Title: New Approaches of Process Visualization and Analysis in Power Plants

Authors: Otte, Ralf (ABB Kraftwerksleittechnik, Product Management);
Goser, Karl (University of Dortmund, Institute for Electronic Components)

Introduction

In order to determine the ongoing status of an entire power plant or one of its components, the procedure generally adopted is to examine the process variables involved. In most process analysis systems, however, these process variables are dealt with in isolation from each other. One very simple and widely used variant for process monitoring, for instance, is to incorporate alarm limits for particularly important process variables. If these alarm limits are violated, the monitoring system will generate an alarm message of predefined priority. This is one of the simplest options for detecting limit-value violations at important signals.

It has, however, emerged that the use of static alarm limits may have crucial disadvantages, since there are numerous equipment states (e.g. transient conditions) where the alarm limits may indeed be violated, without any actual fault having occurred in the plant or component concerned. On the other hand, there are also particularly critical process states where tighter limits would be necessary for certain signals. These more complicated cases need to be analysed with dynamic alarm limits.

Sophisticated analytic and diagnostic tools are able to compute precisely these dynamic alarm barriers or tolerance bands. In dependence on the ongoing load points involved, an anticipated value is estimated for individual process variables, and the currently ongoing actual value then compared with this reference value. This means that the relevant process values can be evaluated *in the context of the ongoing process status*, and any drifting of a process variable can be detected without the process values actually having violated the fixed static alarm barriers at all, see /1/,/2/ and /3/.

But what about process changes where a large number of process variables have moved only slightly away from their setpoint status ? Each signal deviation is so small in itself that it does not reach either a static or dynamic limit value; taken in combination, however, simultaneous small deviations from the ideal value in several hundred process variables may indeed constitute a change in the process status. This form of process change, where the process is considered holistically in terms of process-variable combinations, cannot be detected with the methods outlined above.

But in power plant operations there are indeed numerous process states characterized precisely by these mutual interdependences of the basic process variables (e.g. ageing processes, soiling); this paper accordingly aims to indicate approaches for holistic process evaluation.

The objective is thus to evaluate all the relevant process variables at a power plant in their overall context, precisely in order to allow for the mutual feedback among them. This form of process analysis is to be based on Artificial Neural Networks (ANNs). In this paper a description will be given of an actual implementation option featuring Self-Organizing Maps (SOMs) for a new form of process analysis, which could be used as a preliminary stage for transient state analysis in power plant operations.

Process analysis using SOM

It is generally known from the literature that self-organizing maps can be used to approximate a specified data distribution in a high-dimensional space by means of a two-dimensional surface, see /4/ and /10/. When the high-dimensional data points are then projected by means of a suitable, topology-retaining procedure onto this unfolded surface, the surface concerned can be interpreted as a window into the high-dimensional space, since the topological connectivities in the high-dimensional space are imaged on the map. This is a very simple option for detecting and evaluating structures in the data, e.g. the number and size of data clusters.

Since evolving the map in the input data space is a non-linear method, it assures general applicability, even for very difficult data distributions, not being subject to any linear restrictions (such as principle component analysis), and thus constitutes a fundamental expansion of previous procedures for data evaluation in power plant applications. However, the map's position and extension no longer correspond to any actual physical values, and if a retranslation becomes necessary, it will have to be implemented by means of additional procedures.

Process visualization with SOM

When real process values are used as input variables for the SOM, the map will, so to speak, unfold in the plant's phase space, and the procedure can be utilized for visualization and analysis of concrete plant states. The process state computed by the algorithm is applied to this topology-retaining map of potential process states, and can then be directly compared to other (e.g. previous) process states.

This task is executed in several processing steps, which will be briefly presented below, as *exemplified by analysis of a power plant process* with 40 evaluated process variables:

1. Acquiring and selecting the process variables

Data acquisition and editing. The number of process variables required can be reduced by correlation studies. In the example here, the number could be reduced from 40 to 11 values.

As the result the following process variables were selected for analysis of boiler states:

- temperature, pressure, quantity of feedwater before boiler inlet (3 variables)
- temperature, pressure, quantity of main steam after boiler outlet (3 variables)
- rpm of the coal feeders (2 variables)
- current setpoint value in MW of the power plant unit (0 ... 145 MW) (1 variable)
- temperature of flue gas, concentration emission value of NOx in the flue gas downstream of boiler (2 variables)

This means that 11 process signals in all are evaluated and analysed.

2. Learning phase of the map (off-line)

2.1. Unfolding of the self-organizing neural network in the plant's phase space, based on the algorithm developed by T. Kohonen, see /4/ and /5/.

The physical process values of the above-mentioned process variables are used as input variables for the network.

2.2. Depiction of the unfolded network as a map.

2.3. Projection of the high-dimensional plant states onto the neural map, on the basis of a "winner-takes-all" algorithm.

One of the options dealt with in the literature is to visualize the clusters on the map by means of a *UMatrix*, see /6/. Here, the differences in weighting vectors between a neuron and its neighbours are computed and after graphical editing displayed correspondingly (e.g. colour-coded or as mountains). Coherent areas (valleys) have a small difference, since their neurons have been placed close together in the phase space, and between these individual clusters there are boundaries identified by a high difference between the weighting vectors involved (mountains). This ensures good visualization of the cluster boundaries (plant states). In actual use, however, it has emerged that this form of visualization is frequently insufficient to show the size and position of the individual plant states in detail, which is why a different approach has been chosen here: to project the individual data vectors of the input space onto the unfolded map by an additional processing step.

In this phase, the Kohonen map is no longer learning; its weighting distribution has been frozen. The projection is created using a "winner-takes-all" algorithm, by confronting the map with a randomly selected data vector from the input space and incrementing the winner rate of the neuron concerned by 1, see algorithm (1).

(1) a_r = winnerTakesAll(**v**)
if a_r == 1
$winRate_r = winRate_r + 1$
end
where a_r .... activity of Neuron r

The current winner rate of each neuron is colour-coded on the display; in the illustrations here, an increase in the winner rate means a colour shift towards "lighter". This process is repeated until the individual clusters (light) stand out on the (dark) map. For the projection, precisely the same data records are used as for the map's learning (unfolding) process.

Figure 1 shows an example of a structured, colour-coded neural map for different boiler states (the light coherent areas). It depicts the projection and visualization of several (11-dimensional) boiler load states on a neural map with 15*20 neurons. The current load for the clusters involved has been entered in the picture by hand.

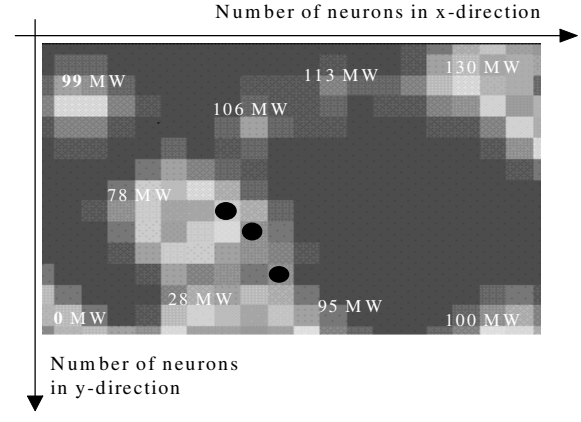

Fig. 1: Projection and visualization of several different boiler load states on an SOM with 15*20 neurons

The dark areas on the map form the boundaries between the different plant states, since they represent neurons which during dynamic visualization with the "winner-takes-all" algorithm have won only very infrequently or not at all, i.e. their winner rates are near to ZERO. High winner rates, as already mentioned, are identified by a colour shift (towards "lighter" in the black-and-white display). This means that the light clusters on the map correspond to frequently occurring real states of the plant equipment! With this approach a light-coloured channel through the dark map can be created and this channel stresses the 'good' range of trajectories, compare Figure 7.

3. Application phase of the map (off-line / on-line)

3.1. Linking the map to the process and depicting the ongoing process state on the structured map.

In the application phase, i.e. while the power plant is actually in operation, the ongoing process state is evaluated by a neural "winner-takes-all" algorithm and applied to the pre-structured and colour-coded map (see Figure 1, where the three black circles correspond to three equipment states at different junctures). This enables the ongoing (generally high-dimensional) plant status to be visualized in the context of other equipment states. The spatial position of the ongoing process state on the structured map provides information on the plant's current state in the reality.

3.2. Graphical connection of winner neurons

In the application phase, the current winner neuron at juncture t_0, for example, is marked by a black circle (cf. Figure 1). When these winner neurons are connected together, the structured map will develop over time t ($t_0 < t_1 < t_2$) a trajectory of the ongoing process states, which will either grow continually or migrate on the map with a fixed and specified length. The Figure 2 shows a start-up routine and a load change over 250 minutes, both recorded using this procedure.

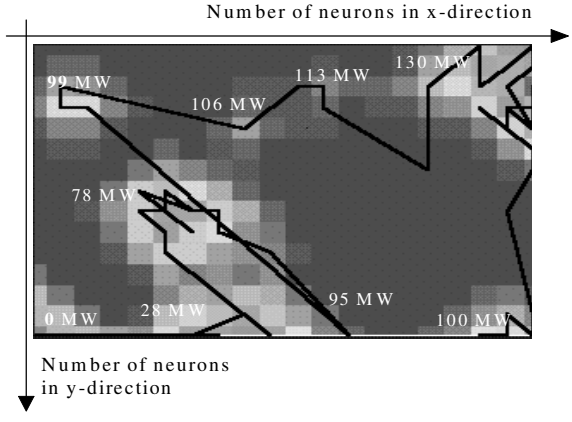

Fig. 2: Recording of a boiler start-up routine on the neural map over a time period of 250 minutes

3.3. Superimposition of trajectories from different process operations for visual comparison and analysis of complex dynamic operations

Since several trajectories can be superimposed on each other on a single map, any desired process operation can be visually compared with a preceding reference operation. This enables us to compare not only individual process states, but also complex process operations with each other; any deviations are very quickly detected. We thus have a holistic, visual fault detection option even for dynamic power plant operations. Figure 3 shows an example with two different boiler runs, with the black trajectory corresponding to a 'good' reference start-up routine, and the white trajectory here representing a load change from 126 MW to 95 MW.

This provides a process observer and operator with an option for intuitively comprehending even highly complex processes, and to observe the effects of his/her interventions in the process directly. Since the procedure has *real-time* capability in the application phase, this form of process visualization can be used to support on-line process control.

The advantage of the procedure described here is that it can not only compare different *process states*, but even record and contrast complex and transient *process operations* (!!) like complete start-up and shut-down routines, qualifying it as a powerful supplementary tool for the monitoring systems commonly used in power plant applications.

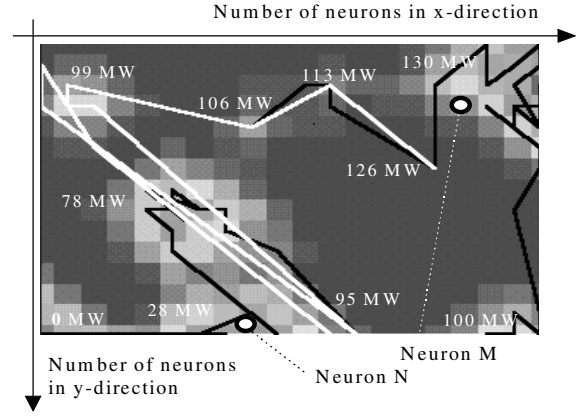

Fig. 3: Projection and comparison of a load change (white) with a reference load change or start-up (black)

Process diagnosis with SOM

Localizing deviations

Once deviations have been recognized on the map between trajectory courses, the crucial question then arises of the reason involved is that the plant operator essentially wants to know which process variables have *caused* the deviations on the map. In order to ascertain this, the weighting distribution of the individual neurons can be analysed, since the real values of the process variables have been encoded in the weightings. Though on this level no absolute data on the real process values can be expected, but the *relative* differences can be determined quite well from the weighting distributions.

To illustrate this, two particularly different weighting distributions of two neurons will be picked out: Neuron 'N' is placed virtually in the 28 MW state in the plant space, while Neuron 'M' responds to a load state of 130 MW; its spatial position on the map is marked in Figure 3 as a white circle. Figure 4 depicts the weighting distribution of these two map neurons.

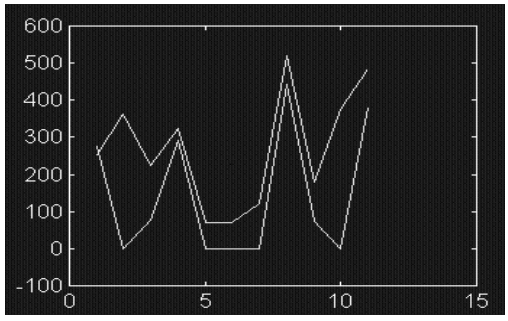

Fig. 4: Depiction of the weighting distribution of two neurons on the SOM

The x-axis in Figure 4 corresponds to the weightings $w_1,w_2,..,w_{11}$ of neurons 'N' and 'M', while the y-axis represents the scaled values of the individual weightings concerned. It can be seen that the neurons differ primarily in their weightings w_{10}, w_2 and w_3, and this is confirmed by computing the standard deviation.

These indices (10,3,2) can now be unambiguously mapped onto the process values involved. In the example here, the differences lie in the following process variables:

- main steam quantity at boiler outlet
- feedwater quantity at boiler inlet
- feedwater pressure at boiler inlet

By continuously computing the *standard deviation* for the weighting distribution of (operator-)selected neurons, the process analysis can be automated. The operator can thus ascertain the cause of any two deviations on-line, since he/she need only "click" those neurons on the map where the trajectories exhibit particular (visual) deviation. The cause of the deviations (i.e. the process variables with the largest standard deviations) are computed automatically. That means the algorithm finds out these process variables which cause the deviation. The map thus enables simple diagnostic options to be implemented.

Data mining: expanding process knowledge

Another possible application for the structured map lies in the options it offers extending into the realm of *data mining*, since this map can be used to postulate new hypotheses (knowledge) on process inter-relationships, though since this cannot be automated it does not qualify as true data mining. The idea is to depict individual weighting distributions over the entire map, so as to discover any new types of causal relationship. Figures 5 and 6 illustrate the principle involved.

As can be imagined, comparing these distributions will not only reveal mutual correlations (these could also be discovered using the histories of the real process values), but will also help new hypotheses to emerge on the various plant states involved when these distributions are compared with the states on the map, e.g. process state P1 is characterized by the fact that process variable1 is in the range B1 and process variable2 in the range B2.

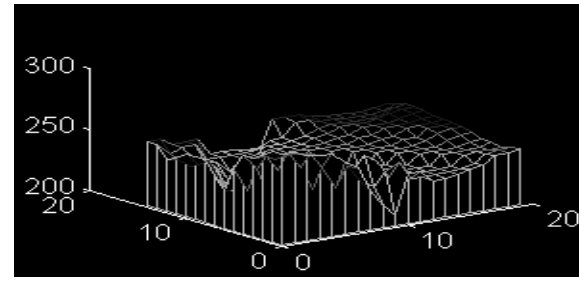

Fig. 5: Distribution of feedwater temperature (z-axis) over the map (x,y-axis)

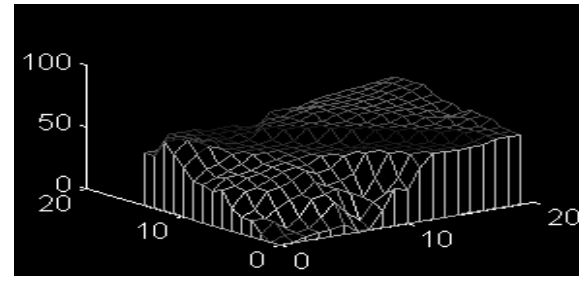

Fig. 6: Distribution of coal feeder rpm (z-axis) over the map (x,y-axis)

Further research

Automatic fault detection

So far, the deviation between two trajectories has been determined only by visual means, i.e. the operator has to keep the map under observation while the ongoing process proceeds. In order to automate this error dedection (in on-line or off-line mode), mainly five approaches are possible:

(A) Defining a tolerance band for 'good' trajectories on the SOM and comparing the position of current trajectory with this tolerance band (channel).

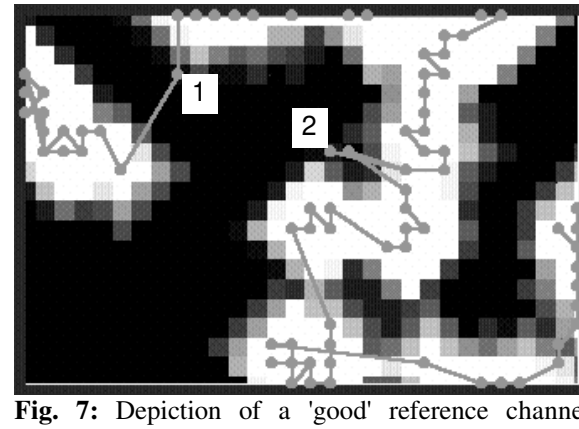

Fig. 7: Depiction of a 'good' reference channel (white) for the evaluation of process operation trajectories (gray line)

In Figure 7, these tolerance band is depicted as white channel. Whenever the current trajectory exceeds

this tolerance band an error message will be provided, see points (1) and (2) in Figure 7. The point (2) represents a really deviation in the transient process comparing with the reference process.

This approaches can be realized by using the algorithm of the projection SOM because the channel of 'good' references is a result of the projected SOM, see algorithm (1).

Advantage:
- Easy to understand and to implement
- Usable for the latest process value (latest trajectory point)
- Usable for on-line analysis

Disadvantage:
- Some wrong error dedection can be made, e.g. the error dedection 1 (point (1) in Figure 7) is not an really error

(B) Calculcation of 'sliding' correlation coefficient between a 'good' reference trajectory and the current trajectory

Advantage:
- Easy to understand and to implement
- Usable for the latest process value (latest trajectory point)
- Usable for on-line analysis

Disadvantage:
- Trajectories must have same length
- Impossible to compare only parts of trajectories like two different load changes

(C) Another option for evaluating the similarity of trajectories (though space precludes a detailed treatment here) is to establish hierarchical SOMs.
A downstream Supervisor-SOM (K2) 'observes' the trajectories (T) on the preceding process SOM (K1), and evaluates them, Figure 8 shows the framework.

Advantage:
- Easy to understand and to implement
- Also parts of trajectories can be compared

Disadvantage:
- Analysis possible only for a fixed part of process values (trajectory points), not for the latest value
- Examples of training data (trajectories on K1) for SOM-K2 are necessary
- All parts of trajectories must have the same length (training vector vor SOM-K2)

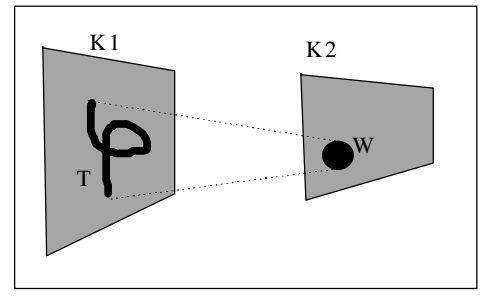

Fig. 8: Framework for hierarchical SOM

(D) Using a self-organizing algorithmus for calculation the similarity of curves. In this approach the number of iterations, which are necessary to map on trajectory to another, will be counted. This number is a measurement for the searched similarity.

Advantage:
- Easy to understand and to implement
- Location and shape of the trajectories will be taken into account in a easy way

Disadvantage:
- Usable for off-line analysis only

(E) Building and using a lateral Self-Organizing Map (LSOM) for calculation of the similarity of trajectories

Because this approach seems to be high-sophisticated and powerful it will be described to the end of this paper.

Automatic fault detection with lateral SOM

A lateral Self-Organizing Map (LSOM) has been developed, with the aim of detecting the *similarity of trajectories* automatically. In mathematical terms, this is not a trivial problem, because it involves computing *segmental similarities of curves*, since the beginning and end points of the trajectories may have been acquired at different times and load points. It is accordingly impossible to compare the trajectory positions of two curves on the map with each other, starting with Index1 of each curve. The curve segments involved first have to be brought into the right position in relation to each other. The following procedure is proposed for this task:

1. Acquire the reference trajectory t_r, e.g. of a "good" start-up routine
2. Store the reference trajectory on the SOM, i.e. "burn in" the trajectory onto the SOM
3. Acquire the operation to be studied t_u, e.g. a load change or another start-up
4. Automatic comparison of the two curve segments for shape and similarity of position
5. Generate an error message in dependence on the similarity analysis

Points 1. and 3. have already been discussed, Point 5. is trivial, so only a brief description of Points 2. and 4. will be provided here.

Creating a lateral SOM (LSOM)

The idea found in the literature of a *Cellular Neural Network* (see /7/ and /8/) is used to store the reference trajectory t_r on the SOM, i.e. the acquired trajectory is encoded in lateral connections between the SOM's neurons (here also called cells). We thus obtain a *combined model of SOM and CNN, in this paper we'll call it LSOM, that means lateral SOM.*

This LSOM is used for storing and evaluating process images by calculation of segmental similarities of trajectories. The reason for this lies in the subsequent option for operating the LSOM as a *cellular automaton* (CA), so as to automate very simple forms of image processing (analysing the similarity of trajectories), i.e. the process images to be handled by the LSOM are the trajectories themselves. The CNN's capabilities for image processing are sufficiently known from the literature, see e.g. /11/. In the application under discussion here, the SOM is transformed into a special CNN (asymmetrical and unhomogeneous), and the trajectories are interpreted as "images". This means that results from the CNN image processing can be directly used for further processing (e.g. stability studies for the learning and operational phases).

Learning phase: storing the reference trajectory $\mathbf{t_r}$

In contrast to frequently published CNN algorithms, the algorithm used here for the **learning** phase is one which produces an unhomogeneous CNN. This algorithm has been slightly modified in broad conformity with those presented in /9/, so as to achieve good weighting differences in the lateral connections between the individual cells (neurons on the SOM); the learning step for establishing the lateral connections between the neurons in the form of a simplified MATLAB implementation is formulated below (no scaling, etc.):

```
For i = 1:MaximumNeuronOn Map
    for j = 1:NeighbourNeuronOfNeuronR
       dc = lr*(a(r)*a(nb)+ sAct/ko * ( a(r) - a(nb) ) )
       c(r,nb) = c(r,nb) + dc;
    end;
end;
```

with the following meanings:
c ... Connection matrix of the lateral connections
between Neuron r and its neighbours
c(r,nb).. Connection from neighbour nb to neuron r
lr ... Learning rate,
sAct .. Sum activity of the map
ko . . Coefficient for setting the weighting differences
a(r) ... Activity of Neuron r
a(nb) . Activity of Neighbour Neurons of r

Since the main loop runs over all the map's neurons (cells), we thus obtain a connection matrix for each individual cell of the lateral map. It is important that the activity **a** of the map originates from the trajectories t_r , i.e. every time a trajectory point is marked for neuron k on the map, simultaneously an internal activity value is set for the activity a_k of this neuron and a smaller value is set to the neighbours of neuron k, e.g. $a_{neighbours} = a_k * 1/\text{distance}$.

Figures 9 depict the lateral weighting matrices C as connections between the cells for two different parameters of the diffusion neighbourhood. In particular, these diagrams can also be compared with Figure 7, since it is precisely the trajectory depicted there which has been stored in the lateral weighting matrix of the cellular map.

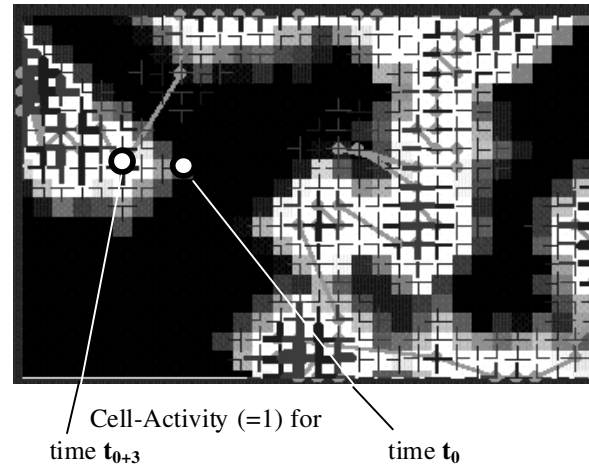

Fig. 9: Depict of lateral weighting matrix built with reference trajectory t_r (gray line within white channel)

Operational phase: evaluating the test trajectory $\mathbf{t_u}$

In the **operational phase** of a LSOM, a cell's activity state are computed in non-linear mode, with a cell's activity being shifted along its maximum lateral connection, coded as c(r,targetCell). The following excerpt from a MATLAB implementation illustrates the basic idea involved:

```
For i = 1:MaximumNeuronOn Map
   if a(i) != 0
     % Search maximum connection to cell i
     targetCell = CellWithMaxConnectionTo(i);
     % Move activity to target cell
      a((targetCell) = a(i);
    end;
end;
```

where targetCell ... neighbourhood cell to Cell r with greatest lateral connection to r

Iterative application of this algorithm will cause the vertices of trajectory t_u in the operational phase of the LSOM to migrate along their maximum connections, provided (in ideal cases) they have arrived in the stored vertices of the reference trajectory t_r (Figure 9 shows migration of a single activity to two selected junctures t_0 and t_{0+3}). The number of necessary (parallel) iterations for all activities (in Fig. 9 the number of iterations for one trajectory point is 3) can now be used as a

coefficient of measure for the similarity of the two curve segments.

The procedure described here was implemented in MATLAB, though in actual use it had to be modified; in particular, provision had to be made for cases where the migrating vertices do not engage into the reference points, but "get stuck" somewhere else on the map.

Further studies must be made to show whether the sensitivity of these iterations is sufficient to actually detect all relevant process deviations. And a next point, the behaviour in the operational phase must be studied, particularly for cases in which the trajectory points are close together.

Summary

The article describes a non-linear, generally applicable method for analysing complex, multi-dimensional process states in a power plant by means of Self-Organizing Maps. This method of process analysis thus extends current methods for process monitoring, since it is generally applicable, and no longer merely examines each process variable in isolation, but evaluates their interrelationships as an integrated whole, as well as providing the appropriate display options. The procedure can be used both for static (steady-state) and for dynamic (transient) power plant phenomena.

Literature

/1/ D. Neupert, M.Schlee, E. Simon: Wissensbasierte Prozeßführung im Kraftwerk Staudinger in VGB Kraftwerkstechnik 74 (1994), Heft 3

/2/ A. Zell: Simulation Neuronaler Netze, Addison-Wesley Verlag, 1994

/3/ S. Hafner: Neuronale Netze in der Automatisierungstechnik, Oldenbourg Verlag, 1994

/4/ T. Kohonen: Self-organizing of topologically correct feature maps in Biological Cybernetics, 43:59-69, 1982

/5/ T. Kohonen: The self-organizing maps in Proc. of the IEEE, 78(9):1464-1480, 1990

/6/ G. Whittington, C. Spracklen: The Application of Neural Network Model to Sensor Data Fusion in Proc. of Applications of ANN, Orlando, USA, 1990

/7/ L. Chua, L. Yang: Cellular Neural Networks, Theory and Application in IEEE Transactions on Circuits and Systems. Vol. 35, no. 10, 1988

/8/ V. Cimagalli, M. Balsi: Cellular Neural Networks, a Review in Proc. of Sixth Italian Workshop on Parallel Architectures and ANN, Italy, 1993

/9/ C. Hillermeier et. al.: Topological features maps with self-organizing lateral connections, a population-coded, one-layer model of associative memory in Biological Cybernetics 72, 1994

/10/ K. Goser: Kohonen's map - their application and implementation on microelectronics in Proc. of the ICANN-91, Helsinki, Jun. 1991

/11/ G. Martinelli: Associative Memory Design using space-varying Cellular Neural Networks in Proc. of Second Intern. Workshop on CNNA, Munich, 1992