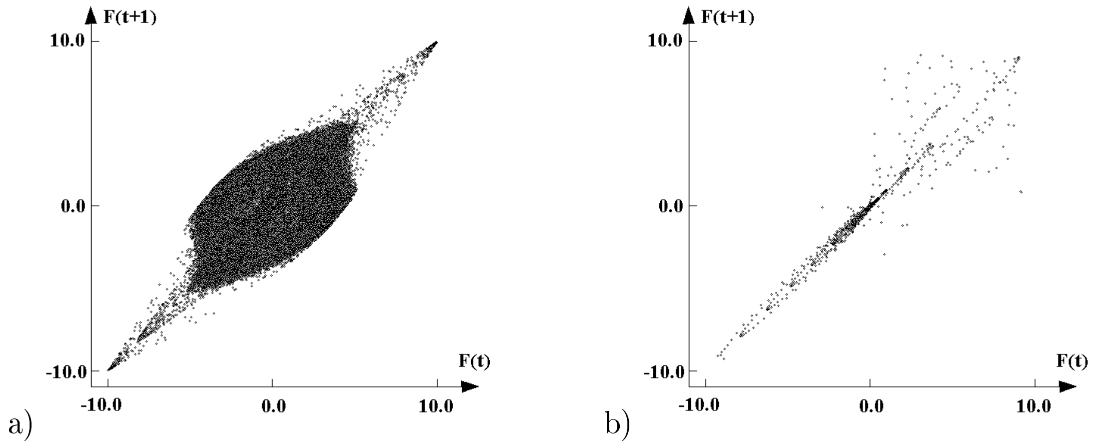


Figure 18: First return map for the force signal of controller a.) w^6 and b.) w^7.

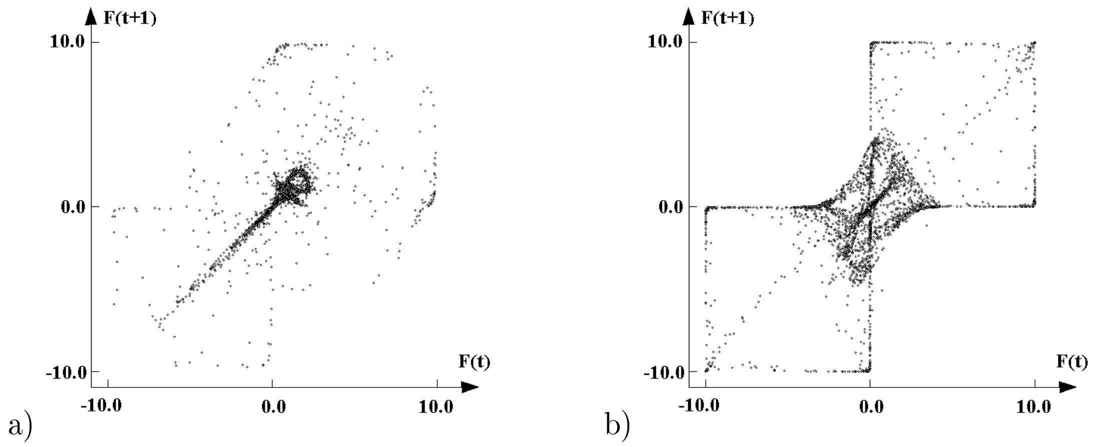


Figure 19: First return map for the force signal of controller a.) w^8 and b.) w^9.