# DETECTING PROCESS STATE CHANGES BY NONLINEAR BLIND SOURCE SEPARATION

*Alexandre Iline*[1], *Harri Valpola and Erkki Oja*

Laboratory of Computer and Information Science
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Espoo, Finland

**ABSTRACT**

A variant of nonlinear blind source separation, the Nonlinear Dynamic Factor Analysis (NDFA) model, is based on noisy nonlinear mixtures of state variables, which are controlled by nonlinear system dynamics. The problem setting is blind because both the state variables, the nonlinear mixing model, and the nonlinear dynamics are unknown. As a special problem we consider the ability of NDFA to detect abrupt changes in the process dynamics. It is shown experimentally that NDFA is highly accurate and outperforms several standard change detection methods in this task.

## 1. INTRODUCTION

In nonlinear blind source separation (BSS), a collection of nonlinear combinations of unknown source signals, possibly with additive noise, are observed and the problem is to estimate the sources from them. Such a problem setting has important applications e.g. in speech, biomedical, industrial, or financial time series processing. Denoting by $\mathbf{x}(t)$the vector whose elements are the observed signals $x_i(t)$, $i = 1, ..., n$, by $\mathbf{s}(t)$ the vector whose elements are the underlying source signals $s_j(t)$, $j = 1, ..., m$, and by $\mathbf{n}(t)$ the additive noise, we have the model

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t)) + \mathbf{n}(t) \tag{1}$$

where $\mathbf{f}$ is the nonlinear mapping from the source signals to the observed signals. The term "blind" refers to the fact that both the source signals and the nonlinear mapping are unknown.

The problem cannot be solved unless some assumptions are made on the sources. In the Nonlinear Dynamic Factor Analysis (NDFA) model introduced by one of the authors [1], the source signals are assumed to follow nonlinear dynamics according to

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1)) + \mathbf{m}(t) \tag{2}$$

where $\mathbf{g}$ is an unknown function controlling the state changes and $\mathbf{m}(t)$ accounts for modeling errors and noise. Assuming $\mathbf{m}(t)$ to be white gaussian noise, the innovation process for the state vector $\mathbf{s}(t)$ is gaussian independent, hence the model (1) can be seen as a nonlinear generalization of Factor Analysis. Note that the source signals $s_j(t)$ are generally not independent, which is reasonable if they are the state space coordinates of some physical multivariate dynamical states.

An important problem from the point of view of certain applications, for example estimation and tracking of industrial processes from mixed observations, is to detect changes in the underlying dynamics. Then we may assume that the nonlinearity $\mathbf{f}$ giving the mapping from the state variables to the sensors stays constant, but the nonlinearity $\mathbf{g}$ controlling the process dynamics changes. Due to the complexity and nonlinearity of the mixing model, it may be very difficult to detect the process change directly from the observed signal vector $\mathbf{x}(t)$, but the actual state has to be estimated first.

In this work we examine the ability of the NDFA algorithm to detect changes in the process properties. We assume that these changes are caused by an abrupt change of the dynamical model of the underlying process.

In Section 2, the principle of the NDFA learning algoritm is briefly explained. The experimental setup is covered in Section 3, showing that in a fairly difficult simulated case with 3 underlying processes, 5 source signals, and 10 observed signals, the NDFA method can detect an abrupt change in one of the underlying processes with high accuracy, while some standard change detection methods perform poorly.

## 2. THE NDFA ALGORITHM

The NDFA algorithm used here is an extension of the nonlinear factor analysis algorithm [2]. In [1] it is shown how the approach can be extended to nonlinear state-space models. This section briefly outlines the model and learning algorithm. A more thorough presentation of the subject will be given in [3].

[1] Александр Ильин

### 2.1. Model structure

The unknown nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$ from (1) and (2) are modeled by multilayer perceptron (MLP) networks with one hidden layer of sigmoidal $\tanh$ nonlinearities. The function realized by the network can be written in vector notation as

$$\mathbf{f}(\mathbf{s}) = \mathbf{B}\tanh(\mathbf{A}\mathbf{s} + \mathbf{a}) + \mathbf{b} \tag{3}$$

where the $\tanh$ nonlinearity is applied component-wise. $\mathbf{A}$ and $\mathbf{B}$ are the weight matrices and $\mathbf{a}$ and $\mathbf{b}$ the bias vectors of the network. The function $\mathbf{g}$ has a similar structure except that the MLP network is used to model only the change in the state values and therefore

$$\mathbf{g}(\mathbf{s}) = \mathbf{s} + \mathbf{D}\tanh(\mathbf{C}\mathbf{s} + \mathbf{c}) + \mathbf{d} \tag{4}$$

The noise terms $\mathbf{n}(t)$ and $\mathbf{m}(t)$ are assumed to be Gaussian and white, i.e. the values at different time instants and different components at the same time instant are independent. Let us denote the observation set by $\mathbf{X} = (\mathbf{x}(1), \dots, \mathbf{x}(T))$, state set by $\mathbf{S} = (\mathbf{s}(1), \dots, \mathbf{s}(T))$ and all the model parameters by $\boldsymbol{\theta}$. The likelihood of the observations defined by the model can then be written as

$$\begin{aligned} p(\mathbf{X}|\mathbf{S}, \boldsymbol{\theta}) &= \prod_{i,t} p(x_i(t)|\mathbf{s}(t), \boldsymbol{\theta}) \\ &= \prod_{i,t} N(x_i(t);\ f_i(\mathbf{s}(t)), \exp(2v_i)) \end{aligned} \tag{5}$$

where $N(x;\ \mu, \sigma^2)$ denotes a Gaussian distribution over $x$ with mean $\mu$ and variance $\sigma^2$, $f_i(\mathbf{s}(t))$ denotes the $i$th component of the output of $\mathbf{f}$, and $v_i$ is a hyperparameter specifying the noise variance. The probability $p(\mathbf{S}|\boldsymbol{\theta})$ of the states $\mathbf{S}$ is specified similarly using the function $\mathbf{g}$. All the parameters of the model have hierarchical Gaussian priors. For example the noise parameters $v_i$ of different components of the data share a common prior [2, 3].

The parameterization of the variances through $\exp(2v)$ where $v \sim N(\alpha, \beta)$ corresponds to log-normal distribution of the variance. The inverse gamma distribution would be the conjugate prior in this case but log-normal distribution is close to it and it is easier to build a hierarchical prior using log-normal distributions than inverse gamma distribution.

### 2.2. Posterior approximation and regularization

The goal of ensemble learning is to fit a parametric approximating distribution $q(\boldsymbol{\theta}, \mathbf{S})$ to the true posterior $p(\boldsymbol{\theta}, \mathbf{S}|\mathbf{X})$. The misfit is measured by the Kullback-Leibler divergence between the approximation and the true posterior:

$$D(q(\mathbf{S}, \boldsymbol{\theta})||p(\mathbf{S}, \boldsymbol{\theta}|\mathbf{X})) = \mathrm{E}_{q(\mathbf{S},\boldsymbol{\theta})}\left[\ln \frac{q(\mathbf{S}, \boldsymbol{\theta})}{p(\mathbf{S}, \boldsymbol{\theta}|\mathbf{X})}\right] \tag{6}$$

where the expectation is calculated over the approximation $q(\mathbf{S}, \boldsymbol{\theta})$. The Kullback-Leibler divergence is always nonnegative. It attains its minimum of zero if and only if the two distributions are equal.

The posterior distribution can be written as $p(\mathbf{S}, \boldsymbol{\theta}|\mathbf{X}) = p(\mathbf{S}, \boldsymbol{\theta}, \mathbf{X})/p(\mathbf{X})$. The normalizing term $p(\mathbf{X})$ cannot usually be evaluated, but it is constant with respect to the model and it is possible to use the cost function

$$\begin{aligned} C &= D(q(\mathbf{S}, \boldsymbol{\theta})||p(\mathbf{S}, \boldsymbol{\theta}|\mathbf{X})) - \ln p(\mathbf{X}) \\ &= \mathrm{E}\left[\ln \frac{q(\mathbf{S}, \boldsymbol{\theta})}{p(\mathbf{S}, \boldsymbol{\theta}, \mathbf{X})}\right]. \end{aligned} \tag{7}$$

In this work we make use of the fact that $e^{-C}$ is roughly proportional to $p(\mathbf{X})$.

Usually the joint probability $P(\mathbf{S}, \boldsymbol{\theta}, \mathbf{X})$ is a product of simple terms due to the definition of the model. In this case $p(\mathbf{S}, \boldsymbol{\theta}, \mathbf{X}) = p(\mathbf{X}|\mathbf{S}, \boldsymbol{\theta})p(\mathbf{S}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ can be written as a product of univariate Gaussian distributions.

The cost function can be minimized efficiently if a suitably simple factorial form for the approximation is chosen. We use $q(\boldsymbol{\theta}, \mathbf{S}) = q(\boldsymbol{\theta})q(\mathbf{S})$, where $q(\boldsymbol{\theta}) = \prod_i q(\theta_i)$ is a product of univariate Gaussian distributions, i.e. the distribution for each parameter $\theta_i$ is parameterized with mean $\overline{\theta}_i$ and variance $\widetilde{\theta}_i$. These are the variational parameters of the distribution to be optimized.

The approximation $q(\mathbf{S})$ takes into account posterior dependences between the values of states at consecutive time instants. The approximation can be written as a product $q(\mathbf{S}) = \prod_i [q(s_i(1)) \prod_t q(s_i(t)|s_i(t-1))]$. The value $s_i(t)$ depends only on $s_i(t-1)$ at previous time instant, not on the other $s_j(t-1)$ with $j \neq i$. The distribution $q(s_i(t)|s_i(t-1))$ is a Gaussian with mean that depends linearly on the previous value as in $\mu_i(t) = \overline{s}_i(t) + \breve{s}_i(t-1, t)(s_i(t-1) - \overline{s}_i(t-1))$, and variance $\mathring{s}_i(t)$. The variational parameters of the distribution are $\overline{s}_i(t)$, $\breve{s}_i(t-1, t)$ and $\mathring{s}_i(t)$.

A positive side-effect of the restrictions on the approximating distribution $q(\mathbf{S}, \boldsymbol{\theta})$ is that the nonlinear dynamic reconstruction problem is regularized and becomes well-posed. With linear $\mathbf{f}$ and $\mathbf{g}$, the posterior distribution of the states $\mathbf{S}$ would be Gaussian, while nonlinear $\mathbf{f}$ and $\mathbf{g}$ result in non-Gaussian posterior distribution. Restricting $q(\mathbf{S})$ to be Gaussian therefore favors smooth mappings and regularizes the problem. This still leaves a rotational ambiguity which is solved by discouraging the posterior dependences between $s_i(t)$ and $s_j(t-1)$ with $j \neq i$.

### 2.3. Evaluating the cost function and updating the parameters

The parameters of the approximating distribution are optimized with gradient based iterative algorithms. During one sweep of the algorithm all the parameters are updated once, using all the available data. One sweep consists of two different phases. The order of the computations in these two phases is the same as in standard supervised back-propagation [4] but otherwise the algorithm is different. In the forward phase, the distributions of the outputs of the MLP networks are computed from the current values of the inputs. The value of the cost function is also evaluated. In the backward phase, the partial derivatives of the cost function with respect to all the parameters are fed back through the MLPs and the parameters are updated using this information.

When the cost function (7) is written for the model defined above, it splits into a sum of simple terms. Most of the terms can be evaluated analytically and only the terms involving the outputs of the MLP networks cannot be evaluated exactly. To compute those terms, the distributions of the outputs of the MLPs are calculated using a truncated Taylor series approximation for the MLPs. This procedure is explained in detail in [2, 3]. In the feedback phase, these computations are simply inverted to evaluate the gradients.

Let us denote the two parts of the cost function (7) arising from the denominator and numerator of the logarithm respectively by $C_p = \mathrm{E}_q[-\ln p]$ and $C_q = \mathrm{E}_q[\ln q]$. The term $C_q$ is a sum of negative entropies of Gaussians and has the form

$$C_q = \sum_i -\frac{1}{2}[1 + \ln(2\pi\widetilde{\theta}_i)] + \sum_{t,i} -\frac{1}{2}[1 + \ln(2\pi\mathring{s}_i(t))]. \tag{8}$$

The terms in the corresponding sum for $C_p$ are somewhat more complicated but they are also relatively simple expectations over Gaussian distributions [2, 1, 3].

An update rule for the posterior variances $\widetilde{\theta}_i$ is obtained by differentiating (7) with respect to $\widetilde{\theta}_i$, yielding [2, 3]

$$\frac{\partial C}{\partial \widetilde{\theta}_i} = \frac{\partial C_p}{\partial \widetilde{\theta}_i} + \frac{\partial C_q}{\partial \widetilde{\theta}_i} = \frac{\partial C_p}{\partial \widetilde{\theta}_i} - \frac{1}{2\widetilde{\theta}_i} \tag{9}$$

Equating this to zero yields a fixed-point iteration:

$$\widetilde{\theta}_i = \left[2\frac{\partial C_p}{\partial \widetilde{\theta}_i}\right]^{-1} \tag{10}$$

The posterior means $\overline{\theta}_i$ can be estimated from the approximate Newton iteration [2, 3]

$$\overline{\theta}_i \leftarrow \overline{\theta}_i - \frac{\partial C_p}{\partial \overline{\theta}_i}\left[\frac{\partial^2 C}{\partial \overline{\theta}_i^2}\right]^{-1} \approx \overline{\theta}_i - \frac{\partial C_p}{\partial \overline{\theta}_i}\widetilde{\theta}_i \tag{11}$$

The posterior means $\overline{s}_i(t)$ and variances $\mathring{s}_i(t)$ of the states are updated similarly. The update rule for the posterior linear dependences $\breve{s}_i(t-1,t)$ is also derived by solving the zero of the gradient [1, 3].

### 2.4. Learning scheme

In general the learning proceeds in batches. After each sweep through the data the distributions $q(\mathbf{S})$ and $q(\boldsymbol{\theta})$ are updated. There are slight changes to the basic learning scheme in the beginning of training. The hyperparameters governing the distributions of other parameters are not updated to avoid pruning away parts of the model that do not seem useful at the moment. The data is also embedded to have multiple time-shifted copies to encourage the emergence of states representing the dynamics. The embedded data is given by $\mathbf{z}^T(t) = [\mathbf{x}^T(t-d), \ldots, \mathbf{x}^T(t+d)]$ and it is used for the first 500 sweeps.

At the beginning, the posterior means of most of the parameters are initialized to random values. The posterior variances are initialized to small constant values. The posterior means of the states $\mathbf{s}(t)$ are initialized using a suitable number of principal components of the embedded data $\mathbf{z}(t)$ and are kept fixed to these values for the first 50 sweeps while only the MLP networks $\mathbf{f}$ and $\mathbf{g}$ are updated. Updates of the hyperparameters begin after the first 100 sweeps.

## 3. EXPERIMENTS

### 3.1. Problem setting

To examine the ability of the NDFA algorithm to detect changes in the process properties, we made an artificial set of 8 time series $s_i(t)$, shown in Fig. 1. The time series were generated in the same manner as was done in [1]. They were a harmonic oscillator with angular velocity $1/3$ and two independent Lorenz processes with three-dimensional parameter vectors.

The changes in the process model were achieved by changing the parameters of one of the Lorenz processes. They were taken to be $[3\ 26.5\ 1]$ before and $[2\ 20\ 1]$ after the change, which occurred at the time instant $\nu = 1500$. The parameters of the other Lorenz process were constant and equal to $[4\ 30\ 1]$. Fig. 1 clearly shows the change in one of the Lorenz processes.

Similar to [1] one dimension of each of the three underlying processes was hidden, to make the problem more difficult. Therefore only five linear projections of the eight states shown in Fig. 1 are present in the observed nonlinear mixtures. An arbitrary nonlinear mapping $\mathbf{f}$ was implemented using a random MLP network with inverse hyperbolic sine activation functions, which produced ten-dimensional observations from the five inputs. Finally a zero-mean Gaussian noise $\mathbf{n}(t)$ was added to the observations. Its standard deviation was chosen to be $0.1$, while the standard deviation of the signal was normalized to unity.

Fig. 2 shows the ten observations that correspond to the underlying states shown in Fig. 1. Note that the data model changes at time instant $\nu = 1500$, but this change has now

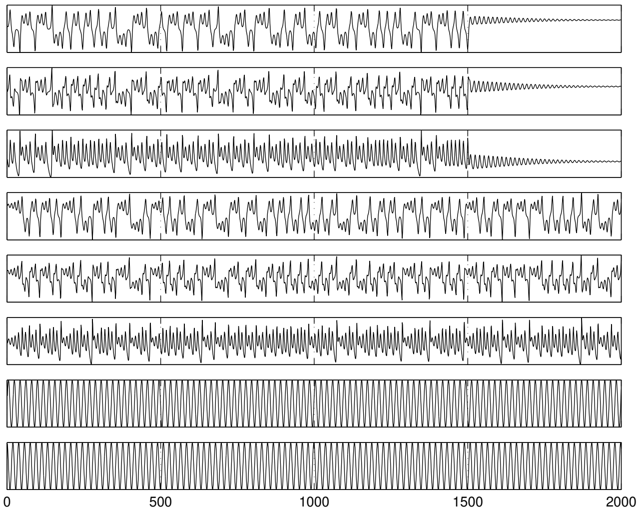



**Fig. 1**. Underlying states.

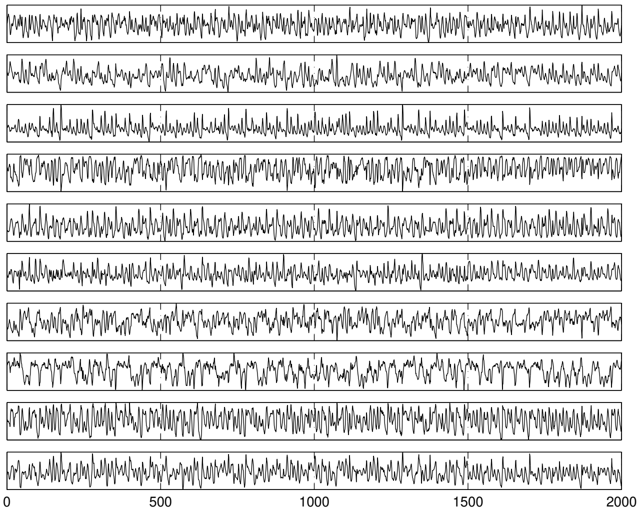



**Fig. 2**. Observations.

been hidden by the nonlinear mixing model and the additive noise.

### 3.2. Learning a dynamical model

For learning the dynamical model of the process using the NDFA algorithm, the observation signals were divided in two parts. The first thousand observations $\mathbf{x}(t)(1 \leq t \leq 1000)$ shown in Fig. 2 were taken as the training data to learn the model. This part contained observations with constant underlying process dynamics. The second data set $(t > 1000)$ was composed of two parts with different models of underlying states and it was used to test the examined properties of the NDFA.

As reported in [1], different numbers of states, MLP structures, and its random initializations were tested. Finally the model which minimized the cost function was taken for further experiments. It was a model having 10 states, and both the observation MLP network and the state dynamics MLP network had one hidden layer of 30 neurons.

The experimental results reported in [1] indicate that 7500 iterations of the NDFA algorithm were sufficient for learning the states and the observation mapping $\mathbf{f}$. However, more iterations were needed to learn the underlying dynamical process. Most of the learning was finished after 100,000 iterations, but some progress was observed even after 600,000 iterations.

### 3.3. Detection of the abrupt model change

The ability of the NDFA algorithm to detect model changes was tested on the second part of the data $\mathbf{x}(t)(1000 < t \leq 2000)$. The observations were considered sequentially in the order of their appearance and the decision about whether the change occurred was made at each time instant.

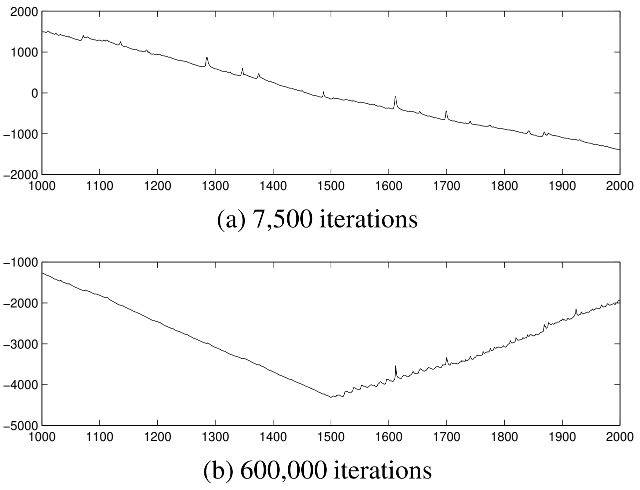
States $\mathbf{s}(t)$ were sequentially estimated for each new observation $\mathbf{x}(t)$ using a few iterations of the NDFA algorithm. First, an initial guess $\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1))$ was made and then the posterior means and variances of $\mathbf{s}(t)$ were updated. Mappings $\mathbf{f}$ and $\mathbf{g}$ were fixed and not adjusted to new observations.

Thus, for each new observation we got the value of the cost function $C(t)$ which can be used to detect the model changes. Recall that according to (7), the cost function implicitly includes the term $-\ln p(\mathbf{X}_t)$ where $\mathbf{X}_t$ is the set of observations $\mathbf{x}(\tau)$ for $1 \leq \tau \leq t$. Therefore the difference of two successive values of $C$ can be used to estimate the probability of new data given the previous one:

$$\begin{aligned} C(t) - C(t-1) &\approx -\ln p(\mathbf{X}_t) + \ln p(\mathbf{X}_{t-1}) \;\Rightarrow \\ p(\mathbf{x}(t)|\mathbf{X}_{t-1}) &= \frac{p(\mathbf{X}_t)}{p(\mathbf{X}_{t-1})} \approx e^{-(C(t)-C(t-1))} \end{aligned}$$

Fig. 3 shows the values of the cost function estimated for the data $\mathbf{x}(t)(1000 \leq t \leq 2000)$ presented in Fig. 2. It is evident that the NDFA algorithm trained with 600,000 iterations has been able to detect changes in the dynamical model of the data whereas 7500 of the NDFA iterations were not enough.

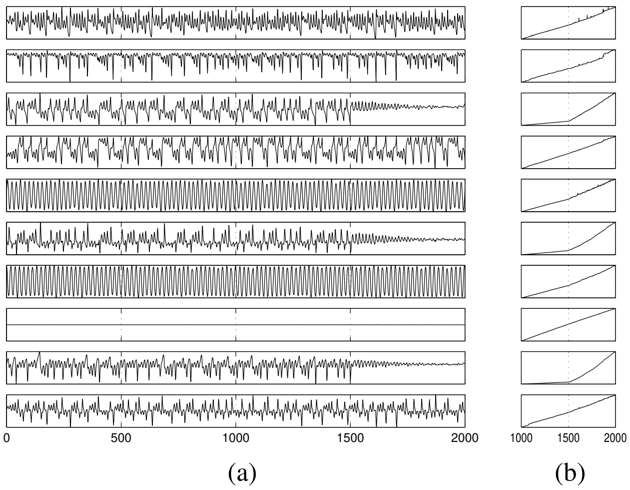
Besides the ability to detect changes of the process model, the NDFA algorithm can find out in which states the changes took place. Fig. 4a presents the underlying states which were estimated by the end of the learning phase $(1 \leq t \leq 1000)$ and during the detection phase $(1001 \leq t \leq 2000)$. The plot shows that only nine out of ten states are actually used. Two states model the harmonic oscillator dynamics, other four states describe the Lorenz process with con-




(a) 7,500 iterations

(b) 600,000 iterations

**Fig. 3**. Cost function.




(a) (b)

**Fig. 4**. Reconstructed states (a) and their contribution to cost function (b).

stant parameters, and the other three states correspond to the Lorenz process with changing parameters.

It is evident that the NDFA method has captured the changes of the underlying state dynamics. It can be proved by splitting the cost function and considering the contributions of different states. Combining the terms of the cost function it can be presented in the following form:

$$C = \sum_i C(s_i) + C_x + c$$

where $C(s_i)$ includes the terms originating from $q(s_i(t)|s_i(t-1))$ and $p(s_i(t)|\mathbf{s}(t-1), \boldsymbol{\theta})$, $C_x$ includes the terms from $p(\mathbf{X}|\mathbf{S}, \boldsymbol{\theta})$ and $c$ includes the rest of the terms originating from the parameters. During the detection phase $c$ is constant since the parameters are not updated.

Fig. 4b shows the state contributions $C(s_i)$ obtained for this problem. It is easy to see that the characteristics of the state contribution have changed after the change instant $\nu = 1500$. The contribution has increased for those states that correspond to the underlying process with changing dynamics.

### 3.4. Performance and comparisons

In order to assess the change detection performance of the NDFA algorithm and to compare it with other methods, the process changes were repeatedly simulated over 100 trials. Based on these results, the method was compared to some well-known classical methods for change detection. These algorithms control the current mean of the process. They were chosen since more complicated methods, which try to find a model of observations (e.g. a nonlinear autoregressive model), have proved to work poorly in this particular problem.

All the used methods are based on the following decision rule for the change detection problem:

$$d_t = \begin{cases} 1, & g_t \geq h \\ 0, & g_t < h \end{cases} \tag{12}$$

where $g_t$ is a statistic following the state of the observed process, $h$ is a chosen threshold, and the decision variable $d_t = 0$(respectively 1) means that at time $t$ the decision "no change" (respectively "change") is made.

Due to fluctuations in the $g_t$ statistic, sometimes false detections occur. Let $\nu$, as before, denote the time instant when the change actually takes place in a given simulation and $N$ the time instant when an algorithm makes the "change" decision according to the above decision rule. We can then define the probability of false detection as

$$P_f = P(N < \nu) \tag{13}$$

In practice, this can be estimated from the trials by counting the relative frequency of false detections.

Another performance measure is the speed with which a decision rule detects a (true) change, or the average time distance between the true change $\nu$ and the detected change $N$:

$$D = E(N - \nu | N \geq \nu, \nu) \tag{14}$$

Both measures (13) and (14) of course depend on the decision threshold $h$, which was varied in the simulations over a suitable range.

In the comparisons, the following decision statistics $g_t$ were considered:

1. NDFA statistics

$$g_t = C(t) - \min_{k \leq t} C(k)$$

2. nonsequential (fixed-size sample (FSS)) strategy based on the repeated Neyman-Pearson test

$$g_t(m) = \|\boldsymbol{S}_{t-m+1}^{t}\|$$
$$\boldsymbol{S}_l^k = \sum_{i=l}^{k}(\mathbf{x}(i) - \boldsymbol{\mu})$$

3. FSS-strategy based on the Bayesian statistics suggested by Chernoff and Zacks [5]

$$g_t(m) = \|\sum_{i=1}^{m}(i-1)(\mathbf{x}(t-m+i) - \boldsymbol{\mu})\|$$

4. Shewhart chart generalization for multi-dimensional processes [6, 7]

$$\begin{aligned} g_t(m) &= m(\bar{\mathbf{x}} - \boldsymbol{\mu})\mathbf{R}^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu}) \\ \bar{\mathbf{x}} &= 1/m\sum_{i=1}^{m}\mathbf{x}(t-m+i) \end{aligned}$$

In these equations, $\boldsymbol{\mu}$ and $\mathbf{R}$ are the mean and the covariance matrix estimated on the training data $\boldsymbol{x}(t)$ and the sample size for moving averages $m$ was chosen to be 80. Note that statistics 2–4 are heuristical generalizations of the corresponding procedures originally designed for one-dimensional processes.

Fig. 5 shows the simulation results. The $D$ -measure of Eq. (14) is plotted against the false detection probability $P_f$ of Eq. (13). Each indicated point on the curves gives one $(P_f, D)$ pair for a given value of the decision threshold $h$ in the rule (12). Thus, one curve corresponds to one of the algorithms with different values of threshold $h$. The closer a curve is to the origin, the faster the algorithm can detect the change with low false detection rate. The NDFA method outperforms the classical ones. It is evident that the standard algorithms are not able to detect the change properly, due to the complexity and nonlinearity of the process, while the NDFA method is highly accurate in this problem, giving a $D$ value close to zero with very low false detection rates.

**Acknowledgments**

This research has been funded by the European Commission project BLISS, and the Finnish Center of Excellence Programme (2000 - 2005) under the project New Information Processing Principles.

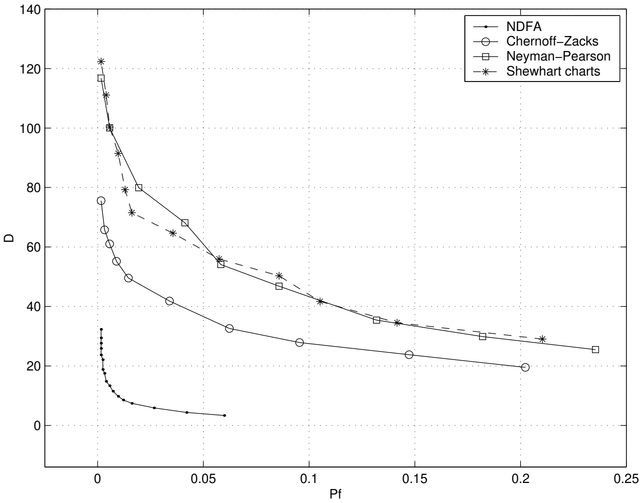



**Fig. 5**. Quality of change detection.

## 4. REFERENCES

[1] H. Valpola, "Unsupervised learning of nonlinear dynamic state-space models," Tech. Rep. A59, Lab of Computer and Information Science, Helsinki University of Technology, Finland, 2000.

[2] H. Lappalainen and A. Honkela, "Bayesian nonlinear independent component analysis by multi-layer perceptrons," in *Advances in Independent Component Analysis*, M. Girolami, Ed., pp. 93–121. Springer-Verlag, 2000.

[3] H. Valpola and J. Karhunen, "An unsupervised ensemble learning method for nonlinear dynamic state-space models," 2001, Submitted to a journal.

[4] S. Haykin, *Neural Networks – A Comprehensive Foundation, 2nd ed.*, Prentice-Hall, 1998.

[5] H. Chernoff and S. Zacks, "Estimating the current mean of a normal distribution which is subjected to changes in time," *The Annals of Mathematical Statistics*, vol. 35, pp. 999–1018, 1964.

[6] P. M. Ghave and P. E. Torgersen, "The multicharacteristic control charts," *Journal of Industrial Engineering*, vol. 19, no. 6, pp. 269–272, 1968.

[7] D. C. Montgomery and P. J. Klatt, "Economic design of $T^2$-control charts to maintain current control of a process," *Management Science*, vol. 19, no. 1, pp. 76–89, 1972.