Sistemi subacquei final course project

Outlier handling techniques in acoustic signals for marine applications

Ing. Robotica e dell'automazione

Fabio D'Onofrio n°556505
f.donofrio6@studenti.unipi.it

Emilio Maoddi n°555566
e.maoddi@studenti.unipi.it

Michael Mugnai n°556448
m.mugnai6@studenti.unipi.it

20 September 2018

Final version

Contents

List of Figures

List of Tables

Abstract

This report analyses three different strategies for outlier management with particular reference to underwater acoustic signals in **USBL** (*ultra short base line*) systems. Two unsupervised robust outlier handling methods are described: *least entropy like* (**LEL**) and *Norm-1 minimization*, and a supervised outlier rejection method based on *Quantile regression*, including notes about their software implementation in a `Matlab` environment.

1 Project Statement

To identify three suitable methods among different outlier management techniques focusing on a given USBL system signal as a reference. To choose the first technique among two proposed articles[1][2] and to implement the selected methods as `Matlab` applications.

2 Milestones

- Analysing a survey article about state of the art methods for outlier management;
- Analysing and choosing a technique among the two proposed articles[1][2] ;
- Replicate the results described in the selected article;
- Choose two further techniques for outlier handling or removal;
- Analyse and compare results among those methods;
- Implement the selected methods as Matlab applications.

3 Deliverables

At the completition of this project two Matlab applications are released:

- An `AppMain.mlapp` application featuring two unsupervised outlier robust filtering methods(*Norm-1 minimization* and *Least Entropy Like*) supported by a tool for supervised signal division, to provide of the necessary preprocessing before filtering for outliers;
- A `QRODApp.mlapp` application featuring a *Quantile regression* based method for supervised outlier detection.

- This report, featuring the description of the respective techniques employed in the applications and further details on performance and implementation.

4 Introduction

4.1 Underwater acoustic signals

Wireless communication systems in underwater environments relies almost exclusively on propagation of sound in water. A sound wave propagating underwater can be considered the product of a pressure wave in a fluid influenced by a series of factors such as sound velocity, temperature, salinity, depth, seabed and water surface reflection and many other characteristic phenomena.

4.2 Baseline systems

A common requirement for ROVs, AUVs and other underwater systems is navigation and tracking. Acoustic waves propagation in water occurs at a precisely measurable and estimable rate, allowing for frequency modulation techniques. **Base line systems** (**BL**) refer to a broad category of underwater positioning systems featuring the following characteristics:

- *Localization*: achieved measuiring the range from a set of different transponders in known positions;
- *Range*: computed by time-of-arrival measurements of acoustic signals.

Speed of sound knowledge is needed, ray path corrections may be needed (depending on the distance that rays will travel and the variation of the speed of sound in the medium). The distance between two transponders is known as the "**base line**".

A distinction between BL systems can be made in respect to base line magnitude, which affects greatly the system performances and characteristics (as shown in table 1).

Type	Base Line	Frequency	Range	Accuracy (% of acoustic path)
LBL	$[500 - 5000]m$	$[10 - 40]KHz$	$[10 - 40]Km$	$[0.1 - 2]\%$
SBL	$[20 - 100]m$	$[50 - 100]KHz$	$1Km$	$[0.1 - 2]\%$
USBL	$< 0.1m$	$[100 - 250]KHz$	$500m$	$[2 - 3]\%$

Table 1: Base Line systems typical performances

Note that the accuracy is expressed in therms of percentage of the wavelength, thus to higher frequencies corresponds better accuracy, but lesser is the range of operation.

LBL (*Long Base Line*) Operates by simultaneously interrogating transponders (*spherical positioning*, active receiver for timer synchronicity) or collecting beacon signals transmitted with fixed delays (*hyperbolic positioning*, passive receiver). Typical installation of an LBL system consists in sea bottom moored hydrophones (Figure 1, top left).

SBL (*Short Base Line*) Operates like LBL, but typically transponders are installed on floating surfaces or big ships (Figure 1, top right).

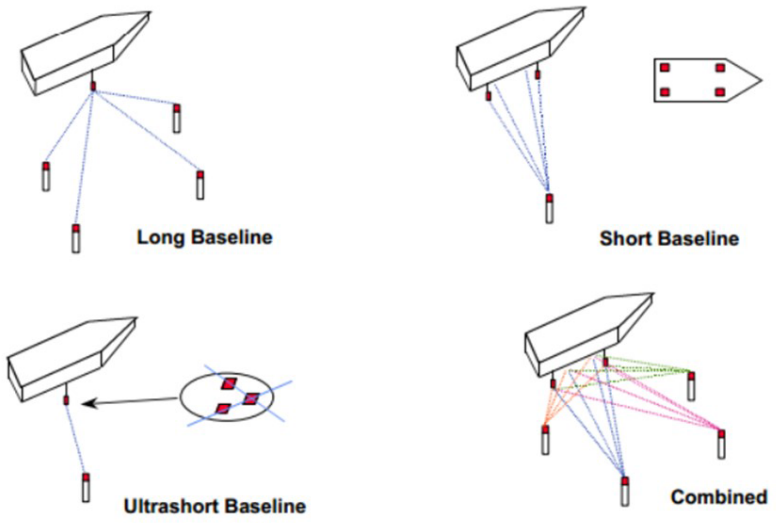

Figure 1: Base Line systems

USBL (*Ultra Short Base Line*) Operates interrogating a transponder installed on the vehicle measuring relative range and angle, as shown in bottom left of Figure 1.

USBL sytems offer the advantage of not requiring sea bottom moored transponder array. The disvantage is that positioning accuracy and robustness is not as good, as fixed angles translates into a larger position error at greater distances. USBL suffers more than LBL for underwater acoustic non-uniformities, due to the higher frequencies employed. Figure 2 summarizes the typical steps of action of a USBL system.

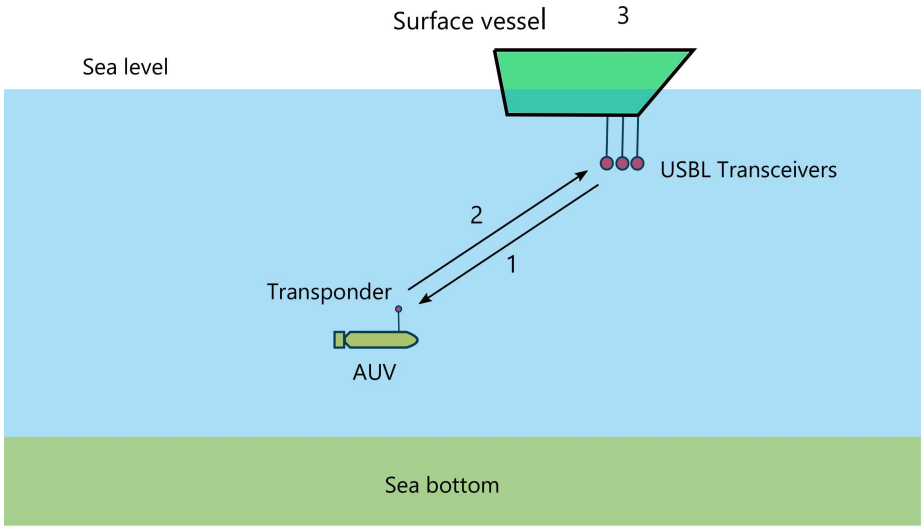

Figure 2: USBL steps of action: 1) An acoustic pulse is transmitted by the transceiver and detected by the transponder. 2) The transponder replies with his distinct acustic pulse. 3) The time interval between transmission and reply detection (time of flight) is measured and converted into a range. To compute the subsea position, angle of the incoming reply is measured via **phase differencing** within the acoustic transducers array making up the transceiver unit.

4.3 Outliers

Formally defined by Barnett and Lewis in 1994, an outlier in a set of data is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.[3]

4.3.1 Outliers affecting USBL measurements

An acoustic signal transmitted underwater, will travel in every direction and reflect off any surface it finds in its way. The end result at the receiver is not a single signal travelling directly from the source, but every reflection that finds its way to the place where the receiver is located.

Underwater propagation suffers **multipath effects** mainly due to **reflection** on water surface and sea bottom, and **ray bending** due to sound's velocity variations caused by to the anisotropy of the medium (*Snell's law*).

In USBL systems, multipath can determine an incorrect time-of-flight estimation.

4.3.2 Outliers management methods classifications

Approaches to the outlier detection problem can be divided into 3 fundamental types: **unsupervised**, **supervised** and **semi-supervised** methods.[3]

Unsupervised methods feature processing of data as a static distribution, without further hypotesis on the distribution type, pinpointing the most remote points and flagging them as potential outliers (ultimately performing a clustering of data). The approach is predominately deployed in post processing analysis, thus it requires that all data will be available at start. Two main techniques are commonly used: *diagnosis* and *accommodation*.

- A *diagnostic* approach iteratively clusters and purge the data of detected outliers.
- An *accommodating* approach incorporates the outliers into the data and employs a robust classification method.

Essentially, *diagnosis* prune out all the suspected outliers, allowing to utilize non-robust methods (e.g. Least Square Minimization), computationally cheap, that wouldn't be permitted in presence of

outliers, while *accommodation* is a robust approach, more expensive in terms of computation but capable to process signals heavily contaminated by outliers without being skewed.

Supervised methods require preprocessing of the data, labelling samples as *normal* or *abnormal* in advance, the abnormal class representing the outliers. Classification is best suited for static data, but on-line processing can still be implemented by adaptive algorithms featuring classification model learning.

The limits of on-line processing is given by the algorithm generalization capability: as the system acquires new exemplars, they may be classified incorrectly since classification is limited to the currently known distribution regions, until a significant number of samples from a new region occurs.

Semi-supervised methods similarly to supervised methods, they need pre-labelled data, but only for the *normal* class. This is easier to achieve in practice, since obtaining outliers may be expensive (e.g. in fault detection you should tamper the monitored machinery only to collect outlying data).

They are suitable for both offline and online data processing, as classification criteria is updated incrementally, tuning the model to improve the fit at each new available data. A new sample is recognized as normal if it lies within a certain defined boundary. Classification can be *hard bounded* or *soft bounded* whether a sample falling outside is sharply considered an outlier, or only in a certain ammount, through a membership degree. If normality shifts, system boundary may be shifted by re-learning the data model.

In conclusion when selecting a method for outlier detection two fundamental considerations have to be made:

- Algorithm should accurately model data distribution, and should be scalable to the data to be processed;
- Selection of suitable neighbour of interest for an outlier is a non-trivial task that may require supervision. Many algorithms autonomously induce a threshold by computing normality while processing the data, however these approaches are often parametric and enforce a previously defined distribution model. Other techniques require user defined parameters to define the size or density of neighbours for outlier thresholding.

A further classification of outlier detection methods can be made taking into account the particular field they originated from: **statistical**, **neural network based** or **machine learning based**.

Statistical methods are generally suited to quantitative real-valued data sets. Statistical approaches include *proximity-based* techniques, *parametric*, *non-parametric* and *semi-parametric* methods. The downside is that they usually need a prior knowledge of the data distribution, and suffer the *curse of dimensionality*, where at the increase of space dimension, the computational cost exponentially increase.

Neural networks provide for a good classification performance if properly trained, capable of learning complex class boundaries. Neural methods can be divided into *Supervised* and *Unsupervised* types, where for the first one a pre-labeled dataset is needed. Usually there's the need of ordinal data in order to properly define vector distances, used in the training process. Curse of dimensionality can be prevented executing a feature selection on the dataset (reducing space dimension to the more significative components).

Machine learning extends data set handled types to categorical data (non-numerical datasets) implementing decision trees that don't need any prior knowledge of the data parameters and distribution. Does not suffer of curse of dimensionality, but are dependent on the coverage of the training data and can suffer overfitting, if the dataset doesn't spread well in the entire space.

4.4 Reference signals

A set of telemetric data is given to provide for a real case scenario of underwater sensoring contaminated by outliers. Telemetry consists in three signals of a vehicle's bearing angle acquired by a USBL system (Figure 3).

The outliers affecting data present high variability in density and value throughout the signal. Regression problem can be tracked back to linear regression adopting a standard linear-in-the-parameter model $y = M\theta + \varepsilon$, dividing the signal into linear traits.

Given these particular characteristics, we are interested in the following features for our regression algorithm:

- Offline data processing
- Unsupervised or Semi-supervised

We are both interested in accomodating methods and diagnostic methods, to perform both robust recostruction of the signals or recognition and extraction of outliers from the processed data.

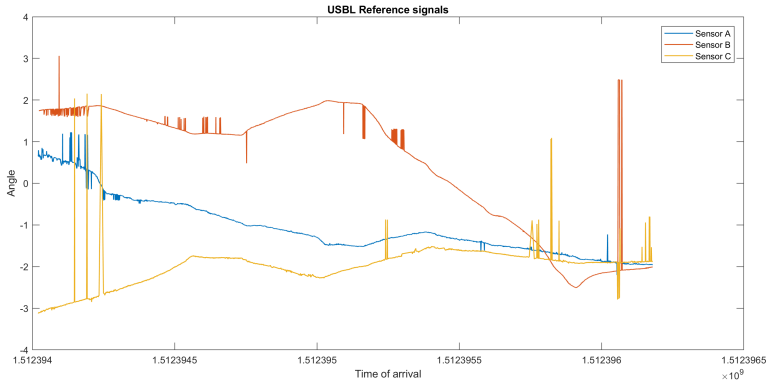

Figure 3: Reference signals

4.5 Comparison between a Kalman filter and entropy minimization based methods

Two articles addressing solutions for outliers in acoustic navigation where given to analyse and provide perspective on how to practically approach the problem.

In the first[1] a solution based on integration of **dead-reckoning** and **range measurement** by the means of a **Kalman filter** is proposed.

The second article[2] proposes a **LEL** (Least Entropy Like) technique, consisting in tracking back the regression problem (when dealing with outlier contamination) to an estimation error optimization problem, by minimizing a cost function related to the system's entropy, proven to be robust against outliers.

4.5.1 Kalman filter based method

The particular application case of this technique refers to an Odyssey II AUV, equipped with a custom LBL navigation system. The proposed algorithm is then applied to process three different datasets obtained during different missions of the AUV.

The AUV computes navigation by *filtering*, in opposition to *fixed* navigation: the vehicle dead reckons its position until LBL data (or position "fix") is available; once the three beacons return the range measurement, solution is computed as the intersection of three spheres centred at the beacon locations with radii corresponding to the measure ranges. Instead of resetting the vehicle position (fixed navigation), the new information is incorporated to the previous

dead reckoning measurements by a weighted combination (*Kalman filter*).

Outlier rejection is performed in *time domain*: a properly tuned Kalman filter discards time of flight falling outside normal distribution. This technique requires a non trivial calibration process, as a-priori initial position estimate is needed in order to predict correctly the expected arrival times to each beacon. This first set of calibration measurements is critical, as may induce in large position errors that result in rejection of fixing signals during navigation.

A more detailed description about calibration techniques can be found in the article.[1]

4.5.2 Entropy minimization method

The method is illustrated[2] in two different applications: simulated data and publicly available third-party data. This technique is meant for off-line processing.

Considering an **LPM** (Linear in the Parameter Model), instead of computing a standard linear regression by the means of **LS** (Least Squares) method, unreliable in the presence of outliers, a **LEL** (Least Entropy Like) method is introduced featuring a cost function inspired by Gibbs entropy. This new cost function will be minimized in place of the the square of residuals, allowing a robust approach in presence of outliers, giving less importance to those residuals that are few in number but high in magnitude (unlike LS minimization that does exactly the opposite). This will be extensively illustrated in Section 5.

Due to the non linear nature of the cost function local minima may be found, requiring a proper initialization of LPM parameters.

Furtherly a **Quasi-Newton** method of minimization is proposed, in order to avoid significantly expensive computation of the Hessian required by standard Newton's methods.

There is no need of a-priori information on the scale of the noise and of outliers.

4.5.3 Conclusive choice

Table 2 express the characteristics of the two presented methods regarding the features we where looking for.

	Intended use	Accommodating	Unsupervised or Semi-supervised
Kalman filtering	online processing	Supported	Unsupervised
LEL	offline processing	Supported	Unsupervised

Table 2: Kalman filtering and LEL features of interest availability.

Given the complex calibration procedure described in the first article[1] and the specificity of its application case, in contrast with the versatility and relative simplicity of entropy minimization algorithm described in the second article,[2] LEL algorithm is chosen to be implemented in our `Matlab` application.

5 Least Entropy Like method

Let Y be a real-valued response variable and X a covariate or predictor variable, possibly high-dimensional. A standard goal of statistical analysis is to infer, in some way, the relationship between Y and X.

Standard regression analysis tries to come up with an estimate $\hat{\mu}(x)$ of the conditional mean $E(Y|X = x)$ of the response variable Y, given $X = x$.

The conditional mean is choosen to minimize a certain cost function, that in the well-known problem of *Least Sqare Error* (**LS**) it's the expected squared error loss $Q = \sum_{i=1}^{N} r_i^2$, where $r_i = y_i - \hat{\mu}_i$ are the residuals between measured and predicted variable. However, when working with data affected by some noise that doesn't have zero mean (like outliers are), weighting each residual as its square will leed to a heavy influence on the bias of the predictions, skewing them toward spurious data.

Here is where cames the need to find a different cost function, that should focus on producing a majority of residuals little in magnitude, togheter with a minority carrying most of the fitting error.

5.1 Gibbs entropy like cost function

As the Gibbs entropy suggests, the core idea is to weight each residual multiplyed by its logarithm:

$$H = \begin{cases} 0 & Q = 0 \\ -\frac{1}{\log N} \sum_{i=1}^{N} \frac{r_i^2}{Q} \log \frac{r_i^2}{Q} & Q \neq 0 \end{cases} \tag{1}$$

Where Q is the squared error loss. The component $1/\log N$ is just a rescale, that leeds $Im\ H = [0, 1]$.

Note that r_i^2/Q are *relative residuals*, that vicolate the cost function domain in $[0, 1]^N$: this is the key concept of the method, that allows function minimum either for $r_i = 0$ or for high r_i (since as the relative residual increases it approaches 1, therefore its logarithm approaches zero).

In this way, searching for a minimum of H means to find where the majority of residuals are next to zero, togheter with a minority that contains most of the fitting error, and thus that represents the significant percentage of Q.

5.2 Minimizing Gibbs entropy like cost function

Due to the non-linear nature of the just introduced cost function the search of a minimum is non-trivial, since the Equation 1 contains multiple local minima, as shown in Figure 4.

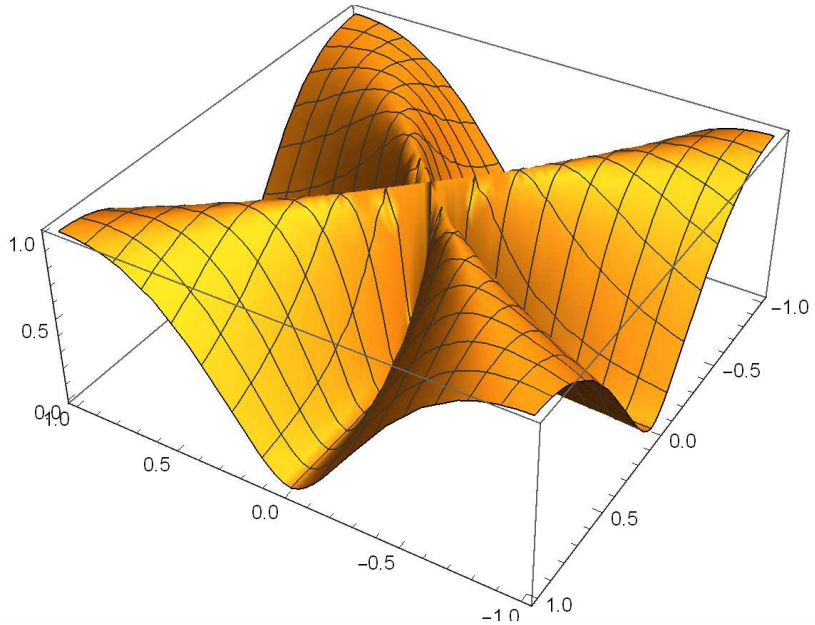

Figure 4: A 2-space representation of Gibbs entropy inspired cost function.

Numerical searches for unconstrained problems, efficient even with non-linear function, are thus needed.

5.2.1 Newton's method

In optimization problems, Newton's method is an iterative method for finding the extremas of a twice-differentiable function f, i.e. roots to the equation $f'(x) = 0$, also known as the stationary points of f. Newton's method assumes that the function can be locally approximated as quadratic in the region around the extrema, and uses the first and second derivatives to find the stationary point.

From an initial guess x_0, Newton's method attempts to construct a sequence x_k that converges towards some value x^* that satisfy $f'(x^*) = 0$, through the help of *Hessian* $H(x_k)$ and *gradient* $\nabla_x f(x_k)$ of f:

$$x_{k+1} = x_k - H(x_k)^{-1} \nabla_x f(x_k) \tag{2}$$

If no closed form solution is given, the evaluation of the Hessian for non-quadratic functions may be costly, since in the most generic case, in a N-dimensional space, N^2 evaluations are needed at each step k.

5.2.2 Quasi-Newton methods

Instead of directly compute $H(x_k)$, its value is approximated by analyzing successive gradient vectors $\nabla_x f(x_k)$. Furthermore, there's no need to invert the Hessian, as Newton's method does, since an estimate of $H(x_k)^{-1}$ can be generated directly.

The main hypotesis exploited for the approximation of $H(x_k)$ is its simmetry, that lets reduce the number of searched variables; many methods suppose that H is positive (semi)-definite too.

The Hessian approximation must be choosen to satisfy:

$$\nabla_x f(x_k + \Delta x) = \nabla_x f(x_k) + H(x_k)\Delta x \tag{3}$$

called *secant equation*, born by the Taylor series of the gradient. Its name derives from the fact that, in one dimension, solving it and applying the Newton's step with the updated value is equivalent to the secant method.

In multi-dimensional problems, the secant equation is underdetermined, not specifying a unique solution, and quasi-Newton methods differ in which constrains are imposed to solve it.

Newton method and Quasi-Newton methods are not guaranteed to converge unless the function has a quadratic Taylor expansion near the searched optimum.

5.2.3 BFGS method

Broyden-Fletcher-Goldfarb-Shanno algorithm is an iterative method for solving unconstrained non-linear optimization problems. BFGS has proven to have good performance even for non-smooth optimizations.

The quasi-Newton condition imposed on the update of $H(x_k)$ is:

$$H(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{4}$$

BFGS algorithm is implemented in Matlab by Dirk-Jan Kroon.[4]

5.3 Implementation

The LEL parameter estimation approach described in Section 5 is first applied to reproduce results obtained by Indiveri in his article,[2] then used on reference signals exposed in Section 4.4.

Splitting each signal in piecewise linear traits (either manually or through the automatic tool explained in Appendix A), the problem of finding the actual values from spurious data is tracked back to the identification of those linear traits: given m subdivisions of the dataset (supposed two-dimensional, formely (x, y), with n samples), we have m LPMs, in which we look for the $2m$ unknown parameters θ_{ki}:

$$\hat{y}_j = \theta_{1i} x_j + \theta_{2i} \qquad x_j \in [x_{n_i}, x_{n_{i+1}}] \tag{5}$$

with $j = 1, \ldots, n$ that spaces over the dataset and $i = 1, \ldots, m$ over the subdivisions.

The problem can assume a matrix form by establishing $\boldsymbol{\theta}$ and $\hat{\boldsymbol{y}}$ as vectors containg their relative variables with subscripts, and developing a matrix M, with dimensions $n \times 2m$, as follows:

$$M = \begin{bmatrix} x_1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ x_2 & 1 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & & & & & & \\ x_{n_1} & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & x_{n_1+1} & 1 & \ldots & 0 & 0 \\ \vdots & & & & & & \\ 0 & 0 & 0 & 0 & \ldots & x_{n_m} & 1 \end{bmatrix} \tag{6}$$

The element to be evaluated is the parameters vector $\boldsymbol{\theta}$ from the LPM $\hat{\boldsymbol{y}} = M\boldsymbol{\theta}$.

BFGS Quasi-Newton method is then applied twice, starting from two different initial conditions: θ_{LS} and θ_{N1}, respectively parameters vector obtained by Least Square and Norm-1 minimizations, as shown in Figure **??** The computation resulting in the lesser cost function value is then chosen as the solution for the Least Entropy Like method.

Given the particular shape of the cost function, as previously shown in Figure 4, the minimum found is closely related to the initialization parameters, so the purpose of employing θ_{LS} and θ_{N1} is to provide for a better initial guess compared to a random point, in the assumption that LS and N1 give a suboptimal solution close

to the LEL one. This way BFGS performances are improved in terms iteration steps.

5.4 Results

Figure 5 shows the result of LEL robust fitting method for the signal generated as reported in article;[2] Figure 6 shows results for the given USBL reference signals 4.4.

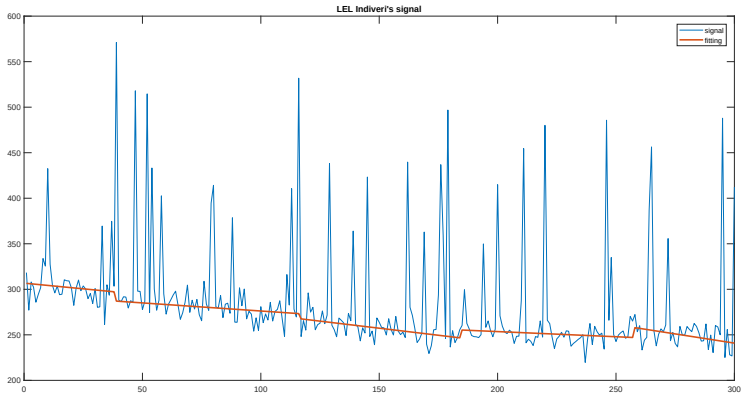

Figure 5: Generated signal and LEL fitting result

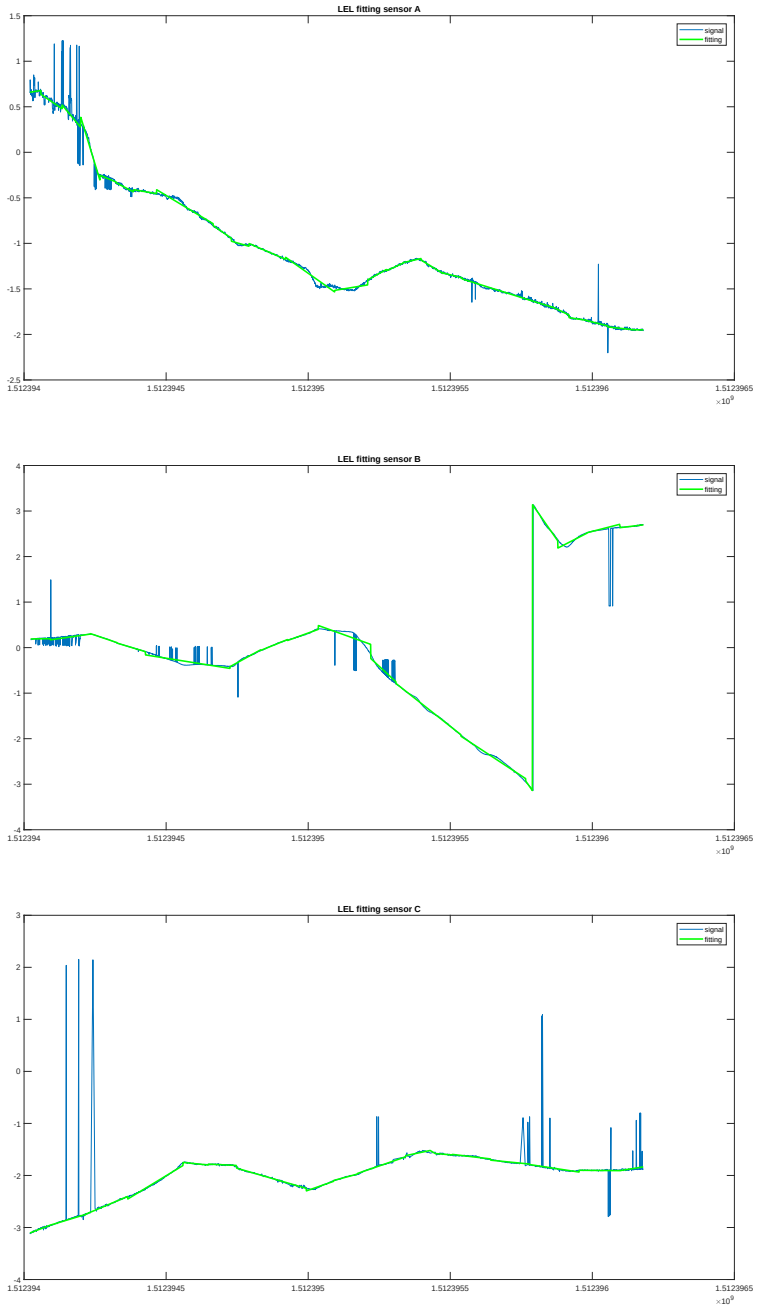

Figure 6: Signal A, B and C and LEL fitting results

6 Norm-1 minimization method

Given a signal $\boldsymbol{y}$ and a *Linear in the Parameter Model* $\hat{\boldsymbol{y}} = M\boldsymbol{\theta}$ for its estime (in which we seek for the regression vector $\boldsymbol{\theta}$), by applying the pseudo-inverse of M to $\hat{\boldsymbol{y}}$ we are implicitly solving the problem $\min_\theta \|\boldsymbol{y} - M\boldsymbol{\theta}\|_2$.

By squaring the residuals, we are giving more weight to the larger ones, and as previously saw in Section 5, this is not the right thing to do in presence of outliers: a better approach could be to give equal emphasis to all observations. From this precepts borns the criterion of *Least Absolute Deviations* (**LAD**), where the problem to solve is:

$$\min_\theta \sum_{i=1}^{n} |\boldsymbol{y} - M\boldsymbol{\theta}| = \min_\theta \|\boldsymbol{y} - M\boldsymbol{\theta}\|_1 \tag{7}$$

and thus also called *Norm-1 minimization method* (**N1**).

Contrary to LS minimization, N1 can have multiple solution: to prove this, let's suppose we are seeking for the two parameters a_1 and a_2 that characterize the rect $\hat{y} = a_1 x + a_2$ and a dataset containing five pairs (x, y), four of them disposed symmetrically along a rect that pass through the fifth. It's clear that the rect that minimize the distance from that dataset, namely the Norm-1, should pass through the fifth point (allowing the fifth residual to be zero) and lay between the other points, as shown in Figure 7.

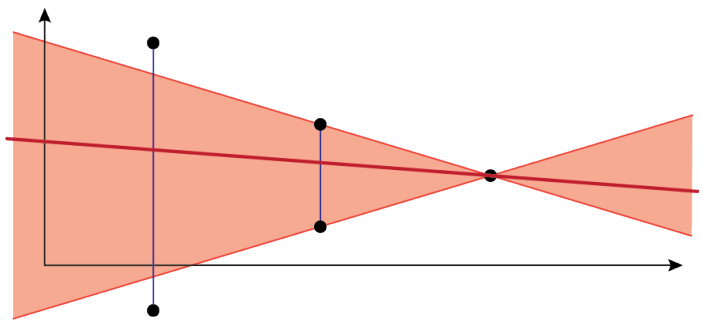

Figure 7: In *red*, one of the infinite N-1 fitting rects for an example of symmetric points (absolute errors in *blue*); the *orange* region entirely covers the set of all possible solution for the minimum.

The entire stack of rects pivoted on the fifth points and contained on the nearest two are all solutions of the minimum problem: suppose to tilt the red rect slightly, the sum of the blue errors would still be the same, thus producing the same cost as the starting rect. Also,

since one can tilt the line in infinitely small increments, this also shows that if there is more than one solution, there are infinitely many solutions.

Focusing on our case of study, as long as we work with timeseries, there isn't any pair of points that posses the same abscissa, so the solution for the N1 minimization problem is unique. All of this is however explained to keep in mind that, generally speaking, a subset of N1 minimization problems can have multiple solutions.

6.1 Implementation

As previously done for the LEL method, any given signal must be splitted in piecewise linear traits, e.g with the procedure described in Appendix A.

The minimum problem exposed in Equation 7 is then solved using `fminunc`[5] Matlab function, with $\boldsymbol{\theta}_{LS} = \arg\min_{\theta} \|\boldsymbol{y} - M\boldsymbol{\theta}\|_2$, the *Least Square* solution, as initial value. This choice is taken only for convenience and it could have been any other value.

6.2 Results

Figure 8 shows the result of N1 fitting method for the signal generated as reported in article;[2] Figure 9 shows results for the given USBL reference signals 4.4.

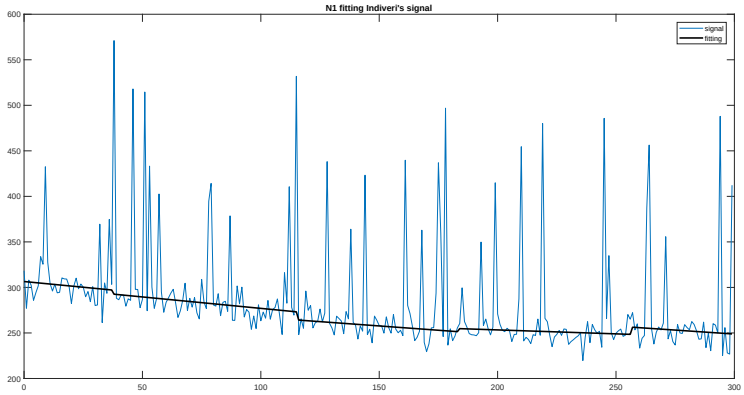

Figure 8: Generated signal and N1 fitting result

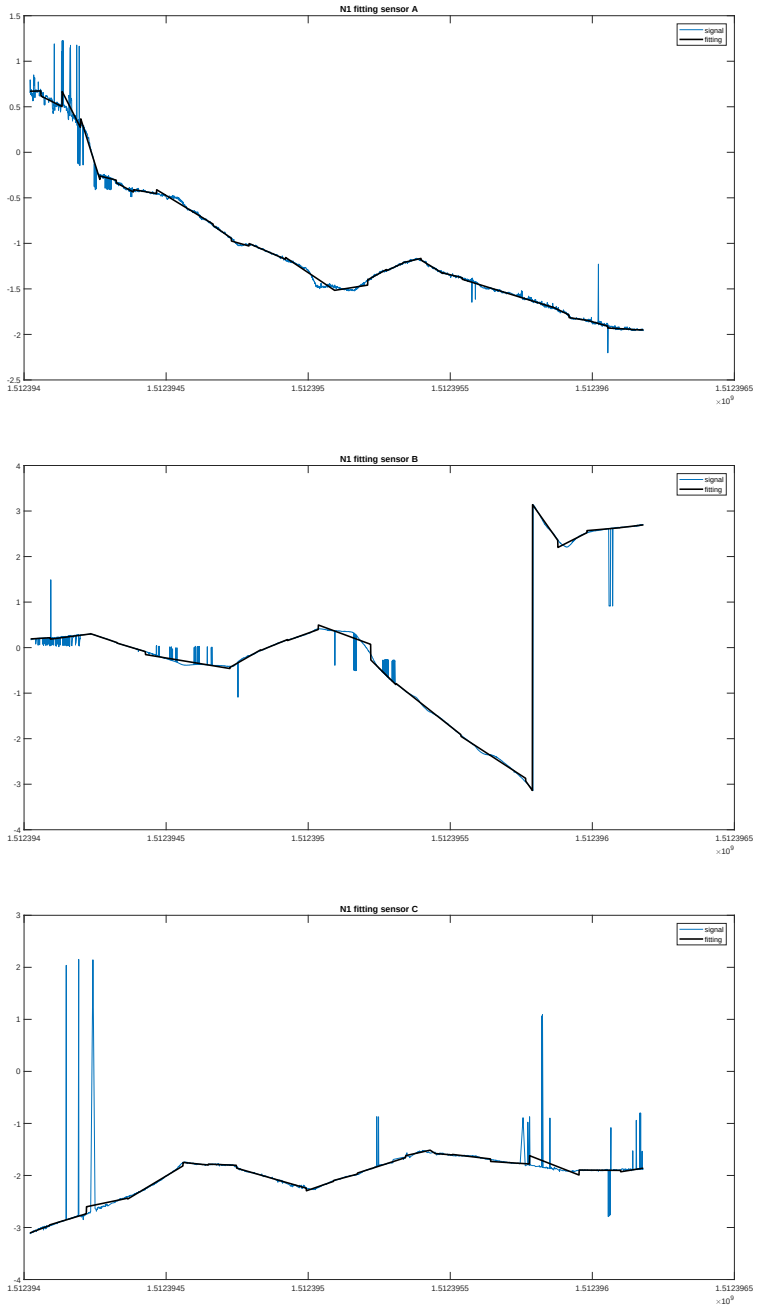

Figure 9: Signal A, B and C and N1 fitting results

7 Quantile regression method[6]

From a statistical point of view, standard linear regression can be seen as a conditional mean estimation.

Given a set of measurements Y his relative covariate X and the conditional mean $E(Y|X = x) = \mu(x)$, the goal is to come up with an estimated value $\hat{\mu}(x)$, by minimizing the squared error loss

$$\hat{\mu}(x) = \arg\min_z E((Y - z)^2|X = x) \tag{8}$$

Conditional mean only returns a partial information of Y conditional distribution. Quantile regression allows for analysis of other features of interest in a distribution.

Considering the conditional distribution function

$$F(y|X = x) = P(Y \leq y|X = x) \tag{9}$$

the α-quantile $Q_\alpha(x)$, is defined as the quantity for which the probability of Y being smaller than $Q_\alpha(x)$ for a given $X = x$ exactly equal to α is:

$$P(Y \leq Q_\alpha(x)|X = x) = \alpha \tag{10}$$

Consequently $Q_\alpha(x)$ can be defined as:

$$Q_\alpha(x) = \inf_y \{F(y|X = x) \geq \alpha\} \tag{11}$$

Quantiles give back more information about y distribution in respect to X than conditional mean as they can be used to build prediction intervals: suppose we want to define a 95% prediction interval for values of Y centred on its median value, then we have:

$$I(x) = [Q_{0.025}(x), Q_{0.975}(x)] \tag{12}$$

This allows for outlier detection, since a new observation (x, y) can be pinpointed as an outlier whereas its observed value y is extreme with regard to the predicted conditional distribution function.

Unfortunately, there is no standard rule to define how much of a deviation makes up for an outlier. The boundary of decision is left to the final user, who has to consider signal's characteristics and estimate the percentage of outlier contamination.

Quantile regression aims to estimate conditional quantiles from data. Usually quantile estimation is tracked back to an optimization problem by minimizing a loss function.

Different approaches to optimize the parameter so that error is minimal include parametric methods, non parametric methods (as quantile smoothing splines) and tree-based methods.

We are interested in a technique different from all of the above, as it does not require the minimization of a loss function, but relies on *Random Forest* computing.

7.1 Random forests

7.1.1 Introduction to Random Forest method

Random forest (shortened **RF**) designates a *supervised learning algorithm*. As the name suggests RF starts from an ensemble of decision trees, trained with techniques such as *bagging* method. RF can be deployed to solve both classification and regression problems.

While a decision tree searches for particular features by node-splitting, a random forest combines the result of the most important multiple decision trees by searching for the best feature among a subset of features.

This results in a wide diversity, generally leading to a better model than a single decision tree estimation.

A simple explanation, by analogy for the sake of comprehension Suppose Andrew wants to go on a vacation, but all he knows is that he likes **hot sea** places, preferably in **Europe**. He then proceeds to take advice from his friends by letting them ask him 2 questions each.

Each of Andrew's friends create rules to guide his decision about what he should recommend, by using the answers that Andrew gives (fig. 10). Let's pretend they only advise 4 places, which features (climate, landscape type and continent) are shown in table 3.

	Sea	Mountain
Hot	Caribbeans (America)	Uluru (Australia)
Cold	Irish Coast (Europe)	Matterhorn (Europe)

Table 3: Possible locations for Andrew's vacation

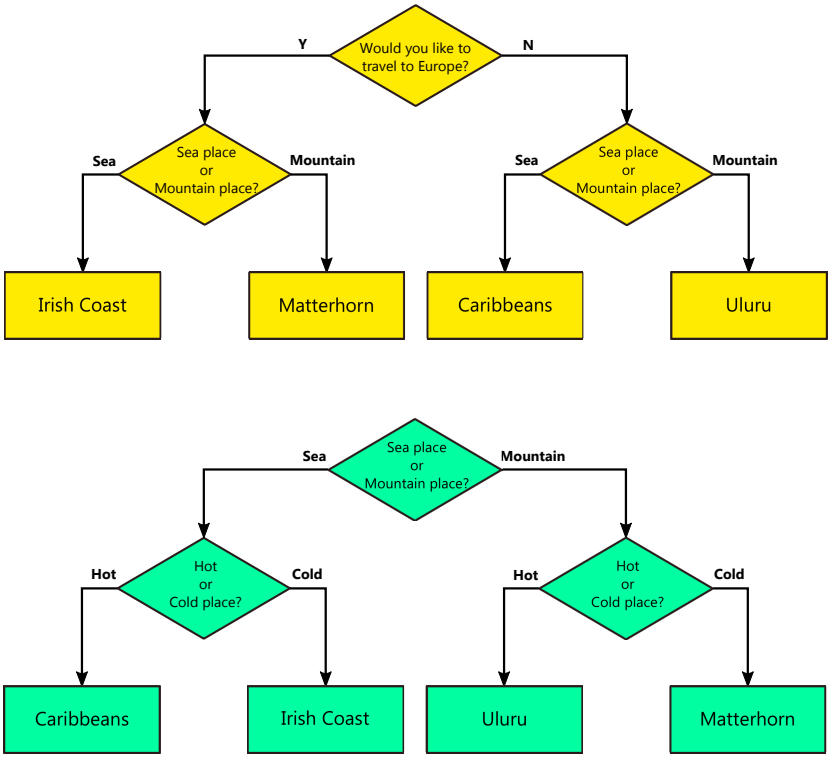

Figure 10: Andrew's friends schemes of advice

Afterwards Andrew starts making more and more of his friends to advise him and they again ask for different questions. Ultimately Andrew chooses a location among the ones advised.

It should be noted how while each of Andrew's friend display the typical tree approach, Andrew follows the typical random forest approach.

Feature importance It is also possible to measure the importance of a feature by assigning a score value to features in relation to their contribution to the solution. Features with little to none impact in solution computation can be discarded as reduction of features number prevents over fitting problems.

Referring to the example above, Andrew values in order the features climate, landscape and continent preferring hot, sea and Europe. By coding this preferences into a score(tab. 4), for each location we have a value shown in tab. 5.

Feature	Score
Hot	0.5
Cold	0.3
Sea	0.3
Mountain	0.25
Europe	0.2
Not Europe	0.1

Table 4: Features scores

Location	Score
Caribbeans	0.9
Uluru	0.85
Irish Coast	0.8
Matterhorn	0.75

Table 5: Location scores

First friend advises "Irish Coast", second friend advises "Caribbeans". Andrew likes the latter better and consequently decides his second friend is more reliable in terms of travel advising. Doing so he will value more his second friend opinion in future questioning (thus, performing a feature selection).

Difference between random forest and random trees Starting from a set of features, a random tree will generate some rules, a RF will generate several decision trees and then will average the results.

In terms of the previous example , Andrew comes up with two results after questioning: Caribbeans (scoring 0.9) and Irish coast (scoring 0.8). By averaging them ($\frac{0.9+0.8}{2} = 0.85$) he finally decides to visit Uluru.

A decision tree with a large enough depth is also prone to suffer over fitting, while RF prevents this problem by creating random subsets of features and building smaller trees (this procedure is known as *bagging*)

Important parameters of a random forest The predictive power of a random forest comes down to essentially three factors:

- **Number of estimators**, namely the number of trees the algorithm builds before taking average of predictions
- **Maximum number of features** that the RF is allowed to try in an individual tree
- The **minimum number of leafs** required to split an internal node

In conclusion RF makes up for an efficient approach solving both regression and classification problem. Limitation of the technique are mainly due overfitting problems, due to the fact that precision is bound to the number of trees (large number of threes may result in expensive computation) and that RF is still a predictive method, not suitable for critical tasks requiring an hard deterministic result.

7.1.2 Notation

Given a random parameters vector θ, we refer to the tree grown from θ (which contains the variables considered for split points at each node) as $T(\theta)$, and to the tree's leaf nodes as ℓ.

For every possible value x of the predictor variable X, there is one and only one leaf $\ell(x, \theta)$ that represents the end of the path when dropping x down the tree.

The prediction of a single tree $T(\theta)$ for a new data point $X = x$ is obtained by averaging over the observed values in leaf $\ell(x, \theta)$.

Let the weight vector $w_i(x, \theta)$, be given a positive constant if an observation X_i is part of leaf $\ell(x, \theta)$, and 0 if not.

The weights sum to one, thus:

$$w_i(x, \theta) = \frac{1_{\{X_i \in R_{\ell(x,\theta)}\}}}{\#\{j : X_j \in R_{\ell(x,\theta)}\}} \tag{13}$$

where $1_{\{X_i \in R_{\ell(x,\theta)}\}}$ is an indicator function that assumes the value 1 when X_i belongs to the subspace corresponding to the leaf $\ell(x, \theta)$, $R_{\ell(x,\theta)}$, or zero otherwise.

A prediction of a single tree, given a covariate $X = x$, is then given by the weighted average of the original observations $Y_i = 1, \ldots, n$:

$$\hat{\mu}_{tree}(x) = \sum_{i=1}^{n} w_i(x, \theta) Y_i \tag{14}$$

Using RF, the conditional mean $E(Y|X = x)$ is approximated by the averaged prediction of k single trees, each constructed with an independent identically distributed vector $\theta_t, t = 1, \ldots, k$. Let $w_i(x)$ be the average of $w_i(\theta)$ over this collection of trees,

$$w_i(x) = \frac{1}{k} \sum_{t=1}^{k} w_i(x, \theta_t) \tag{15}$$

The prediction of random forests is then

$$\hat{\mu}_{RF}(x) = \sum_{t=1}^{k} w_i(x) Y_i \tag{16}$$

The approximation of the conditional mean of Y, given $X = x$, is thus given by a weighted sum over all observations. The weights vary with the covariate $X = x$ and tend to be large for those $i \in \{1, \ldots, n\}$ where the conditional distribution of Y, given $X = X_i$ is similar to the conditional distribution of Y, given $X = x$.

7.2 Quantile regression forest

Conditional mean prediction can be used to track the full conditional distribution. Given

$$F(y|X = x) = P(Y \leq y|X = x) = E(1_{\{Y \leq y\}}|X = x) \tag{17}$$

An approximation of $F(y|X = x)$ can be computed as

$$\hat{F}(y|X = x) = \sum_{i=1}^{n} w_i(x) 1_{\{Y \leq y\}}. \tag{18}$$

using the same weigths $w_i(x)$ as for random forests, defined in Equation 15. Summarizing, the algorithm steps are:

1. Grow k trees $T(\theta)$, for every leaf take note on all observations.
2. Drop $X = x$ to all trees, compute $w_i(x, \theta_t)$, $i = \{1, \ldots, n\}$, $t = \{1, \ldots, k\}$ for each tree
3. Compute $w_i(x)$ as an average of all $w_i(x, \theta_t)$
4. Compute $\hat{F}(y|X = x)$ for all Y using the weights from step 2

The conditional quantiles, can be estimated by applying the conditional distribution estimation $\hat{F}(y|X = x)$ to definition 11, resulting in

$$Q_\alpha(x) = \inf_y \{\hat{F}(y|X = x) \geq \alpha\} \tag{19}$$

There's a key diference between Random Forests and Quantile Regression Forests.

For each node in each tree Random Forests keep only the mean of the observation that fall into this node and neglects all other information

Quantile Regression Forests on the other hand, keeps the value of all observations in the nodes, not just their mean, and uses this information to estimate the conditional distribution.

7.3 Program structure

The main script operates as shown in Fig. 11:

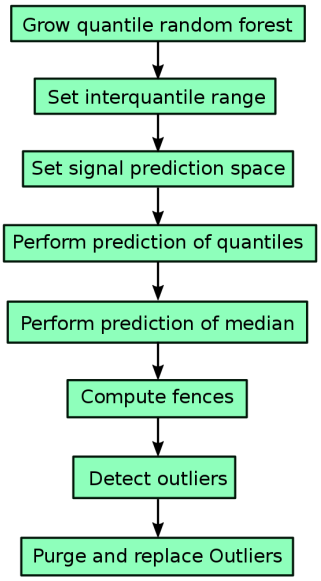

Figure 11: Quantile regression flowchart

Grow quantile random forest MatLab function `TreeBagger`[7] is used to create a structure consisting in a bag of 200 regression trees. A tree bag consists in an aggregation of decision trees which results are combined in order to avoid over fitting problems by improving generalization.

Set inter quantile range Inter quantile range is user defined by setting two values of quantiles. Quantiles Q are defined in a $[0, 1]$ range. To compute a quantile $Q_{\%} = Q * 100$ means to compute the value below which the $Q_{\%}$ percentage of samples population is included, thus dividing the population into two sets: one set containing the samples lesser than the computed quantile and one set of samples greater than the quantile.

Here a **lower** and an **upper** quantile are computed, respectively below and over the 0.5 quantile (median) in order to recognize as outliers those samples lesser than the lower quantile and greater than the upper quantile.

Set signal prediction space Signal is divided into a user defined number of equally spaced intervals.

It should be noted that the intervals should be representative in terms of data distribution, meaning that the corresponding median represents an acceptable fit of the signal.

Perform prediction of quantiles Prediction of quantiles is performed on each interval by MatLab function `quantilePredict`.[8]

Perform prediction of median Predicted median marks the 0.5 quantile.

Compute fences Fences separating samples from outliers are computed by linearly interpolating the predicted quantiles, tolerance of the obtained fences can be adjusted by adding multiples of the interquantile range.

Detect outliers All samples falling outside fences are marked as outliers.

Purge and replace outliers Detected outliers are replaced by a linear interpolation the remaining samples.

7.4 Results

Figures 12, 13, 14 show the elaboration of the three given reference signals.

Table 6 shows the settings used to obtain quantile regressions of each signals. It should be noted how performing new elaborations with the same input parameters results in a slightly different estimation of quantiles and thus fences. This happens because every time the "Compute" button is pressed a new forest is grown by a new execution of the main script (fig. 11).

	Signal A	Signal B	Signal C
Prediction space divisions	70	70	70
Lower Quantle	0.5	0.35	0.3
Upper Quantile	0.8	0.51	0.8
Tolerance	0	8	8

Table 6: QRODApp settings used for elaborations in figures 12, 13, 14

.

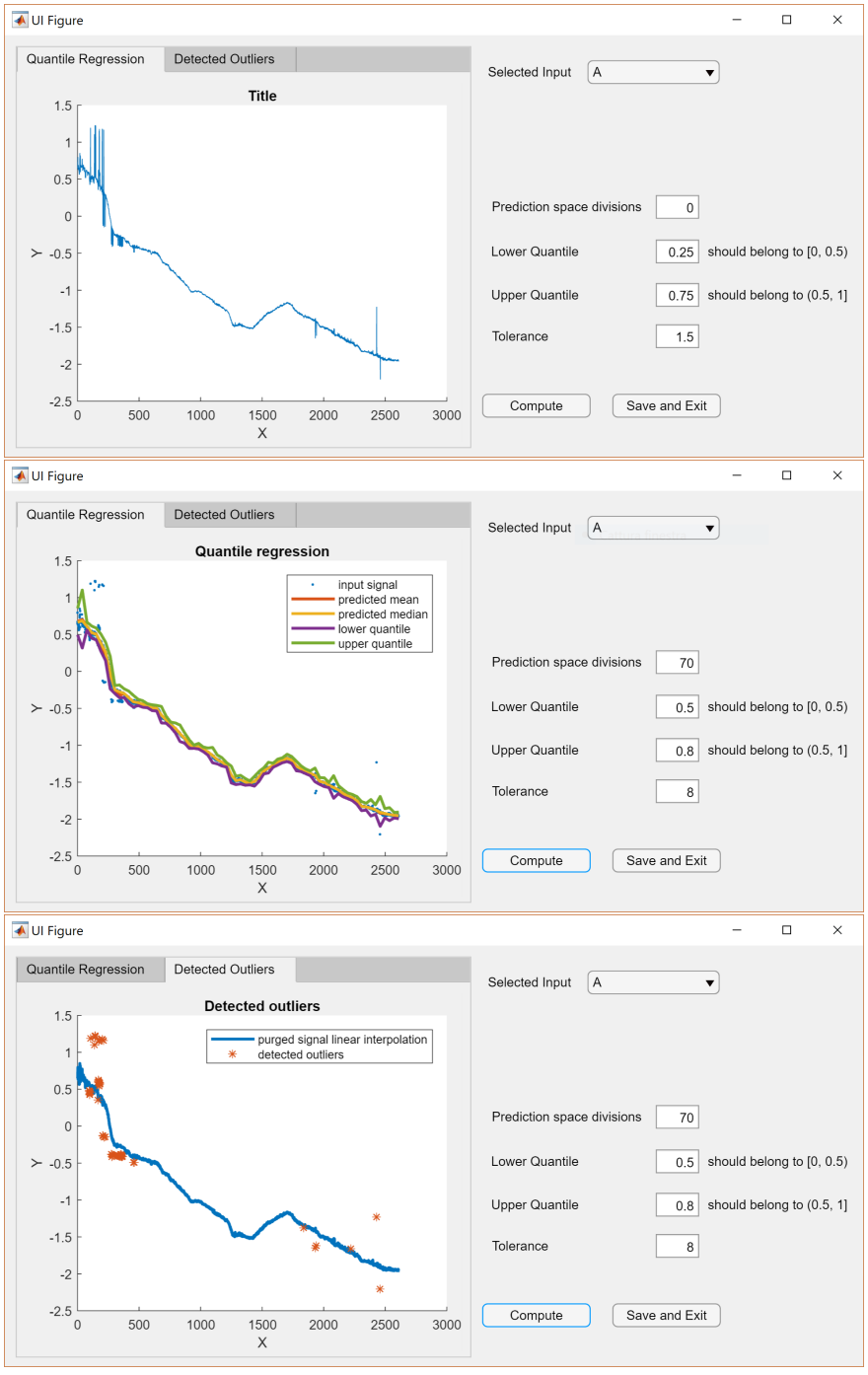

Figure 12: Signal A outliers detection by quantile regression

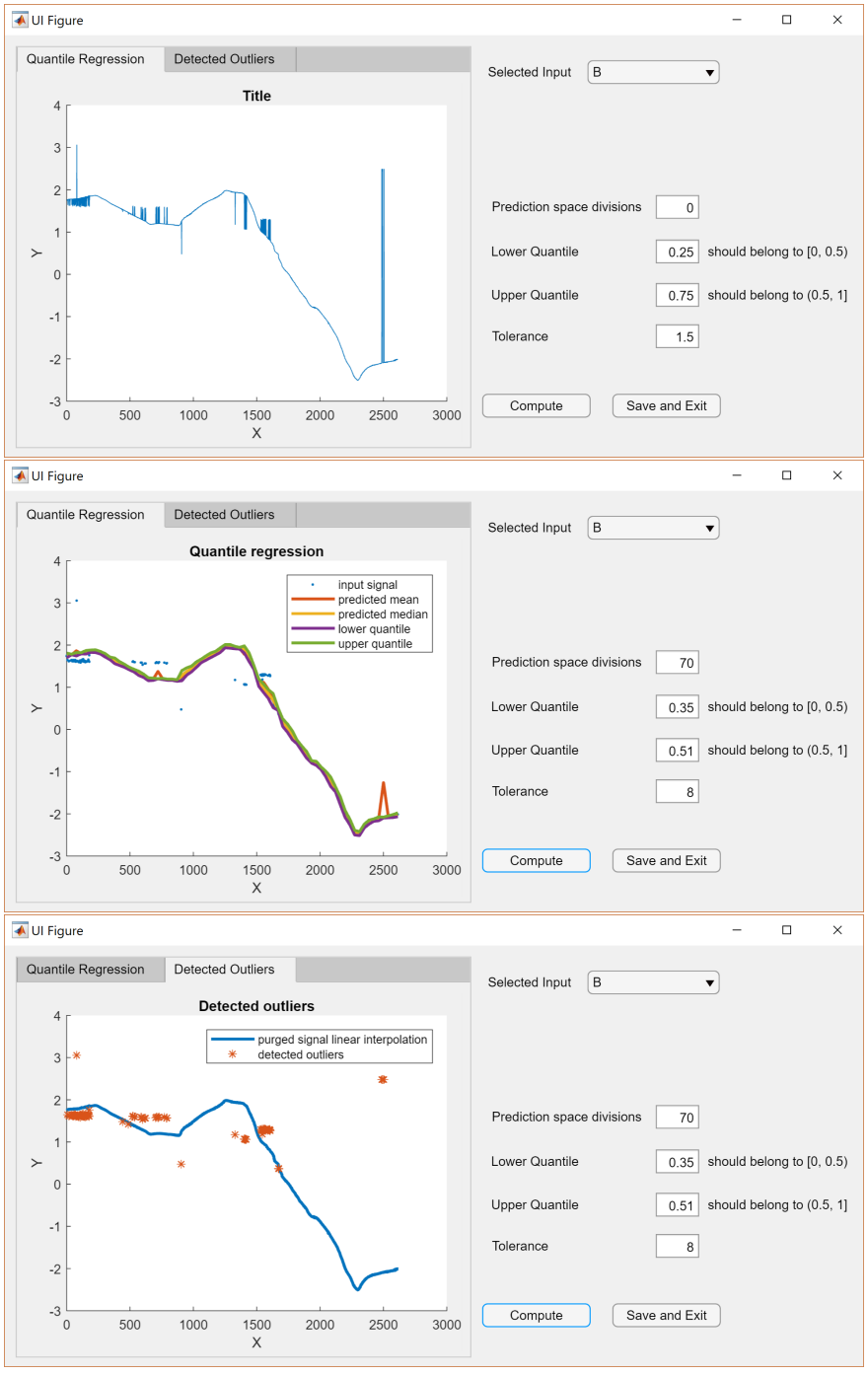

Figure 13: Signal B outliers detection by quantile regression

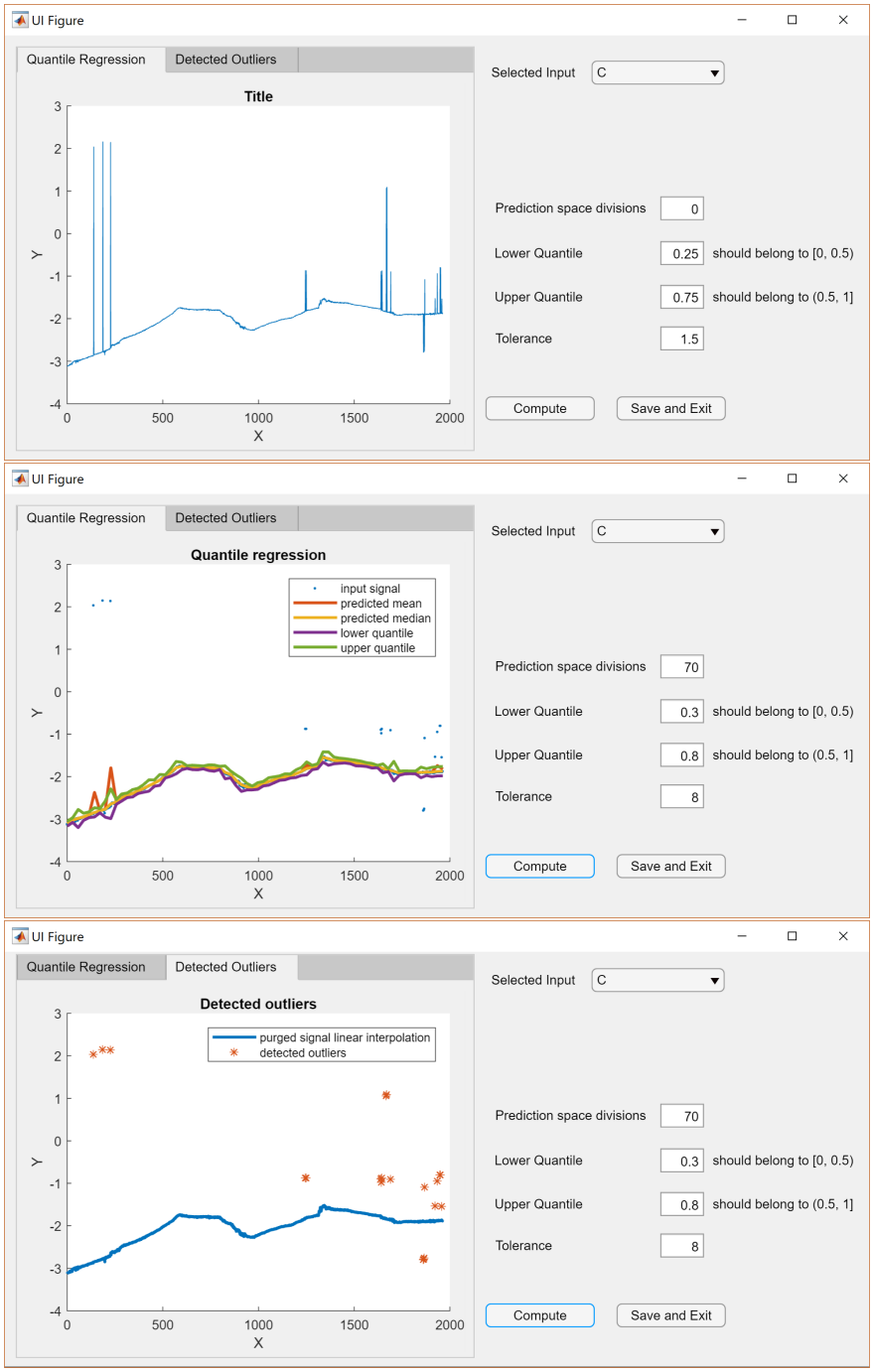

Figure 14: Signal C outliers detection by quantile regression

7.5 QRODApp quick guide

Application `QRODApp.mlapp` implements the quantile regression script described above, providing for a GUI to conveniently supervise the quantile regression computation parameters. In order to compute the detected outliers array, user should follow the steps listed below:

1. The outlier affected signal should be saved in workspace as a $n \times 1$ array of values, or as a $n \times 2$ matrix of pairs (*timestamp*,*values*).
2. Open QRODApp and select the signal from the dropdown menu
3. Set the parameters to perform quantile regression with. Division space requires at least 2 points (in order to generate a single segment).
4. Lower and Upper quantiles should range respectively $[0, 0.5)$ and $(0.5, 1]$. It is still possible to select 0.5 as a value for both, in this case the quantile corresponds to the median.
5. Tolerance parameter (let's call it k) regulates the distance between lower and upper fences relative to the computed quantiles, by respectively subtracting and adding k times the interquantile distance ($I = Q_{upper} - Q_{lower}$) computed for each division point.
6. Clicking the "Compute" button, the app performs computations and displays the results by plotting the original signal with the computed fences in the "Quantile Regression" tab, and the linear interpolation of the purged signal with the points pinpointed as outliers in the "Detected Outliers" tab.
7. "Save and Exit" button closes the app returning a $p \times 1$ array of the signal without the detected outliers (where the holes are filled with linear interpolations) and a $p \times 2$ array containing the position index in first column and the value in second column of the p detected outliers.

8 Conclusions

The three methods proposed proved themselves to be valuable approaches in linear regression problems in presence of high outlier pollution, providing a robust solution compared to classic regression techniques such as standard least squares linear regression, as shown in Figures 15 and 16.

The main advantage is that none of the proposed techniques require a priori information about the magnitude, density and distribution of outliers affecting data.

LEL method results in best performance among the regression methods, but requires longer computational time, thus deployment is preferable over LAD when dealing with heavily polluted signals, as LAD (and even LS) performs sufficiently good for sporadically contaminated data.

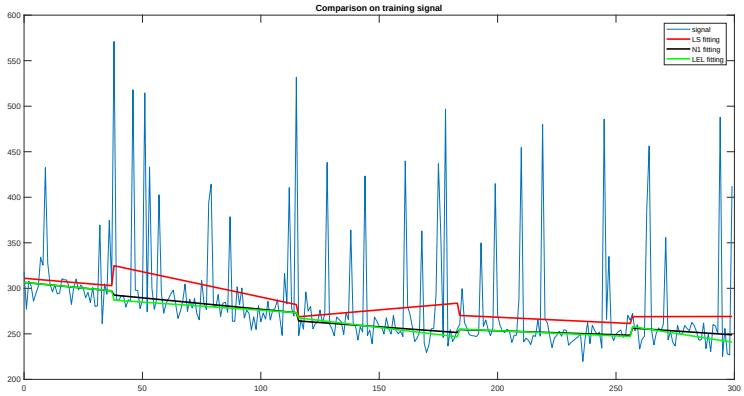

Figure 15: Generated signal and LS, N1 and LEL fitting results

Quantile regression methods provides for a diagnostic approach, in opposition to robust (accomodating) regression methods. Implementation by random forests proved to be a lightweight, still reliable solution to detect outliers.

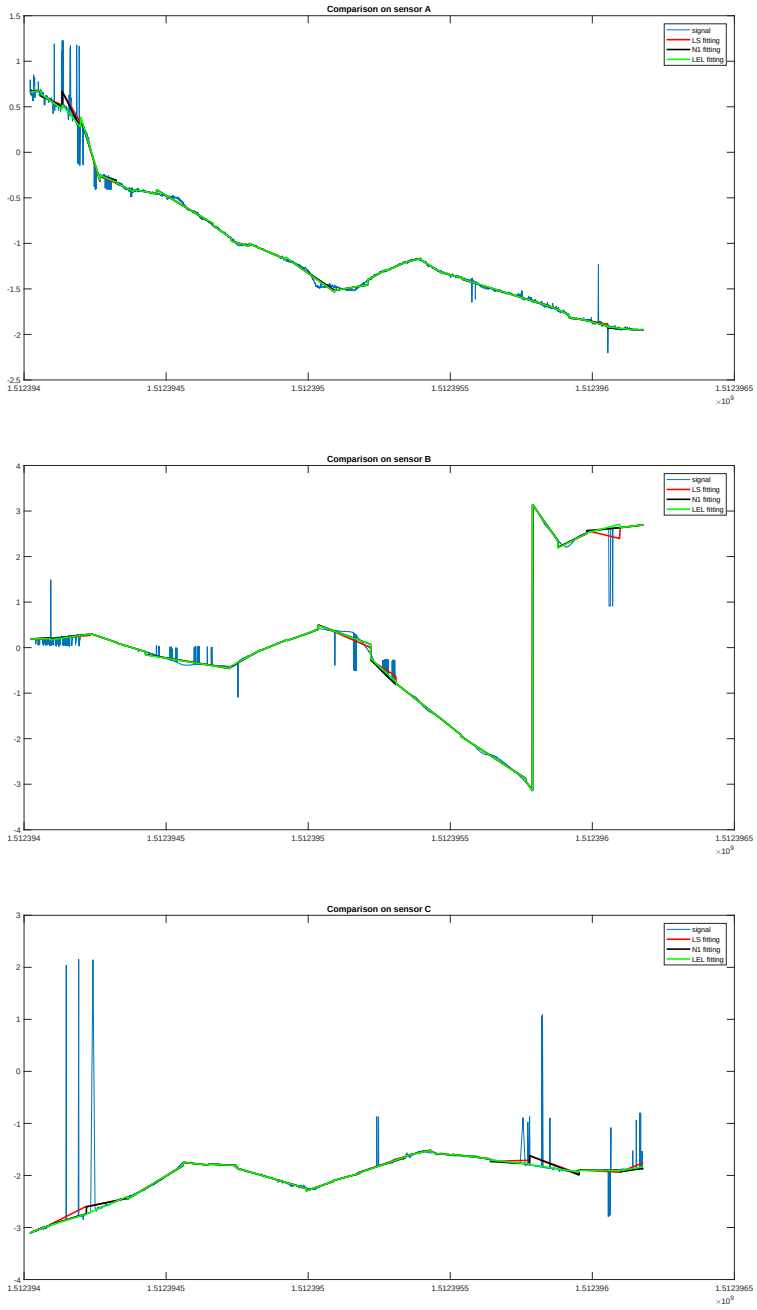

Figure 16: Signal A, B and C and LS, N1 and LEL fitting results

A Automatic data segmentation

Methods exposed in Section 5 (*LEL*) and 6 (*N1*) are built to reconstruct rectilinear segments of data.

In order to provide a versatile method to subdivide any given dataset (interpreted as time-series) in linear regions, an automatic approach is then proposed.

Given a reference signal similar to the ones exposed in Section 4.4, composed by a series of rectilinear traits polluted by noise and outliers, the extremas of each segment can be identified by looking at the second derivative of the signal.

Noise, and expecially outliers, would be amplified by a simple differentiation of the dataset, obligating us to apply *low pass filters* before the procedure.

The complete proceeding is then the following:

- load a discrete time-series and, if formed by angular data, unwrap it (by adding or subtracting 2π each time the signal crosses limit values, e.g. $[-\pi, \pi]$);
- apply a Fast-Fourier-Transform to the signal;
- select a cut frequency beyond which its amplitude should be suppressed, as in Figure 17;
- remove any value of the signal, in the frequency domain, above the chosen threshold;
- revert the Fourier transformation to the time domain to obtain a smoothed signal;
- double differentiate it, simply through incremental differentiation;
- identify which peaks of the second derivative are influential on the recognition of sections limits (e.g. by imposing thresholds), as in Figure 18;
- take the index of those peaks and use them to split the original signal in rectilinear segments.

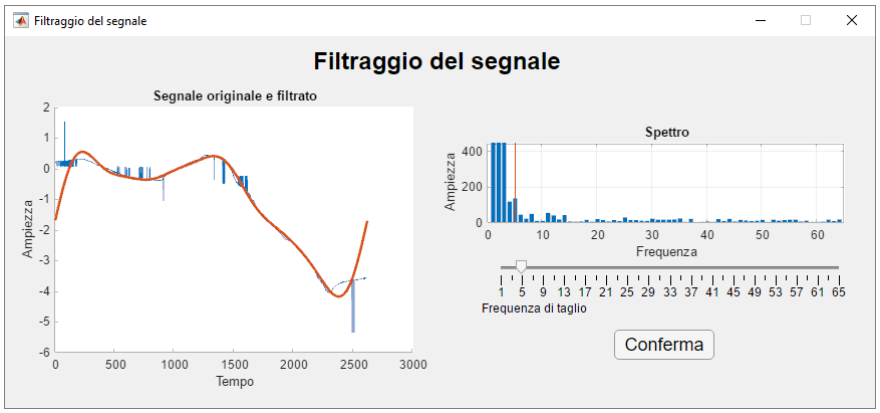

Figure 17: Original and smoothed signal on the left plot; its Fourier transform and the selected cut frequency on the right plot.

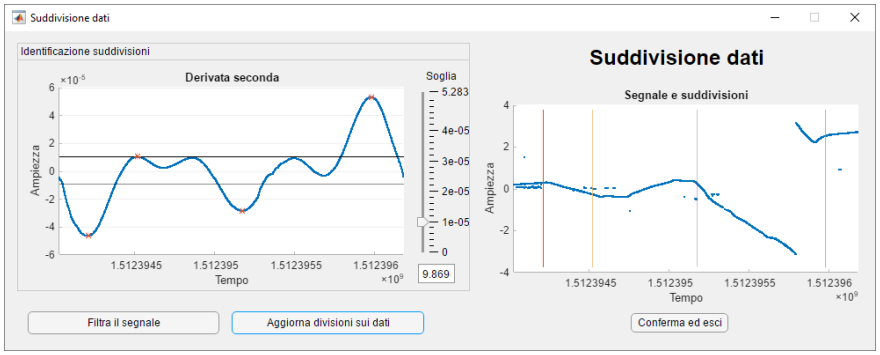

Figure 18: Double derivate of the filtered signal on the left plot; the sections identified on the right.

References

[1] **Jerome Vaganay, Jhon J. Leonard, James G. Bellingham Massachusetts Institute of Technology**
Outlier Rejection for Autonomous Acoustic Navigation.
Proceedeings of the 1996 IEEE International Conference on Robotics and Automation Minneapolis - Minnesota April 1996.

[2] **Giovanni Indiveri**
An Outlier Robust Filter for Maritime Robotics Applications.
PALADYN Journal of Behavioral Robotics.

[3] **Victoria J. Hodge, Jim Austin**
A Survey of Outlier Detection Methodologies.
Artificial Intelligence Review 22: 85-126, 2004.

[4] **Mathworks FileExchange** `fminlbfgs` function reference page.
https://it.mathworks.com/matlabcentral/fileexchange/23245-fminlbfgs-fast-limited-memory-optimizer.

[5] **Mathworks Help** `fminunc` function reference page.
https://it.mathworks.com/help/optim/ug/fminunc.html.

https://it.mathworks.com/help/optim/ug/fminunc.html

[6] **Nicolai Meinshausen**
Quantile Regression Forests.
Journal of machine learning research 7 (2006) 983-999.

[7] **Mathworks FileExchange** `TreeBagger` function reference page.
https://it.mathworks.com/help/stats/treebagger.html

[8] **Mathworks FileExchange** `quantilePredict` function reference page.
https://it.mathworks.com/help/stats/treebagger.quantilepredict.html