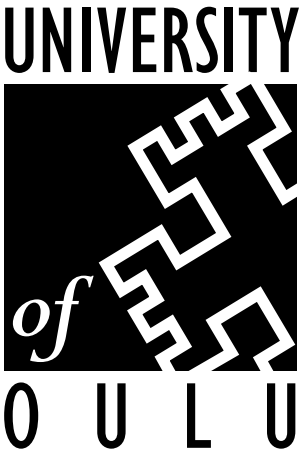


CONTROL ENGINEERING LABORATORY

A GUIDE TO INDEPENDENT COMPONENT ANALYSIS – Theory and Practice

Jelmer van Ast and Mika Ruusunen

Report A No 23, March 2004

University of Oulu
Control Engineering Laboratory
Report A No 23, March 2004

A GUIDE TO INDEPENDENT COMPONENT ANALYSIS - Theory and Practice

Jelmer van Ast and Mika Ruusunen

Control Engineering Laboratory,
Department of Process and Environmental Engineering,
University of Oulu
P.O.Box 4300, FIN-90014 University of Oulu, Finland

Abstract: Independent Component Analysis, or ICA, is a new technique for visualizing measured data. Appropriate data representation is necessary for the extraction of important information that is hidden in the measurements. Feature extraction and data reduction are two of the most important results of ICA that enable the application of the technique in many different fields of engineering. This report presents the theory and practical application of ICA in a way that is easy to understand for engineers from various kinds of expertise and works towards a step-by-step guideline for practical usage of ICA.

ICA assumes a data model that states that a collection of measurements originate from a number of independent sources that are linearly mixed. Based on higher order statistical properties of the measurements, ICA is able to estimate these independent sources or Independent Components (ICs) very accurately. It turns out that ICA is able to separate linearly mixed data from any distribution, but the Gaussian. This being nongaussian is what enables ICA to find the ICs.

In this report, ICA is applied to two cases. The first case is about prediction of failures in a screw insertion process. The second one is about the prediction of an upcoming paper break in a paper factory.

This work is the result of the three months internship of Jelmer van Ast at the Control Engineering Laboratory of the Department of Process and Environmental Engineering at the University of Oulu. He is an MSc student at the Delft Center for Systems and Control of Delft University of Technology under the supervision of Prof. Robert Babuška. The research resulting in this report has been done under the supervision of Mika Ruusunen.

Keywords: Independent Component Analysis, screw insertion, paper break, data analysis, FastICA, feature extraction.

ISBN 951-42-7315-X
ISSN 1238-9390

University of Oulu
Department of Process and Environmental Engineering
Control Engineering Laboratory
P.O.Box 4300, FIN-90014 University of Oulu

1 INTRODUCTION

Science is all about understanding what is going on around us. Applied science, like engineering, is especially focussed on how this understanding can be exploited to create systems that are useful to us. In this report, Independent Component Analysis, or ICA, is treated as a way to enhance our knowledge about measured signals. It assumes that many signals that can be measured actually originate from *independent sources* and provides us with a method to retrieve these sources. In ICA, these independent sources are called Independent Components (ICs). Some clues can be given why knowing the independent components is of value to engineers.

- Knowing the independent components and knowing in which way they contribute to what has been measured can give insight to the process that generated the data. If nothing is known about the process, it maybe possible to retrieve some of the model parameters. Another application may be to *estimate* the ICs from both data of normal and of abnormal process conditions. Possibly, analysing the ICs might reveal what causes the process conditions to be abnormal.
- A finite number of independent components can be linearly combined in infinitely many ways. According to /3/, natural images all have at their source the same independent components. The way each IC contributes to the total signal then determines the image. In this way, it would be possible to *compress* natural images, based on their independent components.
- The most well known application of ICA is the Blind Source Separation (BSS) problem. A famous explanation of BSS is the 'cocktail party problem'. Suppose that we find ourselves on a cocktail party with many other people hanging around. If every person would be holding a microphone to record his voice, we would find out that each microphone didn't only record the voice of its owner, but also the voices of all the other speakers at the cocktail party. So, each microphone records a *linear mixture* of all the speech-signals in the room added with some noise. It is important to notice that all the speech-signals are independent. /2/ shows that ICA is able to determine these independent components (the individual speech-signals) based on only the information from the microphones.

From the above-mentioned examples, it can be noticed that ICA is about extracting features appropriate for a given application. Intuitively, we may also sense that the ICs of a certain dataset can be appropriate features in general.

To enhance the readability of this report, important intermediate conclusions are highlighted with a big leading exclamation mark. Right now, we can already mark the first intermediate conclusion, which is:

! *Finding the independent components of certain observations may be very useful for several applications.*

The goal of this report is to provide the reader with enough information about ICA to enable him or her to apply it in his own field of research. Information from both

theoretical and practical point of view will lead to a guideline for the use of ICA in chapter 4. The next chapter will cover the theory of ICA from the problem to a solution. Since ICA is a very new subject, the theory in that chapter is mainly based on the only book that covers the subject entirely by Hyvärinen, Karhunen and Oja *Independent Component Analysis* /6/. Chapter 3 will cover the application of ICA. Based on two applications, conclusions are drawn about the pitfalls and successes of ICA. As was just mentioned, it all comes together in Chapter 4. The guideline for practical usage of ICA in this chapter can be used as reference for the reader who is using ICA for his own project.

2 THEORY OF INDEPENDENT COMPONENT ANALYSIS

2.1 Introduction

Regarding the ‘cocktail party problem’ from the previous chapter, we know that something like ICA is possible, because the human ears and brains in essence actually perform the same analysis. Having two ears and a typical brain structure enables us to listen carefully to the one we are talking to, hearing everything that is said even in very noisy environments. We can guess, based on what we know about the human brain that we have to find a nonlinear approach to the problem. Of course this is by no means a proof that there exists something like ICA, but it gives us motivation to search for it. Looking around us, especially to what nature has evolved in, is always a good way to get new ideas on scientific topics.

How ICA exactly works, based on which principles and under which assumptions and restrictions will be treated in the following paragraphs in a *step-by-step* manner. The subject will be treated from *problem to solution*, providing additional theory along the way. Wherever important background knowledge is suspected to be not clear to the reader, a textbox appears in the text summarising its ins and outs. Of course, the more advanced reader can skip these textboxes.

The theory is based on the book Independent Component Analysis /6/, but is written in a way that aims at giving the reader the feeling of discovering the solution to the ICA problem by himself. The aim of this is to give, even the reader inexperienced in data analysis, a clear view of the subject. For the same, but higher specialized treatment of the subject, the reader can refer to /6/or e.g. /12/.

2.2 The ICA model

Before we can start analysing independent components, whatever that could be, we have to make clear what exactly we are looking for. A good start is therefore to put our thoughts together in a *mathematical model*.
We already made clear that we could measure some signals that have in fact originated from a finite number of independent sources. These independent sources are *linearly mixed* in the measured signals.
To illustrate this, consider again the cocktail party problem. In this case, the independent sources are all the speech-signals originating from the people at the party. In the medium, in this case the air, the speech-signals are linearly mixed by superposition. Each person hears a different mixture, because the distance between him or her and the other persons determine how strong each independent source contributes to it.

In a mathematical formula, this looks like:

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t) \,. \\ x_3(t) &= a_{31}s_1(t) + a_{32}s_2(t) + a_{33}s_3(t) \end{aligned} \qquad (1)$$

Here we consider the case with only three persons at the cocktail party who are all talking simultaneously. The $x_i(t)$ and $s_j(t)$ with $i,j = 1, 2, 3$ represent the measured and the source signals respectively. The $a_{ij} \in \mathbb{R}$ with $i,j = 1, 2, 3$ represent the different weights.

Formula (1) states that each $x_i(t)$ is a weighted sum of all the $s_i(t)$.

Representation of signals

We can represent a signal by denoting the function that gives the value (amplitude) of the signal at each time instant. In this way, the signal is described at each time instant and can therefore be called a *continuous-time* signal. In (1), the signals are continuous-time, denoted by $x(t)$ and $s(t)$. This is a good representation when it is used for pure theoretical purposes or for analogue system design. For ICA however, the implementation will be an *algorithm* suitable only for *digital computing*. Because of this, the representation of the signal has to be in *discrete-time*. This means that we have to *sample* the signal, i.e. evaluate the signal at a finite number of time instants only. In order to properly sample a signal, one has to consider the sampling rate /13/. In the remaining of this report it is assumed that the signals are properly sampled.

We assume now that the measured signals are sampled. For simplicity we will omit the sample index. In *matrix-vector form*, we can then write (1) as follows:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \,, \qquad (2)$$

where $\mathbf{x}$ is a vector of mixtures $x_1,\ldots,x_n$, $\mathbf{s}$ the vector of independent components $s_1,\ldots,s_n$, and $\mathbf{A}$ is the mixing $n \times n$ matrix with elements a_{ij}. This model is known as the *noise-free* ICA model. It is a *latent variable generative model*. This means that it describes the generation of the observed data with very little knowledge about its variables. In fact, only the observed data, $\mathbf{x}$ is known and both $\mathbf{A}$ and $\mathbf{s}$ are unknown. One can imagine that it will be a very demanding task to determine both $\mathbf{A}$ and $\mathbf{s}$ from the observed data only. We need to retrieve much information from the observed data. Regarding the signals in the model as random variables can do this. Random variables are clearly defined in statistics and many techniques to handle them are available. *Independence*, for example is a well-defined property of random variables. We will see that regarding the variables in the model as random variables is the first step in the direction of solving the ICA problem. The vector $\mathbf{x}$ as well as $\mathbf{s}$ are collections of random variables. The scalars $x_i(t)$ and $s_j(t)$ are then samples at time instants t of these random variables. A great advantage of this is that we can now use the model in (2) also for data different from time signals, such as pixel values in a digital image.

2.3 Describing data as random variables

When we describe data as random variables we assign to each possible value of the variable a certain probability. For continuous valued variables, this is not directly possible, because the possibility that a variable takes a certain value amongst an infinitive number of possible values is of course zero. We therefore speak about *probability density functions (pdf)* and *cumulative distribution functions (cdf)* instead. The cdf $F_x(x_0)$ of a random variable x at a point $x = x_0$ is defined as:

$$F_x(x_0) = P(x \le x_0)\,. \tag{3}$$

The cdf is thus defined as the probability of the *outcome* that x takes a value smaller than or equal to x_0. It can be easily seen from the definition and from what we know about probabilities that $F_x(x)$ can take values in the range $0 \le F_x(x) \le 1$, with $F_x(-\infty) = 0$ and $F_x(+\infty) = 1$. The pdf of a random variable can be obtained from its cdf as:

$$p_x(x_0) = \left.\frac{dF_x(x)}{dx}\right|_{x=x_0}\,. \tag{4}$$

In practice, however, we mostly have knowledge about the pdf of a random variable and its cdf is then computed from it like:

$$F_x(x_0) = \int_{-\infty}^{x_0} p_x(\xi)d\xi\,. \tag{5}$$

A random variable can thus be described by its pdf. Keep in mind that $p_x(x_0)$ doesn't mean the *probability* of x at x_0; it means the density of the probability of x at x_0. To get the probability, one has to take the *area* under the density function, or equivalently the *difference* between $F_x(x)$ and $F_x(x\text{-}dx)$ with dx small.
There are of course an infinite number of possible pdfs and often we do not know the exact pdf of a random variable. It could then be assumed that a *parametric model* describes the pdf of a random variable. In that case, we only have to determine the *parameters* that characterize the pdf. Of course, the resulting pdf is not exactly correct, but hopefully sufficiently accurate for its application.

There are a lot of parametric models of pdfs /5/. There is one family of models that is particularly interesting now, because it will help us in understanding the ICA problem. This is the *exponential power family* of pdfs, which can be expressed for *zero-mean* as:

$$p_x(x) = C\exp\left(-\frac{|x|^{\nu}}{\nu\mathbf{E}\left\{|x|^{\nu}\right\}}\right). \tag{6}$$

The E(…) is called the *expectation*. The definition of expectation is:

$$\mathrm{E}\{\mathrm{g(x)}\} = \int_{-\infty}^{\infty} \mathrm{g(x)}\,p_x(\mathrm{x})dx \tag{7}$$

,

for any function g(x) performed on a scalar, vector or matrix.
In (6), *C* is a *scaling constant* and *v* is a positive real-valued power, which determines the *type* of the pdf. We can distinguish three important types of density functions. For $v = 2$ we get the Gaussian pdf, which will turn out to be of major importance to the ICA problem. When $v < 2$, the pdf in (6) will give rise to *supergaussian* densities, and $v > 2$ to *subgaussian* ones. To illustrate this, we will consider one example of each of these cases. These densities will continue to appear during the remaining of this chapter.

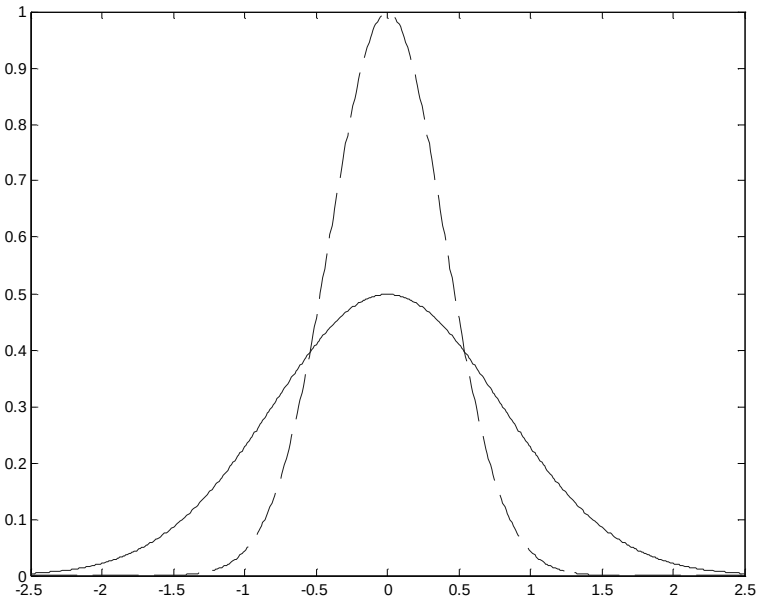


Figure 1. Two Gaussian densities; dashed: $\sigma = 0.4$; solid: $\sigma = 0.8$.

Figure 1 presents the pdf of a Gaussian random variable. In formula, this pdf looks like:

$$p_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-m)^2}{2\sigma^2} \right) \tag{8}$$

.

This is a very important distribution, as we will see in the remaining of this chapter and especially in paragraph 2.4.5. We see that the Gaussian density is characterized by only two variables, *m* and σ^2; the *mean* and the *variance* respectively (σ is called the *standard deviation*). In most of the upcoming theory in this report, the densities are considered to be zero-mean. This means that the center of the density occurs at zero and the formula of the pdf can be simplified to:

$$p_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2} \right). \tag{9}$$

We indeed recognise that this formula results from (6) when $v = 2$ and the expectation is computed. We will see more about expectations, means and variances later on in this

report. Along the way, also more will be clear about this special distribution. For now, it is most important to compare it with an example of a supergaussian and a subgaussian pdf.

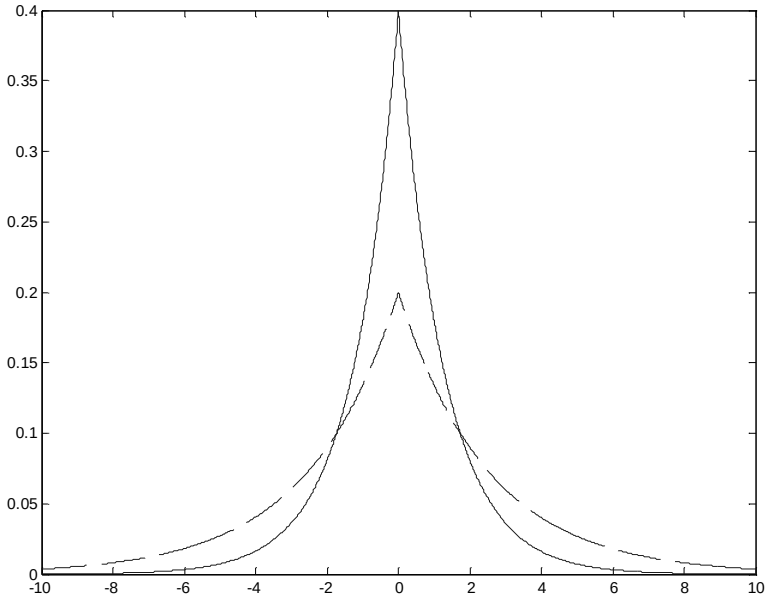


Figure 2. Two Laplacian densities; dashed: $\lambda = 0.4$; solid: $\lambda = 0.8$.

In Figure 2 we see the pdf of a Laplacian random variable, which is a typical supergaussian distribution. It results from (6) when v = 1. A supergaussian pdf is characterized by large values of the density at values close to zero and far from zero, in the *tails* of the distribution, compared to the Gaussian pdf. In between, the density is relatively small. In the case of the Laplacian pdf we can see this very clearly.
If we look at the formula (for again zero-mean) of this pdf:

$$p_x(x) = \frac{\lambda}{2}\exp\left(-\lambda|x|\right), \qquad (10)$$

we see that this pdf is characterized by the only parameter λ . Like the only parameter σ in the Gaussian, it determines both the height of its top and the width of its tails.

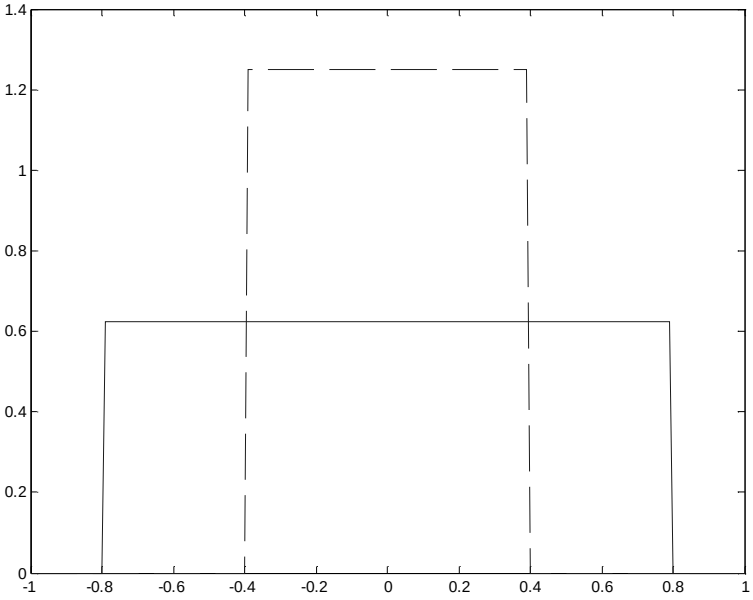


Figure 3. Two uniform densities; dashed: $a = 0.8$; solid: $a = 0.8$.

The *uniform* pdf is shown in Figure 3. Its value is a constant in a certain interval. Its value outside of it is zero. Like any other pdf, the total area under it has to be equal to one, so as the parameter *a* increases, the interval becomes wider and the height smaller. This can also be seen in its mathematical expression:

$$p_x(x) = \begin{cases} \dfrac{1}{2a}, x \in [-a, a] \\ 0, \text{elsewhere} \end{cases} . \tag{11}$$

This is a typical example of a subgaussian distribution, since it has low values for the density at values of *x* that are around zero or very large compared to the Gaussian pdf. In between, the density is relatively large.
The uniform pdf results from (6) when $\nu \to \infty$.

When we want to speak about a density that describes the distribution of more than one random variable, we speak about the *joint density*, or *multivariate density*. In fact we are then speaking about the distribution of a *random vector*, which is nothing more than a vector with random variables as scalar components. To illustrate this, the joint pdf of a Gaussian random vector of N-dimensions is shown here:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{N}{2}} \sqrt{|\mathbf{C}_{\mathbf{xx}}|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{m}_{\mathbf{x}})^T \mathbf{C}_{\mathbf{xx}}^{-1} (\mathbf{x} - \mathbf{m}_{\mathbf{x}}) \right), \tag{12}$$

with $\mathbf{x} = [x_1, x_2, ..., x_N]^T$, $\mathbf{m}_\mathbf{x} = [m_{x_1}, m_{x_2}, ..., m_{x_N}]^T$ and $\mathbf{C}_{xx} = \begin{bmatrix} C_{x_1x_1} & \cdots & C_{x_1x_N} \\ \vdots & \ddots & \vdots \\ C_{x_Nx_1} & \cdots & C_{x_Nx_N} \end{bmatrix}$.

A matrix like $\mathbf{C}_{xx}$ is called a *covariance matrix* as we will see in deeper detail later in this report.

2.4 Understanding the model

In order to solve a problem, it is most important to have a throughout understanding about it. A good way of doing this is to visualise the problem. Let us see how we can do that with ICA.
We start with the model in (2), for convenience displayed below again:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \,. \tag{13}$$

Remember that this model states that we have some independent components **s**, which are linearly mixed through **A** to give us the only measurable data **x**. It was already decided that we will regard **x** and **s** as random variables. We know that random variables are characterized by their probability density function (pdf). We will consider three cases: Gaussian, supergaussian and subgaussian to understand the problem. Later we will try to make the generalisation to other distribution functions. To keep things visible, we restrict ourselves now to only two dimensions. This means that both **s** and **x** only consist of two components. The generalisation to higher dimensions will be made afterwards.

2.4.1 Considering supergaussians

We start our analysis with random variables of a supergaussian distribution. As we saw before, the Laplacian distribution is a frequently used. Based on its pdf we can generate two independent components. Their *scatter-plot* is displayed in Figure 4.

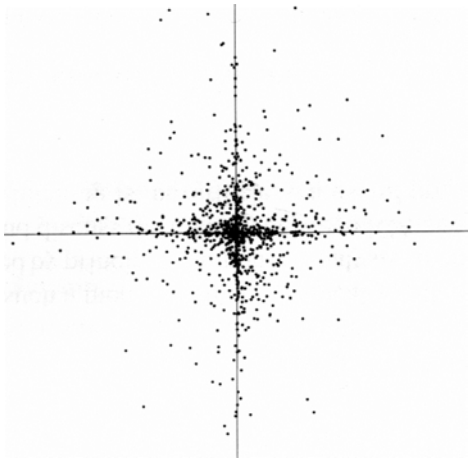


Figure 4. Two independent Laplacian distributions.

Suppose these are the components in **s**. To get the components in **x** we have to multiply the independent components with a mixing matrix **A**. The resulted mixture is displayed below.

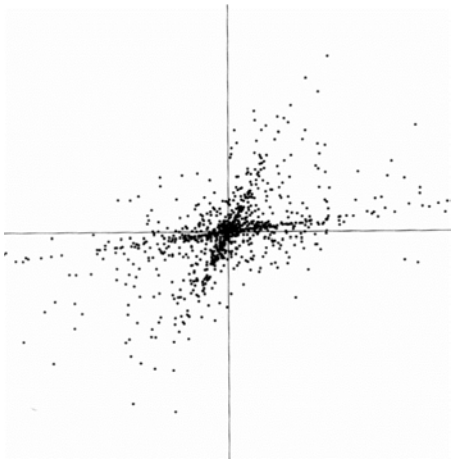


Figure 5. Linear mixture of two Laplacian distributions.

We can understand why this scatter-plot is the result of this transformation by regarding the principle directions of this distribution. The first principle direction is $[0, 1]^T$ in figure 4. If we multiply **A** with this vector we get the vector pointing in the direction of one of the principle directions in Figure 5.

We can clearly see from the distribution of **x** that its components are not independent.

Independence

A joint distributed random vector is independent when one component does not give any information about the value of all the other components. Mathematically we can then say that its pdf can be factorised in its *marginal* pdfs:

$$p_x(\mathbf{x}) = p_{x_1}(x_1)\,p_{x_2}(x_2)\ldots p_{xN}(x_N) = \prod_N p_{x_i}(x_i)\,. \qquad (14)$$

For example, knowing that x_1 is large implies that x_2 can only have a few possible values. To solve the ICA problem, we can rewrite the ICA model in (2) and (13) in a different way:

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}\,. \qquad (15)$$

We see that we can only do this if **A** is *invertible*. This implies that **A** must be a *non-singular square* matrix.

We now have put a *restriction* on the model. One might say that restrictions are a bad thing that should be avoided, but in fact, the problem is not solvable when there are no

restrictions on it. The model, in closed form, has many possible solutions. From a measured random vector **x** there can be an infinite number or possible matrices A and random vectors s. Indeed, when **A** is known, then **s** is fixed, but still there are infinitively many ways of making **A**. Restrictions have to help us reduce the number of possibilities and lead us in the right direction. Of course, one has to be very careful when imposing restrictions, because putting a wrong restriction on the model may cut off the road to the right solution. Since restrictions are that important for this problem, we will keep track of them during this chapter.

The first restriction can already be derived from where we actually started and where we want to go. That is that we want the components of the random vector **s** to be *statistically independent*. So let's mark this important restriction as we just determined to do:

Restriction 1: The independent components are assumed to be statistically independent.

The reason we started talking about restrictions is that we actually restricted **A** to be invertible. So we can also mark this important restriction:

Restriction 2: We assume that the unknown mixing matrix is invertible.

For convenience we will rewrite (15) as:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \,. \tag{16}$$

We can now say that we have the observations in **x** and from that we want to estimate the matrix **W** so that it results in a random vector **s** which components are statistically independent. In real life, real independence is very hard to achieve. It turns out to be enough to say that we want the components in **s** to be as independent as possible. Of course we need to put bounds on this remark, otherwise ICA will always find an **s** no matter how independent the components are. We do not bother with this right now.

We thus have now the task to estimate the matrix **W** while obeying the restrictions. To do this, we have to exploit much of the information that is hidden in the observations **x**. Because **x** is a random vector, we will look at its *statistical properties*.

2.4.2 Some statistical properties of random variables

We can characterise a random variable by its statistics. Examples of statistics of a random variable are the mean and variance. However, in most cases we need more statistics to describe a random variable. There are different ways to do this and this can be done for many orders. We will state here three important types of statistics, for the orders one to four. In each case, we will consider the probability density function $p_x(x)$ of a scalar random variable *x*.

Moments

The *j*th *moment* α_j of *x* is defined as:

$$\alpha_j = \mathrm{E}\{x^j\} = \int_{-\infty}^{\infty} \xi^j p_x(\xi) d\xi \ , \qquad j = 1, 2, ... \tag{17}$$

Only the first two moments are of importance, these are $\alpha_1 = \mathrm{E}\{x\}$, which is the mean m_x and $\alpha_2 = \mathrm{E}\{x^2\}$, which represents the *average power* of *x*.
We can also take these moments around the mean. We then get the *central moments*.

Central moments

The *j*th central moment μ_j of *x* is defined as:

$$\mu_j = \mathrm{E}\{(x - \alpha_1)^j\} = \int_{-\infty}^{\infty} (\xi - m_x)^j p_x(\xi) d\xi \ , \qquad j = 1, 2, ... \tag{18}$$

In this case, the first central moment μ_1 is of no meaning, since it is always zero. The second central moment $\mu_2 = \sigma_x^2$ is the variance of *x*. The third central moment $\mu_3 = \mathrm{E}\{(x - m_x)^3\}$ is called the scewness, which is a measure for the asymmetry of a pdf. For zero-mean, the moments and the central moments become equivalent.

We now have useful expressions for the first three orders of statistics. Of course, there are more orders available from both moments and central moments, but these have in practice some drawbacks that prevent us from being interested in them. Still, it will turn out that we need also a fourth order statistic to solve the ICA problem. We will find this last one in the so-called *cumulants*.

Cumulants

According to /4/, the generating function for cumulants generates the following first four cumulants;

$$\kappa_1 = 0 \,,\ \kappa_2 = E\{x^2\} \,,\ \kappa_3 = E\{x^3\} \text{ and } \kappa_4 = E\{x^4\} - 3[E\{x^2\}]^2 \,. \tag{19}$$

We see that the first three are equivalent to the first three moments, but that the fourth one is different. This is exactly the fourth order statistic that we were looking for. Its name is the *kurtosis*. It has some nice features that make the kurtosis easy to use. These features are:

additivity:

$$\mathrm{kurt}(x + y) = \mathrm{kurt}(x) + \mathrm{kurt}(y) \tag{20}$$

and

$$\mathrm{kurt}(\beta x) = \beta^4 \mathrm{kurt}(x) \,. \tag{21}$$

We can of course also speak about moments, central moments and cumulants for random vectors.
One remark still has to be made. Although some pdfs have infinite moments and although some pdfs have the same values for their moments, for almost every pdf this is not the case. We will therefore assume now that if we consider a certain pdf, it is uniquely defined by its statistics that are also finite valued.

2.4.3 Exploiting statistics

We will thus try to exploit the statistical properties of **x** to estimate its independent components **s** and its mixing matrix **W**. Remember that **A** is just the inverse of **W**, so estimating one of them implies the other.

Exploiting first order statistics
We can easily remove the effect of first order statistics by calculating the mean, storing it and subtracting it from the data. Since we only have sampled data; thus a finite number of data points, we have to make an estimate of the mean. The general estimation of an expectation is:

$$\mathrm{E}\{\mathrm{g}(\mathbf{x})\} \approx \frac{1}{K}\sum_{j=1}^{K}\mathrm{g}(\mathbf{x}_\mathrm{j}). \tag{22}$$

To get an estimation of the mean we can thus take:

$$\hat{\mathbf{m}}_x = \frac{1}{K}\sum_{j=1}^{K}\mathbf{x}_j\ , \tag{23}$$

which is called the *sample mean*.

As already mentioned, in this report we assume that the data are zero-mean. Mostly this is not the case. We then have to make the data zero-mean by estimating the mean and subtracting it from the data:

$$\mathbf{x} \leftarrow \mathbf{x}' - \hat{\mathbf{m}}'_x\ , \tag{24}$$

where $\mathbf{x}'$ is the original data, $\hat{\mathbf{m}}'_x$ its mean and $\mathbf{x}$ the zero-mean data. This makes the independent components **s** also zero-mean, because:

$$\hat{\mathbf{m}}_s = \mathbf{W}\hat{\mathbf{m}}_x\ , \tag{25}$$

and since $\hat{\mathbf{m}}_x$ is zero, $\hat{\mathbf{m}}_s$ is also zero. **W** remains unaffected, so making the data zero-mean will not affect the estimation of the mixing matrix. If we store $\hat{\mathbf{m}}_x$, we can always add it to the obtained independent components afterwards, like:

$$\mathbf{s}' = \mathbf{s} + \mathbf{W}\hat{\mathbf{m}}'_\mathbf{x}\ . \tag{26}$$

We now have exploited the first order statistics of the observed data. Its effects are removed and hopefully this brings us closer to the solution. We will now proceed with removing the effects of second order statistics.

Exploiting second order statistics

The two important second order statistics are the *cross-covariance*:

$$c_{x_1x_2} = \mathrm{E}\{(x_1 - m_{x_1})(x_2 - m_{x_2})\} \tag{27}$$

and the *cross-correlation*:

$$r_{x_1x_2} = \mathrm{E}\{x_1x_2\}. \tag{28}$$

As can be seen very easily, these two are the same for zero-mean data, which is the case after removing the effect of the first order statistics as we did. Both cross-covariance and cross-correlation say something about the *linear dependence* between the two random variables. This can also be explained as *linear predictability*. We say that the distribution in figure 5 is *positively correlated*, since an increase in one component generally also means an increase in the other component. When two random variables are not correlated, we call them *uncorrelated*. In the zero-mean case $c_{x_1x_2} = r_{x_1x_2} = 0$. This means that they are *orthogonal*. We see that making two random variables uncorrelated removes the effect of the second order statistics. This is exactly what we are looking for. When we look at the distribution of the independent components in figure 4, we see that the independent components are indeed orthogonal (i.e. they make an angle of 90°). We therefore also can sense that we are on the right way to solve the problem.

When we want a random vector to be uncorrelated, we want to have the cross-correlation of zero for every two components in the random vector, thus:

$$c_{x_ix_j} = \mathrm{E}\{(x_i - m_{x_i})(x_j - m_{x_j})\} = 0, \qquad \forall i \neq j. \tag{29}$$

When $i = j$ we get the variance of the individual components. To completely remove the effect of second order statistics, we want these variances to equal to one. Another way of motivating this can be found when we look again at (16) or e.g. (13). We can see that a multiplication by a constant at both the left and the right side of (16) results in the same model. So even if we have a good estimate of **W**, we are still not sure if it is the right one, for there are still an infinite number of scaled versions of **W**, that are equally likely to be the right one. We can partly bypass this by restricting ourselves to find only independent components of unit variance.

Restriction 3: We assume that the statistically independent components are of unit variance, that is: $\mathrm{E}\{s_i^2\} = 1$.

We are thus not able to determine the *energy* of the independent components. For most applications however, this is not a big limitation. Notice that the *sign* of the independent components is still ambiguous. Also this turns out to be a minor limitation.

We cannot determine the sign of the independent components.

Now we mentioned this restriction about the variance, we can see a similar ambiguity of the ICA model. Suppose we multiply both the left and the right side of (16) with a permutation matrix **P**:

$$\mathbf{Ps} = \mathbf{PWx}\,. \tag{30}$$

This permutation matrix interchanges position of the independent components. We can see by:

$$\mathbf{s} = \mathbf{P}^{-1}\mathbf{PWx} = \mathbf{Wx}\,, \tag{31}$$

that the model remains the same after this multiplication. This means that we cannot determine the order of the ICs. We cannot put a restriction on the model so that we *can* determine the order. Fortunately, in most applications it is no big difficulty that the ICs are determined in an unknown order. We thus have to keep in mind:

We cannot determine the order of the independent components.

With the components of the random vector to have unit variance, they are even *orthonormal* instead of only orthogonal. If we put all these covariances and variances between all the components in the random vector together, we get the *correlation matrix*:

$$\mathbf{C}_\mathbf{x} = \mathbf{R}_\mathbf{x} = \mathbf{I}\,. \tag{32}$$

This matrix states that the random vector **x** is *white*, or *sphered*.

We can see that our random vector **x** is not white, because its elements are not orthonormal, i.e. the principle axes of the distribution are not of unit variance and do not make an angle of 90°. A standard and frequently used method to *whiten* or *sphering* data is principle component analysis, or PCA. For a detailed description about PCA please refer to /11/. Here, we will only use the results of PCA to whiten the data.

To whiten a random vector **x** we will search for a linear transformation

$$\mathbf{z} = \mathbf{Vx}\,, \tag{33}$$

which results in a new random vector **z** that is white. The matrix **V** is then called a whitening matrix. One result of PCA is that we can use the *eigenvalue decomposition* (EVD) of the covariance matrix to get a *whitening matrix*.
The EVD of the covariance matrix of **x** is:

$$\mathbf{C}_\mathbf{x} = \mathrm{E}\{\mathbf{x}\mathbf{x}^\mathbf{T}\} = \mathbf{EDE}^\mathbf{T}\,. \tag{34}$$

In this formula, **E** is the orthogonal matrix of *eigenvectors* of $\mathbf{C}_\mathbf{x}$ and **D** is the diagonal matrix of its *eigenvalues*, so that the top-left most eigenvalue in **D** corresponds to the first eigenvector in **E**, the one to the right of it to the second eigenvector, and so on. A whitening matrix can now be obtained by:

$$\mathbf{V} = \mathbf{ED}^{-1/2}\mathbf{E}^\mathbf{T}\,, \tag{35}$$

as can be seen quite easily when we substitute (35) and (36) into (34):

$$\mathrm{E}\{\mathbf{z}\mathbf{z}^T\} = \mathbf{V}\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}\mathbf{V}^T = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{E}\mathbf{D}\mathbf{E}^T\mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T = \mathbf{I}\,. \tag{36}$$

When we compute the covariance matrix of the data in Figure 5, perform the EVD and compute from this the whitening matrix, we can left multiply it with **x** to get the whitened **z**. The result is also shown in the figure below.

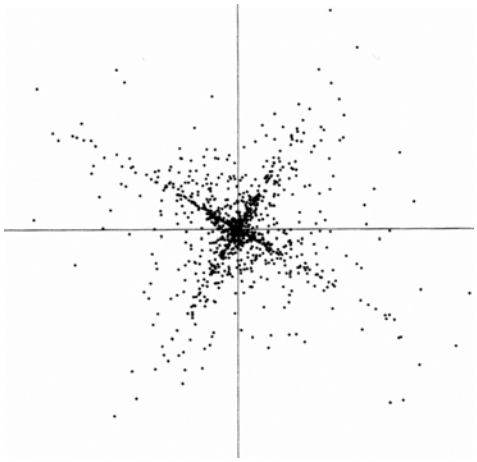


Figure 6. Whitened mixture of two Laplacian distributions.

When we compare this figure with Figure 4, where the original independent sources are displayed, we see that a *rotation* is all what is left in order to solve the ICA problem. In matrix terms, a rotation is equivalent to a left-multiplication by an orthogonal matrix.

After whitening the observations, an orthogonal transformation is what's left in order to solve the ICA problem.

We can also see that whitening the data doesn't solve the ICA problem. To see this, consider an orthogonal transformation U of the whitened data **z**:

$$\mathbf{y} = \mathbf{U}\mathbf{z}\,. \tag{37}$$

If we now compute the covariance matrix of **y** we see that it is also white.

$$\mathrm{E}\{\mathbf{y}\mathbf{y}^T\} = \mathbf{E}\{\mathbf{U}\mathbf{z}\mathbf{z}^T\mathbf{U}^T\} = \mathbf{U}\mathbf{I}\mathbf{U}^T = \mathbf{I}\,. \tag{38}$$

This means that whitening the data can result in infinitive many white distributions; all separated by a rotational angle from each other.
If we now look at our model after this whitening, it can be seen that

$$\mathbf{s} = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{V}^{-1}\mathbf{z} = \mathbf{W}\mathbf{V}^T\mathbf{z} = \tilde{\mathbf{W}}\mathbf{z}\,. \tag{39}$$

We thus have to estimate a different mixing matrix $\tilde{\mathbf{W}}$ to solve the ICA problem. Intuitively, by looking at the figures, we already concluded that whitening brought us a step closer to the solution. We can also see this mathematically, by noticing that $\tilde{\mathbf{W}}$ is also orthogonal:

$$\mathrm{E}\{\mathbf{s}\mathbf{s}^T\} = \tilde{\mathbf{W}}\mathrm{E}\{\mathbf{z}\mathbf{z}^T\}\tilde{\mathbf{W}}^T = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{I}. \qquad (40)$$

In the original case, we had to estimate n^2 parameters of the *n x n* matrix **W**. For orthogonal matrices of this size, we only need to estimate *n(n-1)/2* parameters. In our two-dimensional case, this means that we first needed to estimate all of the 4 parameters, where we now only need to estimate 1 parameter. This is exactly the angle that separates us from the solution.

Whitening the observations reduces the complexity of the estimation of the independent components.

The question is now of course, how to estimate this angle, or in general the last *n(n-1)/2* parameters? We now have exploited the information stored in the first two statistics, so we have to go to higher statistics for this. We will do this later on. First let us look at the case of subgaussians to see how all what we just found applies to that class of pdfs. This is also a good opportunity to summarize all what we have achieved so far.

2.4.4 Subgaussian distributions

We will now generate two independent subgaussian distributions. Since the uniform distribution is a popular example of a subgaussian distribution, we will generate the independent random vectors from this pdf. Their scatter-plot is displayed in Figure 7.

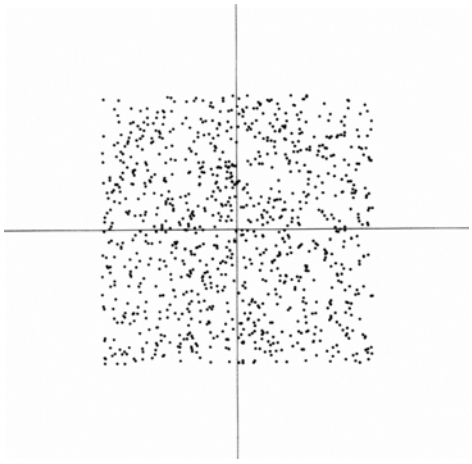


Figure 7. Two independent uniform distributions.

Suppose again these are the components in **s**. To get the components in **x** we have to multiply the independent components with a mixing matrix **A**. The resulted mixture is displayed below.

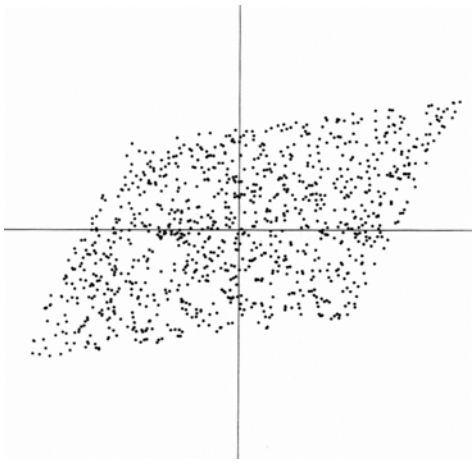


Figure 8. Linear mixture of two uniform distributions.

We see again that the components of **x** are not independent, since knowing this one gives information about the possible values the other can take. We also see very clearly that they are correlated, since an increase in one of the components generally implies also an increase of the other component.
We will do the same as we did with the supergaussian data, i.e. exploiting the information stored in the statistics of the observed **x**. The data is already zero-mean, but when this was not the case, it could be made zero-mean by just subtracting its estimated mean. The correlation of the distribution can be removed by whitening **x**. The result is Figure 9.

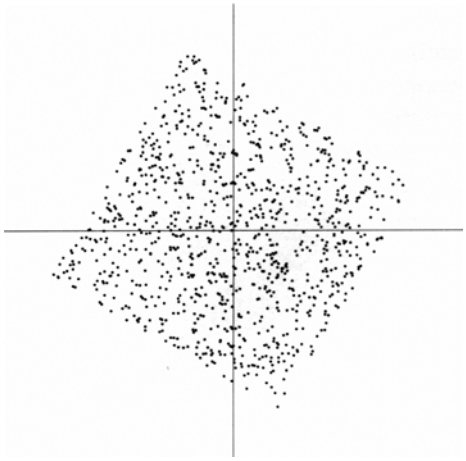


Figure 9. Whitened mixture of two independent distributions.

Again we see that all we need to do now is to determine the orthogonal transformation that makes the components of the data independent. To achieve this, we need information from higher order statistics, like we concluded in the supergaussian case.

There is one more pdf we promised to consider. This is the Gaussian distribution. When we apply what we have achieved so far, we will see that we gain some indications how to proceed.

2.4.5 Gaussian distributions

We will thus generate samples of two independent Gaussian distributions. Their scatter-plot is displayed in Figure 10.

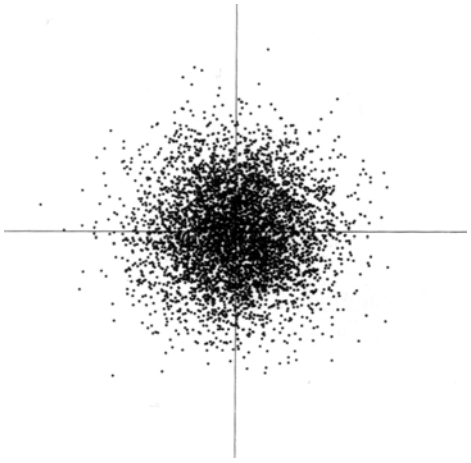


Figure 10. Two Gaussian uniform distributions.

Suppose again these are the components in **s**. To get the components in **x** we have to multiply the independent components with a mixing matrix **A**.

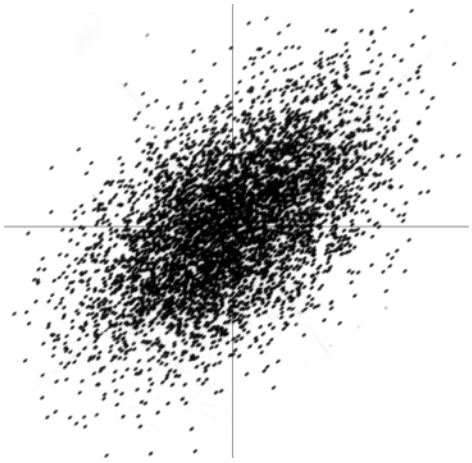


Figure 11. Linear mixture of two Gaussian distributions.

We see again that the components of **x** are not independent and correlated.
We will again try to exploit the information stored in the statistics of the observed **x**. The data is already zero-mean, so we can proceed instantly with the second order statistics. The correlation of the distribution can be removed by whitening **x**. The result is Figure 12.

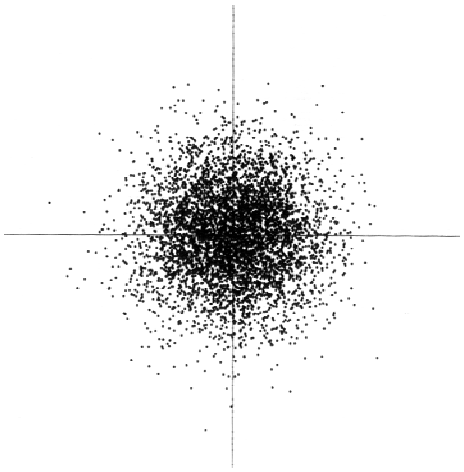


Figure 12. Whitened mixture of two Gaussian distributions.

We now have the astonishing result that this figure looks exactly the same as Figure 10. We can perform any rotation, but keep the same distribution. There is thus no way to distinguish the case of independent components from any other whitened form of the data. We can prove mathematically that our observation is correct.
When we write down the joint pdf of two independent Gaussian variables s_1 and s_2:

$$p(s_1, s_2) = \frac{1}{2\pi}\exp\left(-\frac{s_1^2 + s_2^2}{2}\right) = \frac{1}{2\pi}\exp(-\frac{\|\mathbf{s}\|^2}{2}), \tag{41}$$

we can see by using the formula of transforming pdfs that when we multiply this with the mixing matrix **A** we get the following joint pdf for the mixture of x_1 and x_2:

$$p(x_1, x_2) = \frac{1}{2\pi}\exp(-\frac{\|\mathbf{A}^T\mathbf{x}\|^2}{2})\left|\det \mathbf{A}^T\right|. \tag{42}$$

We know that $\|\mathbf{A}^T\mathbf{x}\|^2 = \|\mathbf{x}\|^2$ and $\left|\det \mathbf{A}^T\right| = 1$ because **A** is orthogonal, so that

$$p(x_1, x_2) = \frac{1}{2\pi}\exp(-\frac{\|\mathbf{x}\|^2}{2}). \tag{43}$$

We don't see the mixing matrix **A** anymore in this distribution and **x** could just as well be **s**, but also infinite other orthonormal random vectors. We can thus conclude that it is impossible for ICA to find the independent components of a Gaussian distribution. We therefore make another restriction to the model:

Restriction 4: The independent components must have nongaussian distributions.

This may seem to be a very serious restriction. However, many observations are nongaussian and we will also see later on that we can relax this restriction a little. What is more important is that this result gives us a clue on how to proceed. Namely, if we can

find some way to measure how close a certain distribution is to the Gaussian one, we can maybe utilise this to determine the important last orthogonal transformation. We will see very soon how we can actually do this.

One important remark has to be made before we proceed. It turns out that *nongaussianity* is the only restriction to the observed data **x**. This means that the components of **x** can be of any distribution, not only sub- or supergaussian. In /6/, it is said that the Gaussian distribution is the least structured pdf. Intuitively we know that it is exactly the structure of the data that helps us in estimating the independent components. So we can also feel that any distribution that has more structure than the Gaussian pdf is allowed for the observation vector.

2.5 Exploiting nongaussianity

We already noted, that nongaussianity of the independent components may be a way to estimate them. For this, it is necessary for the observations to be more Gaussian than their independent components. This is indeed the case according to *central limit theorem*, since theorem states that in the limit, a sum of independent distributions approximates the Gaussian distribution. As the observations are considered to be a weighted sum of their independent components, their distribution is closer to the Gaussian distribution than their ICs.

It has already been noted that higher order statistics are necessary to solve the ICA problem. Especially the kurtosis turns out to be very useful. The kurtosis was defined in (21) as:

$$\text{kurt} = E\{x^4\} - 3[E\{x^2\}]^2 . \qquad (44)$$

The Gaussian pdf is completely described by its first and second order statistics. Higher order statistics exist for a Gaussian random variable, but are expressed using second order statistics. The fourth moment of a Gaussian equals $3(E\{x^2\})^2$, so it can easily be seen that the kurtosis of a Gaussian equals zero. A kurtosis of zero only appears for Gaussian random variables and a few others that are seldom encountered in practice. The kurtosis of a random variable can be either positive or negative. Random variables with a negative kurtosis are called subgaussian and with a positive kurtosis supergaussian.

Since the independent components are maximal nongaussian compared to their mixtures, estimating them consists of finding a rotation that maximizes the nongaussianity of the mixtures. How this exactly works and how different algorithms can be constructed based on these principles can be found in /6/. Using the kurtosis for measuring the nongaussianity of a signal is an intuitive way, but has some drawbacks. The major drawback is that it is very sensitive to *outliers* in the data. An outlier is a value that is much larger than the rest of the data. Only one outlier can thus dominate the kurtosis completely. Other measures of nongaussianity are available, in particular *negentropy*. Negentropy measures how close a random variable is compared to a Gaussian random variable with the same first and second order statistics in the sense of entropy. For more

information about entropy, negentropy and the use of it for the ICA problem, readers can refer to /6/ or /12/.

Higher order statistics, like the kurtosis and negentropy can be used for a measurement of nongaussianity, which is the key in solving the ICA problem.

ICA algorithms can be derived from other points of view, like *estimation theory*, *information theory* and *tensors*. Since this report is only here to give the reader a practical sense of how ICA really works, this is out of its scope. Again, readers can refer to /6/ for more information.

2.6 Estimating more than one Independent Component

We now have seen how it is possible to estimate one of the independent components of a mixture of independent sources. In the two dimensional case, this solves the entire problem, since the second IC is perpendicular to the first one. Of course, this leaves the ambiguity of the sign, but we already concluded that this is a limitation of ICA.
When there are more ICs, thus for dimensions higher than two, we have to estimate more ICs. Running the estimation algorithm just more times is not reliable, since there is no assurance that the other ICs will be found, instead of the ones that are already estimated. However, there is still the property of orthogonality that can be utilized. If we make sure that during the estimation of a new IC, it is constantly orthogonalized with respect to the ICs that are already estimated, we are sure that it will result in really a new one.

In order to determine all the ICs, we have to make sure that they are constantly orthogonalized with respect to the ICs that are already found.

Only the last IC can be determined straight from the others and their orthogonality. Here we see that this last IC can also have a Gaussian distribution, since nongaussianity is not used for its estimation. We can thus relief the fourth restriction as following:

Restriction 4 (relieved): All the independent components, except for at most one, must have nongaussian distributions.

2.7 Summary and concluding remarks

We saw in this chapter how the ICA problem can be solved step by step. The (noise-free) Independent Component Analysis (ICA) model states that there are some measurements that are a linear mixture of a certain number of independent sources as shown by:

$$\mathbf{x} = \mathbf{A}\mathbf{s}\,, \tag{45}$$

where **x** is the random vector of observations, **A** is the mixing matrix and **s** is the random vector of independent sources. This model is a latent variable generative model, which is not solvable in its closed form. With restrictions on the model, it is possible to estimate the independent sources, which are called Independent Components (ICs) and the mixing

matrix. ICA is performed starting by centering and whitening the observations. This removes the effect of the first and second order statistics. After whitening, the complexity of the problem is reduced to the estimation of an orthogonal mixing matrix, which has much less free parameters than the original mixing matrix. Nongaussianity turned out to be the key in solving the ICA problem. Higher order statistics are necessary for a measure of nongaussianity. Maximizing the nongaussianity then will result in finding the ICs. Examples of higher order statistics are the kurtosis and negentropy. Measures for nongaussianity can also be derived from estimation, information and tensor theory. After estimating one of the ICs, the next one can be estimated under the constraint that it is orthogonal to the ICs already found. The last one follows directly from the other ICs. This is why at most one of the ICs can be Gaussian.

The most important things to remember were highlighted and will return in chapter 4. One has to keep in mind that this chapter was mainly to have an idea about how ICA is possible, which is probably enough for the use of ICA in engineering. In the next chapter we will see how ICA can be applied in real world engineering situations. Especially possible pitfalls will be highlighted. In the concluding chapter 4, these highlights will return as well. A guideline for the usage of ICA will be extracted from them.

3 ICA APPLIED

3.1 Introduction

In the previous chapter we went through the process of understanding ICA from problem to solution. This is very important, although we will probably never have to implement the ICA algorithm ourselves. As we also know, there exist many ICA algorithms, of which maybe the most favourite one is FastICA. We will use the Matlab implementation of FastICA /7/. This software package can be downloaded from /9/. Although we don't need to know all the ins and outs of the algorithm, we still need to understand what is going on. In the previous chapter, we treated the theory; in this chapter we will look at two examples of applying ICA to real world problems. The main objective is not to solve the problems, but to learn from mistakes and successes. We will keep track of these to form a guideline for the usage of ICA in the next chapter.

3.2 Screw insertion analysis

3.2.1 Introduction

In this section we analyze the process data from screw insertion. At first, when the data was delivered for this analysis, there was no sign that indicated the origin of the data. Of course, the ICA algorithm does not have to know the origin of the data; it just tries to find the independent components of it. However, the quality of what comes out of the algorithm does not only depend on the algorithm, but also on what is used for input. It is up to the researcher to determine how the data should be pre-processed before the ICA algorithm is applied. For this, it is highly necessary to understand the data very well. This constitutes in visualising the data and in knowing some of the basic properties like mean, variance and missing values.
Therefore the first important step can already be marked:

Make sure that you have a proper understanding of the data by visualization and calculation of the mean and variance.

This affects the way you would preprocess and analyze the data, as well as how you would interpret the output.

For the analysis of the screw insertion process, three datasets are available. The datasets were presented as three plain matrices each of which contains rows that represent the behaviour of a certain variable of the process over time. There was no description about the origin of the data, so at first this process is unknown. It is given that all the data come from the same process, but as different realizations. From two of the datasets it is known that the data comes from a successful process. From one it is known that it contains only

failures. The purpose is to find features from the failure-dataset that indicate the failure, and hopefully its cause.

From what we just stated, it is most useful to interview the owner of the dataset to get to know the details about the process. In this case, important questions are:

- From what kind of process originate the datasets?
- What is the variable, which behaviour is described versus the time?
- What makes a process to fail?

It turned out that the process was about inserting screws to join two pieces of a product. Most of the time this processing stage is successful, but sometimes there is a failure that results in an unreliable connection of the pieces. It is desired to analyse if the insertion of the screw was successful based on certain measurements. It turns out that information necessary for the solution lies in the torque of the screwdriver during the insertion of the screw in a hole. The variable in the datasets represents this torque over time during this process. What's the reason of the failures is not yet known, but it is hoped that it is hidden in one or more of the independent components that generate the torque. At least, it is hoped that some features can be found from the independent components that help us distinguish between the success and failure case. For more information about monitoring automated screw insertion processes, readers can consult for example /14/ and /15/.

3.2.2 Data visualisation

It is always a good way to start with just looking at the data. The Matlab function `icaplot`, from the FastICA toolbox /7/, provides a good way of plotting the different signals in the dataset. The type of the plot should be set to `'complot'`, since in this mode; all the signals will be plotted on the same axes, which is necessary for comparing them by sight. One difficulty remains; the signals were zero padded to make them all of the same size. Since the mean and the variance of all the signals are in the order of 3, and 10^{-6} respectively, the zeros totally ruined the scale. Therefore it was a good idea to replace those zeros at the end of each signal by the mean of that signal. In this way, `icaplot` scaled the signals in the right way.

Below, two plots are displayed. The first one contains seven signals of a corresponding successful insertion and the second of corresponding failed insertions.

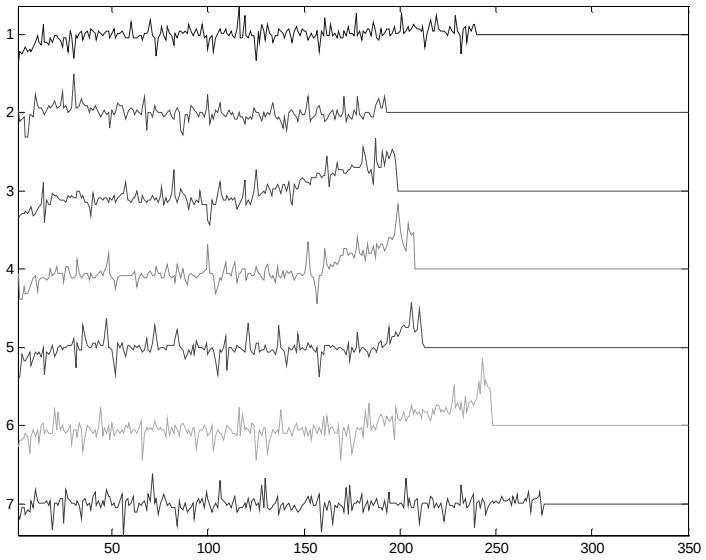


Figure 13. Seven signals of a successful process.

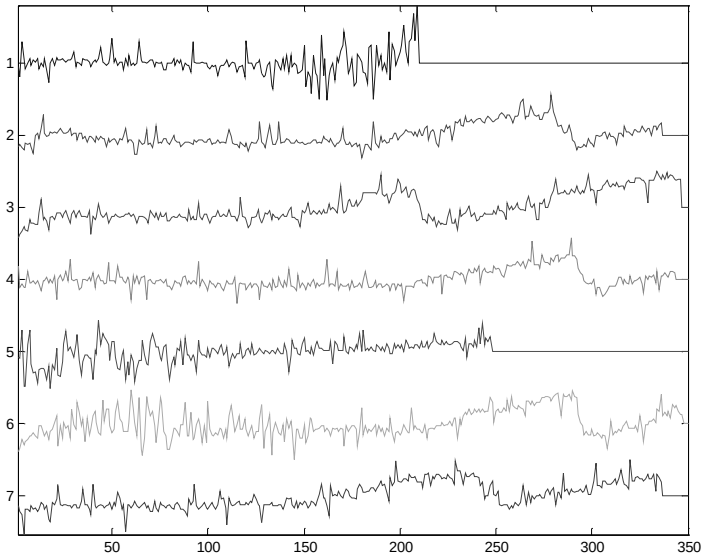


Figure 14. Seven signals of a failed process.

From these two figures, it was concluded that it would be very hard to draw a clear distinction between a succeeded process and a failed one. It was hoped that ICA would be able to find independent components that can make this distinction.

For the purpose of visualisation, zeros at the end of the signals were replaced by the mean value of its corresponding signal. In fact, these zeros weren't part of the signal at all. It would thus make no sense in analysing the signals as they were presented in the dataset. ICA would try to find independent components that also describe the zeros at the end.

Instead of zeros, we could just as well have extended the signals by sine waves. In that case, ICA would have tried to find independent components that include these sine waves as well. Thus, the resulted independent components make no sense at all, since they don't originate from just the measured signals.
All the ends were truncated at the same point. We then created datasets of pure data, with the ends synchronised. From this we will highlight another important step:

Make sure that the data only consists of the signals without artificially introduced values, like zero-padding.

3.2.3 FastICA

Using the FastICA algorithm in Matlab is an easy and efficient way of using ICA on data. One can perform FastICA on the data, without further restrictions. The algorithm then returns a number of independent components, as much as the dimension of the dataset, i.e. the number of measurements. FastICA first whitens the data before estimating the independent components. This is useful, because whitening transforms the mixing matrix in an orthogonal matrix, which has much less elements to be estimated. Furthermore, whitening gives us the eigenvalues of the data, which provide a measure of determining the importance of certain directions of the data. Discarding the smallest eigenvalues is therefore a way of reducing the dimension. In FastICA it is possible to give as an optional argument the number of eigenvalues to be retained in the estimation of the independent components. This value should not be too large, since then unimportant independent components are estimated, but also not too small, since the signal could then not be represented in an accurate way anymore.
To compare the ICs from the datasets, we need to estimate for all the datasets the same number of ICs. Since there are only seven measurements in the dataset that represent the unsuccessful processes, the maximum number of ICs is seven. We can compute the intrinsic dimension of the successful and failed data, by looking at how well the data is reconstructed using only a limited number of PCA components. The percentage of remaining variance is displayed versus the number of dimensions in Figure 15.

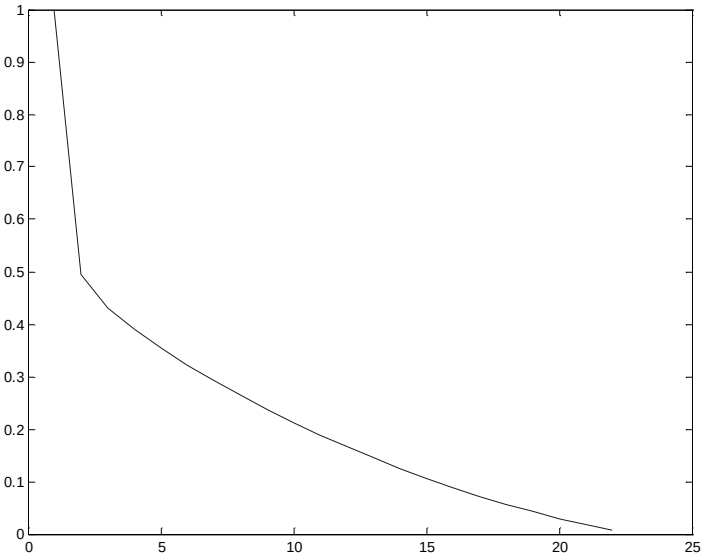


Figure 15. Remaining variance vs. number of PCA components to recontruct the data from successful processes.

We see that in order to retain 80% of the variance, we need at least 10 dimensions. Since we just determined that we could only take a maximum of 7 dimensions, this was used instead.
FastICA was performed on the data of the successful and the failed process. Figure 16 shows the results.

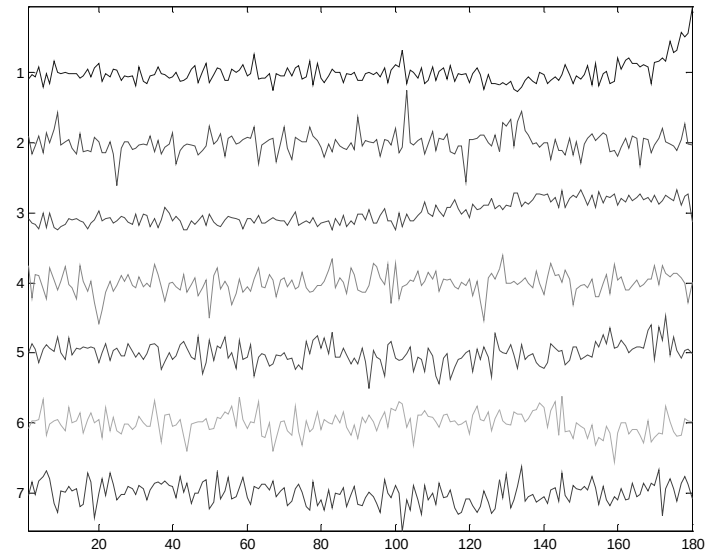


Figure 16. 7 ICs of data from successful processes.

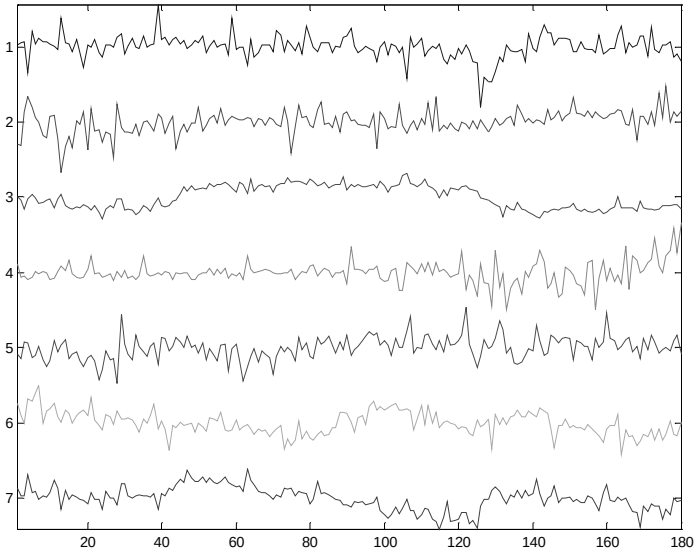


Figure 17. 7 ICs of data from failed processes.

3.2.4 Analysis of the results

We have to keep in mind that we don't know anything about the order of the ICs neither about their relative strength. Also the sign of the ICs is unknown. When we look at the ICs in both figures we see mostly noisy signals. Only a few signals seem to have nice shapes. The first one in Figure 16 rises at the end. This might indicate an increase of the torque because the screw tightens in the material. The third IC in the same figure shows also a small increase after some time. From Figure 17, maybe only the third and fourth signals are worth mentioning. In the third IC, an increase in the middle part can be seen, where in the fourth IC, the beginning is quite flat following by a noisy increase at the end.
Of course, because we cannot say anything about the sign of the ICs, an increase can also mean a decrease and visa versa.

3.2.5 Conclusions

Considering the datasets and the way they are obtained, we might wonder if finding ICs makes sense at all. From the ICA model, we assumed that there are some independent sources that result after a linear transformation in the observations. In the case of this application, there is no such generation of the observations. Each observation comes from different process conditions, so probably is a linear combination of different independent components. For observations that follow the ICA model, we should have had different measurements of one process. Then, each measurement is a component of the observation vector, which are all linear combinations of the same ICs.
Still, FastICA finds independent components to our solution. This is because ICA assumes that the observed signals follow the ICA model, no matter if that's really true or

not. In our case, the data doesn't follow the ICA model, so we shouldn't also expect to explain the ICs in a physical way. The found ICs are features of this particular dataset. Probably, if one of the observations was missing, or another one added the ICs would be different.
In the case of observations from a successful process, we see already from the data that most of them end with an increase of the signal value. We see this increase indeed as a feature in the first component. However, not every observed signal ends with an increase of its value. This might be because of the different process conditions that result in the observations. For a decent analysis of this problem, we would need to have the data separated for each process condition. We can then analyse the difference in IC features between successful and failed processes. It is the question whether to apply ICA, because most likely the found feature can be seen right away from the observed data, like the increase at the end.
An important thing to remember is thus:

As long as the observations are nongaussian, ICA will find independent components that are maximized in their nongaussianity. This doesn't mean that these ICs make sense in a physical way.

We therefore have to ask ourselves the question if we expect that our data follow the ICA model before we try to apply ICA. If this is the case, it is worth trying to apply ICA, because the found ICs can be interpreted as really the independent sources of the observations. If this is not the case, the ICs that ICA finds, if they can be found at all, have no meaningful interpretation.

3.3 Paper Break analysis

3.3.1 Introduction

In the paper industry, long sheets of paper move rapidly through a number of process steps. The faster, the more paper can be produced. However, moving the paper too fast may result in a break of the paper, forcing the process to stop. After the paper is put in place properly again, the process can continue. This interval of non-productivity in combination with the costs of stopping and restarting all the machines along the process line, make the occurrence of such a break an expensive event. It would be highly desirable to be able to predict when a break is at hand, in order to take appropriate actions to avoid the upcoming break to occur.

Ten years of research on this problem is already behind us, with as result a working break-prediction system /1/. This system, however, does not always predict as well as one would like. Therefore more research is still desired.

In this paragraph, ICA is applied as a method to predict a paper break. ICA tries to find the independent components of which the sensor data is built from. The idea is that some of these components might reveal the underlying reason for a break. Perhaps it can detect

an unusual change in some components that indicate an upcoming break in advance. ICA is a very promising method due to its intuitiveness and many good results in other applications. As far as we know, ICA has never been applied to this problem before.

3.3.2 Description of the dataset

ICA is a data-driven method, meaning that it needs process data to extract its information from. The data that is provided for this analysis consists of three datasets, representing the 'normal'- case, a case of only 'two breaks' and a case of 'many breaks'. In the dataset it is marked when a break did occur.
Each dataset is represented in an .xls spreadsheet file. The rows represent time intervals of one minute. There is one column that denotes when a break occurs and 24 columns of sensor data. The meaning of these columns is explained in Table 1.

Table 1. Explanation of the columns in the dataset.

Column	Quantity	Unit
1	Mechanical pulp feed	1/s
2	Chemical pulp feed	1/s
3	Broke feed	1/s
4	Filler to centrifugal cleaner pump	1/s
5	Filler 1 flow	1/s
6	Filler 2 flow	1/s
7	Mechanical pulp feed consistency	%
8	Chemical pulp feed consistency	%
9	Broke to broke screen consistency	%
10	Chemical pulp percentage	%
11	Broke percentage	%
12	Bleached mechanical pulp percentage	%
13	Machine pulp consistency	%
14	Chemical pulp freeness	Ml/min
15	Chemical pulp pH	pH
16	Chemical pulp conductivity	mS/m
17	Mechanical pulp1 tower pH	pH
18	Mechanical pulp2 tower pH	pH
19	Mechanical pulp conductivity	mS/m
20	Broke pH	pH
21	Broke conductivity	mS/m
22	White water pH	pH
23	White water tower temperature	°C
24	Mechanical pulp proportioning chest level	%

We like to convert this data to an appropriate form for MATLAB, since then we can use the FastICA toolbox, which is a useful and fast implementation of the ICA method. The script `paperpulp_read` does this job and returns the three datasets as individual matrices. Now, the rows and columns are also interchanged, because this is the correct form for FastICA input. The number of rows is now in all three datasets equal to 25.

Still, the data is not yet ready for the application of ICA, since it contains missing values mainly at the end of the datasets. This is simply solved by removing the columns

containing a missing value. This is justified because only a small amount of data is removed and there is no data removed from the middle of the datasets. The function `clean_set` performs these operations. Now, two of the datasets are in a clean form for further processing. Only the 'normal' dataset still contains missing values. Looking at the row that indicates when a break occurs, one can conclude that the actual 'normal' data is in between some breaks. The missing values occur at a moment when such a break occurs. We can easily select only the middle part of the dataset as the new 'normal' dataset, also removing the missing values at the same time.

The most important thing to highlight from this is the following:

Make sure you have the data in the right format for further processing. Take care of the missing data and be careful with the orientation of the data set.

3.3.3 How ICA can be applied

As a default, ICA will find just as many independent components as there are rows in **x**. One standard pre-processing step before the actual FastICA algorithm is performed is whitening the data. Whitening means making the data as uncorrelated as possible. FastICA now only needs to find an orthogonal transformation that makes the rows of the data independent. From the whitened data, only the components that contains the most variance (or information) can be selected by PCA, therefore reducing the dimensionality of the data and facilitate the remaining steps. The first step is therefore to determine the intrinsic dimensionality of the data. This value can then be fed to the FastICA algorithm as an extra (optional) variable.

The idea of the break detection is to buffer the real-time sensor data in a fixed sized buffer. On the start-up one has to wait until the buffer is filled before doing any analyses on it. Once it is filled, ICA can be applied using the intrinsic dimension as limit on the number of computed independent components. These independent components can then be put in a classifier in order to determine the possibility of an upcoming break. Each following minute, new sensor data is available. This data is fed to the buffer, while discarding the oldest values from the buffer. ICA can be applied again, followed again by outputting the possibility of an upcoming break.

Some parameters need to be determined for the system to work properly. These are the size of the buffer and the step-size. Each of these will be treated separately in the following paragraphs. But first we will find out if reducing the number of components can reduce the complexity of the problem.

3.3.4 Intrinsic dimensionality of the data

As already mentioned just before, we can use the intrinsic dimension of the data to reduce the complexity of the problem. To determine this value, we can try to use two different methods.

PCA

The first one is PCA (principle component analysis), which assumes that the relation between the different features is linear. From the covariance matrix, the eigenvalues and their corresponding eigenvectors are computed. The eigenvectors are the directions of the uncorrelated components of the data. The corresponding eigenvalues indicate the amount of variance in that direction. We can select some of the (largest) eigenvalues to reduce the number of dimensions. We have to make sure that we still capture enough variance to estimate the data properly. We can also calculate the *mean squared error* (MSE) of the reconstruction very easily from the eigenvalues, since we know that the sum of the remaining eigenvalues after selecting the p-highest eigenvalues for representing the data equals the MSE.

In order to get the most robust measure of the intrinsic dimensionality, we perform PCA on a fixed sized buffer of the 'normal' dataset, which we slide along the data, with steps of half the size of the buffer. A buffer of size 24, applied to data of 800 time instants therefore consists of $\lfloor 800/24 \rfloor \times 2 - 1 = 65$ buffers. In each buffer we compute the remaining variance of a representation of one up to ten dimensions. Of the *15x10*-sized matrix that is produced, we then compute for each column the mean and the variance and plot this result as an error bar. This error bar is created using the PRTools toolbox for Matlab /10/. For describing 95% of the variance of the data we have to take 5 dimensions. If we repeat this experiment for buffer sizes of 48, 72 and 96 it turns out that also 5 dimensions are required to describe 95% of the variance of the data.

The MSE of describing the data with 5 dimensions for each of these buffer sizes has been put into Table 2.

Table 2. Mean and variance of the MSE over all buffers corresponding to different buffer sizes.

Buffer-size	Mean MSE	Variance MSE
24	0.3767	0.0315
48	0.5924	0.0615
72	0.7646	0.1328
96	0.8974	0.2868

Later on we will use this to determine the best choice for the buffer size.

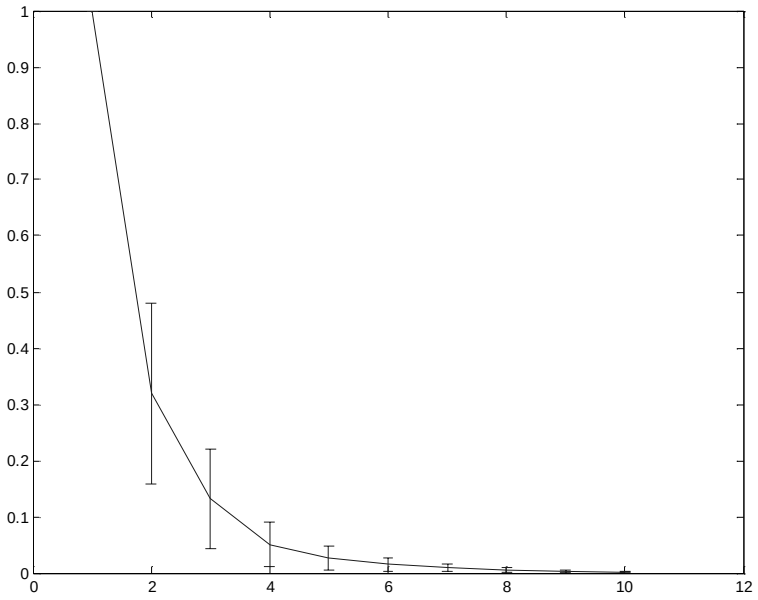


Figure 18. PCA buffer size of 24; error bar of the remaining variance vs. number of dimensions.

Non-linear factor analysis

The second method we can use to determine the intrinsic dimensionality of the data is *non-linear factor analysis* (or NLFA). It finds a nonlinear representation of the data. An algorithm that finds this representation in MATLAB is provided with the NLFA-toolbox /8/. It uses an Artificial Neural Network (ANN) with one hidden layer to compute the components. However, there are some major drawbacks of this method, which may decrease the applicability of the method.

For a network to be fed with the data that is available (of size 24x800), we need a (fully connected) ANN with 24 neurons in the input-layer, much more than 24 in the hidden-layer, and about 5 in the output layer. If we take for the hidden layer 48 neurons (2 times the number of neurons in the input-layer), we have a total number of free parameters of *24x48 + 48x5 = 1392*. A rule of thumb says that if you want to train this network, you need to have at least 10 times as much data samples than free parameters to avoid overtraining, which results in the need to have at least 13920 data samples in out case. If we would use all the data we have from the ´normal´ case, we would have *24x800=19200* samples. This is enough, but still the resulting error bar shows too much uncertainty to determine the intrinsic dimensionality. Also it is not possible to use buffers, as we did before with PCA, because then there are way too few data samples available.

However, there is one thing we can do. This is to reduce the number of features by consulting expert knowledge. An expert can give his view on which features he expects to have the most significant influence on the breaks. The expert selected the following features:

Table 3. Selected features.

New column	Old column	Quantity	Unit
1	3	Broke feed	1/s
2	15	Chemical pulp pH	pH
3	17	Mechanical pulp1 tower pH	pH
4	18	Mechanical pulp2 tower pH	pH
5	20	Broke pH	pH
6	22	White water pH	pH
7	23	White water tower temperature	°C

We can now try again to perform NLFA on the data with only these features. Since we now need 7 input-neurons, we only have to use 14 neurons in the hidden-layer. Still we have too few data samples to use the buffers. We only can have a look at the intrinsic dimensionality of the whole 'normal'-case data.

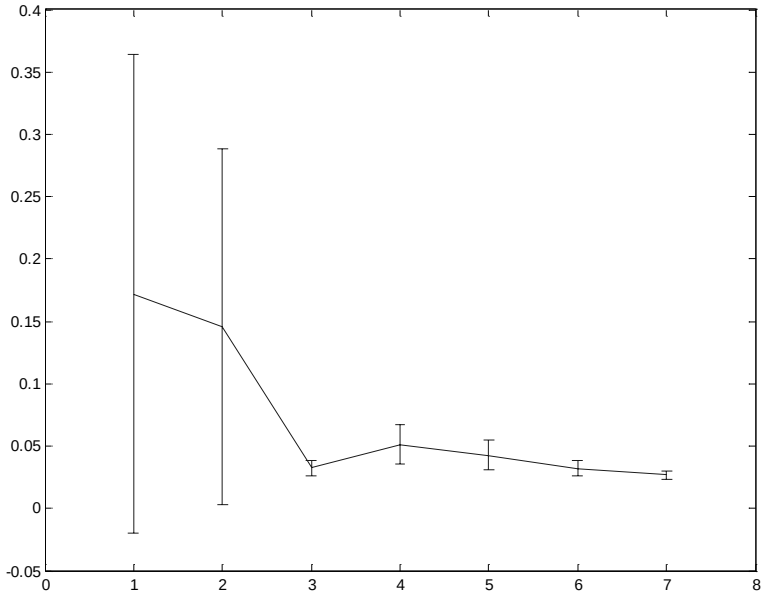


Figure 19. The MSE vs. the dimensionality by NLFA.

We see that a minimum is reached for three dimensions. To compare this with PCA we also have to perform the PCA on the same data. Because of better visualisation, this plot is zoomed in starting with 2 dimensions.

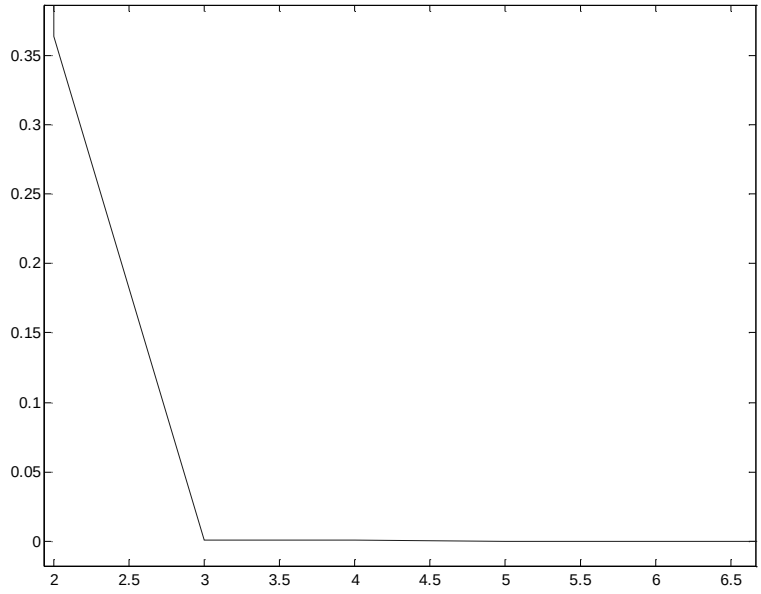


Figure 20. The MSE vs. the dimensionality by PCA.

We can clearly see that three dimensions are enough to describe the data with only seven features. PCA performs the best. To check how this works for different buffer sizes we computed the same graphs for a buffer size of 24, 48, 72 and 96. The same results hold, with the additional observation that even two dimensions might be enough to describe the data sufficiently well. The remaining variance for all is smaller than 5%. We calculate the MSE corresponding to a dimensionality of two and three using the different buffer sizes. The results are in Table 4.

Table 4. MSE for both two and three dimensions for different buffer sizes.

	Two dimensions		Three dimensions	
Buffer-size	**Mean MSE**	**Variance MSE**	**Mean MSE**	**Variance MSE**
24	0.0015	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
48	0.0059	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
72	0.0118	$< 10^{-4}$	0.0001	$< 10^{-4}$
96	0.0195	0.0002	0.0001	$< 10^{-4}$

In the FastICA algorithm, this means that we can provide it with the optional argument `lastEig` to be set on two.

3.3.5 Using all features for ICA

We now split this paragraph in two parts. First we will use all the features and after that we do the same with only the seven selected features as discussed above.

Buffer-size

In the case of only 5 independent components, we have a *5 x 5* mixing matrix to be estimated. Because we can reduce the number of free parameters in this matrix to *5x(5 - 1)/2=10* by first whitening the data, we need 100 data points to avoid overtraining. In this case we take the rule of thumb to use 10 data points for each free parameter. Since we have observation, consisting of 5 rows, we need 20 columns to get to 100 data points. Therefore a buffer size of 20 should be enough. It can be that 20 is still too less, because then the resolution of the ICs is only 20, which may be too small for making clear classification. It is important not to make the buffer-size too large, because this will result in a too long start-up in which the system is unable to make any predictions about the possibility of an upcoming break.
Comparing the MSE at a dimensionality of three at these four different buffer-sizes, as displayed in table 2, we can conclude that a buffer-size of 24 gives the smallest error. Since it was already favoured before, we from now on take this as the buffer-size.

Step-size

As already mentioned before, we also need to consider the best value for the step-size. It is important to keep the step-size as small as possible, because if you take too large steps, you could be too late to detect a break. Our intuition tells us that it is not possible to detect a break very long time in advance. The only reason to make the time-step higher than the minimum is because the system might not be fast enough to perform all the necessary computations in the given time. In this case this seems not to be a problem, since a fast computer can do all the necessary computations in only a few seconds, while new data arrives only each minute. We can therefore safely set the step-size to one minute.

ICA applied

Now we know with which parameter values we will use ICA, we have to determine a strategy to compute the possibility of an upcoming break. At first we will apply ICA in the fixed size buffer on the ‘normal’ dataset. Alternated with intermediate conclusions, we will continue with examining the ‘two-breaks’ and finally the ‘many-breaks’ dataset.

Normal

We don’t know yet where to look at when we get the independent components of a certain buffer. To have some idea, we positioned the buffer at some randomly chosen places to see what the components look like. Four of these results are shown below.

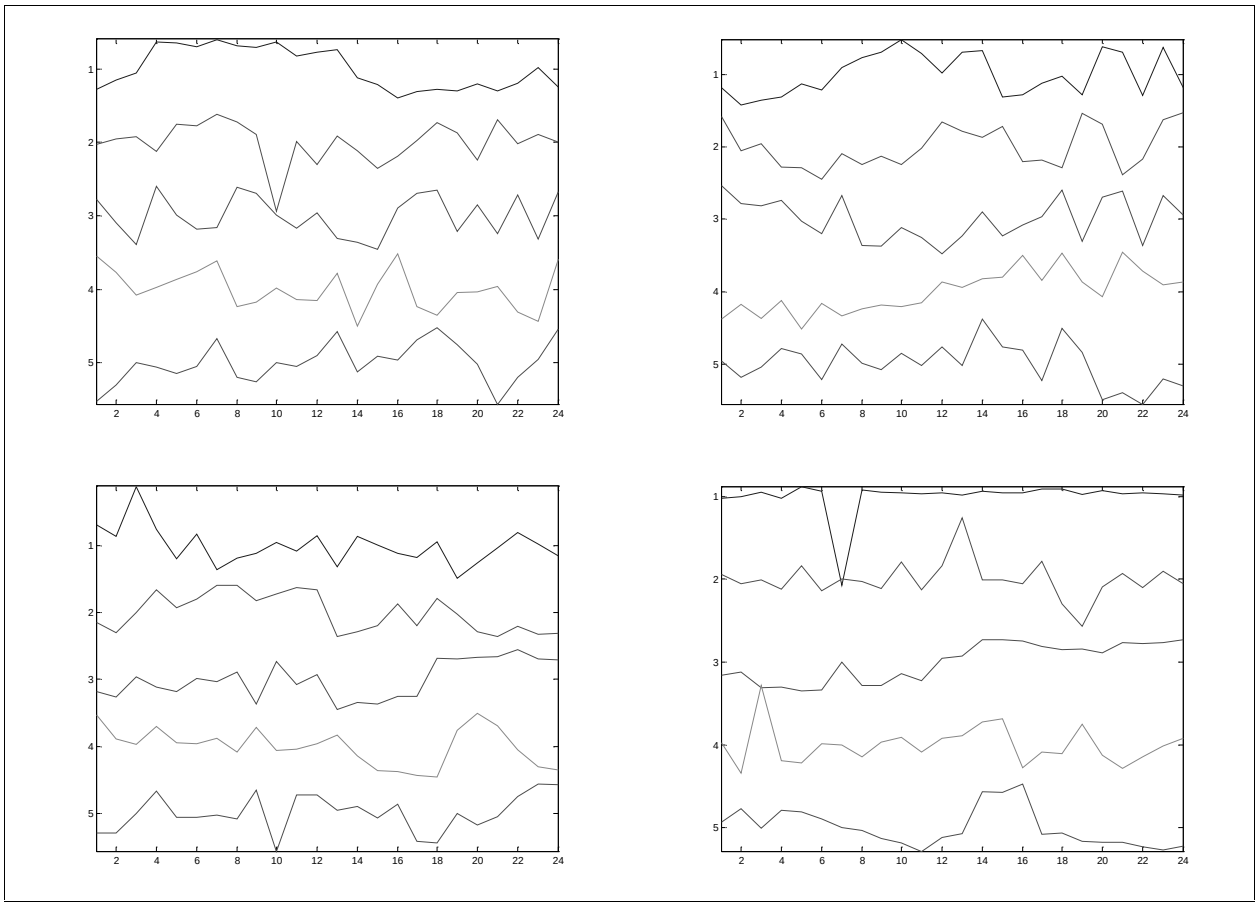


Figure 21. ICs at four randomly chosen buffers of size 24.

We have to remark that at some places ICA won't find ICs. We come back on that later on.
After looking at the figures, we cannot distinguish clear similarities. This indicates that during the normal process the shape of the sensor data changes all the time. Of course we cannot say anything about breaks, since they don't occur yet. We need the results from this analysis for comparison with the results of analysis of the following two datasets.

Two_breaks
In this dataset two breaks occur. We will define the buffer to be one and two minutes before each break, and analyse the ICs. It turns out that FastICA is not able to estimate the ICs. Perhaps this is because the buffer size is too small. We therefore try the same with a buffer-size of 48. These results are displayed in Figure 22 - 25. The first break occurs at time instant 242; the second at 1099.

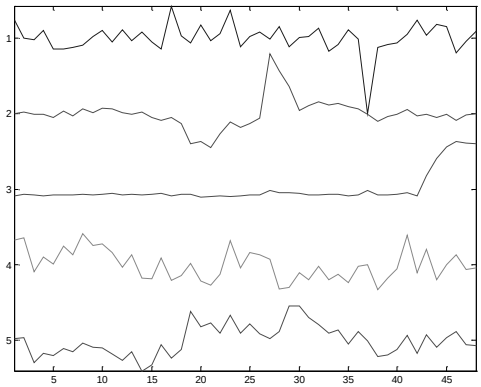


Figure 22. Buffer-size = 48; Endsample = 240.

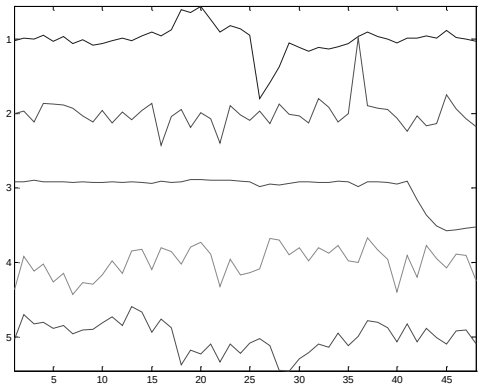


Figure 23. Buffer-size = 48; Endsample = 241.

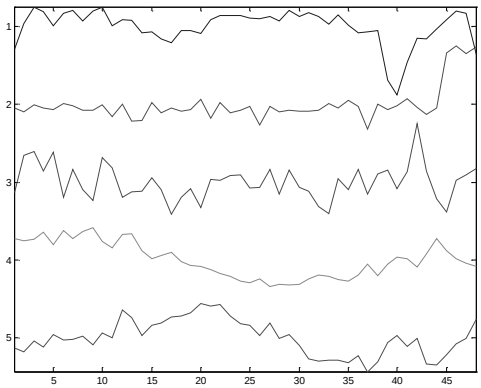


Figure 24. Buffer-size = 48; Endsample = 1097.

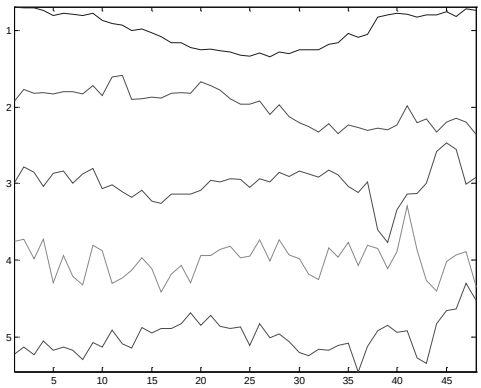
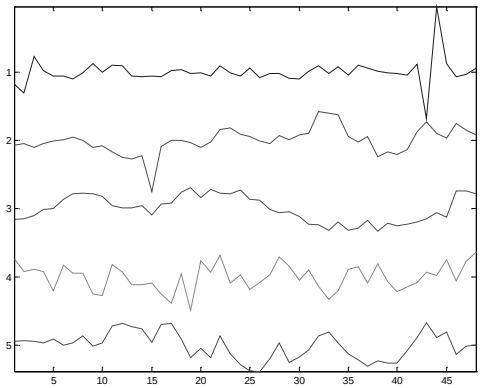
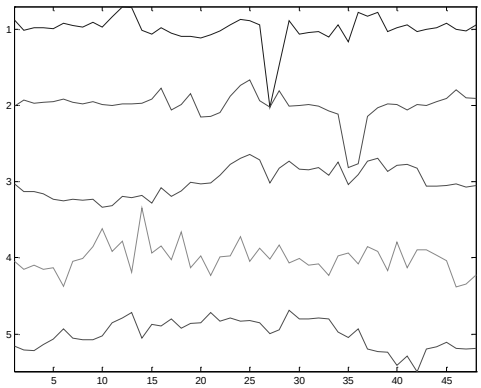


Figure 25. Buffer-size = 48; Endsample =1098.

Now we can immediately see that there is one component that that has a high peak at the end, especially near the break at 242. This might be exactly the feature we are looking for. Let us go back to the 'normal' data to also check that with a buffer-size of 48.





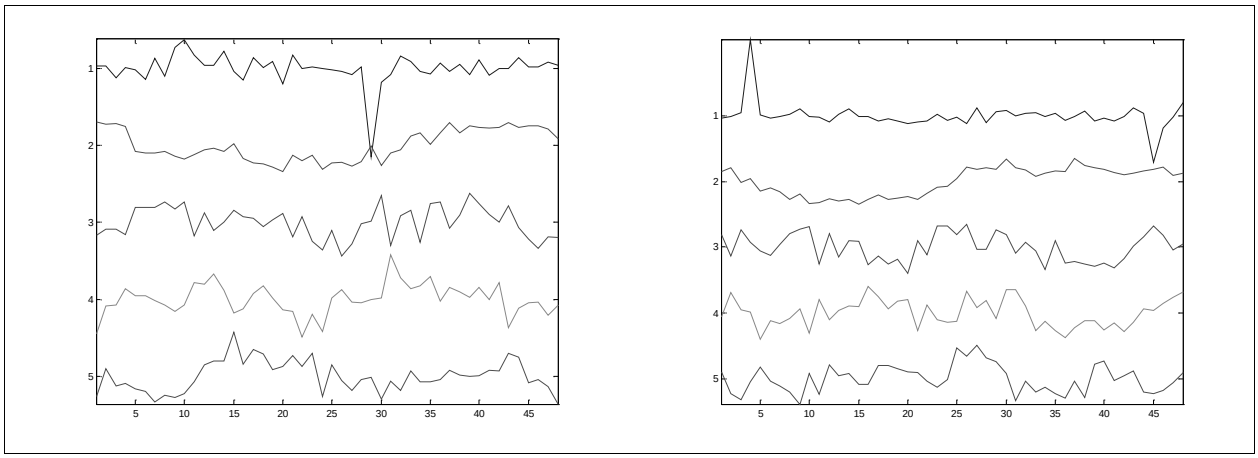


Figure 26. ICs at four randomly chosen buffers of size 48.

From this we can see that also with a buffer-size of 48, the ‘normal’ data is changing all the time and no particularly interesting features occur in all the figures.
Now lets see what happens when we apply all this to the ‘many_breaks’ dataset.

Many_breaks

When we take a look at the frequency at which breaks occur in this dataset, we see immediately that the breaks are so close to each other that the buffer is in many cases not yet completely filled before a new break occurs again. To make this concrete we look at the break-profile of this dataset and that of the dataset before. We put this all together in Table 5.

Table 5. Break profile of ’two_breaks’ and ’’many_breaks’.

Dataset	Normal/Break	Durations of the cases normal/break
‘two_breaks’	Normal	241 801 238
	Break	56 23
‘many_breaks’	Normal	44 156 62 165 160 20 45 54 46 22 133 144
	Break	37 26 26 20 19 23 44 21 30 50 28

The sizes of the breaks are in both datasets more or less the same. The main difference however is the frequency of which the breaks follow each other. One can imagine that after a break occurred, the process needs some time to recover from it, so the normal-cases in between the breaks are even smaller. Even with a buffer-size of 24 we will miss some of the breaks. With a buffer-size of 48, as we are tempted to use from now on, there will be at most 7 out of the 11 breaks that we can detect, but probably even much less.

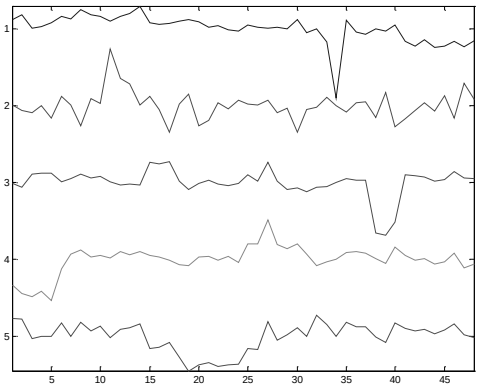


Figure 27. Buffer-size = 48; Endsample = 237.

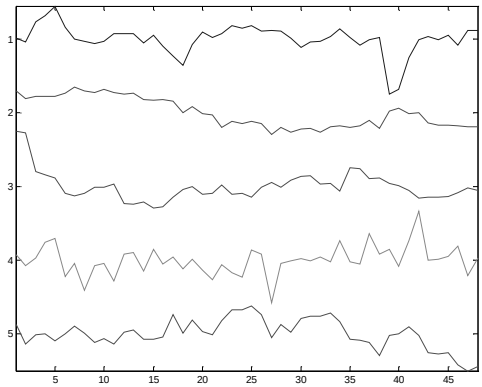


Figure 28. Buffer-size = 48; Endsample = 516.

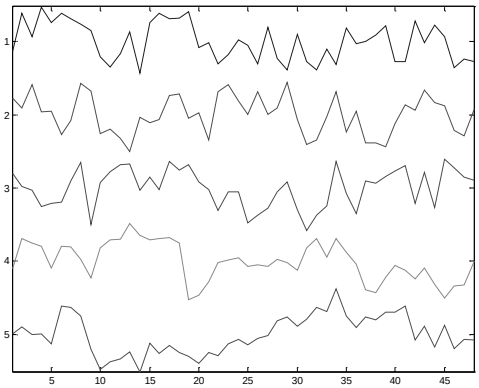


Figure 29. Buffer-size = 48; Endsample = 696.

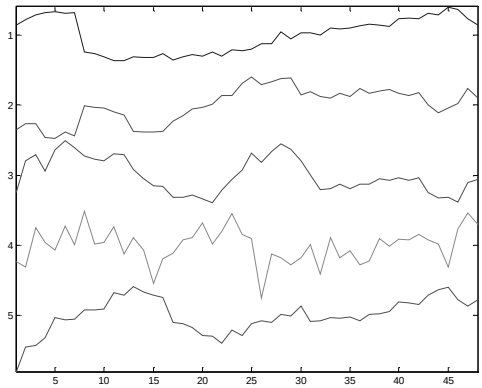


Figure 30. Buffer-size = 48; Endsample = 901.

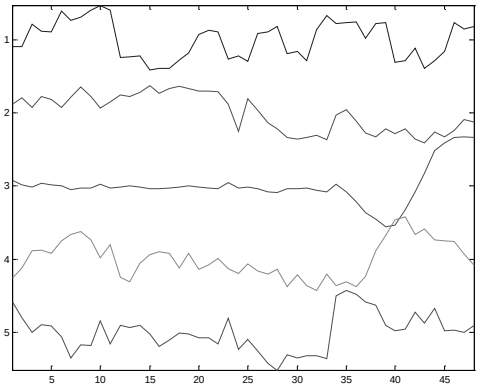


Figure 31. Buffer-size = 48; Endsample = 1203.

Although the above plots of the ICs are after a quite long period of the normal-case, there is nothing really common in all the figures. They also don't resemble the two_breaks-case, except for the graph of Figure 31. Only here there is a peak (in the 3rd component) at the end. The results here can be called very bad.

The possibility of an upcoming break

We can extract some kind of a probability measure from the height of a peak at the end of one of the components. The idea is to compare the last 5 samples of each component to the mean of the rest of the component. From this the probability measure is determined. We will look for this at the two_breaks dataset. We slide the buffer along the data, sample after sample, compute the ICs and applying the just mentioned peak measurement. We get a vector consisting of all those measurements. Sometimes FastICA cannot find the ICs. In this case, a zero is placed at the corresponding position in this vector. This all is done by the function `pulpanalyse`. We plot the result:

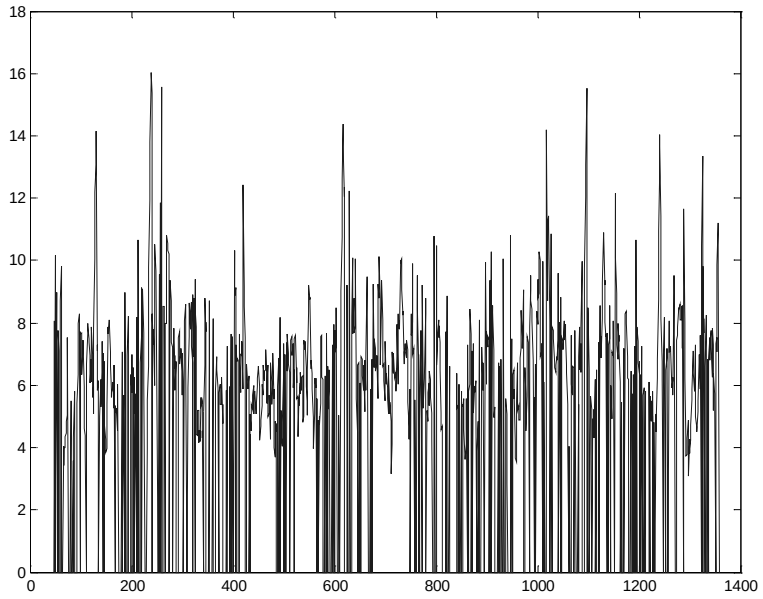


Figure 32. Time sample vs. measure of the height of the end-peak of one of the ICs.

First of all we see that some (25%) of the ICA analyses have failed. This might be a problem.

When we put a threshold of 15 on this result and force the system to detect a break at the first instance this threshold is passed, an upcoming break is detected with corresponding values (which are a measure of the certainty):

239 15.6716
260 15.5543
1098 15.5126

It will thus detect the upcoming breaks well in advance. The detection at 260 occurs in the middle of the break. In reality it wouldn't have been detected, because the process normally stops when a break has occurred.

If we apply this method to the ‘normal’ dataset, we see that no upcoming break is detected, which is of course very good.

As we already expected, applying this method to the ‘many_breaks’ dataset gives many problems. There are many false alarms and almost all of the breaks are not detected. Only one break will be detected and that is the one from Figure 31.

3.3.6 Using selected features only for ICA

We now try if we can get better results if we select only the expert-determined features from the ‘two_breaks’ dataset. We already determined that we can reduce the dimensionality of the data to only two and still describe most of the variance of the data.

Like we did above, we begin by performing ICA on 4 randomly chosen places in the normal data. We use a buffer size of 48.

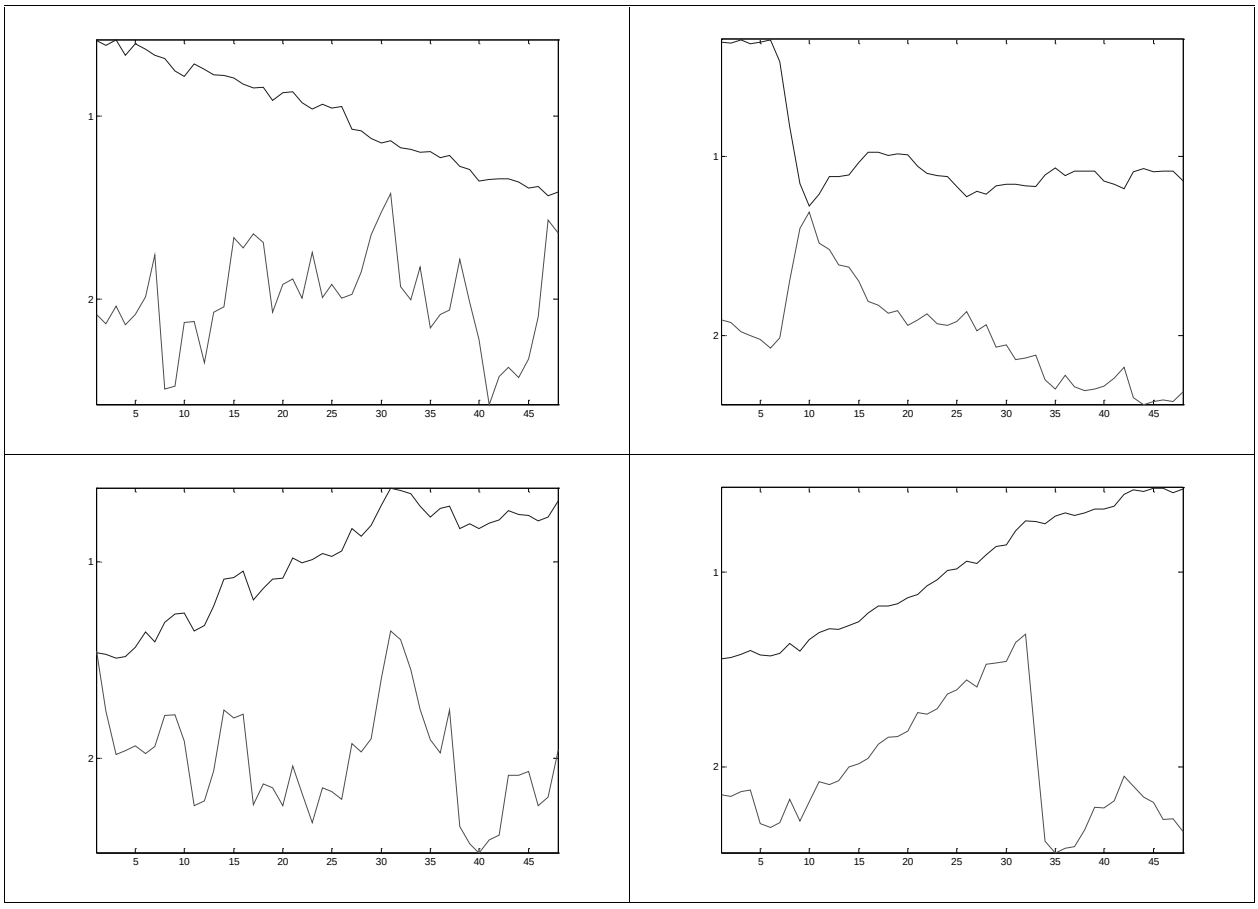


Figure 33. ICs of the selected data at four randomly chosen buffers of size 48.

We don’t know yet what to conclude about this so let’s first look at the ICs of the selected features of the ‘two_breaks’ data.

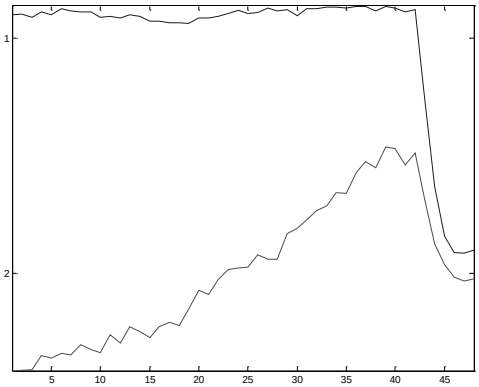


Figure 34. Selected features; Buffer-size = 48; Endsample = 240.

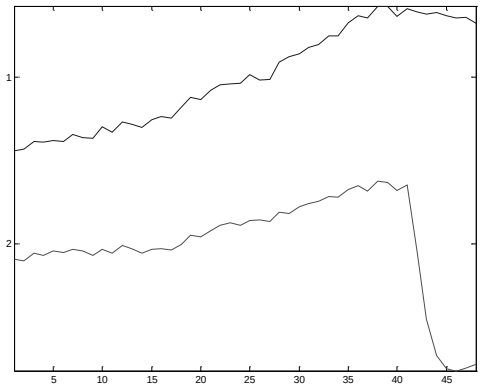


Figure 35. Selected features; Buffer-size = 48; Endsample = 241.

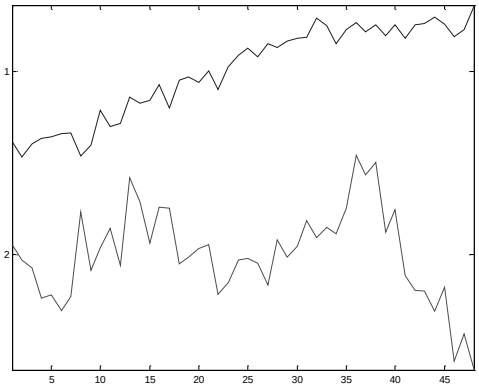


Figure 36. Selected features; Buffer-size = 48; Endsample = 1097.

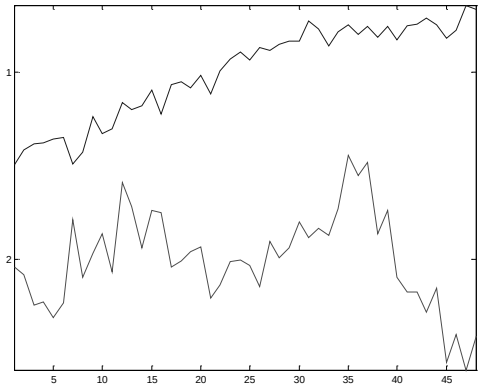


Figure 37. Selected features; Buffer-size = 48; Endsample = 1098.

We see again a very clear peak at the end of one component near the break at 242. Near the break at 1099 this peak is less clear. Let's do again the same end-peak detection as we did above and look at the results.

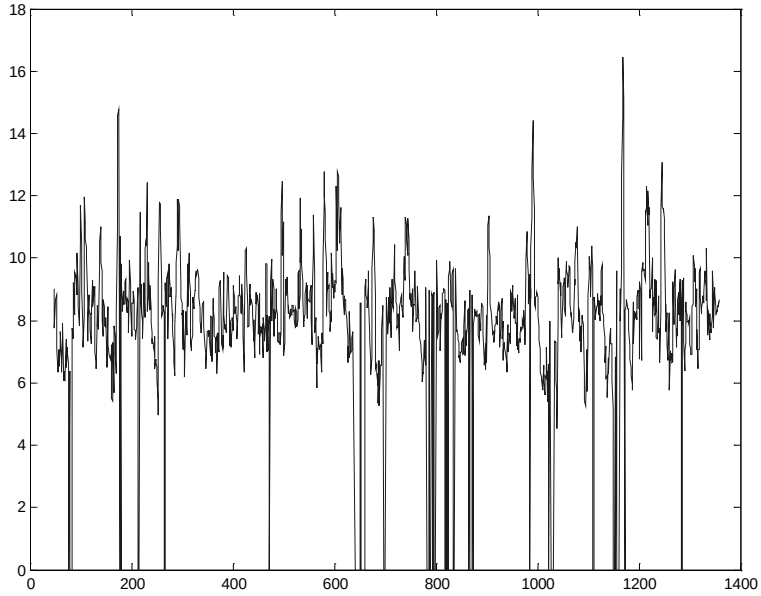


Figure 38. Time sample vs. measure of the height of the end-peak of one of the ICs using only the selected features.

It turns out that the peaks of this graph do not occur at the locations where a break occurs. Perhaps another method is needed to create a probability measure for an upcoming break. This is for future research. It can also be doubted if the selected features are the correct ones. Comparing Figure 33 with Figure 34 – 37 doesn't give us a very clear distinctive feature. Maybe there is one feature missing that contains the most information about an upcoming break. Clearly the results were better when all the data was used. This might indicate that there is indeed one very informative feature missing.

3.3.7 Conclusions

FastICA is a very powerful way of performing ICA on a dataset. It uses batches of data to analyse. In the analyses of break in a paperpulp process, we want to have this analysis online in real-time. Therefore we used a buffer to store the data. It might be a much better idea to have some kind of an online version of FastICA. These algorithms exist, but were not used in this application. Future research might consider applying such algorithms of ICA on this data. The limited buffer size turned out to be the major problem in this application. When all the features were used, nice results were achieved for breaks that occurred only sparsely. This indicates that ICA is a promising method for analysing the occurrence of a break in this process; however it needs more research before it can be used for a real process. As we suggested, an online adaptive ICA algorithm should be tested. Also reconsidering the selection of some features is a good idea.

3.4 More practical considerations

We have now seen from applying ICA some of the things we have to be careful about concerning many parts of ICA. Of course there are more things one has to keep in mind in order to apply ICA successfully, or al least with the biggest chance to be successful. We will now treat some more considerations like pre-processing, overtraining, the choice of algorithm and the number of ICs to be estimated.

3.4.1 Pre-processing

It might not always be a good idea to just input the data we have to the ICA algorithm. There is the risk that ICA won't find the ICs at all, or only meaningless ones. Things like noise reduction, independence enhancement, feature extraction and feature selection can be taken into account and will therefore be treated in this paragraph.

Noise reduction

We have given the noise free ICA model as starting point for the problem. With noise, determining the independent components is much more demanding. We won't treat here how ICs can be estimated in the presence of noise. For this, readers can refer to X. Instead, we will mention techniques to reduce the noise before the data is given as input for ICA. With noise reduction, the noise free ICA model is approximately correct, which might be well enough.

Filtering

The most common way of denoising a signal is to filter it. There is much literature on filtering data. We won't describe here the filtering of the sensor data, for this is a completely different field of engineering. We assume that we already have the sensor data in a discretized form in some dataset. Filtering can in that case only be applied to the numerical data that is available.

If we know that the data is band limited, we know that we are only interested in a limited part of the frequency spectrum. We can then filter out the rest of the spectrum. This reduces the noise level while affecting the signal in a minimal way.

Low pass filtering of the data is equal to smoothing the data. We can also regard this as averaging a consecutive number of samples. In matrix form, we can formulate a low-pass filtering matrix like:

$$\mathbf{M} = \frac{\mathbf{1}}{\mathbf{3}}\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \qquad (46)$$

Or any other matrix that performs a weighted average of the matrix of samples to which it is multiplied. For causal filters, we have to make sure that we don't take samples in the

future for the averaging. For now, we only need the matrix **M** for illustrating the principle of linear numerical filtering, so we don't bother about the exact properties of this filtering matrix.
We can apply this matrix by right multiplying it with the observation matrix. We now speak explicitly about an observation *matrix*, instead of an observation *vector*, because we now want to have the time samples explicitly available. Each row of the observation vector as we have seen it many times before, now consists of different samples $x_i(1), x_i(2), \cdots, x_i(T)$. The ICA model thus looks like:

$$\mathbf{X} = \mathbf{AS}. \tag{47}$$

Filtering **X** thus can be done as:

$$\mathbf{X}^* = \mathbf{XM} = \mathbf{ASM} = \mathbf{AS}^*, \tag{48}$$

where $\mathbf{X}^*$ and $\mathbf{S}^*$ denote the filtered versions of **X** and **S** respectively. It can thus be seen that the mixing matrix remains unaffected by this linear filtering, while the estimated ICs will be the filtered version of the otherwise obtained ICs with the same filtering matrix **M**.

The effect is that higher frequencies are averaged out of the observed data. Only the slower varying part of the data remains. The noise level can be reduced very much in this way, but we have to be careful that we don't filter out important information in the data; information that possibly determines the success of the results of ICA. Especially low-pass filtering makes the data more dependent, which is a bad thing for ICA.

Independence enhancement
High-pass filtering, on the other hand, enhances the independence of the data. This increases the chances that ICA finds the independent components, but on the other hand introduces more noise. A typical way of performing high-pass filtering to numerical data is constructing a matrix that calculates the differences between a sample and the previous one. Compared to the low-pass filtering matrix **M** in (46), we can construct a high-pass filtering matrix **M** like:

$$\mathbf{M} = \begin{bmatrix} -\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & -\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{1} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{49}$$

In practice, we have to make a compromise between low-pass and high-pass filtering, called band-pass filtering. We need to be careful, keeping both noise-reduction and independence in mind.

Noise reduction and feature extraction by PCA
Another way of noise reduction can be obtained by PCA. As we know, PCA computes the principle components of the data. The principle components can be ordered by the information about the retained variance of the data in each component. Very small

principle components thus represent either small ICs, or noise. Very small ICs are mostly not interested for the application. Discarding the small principle components is therefore a good way of reducing the noise. The dimensionality of the data is then reduced. If we perform ICA on this data with a smaller dimensionality, we can estimate the ICs with less noise. How much dimensions we need to have is often determined by trial-and-error. A good way to start is to create a graph in which the number of dimensions is displayed versus the retained variance when we reconstruct the data using only these dimensions.
Regarding the components in the observation vector as features of the data, PCA performs a way of extracting the most useful features for future ICA processing. Other ways of feature extraction may also be considered as a good pre-processing step.

Feature selection
Feature selection can, as we already mentioned in the last paragraph, be a very useful pre-processing step. Suppose we have some observation data from which we can estimate the ICs very good. Instead of this observation data, we could just as well have the same data with some additional features that doesn't contribute to the data model at all. Using this data for ICA may ruin the estimation of the ICs completely, since ICA also tries to include these features in the estimation. Therefore it is very important to find these useless features and exclude them from the data. One way of doing this is consulting expert knowledge as we suggested in the last paragraph.

3.4.2 Overtraining

Overtraining can be catastrophic to the estimation of the independent components. Overtraining occurs when there are too many parameters that have to be estimated compared to the available data. ICA will then estimate a parameter based on only a few samples. The found ICs aren't then a reflection of the process, but of the particular data instead. It can be seen if overtraining has occurred by looking at the found ICs. If a particular IC is more or less flat everywhere, except for a single spike, this is an indication for overtraining. An example of overtraining can be seen in the ICs from e.g. Figure 26. The spike maximized the nongaussianity, in this way dominating completely the variance of the IC. Obviously there are two ways we can protect ourselves from overtraining. That is either increase the number of data points, or decrease the number of parameters that need to be estimated. Since we just have a certain observation vector, we cannot get more data points then there are in the observations. Reducing the number of free parameters is thus the only option. As we already saw, enables PCA us to reduce the number of dimensions. With less dimensions, there are also less ICs to be estimated. This means that there are less free parameters in the problem. PCA is therefore not only useful for noise reduction and feature extraction, but also for avoiding overtraining.

3.4.3 Choice of the algorithm

In this report we only used the FastICA algorithm in its Matlab implementation. With this algorithm we can only analyse batches of data. We tried to make some kind of real-time ICA in the previous paragraph by using a buffer for batching the data. A really real-time ICA implementation is also available in adaptive algorithms. For more information about these kinds of algorithms for ICA, readers can refer to /6/.

Other variations in ICA algorithms exist in whether they estimate the ICs one at a time, or all together in parallel.

3.4.4 Controlling the number of independent components

Using FastICA, the user can define the number of ICs that need to be estimated. This can be useful if we need a smaller number of ICs, but don't want to reduce the dimensionality of the data more by PCA. Estimating only a limited number of ICs is also computational more efficient.

3.5 Conclusions

In this chapter we have seen two examples of applying ICA in real-life problems. We have tried to highlight the most important lessons we can learn from it. Together with the other practical considerations we made in the last part of this chapter, we have a good basis for future application of ICA. In the next chapter, we will both summarize the most important parts of the theory and the lessons learnt in this chapter. There we will make a guideline of the application of ICA for future applications.

4 A GUIDELINE FOR ICA

Based on what was derived in both the theory and practical part of this report, we will suggest a guideline for applying ICA. Engineers that consider using ICA for their particular data could follow this guideline.

1. Understand the data very well; ask questions (amongst many possible others) like:
   - What does the data describe?
   - How is the data obtained?
   - Is the dataset pre-processed in any way?
   - Are there missing values?
2. Visualize the data. This might give more insight in the data, but is also useful for comparison when the ICs are estimated; perhaps the ICs contain the same features as the original data. You then know that ICA was not applied properly, or not successful for this particular application.
3. Have a strong indication that the process and the observations follow the ICA model, i.e. that they are a linear mixture of some underlying independent sources. When the underlying model doesn't hold, there is a good chance that ICA will find ICs that make no physical sense and could therefore be useless for this particular application.
4. Pre-process the data in the following ways:
   - Make sure that any artificially introduced values, like zero-padding, are removed.
   - Remove the measurements of which it is expected that they don't contribute to the ICA model; i.e. they don't originate from the same ICs as the rest of the observations.
   - Linearly filter the data with a band-pass filter. The lower bound is for enhancing the independence of the observations and the upper bound is for filtering the noise maximally. Be careful not to affect the signal in a way that information about important features is lost.
   - Select with PCA only the eigenvalues and eigenvectors that reconstruct the data with the most variance. This will filter some noise too.
5. Decide on an algorithm for performing ICA. It should be taken into account if batch processing of the observations is possible, or that real-time processing is required.
6. Be sure that the observations are in the right format for the particular ICA algorithm. Especially the orientation of the observation matrix should be clarified.
7. Apply the ICA algorithm.
8. Display the results. The most information can be extracted from a plot of the signals. For good comparison, one has to make sure that all the plots of the ICs have the same scale.
9. Keep in mind the following things when looking at the plots:
   - There is no information about the relative strength of the ICs, since their energy cannot be estimated by ICA.

- The signs of the ICs are also unknown, so they can be mirrored with respect to the horizontal axes.
- The order of the ICs is arbitrary.

10. It can be recognized from the plots of the ICs when overtraining did occur; the signal is more or less flat over most of the range, but contains only one ore maybe a few very high peaks. The results are not trustful when overtraining took place.

5 CONCLUSIONS

In this report, the technique of independent component analysis (ICA) was investigated. Mainly based on the theory by Hyvärinen et al. /6/, we posed statements about the strength and limitations of ICA. By applying ICA to two real-world data analysis problems, we were able to come up with a guideline for its usage, which was the ultimate goal of this research.

The most important conclusion that can be drawn from our analysis is that one should be very careful in interpreting the independent components (ICs) that have been found. ICA is very powerful in finding ICs; even when there are no real underlying ICs in the observations, ICA will often find some. It is trivial that these ICs cannot be seen as 'real' ICs, but it can be hard to tell this just from the results. One, therefore, has to be very careful when and how to apply ICA to a certain dataset. Only when there is clear suspicion that the observations in fact result from some underlying components, which are suspected to be independent, one can consider ICA as data analysis tool. This suspicion can only arise when there is sufficient knowledge about the data-generating process and the basic characteristics of the dataset. Make sure to have a sufficient amount of data, in order to avoid overtraining, pre-process the data according to 3.4.1 and /6/ and be sure about the orientation of the dataset.

Once the ICs have been found, they should be examined carefully. Visualisation is a powerful method in this sense. The ICs should be compared to the original signals to make sure that they are different and have the desired or expected shape. Furthermore, spikes in one or more of the ICs can indicate that overtraining has occurred. The inability of ICA to determine the order, sign and energy of the ICs can complicate matters further. Best is to apply ICA to problems where these inabilities are no limitation or can be easily taken care of in after-processing

In spite of its limitations and need for careful treatment, ICA seems to be very powerful for many applications. With the guideline of Chapter 4 and the above-mentioned conclusions in mind, ICA is worth considering in data analysis problems.

REFERENCES

/1/ Ahola, T., Kumpula, H. and Juuso, E.: Case based prediction of paper web break sensitivity, Wissenschaftsverlag, Mainz, Germany, 2003, 161 – 167

/2/ Bell, A.J. and Sejnowski, T.J.: An Information-maximization Approach to Blind Separation and Blind Deconvolution, Neural Computation, 7(1995), 1129 – 1159

/3/ Bell, A.J. and Sejnowski, T.J.: The "Independent Components" of Natural Scenes are Edge Filters, Vision Research, 37(1997), 3327 – 3338

/4/ Cramér, H.: Mathematical Methods of Statistics, Princeton, NJ, Princeton University Press, 1946

/5/ Evans, M., Hastings, N. and Peacock, B., Statistical Distributions, New York, USA, John Wiley & Sons, 2000

/6/ Hyvärinen, A., Karhunen, J. and Oja, E.: Independent Component Analysis, Toronto, Canada, John Wiley & Sons, 2001

/7/ Hyvärinen, A. and Oja, E.: A Fast Fixed-point Algorithm for Independent Component Analysis, Neurocomputing, 9(1997), 1483 – 1492

/8/ http://www.cis.hut.fi/projects/bayes/software/ (03.11.2003)

/9/ http://www.cis.hut.fi/projects/ica/fastica/ (26.09.2003)

/10/ http://www.ph.tn.tudelft.nl/~bob/PRTOOLS.html (15.11.2003)

/11/ Jolliffe, I.T., Principle Component Analysis, New York, USA, Springer, 2002

/12/ Lee, T.: Independent Component Analysis: Theory and Practice, Boston, USA, Kluwer Academic Publishers, 1998

/13/ Nyquist, H.: Certain Topics in Telegraph Transmission Theory, AIEE Trans, 1928, 617 – 644

/14/ Ruusunen, M. and Paavola, M.: Monitoring of Automated Screw Insertion Processes – a Soft Computing Approach, Proceedings of the 7th IFAC Workshop on Intelligent Manufacturing Systems, Budapest, 2003

/15/ Ruusunen, M. and Paavola, M.: Quality Monitoring and Fault Detection in an Automated Manufacturing System – a Soft Computing Approach, Report A No 19, University of Oulu, Control Engineering Laboratory, 2002

APPENDIX A M-FILES USED FOR SCREW INSERTION ANALYSIS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Description:
%           Old is a m x n matrix with m measurements of each
%           samples. When more than 3 zero elements are in a
%           sequence, these are replaced by the mean of the other
%           samples. The result is the new m x n matrix new.
%
%     The purpose is to simplify the visualization scale.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function new = centerzero(old)

new = old;
for i=1:size(old,1)
    clear ev I j newI temp compI;
    I=find(old(i,:)==0);
    j=1;
    compI=[1:size(old,2)];
    newI=1;
    for ev=1:(length(I)-2)
        if (~old(i,I(ev)) & ~old(i,I(ev+1)) & ~old(i,I(ev+2)))
            newI(j)=I(ev);
            compI(newI(j))=1000;
            j=j+1;
        end
    end
    compI(end-1:end)=1000;
    [temp,compI]=find(compI~=1000);
    newI(end+1)=newI(end)+1; newI(end+1)=newI(end)+1;
    new(i,newI)=mean(old(i,compI));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Selects from a matrix, containing signals, the last
%           'length'-elements counting from the last non-zero element
%           of the signal (row)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function new = selectend(old,length)

new = zeros(size(old,1),length);
for i = 1:size(old,1)
    lastel = size(old,2);
    while 1
        if(old(i,lastel)==0)
            lastel = lastel-1;
        else break
        end
    end
    new(i,:) = old(i,lastel-length+1:lastel);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Sequence of commands resulting in the generation of
%           the ICs of the screw insertion data. Variations of these
%           commands have been used for obtaining the results in this
%           report.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all

% Parameters
length = 140;
lasteig = 5;
sprintf('Length = %d',length)
sprintf('LastEig = %d',lasteig)

% Loading the datasets
load checkdata3, load traindata3, load faultydata3;

% Cutting the sets
se_checkdata3=selectend(checkdata3,length);
se_traindata3=selectend(traindata3,length);
se_faultydata3=selectend(faultydata3,length);

% Measure the non-gaussianity by applying the mean-square-error to the
% kurtosis of all the signals in a dataset
sprintf('Kurtosis             of             cut-checkdata3            =
%d',sqrt(mean(kurtosis(se_checkdata3',0).^2)))
sprintf('Kurtosis             of             cut-traindata3            =
%d',sqrt(mean(kurtosis(se_traindata3',0).^2)))
sprintf('Kurtosis            of            cut-faultydata3            =
%d',sqrt(mean(kurtosis(se_faultydata3',0).^2)))

% Doing FastICA..
ica_checkdata3=fastica(se_checkdata3,'lastEig',lasteig,'verbose','on','d
isplayMode','off');
ica_traindata3=fastica(se_traindata3,'lastEig',lasteig,'verbose','on','d
isplayMode','off');
ica_faultydata3=fastica(se_faultydata3,'lastEig',lasteig,'verbose','on',
'displayMode','off');

% Displaying Results..
figure(1); icaplot('complot',ica_checkdata3,0,0,0,'ica checkdata3');
figure(2); icaplot('complot',ica_traindata3,0,0,0,'ica traindata3');
figure(3); icaplot('complot',ica_faultydata3,0,0,0,'ica faultydata3');
```

APPENDIX B M-FILES USED FOR PAPER BREAK ANALYSIS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Reads the paperpulp data and puts it in a format suitable
%           for further IC analysis.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[temp, descr] = xlsfinfo('paperpulp_data');
for i=1:length(descr)
    [A,B] = xlsread('paperpulp_data',descr{i});
    eval([descr{i} '=transpose(A(:,[3:end]));'])
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Removes the trailing NaNs and zeros and normalises the
%           result to create a more useful dataset.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = clean_set(in)

[i,j,s] = find(isnan(in));
loc = min(j);
out = in(:,[1:loc-1]);

[i,j,s] = find(out([2:end],:)==0);
if(~isempty(j))
    if(j(end)==size(out,2))
        out=out(:,[1:min(j)-1]);
    end
end

for i=2:size(out,1)
    out(i,:) = out(i,:)/(max(out(i,:)));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Generates a plot of the intrinsic dimension vs. the
%           retained variance using PCA.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Be carefull!!! BASICPCA takes the features as rowvectors. (in this
% case: data2 24x800)
w_size = 48 % 24 48 72 or 96
clear mse remvar
n_tests = (floor(size(data2,2)/w_size)*2)-1;
for j=1:n_tests
    data = data2(:,((w_size*(j-1)/2)+1:(w_size*(j-1)/2)+w_size));
    [s, V, D] = BASICPCA(data, 7);
    for p=1:10
        remvar(j,p)=sum(D(p:end))/sum(D);
        mse(j,p)=sum(D(p:end));
    end
end
mean(mse(:,[2 3]))
var(mse(:,[2 3]))
mean(remvar(:,[2 3]))
var(remvar(:,[2 3]))

figure(1); errorbar(mean(mse(:,2:end)), sqrt(var(mse(:,2:end))))
figure(2); errorbar(mean(remvar(:,2:end)),
sqrt(var(remvar(:,2:end))))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Generates a plot of the intrinsic dimension vs. the
%           retained variance using NLFA.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w_size = 48
n_tests = (floor(size(data2,1)/w_size)*2)-1;
for j=1:n_tests
    n_tests-j+1
    data = data2(((w_size*(j-1)/2)+1:(w_size*(j-1)/2)+w_size),:)';
    for p=1:10
        searchsources = p;
        nlfa_init;
        nlfa_iter;
        mse = mean((data-founddata.e)'.^2);
        m_err(j,p)=mean(mse);
        v_err(j,p)=var(mse);
        save temp w_size n_tests data2 data p hidneurons m_err v_err j
        clear all, load temp
        end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Returns estimates of upcoming failures.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out = pulpanalyse(in)

% w_size = 3*(size(in,1)-1);
w_size = 48;
lasteig = 2;

timeline = '.';
for i=1:round((size(in,2)-w_size)/20)
    timeline = strcat(timeline,'.');
end

refline = '_';
for i=1:round((size(in,2)-w_size)/20)
    refline = strcat(refline,'_');
end

fprintf(refline);
fprintf('\n');
fprintf(timeline);
for i=1:(size(in,2)-w_size)
    tempicasig                                                                  =
fastica(in([2:end],[i:w_size+i]),'lastEig',lasteig,'displayMode','off','
verbose','off');
    if(size(tempicasig,1)==lasteig)
        for j=1:lasteig
            res(j)          =          sum(abs(tempicasig(j,[end-8:end])-
mean(tempicasig(j,[1:end-7]))));
        end
        out(i) = max(res);
    else
        out(i) = 0;
    end
    if(round(i/20)*20==i)
        fprintf('\b');
    end
end
fprintf('\b');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Description:
%           Filters the result of the pulpanalysis function to better
%           estimates of upcoming faults.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [locs2,vals2] = detectfaults(rezz,w_size)

[I,J] = sort(rezz);
vals = I(end-50:end);
locs = J(end-50:end)+w_size;

[I,J] = sort(locs);

II = I(1);
JJ = J(1);
j=2;
for i=2:length(I)
    if (I(i)~=(I(i-1)+1))
        II(j)=I(i);
        JJ(j)=J(i);
        j=j+1;
    end
end
locs2 = II;
vals2 = vals(JJ);
```

ISBN 951-42-7315-X
ISSN 1238-9390
University of Oulu
Control Engineering Laboratory – Series A
Editor: Leena Yliniemi

20. **Gebus S, Lorillard S & Juuso E,** Defect Localization on a PCB with Functional Testing. 44 p. May 2002. ISBN 951-42-6731-1.
21. **Saarela U, Leiviskä K & Juuso E,** Modelling of a Fed-Batch Fermentation Process.23 p. June 2003. ISBN 951-42-7083-5.
22. **Warnier E, Yliniemi L & Joensuu P**, Web based monitoring and control of industrial processes. 15 p. September 2003. ISBN 951-42-7173-4.
23. **Van Ast J M & Ruusunen M,** A Guide to Independent Component Analysis – Theory and Practice. 53 p. March 2004. ISBN 951-42-7315-X.