Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Sound and Vibration 266 (2003) 601–612
JOURNAL OF SOUND AND VIBRATION
www.elsevier.com/locate/jsvi

Vibration control using self-organizing look-up tables

D. Moshou*, H. Ramon

Laboratory for Agro-Machinery and Processing, Department of Agro-Engineering and Economics, K.U. Leuven, Kasteelpark Arenberg 30, 3001 Heverlee, Leuven, Belgium

Received 13 January 2003

Abstract

The self-organizing map neural network is used in a supervised way to represent a sensor–actuator mapping. The learning of the controller assumes no prior information, but only reward/failure signals that are produced by an evaluation criterion. The evaluation criterion used is based on the low-pass filtering of the gradient of a reward function and the local storing of the filtered gradient value. The control method is tested in vibration isolation of a flexible spray boom used in agriculture for pesticide application. The neural network learns to stabilize the boom on-line without any prior information and with a very high performance.

1. Introduction

Many mechanical structures are subjected to vibrations that can lead to damage or to fatigue and thus shorten the operational lifetime of the structure. Passive vibration isolation gives poor results because of low selectivity. Active vibration isolation is much more effective. However, model-based techniques require continual excitation signals. In practice continual excitation is rarely available during the operating condition of a system. An alternative is to develop an algorithm that can discover the control actions by itself. The only source of information in this case is a "reward function" which specifies at a given moment how well the controller has performed. For this algorithm to be executed the system must now create at each learning step the control action [1].

In the absence of any further information, a stochastic search can be performed in the space of the available control values with the aim to maximize the reward received at each step. Performance-based partitioning of the state space is achieved. Current sensor, actuator and target

*Corresponding author. Tel.: +32-16-321922; fax: +32-16-328590.
E-mail address: dimitrios.moshou@agr.kuleuven.ac.be (D. Moshou).

0022-460X/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0022-460X(03)00587-X

sensor values in a vectorized form become associated with next step control actions. Through the maximization of a certain reward function, a goal-directed plant inversion is performed.

For continuous state spaces, the state-action look-up table refers to a quantization of the states of the system through the use of an adaptive algorithm. Basically two types of learning are present here:

(i) the adaptation of the partitioner, and
(ii) the reinforcement learning of the controller [2].

The determination of quantized states, which are internal states in the full control problem, represents an instance of the hidden state problem [3]. For discrete actions, an ideal partition of the continuous state space consists of domains with each having a unique optimal action for all states belonging to that domain. In this way, an optimal partition is defined by the use of a policy function that assigns states to actions. In the current paper the partitioning is performed by using a self-organizing vector quantizer [1] combined with a reinforcement learning algorithm that switches learning on and off based on the policy function. The distribution of the reference vectors is usually determined by statistical properties of the inputs to the vector quantizer [4] and the switching actions scheduling determined by the policy function.

In this paper, the learning rule for the vector quantizer is based on Kohonen's self-organizing feature map [5], which possesses interesting noise-filtering properties. The Self-Organizing Map commonly referred to as SOM [6] is a neural network (NN) that converts complex, non-linear statistical relationships between high-dimensional data into simple geometric relationships.

The determination of a quantized state cannot by itself represent input–output relationships. By extending the SOM with output weights that store the output part of a mapping can provide the original algorithm with the ability to approximate continuous relationships. Such a network has been introduced earlier [1]. In the current paper for first time a partitioner based on SOM is learned simultaneously with a reinforcement signal-based learning controller. A novel training algorithm is presented for updating the parameters of this network. Then, this training algorithm is successfully applied in the on-line stabilization of a flexible spray boom that is used in pesticide application.

2. Reinforcement learning

In order to learn about various possible actions and the corresponding system performance, the intelligent controller must autonomously generate the control signals. How this can be properly done in order to obtain samples with high-utility needs to be addressed. The problem is related to a proper experimental exploration of the system under consideration, but this is solved by an "intelligent explorer" who performs the measurements. By generating random control actions, the operator begins by exploring the properties of the system and its response to the influences from the environment. Among the various actions, those that correspond to increased performance are then selected to reinforce the control, until an optimal operation in a particular situation is obtained. While exploring the system to be controlled, an intelligent operator also remembers which actions are most favourable in a particular situation, and later uses this information to

predict the proper actions that are associatively based on the perception of the state of the system and its environment [7]. A strategy similar to that described above is applicable to the design of intelligent controllers, which are capable of reinforced learning. In order to obtain proper information, the controller generates random actions and remembers them, together with the corresponding system response as well as the utility. In the optimization, the performance of the past is used to find an action that increases it at a particular time. For this purpose, the previously memorized actions of the controller are varied slightly, and those actions that increase the performance of the system are appropriately reinforced. In order to obtain a reinforcement of the momentary, appropriate actions of the controller, the utility is permanently estimated, and the corresponding information is further utilized to adjust the actions of the controller. The likelihood of obtaining a global extremum of performance can he enhanced by gradually including increasingly more steps into the estimation of appropriate actions. It is therefore reasonable to use these corrected values in the self-organization process by which the prototypes are adapted to the dynamic phenomenon taking place in the controlled system and the environment. This corresponds to approximating the procedure of dynamic programming using neural networks [8]. The corresponding changes are expected to drive the prototypes into the region of the state space with a high average utility. The reinforcement procedure corresponds to a training process of an intelligent controller. It is advantageous to use a variation of sample vectors to find a refined set of learning vectors, by which the memory is then formed. It can be expected that with an increasing number of tests the values of a performance-related reward function will increase (or decrease in the case it is defined as a cost function). At the start of the training the memory is thus mainly filled with prototype vectors, which are randomly distributed in the state space, while later they become more concentrated in the region of maximal average utility. The estimation of the optimal control is thus approximate and inaccurate at the start of the training, becoming refined as the number of tests increases. The reinforcement learning resembles the properties of the calculus of variations that is carried out in arithmetically finding the solution of optimal control problems. The key control problem of reinforcement learning stems from the fact that the gradient of the utility in the state space of control cannot be estimated very accurately from empirical data obtained by only randomly generated samples. A continuing exploration of the state space is also needed when at a certain instant the set of prototypes is indeed an optimal set for a current situation, but later the properties of the environment change. The term optimal does not necessarily mean that the number of prototypes is optimal but that the pre-selected number of prototypes have stored optimal states and actions. For a corresponding change in the optimal controller, the state space must be explored once again. It is characteristic of the proposed system that the random exploration of the state space and the self-organization process can be permanently active during the operation of the system. Such a compound process could be termed "reinforced self-organization". It is a basis of state-space exploration. Based on this, an improvement in the intelligent controller can be achieved.

3. Controller and partitioner learning

The SOM [5] is an NN that maps signals ($\mathbf{x}$) from a high-dimensional space to a one- or two-dimensional discrete lattice of neuron units ($\mathbf{s}$). Each neuron stores a weight ($\mathbf{w}_s$). The map

preserves topological relationships between inputs in a way that neighbouring inputs in the input space are mapped to neighbouring neurons in the map space. When extended with output weights ($\mathbf{y}_s$) it can actually learn in a supervised way the mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$. The input weights consist of vectors of a combination of delayed values of measured system inputs and outputs. The output space consists of the following step values of the control inputs that are associated through the reinforcement learning procedure based on the maximization of a reward function. It is clear that previous output values are included in the input but only next step output values are stored in the output weights. This association of input and outputs is supervised since the association takes place through the reinforcement learning procedure. This association is shown schematically in Fig. 1.

The association of situation–action pairs is based on a reward/punishment scheme where a reinforcement signal is produced that allows or not storage of a certain situation and action pair. A two-state reinforcement signal ($+1$ or 0) is produced after an evaluation of the result that a certain action has brought. A policy to update or not the SOM weights can be based on a reward function by calculating the difference between two consecutive values of the reward function like those presented in Eqs. (4) and (5). The continuous increase of the reward function can be used for the production of a reinforcement signal that allows learning to take place. The creation of prototypes by the self-organization process should be interpreted as a memorization process that optimally preserves the information provided by the samples, while the generation of control actions brings new information. This generation is appropriately guided by the reinforcement process, which in turn leads to a search for new, favourable samples.

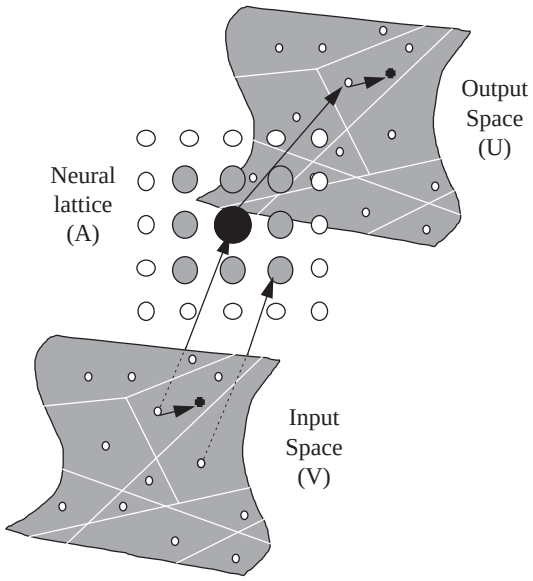

Fig. 1. Association of input–output values by using a SOM. The input space consists of a vector of delayed values of measured system inputs and outputs. The output space consists from the following step values of the control inputs that are associated through the reinforcement learning procedure based on the maximization of a reward function.

In a control situation when a system is driven by a certain control sequence

$$\mathbf{u} = (u(k-1), u(k-2), \ldots, u(k-m))^{\mathrm{T}} \tag{1}$$

and the measured response of the system is

$$\mathbf{y} = (y(k), y(k-1), \ldots, y(k-n))^{\mathrm{T}} \tag{2}$$

and the desired next step output is denoted as $\mathbf{y}_d$ (generally a vector, but in the example with the boom it is assumed to be a scalar), the state vector that is used as input to the state quantizer (SOM) is constructed as follows:

$$\mathbf{x} = (\mathbf{y}_d^{\mathrm{T}}, \mathbf{y}^{\mathrm{T}}, \mathbf{u}^{\mathrm{T}})^{\mathrm{T}}. \tag{3}$$

Subsequently, these state vectors are clustered by the SOM only if a continuous increase of the difference of the reward function defined in Eq. (5) over a stored moving average (Eq. (6)) is actually taking place. This can be achieved by simply comparing the values of the produced difference and the stored difference. The control values $\mathbf{u}(k)$ are stored as an output weight through the training procedure of Kohonen's algorithm [5]. In the case of MIMO systems the data can be concatenated in the same vector. The use of SOMs to cluster concatenated sequential data has first been presented in Ref. [9]. A scalar reward function that determines the association of states to actions can be defined as

$$R(k) = -(\mathbf{y}(k) - \mathbf{y}_d)^{\mathrm{T}}\mathbf{Q}(\mathbf{y}(k) - \mathbf{y}_d), \tag{4}$$

where with $\mathbf{y}(k)$ the vector of current (at $t = k$) output measurements is defined, and with $\mathbf{y}_d$ the vector of current target output values. The weighing matix $\mathbf{Q}$ is a positive definite matrix. The weighing matrix is used to drive the clustering of the prototype vectors based on amplifying or deminishing the effect of certain components of the clustered vectors. The exact form of this weighing matrix depends on the importance and the constraints that have to be imposed on certain components. In the current paper it has been assumed that all the components have the same importance, therefore a unity matrix has been used. A policy can be based on this reward function by calculating the difference between two consecutive values of the reward function:

$$\Delta R = R(k) - R(k-1). \tag{5}$$

In Fig. 2 the evaluation is performed by the look-up table block and produces a firmness signal that modifies the state quantizer. It is clear that every increase of ΔR is desirable, since the maximum target value of the reward function (R) is zero. However, a maximization of R over a

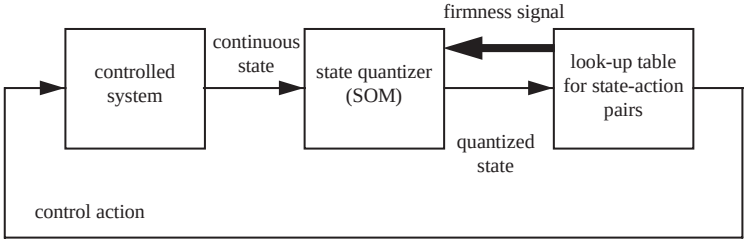

Fig. 2. Scheme of learning problem of combined controller and partitioner learning.

number of steps is better because temporary variations of the reward function can be due to disturbances and not caused by the control sequence. For this reason every neuron stores a moving average of the increase ΔR of the reward function R. This moving average is denoted as $\langle \Delta R \rangle$.

In the presented method when a neuron is selected and a control action produces an increase over the stored moving average for this specific neuron, a positive reinforcement signal is produced. A positive reinforcement signal indicates that this neuron and its immediate neighbours are allowed to learn the state–action pair that has led to the positive reinforcement signal. But since only increases of the reward function that are greater than the stored moving average for each neuron lead to learning, a continuous improvement of the partitioner and the look-up table of control–action pairs is achieved. In the discrete time situation, the moving average of the increases of the reward function can be obtained as

$$\langle \Delta R \rangle_k = \langle \Delta R \rangle_{k-1} + \gamma(\Delta R_k - \langle \Delta R \rangle_{k-1}), \tag{6}$$

where the subscript k denotes the time step $t = k$ and γ is a small positive constant. Note that the momentary value of the increase is denoted without brackets. After the update, the new moving average is stored in the neuron that has been activated. The whole concept has to do with supplying to each neuron a bias term to avoid overtraining. The learning algorithm for the input and output weights is derived from the original Kohonen algorithm [6] based on the competitive selection of the most proximal prototype vector which is called a winner. The updating of the winner and its neighbours is presented in Eqs. (7) and (8) for the input and output weights, respectively. However, the updating due to Eqs. (7) and (8) is performed only if the difference of the reward function from Eq. (5) is larger than the locally stored moving average from Eq. (6).

$$\Delta \mathbf{w}_s^{(in)} = \varepsilon h(\mathbf{x} - \mathbf{w}_s^{(in)}), \tag{7}$$

$$\Delta \mathbf{w}_s^{(out)} = \varepsilon' h'(\mathbf{u} - \mathbf{w}_s^{(out)}), \tag{8}$$

where ε, ε' and h, h' are the learning rates and the neighbourhood kernels, respectively. With $\mathbf{w}_s^{(out)}$ the output weight $\mathbf{y}_s$ is denoted. It must be noted that in the updating equations the winning neuron is denoted with $\mathbf{s}$, i.e. the one that has the smallest euclidean distance from the input $\mathbf{x}$. However the updating equations apply to the lattice neighbours of the winning neuron at every updating step. The neighbourhood kernels used have the form of a Gaussian distribution like

$$h = \exp(-\|\mathbf{x} - \mathbf{w}_s\|^2/\sigma^2), \tag{9}$$

where $\|\cdot\|$ denotes the Euclidean norm and σ denotes the variance of the Gaussian distribution.

In either case, the applied control action that is applied, is constructed by two components

$$\mathbf{u}(k) = \mathbf{w}_s^{(out)} - a_s(\mathbf{y}(k) - \mathbf{y}_d), \tag{10}$$

where a_s is a small positive value that should start from a relatively large value at the initial training phase and subsequently reduced slowly to a final small value. This allows rapid improvement at the initial period of training and allows a small margin for adaptation after the partitioner and look-up table has been learned satisfactorily.

The value of control action $\mathbf{u}(k)$ from Eq. (10) is used for updating the output weight of updating Eq. (8) only in case the reinforcement signal is positive. Such updating of the

output weights ensures that the controller improves continuously. The updating equation for the a_s factor is

$$\Delta a_s = \varepsilon'' h''(a - a_s). \tag{11}$$

In Eq. (11), ε'' and h'' are the learning rate and the neighbourhood kernel, respectively. By setting ε'' very small (in the example of the boom it is set equal to 0.005) the "exploration step" will converge to the final value denoted as (a) slowly enough to allow for satisfactory learning of control actions. If the final value (a) is set different than zero some residual plasticity will allow for continuous adaptation of the controller.

4. Flexible boom stabilization

Flexible spray booms are used in the agricultural domain for pesticide application. They usually consist of lightweight beams on which spraying nozzles are mounted. When driving a tractor over a field, the unevenness of the soil causes the flexible boom to vibrate, leading to under- and over-application of pesticides, thus resulting in environmental pollution. Stabilization of flexible spray booms is needed in order to achieve a uniform spraying liquid distribution and avoid environmental damage.

The learning algorithm of Section 2 is applied to the on-line vibration isolation of a 12th order linearized model of a flexible spray boom that has a total length of 12 m tip to tip. The test set-up from which the model has been obtained is shown in Fig. 3. The spray boom has been modelled using finite element modelling which after model reduction resulted in a state-space model.

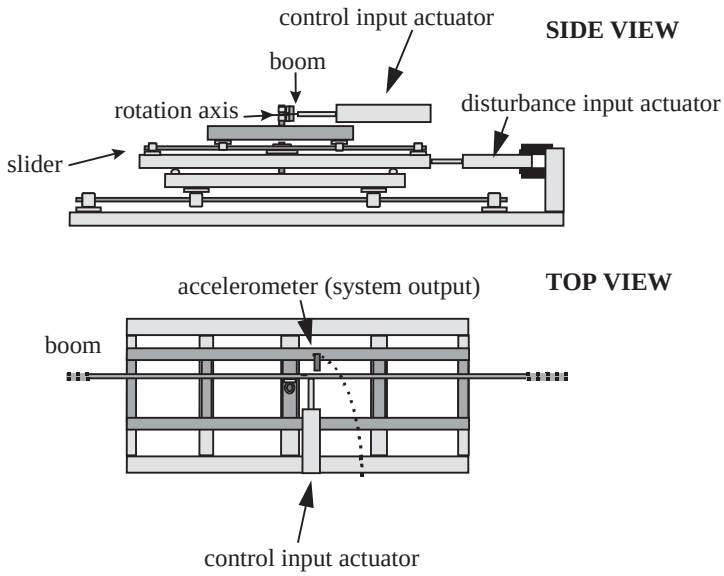

Fig. 3. Set-up for spray-boom measurements (sensor and actuator collocated).

A detailed procedure has been presented in Ref. [10]. The linearized model is of the form

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w},$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + \mathbf{F}\mathbf{v}. \tag{12}$$

The **A** matrix is the state (**x**) transition matrix, **B** is the input (**u**) distribution matrix, **C** is the output matrix, **D** is the direct feed matrix of the input (**u**), **E** is the disturbance input (**w**) distribution matrix and **F** is the sensor noise (**v**) distribution matrix.

For the simulations that are presented, only translational motion in the horizontal plane of the flexible boom are considered, thus resulting in a SISO system. In Fig. 3 the horizontal acceleration in translational motion is measured with the accelerometer. The suppression of horizontal flexible deformations has not been tackled yet successfully, while, for the vertical vibrations a passive suspension usually suffices.

The disturbances used are the accelerations resulting from a standardised field track [11] fed through a model of the tractor wheels and a model of a tractor on which the spray boom has been attached. The excitation signal runs for 23 s and is shown in Fig. 4. The tractor was supposed, in the model, to run with a constant speed of 5 km/h.

The excitation signal of Fig. 4 is used as an input to the system. The uncontrolled response of the system is shown in Fig. 5.

The sensor (accelerometer) and electro-hydraulic actuator are supposed to be collocated at 0.25 m from the connection joint of the flexible boom. The input of the network consists of vectors of previous input and output values of the system determined through a sliding time window of a certain length:

$$\mathbf{x} = (y_d, y(k), \ldots, y(k-n), u(k-1), \ldots, u(k-m))^{\mathrm{T}}, \quad n = 2,\ m = 2. \tag{13}$$

Two delayed values have been used for the actuator (input) and the sensor (output) which are both accelerations which means that $n = 2$ and $m = 2$. The SOM with 100 nodes was trained for the whole period of the test signal (23 s).

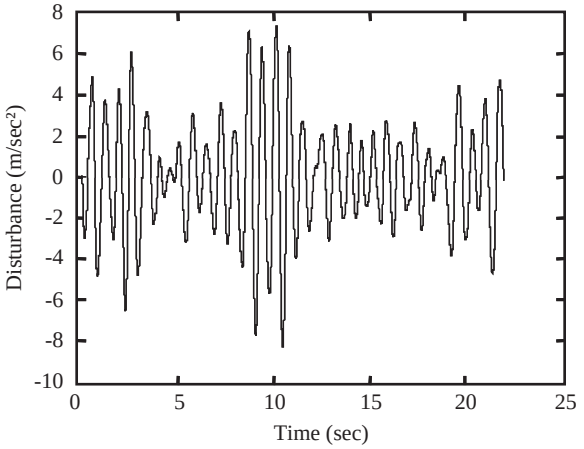

Fig. 4. The acceleration profile of the standardized track used as input to the flexible boom.

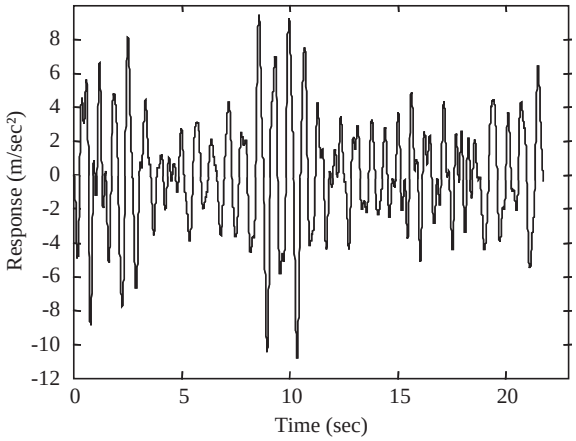

Fig. 5. The response of the system when excited by a standardized track.

The entire systematic procedure of Section 2 has been followed according to a certain series of steps described below:

Step 1: Present an input to the SOM constructed from concatenated input, output data and the target output value. The input is constructed based on Eq. (3). Here y_d is scalar and it represents the measured controlled acceleration. The distances of all prototype vectors from the presented input are calculated. The most proximal prototype is selected as a winning node and becomes activated.

Step 2: The activated neuron has stored an input weight denoted $\mathbf{w}_s^{(in)}$ and an output weight denoted $\mathbf{w}_s^{(out)}$ which have been initialized with random small values. The control action is determined partly from the output weight and partly from the current output measurement (a_s is a small positive step) based on Eq. (10).

Step 3: The reward (R) function for $t=k$ is calculated based on the current output acceleration measurement and also the increase of the reward function calculated based on Eq. (5).

$$R(k) = -(y(k) - y_d)^2. \tag{14}$$

Step 4: Every neuron stores a moving average of the increases of the reward function calculated based on Eq. (6).

Here a value of γ equal to 0.2 has been used. Note that this value is stored in the neuron only when it is activated.

Step 5: A positive reinforcement signal is produced only when

$$\Delta R_k > \langle \Delta R \rangle_{k-1}. \tag{15}$$

Step 6: Given the positive reinforcement the input and output weights of the neuron and its neighbours are updated according to the updating equations:

$$\Delta \mathbf{w}_s^{(in)} = \varepsilon h(\mathbf{x} - \mathbf{w}_s^{(in)}), \tag{16}$$

$$\Delta \mathbf{w}_s^{(out)} = \varepsilon' h'(\mathbf{u} - \mathbf{w}_s^{(out)}). \tag{17}$$

Here with u the control action at $t = k$ is denoted. If the condition that leads to positive reinforcement does not hold, no updating occurs. Thus the SOM learns only from successful sensor/actuator pairs. All the weights are initialized to small random values.

Step 7: The weighting parameter a_s is updated at all times as follows (initial value for a_s was set equal to a value of 10^{-3} and final value equal to 10^{-4}, while ε'' was set equal to 0.005):

$$\Delta a_s = \varepsilon'' h''(a - a_s). \tag{18}$$

Step 8: Go back to step 1 to present a new input.

The initial and final settings of the learning parameters, namely the learning rates ε, and the widths σ of the gaussian kernels h for the updating equations were chosen to be $\varepsilon_I = \varepsilon'_I = \varepsilon''_I = 0.9$ (initial), $\varepsilon_f = \varepsilon'_f = \varepsilon''_f = 0.05$ (final), $\sigma_I = \sigma'_I = \sigma''_I = 0.6 \times$ (number of units in one dimension of the map) (initial), $\sigma_f = \sigma'_f = \sigma''_f = 0.3 \times$ (number of units in one dimension of the map) (final). The values of these parameters were chosen to decrease exponentially with time between the initial and the final value.

The result of following the above on-line learning of control actions is shown in Fig. 6. Both are accelerations at the point where the actuators are collocated with the sensors. From Fig. 7 it is evident that the peaks have been reduced by at least 20 dB.

The evolution of the reinforcement signal through the first sampling steps (at a sampling rate of 1 kHz) is shown in Fig. 8. It is clear that during the first 0.2 s the controller learns most of the time, thus resulting in a very small acceleration from the very beginning of the training session.

A way of visualizing the spatial structure of the representative vectors that the SOM has stored is by plotting these vectors in the case that they are also two-dimensional like the SOM itself. In the case of a higher dimension of the input data the geometrical relations of the representative vectors are difficult to visualize.

As is evident from Fig. 9, in which the first two weights of the map are plotted, there is a very clear ordering at the end of learning. These two weights are representative values of two

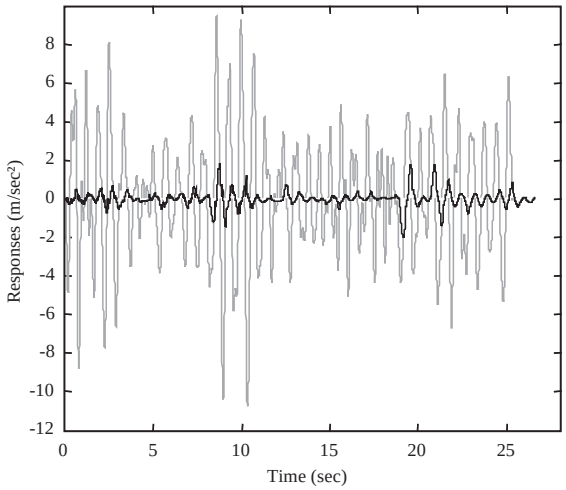

Fig. 6. The controlled versus the uncontrolled time response of the system (the thick line represents the controlled system response).

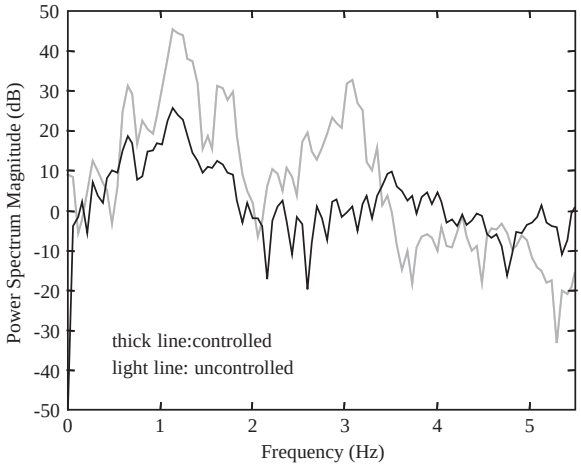

Fig. 7. The controlled versus the uncontrolled frequency response of the system.

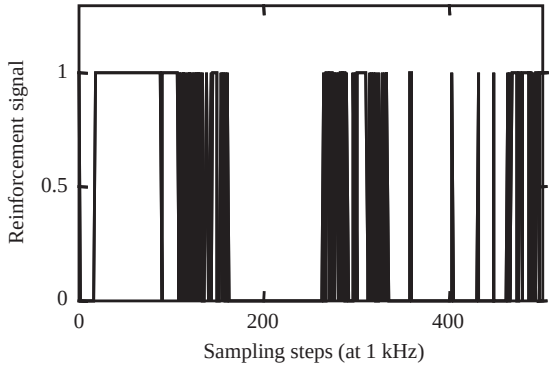

Fig. 8. The reinforcement signal during the initial half second of the training session.

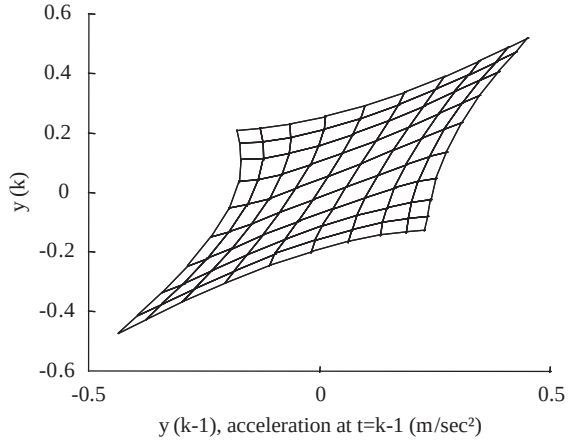

Fig. 9. The SOM at the end of the learning period.

consecutive controlled acceleration values that have emerged through the learning process of the SOM. It has to be mentioned that during learning the SOM tends to represent the states that occur more frequently. However, because of the bias that is introduced through the moving average of the reward function increase in the updating policy (*Step* 5) the states that are visited tend to be equiprobable.

An important aspect of the final stage of the SOM is that the states around the diagonal cover a wider range. This follows from the cooling schedule of the updating equations (*Step* 6); i.e., the learning rate assumes a very small final value.

5. Conclusions

A new neural network method for disturbance suppression of dynamical systems has been presented. The main advantage of this method is the local representation of the controller and state partitioner, which are learned simultaneously by reward and failure signals. The whole learning scheme does not need any prior information but only output measurements of the controlled system. Local updating algorithms assure much faster convergence than global updating algorithms. The method is generally applicable from the point of view that it is not based on a model of the system under control. It only relies on a reward function and the moving average of locally stored rewards over time. It can be used equally well for on-line control of linear and non-linear systems or systems with changing parameters. The new method presented can be applied in the automotive (vehicle suspensions) and the aerospace domain (flexible space structures), in the case of systems with uncertain or complex dynamic behaviour.

References

[1] H. Ritter, T. Martinetz, K. Schulten, *Neural Computation and Self-Organizing Maps: An Introduction*, Addison-Wesley, New York, 1992.
[2] M. Hermann, R. Der, Efficient Q-learning by division of labour, in: *Proceedings of ICANN '95*, Vol. 2, EC2, Paris, 1995, p. 129.
[3] S. Das, M.C. Mozer, Aunified gradient-descent/clustering architecture for finite state machine induction, in: *Advances in NIPS*, Vol. 6, Morgan Kaufmann, San Mateo, CA, 1994, pp. 19–26.
[4] H. Ritter, K. Schulten, On the stationary state of Kohonen's self-organizing sensory mapping, *Biological Cybernetics* 54 (1986) 99–106.
[5] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, Heidelberg, 1995.
[6] T. Kohonen, Self-organized formation of topologically correct feature maps, *Biological Cybernetics* 43 (1982) 59–69.
[7] D.A. White, D.A. Sofge, *Handbook of Intelligent Control, Neural, Fuzzy and Adaptive Approaches*, Van Nostrand Reinhold, New York, 1992.
[8] D. Bertsekas, J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
[9] J. Kangas, Time-delayed self-organizing maps, in: *Proceedings of the IEEE IJCNN-90 Conference*, Vol. 2, San Diego, CA, 1990, pp. 331–336.
[10] H. Ramon, A Design Procedure for Modern Control Algorithms on Agricultural Machinery Applied to Active Vibration Control of a Spray Boom, Ph.D. Thesis, Nr. 213, Department of Agro-Engineering and Economics, Katholieke Universiteit Leuven, Belgium, 1993.
[11] Norme Internationale, Organisation Internationale de Normalisation ISO 5008(F), Tracteurs et matériels agricoles à roues—mesurage des vibrations transmisses, globalement au conducteur, 1979.