Katholieke Universiteit Leuven

Departement Elektrotechniek

ESAT-SISTA/TR 02-93

Data Preprocessing in Soft4s[1]

Oscar Barrero Mendoza and Bart De Moor[2]

November 6, 2002

[1]This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/sista/barrero/reports/Soft4sPre.ps*

[2]K.U.Leuven, Dept. of Electrical Engineering (ESAT), Research group SISTA/SCD, Kastelpark Arenberg 10 3001 Leuven, Belgium, Tel. 32/16/32 17 09, Fax 32/16/32 19 70, WWW: *http://www.esat.kuleuven.ac.be/sista*. E-mail: *bart.demoor@esat.kuleuven.ac.be*. This work was supported in part by the Soft4s project of IWT Belgium.

Contents

1 Outliers Detection in Regression Analysis **1**

1.1 Introduction 1
1.2 Regression Analysis 1
1.3 Outlier Diagnostics 5
1.3.1 Diagnostic Methods Least Squares Based 5
1.4 Residuals from Robust Fits 8
1.5 Description of LMS Procedure 11
1.5.1 Comparison between LS and LMS methods 12
1.6 Conclusions 14

2 Time Delay Estimation **19**

2.1 Introduction 19
2.2 Time Delay Estimation Techniques Based on Cross-Correlation 21
2.2.1 Introduction 21
2.2.2 Parametric Cross-Correlation in Time Domain (CRA) . 22
2.2.3 Non-Parametric Cross-Correlation in Frequency Domain for Single Input - Single Output (SISO) systems . 25
2.2.4 Non-Paramtric Cross-Correlation in Frequency Domain for Multiple Inputs - Single Output (MISO) systems . 29
2.3 Simulation Results 36
2.4 Time Delay Estimation on Polyamide Data 38
2.5 Conclusions 38

Abstract

In this report, we deal with two typical signal preprocessing problems: Outlier Detection, and Time Delay Estimation.

First, for outlier detection several methods like: Hat Matrix, Studentized Residuals, Standardized Residuals, Jackknifed Residuals, and the Mahalanobis Distance based on *Least Squares Regression*, and Least Median of Squares (LMS) based on *Robust Regression* are investigated and compared.

Second, for Time Delay Estimaton, we make a survey of different approaches like: Correlatin Analysis, Roth Window, Smoothed Coherence (SCOT), Phase Transform (PHAT), and Maximum Likelihood (ML) based on *Cross-Correlation Analysis*. Then, we introduce a new regularized method for MISO systems (RTDE) based on Impulse Response Estimation. Later, by mean of simulations, the classical methods and RTDE are compared and analyzed.

Finally, the methods mentioned above are applied to the Basf case studies; Ammonia, PIB, and Polyamide.

1 Outliers Detection in Regression Analysis

[1] [2]

1.1 Introduction

What are outliers and what is the outlier problem?

In a sample taken from a certain population, may happen that some are surprisingly far away from the main typical group. The experimenter is tempted to throw away the apparently erroneous values. On the contrary, he will undoubtedly admit that even if the population has a normal distribution there is a positive although extremely small probability that such values will occur in an experiment. It is rather because he feels that other explanations are more plausible, and that the loss in the accuracy of the experiment caused by throwing away a couple of good values is small compared to the loss caused by keeping even one bad value. The problem, then, is to introduce some degree of objectivity into the rejection of the outlying observation.

1.2 Regression Analysis

The purpose of regression analysis is to fit equations to observed variables. The classical linear model assumes a relation of the type

$$y_i = x_{i1}\theta_1 + ... + x_{ip}\theta_p + e_i, \quad for i = 1, ..., n \tag{1}$$

where n is the sample size. The variables $x_{i1}, \ldots, x_{ip}$ are called the explanatory variables, whereas the variable y_i is called the response variable. The error term e_i is assumed to be normally distributed with mean zero and unknown standard deviation σ. One tries to estimate the vector of unknown parameters

$$\theta = [\theta_1 \ldots \theta_p]^T \tag{2}$$

where the estimates θ are called the regression coefficients. Although the actual θ_j are unknown, one can multiply the explanatory variables with these $\hat{\theta}_j$ and obtain

$$\hat{y}_i = x_{i1}\hat{\theta}_1 + \ldots + x_{ip}\hat{\theta}_p \tag{3}$$

where $\hat{y}_i$ is called the predicted or estimated value of y_i. The residual r_i of the i_{th} case is the difference between what is actually observed and what is estimated:

$$r_i = y_i - \hat{y}_i \tag{4}$$

The most popular regression estimator corresponds to $LeastSquares(LS)$

$$\min_{\hat{\theta}} \sum_{i=1}^{n} r_i^2 \tag{5}$$

where the basic idea is to optimize the fit by making the sum of squares of the residuals very small.

Often, real data do not satisfy the classical assumptions, often with dramatic effects on the quality of the statistical analysis. As an illustration, let us look at the effect of outliers in the simplest regression model

$$y_i = \theta_1 x_i + \theta_2 + e_i \tag{6}$$

in which the slope θ_1 and the intercept θ_2 are to be estimated. In the simple regression model, one can make a plot of the (x_i, y_i), which is sometimes called a scatter-plot, in order to visualize the data structure.

Figure (1) is the scatter-plot of six points, (x_1, y_1), ... , (x_6, y_6), which is a straight line. Therefore, the LS solution fits the data exactly. However suppose that someone gets a wrong value of y_5 because of a copying or transmission error. Then (x_5, y_5) may be rather far away from the ideal line. Figure (1) displays such a situation, where the fifth point has moved down and away from its original position (indicated by 'o'). This point is called an OUTLIER in y-direction, and it has rather large influence on the LS line, which is quite different from the LS line in figure (1).

This phenomenon has received some attention in literature because one usually considers the y_i as observations and the $x_{i1}, \ldots, x_{ip}$ as fixed numbers (which is only true when the design has been given in advance) and because such vertical outliers often possess large positive or large negative residuals. Indeed, in this example the fifth point lies farthest away from the straight line, so its r_i given by eqn (2) is suspiciously large. Even in general multiple regression (1) with large p, where one cannot visualize the data, such outliers can often be discovered from the list of residuals or from so-called residual plots.

Usually also the explanatory variables $x_{i1}, \ldots, x_{ip}$ are observed quantities subject to random variability. Therefore, there is no reason why gross errors would only occur in the response variable y_i. In a certain sense it is even more likely to have an outlier in one of the explanatory variables $x_{i1}, \ldots, x_{ip}$ because usually p is greater than 1, and hence there are more opportunities for something to go wrong.

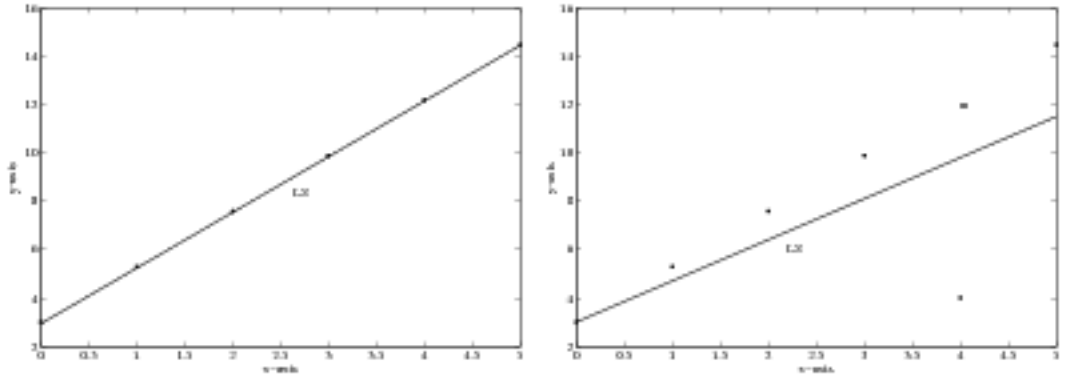


Figure 1: At the right Least Square Regression. At the left Least Squares Regression when the data has outliers

For the effect of such an outlier, let us look at an example of simple regression in figure (2). We can see that the third point has been moved, the resulting point is called

an **OUTLIER in x-direction (leverage point)**, and its effect effect on the least squares estimator is very large.

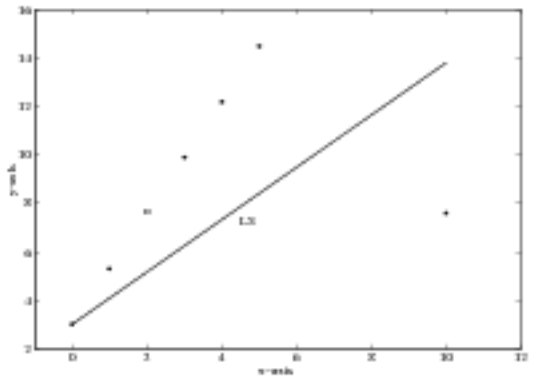


Figure 2: Input-Output signals for time delay estimation

To conclude, regression outliers (either in x or in y) pose a serious threat to standard least squares analysis. Basically, there are two ways out of this problem. The first, and probably most well-known, approach is to construct so-called *regressiondiagnostics*. Diagnostics are certain quantities computed from the data with the purpose of pinpointing influential points, after which these outliers can be removed or corrected, followed by an LS analysis on the remaining cases.

The other approach is *robustregression* [1], which tries to devise estimators that are not so strongly affected by outliers. Hence it looks at the residuals from a robust regression and then outliers may be identified, which usually cannot be done by means of the LS residuals.

Therefore, diagnostics and robust regression really have the same goals, only in the opposite order: When using diagnostics tools, one first tries to delete the outliers and then to fit the good data by least squares, whereas a robust analysis first tries to fit a regression to the majority of the data an then to discover the outliers as those points that possess large residuals from that robust solution.

One of the best methods for Robust Regression is the **Least Median of squares (LMS)** estimator [1]. The LMS method estimates the parameters by solving the nonlinear minimization problem

$$\min_{\hat{\theta}} \underset{i}{med}\, r_i^2 \tag{7}$$

where r_i are the residuals, and the estimated regression coefficients. That is, the estimator must yield the smallest value for the median of squared residuals computed for the entire data set. It turns out that this method is very robust to leverage points as well as outliers. Notice that robust regression techniques are also affected by the percentage of contaminated data. In figure (3), we can see that LMS has the best performance compared with other robust methods, nevertheless if we have *high percentage ($\geq 50\%$) of contaminated data, the method will fail for outlier and leverage detection.*

The minimization problem is solved by using the matlab function *fminunc* which uses a *Quasi-Newton method with a mixed quadratic and cubic line search procedure* to solve the optimization problem (for details see [3], fminunc). Since we have an nonlinear optimization problem, initial conditions have to be given, therefore we take randomly several sets (≈ 10) of initial regressors and the minimization problem is computed, then the one with the smallest median is chosen.

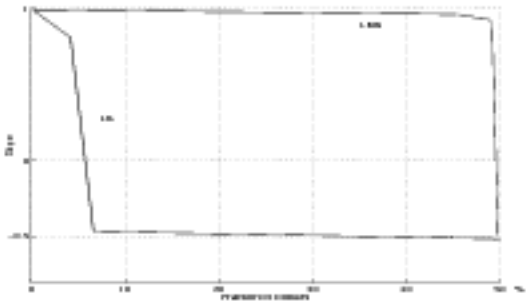


Figure 3: Breakdown plot for several regressors. The LMS only breaks down when 50% of the data is outlying. LS breaks down immediately because every point matters.

1.3 Outlier Diagnostics

[1]

Outlier diagnostics are statistics that focus attention on observations having a large influence on the least squares (LS) estimator, which is known to be non-robust. Several detection measures have been designed to detect individual cases or groups of cases that may differ from the bulk of the data. The field of diagnostics consists of a combination of numerical and graphical tools.

Many diagnostics are based on the residuals resulting from LS. However, this starting point may lead to useless results because of the following reason. By definition, LS tries to avoid large residuals. Consequently, one outlying case may cause a poor fit for the majority of the data because the LS estimator tries to accommodate this case at the expense of the remaining observations. Therefore, it is probable that an outlier has a small LS residual, especially when it is a leverage point (outlier in the x-direction). As a consequence, diagnostics based on LS residuals often fail to reveal such points.

1.3.1 Diagnostic Methods Least Squares Based

1. **Hat Matrix**

   Consider the following situation. Each individual $i = 1, \ldots, n$ is observed on $p + 1$ variables (that is, p explanatory variables and one response variable). The *n-by-1* vector of responses is denoted by $\boldsymbol{y} = (y_1, \ldots, y_n)^t$. The linear model states that

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \tag{8}$$

where $\boldsymbol{A}$ is the n-by-p matrix of observed variables

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{np} \end{bmatrix} \tag{9}$$

$\boldsymbol{x}$ is the vector of unknown parameters, and $\boldsymbol{e}$ is the error vector, that is, $\boldsymbol{e} = (e_1, \dots, e_n)^t$.

The hat matrix $\boldsymbol{H}$ mentioned above is defined by

$$\boldsymbol{H} = \boldsymbol{A}(\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t \tag{10}$$

The n-by-n matrix $\boldsymbol{H}$ matrix is called the *Hat Matrix* because it transforms the observed vector $\boldsymbol{y}$ into its LS estimate $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$.

The elements h_{ij} of $\boldsymbol{H}$ has a direct interpretation as the effect exerted by the j_{th} observation on $\hat{y}_i$. Especially the diagonal elements h_{ii} are of interest, because they equal $\partial\hat{y}_i/\partial y_i$. Therefore, they measure the effect of the i_{th} observation on its own prediction. A diagonal element h_{ii} of zero indicates a point with no influence on the fit. Now, we can determine potentially influential points by looking at the h_{ii} and paying particular attention to points for which $h_{ii} > 2p/n$, because these are candidates to be outliers.

The hat matrix is useful when the data set contains only a single observation with outlying x_i. However, when there are several such observations, it may easily happen that this does not show up in the corresponding h_{ii}. This is because it is based on a classical covariance matrix, which is not robust.

2. **Mahalanobis Distance**

   Defining a_i from (9) as

$$a_i = (a_{i1} \dots a_{ip}) \tag{11}$$

is a p-dimensional row vector. One computes the arithmetic mean $\bar{a}$ and the covariance matrix $\boldsymbol{C}$ of a_i:

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i, \quad \boldsymbol{C} = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^t(v_i - \bar{v}) \tag{12}$$

One then measures how far a_i is from $\bar{a}$ in the metric defined by $\boldsymbol{C}$, yielding

$$MD_i^2 = (a_i - \bar{a})\boldsymbol{C}^{-1}(a_i - \bar{a})^t, \tag{13}$$

which is called the *Squared Mahalanobis Distance* of the i_{th} case. The purpose of these MD_i^2 is to point to observations for which the explanatory part lies far from that of the bulk of the data. The value of MD_i^2 may be compared with 95% quantiles of the chi-squared distribution with p degrees of freedom so as to detect outliers.

On the other hand, it turns out that there is a one-to-one relationship between the Squared Mahalanobis Distance and the diagonal elements of the Hat Matrix (pp. 224, [1]), given by

$$MD_i^2 = (n-1)\left[h_{ii} - \frac{1}{n}\right] \tag{14}$$

3. **Standardized Residuals**

   Standardized residuals are defined as

$$s_r = \frac{r_i}{s} \tag{15}$$

   where s^2 is given by

$$s^2 = \frac{1}{n-p}\sum_{j=1}^{n} r_i^2 \tag{16}$$

   and r_i are the least squares residuals.

4. **Studentized Residuals**

   Studentized residuals are defined as

$$t_i = \frac{r_i}{s\sqrt{1-h_{ii}}} \tag{17}$$

5. **Jackknifed Residuals**

   Jackknifed residuals are defined as

$$t(i) = \frac{r_i}{s(i)\sqrt{1-h_{ii}}} \tag{18}$$

where $s(i)$ is the estimate of σ when the entire regression is run again without the i_{th} case.

The main disadvantage of the methods mentioned above, is that they are based on least squares which is no robust, it means, they are easily influenced by outliers. Hence, a robust regression technique is proposed in order to solve such a problem. Since it was mentioned above LMS (Least Median of Squares) is one of the best robust regression methods due to the fact that it is not strongly affected by outliers.

1.4 Residuals from Robust Fits

The residuals computed from a very robust estimator embody powerful information for detecting all the outliers present in the data. In particular, one can look at the standardized LMS residuals r_i/σ^*, where r_i are the residuals, and σ^* is an estimate of the standard deviation of the error. Of course, this estimate σ^* has to be robust itself, so it depends only on the good data and does not get blown up by the outliers. Therefore the standardized LMS residuals are dimensionless. A plot of these standardized residuals versus the estimated value of y_i or versus the index i gives a visual portrayal of this quantitative information. Whatever the dimension of the data set, these plots enable the data analyst to detect the possible outliers in a simple display.

Calculation of σ^*

First of all, an initial estimation of σ is made based on the minimal median and multiplied by a finite-sample-correction factor (which depends on n and on p) for the case of normal errors:

$$s^o = 1.4826\sqrt{\underset{i}{med}\, r_i^2(\hat{\theta})} \tag{19}$$

This preliminary estimation s^o is then used to determine a weight w_i for the i_ih observation, namely

$$y = \begin{cases} 1 & if \ \ ||r_i/s^o|| \geq 3.0 \\ 0 & otherwise \end{cases} \tag{20}$$

This is a weight function of the hard rejection type. It produces a radical dichotomy between the accepted and the rejected points, therefore in our case it will be used just like an indicator of possible outliers. By means of these weights, the final estimation of σ for LMS regression is calculated:

$$\sigma^* = \sqrt{\frac{\sum_{i=1}^{n} w_i r_i^2}{\left(\sum_{i=1}^{n} w_i - p\right)}} \tag{21}$$

The advantage of this formula for σ^* is that outliers do not influence the estimation anymore.

Example: Given the following data:

x	0	1	**2**	3	4	5
y	0	2.3	**1**	6.9	9.2	11.5

Outlier

Find the outlier data based on LMS regression.

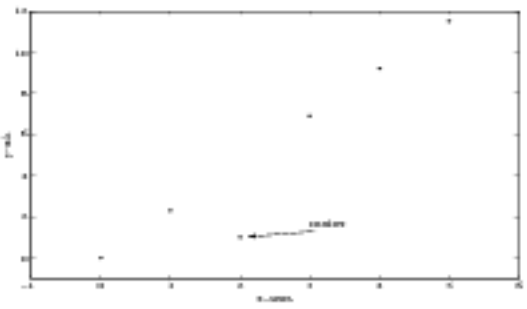


Figure 4: Data with one outlier

In figure (5) below we see the LMS regression compared with the original data, there the robustness of the method is clearly observed because it is not affected by the outlier.

Figure (6) shows the plot of the standardized residuals versus the estimated y and the indexes. The outlier is perfectly observed in both plots.At the right, is observed that there exist an outlier, and at the left is observed that such outlier is the $3^r d$ point.

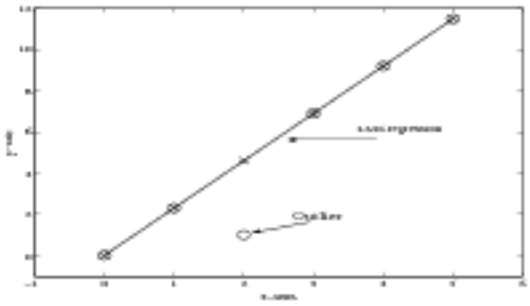


Figure 5: LMS Regression (x) - Original Data (o)

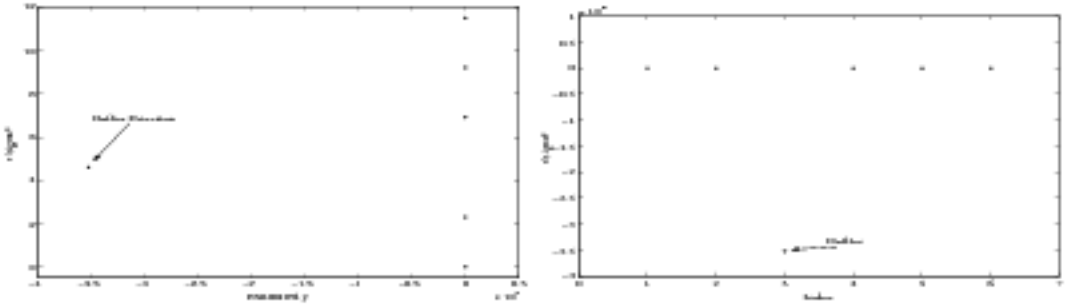


Figure 6: At the right, outlier identification. At the left, index outlier detection

Now suppose that the outlier is in the $x - axis$ (Leverage), so we have the following data set:

x	0	1	**8**	3	4	5
y	0	2.3	**4.6**	6.9	9.2	11.5

Outlier

In figure (7) one observes the robustness of the LMS regression against leverage points.

Figure (8) below shows the plot of the standardized residuals versus the estimated y and the indexes. The leverage point is perfectly observed in both plots.

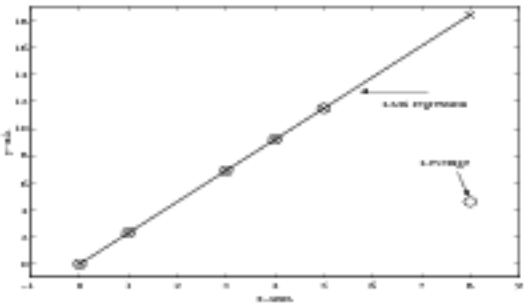


Figure 7: Original Data (o) - LMS Regression (x)

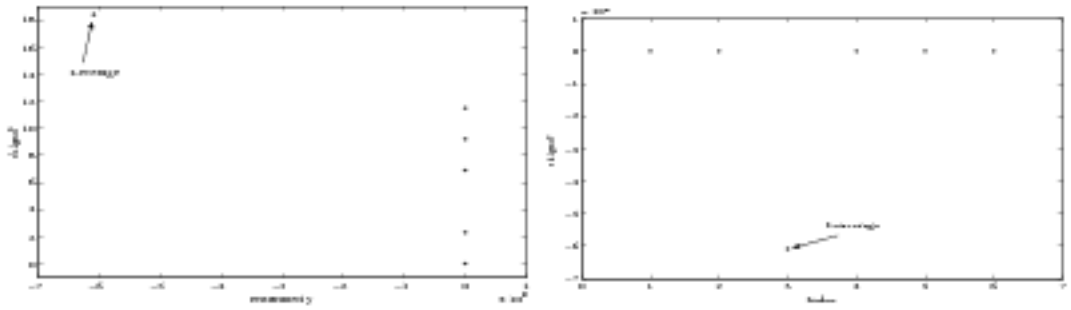


Figure 8: At the right, outlier identification. At the left, index outlier detection

1.5 Description of LMS Procedure

1. Check how much is the data contaminated, because if one of the variables is contaminated more than 50% the method fails.

2. Assuming a linear model as equation (8)

3. Then, to avoid numerical problems the data is normalized to unit variance and zero mean.

$$[Anormalized, ynormalized] = normalized(A, y)$$

4. If the data set is too long, or if it was taken on different operation points, one can divide the data set in several segments in order to avoid numerical troubles when solving the optimization problem. Then, repeat steps 5 to 7 segment-wise. Otherwise, one can skip this step.

5. Now, to solve the *LMS* optimization problem, the minimization is carried out taking randomly 10 different initial regressor sets. The re-

gressor set (x_i) which gets the smallest median over all data sets is chosen:

$$
\begin{aligned}
&Median_0 = 1;\\
&For\ i = 1:10\\
&\quad x_0 = randon(4,1);\\
&\quad [x_i, median] = lminunc(@LMS, x_0, options);\\
&\quad if median < median_0\\
&\quad\quad median_0 = median;\\
&\quad\quad x = x_i;\\
&\quad end\\
&end
\end{aligned}
$$

6. Compute the estimated output and the residuals

$$
\begin{aligned}
&yest = Anormalized * x;\\
&r = ynormalized - yest;
\end{aligned}
$$

7. Compute the standardized residuals (SR)

$$
\begin{aligned}
&so = 1.4826 * sqrt(median(r.^2));\\
&w = abs(r/so) <= 3.0;\\
&s = sqrt(w' * r.^2/(sum(w) - p));\\
&SR = r/s;\\
&w = abs(SR) <= 2.5;
\end{aligned}
$$

8. Plot the Standardized Residuals vs. the estimated output, and check visually for possible outlier or leverage points. Also you can plot SR vs. Indexes to detect exactly the suspected points, those with $SR > 3.0$.

9. Then, the possible outlier and leverage points are checked visually in the original data, so as to discard those points that seem to be bad, but they are not.

1.5.1 Comparison between LS and LMS methods

In this section we are going to apply the approaches seen above on the PIB Basf data. Then, a comparison and analysis of performance is going to be done. Since the data length is 20000, we are going to divide the original data set by 30 so as to avoid numerical problems, this is done for all methods.

Note: The Basf Data presented in this report has been scaled.

Figure (9) shows the scaled data that is going to be studied.

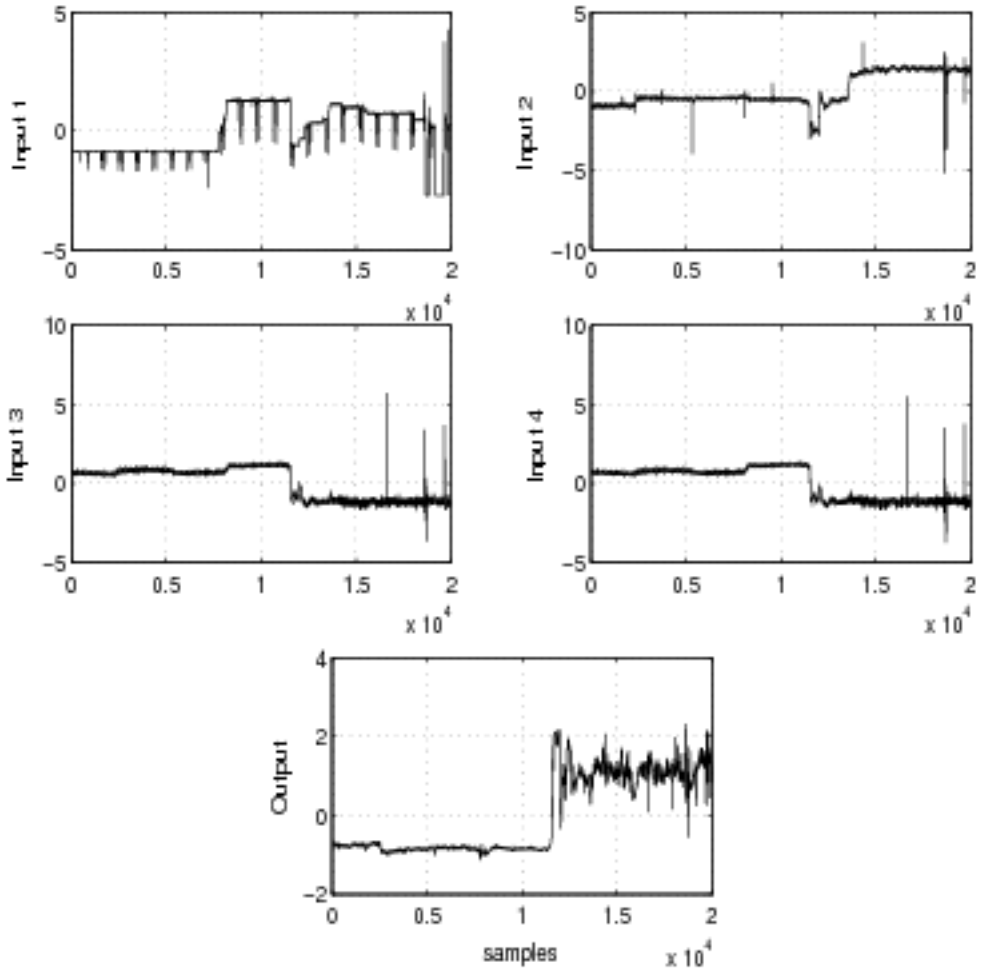


Figure 9: Input and Output PIB Signals

Analysis of Results

Table (1) shows the performance results for the outlier and leverage detection methods introduced before. These results were obtained from figures (10), (11), and (12). As can be seen, the methods based on least squares regression (studentized residuals, standardized residuals, and jackknifed residuals), hardly detect leverage and outlier points for highly contaminated data, this is because least squares regression is strongly influenced for outliers, as it was stated in the previous section. However, for lowly contaminated data they show a good performance for leverage detection. On the other hand, methods like Mahalanobis distance and hat matrix present good performance

for leverage detection whereas they make a totally wrong outlier detection. This is due to the fact that they just take into account the input data. Finally, the LMS performance for leverage as well as outlier detection is quite acceptable in all the cases. Nevertheless, a big difference in computing time is obtained.

Method	Leverage Detection Contaminated Data High — Low	Outlier Detection	Computing Time (sec)
Standardized Residuals	4 % — 80 %	1 right — 2 wrong	0.71
Studentized Residuals	4 % — 80 %	1 right — 2 wrong	2.01
Jackknifed Residuals	4 % — 60 %	1 right — 2 wrong	51.17
Mahalanobis Distance	78 % — 100 %	0 right — 3 wrong	1.93
Hat Matrix	80 % — 100 %	0 right — 3 wrong	1.93
LMS	62 % — 100 %	1 right — 0 wrong	1758.50

Table 1: Comparison of performance on PIB data

1.6 Conclusions

In this section we have introduced several classical methods for outlier and leverage detection. A comparison of performance was done on real data. The results have shown that for outlier-leverage detection in highly contaminated data the least squares based methods are very weak. Anyhow, they show a good performance when the data is lowly contaminated. Methods which take into account just input data work very well for that case, but, they do not perform fine when we have an input-output data set. On the other hand, despite LMS has shown to be very confident for outlier-leverage detection, it is very time consuming compare with the other approaches, moreover, for large scale data sets the optimization problem can become unfeasible. In ordet to try to overcome this problem, we have introduced an approach which consist in divide the original data into shorter segments, and then apply the outlier detection segment-wise. By doing this we can avoid problems like: feasibility of the LMS optimization problem, and percentage of contaminated data. This is valid for all the methods studied here.

As conclusion, for input-output data sets robust regression methods are the best for outlier detection, anyhow, more research has to be done in order to improve the performance for large scale data sets.

References

[1] Rousseeuw P. J., Leroy A. M., 1987, Robust Regression and Outlier Detection, Wiley series in probability and mathematical statistics.

[2] Barnet V., Lewis T., 1984, Outliers in Statistical Data, Wiley series in probability and mathematical statistics.

[3] Matlab Release 12, 2001, Optimization Toolbox, Version 2.1, Mathworks.

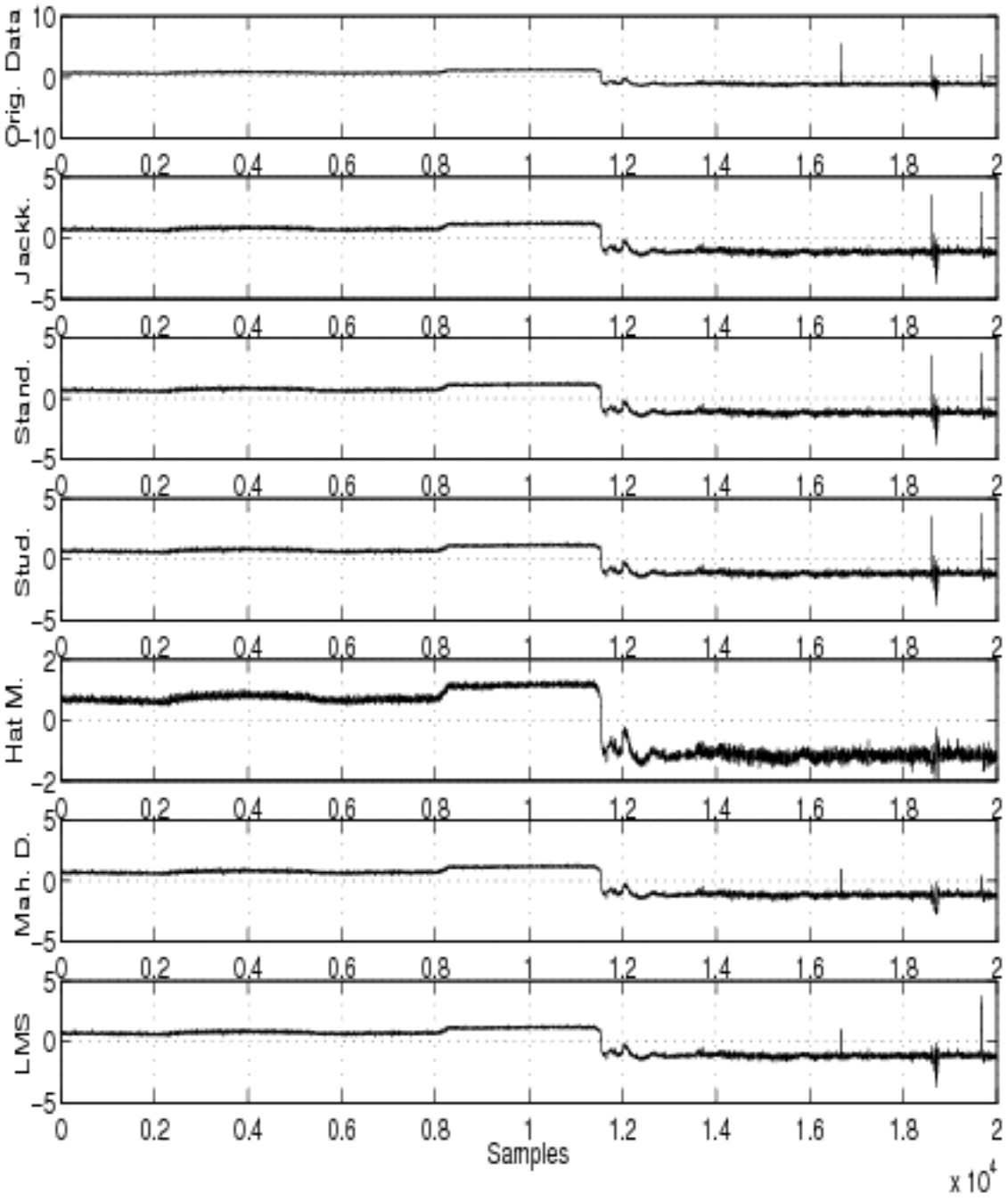


Figure 10: Comparison of performance for low contaminated data - Leverage Detection - Input 3

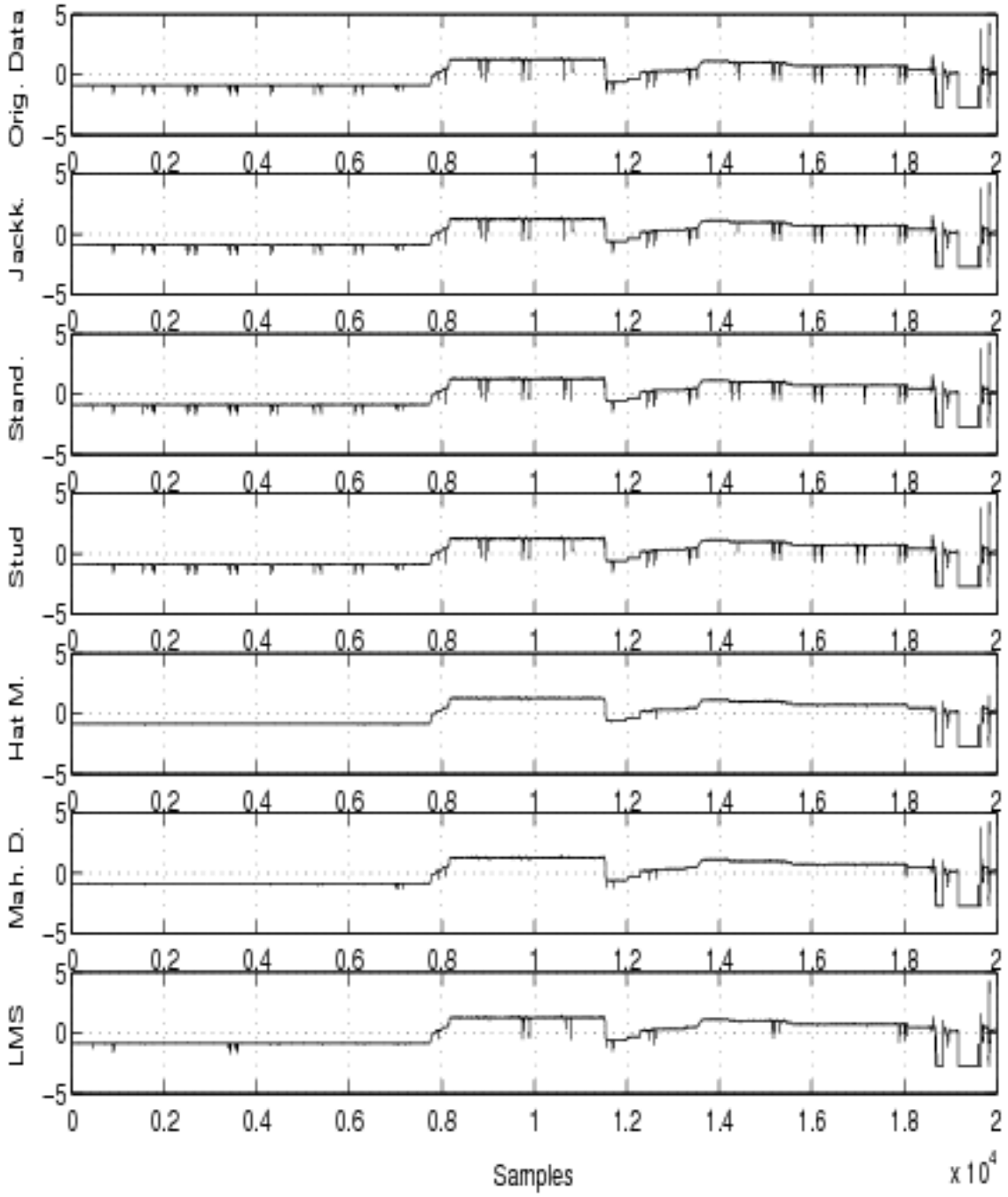


Figure 11: Comparison of performance for high contaminated data - Leverage Detection - Input 1

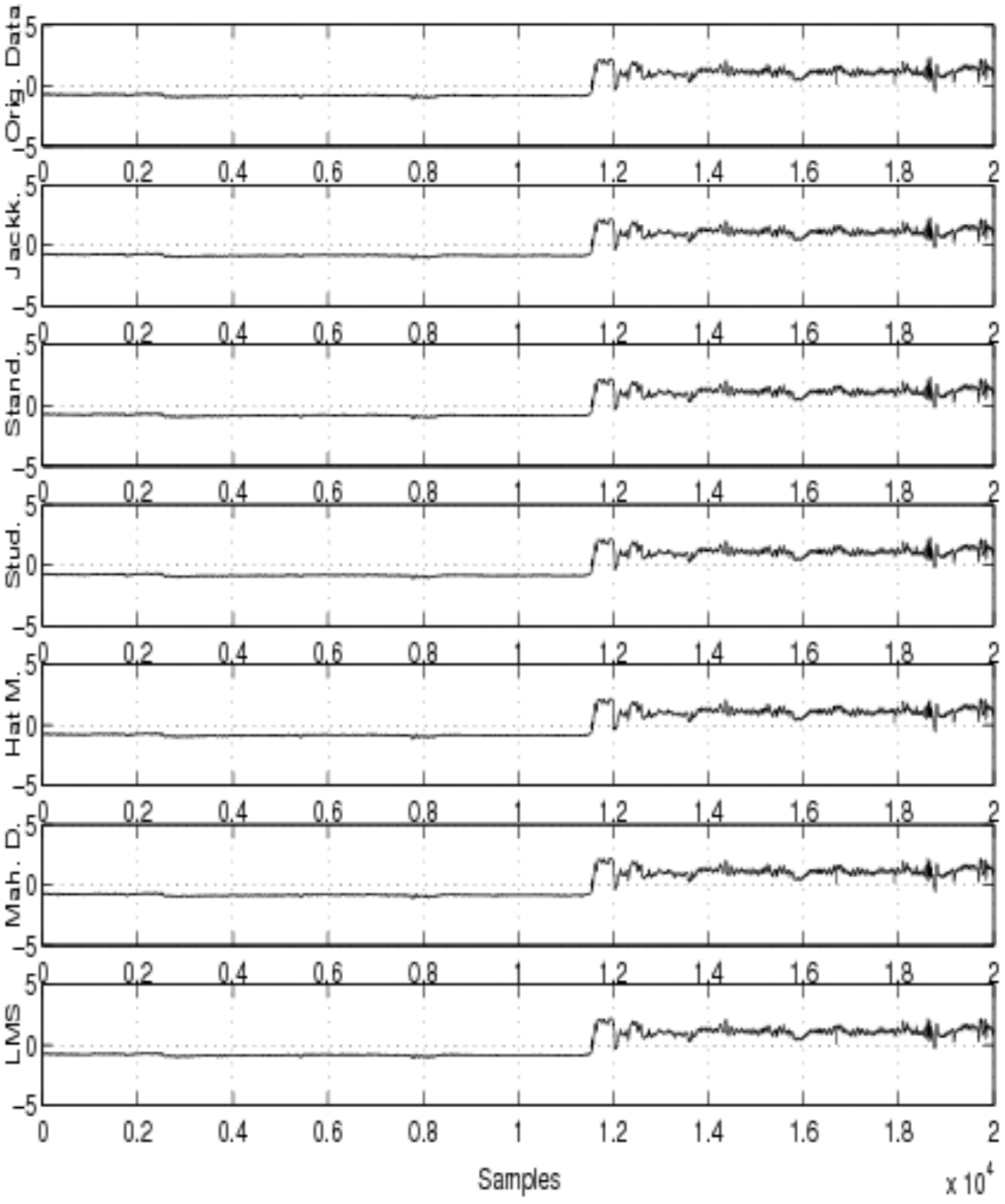


Figure 12: Comparison of performance for Outlier Detection - Output

2 Time Delay Estimation

2.1 Introduction

When one deals with data preprocessing for modeling process, the problem of measuring variables with a time delay is very often encountered. As an example, the measurement of the flow rate at the beginning of a pipeline. Sometimes it is not possible to install the sensor in a desire place due to problems such as high temperatures, pressures, .. etc. Therefore, it is necessary to install the sensors in some remote safe place, as a consequence there is a delay in observing the variable.

Measured signals with time delays cause difficulties when they are used for modeling process. For instance, in the linear static case, in a SISO (Single Input- Single Output) system, if we have an input-output response as shown in figure (13), where the output response y to a step input is given by:

$y = 1 - e^{(t-12)/22}$

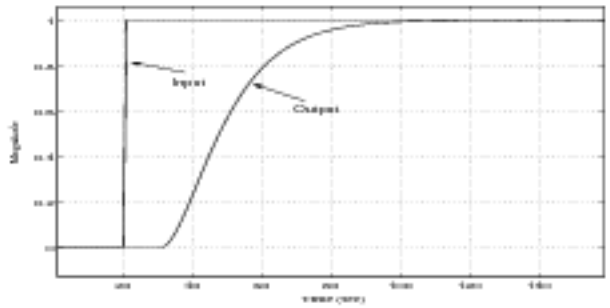


Figure 13: Step Input Response

The least squares solution for the static linear model will be:

$y = 0.83$

with a RMSE $= 0.2891$. Now, taking into account the time delay, the least squares solution will be: $y = 0.93$ with a RMSE $= 0.1408$. This shows that time delays are very important for linear modeling.

In the case of linear dynamic modeling, despite having to estimate the time delay as part of the model, it does not contribute to the further description of the dynamic behavior of the process. Since this delay introduce extra model parameter to be estimated, it is better to estimate them separately. For the previous example, for instance, the dynamic model with time delay is:

$$(1 - 0.9483z^{-}1)y(k) = (0.04848z^{-}20 + 0.004118z^{-}21)u(k) + e(k)$$

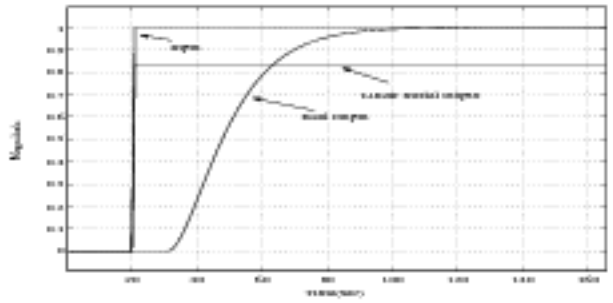


Figure 14: Linear Model Output with time delay

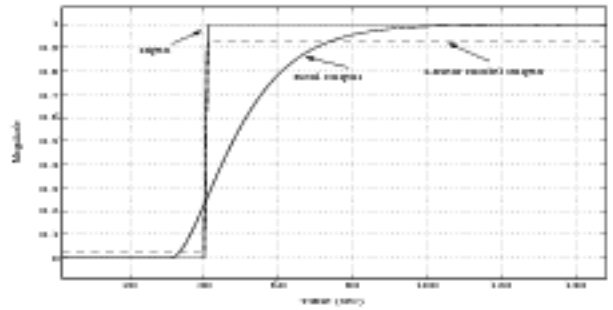


Figure 15: Linear Model Output without time delay

and the same dynamic model without time delay is:

$$(1 - 0.9483z^{-}1)y(k) = (0.04848 + 0.004118z^{-}1)u(k) + e(k)$$

that is a simpler model than the previous one because of the delay. While the first one has a delay of 21 samples; therefore, a dynamic order of 21, the last one has just a delay of one sample with the same result. with a RMSE = 0.0394

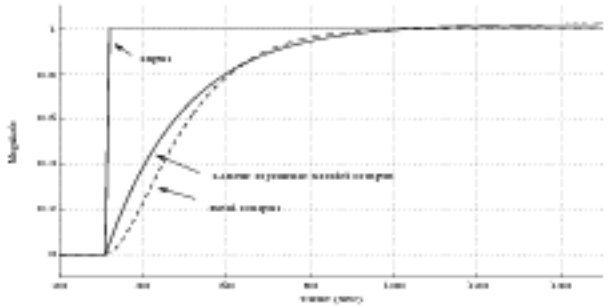


Figure 16: Linear Model Output with time delay

2.2 Time Delay Estimation Techniques Based on Cross-Correlation

2.2.1 Introduction

Let us assume that *x(k)* and *y(k)* are two sensor measurements satisfying

$$x(k) = s(k) + w_x(k) \tag{22}$$
$$y(k) = As(k-D) + w_y(k) \tag{23}$$

where $s(k)$ is an unknown signal and $s(k-D)$ is the same signal, delayed (or advanced) a time D, A is the relative amplitude gain, and $w_x(k)$, $w_y(k)$ are unknown noise sources. The problem is to find from finite length records $x(k)$ and $y(k)$, $k = 0, \ldots, N-1$, an estimate of the time delay D.

The basic approach to the solution of the time delay estimation problem is to shift $x(k)$ with respect to $y(k)$ and compare similarities between the two records at each shift. The best match will occur at a shift equal to D. Assuming that the noise sources are zero-mean independent stationary random processes, the fundamental operation adopted to "compare similarities" between $x(k)$ and $y(k)$ is the cross correlation.

The cross correlation between the two signals $x(k)$ and $y(k)$ is given by,

$$R_{xy}(\tau) = E[x(k)y(k+\tau)] \tag{24}$$

where E denotes expectation. The argument τ that maximizes 24 provides an estimate of delay.

Now, rewriting 24 we have

$$R_{xy}(\tau) = AR_{ss}(\tau - D) + R_{w_x,w_y}(\tau) \tag{25}$$

where

$$R_{ss} = E[s(k)s(k+\tau)] \tag{26}$$

is the autocorrelation of the signal $s(k)$, and $R_{w_x,w_y}(\tau)$ is the cross correlation between the two noise processes. If the noises are uncorrelated, $R_{xy}(\tau)$ will have a peak at $\tau = D$, the unknown delay. In practice due to effects of finite length estimates, and due to the presence of noise, the cross-correlation estimate may not have a sharp peak. In order to sharpen the peak the data may be pre-filtered; equivalently, we can multiply the estimated cross-correlation by a window function. In the following sections different approaches will be studied.

2.2.2 Parametric Cross-Correlation in Time Domain (CRA)

[7]

One common method for determining the time delay is to compute the Cross Correlation function between the input and output process signals. In this approach the input signal applied to the process is assumed to be a stationary, white, zero mean noise sequence with standard deviation s , and uncorrelated with the output noise signal.

The process response to an arbitrary input signal may be written as a convolution of the impulse response hk and the input uk:

$$y_k = \sum_{i=0}^{\infty} h_i u_{k-i} + n_k \tag{27}$$

where n_k is the output noise. The cross-correlation between the process output and process input is given by:

$$\psi_{yu}(\tau) = \lim_{N\to\infty} \frac{1}{N+1} \sum_{i=0}^{N} y_{(i+\tau)} u_i \tag{28}$$

Substitution of $y_{(i+l)}$ by 27, and using the fact that output noise and inputs are uncorrelated, we find:

$$\psi_{yu}(\tau) = \sum_{j=0}^{\infty} h_j \psi_{uu}(j-\tau) \tag{29}$$

with ψ_{uu} the auto-correlation function of the input. Since the input is white noise, its auto-correlation is just an impulse at zero. This leads to:

$$\psi_{yu}(\tau) = \sigma^2 h_\tau \tag{30}$$

Consequently the cross-correlation function obtained will have the shape of the impulse responses in case the input signal is a stationary, white, zero mean noise sequence. Otherwise the cross-correlation function is the convolution of the impulse responses with the auto-correlation function to the input signal.

The time delay can be found from the cross-correlation function by looking for the beginning of a sequence of values that differ significantly from zero. The time delay can then be compensated for by shifting the output signal in the right direction until the time delay is zero.

As an example the previous case is taken (see figure 13), where the time delay is between 15 and 17 samples.

In order to make a good time delay estimation, the original input signal has to be filtered in such a way that the assumption mentioned before is fulfilled, i.e., the input signal applied to the process is assumed to be a stationary, white, zero mean noise sequence with standard deviation s , and uncorrelated with the output noise signal, this process is called *whitening*. For the whitening process, the input is fitted to an AR (Auto-Regressive) model of order n_a as follows,

$$u_k + a_1 u_{k-1} + ... + a_{n_a} u_{k-n_a} = e_k \tag{31}$$

where

$$E[e_k] = 0 \text{ for all } k$$
$$E[e_k e_l] = \delta_{kl}\sigma^2 \text{ for all } k,\ l$$

with,

δ_{kl}= Kronecker delta
u_k, $k = 0, 1, ...N$ a sequence of scalar data
n_a = the order of the AR model

Therefore, an AR model can be considered as a whitening filter: The given datau_k is filtered so as to obtain a white noise sequence. Hence the discrete transfer function of the whitening filter is:

$$H(z) = 1 + a_1 z^{-1} + ... + a_{na} z^{-} na \tag{32}$$

Thereafter, since the determination of an appropriate order n_a is a non-trivial problem, one should try different orders, then by checking the plot of the cross-correlation function it is possible to observe which order n_a give a more clear picture about the signal time delay. Normally one can take $n_a = 10$ as first choice. Therefore, we get the following discrete transfer

function,

$$\begin{aligned} H(z) &= 1 - 0.1z^{-1} - 0.1z^{-2} - 0.1z^{-3} - 0.1z^{-4} - 0.1z^{-5} - 0.1z^{-6} - 0.1z^{-7} + \ldots \\ &\quad -0.1z^{-8} - 0.1z^{-9} - 0.1z^{-10} \end{aligned} \tag{33}$$

Notice that the input and output data have normalized (zero mean, unit variance) and then filtered by H(z) before the cross correlation function is computed. After whitening the data, one computes the cross correlation function as eqn 29 for $t = 1$ to $N - 1$, where N is the length of the data.

From figure 17A), the time delay can be easily detected as the maximum peak in 18 samples. Thus, the output signal can be shifted back over 18 samples to obtain a zero delay, and then find a model without delay. If the decision on the time delay turns out to be ambiguous, it will be better to under estimate it and then add extra delays during the modeling (try out different delays).

Another values of n_a were used to see how they affect the results on the cross correlation function (figures 17B), 17C)), there, one observes that the maximum peaks move between 15 and 20 samples, it means that the order n_a has an influence on the estimation as it was expected. However, by trying different values of n_a one can make a good estimation.

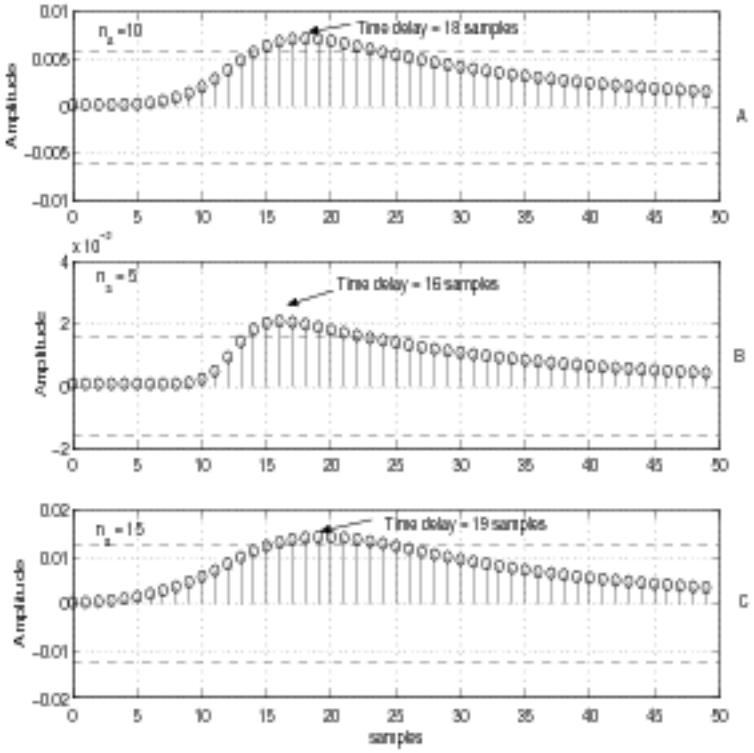


Figure 17: Cross Correlation Function for time delay estimation. The dashed lines are 99 confidence level of the cross correlation function.

2.2.3 Non-Parametric Cross-Correlation in Frequency Domain for Single Input - Single Output (SISO) systems

The cross correlation between $x(k)$ and $y(k)$ is related to the cross power spectral density function by the well-known Inverse Discrete Fourier Transform (IDFT) relationship

$$R_{xy}(k) = \frac{1}{N}\sum_{n=0}^{N-1} G_{xy}(\omega_n) e^{j\omega_n kT} \tag{34}$$

for $\omega_n = \frac{2\pi n}{NT}, n = 0, 1, 2, \ldots, N-1$, and $k \in \mathbb{Z}$

where,

$$G_{xy}(\omega_n) = Y(\omega_n)U(\omega_n)^* \tag{35}$$

with $Y(\omega_n)$ and $U(\omega_n)$ the DFT of $y(k)$ and $x(k)$ respectively, * denotes the complex conjugate, N the lenght of data, and T the sampling time.

Then, the generalized cross-correlation between $y(k)$ and $x(k)$ is given by

$$Gf_{xy}(\omega_n) = \Psi_g(\omega_n) G_{xy}(\omega_n) \tag{36}$$

as depicted in figure 18

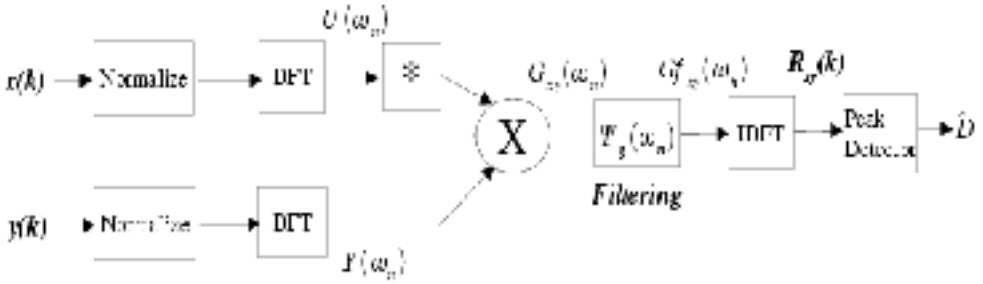


Figure 18: Generalized cross-correlation scheme for time delay estimation

In practice, only an estimate $\hat{G}_{xy}(\omega_n)$ of $G_{xy}(\omega_n)$ [4] can be obtained because of the finite observations of $x(k)$ and $y(k)$. Consequently, equation 34 can be rewritten as

$$\hat{R}_{xy}(k) = \sum_{n=0}^{N-1} \Psi_g(\omega_n) \hat{G}_{xy}(\omega_n) e^{j\omega_n kT} \tag{37}$$

On the other hand, under ideal condition where, $\hat{G}_{xy}(\omega_n) \cong G_{xy}(\omega_n)$, $\Psi_g(\omega_n)$ should be chosen to ensure a large sharp peak in $\hat{R}_{xy}(k)$ rather than a broad one in order to ensure good time-delay resolution. However, sharp peaks are more sensitive to errors introduced by finite observation time, particularly in case of low Signal to Noise ratio (SNR). Thus, as with other spectral estimation problems, the choice of $\Psi_g(\omega_n)$ is a compromise between good resolution and stability.

In this report we will study the performance of different $\Psi_g(\omega_n)$ on the ammonia and PIB data of BASF.

Window Functions for Time Delay Estimation

Some of the most used window functions are *ROTH, SCOT, PHAT, and Maximum Likelihood (ML)* [1], which will be explained and compared later in this section. On the other hand, different choices of the window function lead to different estimates, as it will be shown:

1. **The Roth Window**

   The weighting window proposed by Roth is

$$\Psi_{gR}(\omega_n) = \frac{1}{\hat{G}_{xx}(\omega_n)} \tag{38}$$

   Replacing in 37 we have

$$\hat{R}_{xy}(k) = \sum_{n=0}^{N-1} \frac{\hat{G}_{xy}(\omega_n)}{\hat{G}_{xx}(\omega_n)} e^{j\omega_n kT} \tag{39}$$

   Equation 39 estimates the impulse response of the optimum linear (Wiener-Hopf) filter

$$H_m(\omega_n) = \frac{G_{xy}(\omega_n)}{G_{xx}(\omega_n)} \tag{40}$$

   Now, if $w_x(k) \neq 0$ and $w_y(k) = 0$

$$\hat{G}_{xx}(\omega_n) = G_{ss}(\omega_n) + G_{w_x w_x}(\omega_n) \tag{41}$$

   then,

$$\hat{R}_{xy}(k) = \delta(k - D) \circledast \sum_{n=0}^{N-1} \frac{\alpha G_{ss}(\omega_n)}{G_{ss}(\omega_n) + G_{w_x w_x}(\omega_n)} e^{j\omega_n kT} \tag{42}$$

   Therefore, except when $G_{w_x w_x}(\omega_n)$ equals any constant (including zero) times $G_{ss}(\omega_n)$, the delta function will again be spread out. The ROTH window has the desirable effect of suppressing those frequency regions where $G_{w_x w_x}(k)$ is large and $\hat{G}_{xy}(\omega_n)$ is more likely to be in error.

2. **The Smoothed Coherence[1] Transform (SCOT) Window**

   Errors in $\hat{G}_{xy}(\omega_n)$ may be due to frequency bands where $G_{w_y w_y}(k)$ is large, as well as bands where $G_{w_x w_x}(\omega_n)$ is large. One is, therefore, uncertain whether to form $\Psi_{gR}(\omega_n) = 1/G_{xx}(\omega_n)$ or $\Psi_{gR}(\omega_n) = 1/G_{yy}(\omega_n)$; hence, the SCOT selects

[1]Coherence is a function of frequency with values between 0 and 1 that indicate how well the input X corresponds to the output Y at each frequency

$$\Psi_{gS}(\omega_n) = \frac{1}{\sqrt{G_{xx}(\omega_n)G_{yy}(\omega_n)}} \tag{43}$$

this gives

$$\hat{R}_{xy}(k) = \sum_{n=0}^{N-1} \hat{\gamma}_{xy}(\omega_n) e^{j\omega_n kT} \tag{44}$$

where the coherence estimate

$$\hat{\gamma}_{xy}(\omega_n) = \frac{\hat{G}_{xy}(\omega_n)}{\sqrt{\hat{G}_{xx}(\omega_n)\hat{G}_{yy}(\omega_n)}} \tag{45}$$

The SCOT can be interpreted through figure 18 as pre-whitening filters followed by a cross correlation. The SCOT exhibits the same spreading as the Roth window.

3. **The Phase Transform (PHAT)**

The PHAT uses the following weighting window

$$\hat{\Psi}_{gP}(\omega_n) = \frac{1}{|G_{xy}(\omega_n)|} \tag{46}$$

then we have

$$\hat{R}_{xy}(k) = \sum_{n=0}^{N-1} \frac{\hat{G}_{xy}(\omega_n)}{|\hat{G}_{xy}(\omega_n)|} e^{j\omega_n kT} \tag{47}$$

For the equation 25, with uncorrelated noise (i.e., $G_{w_x w_y}(\omega_n) = 0$),

$$|G_{xy}(\omega_n)| = AG_{ss}(\omega_n) \tag{48}$$

Ideally, when $\hat{G}_{xy}(\omega_n) = G_{xy}$,

$$\frac{\hat{G}_{xy}(\omega_n)}{|\hat{G}_{xy}(\omega_n)|} = e^{j2\pi kD} \tag{49}$$

has unit magnitude and

$$\hat{R}_{xy}(k) = \delta(k - D) \tag{50}$$

In practice, however, when $\hat{G}_{xy}(\omega_n) \neq G_{xy}$ the estimate of $\hat{R}_{xy}(k)$ will not be a delta function. Another defect is that it weights $\hat{G}_{xy}(\omega_n)$ as the inverse of $G_{ss}(\omega_n)$. Thus, errors are accentuated where signal power is smallest.

4. **The Maximum-Likelihood window (ML)**

   The ML estimator selects as the estimate of delay [1] the value of k at which

$$\hat{R}_{xy}(k) = \sum_{n=0}^{N-1} \hat{G}_{xy}(\omega_n) \frac{1}{|G_{xy}(\omega_n)|} \frac{|\gamma_{xy}(\omega_n)|^2}{[1 - |\gamma_{xy}(\omega_n)|^2]} e^{j\omega_n kT} \tag{51}$$

   achieves a peak. Therefore the weighting function is

$$\Psi_{gML}(\omega_n) = \frac{1}{|G_{xy}(\omega_n)|} \frac{|\gamma_{xy}(\omega_n)|^2}{[1 - |\gamma_{xy}(\omega_n)|^2]} \tag{52}$$

   where $|\gamma_{xy}(\omega_n)|^2 \neq 1$, achieves the ML estimator. When $|G_{xy}(\omega_n)|$ and $|\gamma_{xy}(\omega_n)|^2$ are unknown, they have to be estimated.

   Note that the ML estimator, weights the phase according to the strength of the coherence. Moreover, the ML estimator shows the best performance under low SNR of the signals, this makes the ML estimator one of the most used methods.

2.2.4 Non-Paramtric Cross-Correlation in Frequency Domain for Multiple Inputs - Single Output (MISO) systems

When we want to make a time delay estimation for a determined input-output pair in a MISO system using one of the methods mentioned above, we see that is hard for them to find a reliable estimation. This is, because the influences of the other inputs respect to such output are not taken into account, therefore, the estimation of the generalized cross-correlation function is not very accurate.

To get over such problems refered before, we proposed a new method based on the Impulse Response Estimation for MISO systems.

1. **Problem Formulation**

   Let us assume that $\boldsymbol{U} \in \boldsymbol{R}^{N \times p}$ and $\boldsymbol{Y} \in \boldsymbol{R}^{N \times 1}$ are measured data sequences of a MISO linear system as shown in figure 19

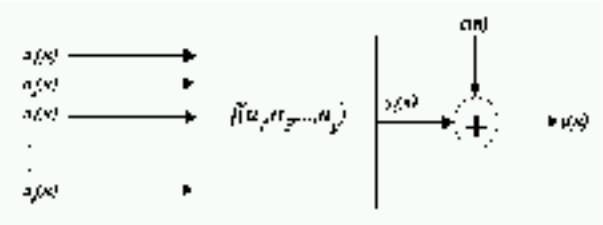


Figure 19: MISO system in time domain

satisfying

$$y(n) = f(u_1(n), u_2(n), \ldots, u_p(n)) + e(n), \quad 0 < n < N-1 \tag{53}$$

where

$\boldsymbol{U} = [u_1(n)\ u_2(n)\ \ldots\ u_p(n)]$ are the input variables.
$\boldsymbol{Y} = y(n)$ is the output variable.
$e(n) =$ Unknown noise signal.

The problem is to estimate the time delay between the i_{th} input with the output, taking into account that $y(n)$ is also affected by the other inputs and by the noise signal $e(n)$.

Some classical methods like *PHAT, ROTH, SCOT, ML* compute time delay between two signals in a SISO system, by estimating the impulse response based on the *Generalized Cross-Correlation function*, eqn 37. Therefore, one can extend this procedure to the MISO case taking into account the coupling between inputs and output as follows

Let us assume the following linear system:

where $U_1(s), U_2(s), ..., U_p(s)$ are the inputs of the system, $Y(s)$ is the output of the system, $Y_1(s)$ is the measured output of the system, $E(s)$ is a random noise signal and *H(s)* is a set of transfer functions

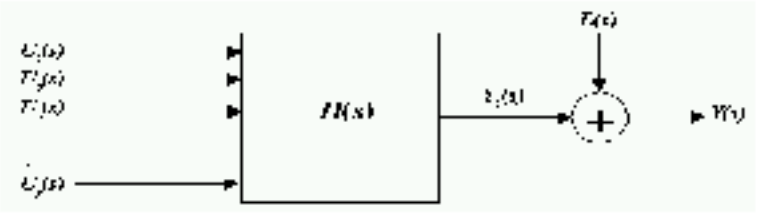


Figure 20: MISO system in Laplace domain

which relate each input with the output. Therefore, we can express the measured output $Y_1(s)$ as follows

$$Y(s) = H_1(s)U_1(s) + H_2(s)U_2(s) + \ldots + H_p(s)U_p(s) + E(s) \quad (54)$$

Now, if we assume that there exist time delay between the inputs and the output of the system we get the following scheme

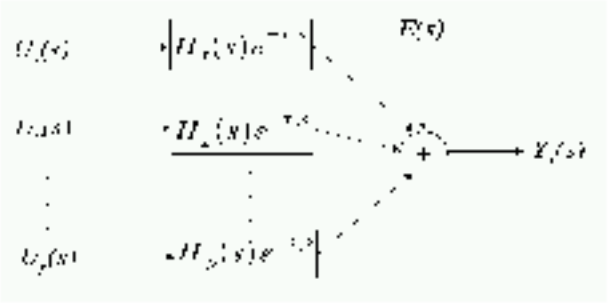


Figure 21: MISO system in Laplace domain with time delay

hence, 54 can be rewritten as

$$\begin{aligned} Y(s) &= H_1(s)e^{-\tau_1 s}(s)U_1(s) + H_2(s)e^{-\tau_2 s}U_2(s) + \ldots \\ &\quad + H_p(s)e^{-\tau_p s}U_p(s) + E(s) \end{aligned} \quad (55)$$

in the Discrete Time Fourier domain

$$\begin{aligned} Y(e^{j\omega}) &= H_1(e^{j\omega})e^{-j\omega\tau_1}U_1(e^{j\omega}) + H_2(e^{j\omega})e^{-j\omega\tau_2}U_2(e^{j\omega}) + \ldots \\ &\quad + H_p(e^{j\omega})e^{-j\omega\tau_p}U_p(e^{j\omega}) + E(e^{j\omega}) \end{aligned} \quad (56)$$

Thus, computing the impulse responses, i.e.

$$H_1(e^{j\omega})e^{-j\omega\tau_1},\ H_2(e^{j\omega})e^{-j\omega\tau_2},\ \ldots,\ H_p(e^{j\omega})e^{-j\omega\tau_p}$$

it is possible to make a time delay estimation between the inputs $u_p(n)$ and the output $y(n)$.

To make such estimation, we first need to define a set of p equations in order to solve the problem. This can be done with the help of the *Cross-Correlation function*.

2. **Cross-Correlation function**

The Cross-Correlation in the discrete time domain is defined as

$$R_{uy}(k) = E[u(n+k)y(n)] \qquad 0 < n < N-1,\ -T < k < T \tag{57}$$

applying the Discrete Fourier Transform in 57

$$G_{UY}(e^{j\omega}) = Y(e^{j\omega})U^*(e^{j\omega}) \tag{58}$$

where * denotes conjugate, $Y(e^{j\omega})$ and $U(e^{j\omega})$ are the Fourier transform of $y(k)$ and $u(k)$, and $G_{UY}(e^{j\omega})$ is the *Cross-Power Spectrum*.

Now, we can express the output of a linear system as a function of the impulse response $h(k)$ as follows

$$y(k) = h(k) * u(k) \tag{59}$$

where '$*$' defines convolution, then, by applying the discrete Fourier transform we have

$$Y(e^{j\omega}) = H(e^{j\omega})U(e^{j\omega}) \tag{60}$$

Replacing 60 in 58 we can rewrite the cross-power spectrum as

$$G_{UY}(e^{j\omega}) = H(e^{j\omega})U(e^{j\omega})U^*(e^{j\omega}) \tag{61}$$

3. **Regularized Time Delay Estimation For MISO systems - RTDE**

As mentioned above, it is needed to define a set of p equations to compute the estimation of the impulse responses of the system, hence, if one computes the *Cross-Power Spectrum* between each input $U(e^{j\omega})_i$ with the output $Y(e^{j\omega})$ replacing 61 in 56, this requirement will be fulfilled, as follows

$$\begin{aligned}
G_{U_1Y}(e^{j\omega}) &= H_1(e^{j\omega})e^{-j\omega\tau_1}U_1(e^{j\omega})U_1^*(e^{j\omega}) + \dots \\
&\quad + H_p(e^{j\omega})e^{-j\omega\tau_p}U_p(e^{j\omega})U_1^*(e^{j\omega}) + E(e^{j\omega})U_1^*(e^{j\omega}) \\
G_{U_2Y}(e^{j\omega}) &= H_1(e^{j\omega})e^{-j\omega\tau_1}U_1(e^{j\omega})U_2^*(e^{j\omega}) + \dots \\
&\quad + H_p(e^{j\omega})e^{-j\omega\tau_p}U_p(e^{j\omega})U_2^*(e^{j\omega}) + E(e^{j\omega})U_2^*(e^{j\omega}) \\
&\vdots \quad \vdots \quad \vdots \\
G_{U_pY}(e^{j\omega}) &= H_1(e^{j\omega})e^{-j\omega\tau_1}U_1(e^{j\omega})Up2^*(e^{j\omega}) + \dots \\
&\quad + H_p(e^{j\omega})e^{-j\omega\tau_p}U_p(e^{j\omega})U_p^*(e^{j\omega}) + E(e^{j\omega})U_p^*(e^{j\omega}) \quad (62)
\end{aligned}$$

then, rewriting 62 in matrix form

$$\begin{aligned}
\begin{bmatrix} G_{U_1Y}(e^{j\omega}) \\ \vdots \\ G_{U_pY}(e^{j\omega}) \end{bmatrix} &= \begin{bmatrix} U_1(e^{j\omega})U_1^*(e^{j\omega}) & \dots & U_p(e^{j\omega})U_1^*(e^{j\omega}) \\ \vdots & \vdots & \vdots \\ U_1(e^{j\omega})U_p^*(e^{j\omega}) & \dots & U_p(e^{j\omega})U_p^*(e^{j\omega}) \end{bmatrix} \begin{bmatrix} H_1(e^{j\omega})e^{-j\omega\tau_1} \\ \vdots \\ H_p(e^{j\omega})e^{-j\omega\tau_p} \end{bmatrix} + \dots \\
&\quad + \begin{bmatrix} E(e^{j\omega})U_1(e^{j\omega}) \\ \vdots \\ E(e^{j\omega})U_p(e^{j\omega}) \end{bmatrix} \quad (63)
\end{aligned}$$

where $U_i(e^{j\omega})U_i^*(e^{j\omega})$ is the *Auto-Power Spectrum* of $U_i(e^{j\omega})$ and $U_i(e^{j\omega})U_j^*(e^{j\omega})$ is the *Cross-Power Spectrum* between $U_i(e^{j\omega})$ and $U_j(e^{j\omega})$. Now, defining

$$\boldsymbol{G} = \begin{bmatrix} G_{U_1Y}(e^{j\omega}) \\ \vdots \\ G_{U_pY}(e^{j\omega}) \end{bmatrix} \quad (64)$$

$$\boldsymbol{UU} = \begin{bmatrix} U_1(e^{j\omega})U_1^*(e^{j\omega}) & \dots & U_p(e^{j\omega})U_1^*(e^{j\omega}) \\ \vdots & \vdots & \vdots \\ U_1(e^{j\omega})U_p^*(e^{j\omega}) & \dots & U_p(e^{j\omega})U_p^*(e^{j\omega}) \end{bmatrix} \quad (65)$$

$$\boldsymbol{H} = \begin{bmatrix} H_1(e^{j\omega})e^{-j\omega\tau_1} \\ \vdots \\ H_p(e^{j\omega})e^{-j\omega\tau_p} \end{bmatrix} \tag{66}$$

$$\boldsymbol{E} = \begin{bmatrix} E(e^{j\omega})U_1(e^{j\omega}) \\ \vdots \\ E(e^{j\omega})U_p(e^{j\omega}) \end{bmatrix} \tag{67}$$

equation 63 can be written as

$$\boldsymbol{G} = \boldsymbol{UUH} + \boldsymbol{E} \tag{68}$$

which is a least squares problem in $\boldsymbol{H}$. Therefore, $\boldsymbol{UU}$ has to be invertible; it means that the inputs can not be correlated.

Thus, after obtaining $\boldsymbol{H}$, we apply the *Inverse Discrete Time Fourier Transform (IDTFT)* to get the delayed impulse response in the time domain.

$$\boldsymbol{R_{uy}} = IDFT(\boldsymbol{H}) \tag{69}$$

and finally, one can estimate the time delay as the instant when the impulse response becomes bigger than certain threshold, such a threshold is defined as the 3σ of the impulse response in the past time, i.e., $time < 0$.

4. **Impulse Response Estimation**

In the previous section we developed the procedure based on Discrete Time Fourier Transform (DTFT), but to compute this signal transformation we need information of the signal in time from $-\infty$ to ∞. In practice, however, we only have access to a finite number of data samples. For this class of sequences $x(k)$, the Discrete-Fourier Transform (DFT) is defined as follows,

The DFT of a signal $x(k)$ for $k = 0, 1, 2, \ldots, N-1$ is given by

$$X_N(\omega_n) = \sum_{n=0}^{N-1} x(k)e^{-j\omega_n kT} \tag{70}$$

for $\omega_n = \frac{2\pi n}{NT}, n = 0, 1, 2, \ldots, N-1$ and sampling time $T \in \mathbb{R}$.

Now, replacing the $DTFT$ by DFT in 64, 65, and 66

$$\begin{bmatrix} \hat{G}_{U_1Y}(\omega_n) \\ \vdots \\ \hat{G}_{U_pY}(\omega_n) \end{bmatrix} = \begin{bmatrix} \hat{U}_1(\omega_n)U_1^*(\omega_n) & \ldots & \hat{U}_p(\omega_n)\hat{U}_1^*(\omega_n) \\ \vdots & \vdots & \vdots \\ \hat{U}_1(\omega_n)\hat{U}_p^*(\omega_n) & \ldots & \hat{U}_p(\omega_n)\hat{U}_p^*(\omega_n) \end{bmatrix} \begin{bmatrix} \hat{H}_1(\omega_n)e^{-j\omega_n\tau_1} \\ \vdots \\ \hat{H}_p(\omega_n)e^{-j\omega_n\tau_p} \end{bmatrix} + \ldots + \begin{bmatrix} E(\omega_n)\hat{U}_1(\omega_n) \\ \vdots \\ E(\omega_n)\hat{U}_p(\omega_n) \end{bmatrix} \tag{71}$$

Now, defining

$$\hat{\boldsymbol{G}} = \begin{bmatrix} \hat{G}_{U_1Y}(\omega_n) \\ \vdots \\ \hat{G}_{U_pY}(\omega_n) \end{bmatrix} \tag{72}$$

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{U}} = \begin{bmatrix} \hat{U}_1(\omega_n)\hat{U}_1^*(\omega_n) & \ldots & \hat{U}_p(\omega_n)\hat{U}_1^*(\omega_n) \\ \vdots & \vdots & \vdots \\ \hat{U}_1(\omega_n)\hat{U}_p^*(\omega_n) & \ldots & \hat{U}_p(\omega_n)\hat{U}_p^*(\omega_n) \end{bmatrix} \tag{73}$$

$$\hat{\boldsymbol{H}} = \begin{bmatrix} \hat{H}_1(\omega_n)e^{-j\omega_n\tau_1} \\ \vdots \\ \hat{H}_p(\omega_n)e^{-j\omega_n\tau_p} \end{bmatrix} \tag{74}$$

$$\boldsymbol{E} = \begin{bmatrix} E(\omega_n)\hat{U}_1(\omega_n) \\ \vdots \\ E(\omega_n)\hat{U}_p(\omega_n) \end{bmatrix} \tag{75}$$

The estimation of the cross and auto power spectrum in 72 and 73 is done based on P. Welch's method [3]. Therefore, equation 71 can be written as

$$\hat{\boldsymbol{G}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{U}}\hat{\boldsymbol{H}} + \boldsymbol{E} \tag{76}$$

Since $\hat{\boldsymbol{UU}}$ is an estimation of $\boldsymbol{UU}$, equation 76 can be written as follows

$$\hat{\boldsymbol{G}} = [\boldsymbol{UU} + \delta\boldsymbol{UU}]\hat{\boldsymbol{H}} + \boldsymbol{E} \tag{77}$$

where $\delta\boldsymbol{UU}$ are the uncertainties of the matrix $\hat{UU}$. Then, equation 77 can be solved for each frequency ω_n as a *Regularized Least Squares* problem for $\hat{\boldsymbol{H}}$ [5] as follows

$$\hat{\boldsymbol{H}} = (\hat{\boldsymbol{UU}}^*\hat{\boldsymbol{UU}} + \alpha\boldsymbol{I_n})^{-1}\hat{\boldsymbol{UU}}^*\hat{\boldsymbol{G}} \tag{78}$$

where α is a regularization parameter that can be used as a low-pass filter; it means, α has to be small for low frequencies and big for high frequencies, see figure 22.

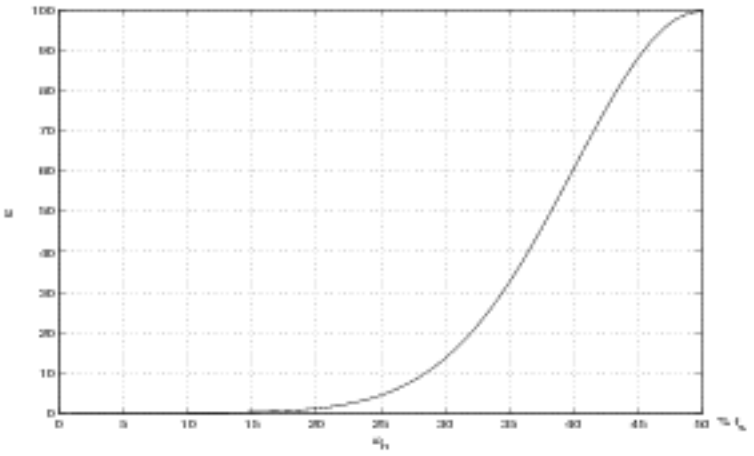


Figure 22: Weigthing function $\alpha(\omega_n)$ as a low pass filter for the Regularized Least Squares problem.

Afterwards, $\hat{R}_{uy}$ is computed by applying the IDFT of $\hat{H}(\omega_n)$, then, we choose as time delay the maximum value of $\hat{R}_{uy}(k)$ for the positive lags. Notice, that the time delay estimation is not done as it was proposed in item 3 of section 2.2.4, because the standard deviation sometimes does not correspond to the real one; approximately zero, due to the estimation of the power spectrum.

2.3 Simulation Results

Here we present computer simulation results we have conducted to investigate the performance of the method based on the impulse response estimation of a

MISO system (RTDE). For comparison purposes, the ML method has been chosen. For this computer experiment, we generated a set of 4 inputs - 1 output data, perturbed with white noise, and delayed 5, 20, 9, and 13 samples respectively (see fig. 23). The length of the data was 4900 points. The signal to noise ratio (SNR) is defined as $20log_{10}(\sigma_s/\sigma_n)$, where σ_s and σ_n are the standard deviation of signal and noise, respectively.

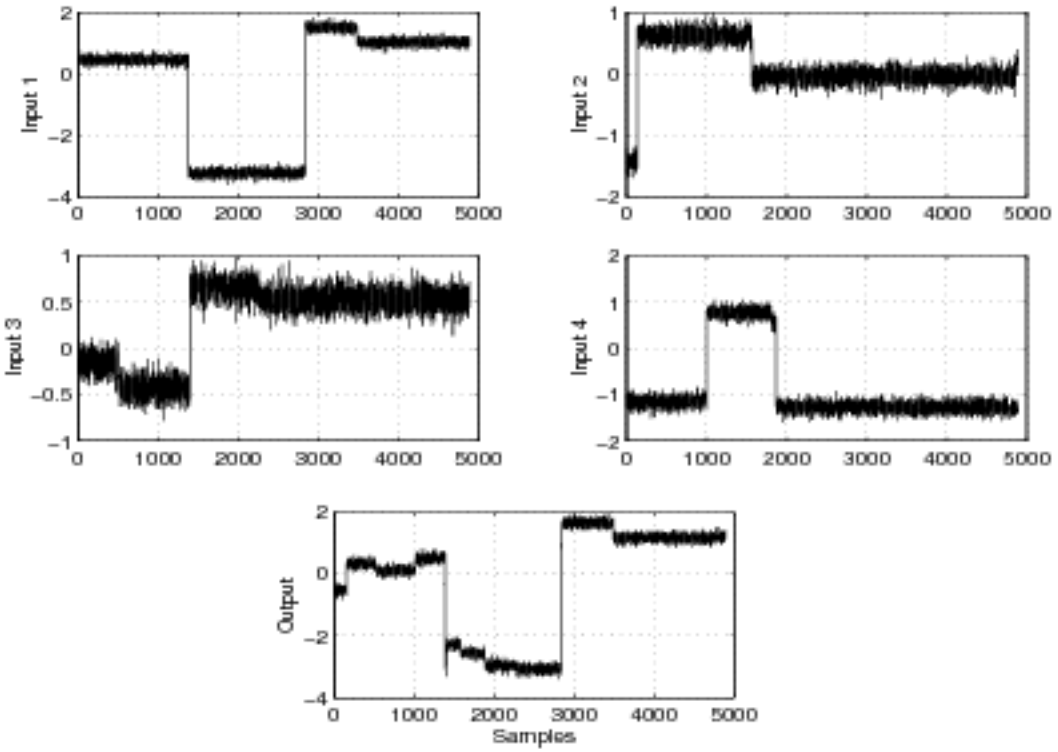


Figure 23: Simulated data set. SNR: output=25 dB, $input_1$=25.5 dB, $input_2$=12 dB, $input_3$=12 dB and $input_4$=18 dB.

As can be seen in table (2) below, RTDE shows better performance than ML and CRA approaches. RTDE makes a good estimation for all inputs, whereas ML and CRA fails in the third input. Moreover, in figure (24) can be seen than the estimation of ML and CRA are much more noisy than RTDE. Therefore, we can conclude that the RTDE method is more confident for MISO systems than classical methods. This is due to the fact that RTDE takes into account the influence of the other inputs.

	$Input_1$	$Input_2$	$Input_3$	$Input_4$
Real Time Delay (samples)	5	20	9	13
ML delay estimation	8	16	0	15
CRA delay estimation	8	24	0	16
RTDE delay estimation	8	22	11	16
ML estimation error (samples)	3	4	9	2
CRA estimation Error (Samples)	3	4	9	3
RTDE estimation error (samples)	3	2	2	3

Table 2: Delay estimation results for ML, CRA, and RTDE

2.4 Time Delay Estimation on Polyamide Data

In this section we apply RTDE to the Polyamide Basf data show in figure (25), this subsystems has 7 inputs, 1 output. Figure (26) shows the RTDE estimated impulse response of the system for each input. As can be seen in figure (25) all the input responses reveal a peak around the zero lag. This peak is neglected, because, it is known in advance that the time delay is bigger than 20 samples. Therefore, the peak is searched after 20 samples in order to make a better estimation. The results are shown in figure (26).

2.5 Conclusions

In this part of the report, we have introduced some correlation analysis based methods often cited in literature as ML (Maximum Likelihood). In addition, a new non-parametric approach based on the impulse response estimation for MISO systems, RTDE stands for *Regualarized Time Delay Estimation*, has been presented. A comparison of performance showed that RTDE is more confident than ML and CRA for MISO systems. For low SNR all the methods have problems to make a good estimation. In order to improve the estimation under low SNR conditions, a regularized term α has been introduced in RTDE, with acceptable results, but more research has to be done in this direction. Moreover, RTDE was applied to a real case, Polyamide data. Prior information was used in order to improve the estimation, and the results obtained were into the range expected.

References

[1] Knapp Charles, 1976, The Generalized Correlation Method for Estimation of Time Delay, IEEE Transactions on Acoustics, Speech, and Signal Processing - August 1976

[2] Nikias Chrysostomos L., Pan Renlong, 1988, Time Delay Estimation in Unknown Gaussian Spatially Correlated Noise, IEEE Transactions on Acoustics, Speech, and Signal Processing - vol 36. No. 11, nov. 1988.

[3] Welch Peter, 1967, The use of Fast Fourier Transform for the Estimation of Power Spectra: A method Based on Time Averaging Over Short, Modified Peridograms, IEEE trans. Audio and Electroacoust., vol. AU-15, pp.70-73, june 1967.

[4] Chandrasekaran et al, 1998, Parameter Estimation in the presence of bounded data uncertainties, SIAM vol. 19, No. 1, pp. 235-252, january 1998.

[5] Kailath T. et al, 2000, Linear Estimation, Upper Saddle River, New Jersey, Prentice Hall.

[6] Matlab Release 12, 2001, Tutorial on Higher-Order Spectral Analysis Toolbox, version 2.0, Mathworks.

[7] Vandewalle J., De Moor B.,Van Overschee P., 1993, Multivariate Systems and Identification, Katholieke Universiteit Leuven - ESAT.

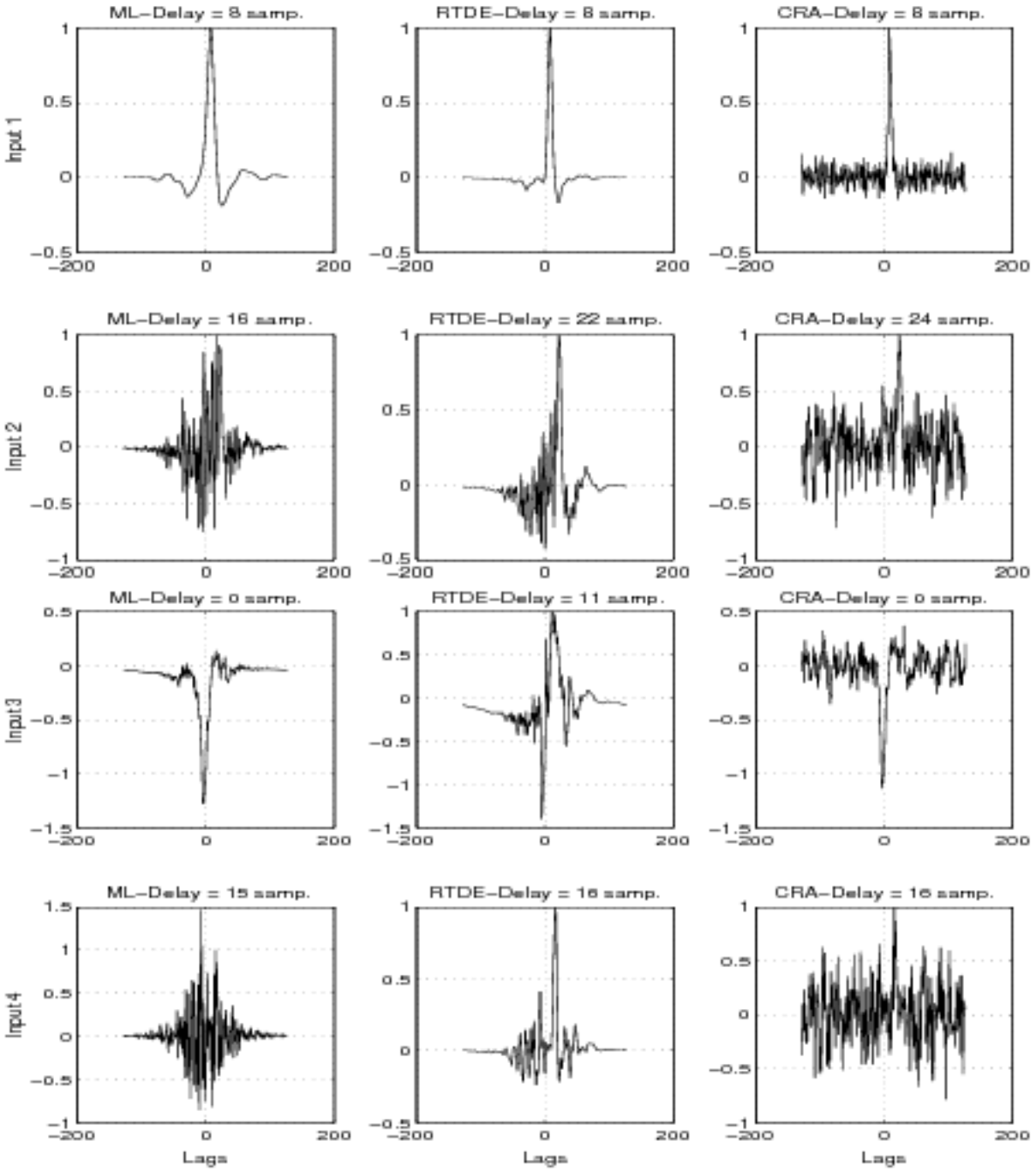


Figure 24: Comparison between ML and RTDE for time delay estimation in a MIMO system.

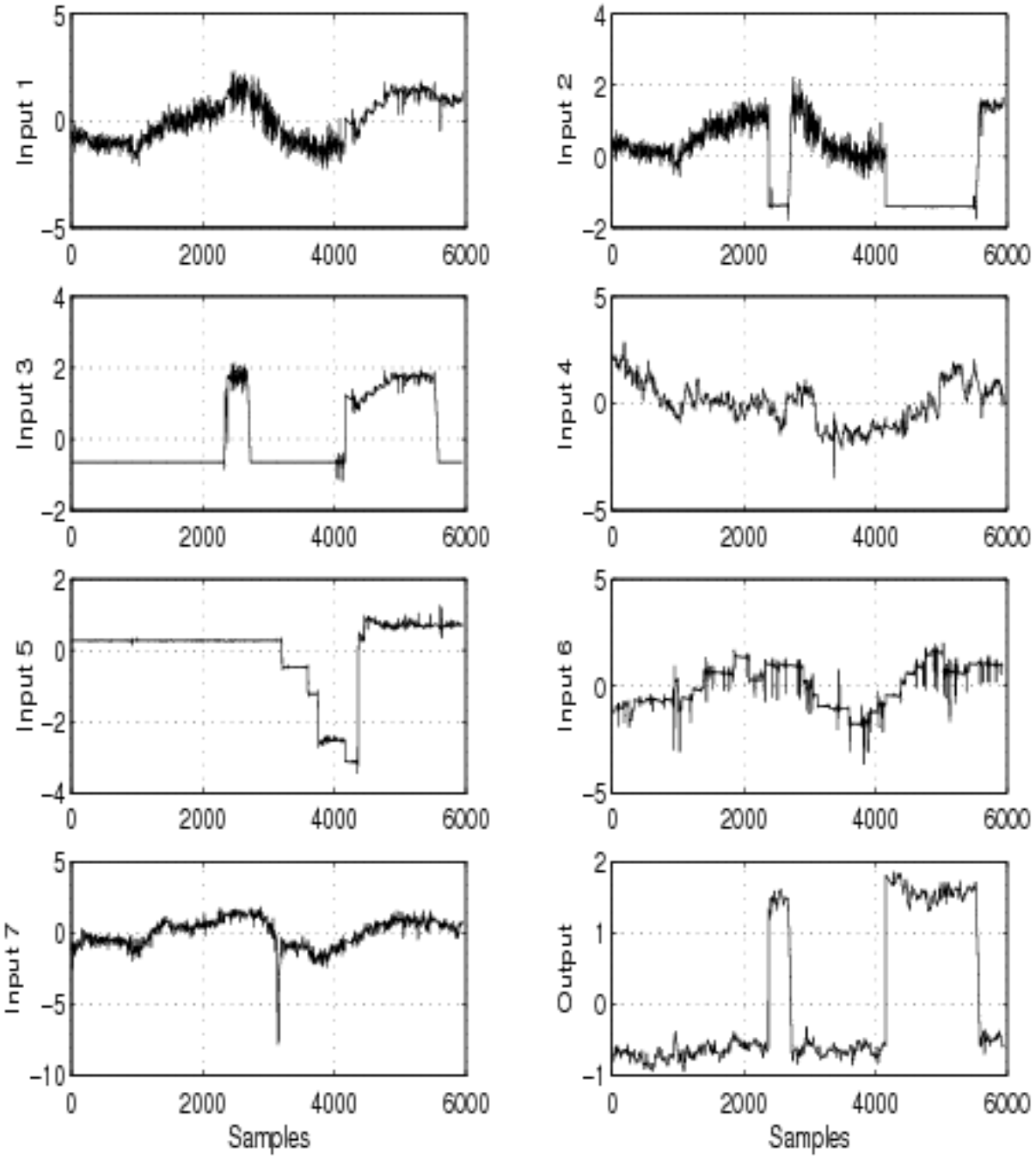


Figure 25: Original Polyamide Input-Output signals.

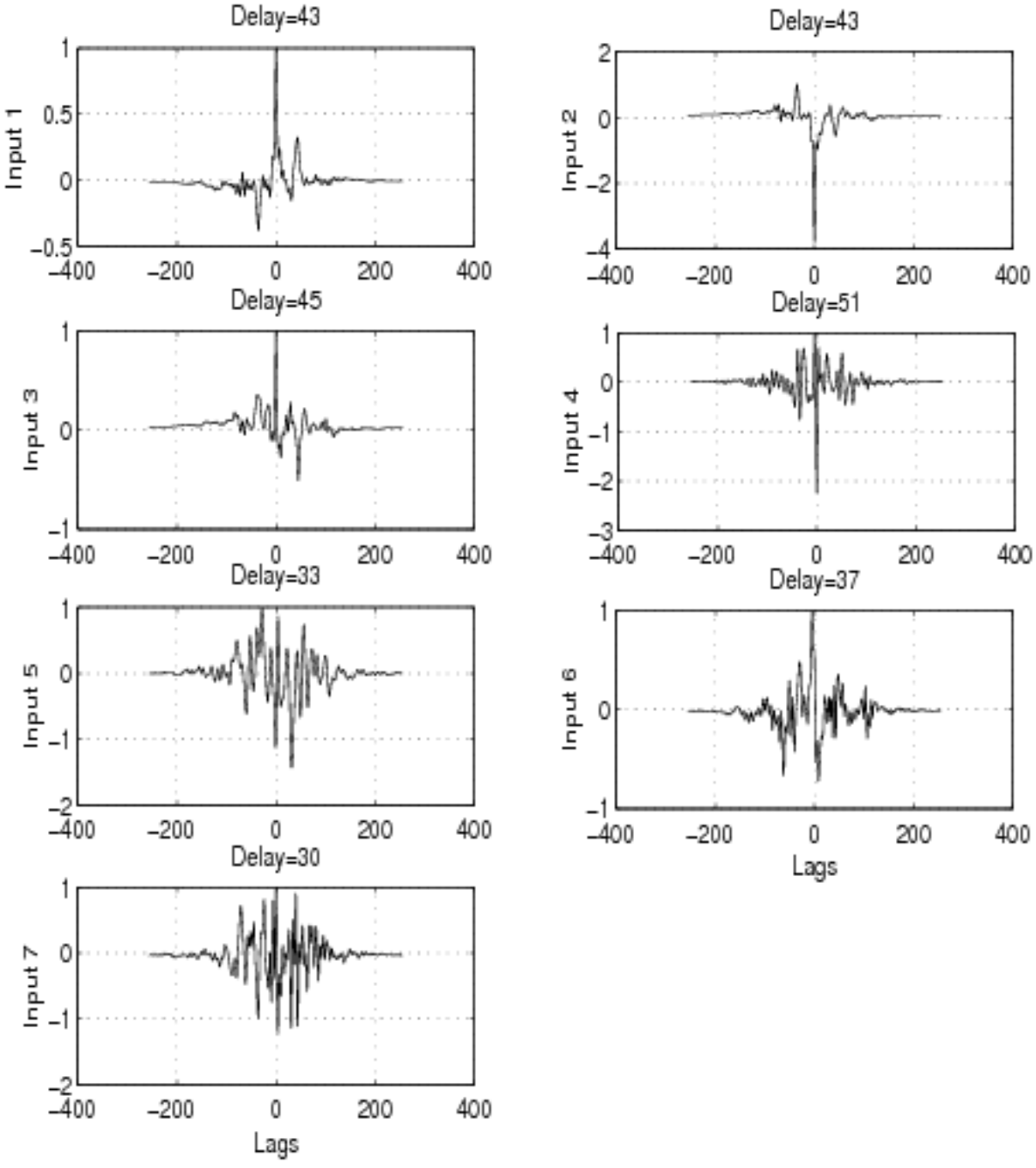


Figure 26: RTDE Estimated Impulse Response - Delay Estimation

3 Irregular to Regular Sampling Problem

3.1 Introduction

The irregular sampling problem is concerned with the problem of recovering a band-limited signal $\mathbf{y[n]}$ with a bandwidth $\mathbf{M}$ from a sequence of samples which may be taken in an irregular way. Therefore a viable strategy consists of resampling at a higher frequency the given irregularly sampled data onto a regular grid, in order to use conventional tools for further analysis [1]. Then filtering the resampled data with a low pass filter, it is possible to recover the original signal $\mathbf{y[n]}$. Resampling always requires some form of interpolation, which permits the construction of an underlying continuous function representing the discrete data.

3.2 Irregular to Regular Sampling Using the Iterative Voronoi Method

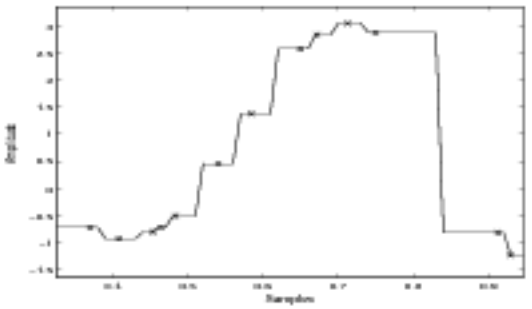


Figure 27: Nearest Neighborhood Interpolation

First of all, we have to resample the signal $y[n]$ at a higher frequency, hence, *Nearest Neighborhood Interpolation* is used. In other words, given the sampling values $y[n]$ at known positions we form a step function at a smaller regular sampling time t_n, which is constant from midpoint to midpoint of the given sampling sequence. In the case where the number of missing points between subsequent sampling positions is odd, the midpoint itself is assigned the arithmetic mean of the two adjacent sampling points. Because of the assumed smoothness of the signal, this step function will not be too far away from the original signal, in the mean-squared sense. Having used the sampling values, it is possible to make a guess about the frequency bandwidth (Ω) of the original signal. The step function (which apparently contains many high frequencies outside of Ω) is filtered, i.e. only those fourier coefficients which

belong to Ω are preserved, the rest is discarded. This operation corresponds to a simple projection operator [1].

Clearly the filtering process destroys the pointwise interpolation property of the step-function, but theoretical considerations show that (given Ω and the sampling rate in terms of the maximal gap) one can give the following guarantee: If the maximal gap is smaller than the Nyquist rate (N/M), then the following iterative scheme will converge to the original signal $y[n]$ [1, pp 37-38] (which will be the better the smaller the actual maximal gap is compared to the Nyquist rate) Indeed, one can show that there is some factor $q < 1$, such that the Ω-filtered version $y_1[n]$ of the nearest neighborhood interpolation satisfies $||y - y_1| < q\ ||y||$. We therefore look at the difference signal $y[n] - y_1[n]$. According to the construction, $y_1[n]$ has its spectrum within the same set Ω, and for obvious reasons we know its coordinates at the given sampling positions. Therefore, the estimate indicated above can be applied and we can again recover a certain portion of the remaining signal by repeating the first step, now starting with the sampled coordinates of $y[n] - y_1[n]$. Technically speaking we recover $100(1-q)$ percent of the missing energy of the difference signal $y[n] - y_1[n]$ at the second step. Continuing to use the difference between the given sampling values of $y[n]$ and those of the $n - th$ approximation we generate additive corrections which lead stepwise to improved approximations. Afterwards, a regular resampling based on the step function sampling time t_n can be done.

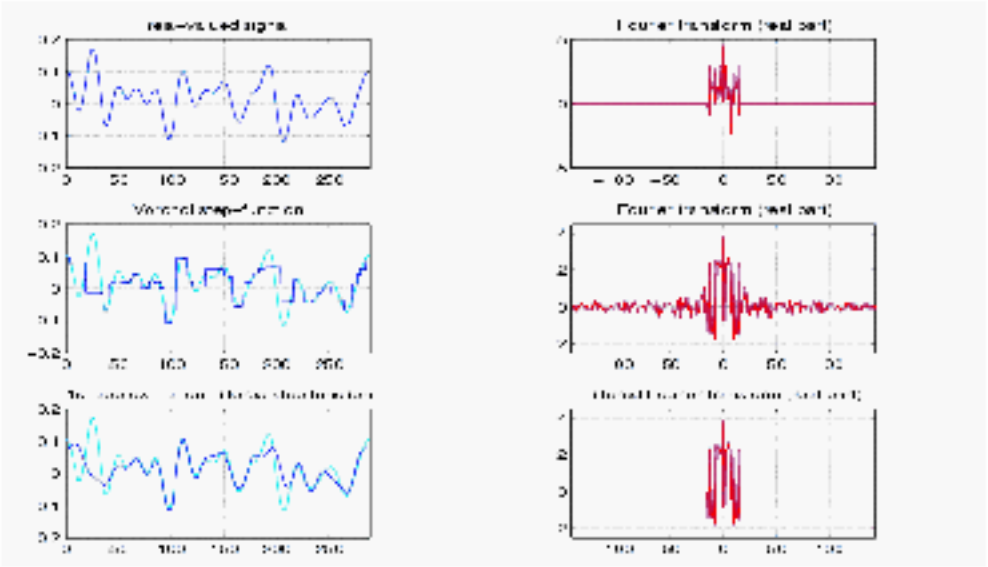


Figure 28: Irregular to regular sampling using the Iterative Voronoi Method - first approximation

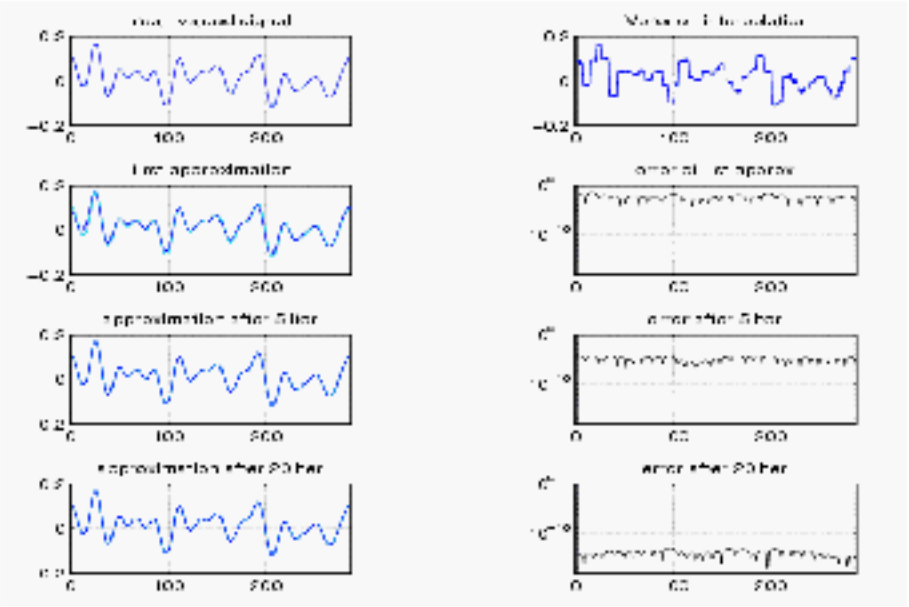


Figure 29: Irregular to regular sampling using the Iterative Voronoi Method

Example: Given the following function

$$y = 2sin(3.4\pi t) - \cos(2\pi t - \pi/6) + 0.5sin(\pi/2t + \pi/3) \tag{79}$$

with its discrete fourier transform

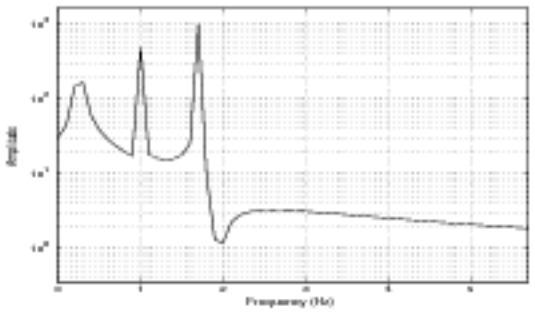


Figure 30: Discrete Fourier Transform of y

where its frequency bandwidth is around 1.9 Hz. It is sample irregularly as shown below in figure (**??**).

Problem

Now taken this irregularly sampled data (Y_n) we want to reconstruct the original signal. Assume we just know the bandwidth of the signal, and it is taken based on the maximal sample time. Let's say

$$Bandwidth = Fcutoff = \frac{1}{2Ts_max} = \frac{1}{2 * 0.2541} = 1.9677\ Hz \tag{80}$$

It is also assumed **Fcutoff > Nyquist frequency**

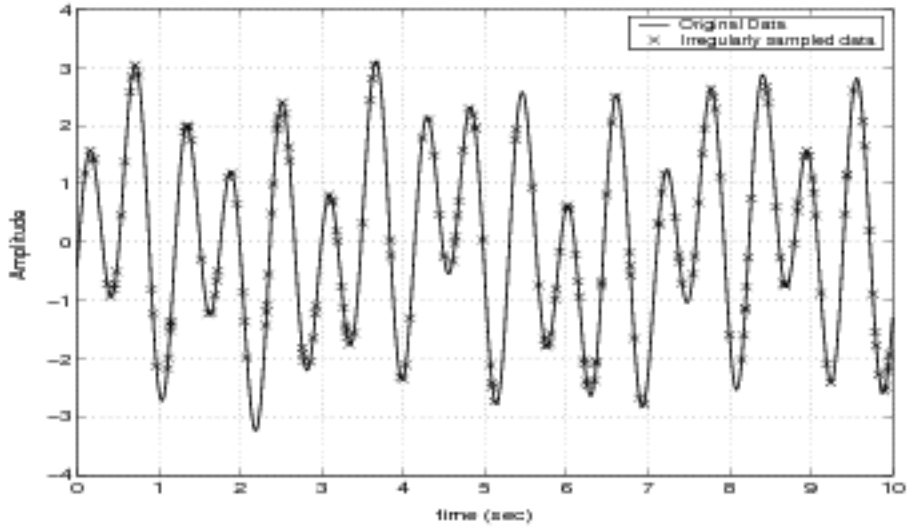


Figure 31: Irregular Sampled Data

In order to reconstruct the original signal we proceed as follows

1. A regular sample time based on the $Fcutoff$ is computed

$$Ts = \frac{1}{10 * Fcutoff} = 0.0508\ sec$$

2. Make a neighborhood interpolation to Yn with a sample time much smaller than Ts

$$Ts_{neighborhood} = \frac{Ts}{20} = 0.0025\ sec$$

3. Filter the signal obtained in 2 with a low-pass filter at the frequency $Fcutoff$

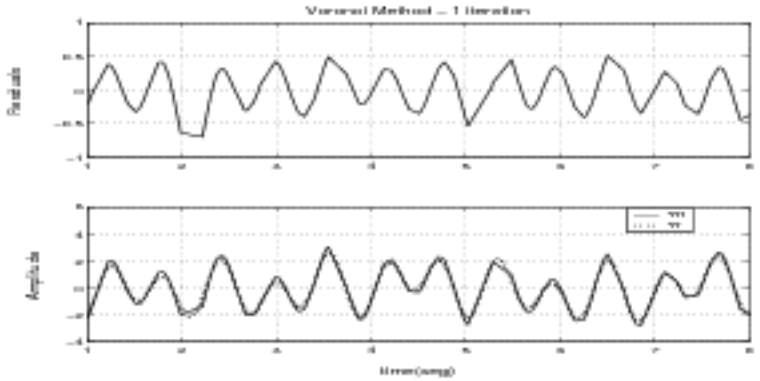
$$Y_f = filter(Low - pass,\ Fcutoff,\ Y_{neighborhood})$$

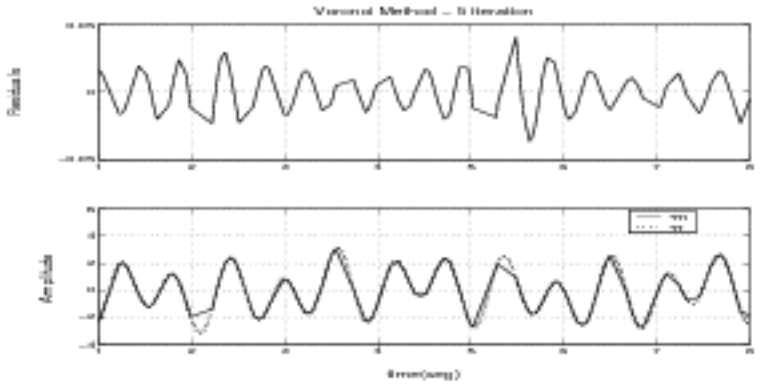
4. Take the filtered signal (Y_f) and compute the residuals (Y_R) respect to the irregularly sampled data (Y_n). Notice that a linear interpolation to Y_f has to be done in order to find the values at the same sampling times of the given data (Y_n).

5. Keep Y_f and replace Y_n by Y_R. Go back to 2 and repeat the procedure, then add the new Y_f to the old Y_f . Repeat until the number of iterations is reached.

6. Afterwards you end up with a very large data set because $Ts_{neighborhood}$ is very small. To correct this, Y_f is resampled at $T_s = 20 * Ts_{neighborhood}$ without corrupting the signal. It is possible because it has been sampled at high frequency (neighborhood interpolation) and as known that the bandwidth of the signal is located at a lower frequency $Fcutoff$.

.





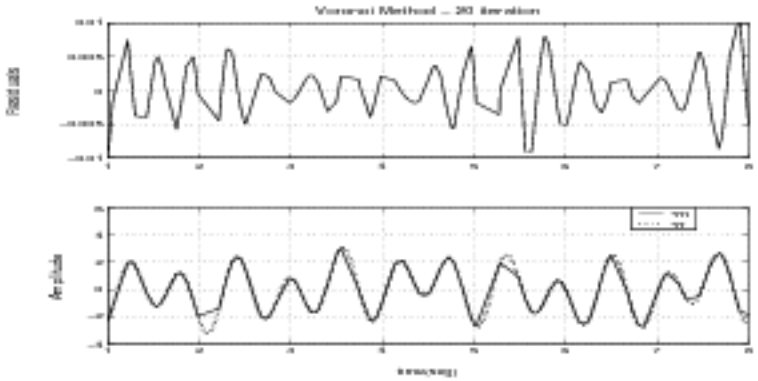


Figure 32: Iterations of Voronoi Method

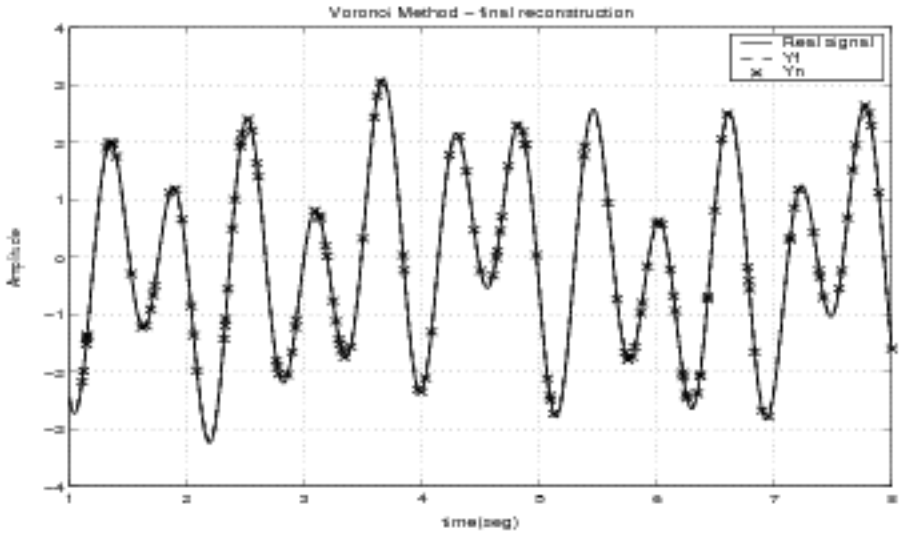


Figure 33: Final Reconstruction after applying the Iterative Voronoi method

Since one can see, the reconstruction in figure (**??**) is very good. After 20 iteration and a computing time of 10.5 seconds we have got a $RMSE = 0.048$, with a regular sampling time $Ts = 0.0508$ *sec.* Notice that in order to obtain good results is very important to know approximately the frequency bandwidth of the signal.

Now we can check what happens if a wrong frequency bandwidth is used: In the first case a cut-off frequency of 1 Hz was chosen, the result is very bad, poor fitting ($RMSE = 1.2508$) and reconstruction (fig. (**??**)).

For the second case a cut-off frequency of 10 Hz was chosen. Good for fit-

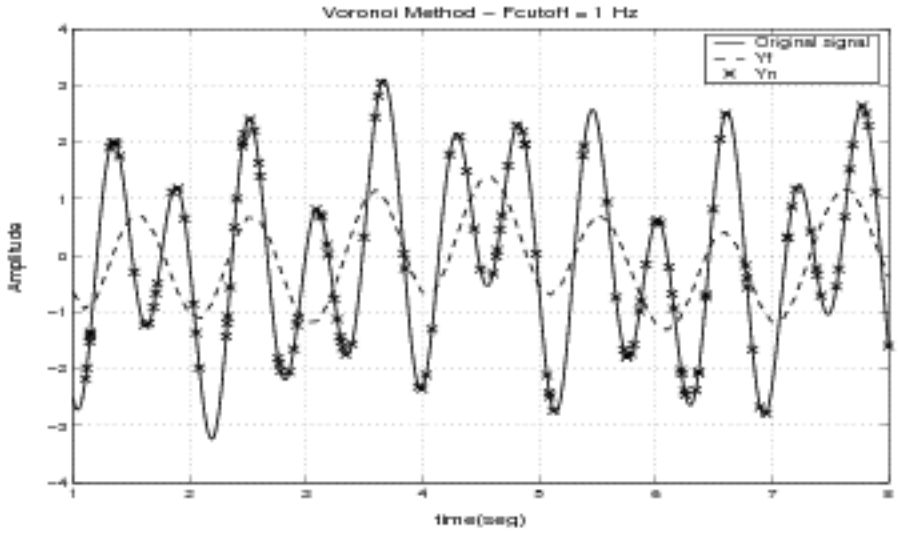


Figure 34: Reconstruction with a lower frequency bandwidth. RMSE=1.2508, Ts=0.1sec

ting the points of Y_n ($RMSE = 0.0038$), but very bad in signal reconstruction (fig. (??)).

We can say that the method is very reliable if we have some information about the frequency bandwidth of the signal. For this example was found that by chosen a bandwidth between 1.5 to 3.0 Hz one obtains good results, it means that even having some error in the estimation of the frequency bandwidth the method has a good signal reconstruction.

3.3 Signal Reconstruction Using the Iterative Voronoi Method

When there are identified missing points in a regularly or irregularly sampled signal, one can assume that such points were never taken, therefore if the signal is into a regular grid, it becomes an irregularly sampled signal, and if the signal is into an irregular grid, it just get more irregular sampling time intervals. Afterwards, it is possible to apply the procedure seen above to the new data set in order to rebuild the complete signal, as shown in figure (??).

The original signal has 492 points, it has some missing points and it is irregularly sampled. Applying the Voronoi method with cutoff frequency 0.2521 *times/day* (base time in days) we got a reconstructed signal with 1800 points, no missing points, and regularly sampled. The computing time was 13.7 seconds, and $RMSE = 0.6077$.

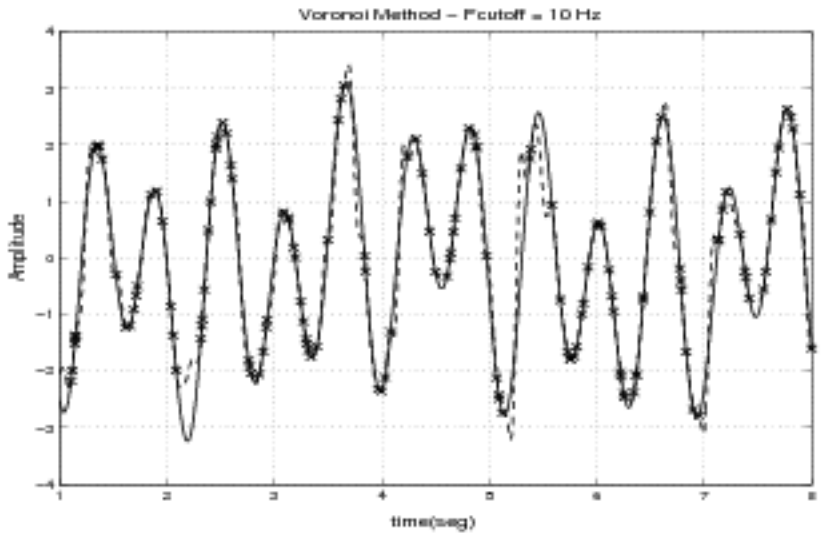


Figure 35: Reconstruction with a higher frequency bandwidth. RMSE=0.0038, Ts=0.01sec

The computations were done in a PC - Pentium III, 800 MHz, 128MB RAM, platform Linux i686.

3.4 Conclusions

In order to solve the problem of how to resample an irregular sampled data into a regular grid, the Iterative Voronoi method has been used with very good results. It is important to note that a good approximation of the frequency bandwidth of the signal has to be done to get a reliable signal reconstruction. On the other hand, this method is also useful for signal reconstruction when the data has missing points, because one can take those point off and treat the rest of data as an irregular sampled data, then by applying the Voronoi method is possible to reconstruct the signal.

References

[1] Strohmer T., 1991, Irregular Sampling, Frames and Pseudo-inverse, Master's thesis, Department of Mathematics, University of Vienna, Austria.

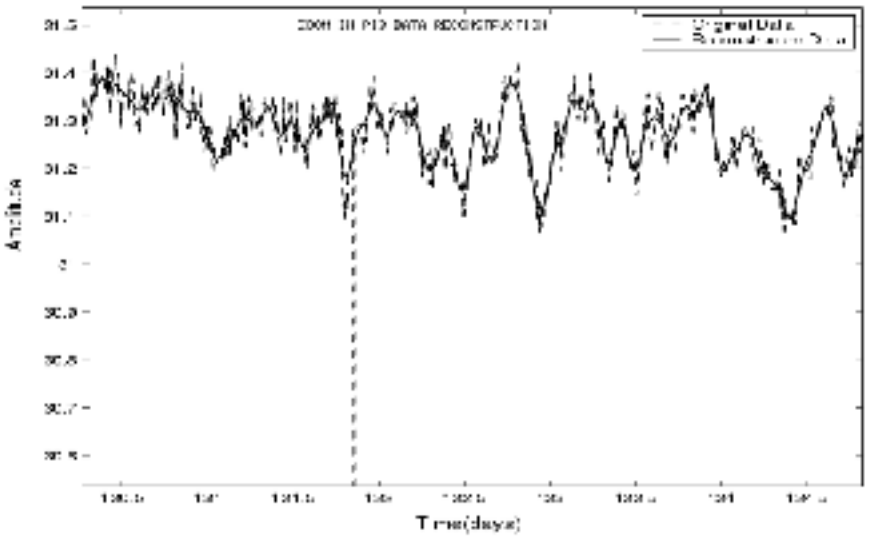


Figure 36: Reconstruction to a regular sampling time