Máster Universitario en Sistemas Espaciales

2021/2022

# Caso de Estudio 2: Anomaly detection in time series for space applications

Autor:

David Huergo Perea

Tutor:

Juan Antonio Hernández

Fecha:

25-01-2022

# Contents

## List of Tables

## List of Figures

# Abbreviated terms

| | |
|---|---|
| **CNN:** | Convolutional Neural Network. |
| **LSTM:** | Long Short Term Memory. |
| **MVD:** | Mean Value Decomposition. |
| **MSL:** | Mars Science Laboratory. |
| **SMAP:** | Soil Moisture Active Passive satellite. |
| **STL:** | Seasonal and Trend decomposition using Loess. |

# 1 Introduction

A *time series* is a set of points that defines a function of the time. Each one of these points have a timestamp, which represents the moment when a certain variable was measured. There is a wide application of time series in almost every field, from medicine, for monitoring heartbeats, to space engineer, when a telemetry signal is received from a satellite.

The motivation behind this research is to improve the traditional methods used for time series analysis, decomposition and anomaly detection. Typically, the time series decomposition has been used for periodic time series analysis, as a completely different subject from the anomaly detection in time series. However, both approaches have several points in common and their objective is to extract some kind of information from the time series, which is usually hidden after a simple visual inspection of the data.

In this research, both concepts, time series decomposition and anomaly detection, will be joined to improve the performance of these methods. Moreover, a new decomposition procedure is proposed: the *Mean Value Decomposition*, which has shown very good results in comparison with other widely used methods, as it could be the STL decomposition.

After this concept has been completely developed, the whole approach has been tested with two validated databases from UCR [1] and NASA [2]. The results show a high similarity between the detected anomalies and the real ones, surpassing the 85% of success for the NASA's database.

The results extracted with this new approach suggest the possibility of using computational analysis for the detection of anomalies in real space missions, where most of anomalies are detected by experts in specific fields that spend several hours looking for any possible anomalous behaviour. Therefore, the use of time series analysis in space missions could lead to a great reduction of costs and time during every phase of the mission and it would allow to have a robust control of every time series and a real time monitoring.

## 2 Time series decomposition

Time series analysis involves extracting the relevant information from the available data, which represent one or more functions of time, usually composed of a discrete number of points. The underlying information of a time series, if correctly processed, should show the patterns, trends and other internal structures. These structures, in conjunction with some characteristic features, such as statistical parameters or principal frequencies among others, make possible to understand the evolution of the time series as well as to detect any kind of anomaly and to forecast its future behavior.

In order to highlight these behavioral structures, a common approach [3] is to decompose the time series in sub-series with less information which show, in a clearer way, the different factors involved in the process.

An usual practice when decomposing a time series is to use an *additive model*. This model formulates the complete time series as the sum of a trend component, a seasonal [1] or cyclic component and the residuals or irregular variations:

$$X_{add}(t) = T(t) + S(t) + R(t) \tag{1}$$

where $T(t)$ is the trend, $S(t)$ is the seasonal variation and $R(t)$ is the residuals.

Also, it is possible to use a *multiplicative model*, where the time series is expressed as the product of a trend component, a seasonal component and the residuals:

$$X_{mul}(t) = T(t)\, S(t)\, R(t) \tag{2}$$

Nevertheless, from here on out the time series analysis will be carried out considering an additive decomposition model, because the multiplicative model can be transformed to an additive model if a logarithmic scale is used:

$$\log X_{mul}(t) = \log T(t) + \log S(t) + \log R(t) \tag{3}$$

[1]The word *seasonal* is reserved for those periodic variations with a calendar-based period, such as daily, weekly or monthly variations, or even if they are influenced by seasons or holidays. However, in most articles the term *seasonal* is used for any kind of periodic variation, due to the development of time series analysis has been significant in many fields with a strong seasonal component.

## 2.1 STL

*Seasonal and Trend decomposition using Loess* (STL) [4] is a robust filtering procedure for decomposing a time series, which follows an additive model. This method has proven to be efficient, simple and flexible, and has become one of the most used for time series analysis.

### 2.1.1 The loess smoothing function

One of the keys behind STL is the use of a loess smoother. The smoothed time series, $\hat{g}(t)$, is defined for every $t$ in $[t_0, t_{end}]$, even if the chosen time is not sampled in the time series data. The value of $\hat{g}(t)$ is computed based on the $q$ closest timestamps to $t$, where $q$ is an arbitrary positive integer. Each selected sample, in a time $t_i$, is given a *neighborhood weight*, $v_i(t)$, depending on the distance between $t$ and the timestamp:

$$v_i(t) = W\left(\frac{|t_i - t|}{\lambda_q(t)}\right) \tag{4}$$

where $\lambda_q(t)$ is the distance between $t$ and $t_q$, the farthest $t_i$ among the $q$ selected; and $W$ is defined as the tricube weight function:

$$W(u) = \begin{cases} (1-u^3)^3 & \text{for } 0 \leq u < 1 \\ 0 & \text{for } u \geq 1 \end{cases} \tag{5}$$

Besides, the weights $v_i$ may be improved with a *reliability factor* $\rho_i$, which represents the relevance of that sample in the whole time series. This factor is obtained based on an iterative process, as it will be explained below.

Let $n$ be the number of samples or timestamps. If the chosen $q > n$, then:

$$\lambda_q(t) = \lambda_n(t)\frac{q}{n} \tag{6}$$

with $\lambda_n(t)$ equal to the distance from $t$ to the farthest $t_i$.

Once the $q$ weights are calculated, $\hat{g}(t)$ is obtained as the value of a locally-fitted polynomial of degree $d$ around $t$, with the weights previously calculated. For most applications, the use of a linear polynomial ($d = 1$) or a quadratic polynomial ($d = 2$) is enough.

### 2.1.2 The decomposition procedure

STL decomposition consists of a recursive procedure based on two different loops. Within the inner loop, the trend and seasonal components are calculated and updated; while each pass of the outer loop includes a whole run of the inner loop and a computation of the reliability factors, to improve robustness in the next run of the inner loop.

The inner loop consists of a trend and seasonal smoothing. It is necessary to suppose certain initial conditions before beginning the iterative process: each reliability factors are equal to 1 and the trend, $T(t)$, may be an arbitrary function, but $T(t) = 0$ works well [4].

#### Seasonal component

First, the time series $X(t)$ is detrended:

$$X^*(t) = X(t) - T(t) \tag{7}$$

Then, the detrended time series is smoothed by the loess smoothing function (sec. 2.1.1) with $q = n_s$ and $d = 1$, with a new time series $C(t)$ as a result. The number, $n_s$, must be an odd integer and the function is smoother as $n_s$ increases. It is a critical parameter to correctly determine the seasonal component and has to be tailored to each application, but should be greater than or equal to 7 [4].

The next step is to apply a low-pass filter to the smoothed time series. It consists of three consecutive moving average transformations, of length $n_p$, $n_p$ and 3 respectively, with $n_p$ as the period of the seasonal component; followed by a loess smoothing with $q = n_l$ and $d = 1$, where $n_l$ is the least odd integer greater than or equal to $n_p$ [4]. The seasonal component can be obtained subtracting the output, $L(t)$, from the smoothed detrended time series:

$$S(t) = C(t) - L(t) \tag{8}$$

The moving average transformation of order $m$, with $m = 2k + 1$, is defined as [3]:

$$\hat{F}_t = \frac{X_{t-k} + X_{t-k+1} + ... + X_t + ...X_{t+k-1} + X_{t+k}}{2k + 1} \tag{9}$$

for each sample $X_i$ within the time series.

Furthermore, the period of the seasonal component, $n_p$, may be unknown in certain applications. To solve this problem, it should be obtained through visual inspection or an estimation can be carried out. In this study, an estimation has been obtained by analyzing the time series in the frequency domain.

### Period estimation

If the original time series has different main frequencies, the seasonal component should include those variations whose frequencies are higher, excluding the noise. The lower frequencies will be included within the trend component. In order to obtain the high frequency period, two different approaches have been tested: the first one simpler and faster, but less accurate, and the second one more complex and precise.

In the first approach, a *Fourier Transformation* is applied over the time series. Then, a threshold is defined, in this case $th = 0.05$. The chosen frequency, $f_{th}$, is the higher one that verifies:

$$A(f) > th \max(A(f)) \tag{10}$$

where $A(f) = Amplitude(\mathcal{F}[X(t)])$. Once $f_{th}$ has been obtained, the period of the seasonal component, $n_p$, measured in number of samples, is computed following the expression:

$$n_p = \text{int}\left(\frac{1}{f_{th}\,\Delta t}\right) \tag{11}$$

with $\Delta t$ as the time between timestamps.

The second approach tries to identify the seasonal period by comparing the first derivative of the time series computed in the time domain and in the spectral domain. The derivative in the time domain is obtained through centered finite differences:

$$\left.\frac{\mathrm{d}X}{\mathrm{d}t}\right|_i = \frac{X(i+1) + X(i-1)}{2\Delta t} \tag{12}$$

Then, the *Fourier Transformation* is applied over $N$ samples of the time series. The spectral derivative for each frequency $f_k$ can be computed following this expression:

$$\left.\frac{\mathrm{d}\mathcal{F}[X(t)]}{\mathrm{d}(i\omega)}\right|_k = i2\pi f_k \mathcal{F}[X(t)]_k \tag{13}$$

where $i$ is the imaginary unit and $\omega = 2\pi f$ is the angular frequency. Once the spectral derivative is known, the *Inverse Fourier Transformation* is performed to recover the derivative in the time domain.

The error $E$ is the result of subtracting both derivatives and it should be zero if $N$ is the period of the time series. However, real life time series usually do not have exactly the same period every cycle and they are not completely periodic functions, so an estimation of the period, measured in number of samples, is the integer $N$ which makes the error minimum.

In order to show graphically this approach, Figures 2.1 and 2.2 show a quasi-periodic time series and the error as a function of the number of samples, $N$, respectively. In this case, the minimum error is achieved with $N = 72$, which is, most of the time, the real period of the time series.

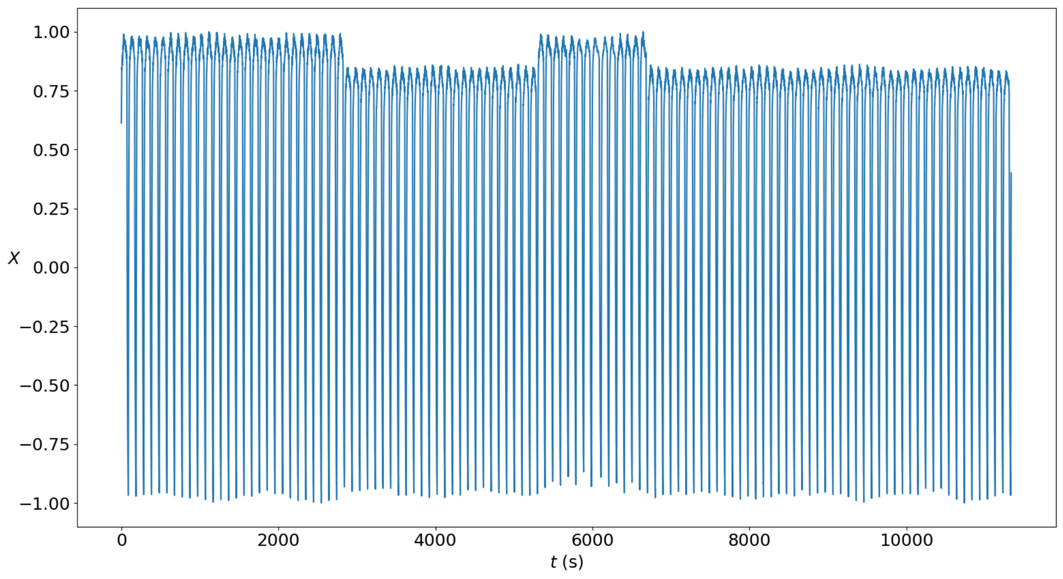

**Figure 2.1.** Quasi-periodic real dataset from a NASA spacecraft, that appeared in a KDD 2018 paper [1].

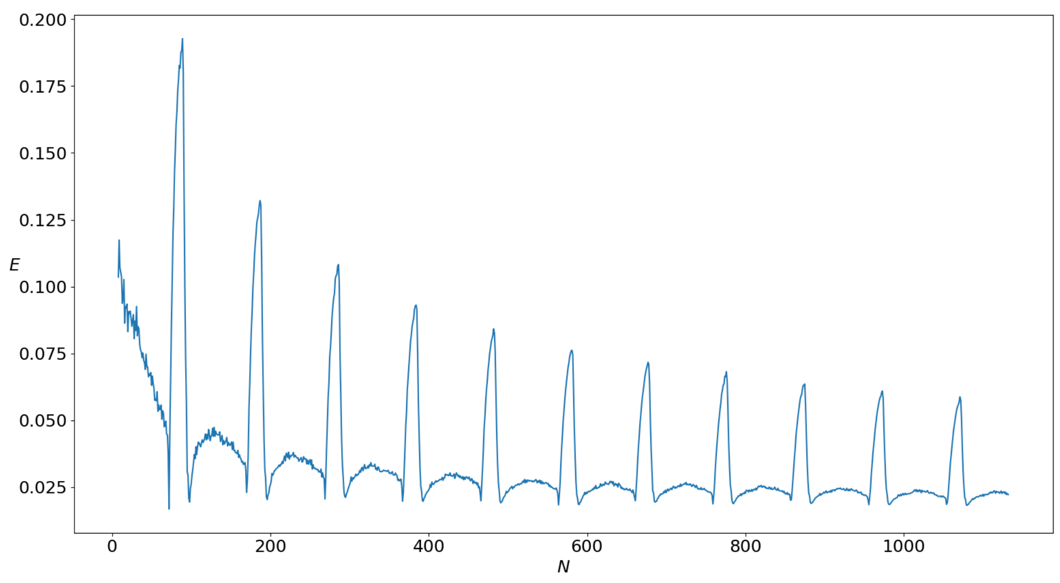

**Figure 2.2.** Error between derivatives as a function of the number of samples $N$.

### Trend component

When the seasonal component $S(t)$ has been computed, it is possible to recalculate the trend as part of the iterative process. The first step is to deseasonalize the time series:

$$X^{**}(t) = X(t) - S(t) \tag{14}$$

After that, $X^{**}(t)$ is smoothed by the loess smoothing function with $q = n_t$ and $d = 1$ and the trend, $T(t)$, is the output of this process. The parameter $n_t$ is an odd integer and the trend is smoother as this parameter increases its value. It is recommended to use [4]:

$$n_t \geq \frac{1.5 n_p}{1 - 1.5 n_s^{-1}} \tag{15}$$

The whole process of obtaining the trend and seasonal components completes the inner loop and every step is repeated $n_i$ times within the outer loop.

### The outer loop

Each time the inner loop runs, it is possible to obtain the third component of the time series, the residuals:

$$R(t) = X(t) - S(t) - T(t) \tag{16}$$

The outer loop involves the calculation of the reliability factors, also called robustness weights, for each sample [4]:

$$\rho_i = B\left(\frac{|R_i|}{6\,\mathrm{median}(|R|)}\right) \tag{17}$$

where $B$ is the bisquare weight function:

$$B(u) = \begin{cases} \left(1-u^2\right)^2 & \text{for } 0 \leq u < 1 \\ 0 & \text{for } u \geq 1 \end{cases} \tag{18}$$

These reliability factors multiply the weights $v_i$ when computing the loess smoothing function in the inner loop.

After $n_o$ iterations of the outer loop, the original time series $X(t)$ is completely decomposed in the trend component, $T(t)$, the seasonal component, $S(t)$, and the residuals, $R(t)$. Finally, it is recommended to define a convergence criterion to stop the iterative process if the criterion is satisfied. Let $U_i^k$ be the trend or the seasonal component after $k$ iterations, for each timestamp $t_i$; a possible convergence criterion could be [4]:

$$\frac{\max\left|U_i^k - U_i^{k+1}\right|}{\max U_i^k - \min U_i^k} < 0.01 \tag{19}$$

## 2.2 MVD

As an alternative to STL for decomposing a time series, a new method is proposed in this article: *Mean Value Decomposition* (MVD). This method makes use of the *Mean Value Theorem* to filter the time series and extract the relevant information. The main advantage of MVD is the small number of independent parameters. This fact facilitates the user the task of tailoring the algorithm to solve a specific problem.

### 2.2.1 The Mean Value Theorem

Let be a continuous function $f(t)$ within the interval $[a, b]$; there is at least one point $c$ at which the value of the function, $f(c)$, is equal to the average value of the function over the interval:

$$f(c) = \frac{1}{b-a} \int_a^b f(t)\,\mathrm{d}t \tag{20}$$

In order to turn this integral into a filter, it is necessary to apply this theorem to every sample of the time series. Furthermore, the main functionality of the filter is to smooth the time series; so it may be possible to remove the noise and separate the trend and the seasonal components. To achieve this smoothness, the new smoothed value must depend on its previous value and the value of adjacent samples.

Having into account these factors, it is possible to rewrite the mean value theorem using a three-point stencil numerical integration:

$$\hat{X}_j = \frac{1}{t_{j+1} - t_{j-1}} \int_{t_{j-1}}^{t_{j+1}} X(t)\,\mathrm{d}t = \frac{X_{j-1} + 2\alpha X_j + X_{j+1}}{2(\alpha + 1)} \tag{21}$$

where $\hat{X}_j$ is the smoothed value of $X_j$ and $\alpha$ is the order of the numerical integration. The higher the order, the softer the filter and smaller is the change between $X_j$ and $\hat{X}_j$.

This filter has to be applied recursively $N$ times to be effective; therefore, there are only two independent parameters: the order of the filter, $\alpha$, and the number of times it is applied, $N$.

### 2.2.2 The Mean Value Filter Gain

The properties of the filter are defined by its *gain*. The gain shows the degree of dampening of a certain sample after one pass of the filter, and it is given by the following expression:

$$G = \frac{\hat{X}_j}{X_j} \tag{22}$$

Therefore, the smaller the absolute value of the gain, the greater the dampening. There are 3 main possibilities:

- If $|G| > 1$, then the system is unstable and the time series will increase indefinitely.

- If $|G| = 1$, then the dampening is zero.
- If $|G| < 1$, then the time series will be smoothed, which is the main purpose of every filter.

In order to understand the behaviour of this filter, it is important to obtain the gain for different kind of functions.

### Periodic function

Periodic functions are represented by a blend of sinusoidal functions with different frequencies and these functions are the seasonal component of a time series. Each harmonic of these functions follows the general expression:

$$S(t) = \exp\left(i\, 2\pi f t\right) \tag{23}$$

with $i$ as the imaginary unit and $f$ as the frequency. When the general equation of the filter (21) is applied over a periodic function, the resulting gain is:

$$G = \frac{\alpha + \cos\left(2\pi f \Delta t\right)}{\alpha + 1} \tag{24}$$

where $\Delta t$ is the time gap between timestamps and $\alpha$ is the order of the filter.

This expression shows that the gain of a periodic function of the time is a periodic function of the frequency; hence, the degree of dampening is directly related to the main frequencies associated with the periodic time series. It is possible to distinguish three different behaviours, which are shown in the Figure 2.3:

- If $\alpha \geq 1$, then the maximum dampening is obtained in the middle point between the lower frequency, $f = 0\,\mathrm{Hz}$, and the higher frequency, $f = 1/\Delta t$. That is the critical frequency, $f_c = 1/(2\Delta t)$.
- If $0 \leq \alpha < 1$, then there are two frequencies with the maximum dampening, one lower than $f_c$ and one higher than $f_c$. As $\alpha$ decreases, the separation between these frequencies increases.
- If $\alpha < 0$, then $|G| > 1$ for some frequencies; hence, it should not be used as a filter.

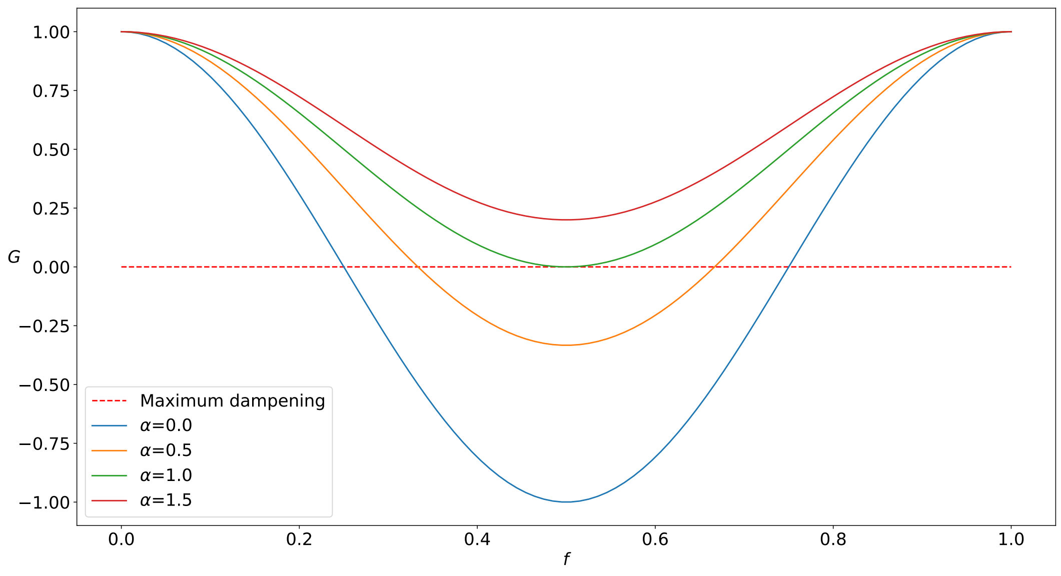

**Figure 2.3.** Periodic gain for different $\alpha$ values.

In conclusion, the $\alpha$ value must be selected to maximise the dampening around the main frequency of the time series (if this frequency is unknown it may be estimated, as it was explained previously, with the expressions (10) to (13)). Therefore, the optimum $\alpha$ for a periodic function is:

$$\alpha = \max(0, -\cos(2\pi f \Delta t)) \tag{25}$$

Even if the $\alpha$ value is constant, it is possible to smooth the time series through an iterative process, due to the *effective gain* after $N$ iterations is $G^N$, which allows to dampen a wide range of frequencies. This behaviour is represented in the Figure 2.4 to Figure 2.6, where it is shown how the maximum dampening area spreads from one or two concrete frequencies defined by the selected $\alpha$ value.

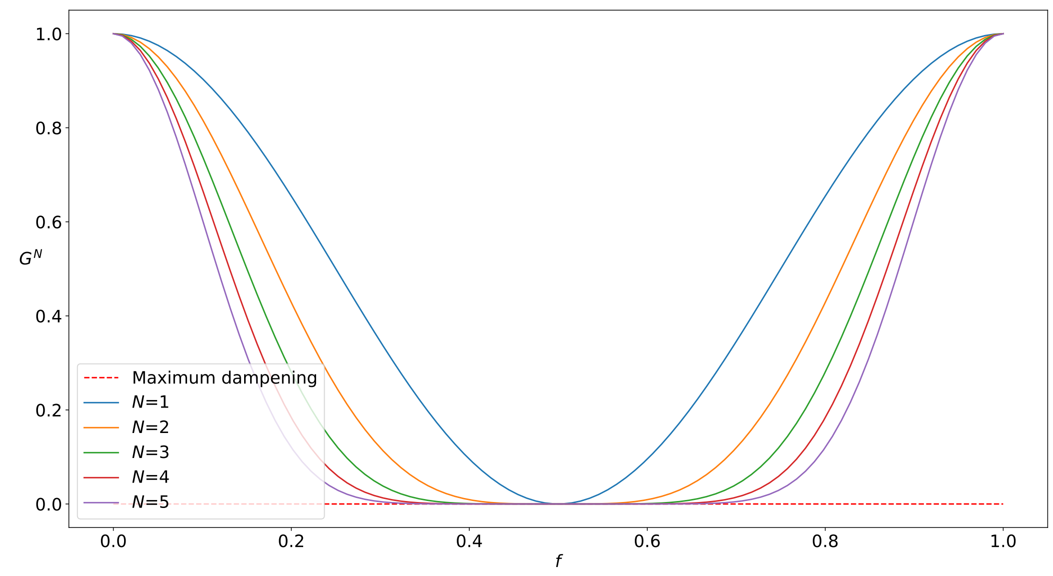

**Figure 2.4.** Periodic gain for $\alpha = 1$ and different number of iterations.

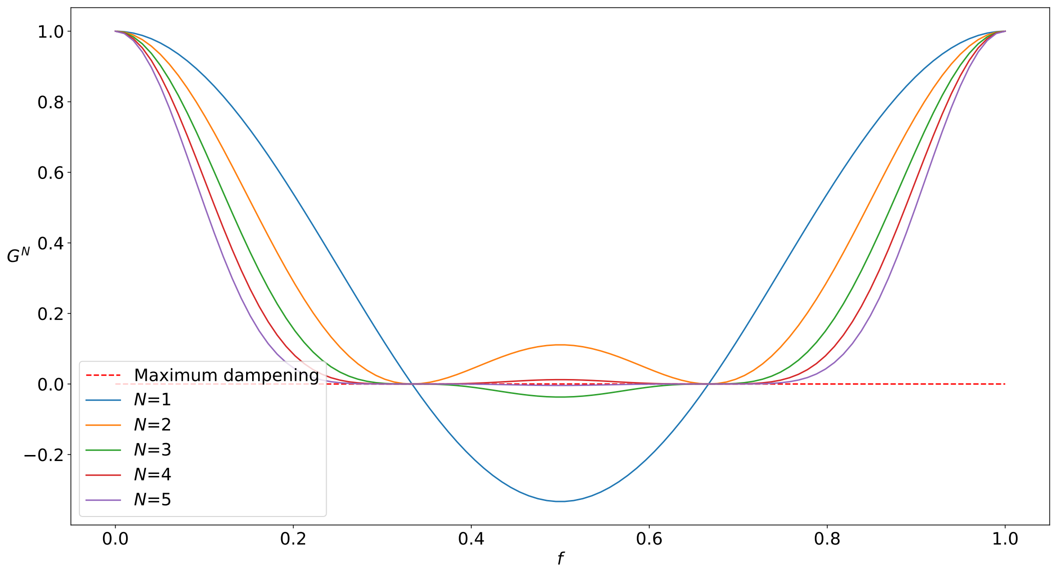

**Figure 2.5.** Periodic gain for $\alpha = 0.5$ and different number of iterations.

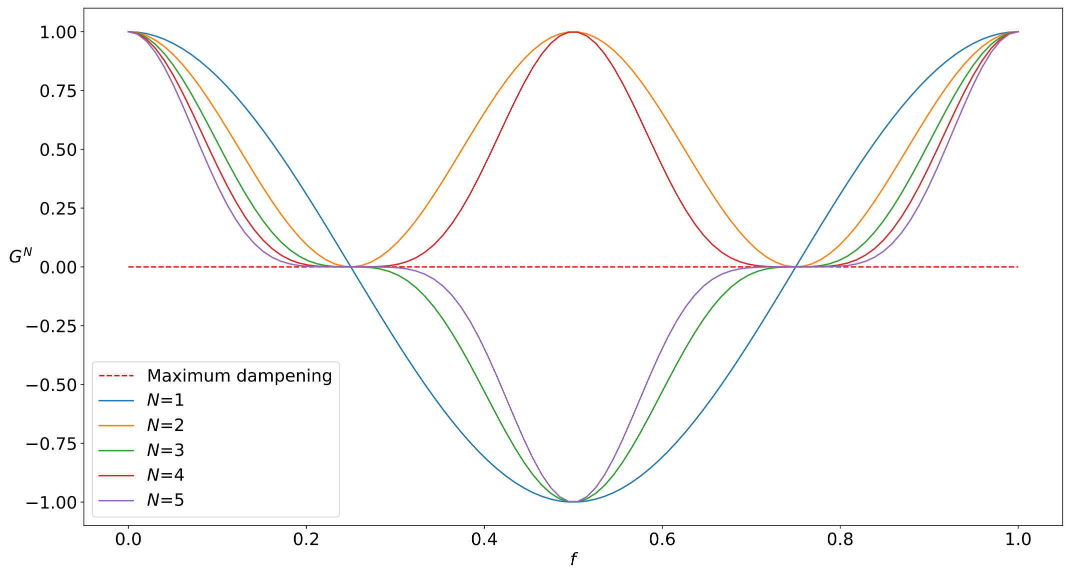

**Figure 2.6.** Periodic gain for $\alpha = 0$ and different number of iterations.

However, in a more general problem with many significant frequencies, it is advisable to change the alpha value to cover a larger section of the spectrum and minimise the number of iterations $N$.

### Polynomial function

Polynomial functions usually change slower in time than periodic functions and they represent the trend of a time series. The general expression of each component in a polynomial function is:

$$T(t) = t^q \tag{26}$$

where $q$ is the degree of the polynomial.

The resulting gain over a sample $X_j$, obtained after the mean value filter is applied, is given by the following expression:

$$G = \frac{1}{2(\alpha+1)} \left[ \left(1 - \frac{\Delta t}{t_j}\right)^q + 2\alpha + \left(1 + \frac{\Delta t}{t_j}\right)^q \right] \tag{27}$$

with $t_j$ as the timestamp associated to the sample $X_j$. As it is shown in the equation (27), the gain depends on the timestamp:

- If $\Delta t \ll t_j$, the gain can be rewritten as:

$$G \approx \frac{1}{2(\alpha+1)}\left[1 - q\frac{\Delta t}{t_j} + 2\alpha + 1 + q\frac{\Delta t}{t_j}\right] = 1 \tag{28}$$

  so the polynomial function remains the same.

- If $\Delta t \sim t_j$, then $\frac{\Delta t}{t_j} = 1 - \varepsilon$, with $\varepsilon \ll 1$. In this case, the gain follows the expression:

$$G \approx \frac{1}{2(\alpha+1)}\left[\varepsilon^q + 2\alpha + (2-\varepsilon)^q\right] \approx \frac{2\alpha + 2^q}{2(\alpha+1)} \tag{29}$$

  The previous equation shows that the first samples (with small timestamps) have a gain greater than 1 if the polynomial order is also greater than 1. Therefore, the filter is unstable and the obtained result may not be adequate in this region. However, if $\Delta t \ll t_c$, with $t_c$ as the characteristic time of the time series, the function can be considered as linear for small timestamps and this instability would be negligible.

### Boundary conditions

The gain obtained in previous sections is only valid for interior points of the time series. Both end points cannot be computed with equation (27).

It is vital to choose adequate boundary conditions in order to preserve the general behaviour of the filter, with the objective of minimising the polynomial instability for small timestamps and maintaining a continuous and soft dampening.

The simplest approach one can think of is to keep the end points constant. Although it may be useful for some specific applications, in general it is not interesting due to the gain is always 1, so these points are never dampened.

Another option is to keep the filter's structure, applied only over two points:

$$\hat{X}_0 = \frac{2\alpha X_0 + X_1}{2\alpha+1} \qquad \hat{X}_N = \frac{2\alpha X_N + X_{N-1}}{2\alpha+1} \tag{30}$$

For these boundary conditions, the resulting periodic gain is:

$$G = \frac{2\alpha + \cos(2\pi f \Delta t)}{2\alpha+1} \tag{31}$$

This gain only matches the general gain (24) if $\alpha = 0$. In other case, there will be a discontinuity in the gain function that may generate worse results in the end points.

On the other hand, the polynomial gain results:

$$G = \frac{1}{2\alpha + 1}\left[2\alpha + \left(1 + \frac{\Delta t}{t_j}\right)^q\right] \tag{32}$$

This equations shows two possibilities:

- If $t_j = t_N$, then $\Delta t \ll t_N$ and $G = 1$, hence the polynomial does not suffer dampening, which is the desired behaviour.
- If $t_j = t_0 = 0$, then $G \to \infty$ and the filter becomes highly unstable around the first point.

As the previous analysis has shown, these easy and intuitive boundary conditions carry with them several problems that are not admissible. Therefore, in this research another method is proposed in order to assure continuity and softness of the filter's properties.

When selecting the boundary conditions, the main purpose is to adjust the gain between end points and interior points. In order to obtain this result, two different and supplementary approaches have been considered:

- To minimise the difference between the variance of the whole time series and that of the interior points only. If the end points do not significantly modify the variance, then they cannot be far from the trend.
- To minimise the difference between the second derivative near the ends with and without the end points. A continuous second derivatives favours the smoothness of the trend.

Mathematically, these two methods can be expressed as:

$$F(X,t) = \sigma^2 - \sigma^{*2} + \left.\frac{\mathrm{d}^2X}{\mathrm{d}t^2}\right|_{t=\Delta t} - \left.\frac{\mathrm{d}^2X}{\mathrm{d}t^2}\right|_{t=2\Delta t} + \left.\frac{\mathrm{d}^2X}{\mathrm{d}t^2}\right|_{t=t_f-\Delta t} - \left.\frac{\mathrm{d}^2X}{\mathrm{d}t^2}\right|_{t=t_f-2\Delta t} \tag{33}$$

where $\sigma^2$ is the variance of the whole time series, $\sigma^{*2}$ is the variance of the interior points and $t_f$ is the last timestamp of the time series.

Therefore, the boundary condition problem has been reduced to an optimization problem, where the objective is to minimise the function $F(X,t)$. Though these boundary conditions have proven to be very effective, they are computationally expensive and they will only be used when it is absolutely necessary.

### 2.2.3 The decomposition procedure

Once the *Mean Value Filter* has been defined and its gain has been analysed, it is time to explain the procedure used for decomposing the time series.

First, the trend component will be obtained, by applying the filter, and then the seasonal component and the residuals will be separated.

#### Trend component

The original time series contains high frequency harmonics which difficult the task of identifying the trend. These harmonics will be removed with the *Mean Value Filter*, which dampens, after several iterations, every undesirable frequency.

The first step is to apply the filter $N$ times, using as boundary conditions those obtained by minimising the function $F$, defined in the equation (33), in each iteration. Also, in each iteration, the $\alpha$ value switches between 1 and 2. The reason of these $\alpha$ values are, first, to dampen intermediate frequencies, $f \approx \frac{1}{2\Delta t}$, as $\alpha \geq 1$; and also, to assure an uniform smoothness by switching between both values. In this research, a number of iterations $N = 2.5P$ has been selected, with $P$ as the main period of the time series, measured in number of samples. This boundary conditions have to be applied in both end points and also in both contiguous points. The result after this operation is the new time series $T_1$.

Then, it is necessary to make some corrections near the end points due to the boundary conditions, even after solving the optimization problem, are not perfect. These correction tasks are based on a linear interpolation. A number of samples $k = \min(4P,\, n_s/4)$ is selected, with $n_s$ as the number of samples in the time series. The first $k/2$ points of $T_2$ are approximated by a line:

$$T_{2,i} = \text{mean}(T_1(1:k)) - \text{mean}\left(\frac{\mathrm{d}T_1}{\mathrm{d}t}(k/2:k)\right)(k/2 - i) \qquad \text{for} \quad i = 1,\, 2,\, ...,\, k/2 \tag{34}$$

The same operation is applied over the last $k/2$ points of the time series. The rest of the interior points remain the same. The resulting time series is called $T_2$.

Finally, the filter has to be applied again, $M = 95P$ times, to smooth the trend. This time, the end points have already been corrected and it is not necessary to solve the optimization problem again. In this case, the optimum $\alpha$ value (see expression (25)) will be used for every iteration. This $\alpha$ value is justified due to, most of times, $\alpha_{opt} \approx 0$; hence, as it was shown in the Figure 2.6, the dampening is concentrated near low and high frequencies only, as intermediate frequencies have already been removed in previous steps.

The end points will be extrapolated with the following expression:

$$Y_1 = 2Y_2 - Y_3 \qquad\qquad Y_{n_s} = 2Y_{n_s-1} - Y_{n_s-2} \tag{35}$$

where $Y_i$ is the time series after each pass of the filter.

In this step, a convergence criterion has been defined to stop the iterative process if the trend, $T$, has been obtained successfully:

$$\max\left(\left|T^i - T^{i-1}\right|\right)\sigma^2 < \left(\max\left(T^i\right) - \min\left(T^i\right)\right)\cdot 10^{-7} \tag{36}$$

where $T^i$ and $T^{i-1}$ are two consecutive trends after one pass of the filter and $\sigma^2$ is the variance. The resulting time series, $T(t)$, is the trend component.

### Seasonal component

The starting point is the detrended time series, $X^*(t)$, as it was shown for the STL method (see equation (7)). The filter is applied 3 times over the detrended time series, using simple extrapolated boundary conditions, the same used in the expression (35). Note that this step may be avoided if the period is small; in this case, the filter is applied only if each period contains more than 20 samples. The resulting time series is called $S_1$.

Then, the *Fourier Transform* is used to analyse $S_1$ in the frequency domain. A threshold $th = 0.02\max(|\mathcal{F}[S_1(t)]|)$ is defined, and each point of $\mathcal{F}[S_1(t)]$ with an amplitude inferior to that threshold is switched for a zero, which allows to remove the less important frequencies. After this process, the time series has to be returned to the time domain through the *Inverse Fourier Transform*, obtaining $S_2$.

Even though $S_2$ is not a bad approximation of the seasonal component, it should be refined due to the previous step may have overlooked some relevant frequencies.

To improve the seasonal component, the first step is to deseasonalize the time series by subtracting $S_2$ from $X^*(t)$, operation which results in $X^{**}(t)$. After that, the previous step is redone, but this time the filter is applied 5 times and the new threshold is limited to $th = 0.005\max(|\mathcal{F}[X^{**}(t)]|)$. With this new criterion, the negligible frequencies are removed and the *Inverse Fourier Transform* returns $S_3$.

Finally, the seasonal component is obtained as:

$$S(t) = S_2(t) + S_3(t) \tag{37}$$

### Residuals

Once the trend and the seasonal components have been obtained, the residuals can be easily calculated, as they represent the noise and other perturbations of the time series:

$$R(t) = X(t) - T(t) - S(t) \tag{38}$$

### Comparison of magnitude orders

Sometimes, once the decomposition is completed, one component is insignificant compared to the others. In these cases, it is advisable to regroup this small component, $Y$, with a bigger one, $Z$.

In this research, the selected criterion is to regroup two components if the maximum value of $Y$ is less than 0.005 times the mean of $Z^*$, which includes the biggest $0.1n_s$ values of $Z$, with $n_s$ as the time series length. Mathematically, the regroup criterion can be expressed as:

$$max(Y(t)) < 0.005\, mean(Z^*) \tag{39}$$

# 3 Anomaly detection

Time series are functions of time which represent the evolution of some variables. Though these functions can be completely chaotic, it is usual to find concrete patterns in real life time series. In particular, when the variables are signals generated by space systems, such as telemetry data from a satellite, these patterns are highly related to the mission parameters (e.g. application orbit, exposure to Sun radiation, etc.). Therefore, time series in space applications are expected to have a certain degree of periodicity, due to the periodicity of orbits. Even if some signals are not periodic, during the operational phase most of the variables should remain stable around equilibrium points to assure the correct functioning of the system.

An anomaly can be defined as any variation of a time series that deviates from the nominal or expected behaviour. The underlying difficulty behind this concept is that some events can be anomalous or not depending on the context, so different techniques should be used to verify the anomaly.

The main types of anomalies may be classified as follows [5]:

1. Outlier: Data point whose value significantly exceeds the mean of the time series, in absolute value, as it is shown in Figure 3.1. It is a time-independent anomaly. In this research, an outlier will be considered as an anomaly if it deviates from the mean $\pm 3\sigma$, with $\sigma$ as the standard deviation.

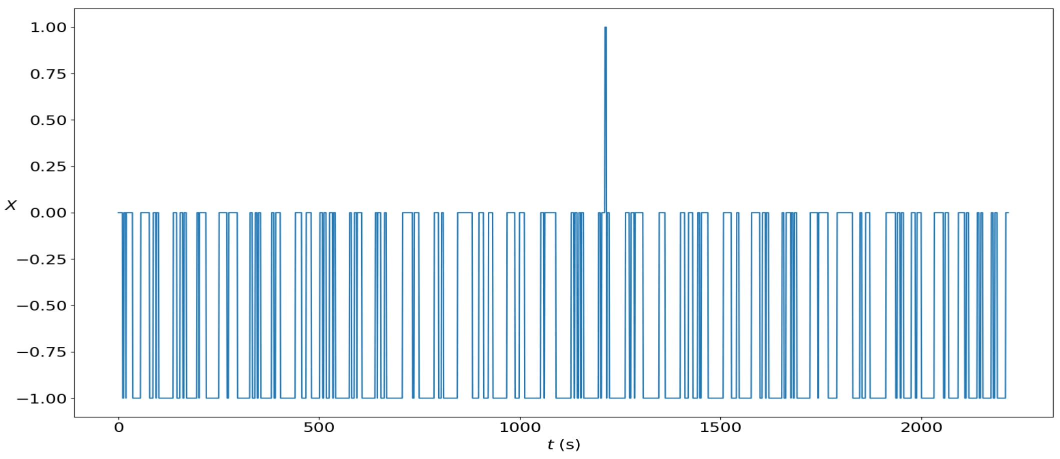

**Figure 3.1.** Outlier anomaly from NASA telemetry signal of the Soil Moisture Active Passive satellite (SMAP) and the Curiosity Rover on Mars (MSL) [2].

2. Spike: Data point whose value significantly exceeds the local mean of the time series, in absolute value, as it is represented in Figure 3.2. It is a time-dependent anomaly.

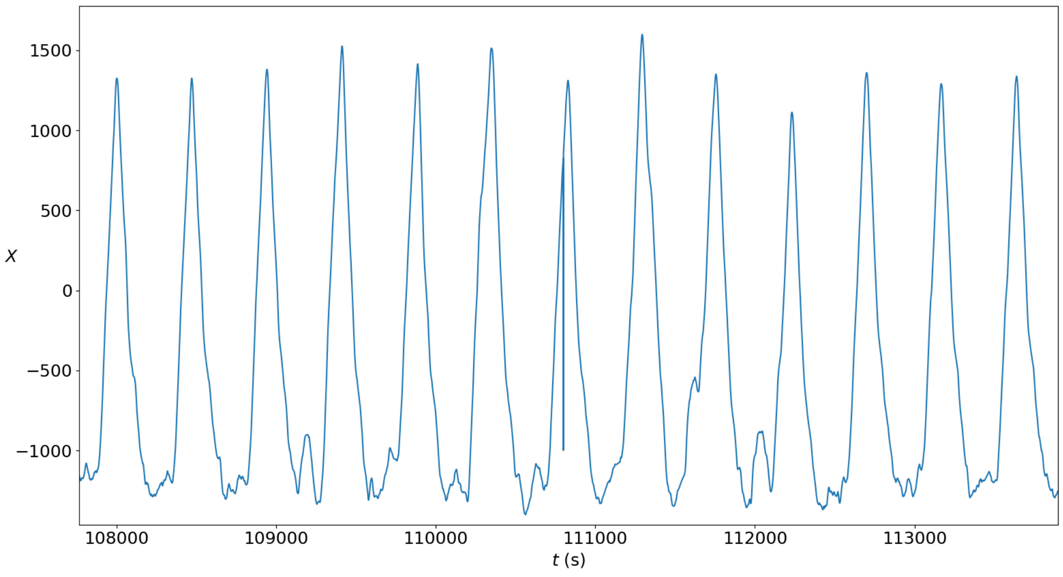

**Figure 3.2.** Spike anomaly in a respiration dataset with regular breath cycles [1].

3. Level shift: It happens when the mean, computed before and after a certain point, changes drastically (see Figure 3.3).

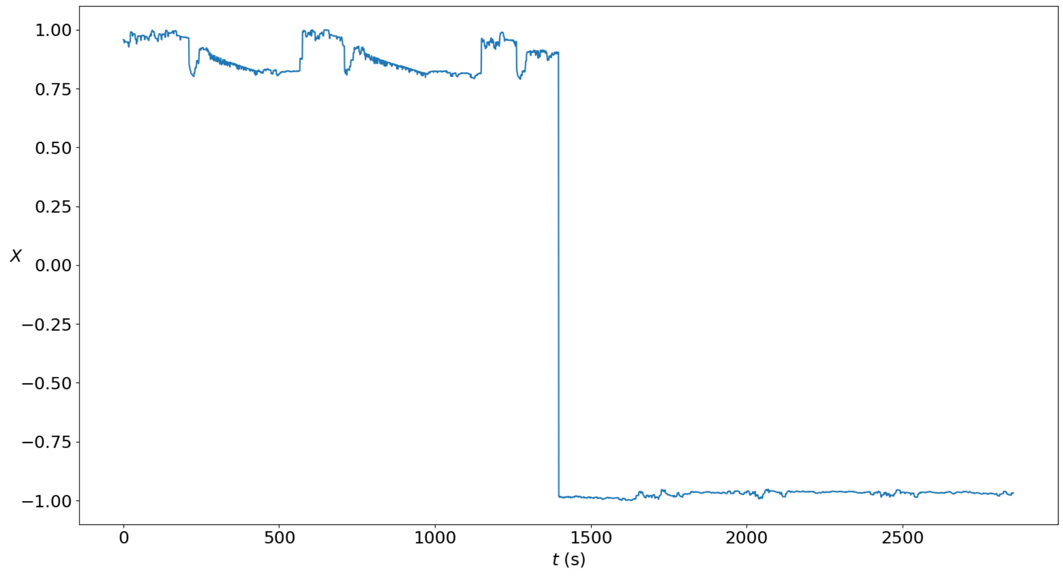

**Figure 3.3.** Level shift anomaly from NASA telemetry signal of the Soil Moisture Active Passive satellite (SMAP) and the Curiosity Rover on Mars (MSL) [2].

4. Volatility shift: It happens when the standard deviation, computed before and after a certain point, changes drastically (see Figure 3.4).

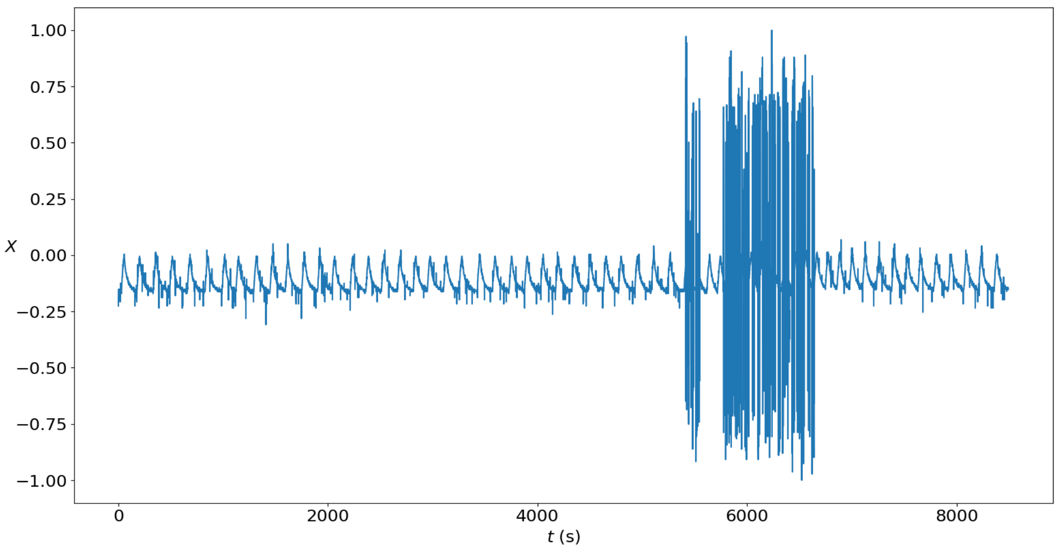

**Figure 3.4.** Volatility shift anomaly from NASA telemetry signal of the Soil Moisture Active Passive satellite (SMAP) and the Curiosity Rover on Mars (MSL) [2].

5. Anomalous seasonal pattern: Interval where the time series does not follow the seasonal pattern, as it is shown in Figure 3.5.

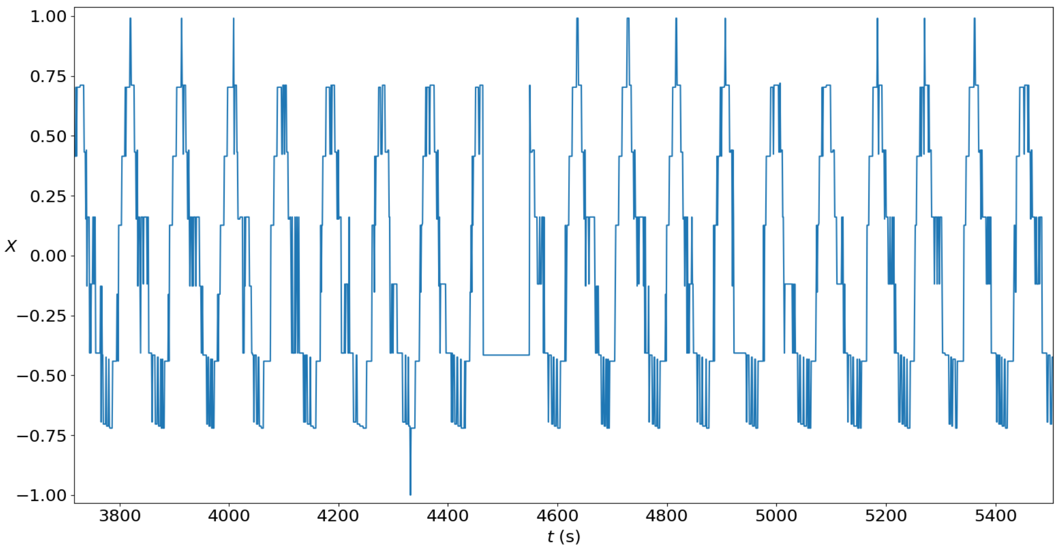

**Figure 3.5.** Seasonal pattern anomaly from NASA telemetry signal of the Soil Moisture Active Passive satellite (SMAP) and the Curiosity Rover on Mars (MSL) [2].

A generic time series may contain one or more types of anomalies, that have to be detected in order to correct the normal activity of the system.

The classical approach for anomaly detection in space systems [6] consists of several alarms that alert the engineering team if any of the variables surpasses the pre-defined limits. However, this method may be ineffective in some situations and extensive expert knowledge is needed.

This is the reason why, during the last years, a great effort has been made to develop reliable algorithms with the capability of detecting and classifying time series anomalies. There are two big families of models for anomaly detection in time series:

- Supervised: The model is trained with labeled anomalies that the algorithm learns how to identify. However, the real time series must be also labeled in order to detect the anomaly correctly.
- Unsupervised: The model detects and classifies the time series based on the data of the time series and does not require labeled data.

In general, it is difficult to know beforehand any information regarding the anomaly and it may be impossible to label the time series. Therefore, in this research the unsupervised option will be used for anomaly detection.

In order to detect the anomalies, it will be used the approach suggested in [5], where it is proposed to transform the time series to other domains where different types of anomalies are highlighted. Then, in the new domain, it is easier to detect any anomaly with a simple detector, as it could be a threshold.

## 3.1 Transformers

As it was explained before, there are certain domains (such as time domain or spectral domain) where anomalies are highlighted and, hence, they are detected with ease. The responsible of changing the domain are called *transformers*. They are functions which may transform the current domain when applied to the time series. There are many types of transformers with different properties depending on their concrete application.

### 3.1.1 Spectral residual

This algorithm, based on the *Fourier Transform*, extracts the residuals of the time series in the spectral domain. Then, through the *Inverse Fourier Transform*, a new signal is obtained in the time domain where some anomalies become more significant. Mathematically, this transformation is computed with the following expressions [7]:

$$A(f) = Amplitude\left(\mathcal{F}[X(t)]\right) \tag{40}$$

$$P(f) = Phase\left(\mathcal{F}[X(t)]\right) \tag{41}$$

$$L(f) = \ln\left(A(f)\right) \tag{42}$$

$$AL(f) = L(f) * h_q \tag{43}$$

$$R(f) = L(f) - AL(f) \tag{44}$$

$$SR(t) = \left\|\mathcal{F}^{-1}[\exp\left(R(f) + iP(f)\right)]\right\| \tag{45}$$

where $*$ is the convolutional operator and $h_q$ is a vector of length $q$ and constant value $1/q$. The value of $q$ has to be selected for each application, but $q = 3$ works well in most of cases. It is recommended to select an odd number as the length of $h_q$ to assure the convolution is locally centered.

In general, this method is efficient to detect amplitude related anomalies, due to the phase remains the same after the transformation. The effectiveness of the spectral residual transformer is clearly shown in Figure 3.6, where the spike of the original time series has been transformed into a huge outlier, much easier to be detected.

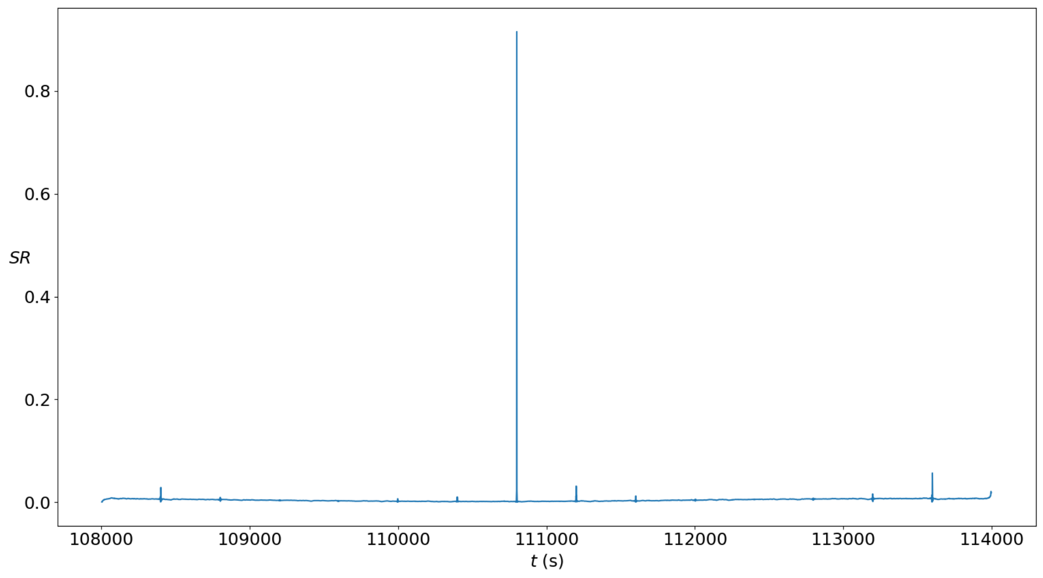

**Figure 3.6.** Spectral residual transformation of the time series shown in Figure 3.2, which has a spike anomaly.

### 3.1.2 Double rolling aggregate

This method [5] consists of two sliding windows rolling side by side (from left to right and from right to left) along the time series. In each step, pre-defined metrics are computed within each sliding window. After the whole time series has been traversed, the transformed data are obtained as the difference between metrics of both sliding windows.

Depending on the selected metrics, a different type of anomaly is highlighted. For example, if the metric is the median, level shift anomalies are much easier to be detected, as it is shown in Figure 3.7; and if the metric is the quantile, volatility shift anomalies are emphasized, as it is represented in Figure 3.8. In both cases, the anomalies are easily recognizable after the transformation.

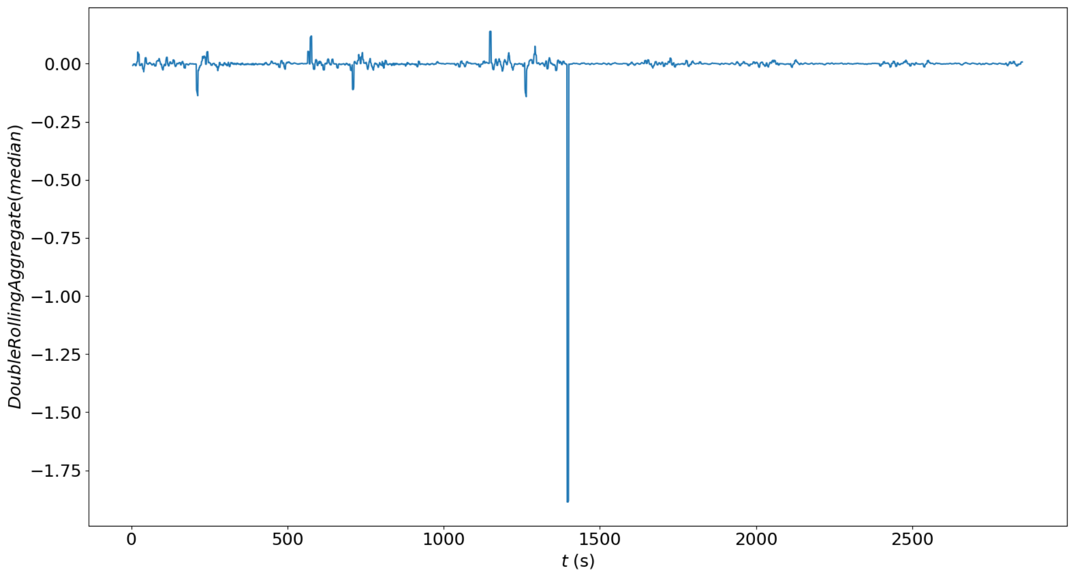

**Figure 3.7.** Double rolling aggregate, based on the median, transformation of the time series shown in Figure 3.3, which has a level shift anomaly.

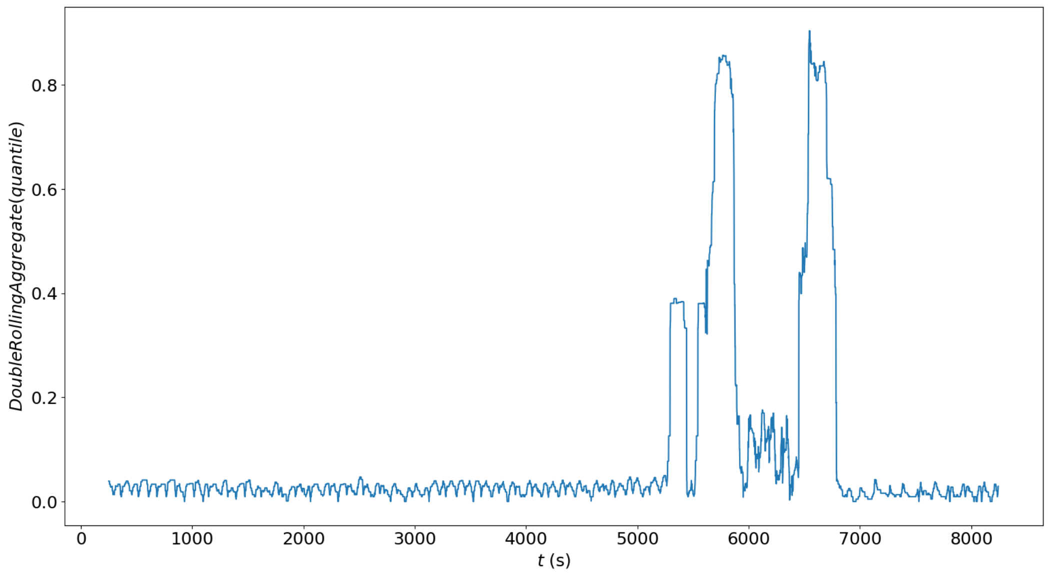

**Figure 3.8.** Double rolling aggregate, based on the quantile, transformation of the time series shown in Figure 3.4, which has a volatility shift anomaly.

## 3.2 Detectors

Once the time series has been transformed, it is necessary to detect the anomalies using a *detector*, which allows to discriminate between anomalous and normal data points. After the transformation, the different anomalies will look like outliers; therefore, they can be detected with a simple threshold of $\pm 3\sigma$ around the mean.

Though this is the simplest approach, some researches [7] have tried to use CNN (Convolutional Neural Networks) as a detector to identify these anomalies. However, due to the limited available time in this project, the first approach will be used as the chosen detector in the following analysis.

## 3.3 Anomaly detection procedure

The main goal is to develop an algorithm that is able to detect every single anomaly present in a time series. However, the use of a single transformer and detector is inefficient, due to some anomalies would be overlooked.

To overcome this problem, a new approach is applied in this research: time series are first decomposed, then transformed and, finally, the anomalies are detected. This method is quite effective with MVD decomposition, but computationally expensive and time consuming; hence, it is necessary to reach an equilibrium between accuracy and time.

The steps to follow to obtain an adecuate solution are detailed bellow:

1. The original time series is decomposed in trend, seasonal component and residuals, using STL decomposition. This decomposition is less accurate than MVD, but faster and precise enough for a first approach.

2. Then the original time series is transformed through the spectral residual transformer. After this transformation, there are 5 time series: the original one, 3 associated with the decomposition procedure and the spectral residual transformation.

3. The next step is to detect anomalies in the 5 time series. *Outliers* will be detected with a $\pm 3\sigma$ threshold; *spikes* will become outliers in the spectral residual and the residuals components and can be detected in a similar way; *level shifts* will be detected combining the rolling double aggregate transformer (based on the median) and a threshold detector;

*volatility shifts* will be detected combining the rolling double aggregate transformer (based on the quantile) and a threshold detector; and any *anomalous seasonal pattern* will be detected in the seasonal component or in the spectral residual, as one of the previous anomalies.

4. After the previous step, there will be a lot of possible anomalies, and most of them are not real. Therefore, a refinement process is needed to discriminate between real and false anomalies. In order to reduce the computational time, specially for long time series, the original time series is split and only some points around each anomaly are preserved (an appropriate value is $\pm l/50$ points, with $l$ as the whole length of the original time series). This procedure allows to significantly reduce the length of the time series.

5. With the new reduced time series, steps 1) to 3) are repeated, but this time the method used in the decomposition is MVD.

6. Finally, a weight is given to each anomaly depending on how many methods detect the same anomaly. The anomalies with a higher weight are called *major*, the next ones with a higher weight are *significant* and the rest of anomalies are *minor*.

To test the functioning of this algorithm, it will be tested with different data, where the anomalies have already been detected by experts in the field.

In order to measure in a quantitative way the accuracy of the algorithm, for each time series of the database, if every anomaly is detected as major, 3 points are given; if it is detected as significant, only 2 points are given; if it is detected as minor, then 1 point is given; and if the anomalies are not detected at all, then 0 points are given. This numerical scale allows to evaluate the performance of the anomaly detection method.

# 4 Results

Once the different methods for time series analysis have been identified and the anomaly detection procedure is completely defined, some results have been obtained by applying these techniques to real time series. Furthermore, the proposed methodology has been tested with two databases from UCR [1] and NASA [2].

## 4.1 Noise reduction

Before the analysis of any time series can be carried out, it is necessary to reduce the noise of the signal. The noise reduction task allows to obtain a clearer time series and makes it easier to detect anomalies and extract conclusions.

The Figure 4.1 shows a time series where the noise is a significant part of the signal. In order to remove the existing noise, the *Mean Value Filter* described in section 2.2.1 is applied 10 times. The resulting time series is shown in Figure 4.2. As it can be observed, after applying the filter only 10 times, despite the high power of the noise, it has significantly decreased and the new time series will be easier to be analysed with other methods.

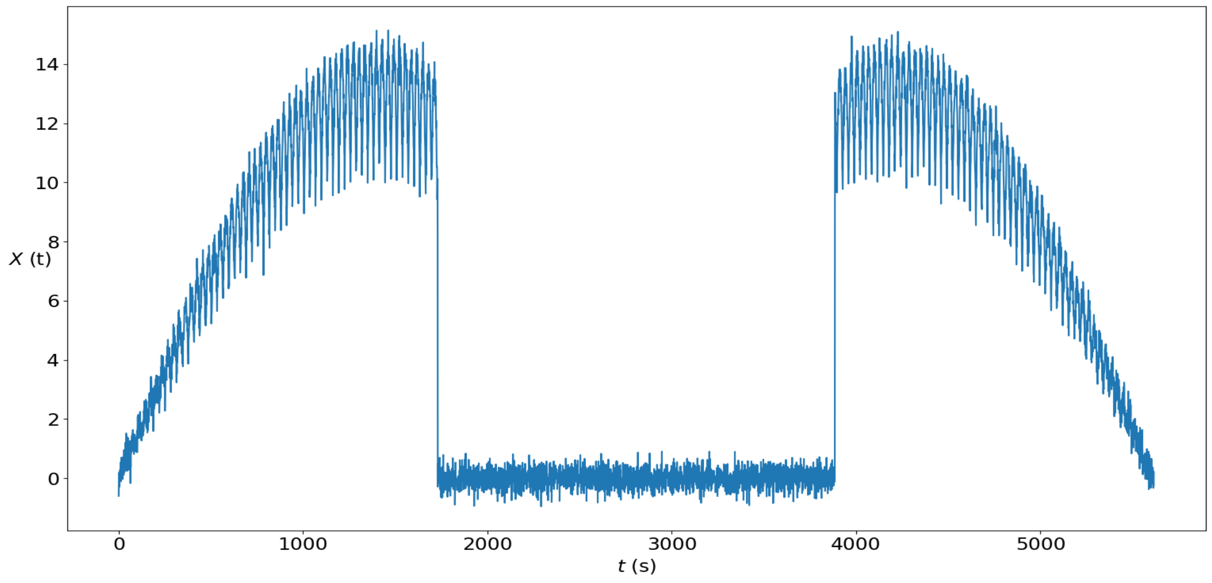

**Figure 4.1.** Power obtained from the solar cells of a spinning satellite.

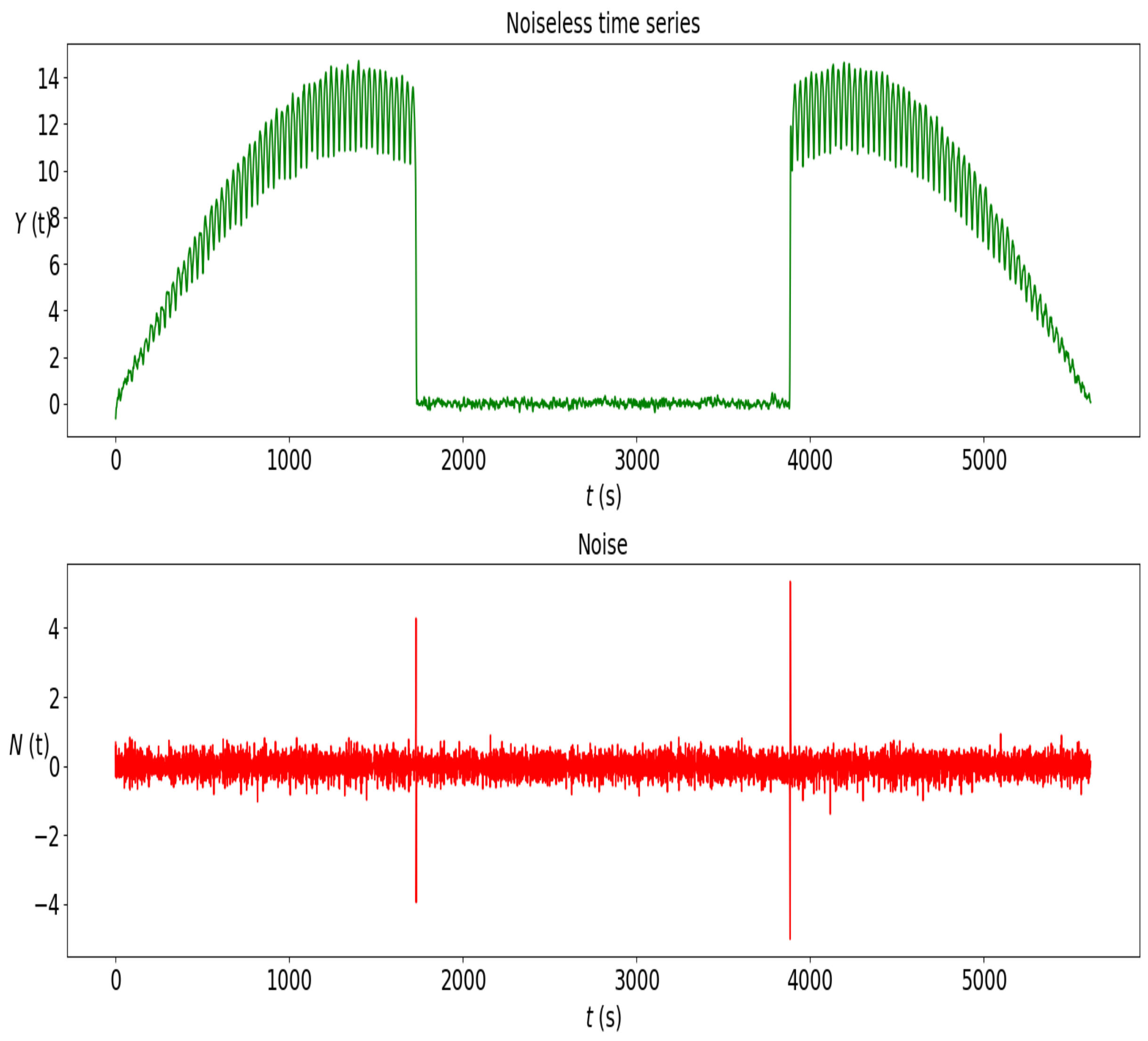

**Figure 4.2.** Noiseless time series (up) and the removed noise (down).

## 4.2 Time series decomposition

As it was explained in section 2, time series decomposition is a methodology which allows to separate a time series into different sub-series. This approach helps to understand the underlying meaning behind the time series and it facilitates the following time series analysis and the anomaly detection task.

In order to apply correctly the decomposition procedure, it is important to know an appropriate estimation of the period, as it it is an indicator of how many times the filter should be applied.

For this analysis, the starting point is the time series shown in Figure 4.3. This time series is very difficult to be analysed by visual inspection due to the high noise power and the strong fluctuations of the signal. The decomposition procedure looks for a deeper understanding of the time series by dividing the information into simpler elements.

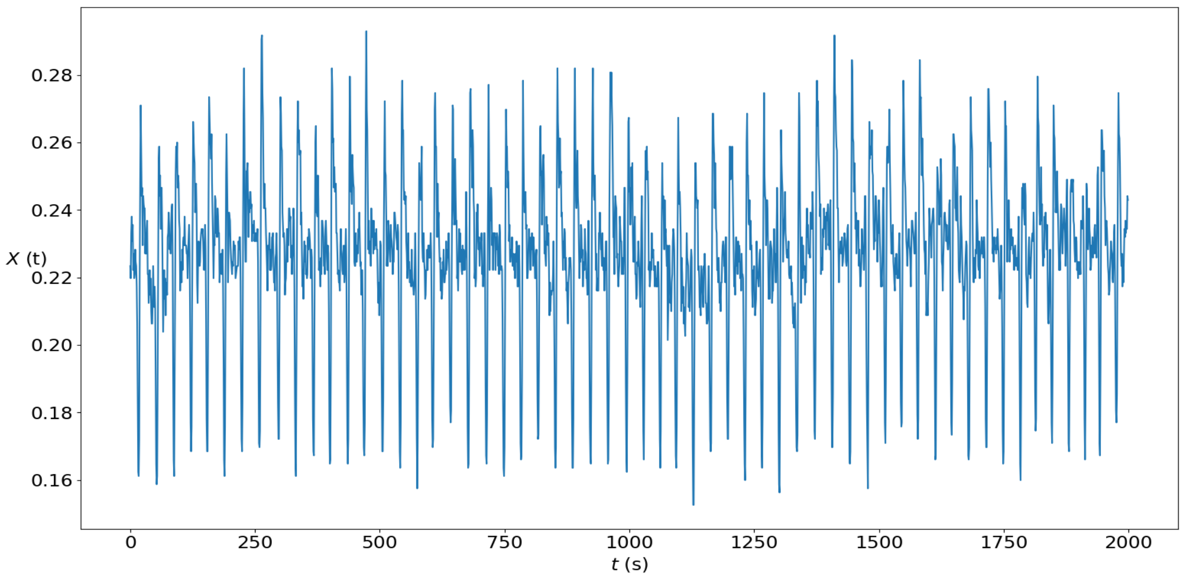

**Figure 4.3.** Real time series from UCR's database [1].

In the first place, the STL decomposition has been applied, following the steps previously described in section 2.1. The results obtained after the application of this method are shown in Figure 4.4. After the decomposition has been carried out, the residuals of the time series has been separated, which makes it easier to focus on the relevant information.

Though this decomposition has proven to be very useful, there are some details that need to be polished:

1. The trend contains irregular oscillations, that should be included within the seasonal component. Furthermore, the trend is not as smooth as it was expected from the main component of the time series.

2. The seasonal component shows a pattern that varies with time and fluctuates, which is undesirable.

3. The irregular variations from the residuals, corresponding to the blue line (the orange line is the noise), are too significant and they may be taking information that should be reflected in the trend or the seasonal component.

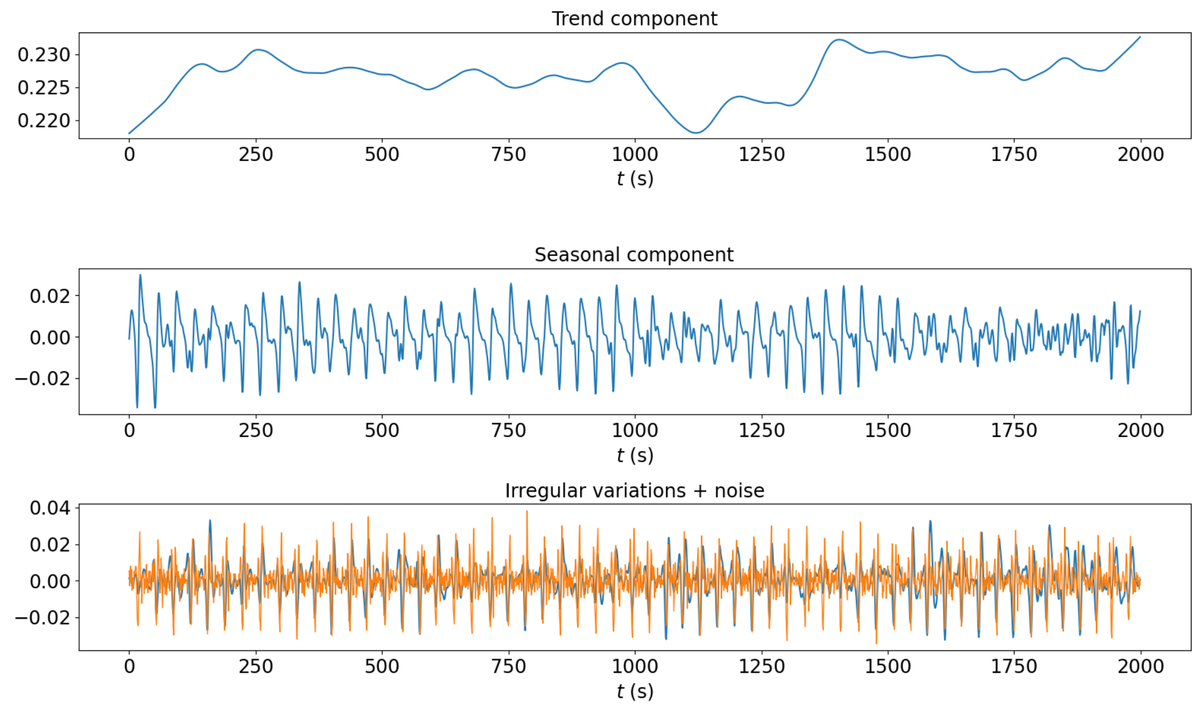

**Figure 4.4.** STL decomposition of the time series shown in Figure 4.3.

In order to solve these issues, the time series is decomposed with the MVD procedure, that has been developed in this research, as it was explained in section 2.2. The decomposed time series using MVD is represented in Figure 4.5.

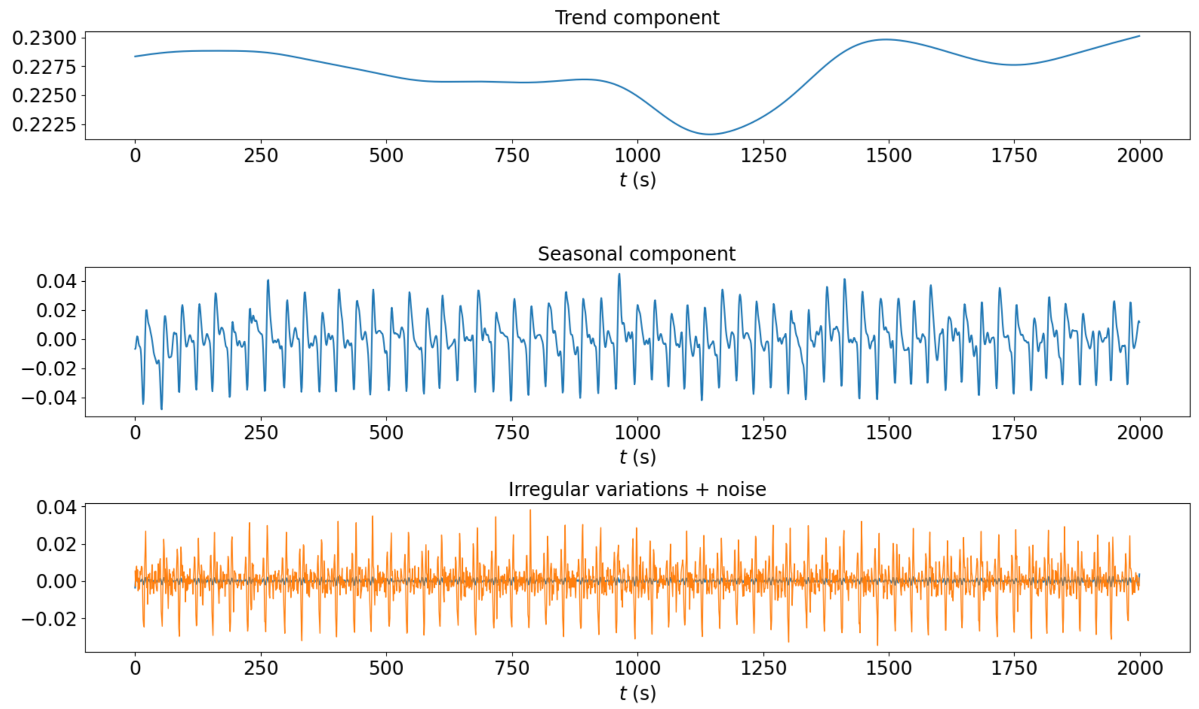

**Figure 4.5.** MVD decomposition of the time series shown in Figure 4.3.

In this case, the trend is smoother than in the previous decomposition. Besides, the seasonal component shows a regular pattern that maintains its shape every cycle. Finally, the irregular variations from the residuals are less significant than in the previous case, which indicates that the main information is kept within the trend and the seasonal components, which is the expected behaviour.

All these reasons may indicate that the MVD procedure is more robust and reliable than STL for time series decomposition. As a disadvantage, this method is more expensive in terms of computational time.

## 4.3 Anomaly detection

The whole methodology explained before can be gathered together, as it was described in section 3.3, to detect anomalies in a robust and reliable way. The use of several techniques at the same time lets to assure the presence of an anomaly that could be overlooked if only one method was used.

In order to show the effectiveness of the proposed procedure, the anomalous time series shown in Figure 4.6 will be analysed. The anomaly, which is located around $t = 4190\,\mathrm{s}$, is very subtle and difficult to be detected, and it represents a region where the local fluctuations are stronger than they should be.

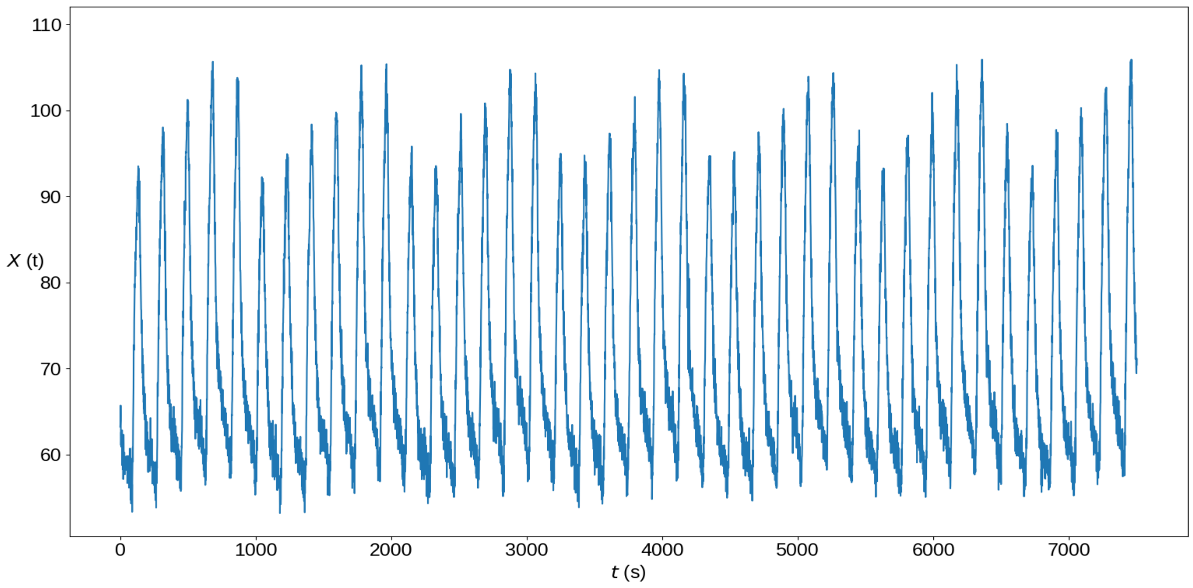

**Figure 4.6.** Anomalous time series from UCR's database [1].

On the one hand, the use of the transformer *double rolling aggregate*, explained in section 3.1.2, and a simple threshold based detector is not enough to detect the anomaly.

On the other hand, the *spectral residual* transformation (see section 3.1.1) shows a better performance in this situation, as it is shown in Figure 4.7. Although the anomaly has been emphasised, there are several outliers that could be detected as possible anomalies too. Therefore, the use of other methods is still necessary in order to verify the supposed anomaly.

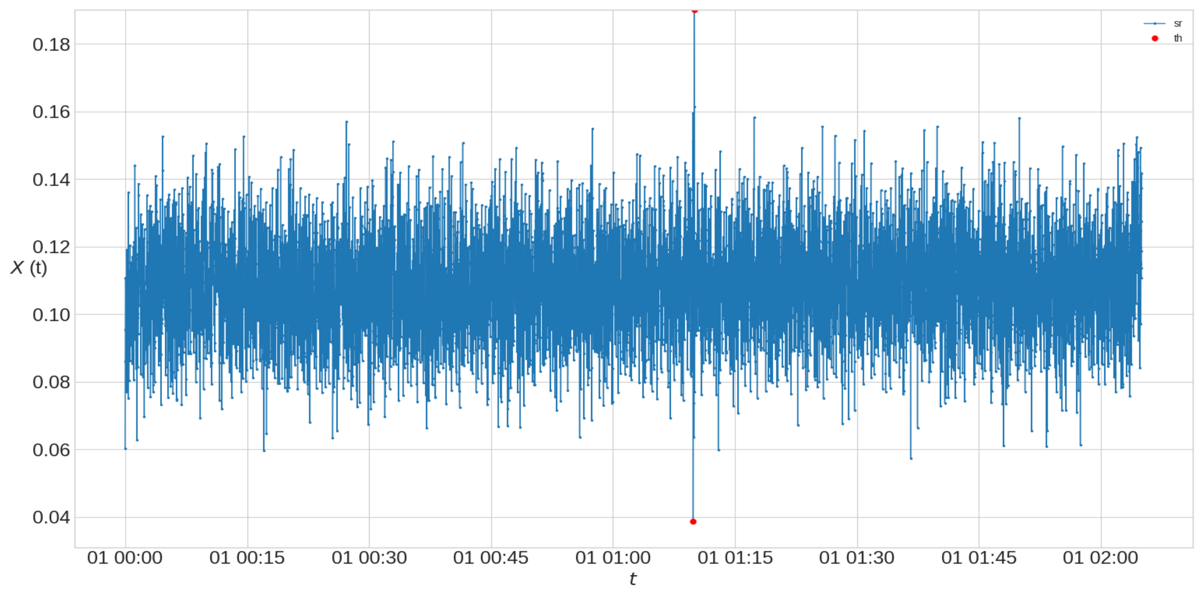

**Figure 4.7.** Spectral residual transformation of the time series shown in Figure 4.6.

Noise reduction and time series decomposition offer additional information about the time series and facilitates the anomaly detection task. The maximum reliability is obtained by using all the previous methods together. This procedure was explained in section 3.3 and the results are better than any of the methods could obtain individually.

The application of this procedure to the previous time series is shown in Figure 4.8. In this case, the whole methodology has allowed to detect the anomaly unambiguously, even if the anomaly was very subtle and almost impossible to be found through a visual inspection.

This result shows the effectiveness of the proposed approach, which has demonstrated to be a precise anomaly detection algorithm.

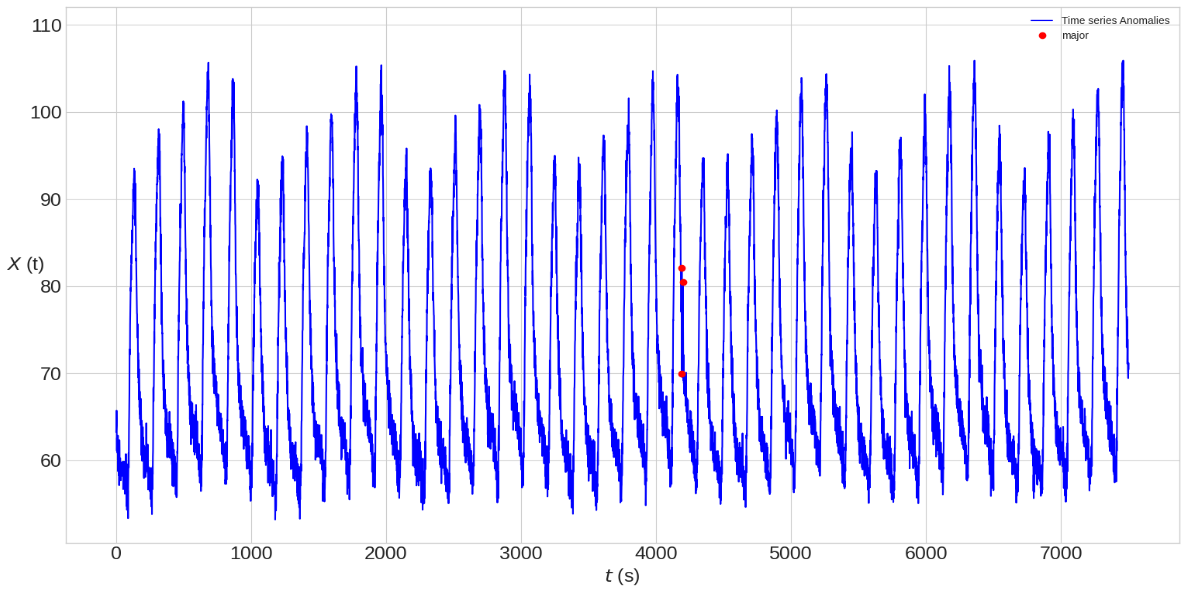

**Figure 4.8.** Anomaly detection procedure applied to the time series shown in Figure 4.6.

## 4.4 Computational time

The procedure explained before shows an excellent performance, but it is very expensive in terms of computational time, which makes it less competitive in comparison with other methods.

The whole algorithm was initially programmed in Python, but the time needed to perform every task was excessive. This fact made it challenging to test this procedure with large databases. The proposed solution to solve this problem was to implement the code in Fortran, due to a compiled Fortran programme runs faster than the same programme in Python.

In order to evaluate the performance of both options, a time series decomposition was carried out for different number of iterations with the selected filter. The result is represented in Figure 4.9, where it is clear that the Fortran code significantly exceeds the performance of the Python code and allows to speed-up each analysis almost 400 times.

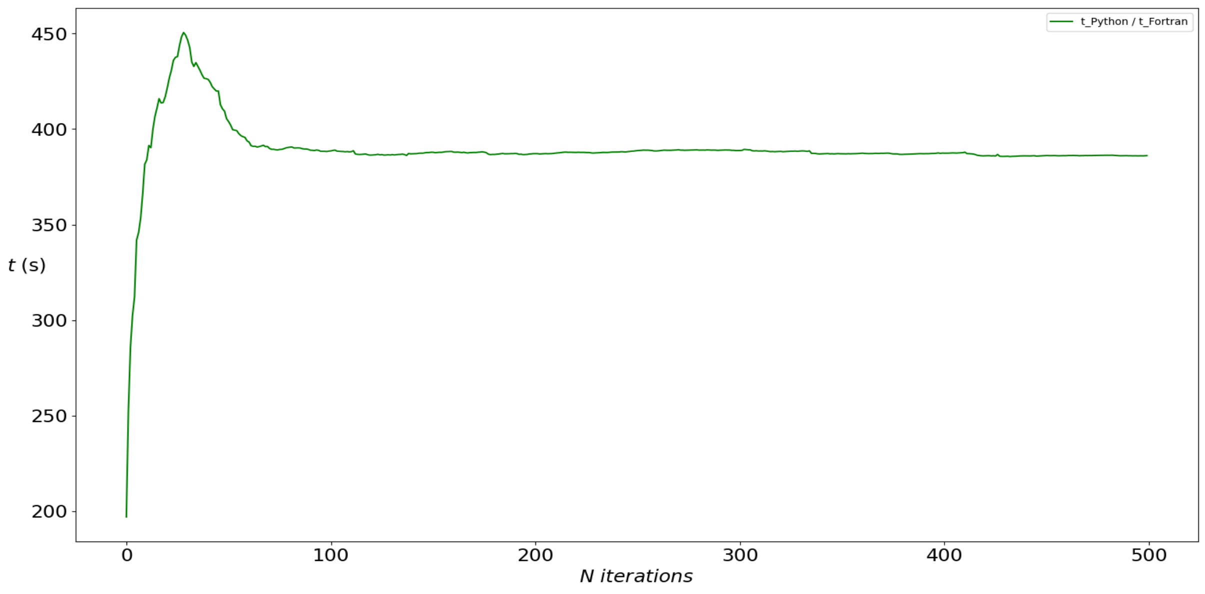

**Figure 4.9.** Quotient between Python and Fortran computational time.

However, Python programming language is very useful due to the several libraries available and the easiness for plotting the results. The optimum solution would be to synchronise Python and Fortran languages, which would let to use Fortran only for computationally expensive tasks.

In order to perform this synchronisation, a wrapper from Fortran to Python, that we developed in a previous research, has been used and it is available in a Github repository [8]. This wrapper allows to create an interface between Fortran and Python; hence, a Fortran module can be called from Python as any other library.

This choice makes the proposed method more competitive, by reducing the time needed for computation. Furthermore, a reduction in time eases the task of testing the algorithm with different anomaly detection benchmarks.

## 4.5 Benchmarks for algorithm testing

Finally, it is important to test the developed algorithm with different examples of time series with labeled anomalies. In this case, the anomaly detection method will be tested with two benchmarks from UCR [1] and NASA [2].

In order to measure the performance of the proposed procedure, a numerical scale has been defined, as it was explained in section 3.3, where an anomaly classified as *major* obtains the maximum score, a *significant* anomaly an intermediate score, a *minor* anomaly the lowest score and an undetected anomaly obtains 0 points.

NASA's database contains 81 time series, which represent telemetry signals from the Soil Moisture Active Passive satellite (SMAP) and the Curiosity Rover on Mars (MSL). The results are shown in Table 4.1, where it is represented the whole performance, the classification of the different anomalies and how many anomalies have been detected successfully by each method. The global result is 8.9/10 which is an excellent performance, with only two undetected anomalies. Furthermore, the best method for anomaly detection has proven to be the *spectral residual transformation*. However, the other analysis help to validate the identified anomalies.

**Table 4.1.** Result of NASA's benchmark analysis.

| **Result** | **Major** | **Significant** | **Minor** | **Undetected** |
|---|---|---|---|---|
| 8.9/10 | 65/81 | 12/81 | 2/81 | 2/81 |
| **Transformed time series** | **Spectral residual** | **Trend** | **Seasonal** | **Residuals** |
| 66/81 | 77/81 | 32/81 | 51/81 | 75/81 |

Finally, UCR's database contains 241 anomalies from different sources. However, these anomalies are extremely subtle and most of them almost impossible to be detected. The results obtained after the evaluation of the algorithm in this benchmark are shown below in Table 4.2. The result of 5.4/10 is significantly lower than the one obtained in NASA's benchmark, but more than half of the anomalies have been successfully detected. Again, the best method has been the *spectral residual transformation.*

**Table 4.2.** Result of UCR's benchmark analysis.

| **Result** | **Major** | **Significant** | **Minor** | **Undetected** |
|---|---|---|---|---|
| 5.4/10 | 96/241 | 52/241 | 2/241 | 91/241 |
| **Transformed time series** | **Spectral residual** | **Trend** | **Seasonal** | **Residuals** |
| 103/241 | 152/241 | 18/241 | 74/241 | 136/241 |

After the previous analysis has been performed, it is clear that the methodology described in this research is a competitive algorithm for anomaly detection in time series.

Furthermore, the excellent results obtained, specially in NASA's benchmark, show the possibility of using computational based techniques for time series analysis in space applications. The developed algorithm could perform the task of monitoring telemetry signals and the identification of possible anomalies in almost real time, which would let to reduce time and costs of the space project.

# 5 Github repository

In order to group together every explained method for the analysis of time series and the anomaly detection procedure, a Github repository [9] has been created. This repository contains several examples to show the main results described in this report; therefore, anyone can use this development to learn more about time series or as a tool for time series analysis in future researches.

The main sections of the repository are described below:

1. The main programme is called *main.py* and it is the starting point for new users. This programme allows to run several examples based on the results obtained through this research.

2. The folder *sources* contains every supplementary file needed for the decomposition of the time series and the anomaly detection procedure.

3. The folder *fortran_interface* contains the *Mean Value Filter* in fortran language to improve efficiency and reduce the computational time. There are also several interfaces to call, from python, the compiled fortran programme.

4. The folder *Figures* contains several images, shown in the main page of the Github repository.

5. The folder *test_NASA* contains the NASA's database with 81 time series. Their labeled anomalies are in the file *NASA_anomalies.txt*.

6. The folder *UCR_Anomaly_FullData* contains the UCR's database with 241 time series and their labeled anomalies.

7. The files *results_NASA.txt* and *results_UCR.txt* show the obtained results after these databases have been analysed.

8. Finally, the file *README.md* contains general information regarding this repository and its content.

# 6 Future research

In this research, the main objective has been to improve traditional methods used for time series decomposition and anomaly detection, its application for time series analysis and its possible use in space systems.

Though the results show a high similarity between the detected anomalies with this approach and the real ones, there are several methods regarding the anomaly detection in time series that have not been covered in this report.

Nowadays, *Neural Networks* are a revolution in many computational fields, due to the excellent results obtained for classification, forecasting and pattern recognition among others. In the field of time series, there are several researches which use LSTM (*Long Short Term Memory*) neural networks for time series forecasting or CNN (*Convolutional Neural Networks*) for classification.

A future research could be oriented to improve even more the recognition of anomalies with a CNN as a new method for anomaly detection and its use in conjunction with the traditional detectors, based on a threshold.

The use of neural networks for time series analysis could lead to results very close to reality and, hence, it would allow its massive use in several systems, improving their efficiency and performance.

Finally, the development of a CNN in fortran could speed-up this approach hundreds of times, reducing the time required for training, analysis and anomaly detection, which would allow an almost real time analysis.

## References

[1] E. Keogh, T. Dutta Roy, U. Naik, A. Agrawal, Multi-dataset time-series anomaly detection competition, SIGKDD 2021 (2021).
URL `https://compete.hexagon-ml.com/practice/competition/39/`

[2] K. Hundman, Anomaly detection in time series data using LSTMs and automatic thresholding (2020).
URL `https://github.com/khundman/telemanom`

[3] D. A. Pal, D. P. Prakash, Practical time series analysis: master time series data processing, visualization, and modeling using Python, Packt Publishing, Birmingham, 2017.

[4] R. B. Cleveland, W. S. Cleveland, J. E. McRae, I. Terpenning, STL: A seasonal-trend decomposition procedure based on loess, Journal of Official Statistics 6 (1990) 3–73.

[5] Arundo Analytics, Inc., Anomaly Detection Toolkit (ADTK) (2020).
URL `https://adtk.readthedocs.io/en/stable/index.html`

[6] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding, In KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery  Data Mining, August 19–23, 2018, London, United Kingdom (2018) 9.
URL `https://doi.org/10.1145/3219819.3219845`

[7] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, Y. Tong, Q. Zhang, Time-series anomaly detection service at microsof, In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA. ACM, New York, NY, USA (2019) 9.
URL `https://doi.org/10.1145/3292500.3330680`

[8] D. Huergo Perea, N. Martínez Figueira, M. Ramiro Aguirre, P. Romero Ramos, Fortran to Python wrapper (2020).
URL `https://github.com/mramagu/Fortran-to-Python-Wrapper/releases/tag/1.0`

[9] David Huergo Perea, Anomaly detection in time series for space applications (2022).
URL `https://github.com/Dhueper/TimeSeries-AnomalyDetection`