*

Cergy Pontoise University— ENSEA

PROJECT REPORT

Master SIC—2006-2007

By

Abdoulaye BAGAYOKO

Supervisor:

Carmeliza NAVASCA/Lieven De Lathauwer

Title:

NON GAUSSIAN NOISE IN BLIND SOURCE SEPARATION

Project carried out at

ETIS

The project described in this report was carried out at the laboratory ETIS between January and April 2007 under the supervision of Carmeliza NAVASCA. We would like to thank D. Nion for his $ALS - l_2$ codes.

Contents

1 Introduction

One of the main objectives of signal processing is to uncover particular information hidden in observed data. In some cases, we might want to predict the future samples or interpolate the missing samples of data using the information from available data. In other cases, the desired signal might be interleaved with or added to another signal and we would like to separate them. In almost all of these cases, our task is made difficult by the fact that the corresponding signal is corrupted with unwanted random disturbances which we call noise [1].

Noise can arise from several reasons: it can be the product of some neighboring signal sources, natural or man made or it can result from our measuring processes.
Among various noise models, the additive noise model which can be represented as

$$x(n) = d(n) + w(n) \qquad n = 1, 2, 3... \tag{1}$$

is by far the more popular. In Eq.(1), x and d represent the corrupted signal and the clean signal, respectively and w is the noise which is generally assumed to be white and distributed with a Gaussian distribution. But the Gaussian distribution is less tailed and can't represent in general a realistic noise such as impulsive noise. Many signal and noise sources in the physical world for example present the impulsive behavior, sonar, satellite communications, econometrics...

One the other hand, the decomposition of tensors (arrays of order equal to 3) has proven to be useful in a number of applications. The Factor Analysis (PARAFAC) is perhaps the most striking case [2].
Lastly, the solution of some problems, including the so-called Blind Source Separation (BSS) has required the use of tensors [2].
BSS finds applications in Sonar, Radar (Chaumette Comon and Muller 1993), Electrocardiography (DeLathauwer DeMoor at alterae 2000), Speech (Nguyen-Thi and Jutten 1996; Lee and Lewicki 1999; DeLathauwer 1997), and Telecommunications (Ferreol and Chevalier 2000; Gassiat and Gamboa 1997; Van der Veen 1996; Castedo and Macchi 1997; Grellier and Comon 2000), among others. In particular, the surveillance of radiocommunications in the civil context, or interception and classification in military applications, resort to BSS. Moreover, in Mobile Communications, the mitigation of interfering users and the compensation for channel fading effects are now devised with the help of BSS; this is closely related to the general problem of *Blind Deconvolution*[2] *.

The PARAFAC model of a tensor $\underline{X}$ is given by three matrices, A, B and C, or in general, the tensor $\underline{X}$ to decompose is corrupted with noise. The known tensor $\underline{\tilde{X}}$ is then given (according to the model of the equation (1)) by

$$\underline{\tilde{X}} = \underline{X} + \underline{V} \tag{2}$$

where $\underline{\mathrm{V}}$ is a tensor of noise.
The algorithms of PARAFAC decomposition aim to estimate the matrices A, B and C given the disturbed tensor $\underline{\tilde{X}}$. It's a believe that these algorithms must depend on the type of the noise.

*See the references of these applications in [2]

Perhaps the most known algorithm of PARAFAC decomposition is the ALS(Alternating least squares) algorithm.
ALS algorithm is optimal for Gaussian noise but not for non-Gaussian [3].
The authors of [3] develops an iterative algorithm for the least absolute error fitting of general multilinear models and shows that this l_1 fitting is better for Laplace noise (which is more tailed distributed than the Gaussian noise).
This project aims to study and compare the least absolute error fitting and the ALS fitting, for one thing in the case of the Laplace noise in order to be conform to the result of [3], for another thing in the case of Cauchy noise which will be new to our knowledge.
In this document we will begin to focus on the noises giving some details on the Gaussian noise, the α-stable distributions and finally glance at the Laplace noise and the Cauchy noise, particularly . Then, it will follow the PARAFAC decomposition and the description of the ALS fitting and the least absolute error fitting. Finally, we will discuss about some simulation results.

2 Noises

In this chapter, we will talk about a class of noise: the α-stable noises and the Laplace noises. In particular, we will give details about the Cauchy and the Laplace distribution. To begin, let's point out some aspects of the Gaussian noise, that will lead to motive the study of the impulsive noises.

2.1 Gaussian noise

According to the equation (1), the p.d.f (probability density function) of the Gaussian distribution is completely determined by its mean μ and its variance σ^2. A graphic representation is given by the figure (1).

$$p(w, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(w-\mu)^2}{2\sigma^2}) \tag{1}$$

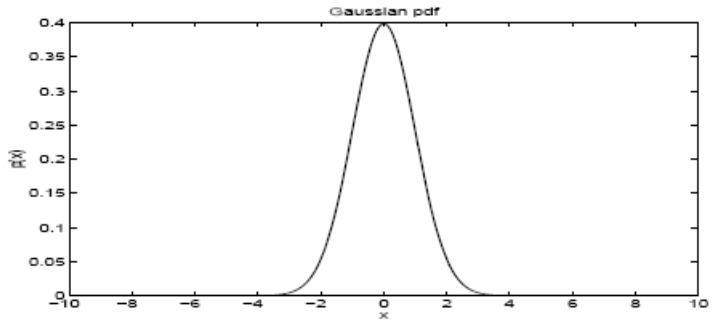

Figure 1: The Gaussian p.d.f, $\sigma^2 = 1$ and $\mu = 0$.

Many techniques in signal processing applications are developed to work ideally in Gaussian noise environments. The Gaussian assumption leads to least squares estimation which is the maximum likelihood estimate for Gaussian data. When we do not have a priori information about the noise process, it is good to use the Gaussian distribution due to the Central Limit Theorem which states that *a random variable which can be represented as the sum of infinitely many independent random variables with finite variances is distributed asymptotically with the Gaussian distribution.*
Also the popularity of the Gaussian assumption is due to tractable analytical equations that are in general linear, as opposed to the non-linear equations when non-Gaussian distributions are involved. Furthermore, linear combinations of Gaussian variables are also Gaussian and the output of a linear system to a Gaussian input, is also Gaussian, which makes the Gaussian distribution the more attractive [1].

Despite all these strong motivations of the Gaussian assumption, the Gaussian p.d.f is an exponential of the square of the deviation of the random sample from its mean, that is, the probability associated with the tails, tends to zero quickly. For example, the probability

that a sample from a Gaussian random variable deviates from the mean by $\pm 10\sigma$ is estimated at 1.5×10^{-23}. Therefore, there are many signals and noise sources, frequently encountered in the physical world, which are not Gaussian. It is a common observation in fields such as audio restoration, sonar, telephone line, satellite communications and econometrics that a large class of noise encountered in many real-world problems can be characterized as non-Gaussian and frequently as impulsive [1]. Examples of such noise are relay switching noise on telephone lines [4], noise on image caused by scratches or dropouts [5] and measurement errors.

The common property of these noise examples is that they show an impulsive behaviour. They produce large-amplitude outliers much more frequently than Gaussian noise. These high amplitude samples correspond to the very low probability parts of the Gaussian p.d.f, and therefore are the most unlikely to be samples from a Gaussian p.d.f [1].

Figure (2) shows an example of signal corrupted by such noise, a noise on the audio signal [1].

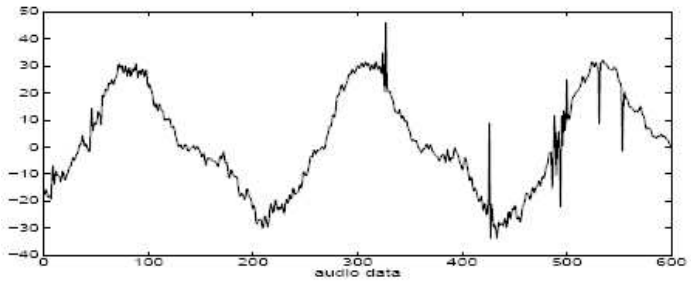

Figure 2: Corrupted audio data.

These observations show that there is a need of using another probability density functions which satisfies the impulsive properties.

2.2 The α-stable distributions

The term α-*stable* is the name given to a class of distributions which is represented in literature to be the best candidate to describe the impulsive noise [1]. In general, the α-stable distributions are defined by their characteristic function, which is the Fourier transform of their p.d.f*

$$\varphi(t) = \begin{cases} \exp\left\{j\mu t - \gamma |t|^{\alpha} \left[1 + j\beta \text{sign}(t) \tan(\frac{\alpha\pi}{2})\right]\right\} & \text{if} \quad \alpha \neq 1 \\ \exp\left\{j\mu t - \gamma |t|^{\alpha} \left[1 + j\beta \text{sign}(t) \frac{2}{\pi} \lg |t|\right]\right\} & \text{if} \quad \alpha = 1 \end{cases} \tag{2}$$

where α, β, γ and μ are the parameters which uniquely determine the distribution. In particular, the characteristic exponent α determines the impulsiveness of the distribution, that is, the smaller values of α correspond to more impulsive data.
The parameters α, β, γ and μ have the following physical meanings [1]:

(i) The characteristic exponent α is the parameters that sets the degree of the impulsiveness of the distribution. The smaller values of α correspond to heavier tailed distributions and hence to more impulsive behavior. The special case of $\alpha = 2$ corresponds to the Gaussian distribution. The case $\alpha = 1$ and $\beta = 0$ corresponds to the Cauchy distribution. Figure (3) shows the effect of α with $\beta = 0$ †.

*The α-stable p.d.f cannot be represented in a compact analytical form which makes impossible the application of popular statistical signal processing techniques (maximum likelihood estimation, Bayesian estimation) [1].

†When $\beta = 0$, the corresponding distribution are called symmetric α-stable distribution because of the symmetry of their p.d.f

(ii) The symmetry parameter β is . It determines the skewness of the distribution. When $\beta = 0$, the corresponding distribution is called symmetric α-stable distribution (SaS).

The Gaussian and the Cauchy distributions are both SaS. In the case of the SaS, power series expansions can be derived from the p.d.f. Moreover, for $\mu = 0$, that is when the corresponding SaS distributions are centred at the origin, the standard SaS density function is given by [6] :

$$f_\alpha(x) = \begin{cases} \frac{1}{\pi x}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k!}\Gamma(\alpha k+1)x^{-\alpha k}\sin(\frac{k\alpha\pi}{2}) & \text{for} \quad 0<\alpha<1 \\ \frac{1}{\pi(x^2+1)} & \text{for} \quad \alpha = 1 \\ \frac{1}{\pi x}\sum_{k=0}^{\infty}\frac{(-1)^{k}}{2k!}\Gamma(\frac{2k+1}{\alpha})x^{2k} & \text{for} \quad 1<\alpha<2 \\ \frac{1}{2\sqrt{\pi}}\exp\left[-\frac{x^2}{4}\right] & \text{for} \quad \alpha = 2 \end{cases} \tag{3}$$

Then, we can simulate the behavior of the p.d.f by giving to the variable k, high values. Figure (4) shows the effect of the parameter β.

(iii) The scale parameter is γ. It is sometimes referred to as the dispersion. The parameter γmeasures the spread of the samples from a distribution around its mean. In the particular case of Gaussian distribution, γ is the variance. Figure (5) shows its effect on the p.d.f graphically.

(iv) finally, the variable μ represents the location parameter. For SaS, μ is the mean when $1 < \alpha < 2$ and the median when $0 < \alpha < 1$. A stable distribution is said to be standard if $\mu = 0$ and $\gamma = 1$.

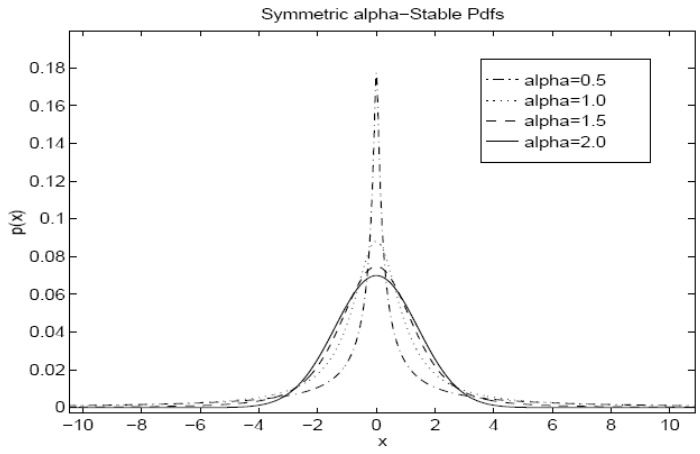

Figure 3: Effect of the Characteristic exponent α, with $\beta = 0$, $\mu = 0$ and $\gamma = 1$.

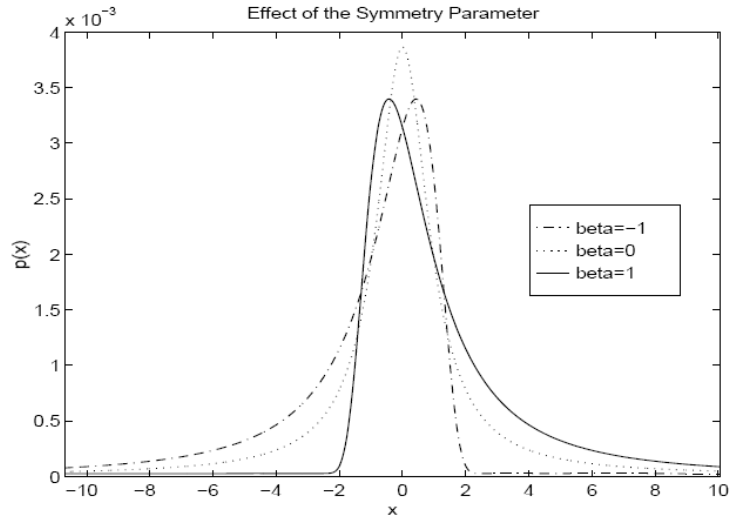

Figure 4: Effect of the symmetry parameter β, with $\alpha = 1.5$, $\mu = 0$ and $\gamma = 1$.

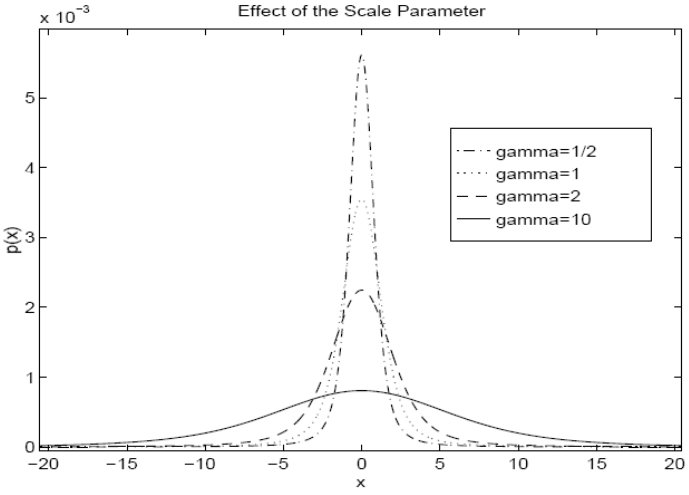

Figure 5: Effect of the dispersion γ, with $\alpha = 1$, $\beta = 0$ and $\mu = 0$.

Zolotarevs theorem [1] shows that, with the exception of Gaussian case, α-stable distributions do not have finite variance and since variance is generally associated with the power of the signal, there is a belief that, signal waveforms represented by α-stable distribution cannot be physical. But some researchers gave many reasons to use these distributions (see for example [7]).

Now, we will give some details on some particular noises: the Cauchy noise which is a symmetric-α-stable distribution with $\alpha = 1$ and $\beta = 0$, and the Laplace noise which is not in the α-stable distribution class, but presents impulsive properties.

2.3 The Cauchy distribution

The p.d.f of the Cauchy distribution is given by:

$$f_{Cauchy}(x) = \frac{1}{\pi\gamma\left[1 + (\frac{x-x_0}{\gamma})^2\right]} \qquad \text{for } x \in (-\infty; +\infty) \tag{4}$$

Then the cumulative distribution function is expressed as

$$F_{Cauchy}(x) = \int_{-\infty}^{x} f_{Cauchy}(u)du = \frac{1}{\pi}\arctan(\frac{x-x_0}{\gamma}) + \frac{1}{2} \tag{5}$$

where γ is the scale parameter and x_0 is the location parameter of the distribution. Neither the mean nor the variance is defined. The location parameter x_0 represents the median of the distribution.

Figure (6) represents the p.d.f for different values of the parameters x_0 and γ.

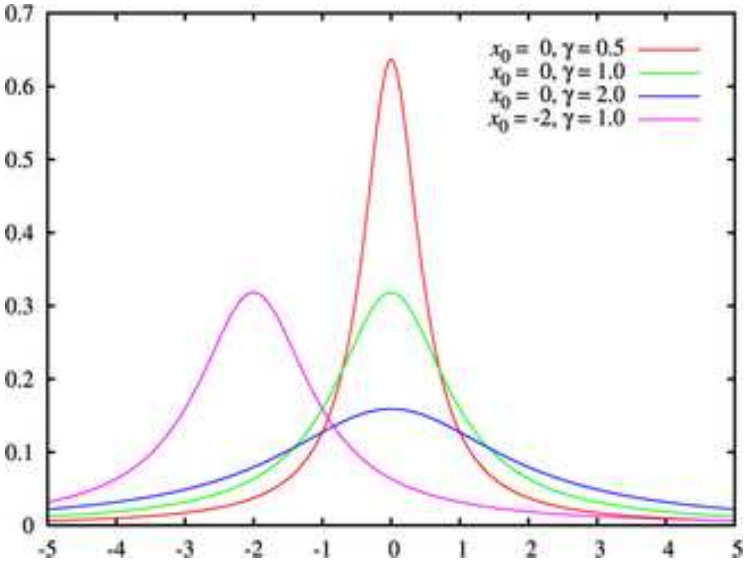

Figure 6: Cauchy p.d.f for different values of x_0 and γ. For the case $x_0 = 0$ and $\gamma = 1$ we obtain the standard Cauchy distribution.

Note that, for a given value x_0, as γ increases, the more the distribution is tailed and tends to an impulsive behavior.

2.4 The Laplace distribution

The p.d.f of the Laplace distribution is given by:

$$f_{Laplace}(x) = \frac{1}{2b}\exp(-\frac{|x-\mu|}{b}) = \frac{1}{2b}\begin{cases} \exp(-\frac{\mu-x}{b}) & \text{if} \quad x < \mu \\ \exp(-\frac{x-\mu}{b}) & \text{if} \quad x \geq \mu \end{cases} \tag{6}$$

where μ is a location parameter and $b > 0$ is a scale parameter. Note that if $\mu = 0$ and $b = 1$, the positive half-line is exactly an exponential distribution scaled by $1/2$.

Figure (7) below represents the p.d.f of the Laplace distribution for different values of the parameters μ and b.

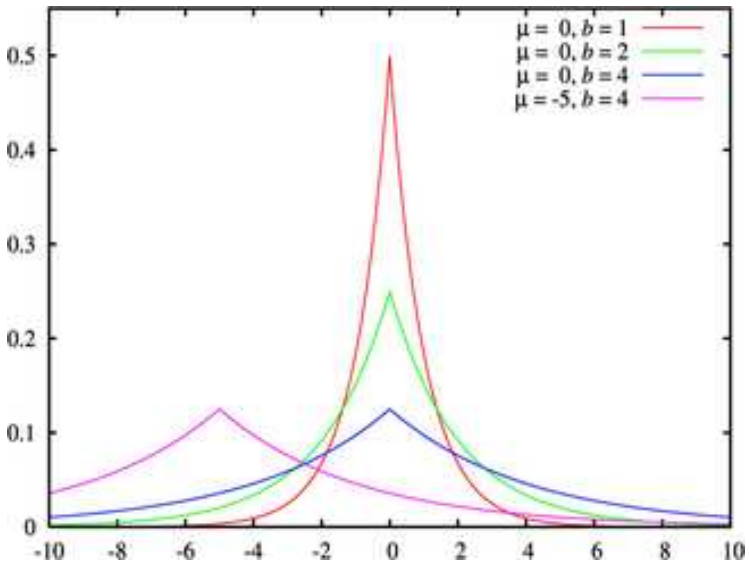

Figure 7: Laplace p.d.f for different values of μ and b.

As we can see on this figure, the Laplace distribution has "fatter" tails than the normal distribution. Then the Laplace distribution is better than the Gaussian noise for studying impulsive noises. The cumulative distribution function is as follows

$$F_{Laplace}(x) = \int_{-\infty}^{x} f_{Laplace}(u)du$$

then

$$F_{Laplace}(x) = \begin{cases} \frac{1}{2}\exp(-\frac{\mu - x}{b}) & \text{if} \quad x < \mu \\ 1 - \frac{1}{2}\exp(-\frac{x-\mu}{b}) & \text{if} \quad x \geq \mu \end{cases} = 0.5\left[1 + \text{sign}(x-\mu)(1 - \exp(-\frac{|x-\mu|}{b}))\right] \tag{7}$$

2.5 Generation of noises

In this part, we show a method to generate the sample noises given the corresponding cumulative distribution. So next, we use the result of this method in programming. Suppose that $F : \mathbb{R} \to [0,1]$ is the cumulative distribution for a random variable X. Thus,

$$F(a) = \mathbb{P}\left\{X \leq a\right\}.$$

Now, let U a uniform random variable on $[0,1]$. This means that:

$$\mathbb{P}\left\{U \in E\right\} = |E| \,^{\ddagger}$$

for all measurable subsets $E \subset [0,1]$. Define $Y = F^{-1}(U)$. Then, for any interval (a,b), we have, by a long string of trivial identities

$$\begin{aligned} \mathrm{P}(Y \in (a,b]) &= \mathbb{P}\left\{F^{-1}(U) \in (a,b]\right\} \\ &= \mathbb{P}\left\{U \in F((a,b])\right\} \\ &= \mathbb{P}\left\{U \in (F(a), F(b)]\right\} \\ &= F(b) - F(a) \,^{\S} \\ &= \mathbb{P}(X \leq b) - \mathbb{P}(X \leq a) \\ &= \mathbb{P}(a < X \leq b) \\ &= \mathbb{P}(X \in [a,b]) \ . \end{aligned}$$

[‡]For $E = [a,b]$, $|E| = b - a$

Then we conclude that X and Y have the same distribution. In other words if one have F^{-1} the inverse of the cumulative distribution, then a good way to simulate the random variable X is by the random variable $Y = F^{-1}(U)$ assuming that one have a good (pseudo-)random number generator that can simulate U well.

2.5.1 Generation of Cauchy noise

The cumulative distribution of the Cauchy noise is given by the equation (5) Let $u = F_{Cauchy}(x)$ where u is a uniform random variable on $[0, 1]$, it follows

$$x = x_0 + \gamma \tan(\pi(u - \frac{1}{2})) \tag{8}$$

where x is a sample of the Cauchy noise.
To generate Cauchy noise, we just take the expression (8) with the samples u of the random distribution on $[0, 1]$ that is generated using the function *rand* of Matlab.

2.5.2 Generation of Laplace noise

The cumulative distribution of the Laplace noise is given by the equation (7). In the same way as the Cauchy case above, let $u = F_{Laplace}(x)$, where u is a uniform random variable on $[0, 1]$. Then it follows that

$$x = \mu - b\text{sign}(u - 0.5) \ln(1 - 2\,|u - 0.5|) \tag{9}$$

To simplify the expression, let v drawn from the uniform distribution in the interval $(-1/2, 1/2]$, the variate,

$$x = \mu - b\text{sign}(v) \ln(1 - 2\,|v|), \tag{10}$$

has a Laplace distribution with parameters μ and b.

3 Tensor decompositions

The decomposition of arrays of order higher than 2 has proven to be useful in a number of applications. The most striking case is perhaps *Factor analysis* (PARAFAC) where statisticians early identified difficult problems, tackling the limits of linear algebra. The difficulty lies in the fact that such arrays may have more factors than their dimensions. Next, data are often arranged in many-way arrays, and the reduction to 2-way arrays sometimes results in a loss of information. Lastly, the solution of some problems, including the so-called *Blind Source Separation* (BSS) generally requires the use of High-Order Statistics (HOS), which are intrinsically tensor objects (McCullagh 1987)[2]. In this chapter, we describe only the PARAFAC decomposition (which is one of several decomposition methods for multi-way data),and one after the other, we give a programming way to simulate this decomposition. PARAFAC is a multi-way method originating from psychometrics[8]. The model was independently proposed by Harshman[9] and by Carroll and Chang[10] who named the model CANDECOMP (canonical decomposition).

Consider an $I \times J \times K$ three-way array $\underline{X}$ with elements x_{ijk} and $F - trilinear$ decomposition

$$x_{ijk} = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf}. \tag{1}$$

For all $i = 1, ..., I$, $j = 1, ..., J$ and $k = 1, ..., K$. Assume a_{if} stands for the $(I, f)th$ element of $I \times F$ matrix A, and similarly, b_{jf} and c_{kf} stand for $(j, f)th$ and $(k, f)th$ element of $J \times F$ and $K \times F$ matrices B and C, respectively. Equation(1) is known as trilinear decomposition or PARAFAC analysis of $x_{i,j,k}$.If $F \leq (\frac{k_A + k_B + k_C}{2}) - 1$, where k_A, k_B, k_C are the Kruskal rank of the matrices A, B and C respectively, then $rank - F$ decomposition of the three-way array $\underline{X}$ is unique[3].

3.1 Practical use of PARAFAC decomposition

A PARAFAC model of a three-way array is then given by three matrices, A, B and C with elements a_{if}, $b_{j,f}$ and $c_{k,f}$. The trilinear model is found to minimize the sum of squares of the residuals, $e_{i,j,k}$ in the model

$$x_{i,j,k} = \sum_{f=1}^{F} a_{i,f} b_{j,f} c_{k,f} + e_{i,j,k}. \tag{2}$$

This equation is shown graphically in Fig.(1) for two components ($F = 2$).

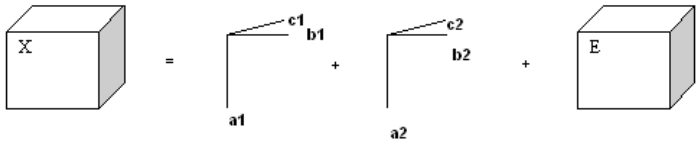

Figure 1: A graphical representation of a two-component PARAFAC model of the data array $\underline{X}$

Now, let's present the matrix representation of the tensor $\underline{X}$ in PARAFAC decomposition. Fig.(2) below represents the *slicing* of the tensor $\underline{X}$ in some I matrices of dimension $J \times K$. We can express the 2-D *slabs* X_i as:

$$X_i = BA_iC^T, \qquad i = 1, ..., I \tag{3}$$

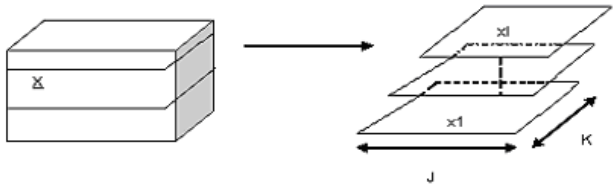

Figure 2: Horizontal *slicing* of the tensor.

Here, $A_i = D_i(A)$ denotes the operator which takes the ith row of matrix A and produces a diagonal matrix by placing this row on the main diagonal.
Due to symmetry, there are two other types of *slicing*

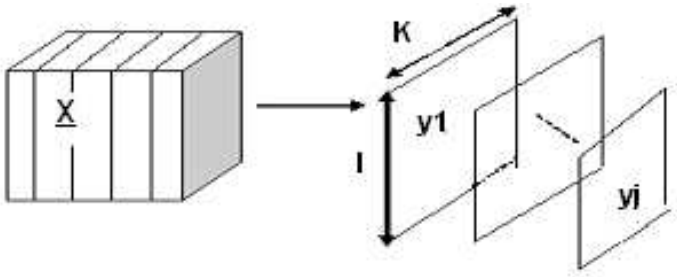

Figure 3: Vertical *slicing* of the tensor $\underline{X}$.

In the same way, we can express Y_j as:

$$Y_j = CB_jA^T, \qquad j = 1, ..., J \tag{4}$$

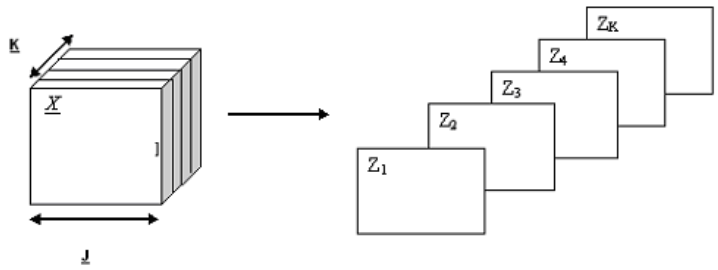

Figure 4: frontal *slicing* of the tensor $\underline{X}$.

Then,

$$Z_k = AC_kB^T, \qquad k = 1, ..., K \tag{5}$$

Where B_j and C_k are defined to be the operators which take the jth and the kth row respectivily of the matrices B and C and produce two diagonal matrices by placing this row on the main diagonal.

3.2 The PARAFAC fitting problem formulation

Now, let's stack these *slabs* as shown on the figure below Fig. (5)

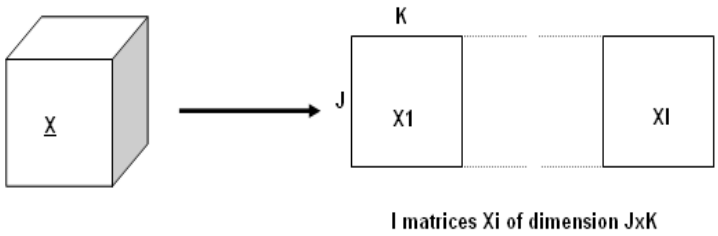

Figure 5: Stack of the *slabs* X_i.

We can, in the same way stack the "slabs" Y_j and the "slabs" Z_k.
Let X, Y and Z the matrices resulting from stacks:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_I \end{bmatrix}_{JI \times K} = \begin{bmatrix} BA_1 \\ BA_2 \\ \vdots \\ BA_I \end{bmatrix} C^T = (A \odot B)C^T \tag{6}$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_J \end{bmatrix}_{KJ \times I} = \begin{bmatrix} BC_1 \\ BC_2 \\ \vdots \\ BC_J \end{bmatrix} A^T = (B \odot C)A^T \tag{7}$$

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_K \end{bmatrix}_{IK \times J} = \begin{bmatrix} CA_1 \\ CA_2 \\ \vdots \\ CA_J \end{bmatrix} B^T = (C \odot A)B^T \tag{8}$$

where $\odot$ stands for the Khatri-Rao matrix product.
As we pointed out, in practice, the three-way array will contain measurement noise, i.e., $\underline{\tilde{X}} = \underline{X} + \underline{V}$. Then the (ijk)th element of $\underline{\tilde{X}}$ can be written as

$$\tilde{x}_{i,j,k} = x_{i,j,k} + v_{i,j,k}. \tag{9}$$

where v_{ijk} denote the additive complex i.i.d zero-mean measurement noise with statistically independent real and imaginary parts.
Considering the case of X_i, $i = 1...I$ stack, we can write

$$\tilde{X} = \begin{bmatrix} \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_I \end{bmatrix}_{JI \times K} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_I \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_I \end{bmatrix} \tag{10}$$

Then, according to the equation (6)

$$\tilde{X} = X + V = (A \odot B)C^T + V \tag{11}$$

The PARAFAC fitting problem is to find the matrices A, B and C given the noisy data $\underline{\tilde{X}}$.
Considering the equation(11), the least squares estimation problem is expressed as:

$$\min_{A,B,C} \left\| X - (A \odot B)C^T \right\|_F^2 \tag{12}$$

Let $\Delta_{AB} = (A \odot B)$, then to estimate the matrix C in least squares sense, given the matrices A and B, the problem can be formulated as:

$$\widehat{C} = \arg\min_{C} \left\| X - \Delta_{AB} C^T \right\|_F^2 \tag{13}$$

In the same way:

$$\widehat{A} = \arg\min_{A} \left\| Y - \Delta_{BC} A^T \right\|_F^2 \tag{14}$$

$$\widehat{B} = \arg\min_{B} \left\| Z - \Delta_{CA} B^T \right\|_F^2 \tag{15}$$

where $\Delta_{BC} = (B \odot C)$ and $\Delta_{CA} = (C \odot A)$. $\|.\|_F$ denotes the Frobenius matrix norm.

We can build another estimate method of matrices A, B, C based on l_1 norm minimization. That requires some necessary formulations.
We will assume all data real(imaginary part egal to zero). We introduce the operator $\mathcal{F}(\bullet)$ defined as

$$s = \mathcal{F}(\boldsymbol{S}) \triangleq \begin{bmatrix} \boldsymbol{S}_{.,1} \\ \boldsymbol{S}_{.,2} \\ \vdots \\ \boldsymbol{S}_{.,L} \end{bmatrix}$$

where $\boldsymbol{S}$ is a real-valued $M \times L$ matrix and $\boldsymbol{S}_{.,l}$ denotes its lth column.

We can are the following property (see [3] for the demonstration):

$$\mathcal{F}(DF) = (I \otimes D)\mathcal{F}(F) \tag{16}$$

where I is the identity matrix of commensurate dimension, D and F are any real-valued matrices of commensurate dimensions, $\otimes$ denotes the Kronecker matrix product.
Using this property (16), the absolute error model fitting criterion can be written as:

$$\|\tilde{x} - (I_{\mathcal{K}} \otimes \{A \odot B\})c\|_1 \tag{17}$$

with $\tilde{x} = \mathcal{F}(\tilde{X})$, $c = \mathcal{F}(C^T)$ and $I_{\mathcal{K}}$ is the $K \times K$ indentity matrix.
note that expression (17) is the norm of a vector in l_1 norm sense.
The absolute error models corresponding to two others kinds of "slicing" of the tensor $\underline{X}$ are:

$$\|\tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a\|_1 \tag{18}$$

and

$$\|\tilde{z} - (I_{\mathcal{J}} \otimes \{C \odot A\})b\|_1 \tag{19}$$

with $\tilde{y} = \mathcal{F}(\tilde{Y})$, $a = \mathcal{F}(A^T)$, $\tilde{z} = \mathcal{F}(\tilde{Z})$ $b = \mathcal{F}(B^T)$, and $I_{\mathcal{I}}$ and $I_{\mathcal{J}}$ denote respectivily the $I \times I$ and the $J \times J$ identity matrix.
Now, the necessary formulations are made to build the PARAFAC fitting algorithms.

4 Description of algorithms

We will use the equations built in the last section to make two fitting algorithms for the PARAFAC decomposition. The first is based on the least squares estimation and the second uses the l_1 norm minimization. To understand the first algorithm, let's begin with an outline of the least squares estimation.

4.1 Least squares estimation

Consider the bilinear model:

$$M\Phi = \Theta + E \tag{1}$$

where E represents the measurement noise in the model. Φ is the unknown of the model. Solving this problem in least squares sense, leads to minimize the l_2 norm

$$\|M\Phi - \Theta\|_2^2 \tag{2}$$

Let $\hat{\Phi}$ the value of Φ that minimizes this norm:

$$\hat{\Phi} = \arg\min_{\Phi} \|M\Phi - \Theta\|_2^2 = \arg\min_{\Phi} (M\Phi - \Theta)^T (M\Phi - \Theta) \tag{3}$$

Then, $\hat{\Phi}$ checks the following equation:

$$\frac{\partial(\hat{\Phi}^T M^T M\hat{\Phi} - \hat{\Phi}^T M^T\Theta - \Theta^T M\hat{\Phi} + \Theta^T\Theta)}{\partial\hat{\Phi}} = \boldsymbol{O} \tag{4}$$

where $\boldsymbol{O}$ is the zero vector with commensurate dimension. It follows that (4)

$$\hat{\Phi} = (M^T M)^{-1} M^T\Theta \tag{5}$$

4.2 Alternating least squares PARAFAC algorithm

Algorithms for fitting the PARAFAC model are usually based on alternating least squares. We aim to find the 2-D matrices A, B and C to solve the problem. We can solve the equation (13), (14) and (15) formulated the corresponding least squares estimation problem. From the previous section, the solution to these equations are:

$$\hat{A} = Y^T\Delta_{BC}(\Delta_{BC}\Delta_{BC}^T)^{-1} \tag{6}$$

$$\hat{B} = Z^T\Delta_{CA}(\Delta_{CA}\Delta_{CA}^T)^{-1} \tag{7}$$

$$\hat{C} = X^T\Delta_{AB}(\Delta_{AB}\Delta_{AB}^T)^{-1} \tag{8}$$

The alternating least squares algorithm is based on the following statement: given the matrices B and C, we can estimate the matrice A using the equation (6), and with this new value of A and the previous value of C, we can estimate the matrice B using the

equation (7), and finally, we estimate the matrice C with the previous values of A and B using the equation (8). We can reiterate this process and measure the relative error between the previous value of the resulting tensor and the current value of the resulting tensor. The convergence is reaching when the error has decreased to a criterion.
Some works in the signal processing community show that convergence is usually guaranteed, but convergence to a global minimum is not guaranteed [11].
The pseudo-code of this algorithm can be the following:

(i) Initialize the paramaters, we must initialize two of the three matrices A, B and C. Let given B and C for example. The tensor $\underline{\tilde{X}}$ is known, it represents the tensor $\underline{X}$ added to the noise. By "slicing" $\underline{\tilde{X}}$, do the following operations

- Calculate $X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_I \end{bmatrix}_{JI \times K}$
- Calculate $Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_J \end{bmatrix}_{KJ \times I}$
- Calculate $Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_K \end{bmatrix}_{IK \times J}$

(ii) Calculate $\hat{A} = Y^T \Delta_{BC} (\Delta_{BC} \Delta_{BC}^T)^{-1}$

(iii) Calculate $\hat{B} = Z^T \Delta_{CA} (\Delta_{CA} \Delta_{CA}^T)^{-1}$

(iv) Calculate $\hat{C} = X^T \Delta_{AB} (\Delta_{AB} \Delta_{AB}^T)^{-1}$

(v) Build X (let's call $\hat{X}_{current}$ this estimation) with the estimations $\hat{A}$, $\hat{B}$ and $\hat{C}$. We can build also Y and Z, but only build a matrix (X or Y or Z) because everyone contains the same information on the tensor $\underline{X}$.

(vi) Calculate the error between $\hat{X}_{current}$ and $\hat{X}_{previous}$. Where $\hat{X}_{previous}$ represents the "$\hat{X}_{current}$" of the previous iteration.

(vii) Do $\hat{X}_{previous} = \hat{X}_{current}$

(viii) Return to step (ii) until convergence (that is, obtain the desirable error)

4.3 PARAFAC algorithm based on the l_1 norm minimization

The estimation problems corresponding to the equations (17), (18) and (19) can be expressed as:

$$\hat{a} = \arg\min_a \|\tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a\|_1 \tag{9}$$

$$\hat{b} = \arg\min_b \|\tilde{z} - (I_{\mathcal{J}} \otimes \{C \odot A\})b\|_1 \tag{10}$$

$$\hat{c} = \arg\min_c \|\tilde{x} - (I_{\mathcal{K}} \otimes \{A \odot B\})c\|_1 \tag{11}$$

where $I_{\mathcal{I}}$, $I_{\mathcal{J}}$ and $I_{\mathcal{K}}$ denote respectively the $I \times I$, the $J \times J$ and the $J \times J$ identity matrices.
These l_1 minimization problems can be solved using the linear programming. To do that, we must reform the expressions (9), (10)and (11).
Let's consider the equation (9). Assume y_i(for $i = 1...IJK$) are the elements of the vector $\tilde{y}$ (In general, for a $M \times N$ matrix $\mathfrak{M}$, $\widetilde{\mathfrak{M}} = \mathcal{F}(\mathfrak{M})$ is a vector of length MN, so the length of $\tilde{y}$ is IJK, because $\tilde{y} = \mathcal{F}(Y)$ and Y is a $KJ \times I$ matrix), and assume Ψ_i (for $i = 1...IJK$) are the elements of the vector $(I_{\mathcal{I}} \otimes \{B \odot C\})a$. Then

$$\|\tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a\|_1 = \sum_{i=1}^{IJK} |y_i - \Psi_i| \tag{12}$$

To Minimize this sum leads to minimize the slack variables q_i such that:

$$|y_i - \Psi_i| + q_i = 0$$

for $i = 1...IJK$. Let q_i stands for the vector $\mathbf{q}$. So it follows

$$\min_a \|\tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a\|_1 \Leftrightarrow \min_{a,\mathbf{q}} e^T\mathbf{q}$$

subject to

$$-\mathbf{q} \preceq \tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a \preceq \mathbf{q}$$

where $e = [1, 1, ..., 1]^T$ is a vector of length IJK and $\preceq$ denotes the usual *pointwise* ordering.
Now, introduce the vector $\mathbf{a}$ defined as:

$$\mathbf{a} = \begin{bmatrix} a \\ \mathbf{q} \end{bmatrix}$$

and introduce also the vector $\mathbf{e}_{FI-IJK}$ defined as $\mathbf{e}_{FI-IJK} = [0, \dots 0, 1, \dots 1]^T$ with FI elements equal to zero (because a is a vector of length FI) and IJK elements equal to "1" (the same length as the vector $\mathbf{q}$). Then

$$\min_{a,\mathbf{q}} e^T\mathbf{q} \Leftrightarrow \min_{\mathbf{a}} \mathbf{e}^T_{FI-IJK}\mathbf{a} \tag{13}$$

Define $\mathbf{\Delta}_{BC}$ as:

$$\mathbf{\Delta}_{BC} = I_{\mathcal{I}} \otimes \{B \odot C\} = I_{\mathcal{I}} \otimes \Delta_{BC}$$

Then we are:

$$-\mathbf{q} \preceq \tilde{y} - (I_{\mathcal{I}} \otimes \{B \odot C\})a \preceq \mathbf{q} \Leftrightarrow -\mathbf{q} \preceq \tilde{y} - \mathbf{\Delta}_{BC}a \preceq \mathbf{q}$$

It follows

$$\begin{bmatrix} -\mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \end{bmatrix} \mathbf{a} \preceq \begin{bmatrix} -\tilde{y} \\ \tilde{y} \end{bmatrix} \tag{14}$$

Therefore, we obtain the standard form of the linear programming:

$$\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} \mathbf{e}_{FI-IJK}^T \mathbf{a} \qquad \text{subjet to} \qquad \begin{bmatrix} -\mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \end{bmatrix} \mathbf{a} \preceq \begin{bmatrix} -\tilde{y} \\ \tilde{y} \end{bmatrix} \tag{15}$$

This linear programming (15) can be efficiently solved using the funtion *linprog* of Matlab. Considering the symmetry of the problem, equations (10) and (11) become respectively:

$$\hat{\mathbf{b}} = \arg\min_{\mathbf{b}} \mathbf{e}_{FJ-IJK}^T \mathbf{b} \qquad \text{subjet to} \qquad \begin{bmatrix} -\mathbf{\Delta}_{CA} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{CA} & -I_{\mathcal{IJK}} \end{bmatrix} \mathbf{b} \preceq \begin{bmatrix} -\tilde{z} \\ \tilde{z} \end{bmatrix} \tag{16}$$

and

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \mathbf{e}_{FK-IJK}^T \mathbf{c} \qquad \text{subjet to} \qquad \begin{bmatrix} -\mathbf{\Delta}_{AB} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{AB} & -I_{\mathcal{IJK}} \end{bmatrix} \mathbf{b} \preceq \begin{bmatrix} -\tilde{x} \\ \tilde{x} \end{bmatrix} \tag{17}$$

The pseudo-code of the resulting algorithm is:

(i) Initialize the paramaters, we must initialize two of the three matrices A, B and C. Let given B and C for example. The tensor $\underline{\tilde{X}}$ is known, it represents the tensor $\underline{X}$ added to the noise. By "slicing" $\underline{\tilde{X}}$, do the following operations

- Calculate $X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_I \end{bmatrix}_{JI \times K}$ and calculate $\tilde{x} = \mathcal{F}(X)$; for that, we can make a function $F\_mapping$, such that, if we apply $F\_mapping$ to F, we obtain $\tilde{x}$ ($\tilde{x} = F\_mapping(X)$). We can also make a function $inverse\_F\_mapping$ to calculate the inverse of the operator $\mathcal{F}(\bullet)$.
- Calculate $Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_J \end{bmatrix}_{KJ \times I}$ and call the function $F\_mapping$ to calculate $\tilde{y} = F\_mapping(y)$.
- Calculate $Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_K \end{bmatrix}_{IK \times J}$ and call the function $F\_mapping$ to calculate $\tilde{z} = F\_mapping(z)$.
- Build $\mathbf{e}_{FI-IJK}$, $\mathbf{e}_{FJ-IJK}$ and $\mathbf{e}_{FK-IJK}$.
- Build the vectors $\mathfrak{m}_{A2} = \begin{bmatrix} -\tilde{y} \\ \tilde{y} \end{bmatrix}$, $\mathfrak{m}_{B2} = \begin{bmatrix} -\tilde{z} \\ \tilde{z} \end{bmatrix}$ and $\mathfrak{m}_{C2} = \begin{bmatrix} -\tilde{x} \\ \tilde{x} \end{bmatrix}$.

(ii) Update A:

- Calculate $\mathbf{\Delta}_{BC} = I_{\mathcal{I}} \otimes \left\{ \hat{B} \odot \hat{C} \right\}$.
- build the matrix $\mathfrak{m}_{A1} = \begin{bmatrix} -\mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{BC} & -I_{\mathcal{IJK}} \end{bmatrix}$.

- Call *linprog*: $\hat{\mathbf{a}} = linprog(\mathbf{e}_{FI-IJK}, \mathfrak{m}_{A1}, \mathfrak{m}_{A2})$.
- extract $\tilde{a}$ from $\hat{\mathbf{a}}$: $\tilde{a} = \hat{\mathbf{a}}(1 : FI)$. Where $\hat{\mathbf{a}}(1 : FI)$ denotes the vector extracted from $\hat{\mathbf{a}}$ taking the first to the (FI)th element*.
- Deduct $\hat{A} = inverse\_F\_mapping(\tilde{a})$.

(iii) Update B:

- Calculate $\mathbf{\Delta}_{CA} = I_{\mathcal{J}} \otimes \left\{\hat{C} \odot \hat{A}\right\}$.
- build the matrix $\mathfrak{m}_{B1} = \begin{bmatrix} -\mathbf{\Delta}_{CA} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{CA} & -I_{\mathcal{IJK}} \end{bmatrix}$.
- Call *linprog*: $\hat{\mathbf{b}} = linprog(\mathbf{e}_{FJ-IJK}, \mathfrak{m}_{B1}, \mathfrak{m}_{B2})$
- extract $\tilde{b}$ from $\hat{\mathbf{b}}$: $\tilde{b} = \hat{\mathbf{b}}(1 : FJ)$. Where $\hat{\mathbf{b}}(1 : FJ)$ denotes the vector extracted from $\hat{\mathbf{b}}$ taking the first to the (FJ)th element .
- Deduct $\hat{B} = inverse\_F\_mapping(\tilde{b})$.

(iv) Update C:

- Calculate $\mathbf{\Delta}_{AB} = I_{\mathcal{K}} \otimes \left\{\hat{A} \odot \hat{B}\right\}$.
- build the matrix $\mathfrak{m}_{C1} = \begin{bmatrix} -\mathbf{\Delta}_{AB} & -I_{\mathcal{IJK}} \\ \mathbf{\Delta}_{AB} & -I_{\mathcal{IJK}} \end{bmatrix}$.
- Call *linprog*: $\hat{\mathbf{c}} = linprog(\mathbf{e}_{FK-IJK}, \mathfrak{m}_{C1}, \mathfrak{m}_{C2})$.
- extract $\tilde{c}$ from $\hat{\mathbf{c}}$: $\tilde{c} = \hat{\mathbf{c}}(1 : FK)$. Where $\hat{\mathbf{c}}(1 : FK)$ denotes the vector extracted from $\hat{\mathbf{c}}$ taking the first to the (FK)th element .
- Deduct $\hat{C} = inverse\_F\_mapping(\tilde{c})$.

(v) Build X (let's call $\hat{X}_{current}$ this estimation) with the estimations $\hat{A}$, $\hat{B}$ and $\hat{C}$. We can build also Y and Z, but only build a matrix (X or Y or Z) because everyone contains the same information on the tensor $\underline{X}$.

(vi) Calculate the error between $\hat{X}_{current}$ and $\hat{X}_{previous}$. Where $\hat{X}_{previous}$ represents the "$\hat{X}_{current}$" of the previous iteration.

(vii) Do $\hat{X}_{previous} = \hat{X}_{current}$. Return to step (ii) until convergence (that is, obtain the desirable error)

Due to the complexity of the Matlab function *linprog* †, this algorithm is very slow as we will see in the next section. Another alternative exists, interior-point-method. But we don't have more time to study this method.

The Matlab code of the two algorithms is jointed at the end of the report.

*This is the notation of Matlab.

† *linprog (f,A,B)* solve the linear programming $\min_x f^T x$ subject to $Ax \preceq B$. See Matlab for more information.

5 Numerical results

In this part of the report, we will present some simulation results. The two last algorithms are simulated in Matlab environment. We aim to compare PARAFAC l_1 and l_2 algorithms in case of three noises: the Gaussian noise, the Laplace noise and the Cauchy noise. For the examples below, let $I = 4$, $J = 5$, $K = 10$ and $F = 3$. Also, we set two stopping conditions of the algorithms: $param\_vect(1)$ for the time allowed to reach convergence, and $param\_vect(2)$ for convergence criterion, i.e. tolerance to decide whether convergence has been reached (e.g. 10^{-9})*

5.1 Simulation with Gaussian noise

Consider the samples of the noise following the Gaussian distribution with mean equal to zero and unit variance: $m = 0$ and $\sigma^2 = 1$. Next, add this noise to a tensor $\underline{X}$ with random elements. Assume we don't know $\underline{X}$, but we have only the noisy tensor $\underline{\tilde{X}}$. We aim to estimate $\underline{X}$ given $\underline{\tilde{X}}$. Using the alternating algorithms above, we plot the graph corresponding to the error $\left\|\hat{X}_{previous} - \hat{X}_{current}\right\|_F^2$ † at each iteration. The tolerance to decide whether the convergence has been reached is set to $param\_vect(2) = 10^{-9}$ and the time allowed is set to $param\_vect(1) = 60s$ for the ALS algorithm and $param\_vect(1) = 160s$ for the l_1 algorithm ‡. Then, we obtain the figure(1).
In this case, we notice that the ALS § algorithm is more efficient than the algorithm based on l_1 minimization because the error decreases more quickly in the case of ALS. This result is expected since l_2 fitting is the best treatment for Gaussian noise(see [12]). Furthermore, considering calculation time, the ALS is by far faster than the l_1 algorithm. For example on our machine we obtain, $time{-}ALS{-}G = 0.0843s$ and $time{-}l_1{-}G = 65s$, where $time{-}ALS{-}G$ and $time{-}l_1{-}G$ denote the time spent running the ALS algorithm and the time spent running the l_1 algorithm, respectively. Note that this time depends on the machine and processor, but the ratio may be constant whatever the machine.

*We take the same notations as in our Matlab program. $param\_vect$ is the parameter vector with three elements: $param\_vect(1)$, $param\_vect(2)$ and $param\_vect(3)$. $param\_vect(3)$ is set to 1, so we compare $\hat{X}_{previous}$ to $\hat{X}_{current}$ to evaluate the relative amount of correction brought by the current iteration (see the Matlab code).

† $\|\bullet\|_F$ is the Frobenius norm of a matrix. Here $\hat{X}$ results of the stacking of the tensor $\underline{\hat{X}}$. See the chapter 3 for more information.

‡That permits to compare the two graphs through great numbers of iterations because ALS algorithm is by far faster than the l_1 algorithm.

§Alternating Least Squares: this is the name given to the algorithm based on the least squares estimation (see chapter 4 for more information).

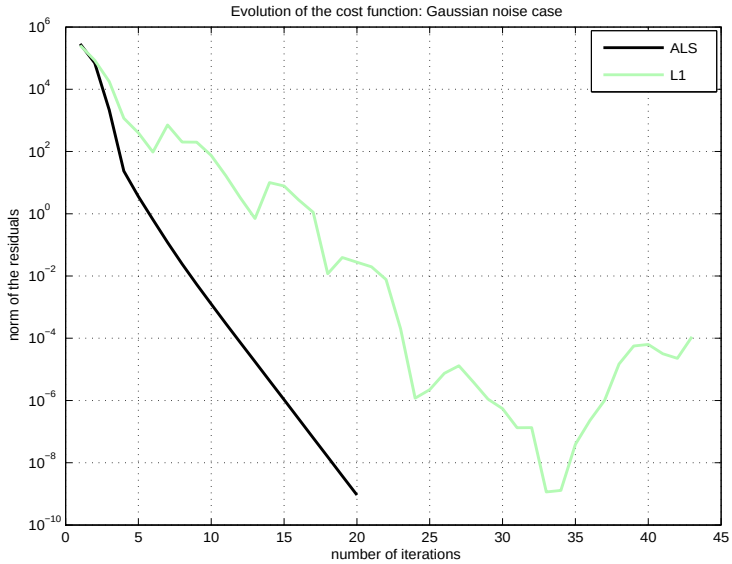

Figure 1: Evolution of the cost function. Tensor with Gaussian noise in case of ALS and L1 algorithms.$\sigma = 1, \quad m = 0$

5.2 Simulation with Cauchy noise

Now, consider the case of Cauchy noise. We choose the same simulation parameters as in Gaussian case ($param\_vect$ vector). To simulate a more tailed distributed noise, let's set $\gamma = 4$ (see chapter 2) because it's more impulsive noise than all Gaussian noise. We obtain the Fig. (2)

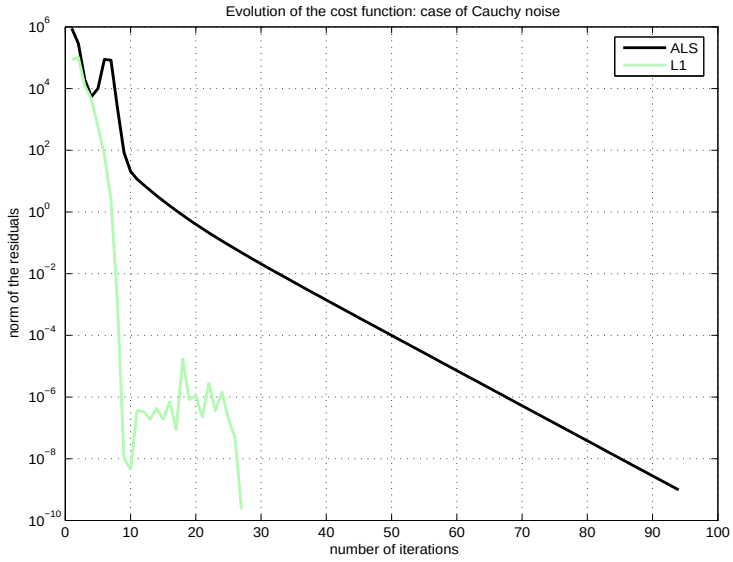

Figure 2: Evolution of the cost function. Tensor with Cauchy noise in case of ALS and L1 algorithms.$\gamma = 4, \quad x_0 = 0$

We note a net efficiency of the l_1 minimization algorithm. The convergence of the l_1 minimization algorithm is reached before 30 iterations, while the convergence of the ALS algorithm is reached after 90 iterations. On the other hand, the time of calculation is better for the ALS ($time-ALS-C = 0.0802s$)than for the l_1 algorithm ($time-l_1-C = 160.8s$), but as we have mentioned above, this is due to the use of the linear programming solver available in Matlab. There are some methods to speed up the process of solving (see [12] and [13]for using interior point method)

To compare the two algorithms in terms of impulsiveness, we simulate the case of two different tails of Cauchy distribution: $\gamma = 0.5$ (less impulsive noise) and $\gamma = 3$ (more impulsive noise). We obtain Fig. (3).

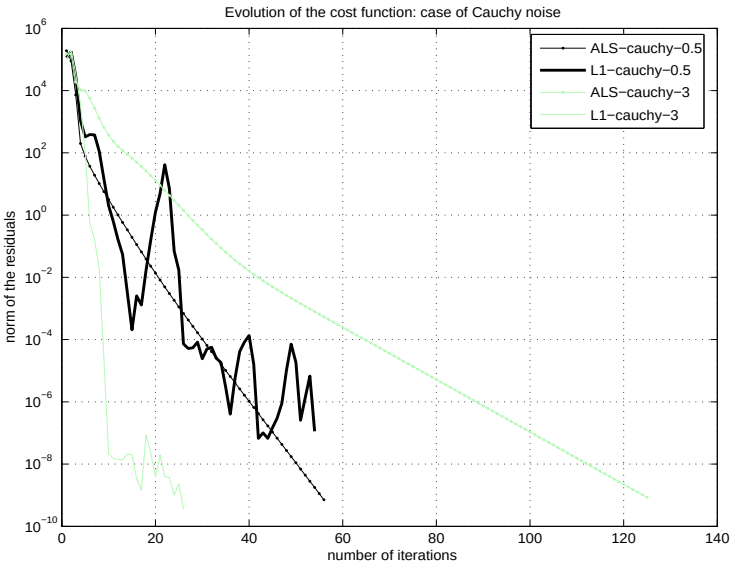

Figure 3: Evolution of the cost function. Tensor with Cauchy noise in case of ALS and L1 algorithms. Comparison of two different tails: $case1 \rightarrow \gamma = 0.5$, $x_0 = 0$ and $case2 \rightarrow \gamma = 3$, $x_0 = 0$.

This simulation shows that the more the noise is impulsive, the more the l_1 minimization algorithm is efficient than the ALS algorithm. Thus, we conclude for tensor with impulsive noise, the l_1 minimization is a better method.

5.3 Simulation with Laplace noise

For the Laplace noise, let's choose to compare two different tails (scales) of the distribution: $b = 1$ (less tailed noise) and $b = 4$ (more tailed noise). We obtain Fig.(4).

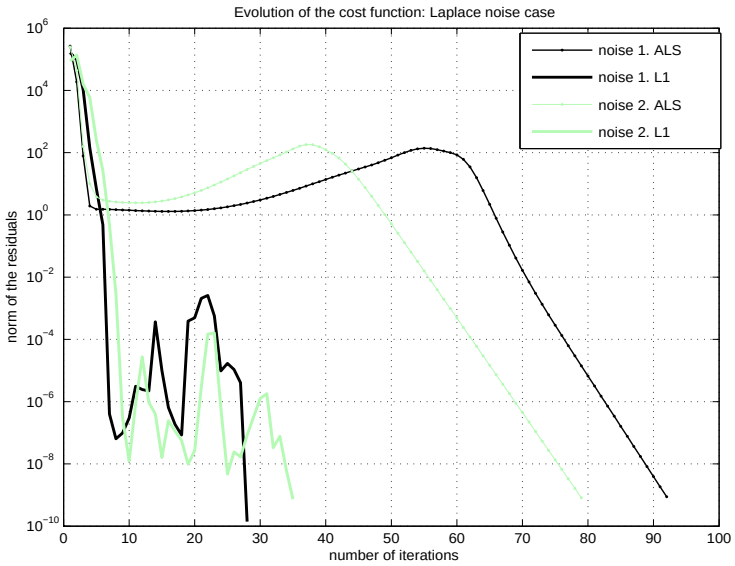

Figure 4: Evolution of the cost function. Tensor with Laplace noise in case of ALS and L1 algorithms. Comparison of two different tails: $case1 \rightarrow b = 1,\ \mu = 0$ and $case2 \rightarrow b = 4,\ \mu = 0$.

The convergence is reached for the ALS method after 80 iterations (noise 2) and 93 iterations (noise 1) while with l_1 minimization the convergence is obtained after 35 iterations (noise 2) and 28 iterations (noise 1) with tolerance set to 10^{-9}.
Note that if the tolerance is set to 10^{-7}, the convergence for l_1 takes a lot less iteration, i.e. 8 iterations (noise 1) and 10 iterations (noise 2).

6 Conclusion

In this report, two types of noises have been studied, the α-stable distributed noises particularly, Gaussian and Cauchy noises, and the Laplace noise. Then, two iterative algorithms for fitting trilinear PARAFAC models have been studied, too, the alternating least squares algorithm ALS based on the least squares estimation and the least absolute error l_1 algorithm based on the linear programming. We have compared these two algorithms in the case of the Cauchy noise, the Laplace noise and the Gaussian noise. Our results show that the ALS fitting is better for the Gaussian noise, but if we have Laplace or Cauchy noise, the l_1 fitting is more efficient than the ALS. This result may be expected in the case of any tailed distribution, including the impulsive noises.
However, the algorithm of l_1 minimization based on the linear programming is slower in computation due to the use of the Matlab linear programming function *linprog*, but there are some methods to speed up the process (see [12] and [13]for using interior point method). We do not have any time to use these solving methods.

Bibliography

[1] ERCAN ENGIN KURUOGLU. *signal processing in α-stable noise environment: a least lp-norm approach*. PhD thesis, University of Cambridge, November 22 1998.

[2] J. G. McWhirter and I. K. Proudler Eds. Mathematics in signal processing 5. *Oxford University Press*, 2001.

[3] Nicholas D. Sidiropoulos Sergiy A. Vorobyov, Yue Rong and Alex B. Gershman. Robust iterative fitting of multilinear models. *IEEE*, 53(8), August 2005. transaction on signal processing.

[4] D. Middleton. Statistical-physical models of electromagnetic interference. *IEEE*, EMC-19(3):106–127, 1977. Transactions on Electromagnetic Compatibility.

[5] W.J.Fitzgerald A.C.Kokaram, R.D. Morris and P. J.W. Rayner. Interpolation of missing data in image sequences. *IEEE*, 4(11):1509–1519, 1995. Transactions on Image Processing.

[6] P. Tsakalides. *Array signal processing with α-Stable Distributions*. PhD thesis, University of Southern California, 1995.

[7] C. L.Nikias and M.Shao. *Signal Processing with α-Stable Distributions and Applications*. John Wiley Sons, 1995.

[8] Rasmus Bro. Parafac. tutorial and applications. *ELSEVIER*, March 1997. Chemometrics and intelligent laboratory systems.

[9] R.A Harshman. Foundation of the parafac procedure: Model and conditions for an 'explanatory' multi-mode factor analysis. *UCLA Working papers in phonetics*, 16(1), 1970.

[10] J.Chang J.D. Carroll. Analysis of individual differences in multidimensional scaling via n-way generalization of and eckart-young decomposition. *Psychometrika*, 35(283), 1970.

[11] R. Bro A. Smilde and P. Geladi. *Multi-way Analysis With Applications in Chemical Sciences*. John Wiley Sons, 2004.

[12] Nicholas D. Sidiropoulos and Rasmus Bro. Mathematical programming algorithms for regression-based nonlinear filtering in rn. *IEEE*, 47(3), March 1999. transaction on signal processing.

[13] Stephen Boyd and Lieven Vandeberghe. *Complex Optimization*. Cambridge Press, 2006.

[14] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. John Wiley Sons, 1966.

[15] L. De Lathauwer. Decompositions of a higher-order tensor in block terms. *SIAM J. Matrix Anal. Appl.*, 2006. submitted.