Neural Control Theory: an Overview*

J.A.K. Suykens, **H. Bersini*****

** Katholieke Universiteit Leuven
Department of Electrical Engineering, ESAT-SISTA
Kardinaal Mercierlaan 94, B-3001 Leuven (Heverlee), Belgium
Tel: 32/16/32 18 02 Fax: 32/16/32 19 86
E-mail: johan.suykens@esat.kuleuven.ac.be

*** Universite Libre de Bruxelles
50, av. Franklin Roosevelt, B-1050 Bruxelles, Belgium
Tel: (322)-650-27-33 Fax: (322)-650-27-15
E-mail: bersini@ulb.ac.be

Abstract

In this paper we present a short introduction to the theory of neural control. Universal approximation, on- and off-line learning ability and parallelism of neural networks are the main motivation for their application to modelling and control problems. This has led to several existing neural control strategies. An overview of methods is presented, with emphasis on the foundations of neural optimal control, stability theory and nonlinear system identification using neural networks for modelbased neural control design.

Keywords. Multilayer perceptron, radial basis function, feedforward and recurrent networks, static and dynamic backpropagation, nonlinear system identification, neural optimal control, reinforcement learning, NL_q stability theory.

*Part of the research work was carried out at the ESAT laboratory and the Interdisciplinary Center of Neural Networks ICNN of the Katholieke Universiteit Leuven, in the framework of the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture (IUAP-17) and in the framework of a Concerted Action Project MIPS (Modelbased Information Processing Systems) of the Flemish Community.

1 Introduction

The recent success of multilayer perceptron and radial basis function neural networks for modelling and control applications is basically thanks to their universal approximation ability, on- and off-line learning ability and parallelism. It has been mathematically proven that any continuous function can be approximated arbitrarily well over a compact interval by a multilayer neural network with one or more hidden layer (e.g. [9],[12]). Also radial basis function neural networks have this universal approximation property [18]. Hence parametrizing models and controllers by neural network architectures leads to general nonlinear models and control laws. Moreover, multilayer perceptrons can avoid the curse of dimensionality [1], which means they are very useful in order to realize nonlinear mappings for systems with many inputs and outputs, in contrast e.g. to polynomial expansions. Learning rules such as the backpropagation algorithm [21] exist, which can be used off-line as well as on-line, which is interesting for adaptive control applications. In addition, neural networks possess a massively parallel architecture, which is attractive from the viewpoint of implementation.

This variety of interesting properties led to an emergence of many neural control methods in recent years [13] [31]. Both modelbased and direct neural control strategies exist. This distinction is reflected in the early history of neural control. Barto's broomstick balancing problem was a first example on reinforcement learning, which is a direct method for controlling the plant, by experimenting through a number of trials without using a model for the plant [2]. A first illustration of a modelbased neural control strategy is Nguyen & Widrow's backing up of a trailer-truck. In order to be able to apply backpropagation for training the neural controller, a neural network model is trained first in order to emulate the trailer-truck dynamics [16]. Among the many neural control strategies existing now are direct and indirect neural adaptive control, neural optimal control, reinforcement learning, model predictive control, internal model control, feedback linearization etc. Moreover a stability theory for multilayer recurrent neural networks (NL_q theory) has been developed recently, which can be used for modelbased neural control design [31].

This paper is organized as follows. In Section 2 we present a list of neural control strategies. In Section 3 we discuss nonlinear system identification using neural networks with respect to modelbased neural control design. In Section 4 neural optimal control is explained. Finally, in Section 5 NL_q stability theory for neural control design is presented.

2 Overview of neural control strategies

Here we discuss some basic features of existing neural control strategies [13] [11] [31] :

- *Neural adaptive control:*
  In indirect adaptive control first a neural network model is derived on-line from I/O measurements on the plant. The neural controller is trained then e.g. for tracking of specific reference inputs. This is done by defining a cost function based on the model and the controller. In direct adaptive control the cost function is defined with respect to the real plant, instead of on the basis of a model. The controller is adapted in time.

- *Neural optimal control:*
  In this method the neural controller is designed based on a deterministic nonlinear model, which might be either a black box neural network model or a model resulting

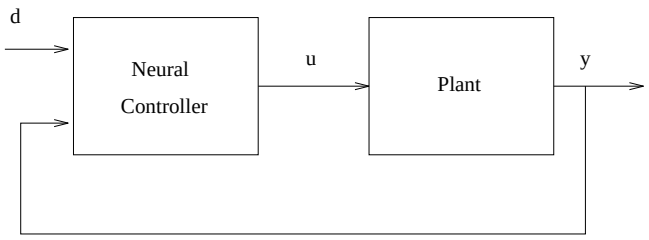

Figure 1: *Neural control scheme with input u and output y of a plant and reference input d to be tracked.*

from physical laws. The control law results from Pontryagin's maximum principle or Lagrange multiplier methods, leading to costate equations and two-point boundary value problems. Suboptimal solutions are obtained by imposing a neural control law that is independent of the costate or by formulating parametric nonlinear optimization problems in the unknown neural controller.

- *Reinforcement learning:*
  In this method situations are mapped to actions so as to maximize a scalar reward (reinforcement signal). In the adaptive critic methods the reinforcement learning controller is rewarded or punished by a critic element through a number of trials on the systems. The plant is directly controlled without having a model for the plant. The Q-learning method can be considered as dynamic programming without having a model.

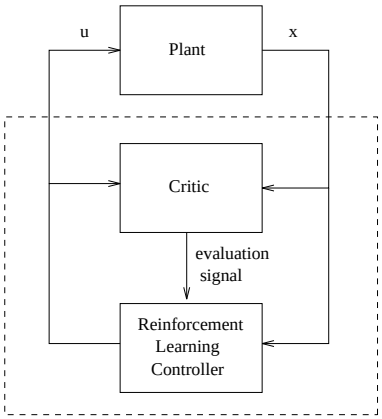

Figure 2: *Reinforcement learning control scheme. The plant is directly controlled without having a model for the plant.*

- *Model predictive control and internal model control:*
  These are modelbased control strategies, making explicit use of a neural network model in the control scheme. In internal model control the difference between the output of the plant with the output of the model is fed back. In model predictive control (Fig.3) the control signal u of the controller C is determined from predictions of the neural network model M for the plant P over a given time horizon.

- *NL_q stability theory:*
  In this modelbased framework, neural network models and neural controllers are taken

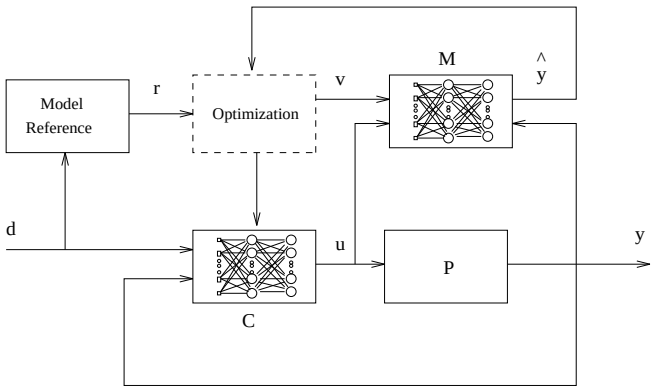

Figure 3: *Model predictive control scheme. The neural network model provides predictions for the output of the plant model over a specified time horizon.*

in state space form. Process and measurement noise can be taken into account. The theory allows to design the controllers based upon closed-loop stability criteria. Specific reference inputs can be tracked under guaranteed closed-loop stability or the neural controller can be designed without training on specific reference inputs.

The method to be selected depends on the specific system and the specific control task. Important criteria are e.g. stabilization, optimal tracking, internal stability, closed-loop stability, adaptivity, disturbance rejection, availability of a model, local control at one single working point or global, robustness, certain or uncertain environment etc.

3 Nonlinear system identification using neural networks

In this Section we discuss some basic aspects of nonlinear system identification using multilayer perceptrons and radial basis function networks, with respect to modelbased neural control strategies where the control law is based upon the neural network model.

A multilayer perceptron with one hidden layer or multilayer feedforward neural network is described as [34]

$$y = W\sigma(Vu + \beta) \tag{1}$$

Here $u \in \mathbb{R}^s$ is the input vector and $y \in \mathbb{R}^r$ the output vector of the network and the nonlinear operation $\sigma(.)$ is taken elementwise. The interconnection matrices are $W \in \mathbb{R}^{r \times n_h}$ for the output layer, $V \in \mathbb{R}^{n_h \times s}$ for the hidden layer, $\beta \in \mathbb{R}^{n_h}$ is the bias vector (thresholds of hidden neurons) with n_h the number of hidden neurons. For $\sigma(.)$ a saturation-like characteristic is taken such as tanh(.).

Given a training set of input/output data, the original learning rule for multilayer perceptrons is the backpropagation algorithm [21]. Let us denote the neural network with L layers (i.e. $L-1$ hidden layers) as

$$z_{i,p}^l = \sigma(\xi_{i,p}^l), \quad \xi_{i,p}^l = \sum_j w_{ij}^l z_{j,p}^{l-1}, \quad l = 1, ..., L \tag{2}$$

where l is the layer index ($l = 1$ corresponds to the input layer and $l = L$ to the output layer), p the pattern index for the training data and $\sigma(.)$ the activation function. Given are

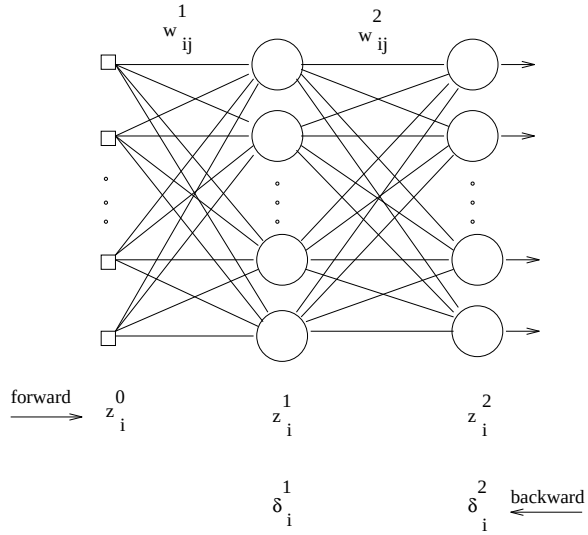

Figure 4: *Backpropagation algorithm for a multilayer perceptron.*

a number of P input patterns ($p = 1, ..., P$) and corresponding desired output patterns. w_{ij}^l denotes the ij-th entry of the interconnection matrix of layer l. Usually, the objective to be minimized is the cost function

$$\min_{w_{ij}^l} E = \frac{1}{P}\sum_{p=1}^{P} E_p \qquad E_p = \frac{1}{2}\sum_{i=1}^{N_L}(z_{i,p}^d - z_{i,p}^L)^2 \tag{3}$$

where z_p^d is the desired output vector corresponding to the p-th input pattern and z_p^L is the actual output of the neural network for the p-th input pattern. E_p is the contribution of the p-th pattern to the cost function E and N_l denotes the number of neurons at layer l. The backpropagation algorithm or generalized delta rule is given by

$$\begin{cases} \Delta w_{ij}^l &= \eta\, \delta_{i,p}^l\, z_{j,p}^{l-1} \\ \delta_{i,p}^L &= (z_{i,p}^d - z_{i,p}^L)\, \sigma'(\xi_{i,p}^L) \\ \delta_{i,p}^l &= (\sum_{r=1}^{N_{l+1}} \delta_{r,p}^{l+1} w_{ri}^{l+1})\, \sigma'(\xi_{i,p}^l), \quad l = 1, ..., L-1 \end{cases} \tag{4}$$

where η is the learning rate and the so-called δ variables are defined as $\delta_{i,p}^l = \frac{\partial E_p}{\partial \xi_{i,p}^l}$. The term backpropagation refers to way in which the adaptation of the weights is calculated: first the outputs at the several layers are computed in a forward step starting from the input layer toward the output layer and then the δ variables are computed starting from the output layer toward the input layer by backpropagating the error between the desired and actual output of the multilayer perceptron. In fact, the algorithm (4) is nothing else but a steepest descent local optimization algorithm for minimizing the cost function. Often an adaptive learning rate and a momentum term is used or more efficient local optimization methods such as quasi-Newton, Levenberg-Marquardt or conjugate gradient algorithms are applied [32] [8].

Given I/O measurements on a plant, nonlinear system identification is done then by using e.g. the following model structures [6] [27] [30] [31] :

- *NARX model:*

$$y_k = f(y_{k-1}, ..., y_{k-n_y}, u_{k-1}, ..., u_{k-n_u}; \theta) + \epsilon_k \tag{5}$$

NARMAX model:

$$y_k = f(y_{k-1}, ..., y_{k-n_y}, u_{k-1}, ..., u_{k-n_u}, \epsilon_{k-1}, ..., \epsilon_{k-n_e}; \theta) + \epsilon_k \tag{6}$$

with output vector $y_k \in \mathbb{R}^r$, input vector $u_k \in \mathbb{R}^s$ and n_y, n_u, n_e are the lags for the output, input and noise signal respectively. The nonlinear mapping $f(.)$ is parametrized then by a multilayer perceptron with parameter vector θ, containing the weights w_{ij}^l of (2).

- *Neural state space model:*

$$\left\{ \begin{array}{rcl} \hat{x}_{k+1} & = & f(\hat{x}_k, u_k; \theta) + K\epsilon_k \\ y_k & = & g(\hat{x}_k, u_k; \theta) + \epsilon_k \end{array} \right. \tag{7}$$

with state vector $x_k \in \mathbb{R}^n$, Kalman gain K and prediction error $\epsilon_k = y_k - \hat{y}_k(\theta)$ and two multilayer perceptrons $f(.)$ and $g(.)$. For deterministic identification one has $K = 0$.

Identification can be done by using a prediction error algorithm

$$\min_\theta J(\theta) = \frac{1}{2N} \sum_{k=0}^{N-1} \epsilon_k^T \epsilon_k \tag{8}$$

where θ denotes the unknown parameter vector of the neural network and N the number of data points. When the resulting neural network model is feedforward the backpropagation algorithm (4) is directly applicable. In the case of neural state space models (7) the network is however recurrent, which means it is a dynamical system. Computing the gradient of the cost function is more difficult then: Narendra's dynamic backpropagation can be applied which involves using a sensitivity model, which is in itself a dynamical model, that generates the gradient of the cost function [15]. The problem (8) is a nonlinear optimization problem with many local optima. Hence one has to try several random starting points or one may use results from linear identification as starting point [30]. Another important issue is the choice of the number of hidden neurons of the neural network. Enough neurons should be taken in order to model the underlying nonlinear dynamics, but taking too many might lead to overfitting if one keeps on optimizing on the training data for too many iterations. One has to stop training when the minimal error on an independent test set of data is obtained [26] [27]. Finally, model validation has to be done, including e.g. higher order correlation tests [6].

Another commonly used neural network architecture is the radial basis function (RBF) network, which makes use of Gaussian instead of saturation-like activation functions. RBFs are also a universal approximator [18]. In the single output case the RBF network can be described as [19]

$$y = \sum_{i=1}^{n_h} w_i \, \phi(\|x - c_i\|_2), \tag{9}$$

with $x \in \mathbb{R}^s$ the input vector and $y \in \mathbb{R}$ the output, w_i are the output weights, $c_i \in \mathbb{R}^s$ the centers and $\phi(.)$ the Gaussian activation function. Parametrizing e.g. the NARX model by

means of the RBF network is done by taking $y = y_k$ and $x = [y_{k-1}, ..., y_{k-n_y}, u_{k-1}, ..., u_{k-n_u}]$. A nice feature of RBF networks is that one can separate the training of the output weights and the centers. Taking as many hidden neurons as data points, the centers are placed at the data points. RBF networks possess then a best representation property in the sense of a regularized approximation problem [19]. However, for system identification purposes fewer parameters are required for good generalization ability. A clustering algorithm is used then to place the centers c_i at the center of n_h clusters of data points. The determination of the output weights is then a linear least squares problem [7]. The use of RBF networks for use in direct adaptive neural control is described e.g. in [24]. Also in [3], the strong resemblance of RBF with one type of fuzzy model (called Takagi-Sugeno type), commonly used for control and process identification, has been discussed. The similarities between both models allow to analyze neurocontrol and fuzzy control on common basis.

4 Neural optimal control

In this Section we discuss the N-stage optimal control problem and tracking problem in neural optimal control, together with dynamic backpropagation, backpropagation through time and Q-learning.

4.1 N-stage optimal control problem

The N-stage optimal control problem from classical optimal control theory can be stated as follows. Given a nonlinear system

$$x_{k+1} = f_k(x_k, u_k), \qquad x_0 \text{ given} \tag{10}$$

for $k = 0, 1, ..., N-1$ with state vector $x_k \in \mathbb{R}^n$ and input vector $u_k \in \mathbb{R}^m$, consider a performance index of the form

$$J = \psi(x_N) + \sum_{k=0}^{N-1} l_k(x_k, u_k) \tag{11}$$

with $l_k(.)$ a positive real valued function and $\psi(.)$ a real valued function function, specified on the final state x_N. In our case the model (10) might result from physical laws or from black-box nonlinear system identification using neural nets. The problem is then to find the sequence u_k that minimizes (or maximizes) J. Using a Lagrange multiplier technique one obtains as solution a two point boundary value problem with the state equation running forward in time for a given initial state and a costate equation running backward in time with given final costate. In general this results into a control law which depends on the state and the costate. In [23] a suboptimal solution is investigated for the full static state feedback control law

$$u_k = g(x_k; \theta), \tag{12}$$

where $g(.;\theta)$ represents a multilayer perceptron with parameter vector θ, containing the weights w_{ij}^l of (2). Hence for neural optimal control one has to optimize the performance index (11) subject to the system dynamics (10) and the control law (12). Introducing the multiplier sequences $\{\lambda_k\}$ and $\{\mu_k\}$, this leads to the Lagrangian:

$$\mathcal{L} = \psi(x_N) + \sum_{k=0}^{N-1} l_k(x_k, u_k) + \sum_{k=0}^{N-1} \lambda_{k+1}^T (f_k(x_k, u_k) - x_{k+1}) + \sum_{k=0}^{N-1} \mu_k^T (g(x_k, \theta) - u_k) \tag{13}$$

The conditions for optimality are given by

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x_k} &= \frac{\partial l_k}{\partial x_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial x_k} - \lambda_k^T + \mu_k^T \frac{\partial g}{\partial x_k} = 0 \\ \frac{\partial \mathcal{L}}{\partial x_N} &= \frac{\partial \psi}{\partial x_N} - \lambda_N = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda_{k+1}} &= f_k(x_k, u_k) - x_{k+1} = 0 \\ \frac{\partial \mathcal{L}}{\partial u_k} &= \frac{\partial l_k}{\partial u_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k} - \mu_k^T = 0 \\ \frac{\partial \mathcal{L}}{\partial \mu_k} &= g(x_k, \theta) - u_k = 0. \end{cases} \tag{14}$$

On the other hand minimizing the performance index (13) with respect to θ by means the generalized delta rule yields:

$$\begin{cases} \Delta w_{ij}^l &= \eta\, \delta_{i,k}^l\, z_{j,k}^{l-1} \\ \delta_{i,k}^L &= \frac{\partial \mathcal{L}_k}{\partial z_{i,k}^L}\, \sigma'(\xi_{i,k}^L) \\ &= \mu_{i,k}\ \sigma'(\xi_{i,k}^L) \\ \delta_{i,k}^l &= (\sum_{r=1}^{N_{l+1}} \delta_{r,k}^{l+1} w_{ri}^{l+1})\, \sigma'(\xi_{i,k}^l), \quad l = 1, ..., L-1. \end{cases} \tag{15}$$

Based on (14) and (15), the neural controller can be trained as follows:

- Generate random interconnection weights w_{ij}^l for the neural controller.
- Do until convergence:
  1. *Forward pass:*
     Compute the sequences $\{u_k\}_{k=0}^N$ and $\{x_k\}_{k=0}^N$ from $x_{k+1} = f_k(x_k, u_k)$ and $u_k = g(x_k; \theta)$.
  2. *Backward pass:*
     (a) Compute backward in time $(k = N-1, ..., 0)$

$$\begin{cases} \lambda_N &= \frac{\partial \psi}{\partial x_N} \\ \mu_k^T &= \frac{\partial l_k}{\partial u_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k} \\ \lambda_k^T &= \frac{\partial l_k}{\partial x_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial x_k} + \mu_k^T \frac{\partial g}{\partial x_k} \end{cases}$$

     in order to obtain the sequence $\{\mu_k\}_{k=0}^{N-1}$.
     (b) Apply the generalized delta rule (15) for adapting the weights of the neural controller.

  End

A stability analysis of this control scheme can be made under certain simplifications [23].

4.2 Tracking problem

The tracking problem has been investigated in [17]. The performance index to be considered is

$$J = \sum_{k=0}^{N-1} [h_k(x_k, u_k) + \rho_{k+1}(\|r_{k+1} - x_{k+1}\|_2)] \tag{16}$$

which is in general nonquadratic with $h_k(.)$ and $\rho_{k+1}(.)$ positive nonlinear functions and $r_k \in \mathbb{R}^n$ is a reference state vector. A suboptimal solution has been proposed according to the so-called linear structure preserving principle, which assumes that the optimal control strategy takes the same form as for a linear quadratic (LQ) problem:

$$\begin{cases} u_k &= \gamma(x_k, v_k) & k = 0, 1, ..., N-1 \\ v_k &= \phi(v_{k+1}, r_{k+1}) & k = 0, 1, ..., N-2 \\ v_{N-1} &= \varphi(r_N) \end{cases} \tag{17}$$

but where the nonlinear mappings $\gamma(.)$, $\phi(.)$ and $\varphi(.)$ are parametrized now by multilayer perceptrons, instead of linear mappings.

4.3 Dynamic backpropagation and backpropagation through time

Assuming the nonlinear plant model is described by a nonlinear state space model

$$\begin{cases} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k) \end{cases} \tag{18}$$

and considering the nonlinear dynamic output feedback law

$$\begin{cases} z_{k+1} &= h(z_k, y_k, d_k) \\ u_k &= s(z_k, y_k, d_k) \end{cases} \tag{19}$$

with $x_k \in \mathbb{R}^n$, $z_k \in \mathbb{R}^{n_z}$ the state of the model and the controller respectively and $u_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}^l$ the input and output of the model respectively, and $d_k \in \mathbb{R}^l$ the reference input. In the case of neural control at least one of the mappings $f(.)$, $g(.)$, $h(.)$ and $s(.)$ is parametrized by a feedforward neural network. The equations for the closed-loop system are of the form:

$$\begin{cases} p_{k+1} &= \zeta(p_k, d_k; \alpha) \\ y_k &= \chi(p_k, d_k; \beta) \\ u_k &= \vartheta(p_k, d_k; \gamma). \end{cases} \tag{20}$$

Suppose the neural controller is parametrized by the parameter vector $\theta \in \mathbb{R}^{p_c}$, which contains the elements α, β and γ that correspond to parametrizations for $\zeta(.)$, $\chi(.)$ and $\vartheta(.)$ respectively. Let us consider now the tracking problem for a specific reference input d_k:

$$\min_\theta J(\theta) = \frac{1}{2N} \sum_{k=1}^{N} \{[d_k - y_k(\theta)]^T [d_k - y_k(\theta)] + \lambda . u_k(\theta)^T u_k(\theta)\} \tag{21}$$

with λ a positive constant. A gradient based optimization scheme makes use then of the gradient

$$\frac{\partial J}{\partial \theta} = \frac{1}{N} \sum_{k=1}^{N} \{[d_k - y_k(\theta)]^T (-\frac{\partial y_k}{\partial \theta}) + \lambda . u_k(\theta)^T \frac{\partial u_k}{\partial \theta}\} \tag{22}$$

where $\frac{\partial y_k}{\partial \theta}$ and $\frac{\partial u_k}{\partial \theta}$ are the output of the sensitivity model

$$\begin{cases} \frac{\partial p_{k+1}}{\partial \alpha} &= \frac{\partial \zeta}{\partial p_k}.\frac{\partial p_k}{\partial \alpha} + \frac{\partial \zeta}{\partial \alpha} \\ \frac{\partial \hat{y}_k}{\partial \alpha} &= \frac{\partial \chi}{\partial p_k}.\frac{\partial p_k}{\partial \alpha} \\ \frac{\partial y_k}{\partial \beta} &= \frac{\partial \chi}{\partial \beta} \\ \frac{\partial u_k}{\partial \gamma} &= \frac{\partial \vartheta}{\partial \gamma}, \end{cases} \tag{23}$$

which is a dynamical system with state vector $\frac{\partial p_k}{\partial \alpha}$, driven by the vectors $\frac{\partial \zeta}{\partial \alpha}$, $\frac{\partial \chi}{\partial \beta}$ and $\frac{\partial \vartheta}{\partial \gamma}$. The Jacobian matrices are $\frac{\partial \zeta}{\partial p_k}$ and $\frac{\partial \chi}{\partial p_k}$. Dynamic backpropagation as defined by Narendra & Parthasarathy in [14] [15] is the steepest descent algorithm that minimizes (21) and uses the sensitivity model (23). The cost function (21) corresponds to the off-line (batch) version of the algorithm. The on-line algorithm for indirect adaptive control works basically the same, but a shorter time horizon is taken for the cost function. In [23], it is shown how dynamic backpropagation can be considerably simplified by performing a truncation in time. This simplification is further generalized to the regulation of processes with arbitrary time delay. Stability analysis of this simplified algorithm has been done based on Lyapunov stability theory. In [29] methods for incorporating linear controller design results are discussed, such that e.g. a transition between working points can be realized with guaranteed local stability at the target point.

Another formalism for calculating the gradients has been proposed by Werbos [33]. This is done by considering an ordered set of equations and ordered partial derivatives. Let the set of variables $\{z_i\}$ $(i = 1, .., n)$ describe the variables and unknown parameters of a static neural network or a recurrent neural network through time. Then the following ordered set of equations is obtained:

$$\begin{cases} z_2 &= f_1(z_1) \\ z_3 &= f_2(z_1, z_2) \\ z_4 &= f_3(z_1, z_2, z_3) \\ \vdots &= \vdots \\ z_n &= f_{n-1}(z_1, z_2, z_3, ..., z_{n-1}) \\ E &= f_n(z_1, z_2, z_3, ..., z_{n-1}, z_n). \end{cases} \tag{24}$$

In the last equation of the ordered set the cost function E is expressed as a function of the variables and the parameters of the network. An ordered partial derivative is then defined as

$$\frac{\partial^+ z_j}{\partial z_i} = \frac{\partial z_j}{\partial z_i} \mid_{\{z_1,...,z_{i-1}\}\text{held constant}} \tag{25}$$

The following chain rules for the ordered derivatives hold then

$$\begin{cases} \frac{\partial^+ E}{\partial z_i} &= \frac{\partial E}{\partial z_i} + \sum_{k>i} \frac{\partial^+ E}{\partial z_k} \frac{\partial z_k}{\partial z_i} \\ \frac{\partial^+ E}{\partial z_i} &= \frac{\partial E}{\partial z_i} + \sum_{k>i} \frac{\partial E}{\partial z_k} \frac{\partial^+ z_k}{\partial z_i}. \end{cases} \tag{26}$$

This procedure is called backpropagation through time. The first equation in (26) leads to a costate equation, while the second one leads to the previous sensitivity method of Narendra's dynamic backpropagation. In [4] a simplification of the backpropagation through time has been proposed and successfully tested. It is obtained by more faithfully respecting the principle of optimality which underlies the computational economy allowed by dynamic programming for optimal control.

4.4 Q-learning

The Q-learning method in reinforcement learning is related to dynamic programming. Whereas dynamic programming can be applied when a model for the plant is given, Q-learning applies to the case where a model is not available and is in fact a direct adaptive optimal control strategy [28]. Q-learning is an on-line incremental approximation to dynamic programming. Considering a finite state finite action Markov decision problem, the controller observes at each time k the state x_k, selects an action a_k, receives a reward r_k and observes the next state x_{k+1}. The objective is to find a control rule that maximizes at each step the expected discounted sum of future reward

$$E\{\sum_{j=1}^{\infty} \gamma^j r_{k+j}\} \tag{27}$$

with discount factor γ $(0 < \gamma < 1)$. The basic idea in Q-learning is to estimate a real valued function $Q(x, a)$ of state x and action a, which is the expected discounted sum of future reward for performing action a in state x and performing optimality thereafter. This function satisfies the recursive relationship

$$Q(x, a) = E\{r_k + \gamma \max_b Q(x_{k+1}, b) | x_k = x, a_k = a\}. \tag{28}$$

Given x_k, a_k, r_k, x_{k+1}, the Q-learning scheme then works with an estimate $\hat{Q}$ that is updated as

$$\hat{Q}(x_k, a_k) := \hat{Q}(x_k, a_k) + \beta_k[r_k + \gamma \max_b \hat{Q}(x_k, b) - \hat{Q}(x_k, a_k)] \tag{29}$$

where β_k is a gain sequence $(0 < \beta_k < 1)$ and $\hat{Q}(x, a)$ remains unchanged for all pairs $(x, a) \neq (x_k, a_k)$. One disadvantage of this method is that the required memory is proportional to the number of (x, a) pairs which leads to a curse of dimensionality.

5 $\mathbf{NL}_q$ stability theory

A modelbased framework for neural control design, with neural state space models

$$\left\{ \begin{array}{rcl} \hat{x}_{k+1} & = & W_{AB}\sigma(V_A\hat{x}_k + V_B u_k + \beta_{AB}) + K\epsilon_k \\ y_k & = & C\hat{x}_k + Du_k + \epsilon_k \end{array} \right. \tag{30}$$

and either linear controller

$$\left\{ \begin{array}{rcl} z_{k+1} & = & Ez_k + Fy_k + F_2 d_k \\ u_k & = & Gz_k + Hy_k + H_2 d_k \end{array} \right. \tag{31}$$

or neural state space controllers

$$\left\{ \begin{array}{rcl} z_{k+1} & = & W_{EF}\sigma(V_E z_k + V_F y_k + V_{F_2} d_k + \beta_{EF}) \\ u_k & = & W_{GH}\sigma(V_G z_k + V_H y_k + V_{H_2} d_k + \beta_{GH}) \end{array} \right. \tag{32}$$

has been proposed in [31]. Here W_*, V_* denote interconnection matrices and β_* bias vectors. One takes tanh(.) for $\sigma(.)$. Like in modern control theory one works then with standard plant forms with exogenous input w_k (consisting of the reference input d_k, noise ϵ_k and a constant due to the bias terms of the neural nets), regulated output e_k (consisting of tracking error and possibly other variables of interest), sensed output y_k and actuator input u_k.

The equations for the closed-loop system are transformed then into a so-called NL_q system form:

$$\left\{ \begin{array}{rcl} p_{k+1} & = & \Gamma_1(\, V_1\, \Gamma_2(\, V_2\, ...\Gamma_q(\, V_q\, p_k + B_q w_k)... + B_2 w_k) + B_1 w_k) \\ e_k & = & \Lambda_1(W_1 \Lambda_2(W_2 ... \Lambda_q(W_q p_k + D_q w_k)... + D_2 w_k) + D_1 w_k) \end{array} \right. \tag{33}$$

where Γ_i, Λ_i $(i = 1, ..., q)$ are diagonal matrices with diagonal elements $\gamma_j(p_k, w_k)$, $\lambda_j(p_k, w_k)$ $\in [0, 1]$, depending continuously on the state p_k and the input w_k. The matrices V_*, B_*, W_*, D_* are constant and have compatible dimensions. The term 'NL_q' refers to the alternating sequence of nonlinear and linear operators (q layers).

In [31] sufficient conditions for global asymptotic stability and I/O stability (dissipativity with finite L_2-gain) have been derived. The criteria are typically of the form:

$$\|DZD^{-1}\|_2 < 1 \tag{34}$$

or

$$c(P)\, \|PZP^{-1}\|_2 < 1 \tag{35}$$

where the matrix Z depends on the interconnection matrices of the model and the controller and one has to find a diagonal matrix D or a blockdiagonal matrix P such that the inequality is satisfied. $c(P)$ is a correction factor ($c(P) > 1$), which depends on the degree of diagonal dominance of the matrix $P^T P$ or on the condition number of P. For a given matrix Z the conditions (34) and (35) can be expressed as linear matrix inequalities (LMIs), which leads to convex optimization problems [5]. Under certain conditions the criteria can be interpreted as extensions towards the state space upper bound test in μ robust control theory.

With respect to neural control, Narendra's dynamic backpropagation procedure can be modified then with a stability constraint in order to track on specific reference inputs such that closed-loop stability is guaranteed. Moreover it is possible to avoid tracking on specific reference inputs, by using the NL_q theory for nonlinear H_∞ control on multilayer recurrent neural networks in the standard plant framework of modern control theory. It has been shown that several types of nonlinear systems (stable with a unique or multiple equilibria, periodic, quasi-periodic, chaotic) can be controlled by taking this approach [31].

6 Conclusion

In this paper we gave a short introduction on the theory of neural control. Many methods emerged in recent years. The emphasis in this paper was on some basic aspects of neural optimal control, stability theory and nonlinear system identification using neural networks with respect to modelbased neural control design.

References

[1] Barron A.R. (1993). Universal approximation bounds for superposition of a sigmoidal function, *IEEE Transactions on Information Theory*, Vol.39, No.3, pp.930-945.

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

[3] Bersini H., Bontempi G., Decaestecker C. (1995). Comparing RBF and Fuzzy Inference Systems on Theoretical and Practical Basis, *Proceedings of ICANN'95*. pp.169-174.

[4] Bersini H. (1995). A simplification of the Back-Propagation-Through-Time algorithm for Optimal Neurocontrol, *Proceedings of the World Congress on Neural Networks '95 Conference*.

[5] Boyd S., El Ghaoui L., Feron E., Balakrishnan V. (1994). *Linear matrix inequalities in system and control theory*, SIAM (Studies in Applied Mathematics), Vol.15.

[6] Chen S., Billings S., Grant P. (1990). Nonlinear system identification using neural networks, *International Journal of Control*, Vol.51, No.6, pp. 1191-1214.

[7] Chen S., Cowan C., Grant P. (1991). Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Transactions on Neural Networks*, Vol.2, No.2, pp.302-309.

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

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

[10] Hunt K.J., Sbarbaro D. (1991). Neural networks for nonlinear internal model control, *IEE Proceedings-D*, Vol.138, No.5, pp.431-438.

[11] Hunt K.J., Sbarbaro D., Zbikowski R., Gawthrop P.J. (1992). Neural networks for control systems - a survey, *Automatica*, Vol. 28., No. 6, pp.1083-1112.

[12] Leshno M., Lin V.Y., Pinkus A., Schocken S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural Networks*, Vol.6, pp.861-867.

[13] Miller W.T., Sutton R.S., Werbos P.J. (1990). *Neural networks for control*, Cambridge, MA: MIT Press.

[14] Narendra K.S., Parthasarathy K. (1990). Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, Vol.1, No.1, pp. 4-27.

[15] Narendra K.S., Parthasarathy K. (1991). Gradient methods for the optimization of dynamical systems containing neural networks, *IEEE Transactions on Neural Networks*, Vol.2, No.2, pp.252-262.

[16] Nguyen D., Widrow B. (1990). Neural networks for self-learning control systems, *IEEE Control Systems Magazine*, 10(3), pp.18-23.

[17] Parisini T., Zoppoli R. (1994). Neural networks for feedback feedforward nonlinear control systems, *IEEE Transactions on Neural Networks*, Vol.5, No.3, pp.436-449.

[18] Park J., Sandberg I.W. (1991). Universal approximation using Radial-Basis-Function networks, *Neural Computation*, 3, pp.246-257.

[19] Poggio T., Girosi F. (1990). Networks for approximation and learning, *Proceedings of the IEEE*, Vol.78, No.9, pp.1481-1497.

[20] Psaltis D., Sideris A., Yamamura A. (1988). A multilayered neural network controller, *IEEE Control Systems Magazine*, April, pp.17-21.

[21] Rumelhart D.E., Hinton G.E., Williams R.J. (1986). Learning representations by back-propagating errors, *Nature*, Vol.323, pp.533-536.

[22] Saerens M., Soquet A. (1991). Neural controller based on back-propagation algorithm, *IEE Proceedings-F*, Vol.138, No.1, pp.55-62.

[23] Saerens M., Renders J.-M., Bersini H. (1995). Neural controllers based on backpropagation algorithm. In *IEEE Press Book on Intelligent Control: Theory and Practice*, M. M. Gupta, N. K. Sinha (Eds.), IEEE Press.

[24] Sanner R.M., Slotine J.-J. E. (1992). Gaussian networks for direct adaptive control, *IEEE Transactions on Neural Networks*, Vol.3, No.6, pp. 837-863.

[25] Schiffmann W.H., Geffers H.W. (1993). Adaptive control of dynamic systems by back propagation networks, *Neural Networks*, Vol.6, pp.517-524.

[26] Sjöberg J., Ljung L. (1992). Overtraining, regularization and searching for minimum in neural networks, *4th IFAC International Symposium on adaptive systems in control and signal processing*, ACASP 92, pp.669-674, Grenoble, France.

[27] Sjöberg J., Zhang Q., Ljung L., Benveniste A., Delyon B., Glorennrc P., Hjalmarsson H., Juditsky A. (1995). Nonlinear black-box modeling in system identification: a unified overview, *Automatica*, Vol.31, No.12, pp.1691-1724.

[28] Sutton R.S., Barto A., Williams R. (1992). Reinforcement learning is direct adaptive optimal control, *IEEE Control Systems*, April, pp.19-22.

[29] Suykens J.A.K., De Moor B., Vandewalle J. (1994). Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points, *Neural Networks*, Vol.7, No.5, pp.819-831.

[30] Suykens J.A.K., De Moor B., Vandewalle J. (1995). Nonlinear system identification using neural state space models, applicable to robust control design, *International Journal of Control*, Vol.62, No.1, pp.129-152.

[31] Suykens J.A.K., Vandewalle J.P.L., De Moor B.L.R. (1995). *Artificial Neural Networks for Modelling and Control of Non-Linear systems*, Kluwer Academic Publishers, Boston.

[32] van der Smagt P.P. (1994). Minimisation methods for training feedforward neural networks, *Neural Networks*, Vol.7, No.1, pp.1-11.

[33] Werbos P. (1990). Backpropagation through time: what it does and how to do it, *Proceedings of the IEEE*, 78 (10), pp.1150-1560.

[34] Zurada J.M. (1992). *Introduction to Artificial Neural Systems*, West Publishing Company.