AN ASSOCIATIVE MEMORY BASED LEARNING CONTROL SCHEME WITH PI-CONTROLLER FOR SISO-NONLINEAR PROCESSES

E. Ersü and S. Wienand

Isra Systemtechnik GmbH, Schöfferstraße 15, 6100 Darmstadt, Federal Republic of Germany

Abstract. The paper discusses the real-time implementation of an associative-memory-based learning control scheme with PI-controllers for nonlinear processes. Starting with a pre-assumed PI-controller which only has to stabilize the process the controller parameters are optimized on-line by a predictive optimization. This optimization uses for prediction the model of the process stored in an associative memory which is also learned on-line. The situation-dependent optimized controller parameters are also stored in an associative memory. The concept is a modification of the LERNAS-system (Ersü,1984), which is also shortly described and compared to the system described here. Some experimental results with a nonlinear pH-control demonstrate the performance of the system.

Keywords. Adaptive control; Artifical intelligence; Learning systems; nonlinear control system; pH control; Predictive control; self-organising storage.

INTRODUCTION

The fact, that most of the complex industrial processes, especially if they are uncertain or fully or partially unknown, have made self-organizing control schemes into an important research area in control theory. Since the beginning of the 1950's very active research has resulted in various forms of self-organizing control, e.g. adaptive, learning, fuzzy, linguistic, etc. An efficient implementation of these algorithms could only be realized by the rapid and revolutionary progress in microelectronics. A major part of recent industrial applications uses parameter adaptive control schemes by which a linear feedback law is on-line updated depending on an implicit or explicit identification of a linear model of the unknown process.

The learning control method LERNAS (Learning Control with Neuron-like Associative Memory Systems) represents a novel type of self-organizing control system which is motivated by the neurobiological and psychological research on the brain and the growth of the VLSI-technology as well. The main novelty of the method introduced by Ersü (1980, 1983) and discussed by Ersü and Tolle (1984) is that the concept uses a control action test and optimization through a 1-step ahead output predictive algorithm (similar to human problem solving) with both the predictive model of the unknown environment and the control strategy being represented by general mathematical (i.e. nonlinear) mappings. Both mappings are carried out by special neuron-like associative memory systems similar to information storage in neuronal networks.

The concept described in this paper is a modification of this method. The model for the process is the same and it is learned in the same way. The control-strategy, however, is different. The LERNAS concept described above assumes that a priori information about the process is hardly available. But in most cases this assumption is not valid. In general there exists sufficient a priori information to design a PI-controller which only stabilizes the process. Now instead of the controller-response to a given state the optimal parameter set for a given state of the process is learned. These parameters are optimized utilizing the predictive process model stored in an associative memory.

The advantages and disadvantages of this method with respect to the method described above will be demonstrated in this paper.

LEARNING CONTROL CONCEPT

In classical system theory input-output descriptions are based on an assumed or predetermined mathematical structure, normally a set of differential equations. Replacement of these predetermined structures by input-output type of associative mappings leads to more general representations by (in general nonlinear) mathematical mappings between n-dimensional input vectors I and m-dimensional output vectors O:

Original publication in IFAC Proceedings Volumes (now IFAC-Papers online), https://doi.org/10.1016/S1474-6670(17)55837-1

$$M: \underline{I} \rightarrow \underline{O} \qquad (2.1)$$

Given a time-descrete representation of a time-invariant deterministic system

$$\hat{\underline{y}}(k+1)=f(\underline{\psi}(k), \underline{v}(k), \underline{u}(k)) \qquad (2.2)$$

with input vectors $\underline{u} \in U \subset R$

output vectors $\underline{y} \in Y \subset R$

measurable disturbance vector $\underline{v}$ $V \subset R^{n_v}$

state description

$$\underline{\psi}(k)=(y(k), \underline{y}(k-1),\ldots, \underline{y}(k-\alpha), \underline{u}(k-1),\ldots, \underline{u}(k-\beta), \underline{v}(k-1),\ldots, \underline{v}(k-\gamma))$$

$\underline{\psi} \in \Psi$

scalars α, β, γ

sampled time $t=k.T_s$

and sampling period T_s

an associative mapping for (2.2) can be defined by

$$S:(\underline{\psi}(t), \underline{v}(k), \underline{u}(k)) \rightarrow \underline{y}(k+1) \qquad (2.3)$$

which can be learned on-line via associative memorization. This procedure does not require any structural information about the process at hand except the scalars α, β and γ representing the amount of history to be used in the model.

Associative memorization is a storage technique in which the location of the output information is addressed by the contents of the input information only (Kohonen, 1977).This type of information retrieval which is an outstanding feature of the associative neuronal network models avoids long searching procedures as they are generally necessary in conventional memory systems. The special memory system AMS (Associative Memory System) used for the implementation of the control concept is a Perceptron-like (Rosenblatt, 1962) distributed trainable associative memory system by which a real-valued function or mapping of several variables can be evaluated via simple content-addressing mechanisms rather than by complex mathematical operations.

Based on the concept of CMAC (Cerebellar Model Articulation Controller) proposed by Albus (1972, 1975) as a neuronal network model of the human cerebellar cortex AMS has been further modified and developed for real-time control purposes (Ersü, 1982). In AMS the information storage for an association pair $(\underline{I}_i, \underline{O}_i)$ occurs by distributing the novel output information over a constant number (r^*) of memory locations, thus for any input information the output is always formed by a recollection of the information elements located in the corresponding (to $\underline{I}_i$) r^* memory cells. Depending on the similarity of input information, e.g. the difference between two inputs $\|\underline{I}_i - \underline{I}_j\|$ the mapping procedure recalls corresponding sets of information elements which share the memory cells to an extent which is a function of the input similarity. This leads to one of the most important features of AMS called generalization, i.e. similar inputs tend to produce similar outputs. Mathematically it can be interpreted as a multi-dimensional interpolation in the neighbourhood of a trained point $(\underline{I}^o, \underline{O}^o)$ illustrated in figure 1. The degree of generalization which is fundamental for on-line learning can be thus pretuned by the generalization variable r^* and the quantization variable. As memory locations are shared by similar inputs the AMS procedure also reduces the amount of memory to a great extent.

CONTROL STRATEGY

For simplification here only a single-variable process is considered. The method is nevertheless also applicable for multivariable-processes with some modifications.

The concept shown in fig. 2 incorporates two AMS-type memory systems, one for the predictive model of the unknown process

$$M: (\underline{\psi}_M(k), \underline{v}(k), u(k)) \rightarrow \hat{y}(k+1) \qquad (3.1.)$$

corresponding to $\underline{\psi}$ with some $\alpha_M, \beta_M, \gamma_M$ (instead of α, β, γ)

and one for the control strategy

$$P: (\underline{\psi}_P(k), \underline{v}(k), y(k)) \rightarrow \underline{p}(k) \qquad (3.2.)$$

With setpoint value $w(k) \in Y$ and $\underline{\psi}_P$ in practice similar to $\underline{\psi}_M$.

u(k) is calculated in a PI-control-algorithm:

$$u(k)=u(k-1)+q_0 x_d(k)+q_1 x_d(k-1) \qquad (3.3.)$$

Where $\underline{p}(k)=(q_o, q_1)^T$

The PI-algorithm was choosen because it is a robust algorithm with respect to model uncertainities. Further there are only two parameters to optimize.

The behaviour of the process is determined by a reference-model. This method results in a better learning-behaviour of the system because it always follows similar trajectories.

Assuming that

i. the unknown multivariable process at hand is a determinstic time-invariant or weakly time-variant BIBO-stable process,

ii. u,y and $\underline{v}$ are quantized and of finite subspaces,

iii. the overall goal

$$I_G=\sum_{k=0}^{k_e-1} L_G(y(k+1), w(k+1), u(k)) \quad (3.4.)$$

can be substituted by a 1-step ahead subgoal

$$I_S=\sum_{i=0}^{1} L_S(y(k+i), w(k+i), u(k+i-1)) \quad (3.5)$$

IV. a parameterset $\underline{p}_v$ is known which stabilizes the process

V. the process is able to follow the model-trajectory

the algorithmic scheme in each sampling period is as follows:

1. the prediction-model is updated by the measured prediction-error $e(k)=y(k)-\hat{y}(k)$ where $\hat{y}(k)$ is the predicted value from step k-1.

2. an optimization scheme is activated. It tries to calculate the parameter-set $\underline{p}^*(k)$ which is the estimated optimal set for the subgoal $I_S(k)$. In fact only $\hat{I}_S(k)$ - the estimated subgoal - can be calculated, because for future y(k+i) only the estimated value $\hat{y}(k+i)$ is available. Besides the whole prediction has to take place in the trained area of the model, because otherwise no information about the behaviour is available. That means that the resulting $\underline{p}^*(k)$ also depends on the information available in the process-model. The prediction-length l is adapted to the training-situation of the process-model in the boundaries

   $lmin < l < lmax$

   lmin is the shortest prediction which allows a critical evaluation of the parameter-set. lmax is the longest prediction which can assure estimated values which are to some extent accurate.

   There are three different possibilities for the starting point of the optimization:

   1. the initial parameter-set $\underline{p}_v$
   2. the old parameter-set $\underline{p}(k-1)$
   3. the parameter-set $\underline{p}^*(k)$ which is the preliminary optimal parameter-set out of the parameter-AMS.

   3. can only be used if the parameter-AMS is trained for the appropriate state of the process.

   2. can be used as long as this parameterset doesn't lead to an undesired behaviour of the process. This can be the case when the behaviour of the process in the actual state is (because of nonlinearities) very different to the behaviour in the state the previous parameter-set was originally optimized. To detect such a situation a supervisor-level was introduced.

   If none of the two parameter-sets is applicable the initial parameter-set $\underline{p}_v$ is used.

3. The parameter-set $\underline{p}^*(k)$ optimized by (2.) is memorized in parameter memory to be used as the best decision, making (2.) superfluous in the long range, and giving either an excellent optimization starting point or being used without further inclusion of the predictive learning loop after a user-defined point of time.

4. The last step is the calculation of the control-output $u^*(k)$. This is done via the PI-algorithm

   $$u^*(k)=u(k-1)+q_0 x_d(k)+q_1 x_d(k-1)$$

   where $(q_0, q_1)^T=\underline{p}(k)$.

   Again there are the three possibilities mentioned above for $\underline{p}(k)$.

   Before applying a control input to the process a supoptimal control input $\bar{u}(k)\neq u^*(k)$ with

   $$\| \bar{u}(k) - u(k) \| < \delta$$

   but

   $\bar{u}(k) \notin G_T$

   G_T = trained area of model-AMS

   for some specified δ is determined.

   $\bar{u}(k)$ applied to the process excites it to further untrained information for the model AMS and, so, enlarges G_T. This exploratory procedure called active learning is speeding up the learning.

It should be pointed out that the fact that the old parameterset $\underline{p}(k-1)$ can be used for control is an advantage of this method in comparison to the original LERNAS-method. A parameterset in connection with a fixed controller-structure can provide a response to every state of the process. Depending on the nonlinearity of the process this will not be the optimal response in all cases, but in most cases it is similar to it.

This is not true if the u(k) are stored directly. Then the learned u is only a response to that specific state of the process.

With increasing learning-time the system can adapt the controller-parameters to the nonlinearities of the process. But, as the information "parameterset" is more global than the information "controller-output", it can not be adapted to such an extend. That means that the original LERNAS-system is better suited for very high nonlinear processes with fast changes in behaviour, whereas the described system is well suited for

processes without these fast changes. In the latter case it has three major advantages:

1. It requires less memory, because a parameterset can be used for a greater area than a control-output and therefore it is not necessary to distinguish so many states of the process.
2. For the same reason it can learn more quickly: A parameterset once learned can be used in a greater area and there are less different parameter sets to learn than controller-outputs.
3. In the case of very high disturbances it leads to much better behaviour. The original LERNAS must learn the response to each disturbance-input. When this disturbance-input occurs the first time the reaction is slow, because the control-AMS has not yet learned a response to it. The system described here uses a PI-controller as backup. Therefore it can immediately react on a disturbance-input because the PI-controller "knows" an answer to it (even if it is not the optimal answer). It should be pointed out that in the case of smaller disturbances also the original LERNAS is able to react due to the ability of pluralization.

SOME EXPERIMENTAL RESULTS

The described control-algorithm has been tested with a simulated nonlinear pH-control-process. The major nonlinearity of this process is the titration-curve (fig. 3). The system had to learn to change the pH-value periodically between 7 and 9 (the most nonlinear area) according to the given desired trajectory.

Fig. 4 shows the behaviour with the predetermined parameterset. It can easily be seen that the behaviour is not very good and that it depends on the set-point. In fig. 5 the behaviour in the region between 7 and 8 is shown after the system has followed three times the model-trajectory shown in fig. 4. In fig. 6 the same is shown after following the trajectory 15 times. Figure 7 shows the parameter-sets used in every step (the optimized parameters from the parameter-AMS) after 3 learning-cycles, fig. 8 after 15 learning-cycles.

The figures demonstrate that with increasing learning-time the system is able to follow the model-trajectory better and that it is able to adapt the parameters to the nonlinearity of the process.

The response to very high disturbance-inputs (up to 100 %) have also been tested. For this test a well trained parameter-AMS was used. Figure 9 shows that the system is able to achieve a good behaviour.

CONCLUSION

A modification of the LERNAS-concept was described which uses associative memories to store the optimized parameters of a PI-controller. The parameters are optimized on-line with the help of a process-model which is also stored in an AMS and also learned on-line. The simulation shows that it is well suited for nonlinear processes with not too high nonlinearities. Only very few a priori-information about the process is needed. It is especially not necessary to have a mathematical model for the process.

The concept is at the moment very well applicable for SISO processes. The ongoing work will be concentrated on an extention to MIMO processes.

REFERENCES

Albus, J.S., (1972). Theoretical and experimental aspects of a cerebellar model. PhD Thesis, Univ. of Maryland.

Albus, J.S., (1975). The cerebellar model articulation controller. Trans ASME, series G, Vol. 97, No. 3.

Ersü, E., (1980). Self-organizing control with associative memories. Internal Report at the Inst. of Control Engineering, Techn. Univ. of Darmstadt (in German).

Ersü, E. and Militzer, J., (1982). Software implementation of a neuron-like associative memory system for control applications. Proc. of the ISMM 8th Int. Symp. MIMI, Davos, Switzerland.

Ersü, E., (1983). On the application of associative neural network models to technical control problems. Proc. in Life Sciences: Localization and orientation in biology and engineerin ed. Varju and Schnitzler, Berlin, Heidelberg, New York.

Ersü, E. and Tolle, H., (1984). A New Concept For Learning Control Inspired by Brain Theory. Proc. of the IFAC 9th World Congress, Budapest/Hungary.

Ersü, E. and Militzer, J., (1984). Real-time implementation of an associative memory-based learning control-scheme for nonlinear multivariable process. Symposium "Application of multi-variable system technique", Plymouth.

Kohonen, T., (1977). Associative Memory. Springer, Berlin, Heidelberg, New York.

Rosenblatt, F., (1962). Principles of Neurodynamics. Spartan Books, Washington.

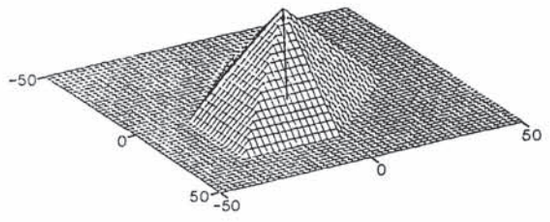

Fig. 1 Generalization demonstrated by training at the origin of the input plane of an AMS with two inputs and one output (r*=32 and ε =1). The peak at the origin would be the AMS-response for r*=1, i.e. no generalization, the case of conventional storage

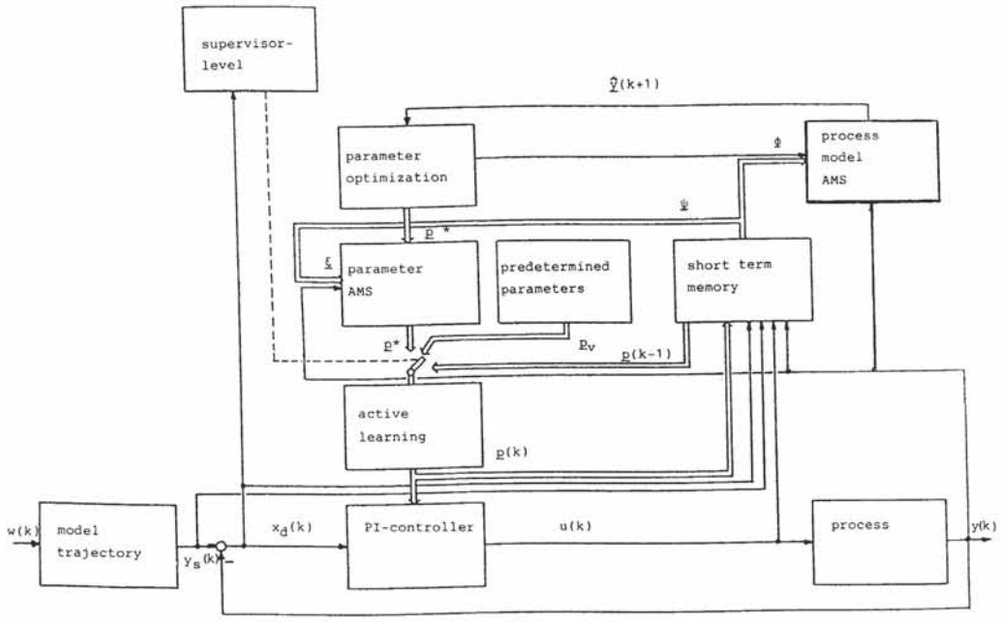

Fig. 2 Control-scheme

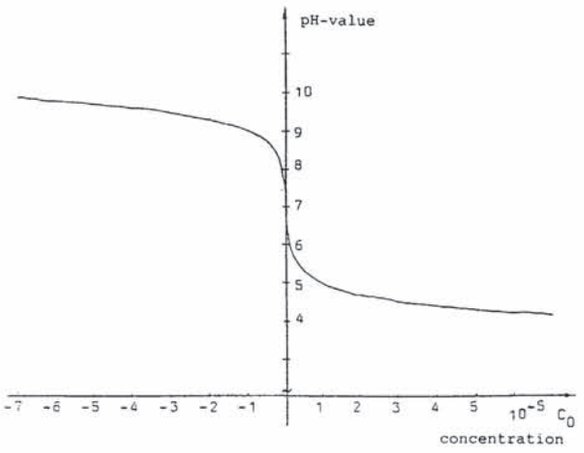

Fig. 3 Titration-curve

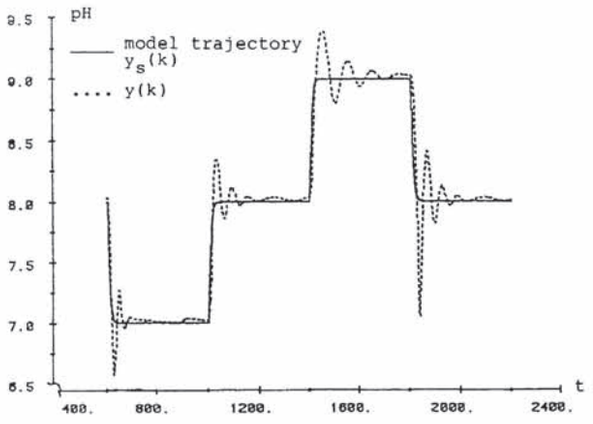

Fig. 4 Behaviour with initial parameterset

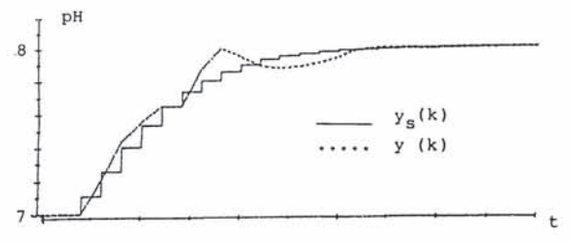

Fig. 5 Behaviour after 3 learning-cycles

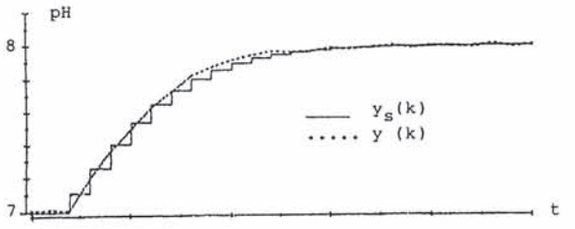

Fig. 6 Behaviour after 15 learning-cycles

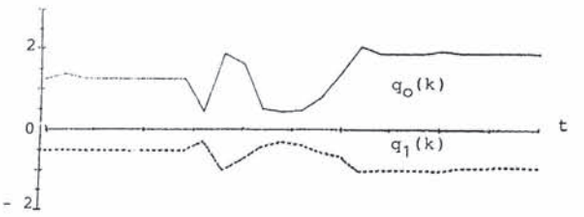

Fig. 7 Parameters after 3 learning-cycles

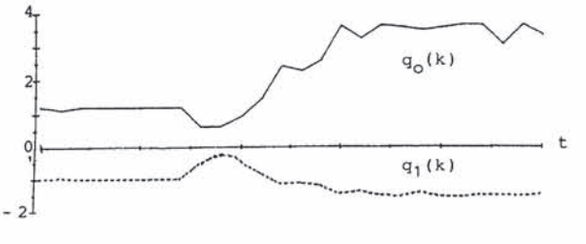

Fig. 8 Parameters after 15 learning-cycles

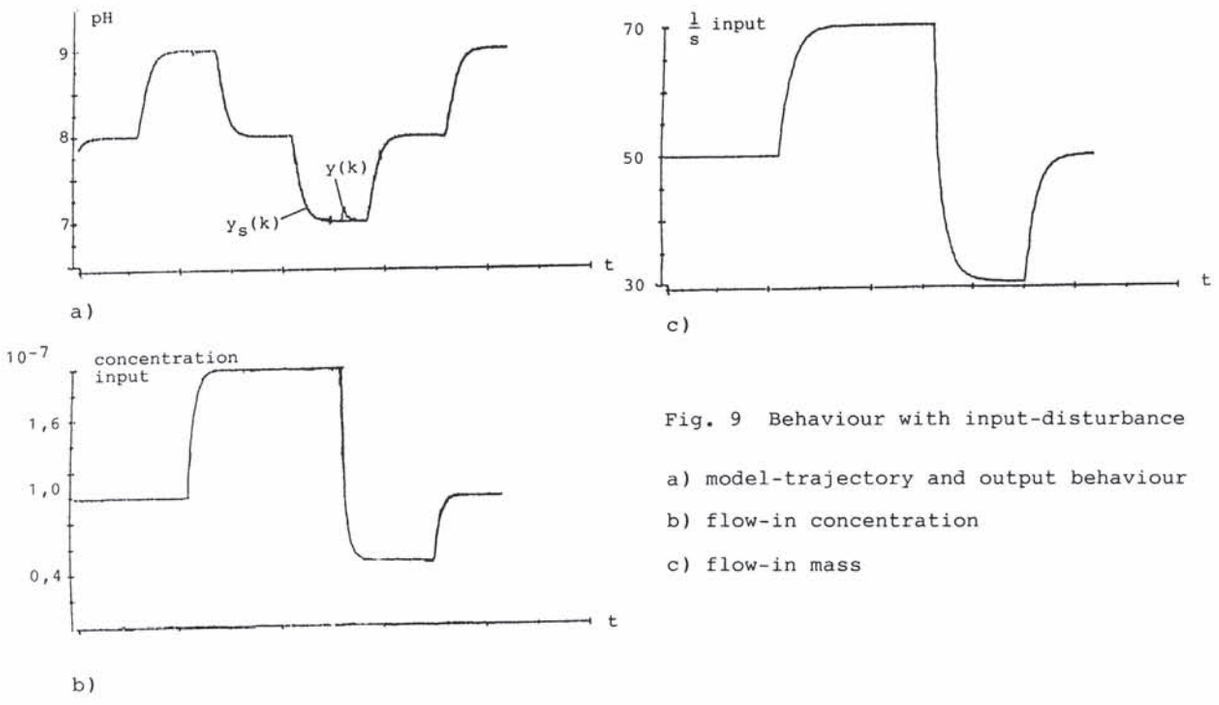

Fig. 9 Behaviour with input-disturbance

a) model-trajectory and output behaviour

b) flow-in concentration

c) flow-in mass