Structure from Data: AI Approaches to Systems Modeling

Heikki Hyötyniemi
Helsinki University of Technology
Control Engineering Laboratory
Otakaari 5 A, FIN-02150 Espoo, Finland
Phone: +358-9-4513327
E-mail: heikki.hyotyniemi@hut.fi

July 27, 1998

Abstract

The results from *cognitive science* have not yet been widely applied in control engineering. The models of human behavior have been applied only for attacking ergonomical problems in user interfaces of complex automation systems. In this paper, the starting point is that cognitive phenomena should be utilized in systems engineering also on the more fundamental level.

The current status in the field of complex systems modeling is presented, and the problems in the contemporary approaches are discussed. It is claimed that the main problem is the missing view of the measurement data properties.

A model structure is presented that implements the proposed new data view. An example application visualizes how the presented approach can be applied for automatically constructing a model for a practical system, and how that model can be used as a 'soft sensor'.

Keywords. Artificial Intelligence (AI), automation, modeling, large-scale systems, industrial applications

1 Introduction

To a layman, the behavior of a cybernetic, self-controlled system seems intelligent: it is capable of reacting to changes in its environment in an appropriate way. However, there are different levels of 'intelligence' — the more one understands the underlying principles of control theory, the higher are the standards. Traditionally implemented control systems react to measurements in a strictly predefined manner, so that the only truly intelligent task is that of explicitly defining the control rules based on the knowledge of causal dependencies in the system; this is done by a human. Another level of intelligence is found in *adaptive control:* the controller parameters are automatically adjusted to better compensate for the changes in the observed process dynamics. Again, adaptation is a matter of mechanical optimization, and, strictly speaking, no intelligence is still involved. The real challenge is the case where the system structure is not clear; the dependencies between the signals are unknown, and, what is more, *it may not even be clear what one should do with the measurements,* or how to utilize them. It is this last problem that is becoming more and more acute in systems engineering, and AI approaches are needed[1].

In modeling, the phase of structure identification is usually much more difficult than the subsequent parameter estimation phase. Structure identification is usually very much based on heuristics and intuition. That is why, the structuring of a complex environment is done by a human expert: he selects the most important variables and determines the relationships between them. The new problem is the explosion of information; the networked process environments and new measuring devices deliver more or less relevant information at increasing rates — there just is not enough time or human resources to go through all the

[1] As an indication of the changing nature of the control engineering work, the name of the unit 'Säätötekniikan laboratorio' (or Control Engineering Laboratory) of Helsinki University of Technology was lately renamed 'Systeemitekniikan laboratorio' (Systems Engineering Laboratory)

measurement data by hand. There is need for automatic tools that would do the structure identification as well as the parameter estimation with no external supervision.

This paper discusses an approach to realize a general structure for system models to face the future needs. This kind of general environment for modeling tasks must be very flexible, and it is not easy to define a framework fulfilling such specifications. There are various demands that should be met — in this context, we focus on the following ones:

1. model generality,
2. application-orientation,
3. simplicity and easy analyzability,
4. self-adaptation, and
5. understandability.

It seems that many of these items are mutually contradictotry. For example, the *generality* objective means that one must be able to include nonlinearities and structure variations in the model, while the *analyzability* means that the model should be based on linear theory. Good models are tailor-made or applicatio-oriented — how about the generality objective then? To operate autonomously, *self-adaptation* is vital, but if the model is constructed without human intervention, the results are often rather cryptic; however, the model structure must simultaneously be *understandable* or easily interpreted in concrete terms.

These demands are elaborated on in what follows — and, as it will turn out, a rather nice compromise can be reached.

2 About systems engineering

2.1 Role of models

One of the most efficient paradigms of the engineering work is the idea of *reductionism:* a complex problem can better be mastered if it is divided in subtasks. In the control engineering work, this means that the modeling and the controller construction is done separately. To achieve real benefits when using this 'divide and conquer' approach, the critical point is the 'interface' between these subtasks, or the *model* (see Fig. 1). What kind of structure should the model have to bring some added value?

There are different kinds of models for different purposes, and there is not just one 'correct' model structure. A good model makes further applications simple, and different applications benefit from different model properties. As the systems to be modeled are becoming more and more complex, it seems that the role of the model as a tool towards 'understanding' the system behavior is becoming more and more prominent. Controlling is not the only goal of modeling — in many cybernetic systems, there are no control actions available in the first place, and the best one can do is to predict their behavior and be prepared for the future events.

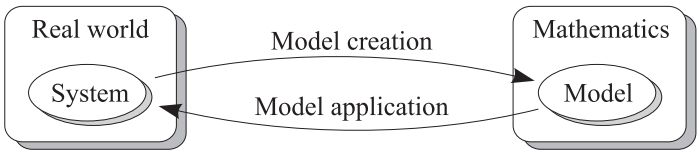

Figure 1: Models are the key to reducing the analysis-synthesis problems in control engineering applications

It sounds natural that the more information you have about a system, the easier it should be to understand and master its behavior. However, there is an interesting paradox here: for example, the controllers with a single input and a single output (SISO) are (often) easily constructed and tuned, while controllers with various inputs are still a frontier research problem — no generally applicable approaches exist. Even if there is more measurement information available, the complexity of the controller construction task becomes more difficult. Usually, in practice, multivariable systems are controlled as a set of independent SISO processes — this approach hardly is optimal.

The starting point in this discussion is the assumption that *the more you receive information, the easier it should be for you to find the model for the system.* Redundancy of data is a smaller problem than the lost pieces of information; efficient tools exist for compressing redundant information.

2.2 Statistical approaches

When there is plenty of data and one wants to extract information from that data automatically, one has to rely on statistical approaches in a 'bottom-up' way. The statistical properties of the measurements, specially the correlations between the data samples, should dictate what the resulting model looks like and how the various observations contribute in the model.

Statistical methods have been extensively utilized in systems analysis — however, it can be said that the development of the statistical approaches has been dictated more by mathematics, not by the actual properties of the physical systems. The brute-force statistical analyses cannot always give the most useful information.

Usually statistical data is modeled using the assumption of Gaussian distribution. For the Gaussian distribution the high-order statistical cumulants are zero, and the best one can do, is to concentrate on the second-order moment, or the variance. This is why, most of the today's modeling approaches concentrate on the error variance minimization. However, even if this approach is simple, it does not necessarily reflect the real nature of the data, and the resulting models are not optimally suited for that data. The objective of explaining only the data variance is the main reason also to the somewhat disappointing results that have been achieved using the Principal Component Analysis (PCA) for feature extraction.

If the measurements consist of independently distributed random variables, the assumption of Gaussian distribution is asymptotically optimal and the best one can do. However, it has been claimed that *originally* natural signals have rather non-Gaussian distribution, and only after being mixed with other signals, the distribution approximates Gaussian. If one is searching for the underlying system structure, one should try to extract the original signals rather than the mixtures. For example, Independent Component Analysis (ICA) extracts signals minimizing (or maximizing) the fourth cumulant of the data distribution, or *kurtosis* [7].

In the case of a complex dynamic system, the original data distribution is normally far from Gaussian: outliers make the 'tails' of the distributions longer, and quantizations introduce uniformly distributed noise — in both of these cases, kurtosis is far from zero, and this information can be utilized when searching for the underlying system structure. Another thing that makes it important to study 'independent components' is that in complex systems the measurements at different time instances may represent different data distributions; this problem of mixed distributions will be discussed more later.

2.3 Current trends

It is perhaps instructive to study briefly the field of contemporary control engineering practice. It seems that many of the theoretically very advanced control engineering approaches never enter practical life. Putting it rather boldly, it can be claimed that this applies to the whole paradigm of so called 'modern control': even if the theories are very powerful, these methods were not widely accepted in industry — the classical PID controller still rules. The new, sophisticated state space methods were perhaps too abstract to be widely accepted.

What happened, instead, was the emergence of 'postmodern' methods: neural networks, fuzzy controllers, etc., were introduced and eagerly adopted in practice. This happened even if there often exist modern methods that would outperform the new, more or less heuristic approaches. The main reason for the current popularity of these *soft computing* methods is perhaps their intuitive appeal — they 'work like the brain', they can carry out complex tasks.

The soft computing paradigms have been there now for quite a long time, and they can be seen in a perspective. They did not solve all of the problems — hopefully, there are lessons to be learned. The problem with the neural networks and fuzzy systems is that implementing just one aspect of human behavior (either the neuronal structure as in the perceptron networks or the fuzziness of the categories as in the fuzzy systems) does not necessarily result in optimal realizations from the holistic point of view. It can be claimed that the view of the actual data structure has not been elaborated on sufficiently in the soft computing approaches.

> For example, one can prove that recurrent perceptron networks do have the computational power of a Turing machine, so that any function can be implemented also in this medium (this fact has been used to motivate their use in complex applications). Perhaps they can, but — take an example from another field: even if a complex program can be implemented using raw machine coding, a better approach is to use higher-level tools. Just as well, a set of individual perceptrons can be used for implementing arbitrary functions, but managing thousands of perceptrons is not simple. Because the perceptron network with feedback can do anything, it takes very much data

to constrain its behavior to what one wants. This problem that becomes more and more acute as the input dimension grows.

The soft computing methods are more or less 'behavioral', that is, they are only constructing a mapping between the inputs and the corresponding responses. These methods are often advertised to be simpler than the traditional ones, because no models are involved. It can be questioned whether this kind of simplicity is really the panacea: what you get is a black-box function, with no intuition of the internal system structure. It can be put rather boldly: the more one studies the highly nonlinear neural network structures, the more general (and 'intelligent') the linear model structures seem to be!

Contrary to the current trends it is now assumed that the *tailored model is the primary goal.* All other tasks are based on the model — 'understanding' the process is vital, and this understanding can facilitate, for example, optimized control actions.

When trying to find a good model for measurement data, we are facing the same problem as our mental machinery — now the huge numbers of data is delivered through sensors rather than senses. How to utilize the principles of cognition in data modeling?

> One of the traditional neural network architectures, the *Kohonen network* applies the principles of cognition for data analysis in a rather smart way: the complexity of a system is transformed into visual form, so that the human pattern recognition capability that is specially powerful in visual image processing tasks can be utilized for finding relationships between the originally non-visual data units. However, now the high-dimensional data has to be projected to a low-dimensional space to be visually represented, and, inevitably, some information is lost. It would be excellent if we could apply the mental processes to the measurement data directly, with no intermediate steps of data visualization. What one would need, is a tailored model structure that can do the same things as the brain does!

To achieve the 'smart' model structure, the properties of the measurement data need to be studied closer. What kind of behavior can be regarded as clever and what cannot, is very much dependent of the environment — intelligence can be defined as an *ability to adapt to an unknown environment.*

2.4 Starting point: data ontology

It is typical that in different operating points different (either quantitatively or qualitatively) dynamic relationships apply and different system variables are needed. The measurement data usually come from various mutually independent distributions — one sub-distribution for each of the operating modes. Different variables are needed to express the variations around the centers of the sub-distributions; modeling just one 'average' distribution does not give good results then (standard statistical methods do this, extracting a single global model rather than a set of local submodels). This means that only a subset of the available variables are needed at any time to present the data in a reasonable way. The data model becomes *sparse.*

It is this sparseness that makes the representations tailor-made and, hopefully, 'smart': qualitative changes in the model take place when different variables are selected. The cut connections mean structural changes in the model. This means that *within the same framework, structurally different models can be represented.* As contrasted with neural networks, it is not the number of connections between the processing units, it is the number of *missing connections* that is the key to achieve good representation for data!

The measurements are distributed so that they create clusters in the data space. To introduce some inner structure within a cluster, the features ('hidden variables') span linear, rather low-dimensional subspaces around the prototype cluster center. Now, the sparsity means that different sets of features may be used as basis vectors to define the subspaces. In this context, the various subspace options are seen as clusters — however, it needs to be noted that these clusters differ from the statistical, visually obvious data-level clusters. It turns out that this kind of a rather involved view of data clusters is useful when modeling dynamic phenomena. To summarize:

> *All measurement data is assumed to be distributed in clusters of subspaces.*

This view of data has been implemented, and the algorithm called GGHA is presented, for example, in [5]. This algorithm explicitly utilizes the assumption of various mixed distributions. The operation of the algorithm can be interpreted also in statistical terms: it is an unsupervised combination of cluster analysis and principal component analysis. It can also be seen as non-orthogonal, sparsely coded factor analysis approach. The algorithm has been designed so that high-dimensional input vectors can be tolerated — this means that one can use all available information that can help in clustering. For example, qualitative (binary) status information can be included in the processed data.

The cluster centers and the subspace axes together are the features that define the signal dependencies in the system. When the GGHA algorithm is applied, all of the features have the same vector representation. The mathematical structure of the model will be discussed later in concrete terms.

2.5 About understandability

The previous section discussed *generality* (within the framework of the assumed data structure), *application-orientation* (the sparse representation makes structural changes possible), *simplicity* (the subspaces were assumed linear), and *self-adaptation* (there exists an algorithm for carrying out the data processing). The last item in our list remains — is the data model *understandable?*

It has been assumed that our mental machinery has adapted to maintaining observation data from the real world in an optimal way. On the other hand, there is some evidence that *all* of our observation data has the same sparse structure of clustered subspaces; it is not only system theoretic applications that should share the same ontological essence. If these assumptions hold, and if the empiristic assumption of no prior data structures in the brain is adopted, it is enough to model the data optimally in the presented framework: *the obtained model then has to reflect the cognitive structures of a human.* If our model structure mimics our own mental structures, the model should be easily interpreted — that is, it should be understandable.

This all can be seen also from the opposite point of view: if something does not easily fit in our understanding, it should not be included in the model. We cannot know anything about the 'reality' behind our observations — should we forget about the reality altogether and only concentrate on our concrete data? This view opens up interesting horizons. As a consequence, one could claim, for example, that the *discrete-time representation of system dynamics is everything one needs:* it is easier to grasp samples than n'th order derivatives of signals; random signals like noise are better understandable in discrete time, etc. It can be argued that the continuous-time representation is only a 'Platonian' idea, beyond the real (human) life.

'If it is difficult to grasp, it cannot be important' — this is an extreme view, and it is specially difficult for paradigms like this to get approved in the field of traditionally very mathematical control engineering[2]. However, to emphasize the fundamental role of intuition in our field, one could take the following interesting definition of a 'system' (a verbose motivation for this definition is given in [4]):

A 'system' is what can be distinguished as a system.

This means that even the very basic concepts in the field of systems engineering are vague and intuitive. The only way to include all possible aspects is to trust connotations and associations — 'understandability' is one of the core premises in systems theory.

3 'Generalized state systems'

3.1 Mathematical framework

In this section, the proposed approach is presented in concrete terms. The system model consists of the collection of the feature vectors, so that when written in the matrix form one has

$$\Theta = \begin{pmatrix} \theta_1 & \cdots & \theta_N \end{pmatrix}. \tag{1}$$

Assume that the number of measurement signals is n, and assume that the system can be described using N features, so that Θ is an $n \times N$ matrix (normally $n \gg N$). This means that the dimension of input vector f of the model is n and the dimension of the 'generalized state vector' ξ is N. The generalized state vector contains the 'loadings' of the various process features at any given time. In statistical terms, these features have close connection to *factors* [2].

The basic structure of the model reveals that it is simply compression of data that is carried out, this means, a low-dimensional state vector stands for the high-dimensional input vector. This matrix formulation is just a framework and it alone does not guarantee that the model is good — the most important thing when aiming at a structured representation of the system is the selection of the feature vectors θ_i. Because of the high dimensionality of the input vector, the feature model is far from minimal, and it would seem that it inevitably becomes numerically involved and difficult to read. Paradoxically enough, the optimized feature model is still *more unique* (and often easier to interpret) than the standard state representation, for example: the key point is the sparsity objective. It turns out that 'maximizing the zeros', so that only

[2]The data is always subjective — what a fiddler's paradise this is: trust your introspection, the truth is how you see it ...!

very few units are active at any time, reveals the underlying structure of the system. The experiments seem to support the hypothesis that system behavior often can be efficiently captured using this kind of feature framework, and, correspondingly, the models become rather compact (see next sections for more concrete motivation).

When the input vector f is given, the corresponding state vector $\xi(f)$ can be found out in various ways using the feature model. To find a sparse state representation, some nonlinear strategy is needed (see [6]). Assume that the set of appropriate features has been selected somehow, so that Θ_f consists of a subset of m features in Θ so that only the *most relevant* features with respect to the input vector f are included. Because the subset of feature vectors spans a linear non-orthogonal basis, the easiest approach to finding the non-zero elements of the generalized state vector is to calculate the best matching feature loadings in the pseudo inverse sense (see [5]) as

$$\xi(f) = (\Theta_f^T W \Theta_f)^{-1} \Theta_f^T W \cdot f. \tag{2}$$

The role of the $n \times n$ non-negative definite (diagonal) weighting matrix W is to emphasize the measurement signals appropriately — the higher a weighting is, the more that signal contributes to the calculation of the state. In fact, W should be selected as the inverse of the measurement error covariance matrix.

There is no fixed output defined in the model structure. This means that no prior distinction between inputs and outputs is necessary — and in many cases it turns out that this flexibility is a valuable property. How the output signals can be calculated then using this model? The key is *associative search*. When the state vector $\xi(f)$ corresponding to the input vector f has been calculated, the input estimate can be calculated simply as

$$\hat{f} = \Theta_f \cdot \xi(f). \tag{3}$$

This means that if the unknown signal values to be determined by the model, usually the outputs, are *not weighted* in the formula (2), the missing values are filled in automatically in the $\hat{f}$ vector so that the weighted signals will have the best possible fit. The model can readily be applied in prediction or filtering applications. Using the formula (2), the linearity of the expression gives us tools for analyzing the reliability of the results: because the invertibility of

$$\Theta_f^T W \Theta_f \tag{4}$$

determines how badly behaving the reconstructed vector is, the condition number of this expression can be used as a measure for the 'observability' of the unknown signals.

There are no state transitions defined in the system structure, but dynamic behaviors can be captured if various time point samples are included in the measurement vector. As presented in [5], using the feature extraction algorithm, pulse responses can automatically be extracted from the measurements, so that the cumulative features constitute FIR models for the system (see next section).

The approach is purely based on associative relationships, not on causal dependencies. This is natural, because causality can never be induced automatically from measurement data. Inevitably, this means that the presented structure is not well suited for control applications directly: even if there seems to be correlation between signals, one should not think that changing one of them would change the others. Rather than using the presented framework for associative control tasks, the structure of the extracted model should be studied, and it should be further restructured appropriately (hopefully the created model really is understandable, then). The controller design should be left to the domain area expert — this task where the common sense knowledge plays a major role can probably never be given to a machine.

3.2 What are the 'system features'?

To have a more concrete view of the above discussion, let us briefly study what the features typically might be in a dynamical system. Typically, the feature representation of a measurement vector is of the form

$$f(k) = \bar{\theta} + \sum_{i=2}^{m} \xi_i(f(k)) \cdot \theta_i, \tag{5}$$

where $\bar{\theta}$ stands for the category prototype, in this case it contains the *operating point biases* of the signals (in the feature framework, this vector is multiplied formally by $\bar{\xi} = 1$). In a sparse system model, there are various such mutually exclusive sets of features; what does that mean?

Because of the additivity of the vectors, one set of features defines a linear model only, whereas a typical dynamical system is nonlinear. Locally, however, if the nonlinearities are smooth enough, *linearization* can be applied, so that the global models consist of *piecewise linear* approximations. These linearized submodels

can be represented by the features, so that the different $\bar{\theta}$ vectors stand for the linearization centers. What is more, the nonlinearity in the system can also be caused by structural, abrupt changes from a linear model to another, and the feature approach still works. To distinguish between operating points, to be able to tell the difference between them, it is important to have plenty of measurement information; it can be said that this approach and also the coresponding algorithm really *love data.*

Above, only the static features $\bar{\theta}$ were discussed — these features capture the operating point. From now on, it can be assumed that the additional features only modify the center $\bar{\theta}$ in either direction, and they are zero-mean.

What are the 'dynamic features' then? The idea is to use static regression models, finding the dependencies between the 'frozen' signals — to represent d'th order dynamics, $d+1$ samples of each of the signals are needed (at least in principle). As presented for example in [1], the discrete-time, linear d'th order proper single-input single-output (SISO) system can be represented in the ARX form

$$a_0 y(k) + a_1 y(k-1) + \cdots + a_d y(k-d) = b_0 u(k) + b_1 u(k-1) + \cdots + b_d u(k-d), \tag{6}$$

or, assuming that $a_0 \neq 0$,

$$y(k) = -\frac{a_1}{a_0} \cdot y(k-1) - \cdots - \frac{a_d}{a_0} \cdot y(k-d) + \frac{b_0}{a_0} \cdot u(k) + \frac{b_1}{a_0} \cdot u(k-1) + \cdots + \frac{b_d}{a_0} \cdot u(k-d). \tag{7}$$

There are $2d+2$ free variables and just one constraint equation; this means that there are $2d+1$ degrees of freedom, so that $2d+1$ features are needed to exactly represent the dynamical varieties of the system. The feature representation is not unique; one possibility (easiest to interpret?) can be based on the observed correlation between the variable $y(k)$ and the other variables — one of the $2d+1$ features, binding the measurements on $y(k)$ and $y(k-1)$ together, would (without normalization) look like

$$\theta_{a_1} = \begin{pmatrix} \vdots \\ -a_1/a_0 \\ \vdots \\ 1 \\ \vdots \end{pmatrix} \qquad \text{if} \qquad f(k) = \begin{pmatrix} \vdots \\ y(k) \\ \vdots \\ y(k-1) \\ \vdots \end{pmatrix}. \tag{8}$$

All of the elements in θ_{a_1} that are not explicitly shown are zeros. It can easily be verified that the weighting of this feature must be $\xi_{a_1}(f(k)) = y(k-1)$ to fulfill (together with the other analogously constructed features) the dynamic equation constraint.

It needs to be noted that even if signals in a linear system are additive, so that this kind of feature vectors can be added together, there are some signals where this property *does not apply:* these are the independent input signals. Because the causality graphs only reveal the flow of information, it does not matter how many times an input source is utilized — in all parallel channels the original signal value is intact. Of course, the same holds also in the other direction: two copies of the same signal do not add up. This problem becomes evident in systems with various outputs; for example, study the following system structure (one input signal is used in two places):

$$\begin{cases} y_1(k) = -\frac{a_{11}}{a_{10}} \cdot y_1(k-1) - \cdots - \frac{a_{1d}}{a_{10}} \cdot y_1(k-d) + \frac{b_{10}}{a_{10}} \cdot u(k) + \cdots + \frac{b_{1d}}{a_{10}} \cdot u(k-d) \\ y_2(k) = -\frac{a_{21}}{a_{20}} \cdot y_2(k-1) - \cdots - \frac{a_{2d}}{a_{20}} \cdot y_2(k-d) + \frac{b_{20}}{a_{20}} \cdot u(k) + \cdots + \frac{b_{2d}}{a_{20}} \cdot u(k-d) \end{cases} \tag{9}$$

The key point is that an independent input sample can be simultaneously used *only in one feature.* This means that there will not be $2 \cdot (2d+1)$ independent features, one set for each equation, because the features involving the input signal have to be combined; for example, one of these more loaded features can look like

$$\theta_{b_0} = \begin{pmatrix} \vdots \\ b_{10}/a_{10} \\ \vdots \\ b_{20}/a_{20} \\ \vdots \\ 1 \\ \vdots \end{pmatrix} \begin{matrix} \\ \leftarrow \quad y_1(k) \\ \\ \leftarrow \quad y_2(k) \\ \\ \leftarrow \quad u(k). \\ \end{matrix} \tag{10}$$

The features do not necessarily become easily interpreted, specially if the dynamic order of the system has been incorrectly estimated (there will be various signals that have mutual correlation, and the feature representation may become tangled). A simple approach to feature generation is based on the *weight function* $h(\kappa)$ that can also be used for defining the system response:

$$y(k) = \sum_{\kappa=0}^{\infty} h(\kappa)u(k-\kappa). \tag{11}$$

This means that instead of being auto-regressive as the above model structures, this formulation only utilizes the old input signal values; the feature construction follows the same lines as before. The main problem with this brute-force approach is that it is not exact if (when) the sequence is truncated; long enough sequence of (assumed!) input signals must be included to reach acceptable accuracy. This method is not so sensitive to structure, and the results remain nicely interpretable also in large systems — and, what is more, it seems to work nicely in practice (at least in simulations; see [5]).

3.3 Example application

The above presented model structure is illustrated using a 'typically non-typical' application example. It is a benchmark problem that has been introduced for comparing different kinds of soft-computing methods [3].

The problem consists of a nuclear plant data. There are 12 measurements available, but the physical interpretation of the signals is not given. These measurements are somewhat redundant — the goal is to utilize this redundancy and construct a 'supervisor' that would detect corrupted or missing measurement values (see Fig. 2). It is now assumed that the dynamical nature of the process can be neglected, that means, the present measurement values alone are needed to determine the process state.

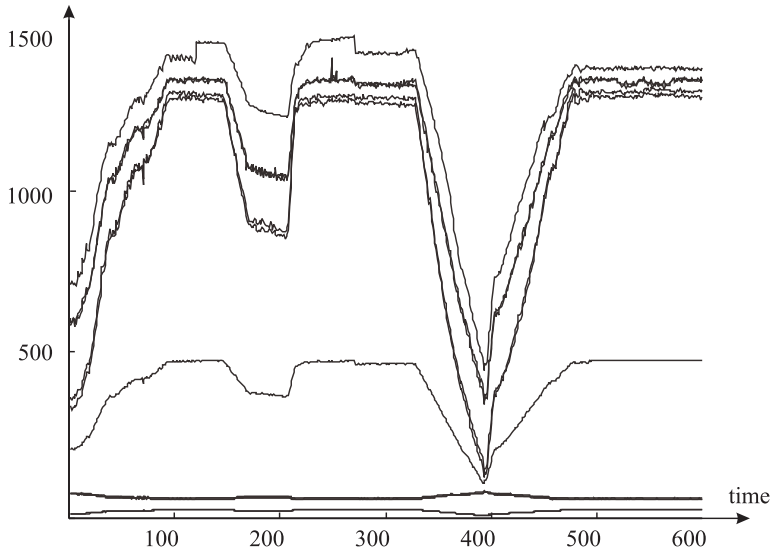

Figure 2: The observed process behavior: 12 measurements available

Originally, the problem was presented to the neural networks community, so that some kind of black-box 'filters' were expected: the original measurement vector is used as input, and the corrected one is given as output. Now, however, the problem is attacked by dividing it in subtasks — first, a model for the internal dependencies between signals is constructed, according to the presented guidelines, and only after that, the signals that do not match this model are detected.

The signals are first normalized, so that each signal sequence has mean value zero and variance 1. There were about 600 samples $f(k)$ of the process behavior. The adaptation algorithm that is presented, for example, in [5], is applied for extracting the features that should describe the process. Three feature vectors seems to be enough in this case (see Fig. 3); note that the feature vectors are normalized so that their length equals 1. The feature #1 (the feature vector being denoted as θ_1 in the formulas) seems to be the most important factor when describing the signals — the structure of this feature vector can be interpreted so that the signals 1, 4, 5, 6, 7, 10, 11, and 12 correlate positively with each other (and simultaneously also with this feature vector), whereas signals 2, 3, 8, and 9 correlate negatively. Features #2 and #3 are needed to modify this basic behavior of the signals in special operating regimes. Perhaps a domain area expert could see some meaning in these features.

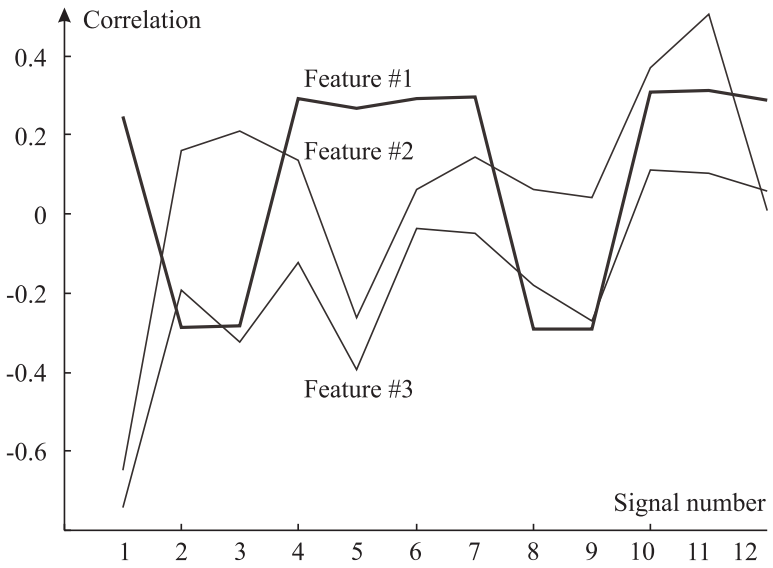

Figure 3: The extracted features

The 'loadings' of the different feature vectors at different time points in Fig. 2, or the contents of the generalized state vector $\xi(k)$, are shown in Fig. 4. It turns out that the feature #1 is always active, and its weighting is usually the most dominant, while the features #2 and #3 are mutually exclusive and their weighting never grows very high (this means that the signals really are rather redundant, so that the feature #1 alone can explain most of the signal variations). The feature #2 is mostly needed only when the signals have the steepest slope, while feature #3 is used otherwise.

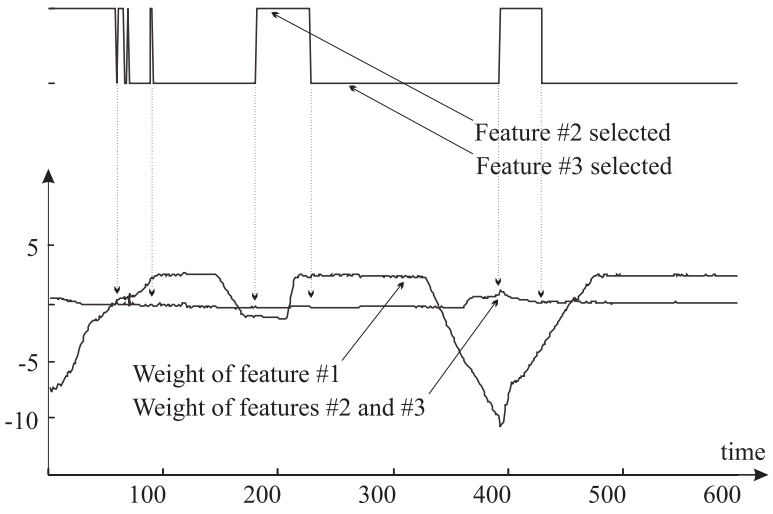

Figure 4: The weightings of the three features, or the elements of the generalized state vector $\xi(f(k))$ (compare to Fig. 2)

As presented in Fig. 5, the unknown signal can be readily reconstructed by setting the corresponding diagonal element in the weighting matrix W to zero while all other diagonal elements are ones (for details, see [5]). Using the model for reconstruction, the original signal can be nicely traced, and, furthermore, it seems that the spurious measurement errors have been eliminated in a robust way. It seems that at least for the presented benchmark problem, *soft sensors* can easily be constructed using the proposed feature model based approach.

4 Conclusions

It *seems* that often when an algorithm for implementing the presented approach is applied, something that looks 'surprisingly smart' emerges (for a list of examples, see [6]). The automatically extracted feature vectors tend to have some intuitively meaningful interpretation. This is fundamentally the mechanism that we were looking for in the beginning: having semantically relevant constructs emerge from the unstructured measurement data, is the key to reaching automated structure identification.

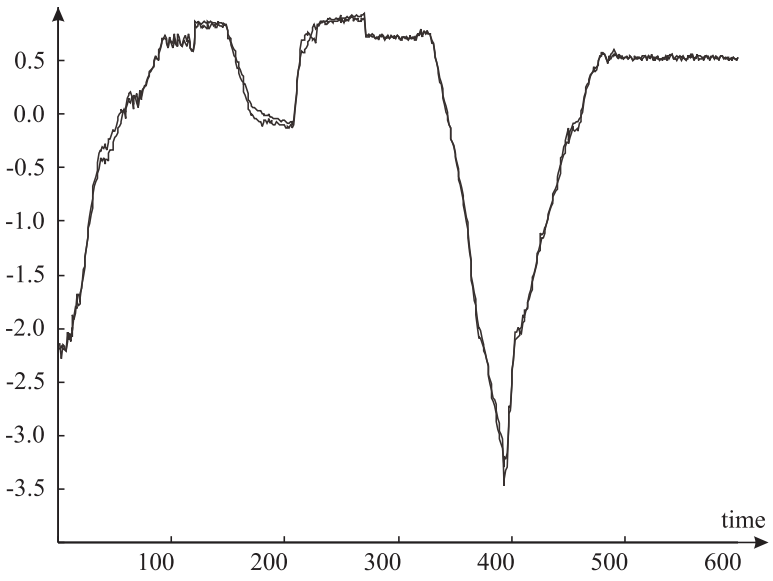

Figure 5: One typical example of the original signals and its reconstruction using the model (normalized)

In the control engineering field, the presented approach is being applied in a factory-scale process: in an on-going project, the state of a *flotation process* should be supervised using visual froth image data [8]. The flotation process where small air bubbles carry the valuable minerals out from the pulp is extremely difficult to model using traditional methodology. One of the goals in the project is to construct a *sensor fusion* tool where the numerical measures that are extracted from the froth images and spectral measurements are combined into meaningful process features. Hopefully, these features correlate with the physical state of the flotation cell, and the model can be used for estimating the mineral concentrations in the froth.

Acknowledgement

This study was financed by the Academy of Finland, and this support is gratefully acknowledged.

References

[1] Åström, K.J. and Wittenmark, B.: *Computer-Controlled Systems — Theory and Design* (2nd edition). Prentice-Hall International, Englewood Cliffs, New Jersey, 1990.

[2] A. Basilevsky: *Statistical Factor Analysis and Related Methods.* John Wiley & Sons, New York, 1994.

[3] Ciftcioglu, O. and Turkcan, E.: *Neural Network Benchmark for Power Plant Monitoring and Diagnostics.* NEA-NSC-DOC (96) 29, August 1996.

[4] Gaines, B.: *General Systems Research: Quo Vadis.* General Systems Yearbook, Vol. 24, 1979, pp. 1–9.

[5] Hyötyniemi, H.: *Automatic Structuring of Unknown Dynamic Systems.* In "Soft Computing in Engineering Design and Manufacturing" (Chawdhry, P.K., Roy, R., and Pant, R.K., eds.), Springer-verlag, London, 1998, pp. 410–419. Available at `http://130.233.148.14/Hyotyniemi/publications/`.

[6] Hyötyniemi, H.: *On the Statistical Nature of Complex Data.* In "SCAI'97 — Sixth Scandinavian Conference on Artificial Intelligence: Research Announcements" (ed. Grahne, G.), Helsinki University, Department of Computer Science, Report C–1997–49, Helsinki, Finland, 1997, pp. 13–27. Available at `http://130.233.148.14/Hyotyniemi/publications/`.

[7] C. Jutten and J. Herault, *Blind Separation of Sources, Part 1: An Adaptive Algorithm Based on Neuromimetic Architecture.* Signal Processing, Vol. 24, 1991, pp. 1–10.

[8] Niemi, A.J., Ylinen, R., and Hyötyniemi, H.: *On Characterization of Pulp and Froth in Cells of Flotation Plant.* International Journal of Mineral Processing, Vol. 51, 1997, pp. 51–65.