Data processing and e-health

PART II : SIGNAL PROCESSING

Laurent Oudre
laurent.oudre@ens-paris-saclay.fr

Master Erasmus Mundus CYBER
2023-2024

Contents

Organization of the three next lectures

- Teaching assistant : Charles Truong ⟨ctruong@ens-paris-saclay.fr⟩

  Teaching material: http://www.laurentoudre.fr/cyber.html

Outline of the three next lectures

- In several protocols, we are interested in monitoring the behavior of a subject or a system over time
- This necessitates the use of adequate **sensors** and **algorithms** to extract relevant information (**features** or **events**) that can be used to provide a quantitive analysis of the phenomena of interest
- In order to conceive a *good* processing pipeline, several aspects must be taken into account, such as the physical, mathematical and statistical properties of the recorded data

Contents

What is a time series?

- A time series is a series of data points indexed in time order
- In practice, array of real numbers of size $D \times N$ where D is the number of dimensions and N the number of samples
  - Sample number n

n	0	1	2	3	4	5	6

  - Time series values $x[n]$

$x[n]$							
	0.7	0.2	0.8	0.9	0.3	0.2	0.7
	0.4	0.1	0.6	0.2	0.5	0.6	0.3

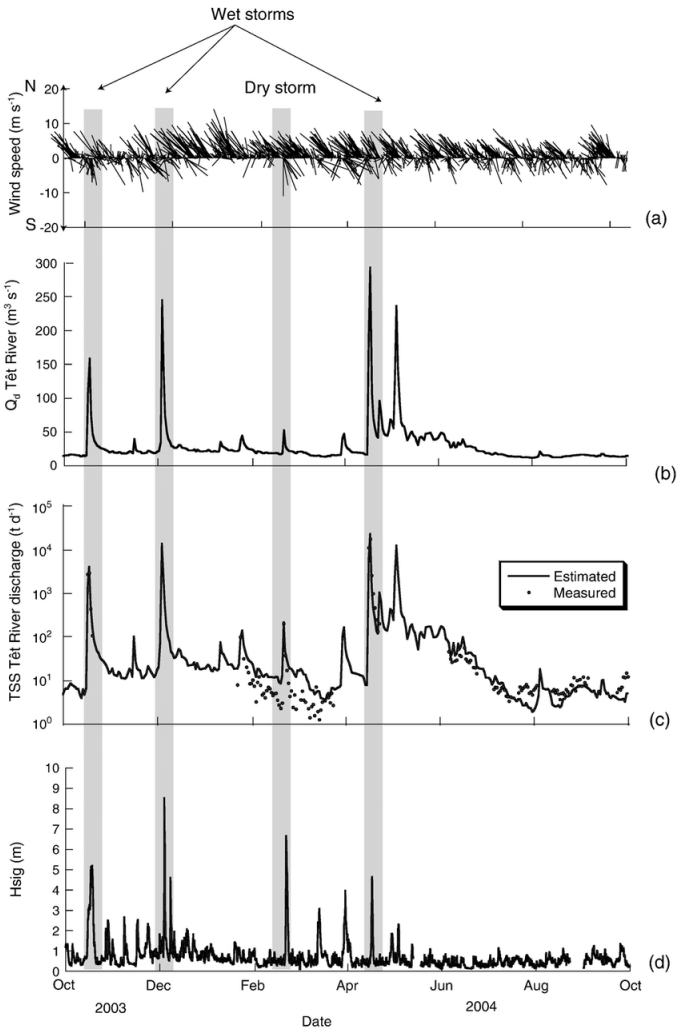
  - Timestamps $t[n]$

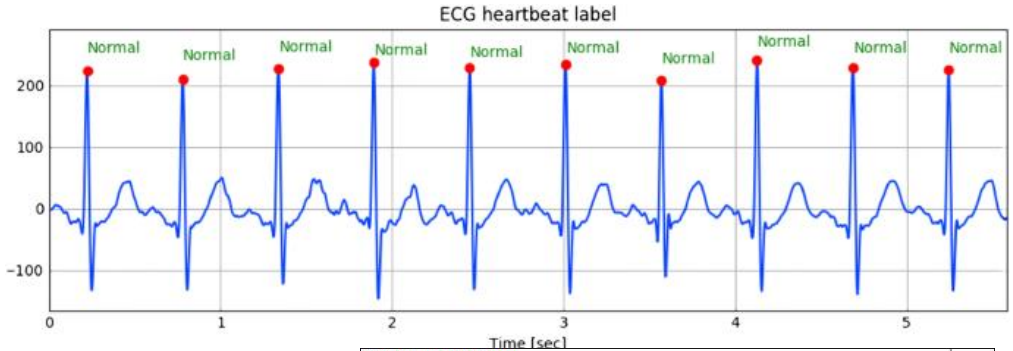
$t[n]$							
	16:30:01	16:30:23	16:31:43	16:32:38	16:33:06	16:33:16	16:33:56

An un-unified field

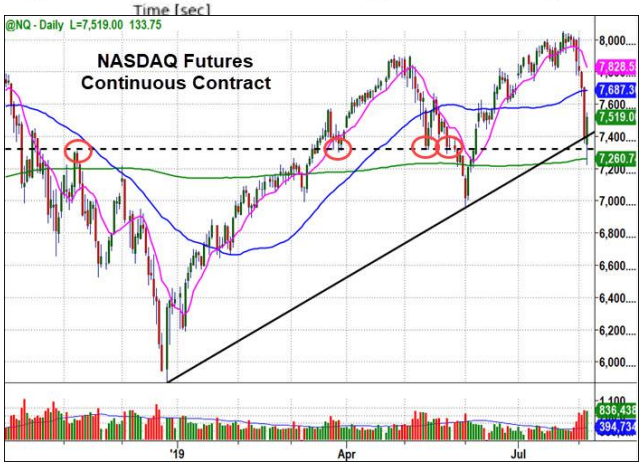
- Different scientific communities have given different names to the same mathematical object.
  - **Time series**: mathematics, statistics, economics, finance...
  - **Signals**: signal processing, physics, engineering, simulation...
  - **Sequences**: computer sciences, bioinformatics, data mining...
- In this course, we will use indifferently one of these terms.
- Typical definition: real-valued (or at least ordered) sequential data

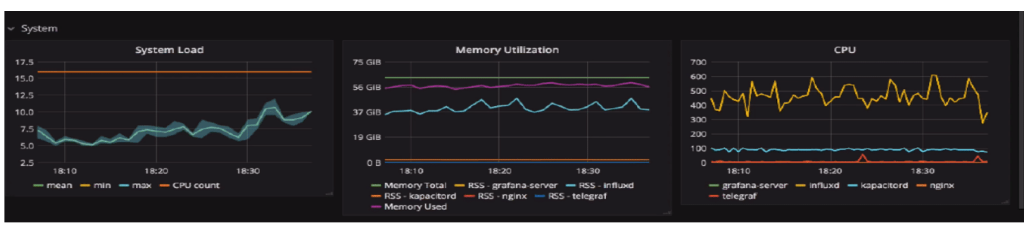
Time series are everywhere





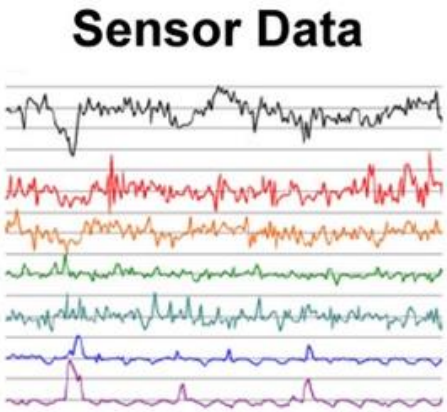
Meteorology,
Finance,
Healthcare,
Monitoring,
Epidemiology,
Sensor networks...

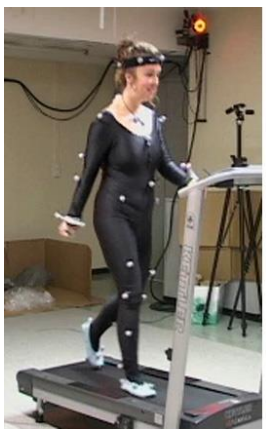


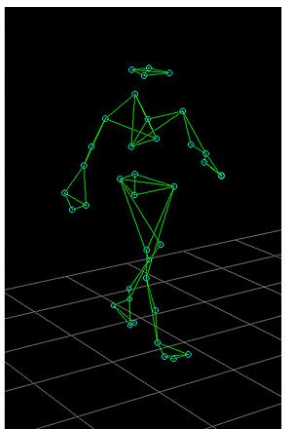


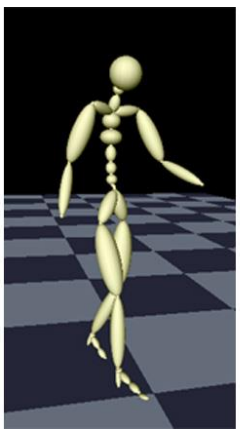
Univariate vs. multivariate

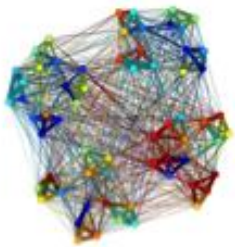
2D/3D trajectories,
Multivariate time series,
Multimodal data from sensor networks,
Graph signals

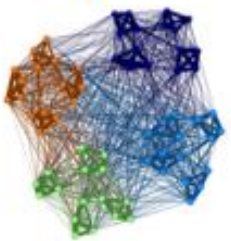


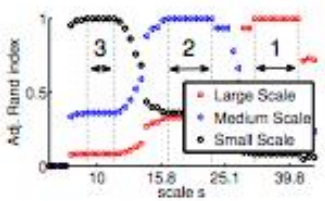












Time series are complex

- Potentially massive data (e.g. sound : sampling frequency 44.1 kHz)
- Multivariate, multimodal, heterogeneous
- Noisy, missing data, trends, mixture of sources
- Often linked to an application context: data scientist is not trained to understand the data

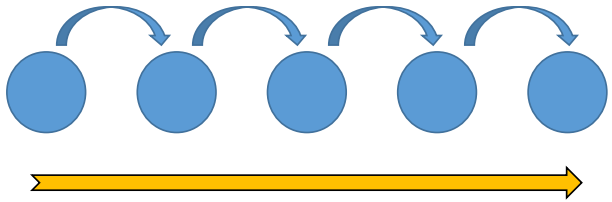
Annotations and ground truth

- Contrary to basic image processing tasks (e.g. classification of cats and dogs), annotating time series often require expertise
- Typical context:
  - Noisy and dirty data
  - A few annotated signals with blurry labels (confusing and hyper-specialized annotations that cannot be transformed into class labels)
  - An expert with several years in the business, but unable to translate it into categorial annotations

How to perform quantitive analysis in this context?

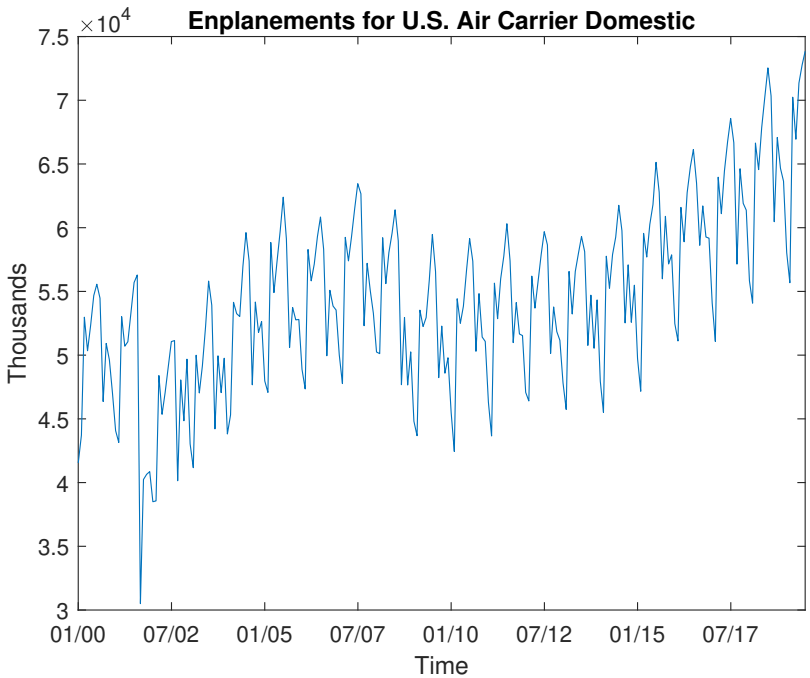
What about time?

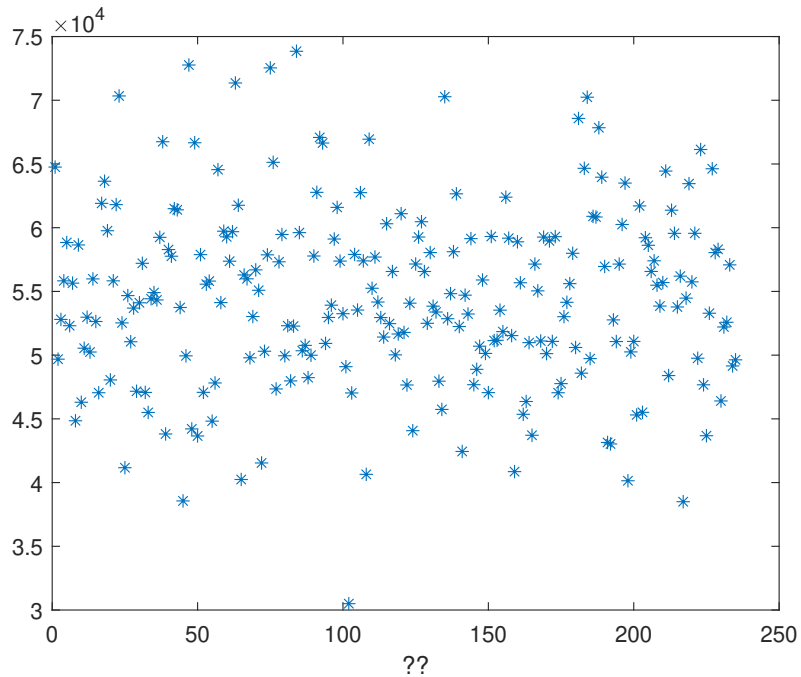
- What is the difference between regular data and time series ? Notion of sequence and chronology



- Each sample corresponds to the measurement of a phenomenon at a given time stamp.
- Time allows to study the evolution of the phenomenon and should be taken into account for processing the data

What about time?





Same time series... but mixed up times

World vs. Machine Learning

0.2
0.1
0.4
1.1
0.9
0.3
0.5
0.2

ML

Male
Young
British accent
« Hello »

43
28
21
08
53

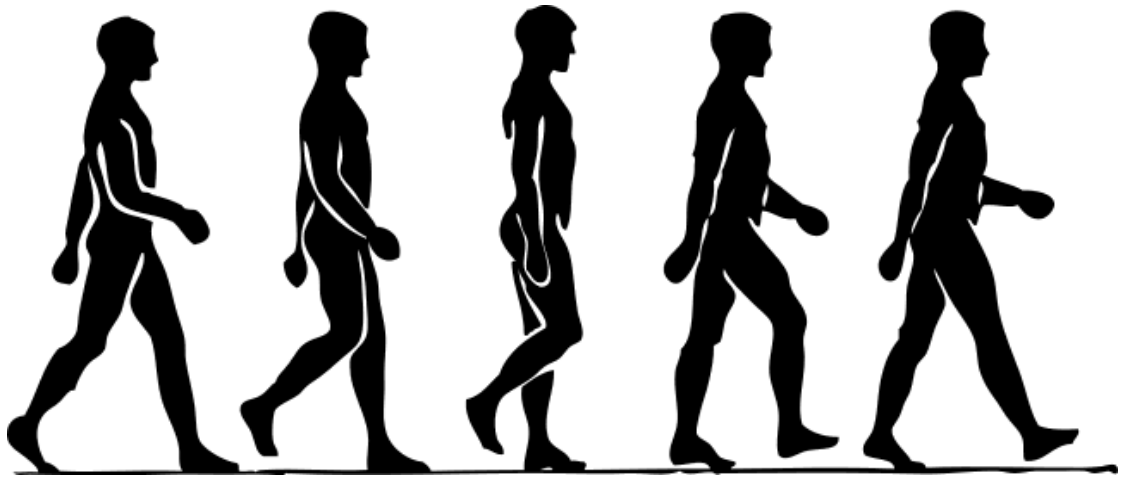
ML

New user
Likely to buy a car

- Most ML algorithms do not care for time.
- How can we still use the time information to extract relevant features/patterns that can be used within a ML procedure ?

Contents

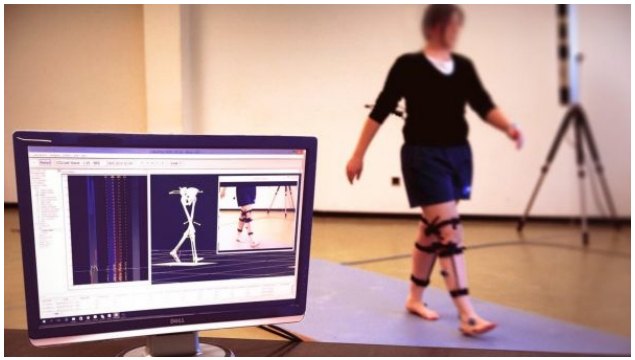
Gait analysis



Why is it important to study locomotion?

- Most common dynamic human activity
- Can reveal a large number of neurological, orthopedic, rheumatological disorders...
- Strong influence on daily life : risk of falling, frailty, autonomy, dependency...

Gait analysis



How can we study locomotion?

- Early tests: clinical examination by the physician, functional tests, clinical questionnaires

+	Easy to perform, use of clinical expertise
-	Lack of precision, difficult to objectively compare two sessions

- Dedicated platforms for the study of locomotion: instrumented mats, video/optical systems

+	Great precision, extraction of a large number of useful features, objective quantification
-	Expensive, difficult to put in practice

Main principles

★ Objective quantification of human gait

→ Use of sensors and physiological measurements

★ Longitudinal follow-up and inter-individual comparison

→ Need for a fixed protocol

★ Experimentation outside the laboratory and on the field

→ User-mounted sensors and fully automatic device for consultation and routine use

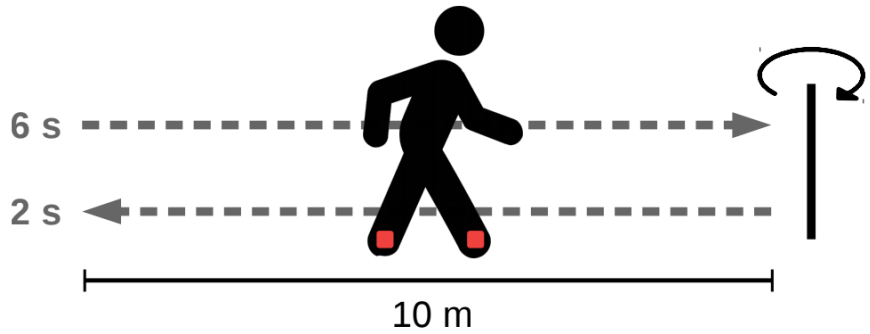
★ Clean data

→ Control of the entire measurement chain, robust and reproducible algorithms

★ Willingness to capture the expertise of the clinician

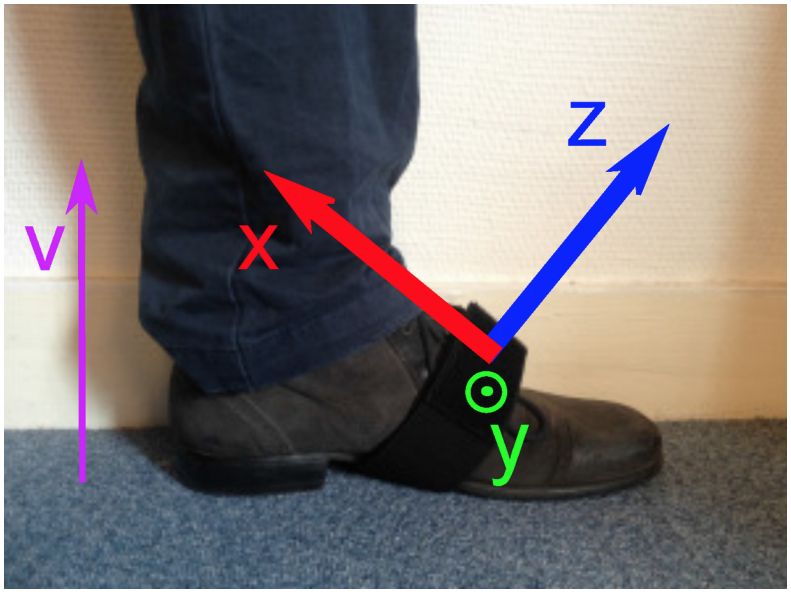
→ Clinical annotations and metadata

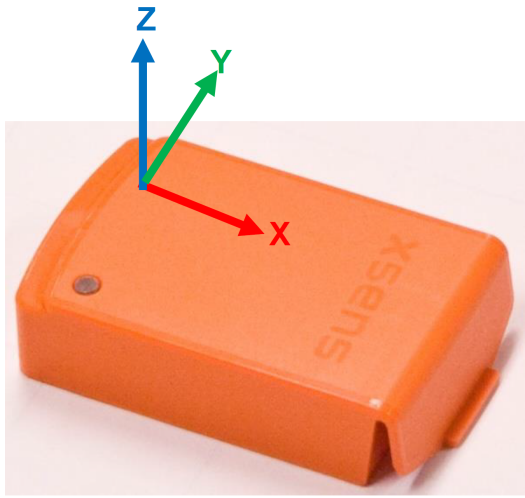
Protocol



- Sequence of activities:
  - stand for 6 s,
  - walk 10 m at preferred walking speed on a level surface to a previously shown turn point,
  - turn around (without previous specification of a turning side),
  - walk back to the starting point,
  - stand for 2 s.
- Subjects walked at their comfortable speed with their shoes and without walking aid.

Sensors





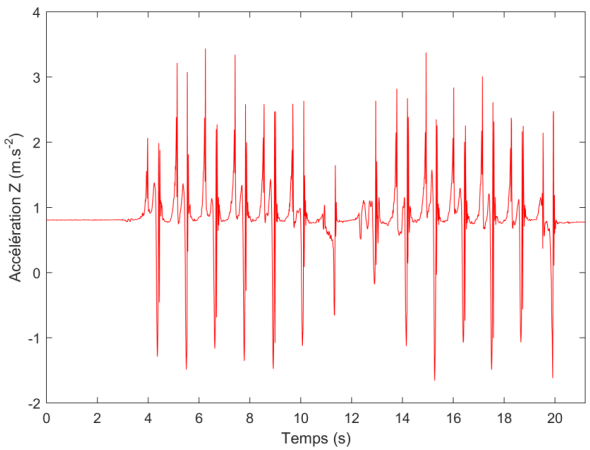
- IMU (Inertial Measurement Unit) record linear accelerations (3D), angular velocities (3D) and magnetic fields (3D) on each foot
- Sensor frame consists of 3-axis (X, Y, Z)
- **For the tutorial: angular velocity around axis Y**

Database

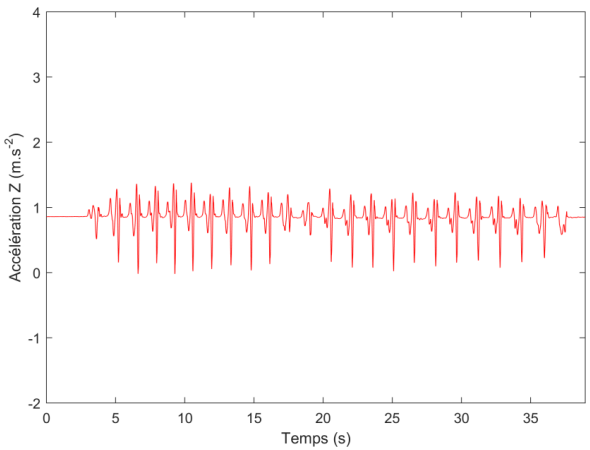
221 recordings:

- Healthy subjects had no known medical impairment (labelled as ”T” for Témoin).
- The orthopedic group is composed of 3 cohorts of distinct pathologies: lower limb osteoarthrosis (ArtH, ArtG), cruciate ligament injury (LCA), knee injury (Genou)
- The neurological group is composed of 2 cohorts: cerebellar disorder (CER) and radiation induced leukoencephalopathy (LER)

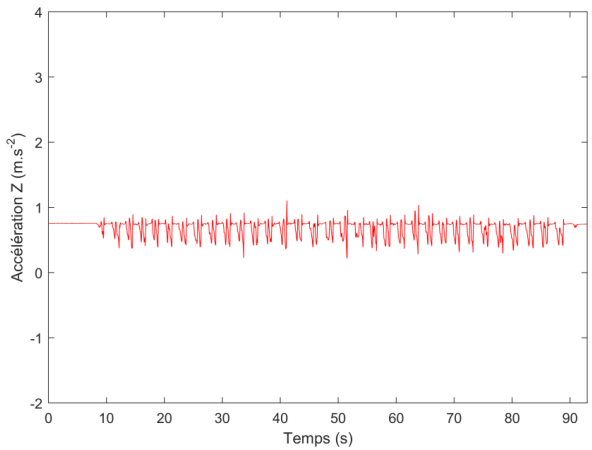
The raw signals



Sujet sain



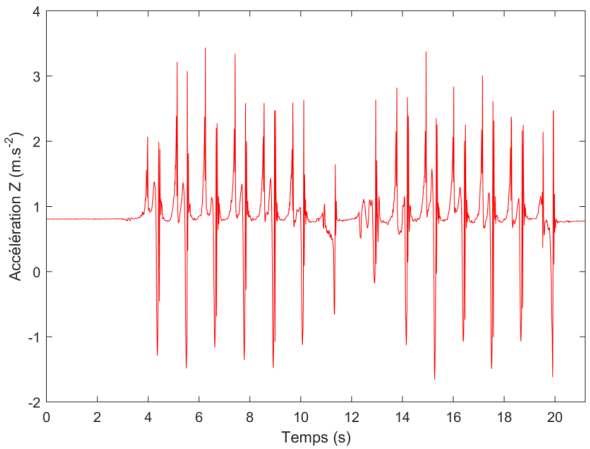
Pathologie neurologique peu sévère



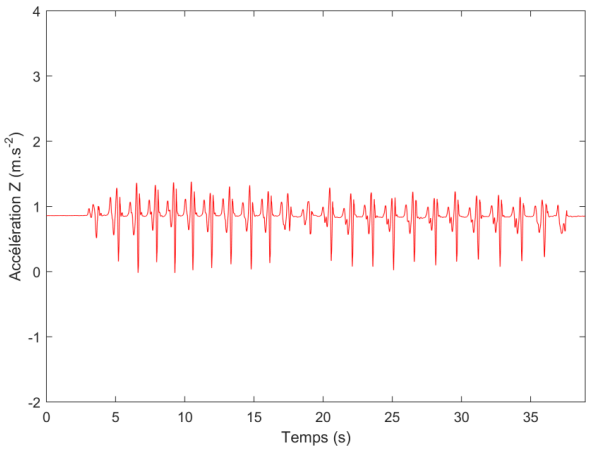
Pathologie neurologique sévère

Contents

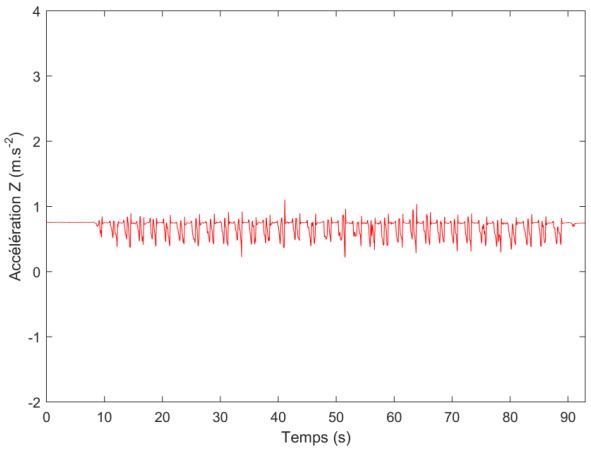
Main scientific questions



Sujet sain



Pathologie neurologique peu sévère



Pathologie neurologique sévère

Non-stationary signals

→ How can be detect the different regimes (stop, walking, U-turn...)?

Presence of repetitive patterns: the steps

→ What are they? How could we automatically extract them?

Robust feature extraction

→ How could be extract relevant features for longitudinal follow-up and inter-individual comparison?

Scientific questions

- How can we construct a good instrumented protocol ? How to choose the sensors ? How to consolidate the data ?
- How to extract relevant features from the data ? How to make sure that those are robust and well-computed ?
- Are there some meaningful events that we could extract from the data and could be used for quantitative analysis ?

Outline of the course: first part

2. Basic signal processing tools
2.1 Sampling theory
2.2 Discrete Fourier Transform
2.3 Digital filters
2.4 Stationarity, ergodicity and autocorrelation function
2.5 Spectrogram

Outline of the course: second part

Outline of the course: third part

Contents

Two visions: physics vs. statistics

- The notion of time have been used and modeled in physics since 18th century and before (eg. Fourier transform).
  First vision : a time series $x[1:N]$ is the result of the digitization of a physical phenomenon $x(t)$. Physical properties of this phenomenon can be retrieved and analyzed through the study of $x[1:N]$ (and vice/versa).
- Randomness can also play a part to model a wider class of signals.
  Second vision : a time series $x[1:N]$ is a realization of a stochastic process $X[1:N]$. Statistical properties of this phenomenon can be retrieved and analyzed through the study of $x[1:N]$ (and vice/versa).

In most cases, both approaches can be combined.

Some useful signal processing tools

In the following, we will review basic signal processing tools and apply them to our signals:

- Sampling theory
- Discrete Fourier Transform (DFT)
- Digital filters
- Notion of stationarity, ergodicity and autocorrelation function
- Spectrogram

Contents

What are we recording ?

- Often, we want to capture a physical phenomenon by using a sensor
  - Sound from a microphone
  - Temperature from a thermometer
  - Linear acceleration with an embedded inertial measurement unit...
- These sensors allows to record what happen in the real world and to transform this into understandable information that we can process on our computers

Continuous and discrete signals

There exist two types of signals:

- **Continuous :** the value is known for each time value t

$$\boxed{x(t) \text{ with } t \in \mathbb{R}}$$

t : time (often given in seconds)
Ex : waves, electrical signal, light, sound...

- **Discrete :** the value is only known for a limited set of time values $t[n]$

$$\boxed{x[n] \text{ with } n \in \mathbb{Z}}$$

n : sample (no standard unity)
Ex : rainfalls recorded each hour, number of people in ICU each day ...

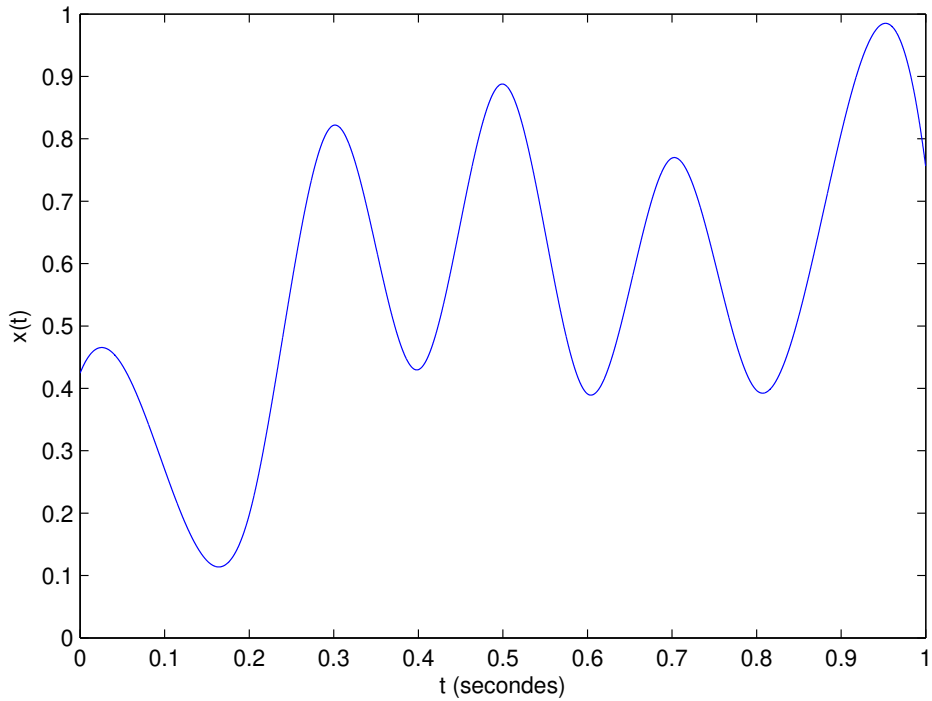
Continuous and discrete signals

In practice

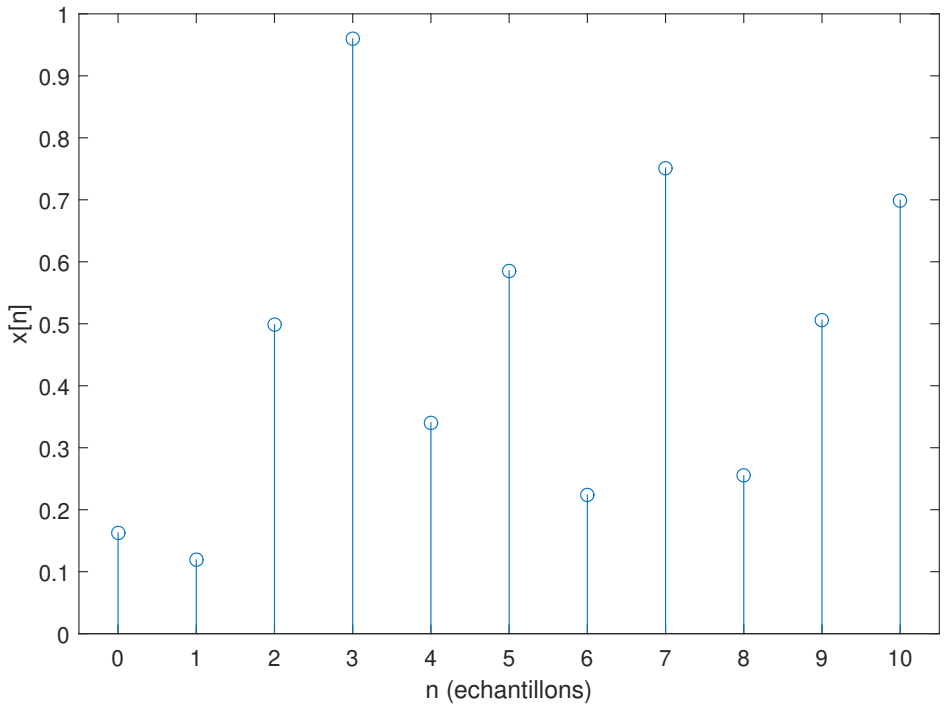
- The continuous signals $x(t)$ cannot be stored on a computer since they contain an infinite number of values! Those can be seen as mathematical functions and are used to construct theoritical models.
- The discrete signals $x[n]$ can be stored on a computer (provided that there is a limited number N of samples). A discrete signal can be seen as a table of N cells, where each cell is a sample value $x[n]$. Most of the time, we also store a timestamp table that contains all time values $t[n]$ at which the signal has been recorded

Remark : All the signal you will process in tutorials are necessarily discrete signals discrets.

Examples



Continuous signal $x(t)$
$t \in [0, 1]$



Discrete signal $x[n]$
$n \in [\![0, 10]\!]$

Sampling theory

- When we use a sensor for a protocol, the sensor will perform a task, called **sampling**, that aims at converting the physical phenomenon of interest (continuous signal) into a quantified discrete signal that can be stored on the computer
- This sampling step can be performed on a regular temporal grid (e.g. record a value each second) or irregular (e.g. record a value each time a button is pressed)
- Careful ! When the sampling is irregular, it is waaaaaay more complicated to process the data.
- Most of the time, we use a regular sampling, which is referred to as **uniform sampling**

Uniform sampling

- Principle : Convert a continuous signal $x(t)$ into a discrete signal $x[n]$ by only storing what happens at certain timestamps $t[n]$

$$x[n] = x(t[n])$$

- For uniform sampling, we record a value each T_s seconds, where T_s is fixed:

$$t[n] = nT_s = \frac{n}{F_s}$$

- T_s is called the **sampling period** (in seconds)
- $F_s = \frac{1}{T_s}$ is called the **sampling frequency** (in Hertz)

Important

- The sampling frequency F_s corresponds to the number of samples that we will be recorded in one second
- It is a **crucial** parameter when designing a protocol or a study, and a very important feature to take into account when choosing the sensor
- As will be seen, the sampling frequency will impact all the processing steps that will be applied to the signal

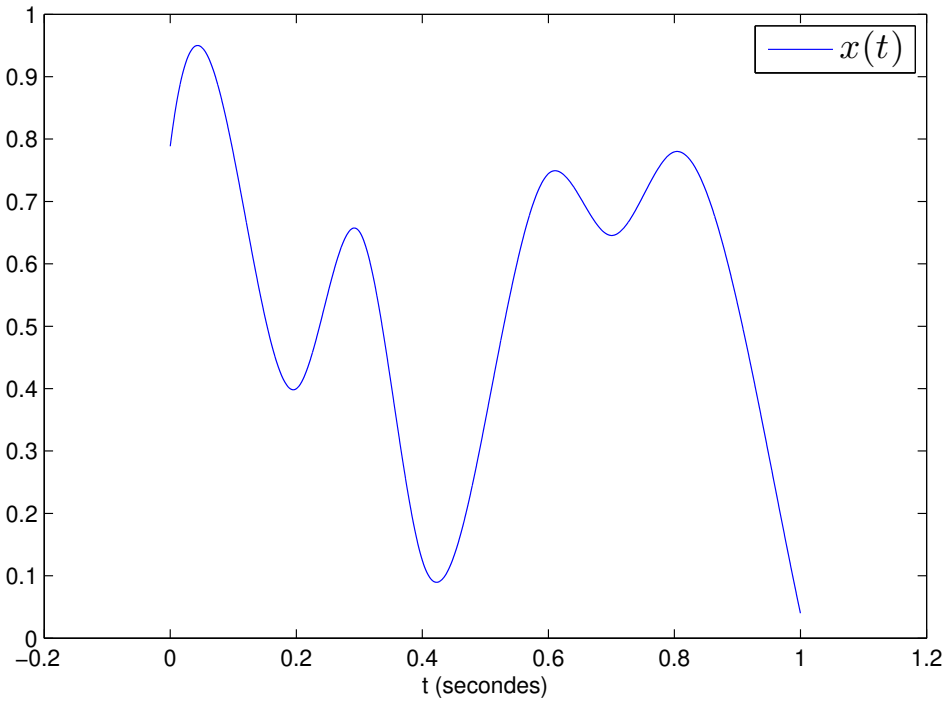
Uniform sampling

- When sampling a signal with a sampling frequency F_s, we store the following quantities :

Sample n	Timestamp $t[n]$	Signal value $x[n]$
0	0	$x(0)$
1	T_s	$x(T_s)$
2	$2T_s$	$x(2T_s)$
3	$3T_s$	$x(3T_s)$
$\vdots$	$\vdots$	$\vdots$

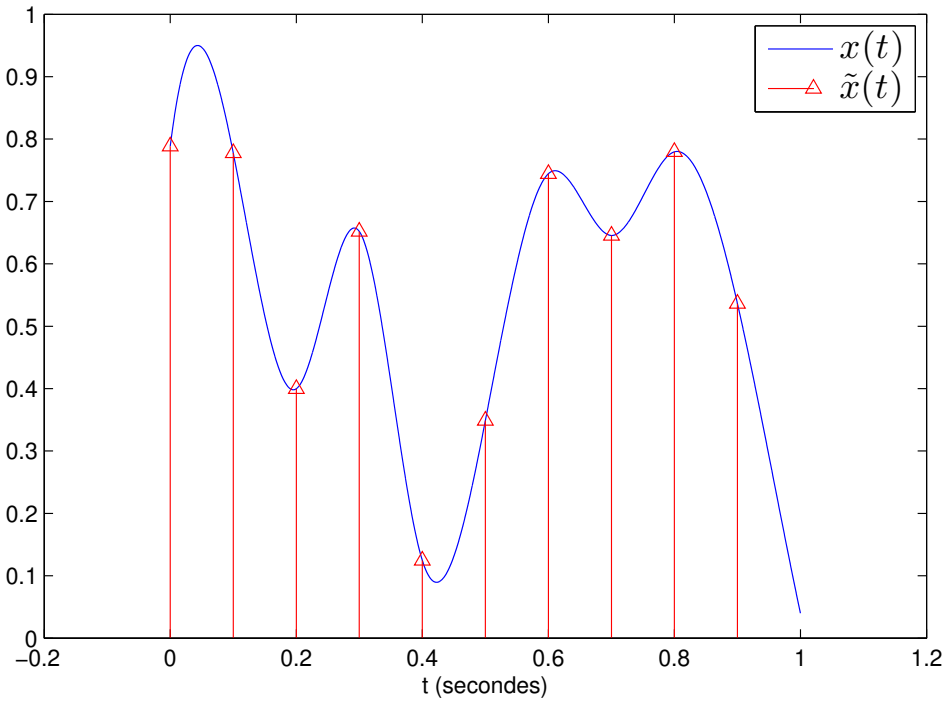
- The total number of samples will be $N = d \times F_s$ where d is the duration (in seconds) of the signal and F_s the sampling frequency

Example



- Continuous signal $x(t)$ defined for $t \in [0, 1[$

Example

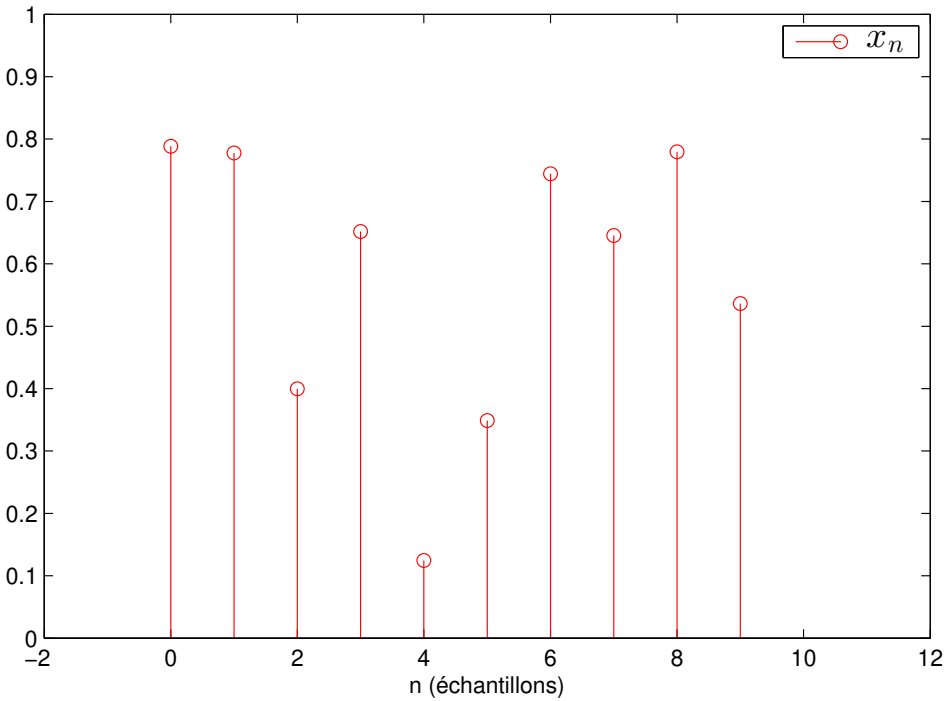


- We take a value each 0.1 seconds by starting at $t = 0$ and stopping at $t = 0.9$:
  - $T_s = 0.1$ seconds
  - $F_s = 10$ Hz
- Timestamps $t[n]$ defined as

$$t[n] = nT_s = \frac{n}{F_s} \text{ for } n \in [\![0, 9]\!]$$

$$t[0] = 0, t[1] = 0.1, t[2] = 0.2, \cdots$$

Example



- Each signal value

$$x(t[n]) = x(nT_s) = x\left(\frac{n}{F_s}\right)$$

  is stored in a table containing $N = 10$ samples
- $x[n] = x(t[n])$ with $n \in [\![0, 9]\!]$

Sampling theorem

- Intuitively, the choice of the sampling frequency depends on the speed of the signal variations
  E.g : the temperature in the room will not change each second, so maybe taking a sampling period of a few minutes is enough
- How can we choose the best sampling frequency ?
- Compromise
  - Large enough to take account the variations that are useful for the study
  - As small as possible to keep the memory storage limited

Shannon-Nyquist sampling theorem

Shannon-Nyquist sampling theorem (unformal form)

If f_{max} is the maximum frequency present in the phenomenon of interest, you should have $F_s > 2f_{max}$

- Conversely, if you have recorded a signal with sampling frequency F_s, you will only be able to study its frequency content for $|f| < \frac{F_s}{2}$ (very very important property !)
- The frequency $\frac{F_s}{2}$ is often called the **Nyquist frequency** : larger observable frequency in a discrete signal sampled at F_s Hz

Examples

- An (young healthy) human ear can hear frequencies up to 20kHz, so if you can to record HD sound, you should select $F_s > 40$kHz
- The maximum frequency of the acceleration produced by the human body is around 20Hz, so if you plan to use an accelerometer to record human movement, you should select $F_s > 40$Hz
- For eye movements it depends if you want to have access to the global gaze movements or to very rapid eye movements such as saccades, etc...

Contents

Sampling and Fourier analysis

- Most tools for signal processing are derived from Fourier analysis
- In this context, we assume that $\mathbf{x}$ corresponds to the discrete measurement of a continuous signal $x(t)$
- We consider uniform sampling period T_s and sampling frequency $F_s = \frac{1}{T_s}$

$$x[n] = x(nT_s)$$

Discrete Fourier Transform (DFT)

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{kn}{N}} \text{ pour } 0 \leq k \leq N-1$$

where N is the number of samples

- The space between two observable frequencies is called **frequency resolution**

$$\Delta f = \frac{F_s}{N}$$

- $X[k]$ corresponds to the DFT for the physical frequency

$$f[k] = k\frac{F_s}{N} \text{ for } 0 \leq k \leq N-1$$

Discrete Fourier Transform (DFT)

- Remark : the Discrete Fourier Transform (DFT) is $N-$periodic:

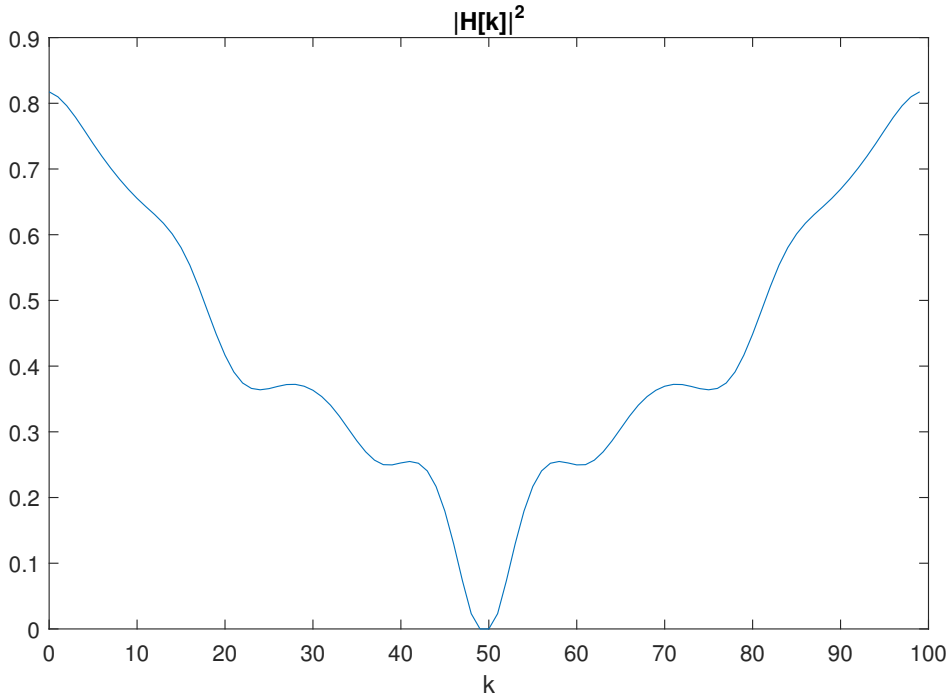
$$\begin{aligned} X[k+N] &= \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{(k+N)n}{N}} \\ &= \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{kn}{N} - j2\pi n} \\ &= X[k] \end{aligned}$$

- In particular, the frequencies $f[k] = k\frac{F_s}{N}$ with $k > \frac{N}{2}$ do not verify the Shannon-Nyquist theorem (Remember : if you have recorded a signal with sampling frequency F_s, you will only be able to study its frequency content for $|f| < \frac{F_s}{2}$)...
- In reality, they correspond to the frequencies $f[k-N] = k\frac{F_s}{N} - F_s$ which are actually comprised between $-\frac{F_s}{2}$ et 0
- When you compute the DFT with the standard FFT (Fast Fourier Transform) algorithm, for N even, you will therefore observe the frequencies

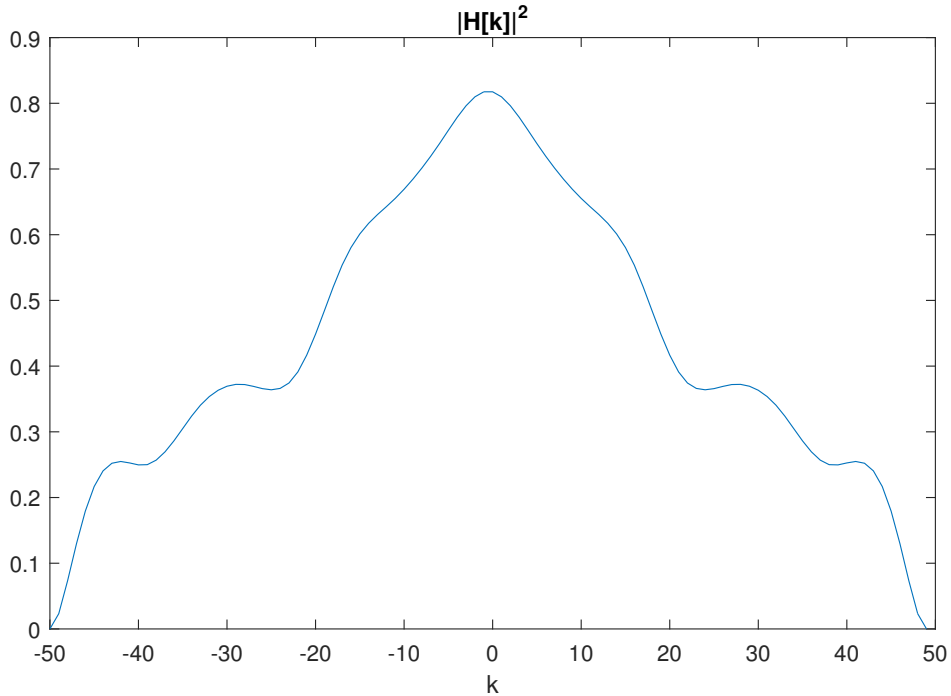
$$0, \frac{F_s}{N}, 2\frac{F_s}{N}, \ldots, \frac{F_s}{2}, -\frac{F_s}{2} + \frac{F_s}{N}, \ldots - \frac{F_s}{N}$$

- This can be a bit confusing, and most packages (Matlab or Python) have functions that allows to shift the frequencies back to the right order $\left[-\frac{F_s}{2}, \frac{F_s}{2}\right]$

Use of the FFT algorithm



Raw FFT



Reshifted FFT

Frequency resolution

- The DFT only allows to have access to N frequencies :

$$f[k] = k\frac{F_s}{N} \text{ for } 0 \leq k \leq N-1$$

- Actually, if the original signal is real, only the positive frequencies have a physical meaning, so most of the time there is only $\frac{N}{2}$ useful frequency bins
- The frequency resolution $\Delta f = \frac{F_s}{N}$ is therefore crucial. If we note d the original signal durations in seconds, and by recalling the fact that $N = d \times F_s$, we have that

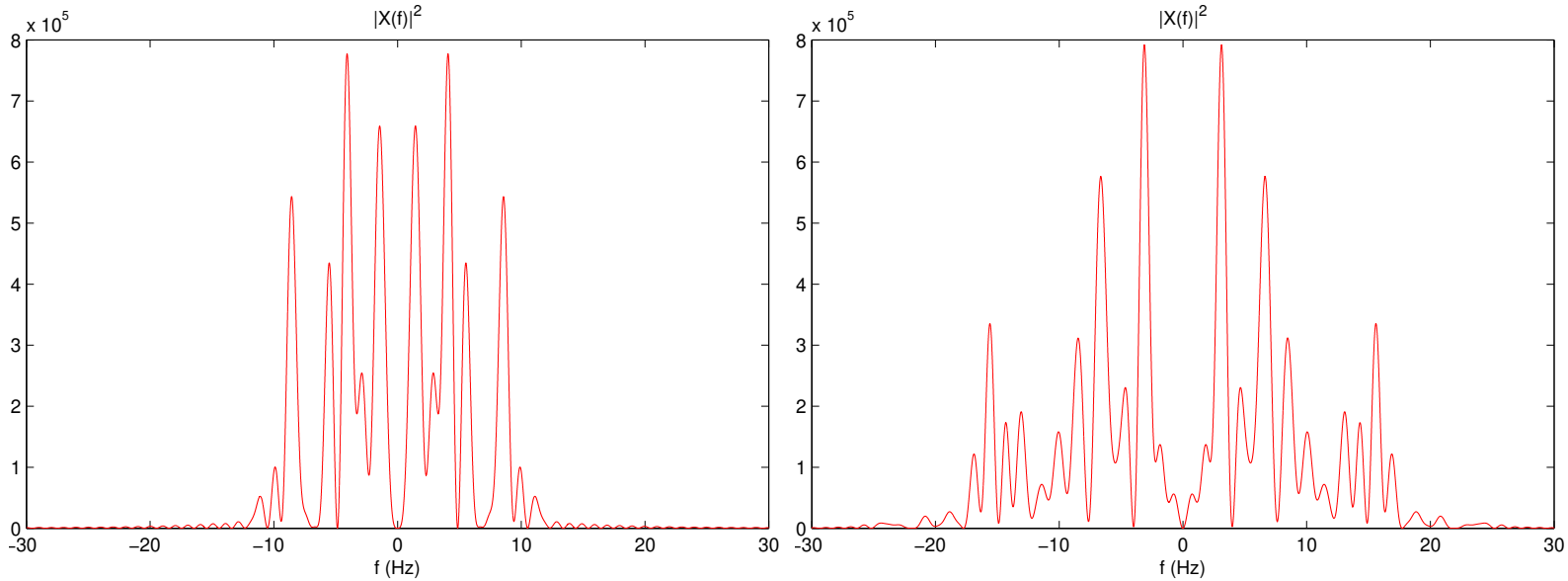
$$\Delta f = \frac{1}{d}$$

- The longer the observation time, the better the frequency resolution.

Spectral analysis

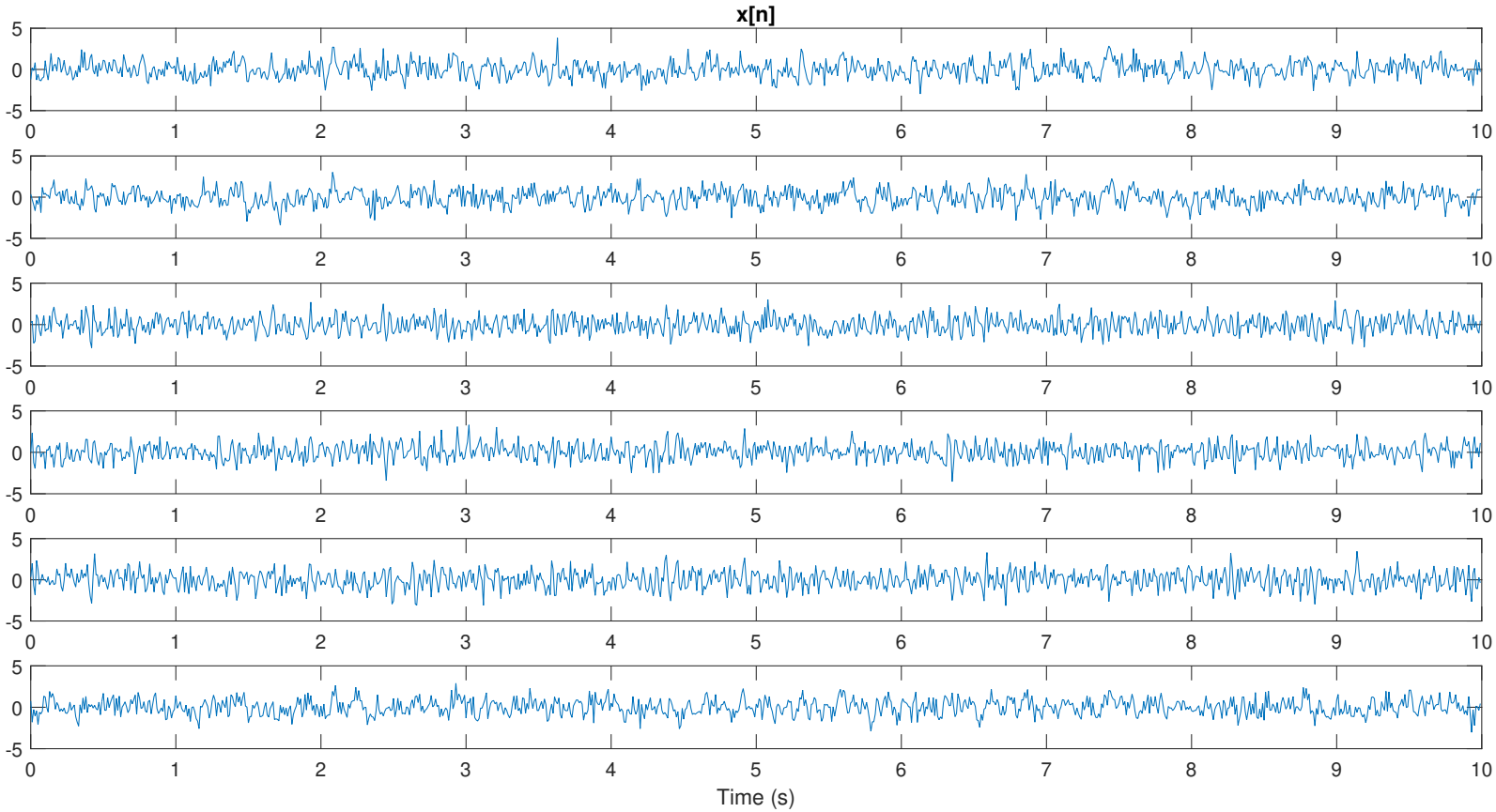
- $X[k]$ is a complex quantity: most of the time, we use the squared absolute values $|X[k]|^2$ instead
- The analysis of the quantity $|X[k]|^2$ can allow to discover interesting properties of the time series
- $|X[k]|^2$ with low frequencies $f[k]$ correspond to phenomena with smooth variations
- $|X[k]|^2$ with large frequencies $f[k]$ correspond to phenomena with fast variations
- One very useful plot consists in plotting $|X[k]|^2$ as a function of $f[k]$: such plot is often referred to as **spectrum** (hence spectral analysis)

Example



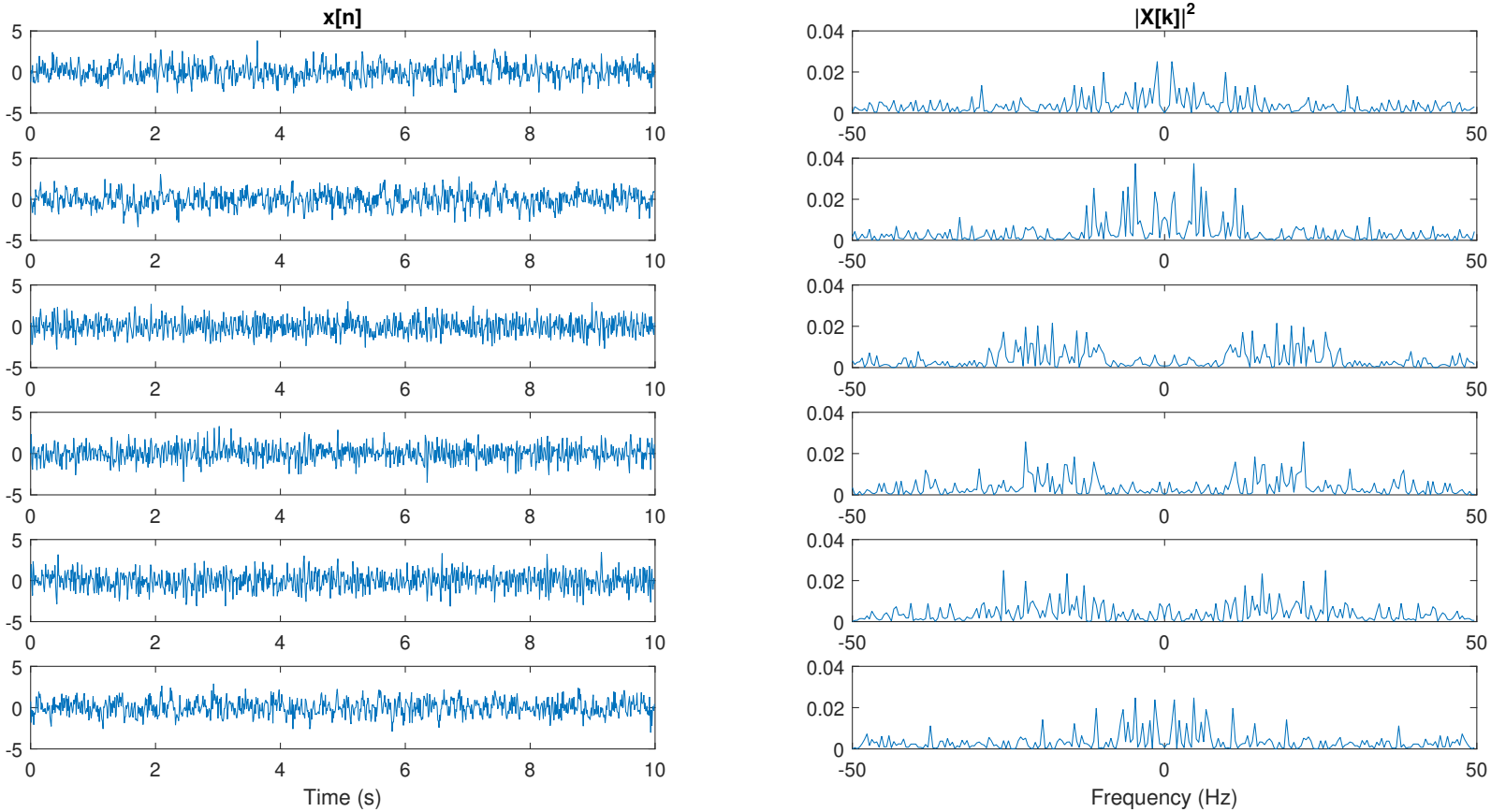
- Two signals sampled at 60 Hz : we therefore observe the frequency band $[-30, 30]$
- Only positive frequencies are relevant : the first signal has frequencies up to 10Hz and the second up to 20Hz
- The second signal is likely to contain faster variations than the first

Example



Two classes of signals?

Example



Can be distinguished based on their DFT coefficients

DFT features

- $|X[k]|^2$ coefficients can be useful features
- In most cases, we can merge them on a frequency band of interest $[f_1, f_2]$, with $0 \leq f_1 < f_2 \leq \frac{F_s}{2}$
- The computed quantity is called relative energy, and is often renormalized by the total energy of the signal

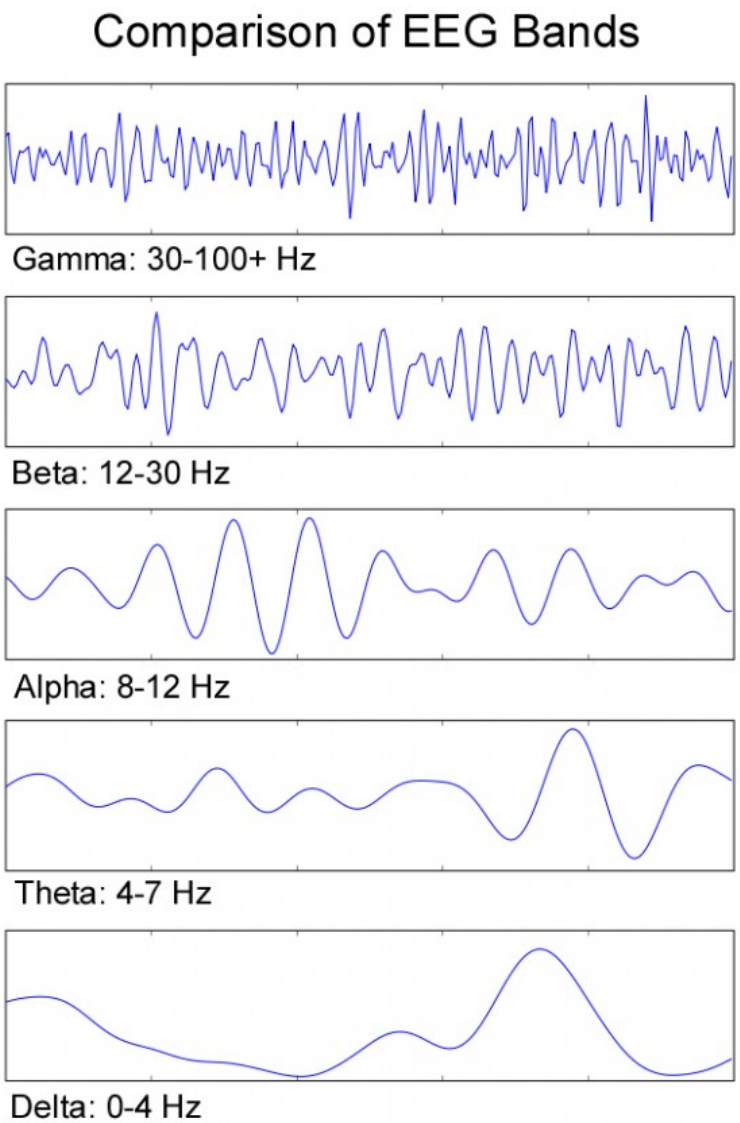
$$E_{[f_1,f_2]} = \frac{\sum_{k, f[k] \in [f_1, f_2]} |X[k]|^2}{\sum_{k, f[k] \in [0, \frac{F_s}{2}]} |X[k]|^2}$$

Example: EEG

Electroencephalography (EEG):

- Measures the electrical activity of the brain with a sensor network
- Used to study several neurological diseases (epilepsy, stroke...) and sleep
- Each sleep phase corresponds to the emergence of typical frequencies in the brain

Practical example: EEG



Each sleep phase corresponds to the emergence of typical frequencies in the brain:

- Delta waves: deep sleep without dreams
- Theta waves: deep relaxation
- Alpha waves: light relaxation
- Beta waves : awake, active information processing

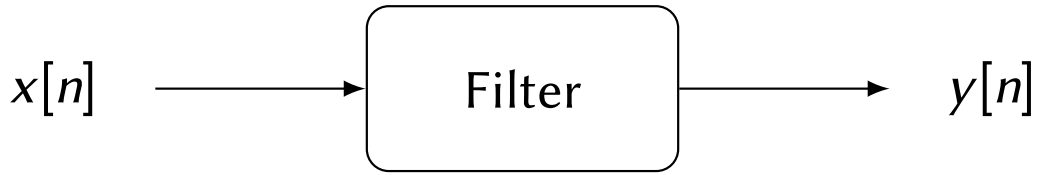
Spectral features allow to figure out in which sleep phase a subject is

Contents

From spectral analysis to digital filters

- As seen previously, the spectral analysis allows to see what *hides* behind a signal
- But the frequency domain can also be useful to modify the signal in order to hightlight or remove certain phenomena
- This task is called **filtering** : we will transform a signal $x[n]$ into a signal $y[n]$ with different properties
- In this lecture, we will only consider **linear filtering**

Linear digital filters



$$a_0 y[n] = \sum_{i=0}^{q} b_i x[n-i] - \sum_{j=1}^{p} a_j y[n-j]$$

- The linear digital filtering operation computes the filtered sample $y[n]$ as a linear combination of previous input values $x[n-i]$ and output values $y[n-j]$
- The choice of the a_j and b_i coefficients and of the orders p and q will impact the type of treatment that the filter will perform
- Most of the time, the conception of the filters is performed in the frequency domain with some standard patterns

Filter bandwidth

- When conceiving a filter, we first consider the quantity $H(f)$ which is the frequency representation of the filter effects. This quantity is called a **transfer function**
- Frequencies for which $|H(f)|^2$ is large will be kept (or amplified) in the filtered signal
- Frequencies for which $|H(f)|^2$ is low will be suppressed (or attenuated) in the filtered signal
- The interval of frequencies that are kept by the filter is called the **bandwidth** of the filter $B = [f_{min}, f_{max}]$ with $f_{min} \geq 0$ and $f_{max} \geq 0$

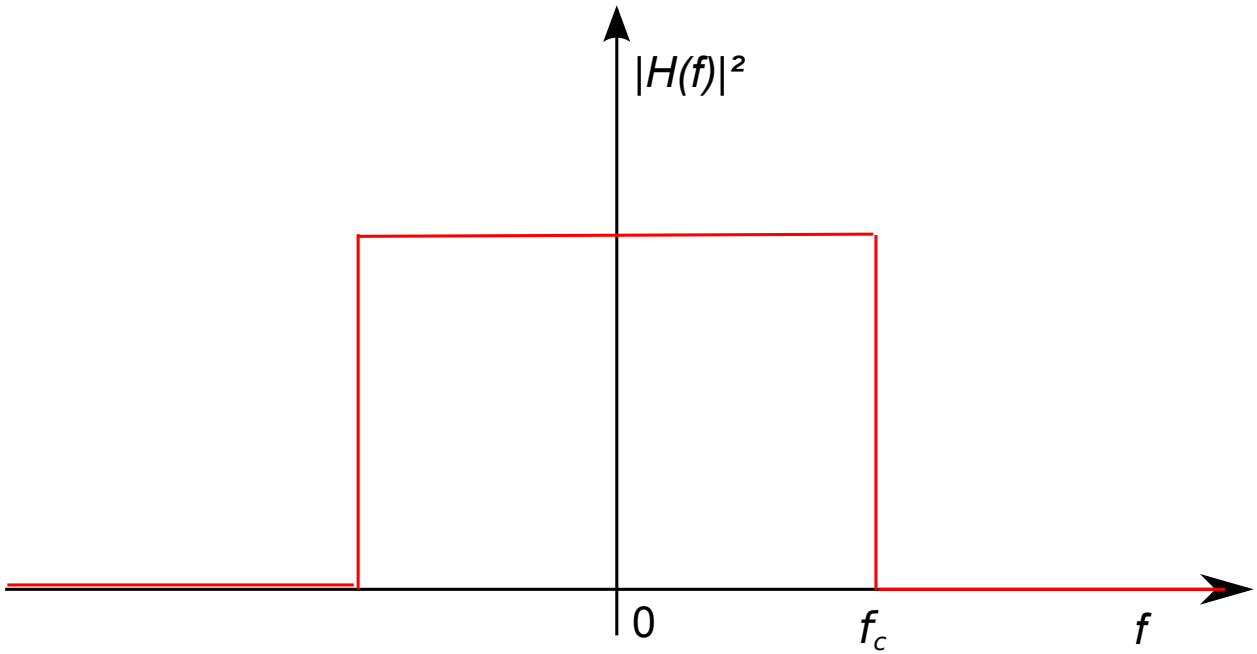
Filter design: ideal filters

- To simplify the filter design, we often consider ideal filters whose transfer function is simply

$$H(f) = \begin{cases} 1 & \text{if } |f| \in B \\ 0 & \text{otherwise} \end{cases}$$

- There are four main types of behaviors : low-pass, high-pass, band-pass and band-reject
- Each of these ideal filters are somehow *perfect*: some frequencies are conserved and kept untouched, while others are completely removed
- In practice, when we will conceive the actual digital filters (i.e. the a_j, b_i coefficients and the orders p and q), we will only get some approximation of this ideal behavior

Low-pass filter

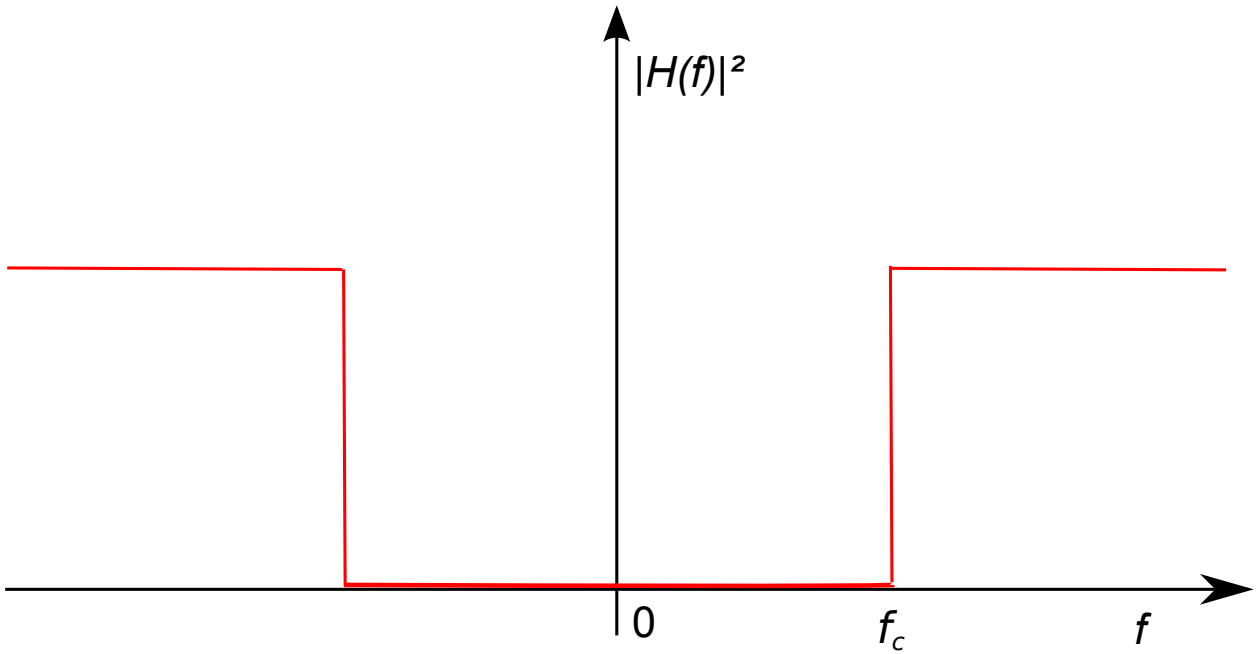


Cut-off frequency : f_c

All frequencies out of the band $[-f_c, f_c]$ will be removed

Bandwidth $B = [0, f_c]$

High-pass filter

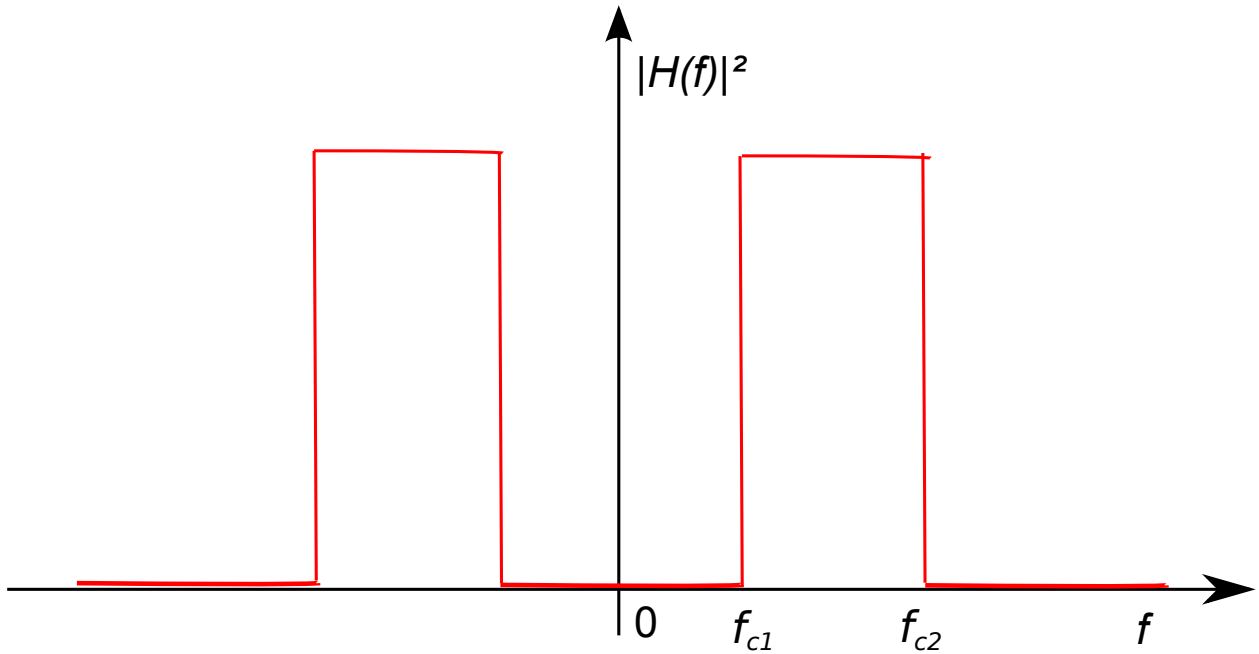


Cut-off frequency : f_c

All frequencies in the band $[-f_c, f_c]$ will be removed

Bandwidth $B = [f_c, +\infty[$

Band-pass filter

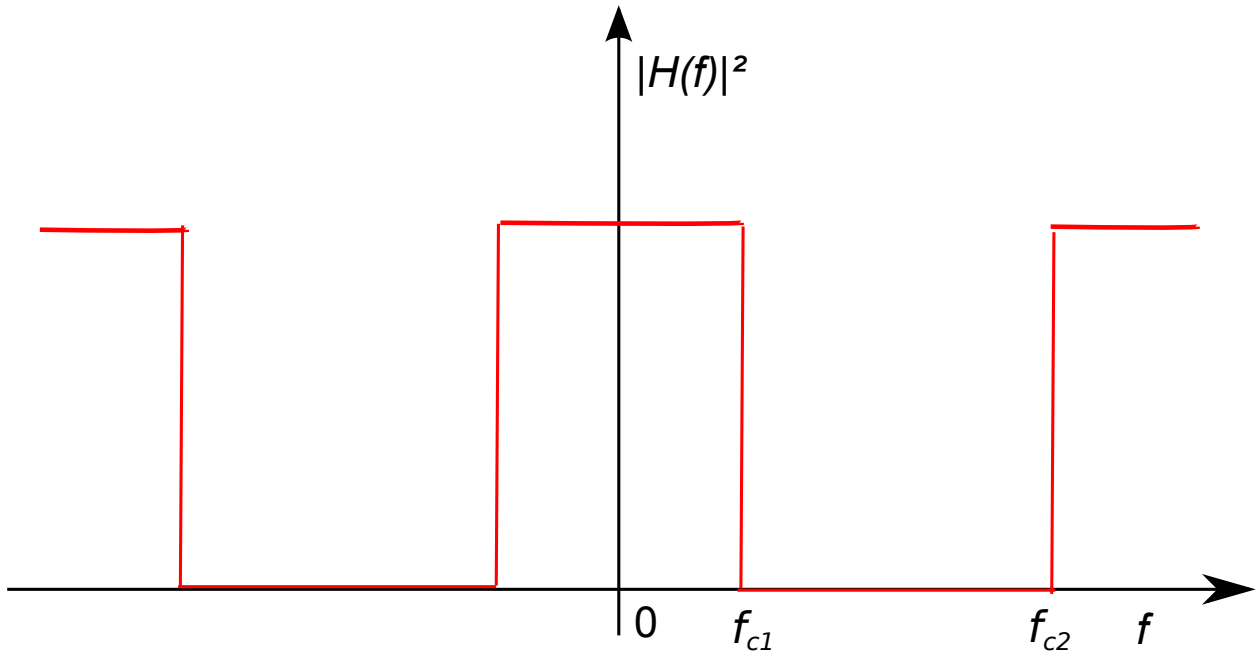


Cut-off frequencies : f_{c1}, f_{c2}

All frequencies out of the band $[-f_{c2}, -f_{c1}] \cup [f_{c1}, f_{c2}]$ will be removed

Bandwidth $B = [f_{c1}, f_{c2}]$

Band-reject filter

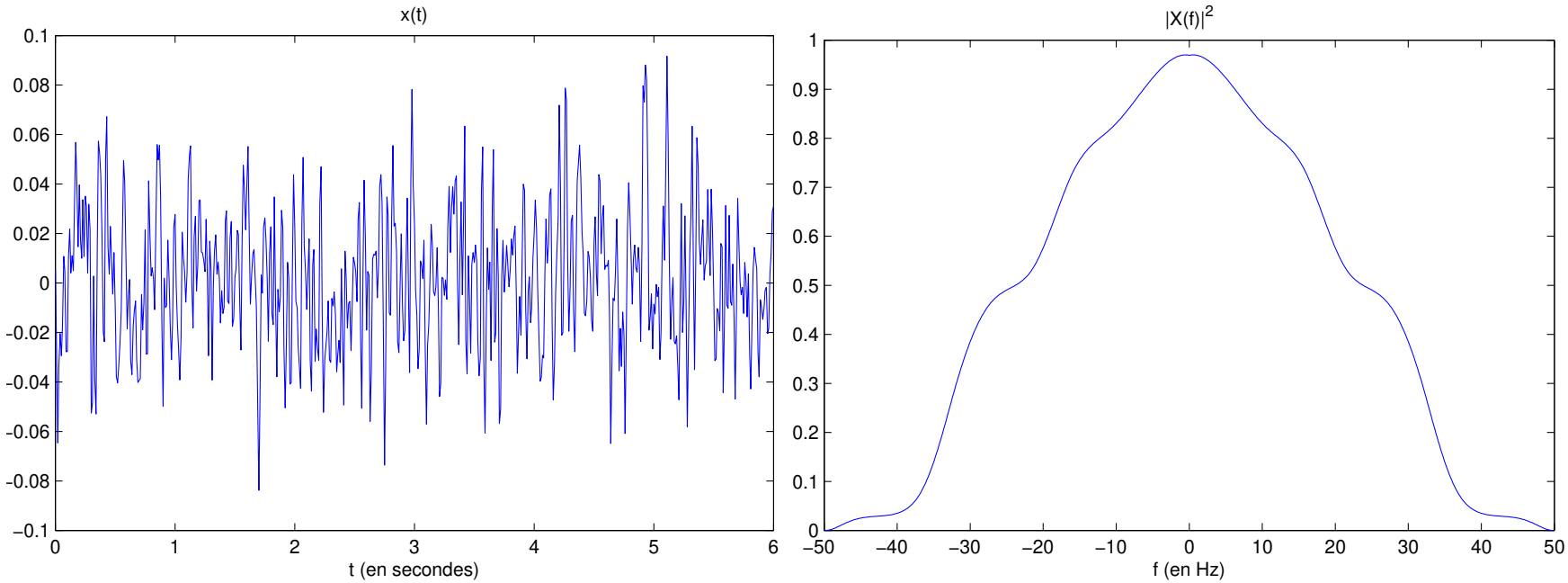


Cut-off frequencies : f_{c1}, f_{c2}

All frequencies in the band $[-f_{c2}, -f_{c1}] \cup [f_{c1}, f_{c2}]$ will be removed

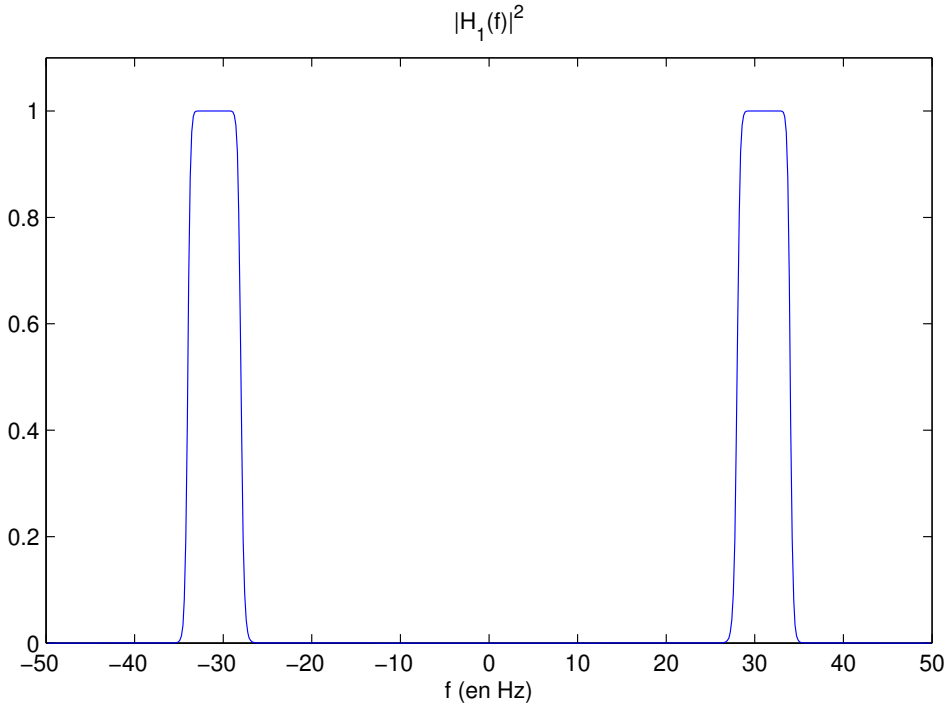
Bandwidth $B = [0, f_{c1}] \cup [f_{c2} + \infty[$

Example



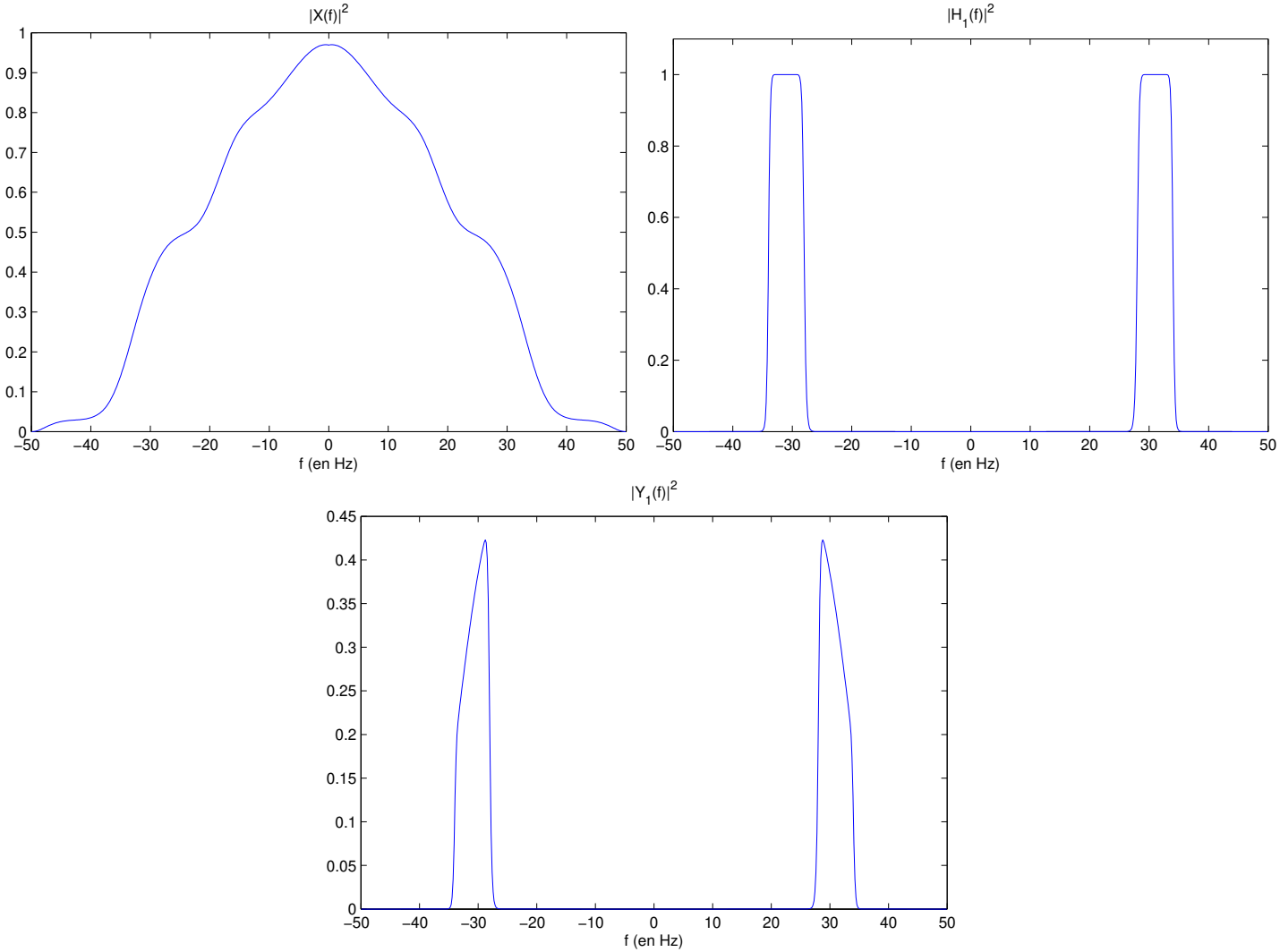
We consider a signal sampled at 100Hz and we will see the effects of the ideal filters on it

Example 1 : Band-pass



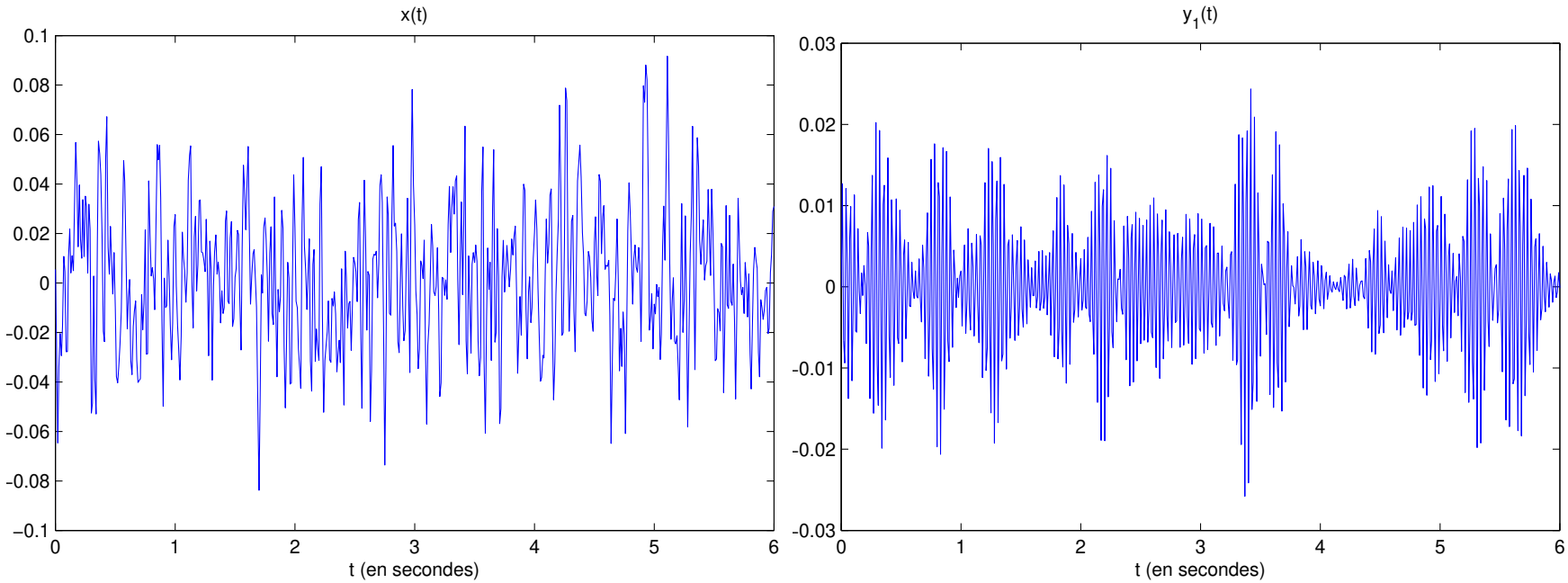
Band-pass filter
Bandwidth $B \approx [28, 34]$ Hz

Example 1 : Band-pass



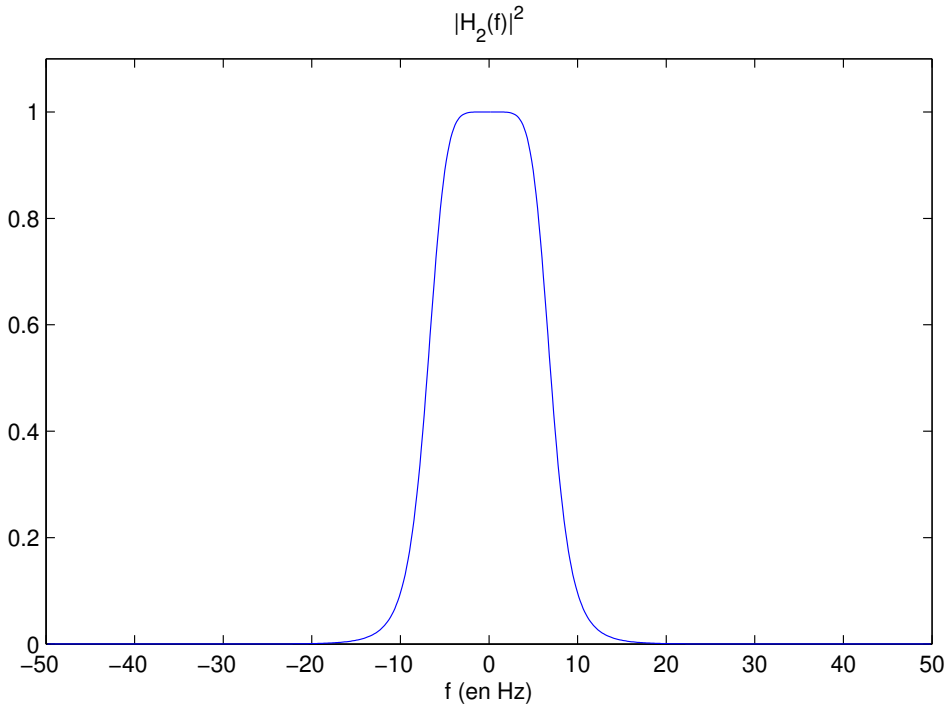
When we look at the spectrum of the filter output, we can see that all frequencies outside the bandwidth have been attenuated

Example 1 : Band-pass



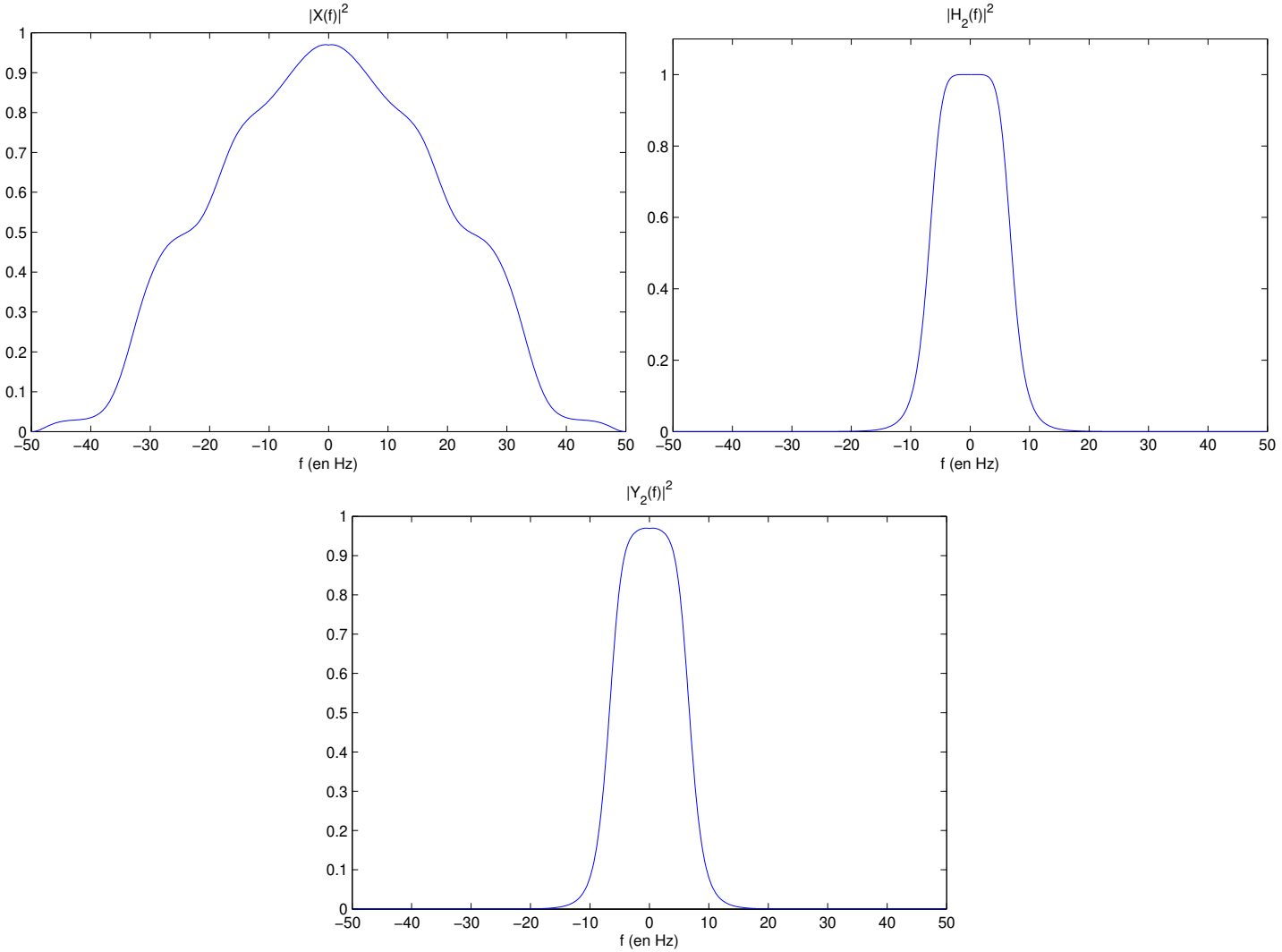
The signal seems more structured: low frequencies and high frequencies have been removed

Example 2 : Low-pass



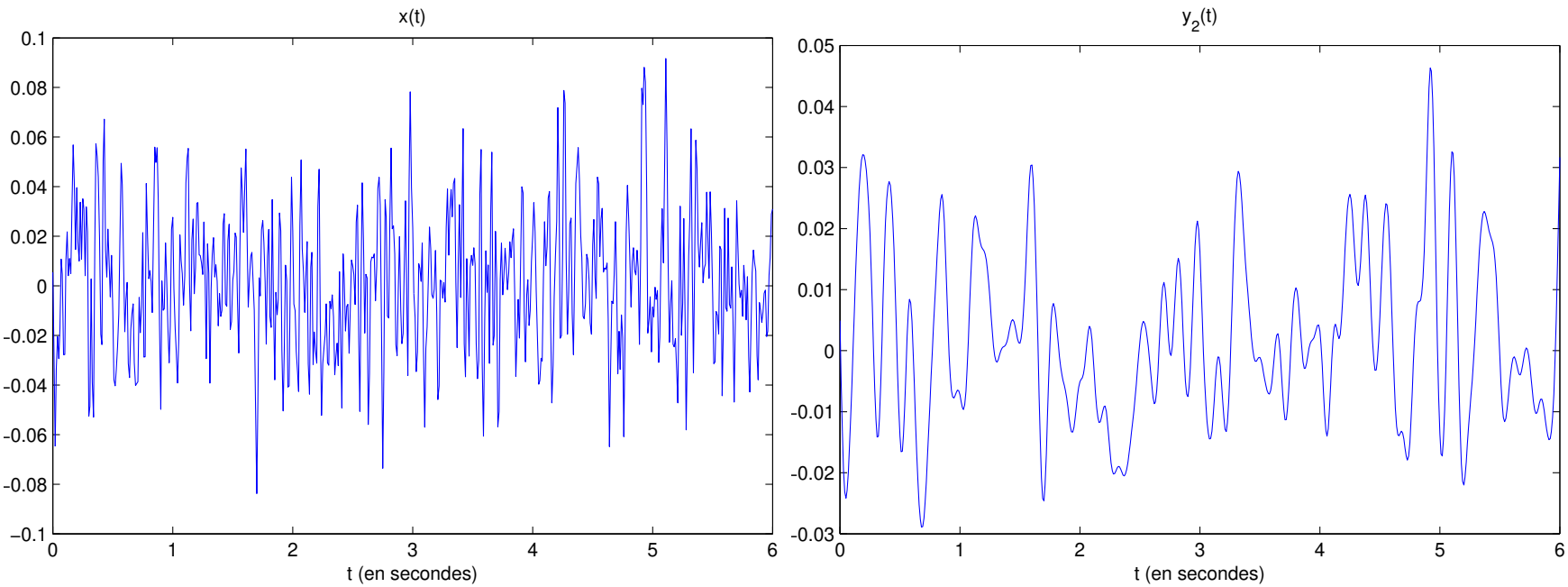
Low-pass filter
Bandwidth $B \approx [0, 7]$ Hz

Example 2 : Low-pass



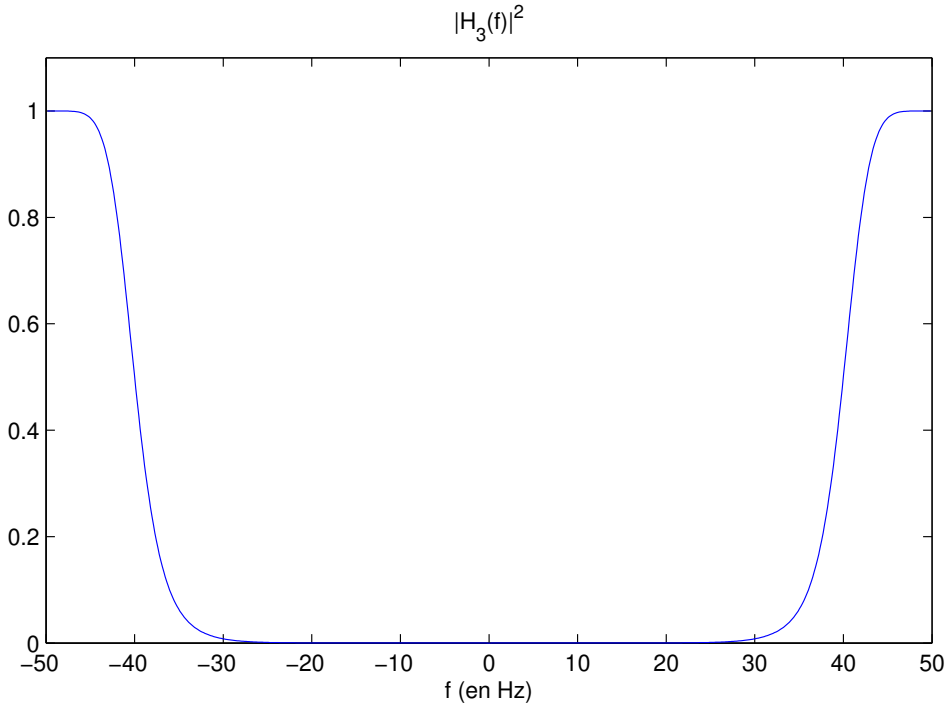
When we look at the spectrum of the filter output, we can see that all frequencies outside the bandwidth have been attenuated

Example 2 : Low-pass



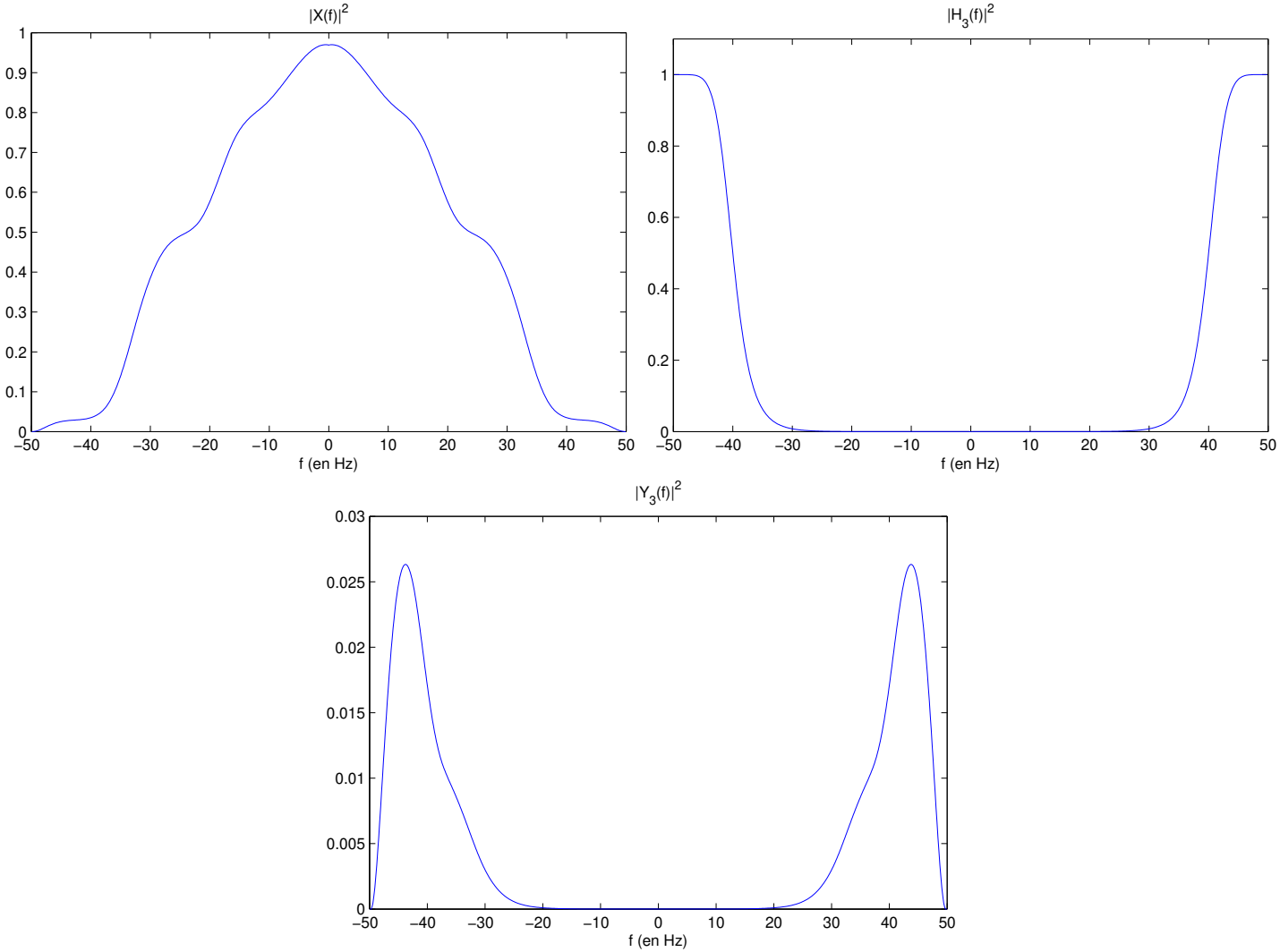
The signal is slightly smoothed: high frequencies have been removed

Example 3 : High-pass



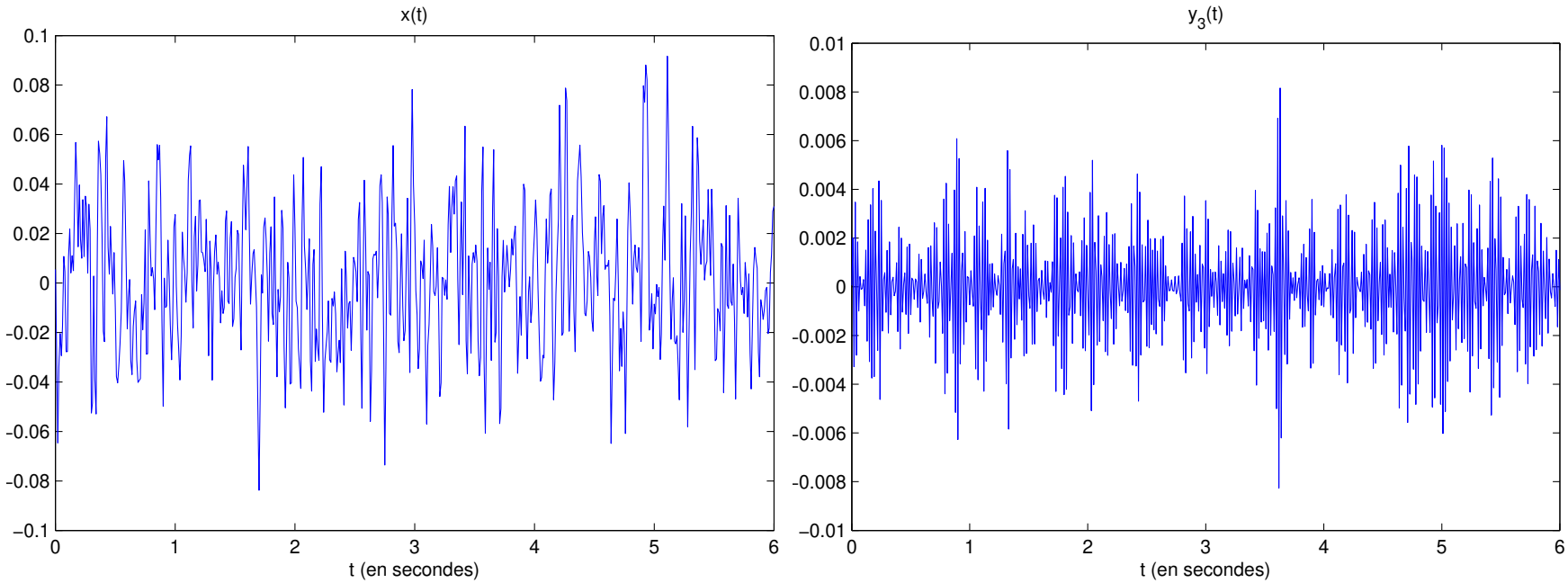
High-pass filter
Bandwidth $B \approx [40, 50]$ Hz

Example 3 : High-pass



When we look at the spectrum of the filter output, we can see that all frequencies outside the bandwidth have been attenuated

Example 3 : High-pass



Only fast variations of the signal have been kept: the low frequencies have been removed

How and when to use a filter

- Low-pass: Smooth the signal; denoise a signal
- High-pass: Highlight punctual events, discontinuities, impulses; remove continuous components or trends.
- Band-pass: Recover a signal emitted in a given frequency band (e.g. alpha waves in EEG); denoise a signal whose bandwidth is known.
- Band-reject: Remove a component from a mixed signal (e.g. 50Hz or 60Hz A.C.); perform source separation.

From ideal filters to digital filters

- Most of the filters used in practice are digital filters of the form

$$a_0 y[n] = \sum_{i=0}^{q} b_i x[n-i] - \sum_{j=1}^{p} a_j y[n-j]$$

- These filters will not achieve the perfect performances of the ideal filters, but only approximation. But the output can be computed with only a few elementary operations
- Two popular solution :
  - Moving average filters (only for low-pass)
  - Butterworth filters (which are approximations of the analog filter)

Moving average filters

- Equivalent to low-pass filter
- One parameter: the length of the filter L

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k]$$

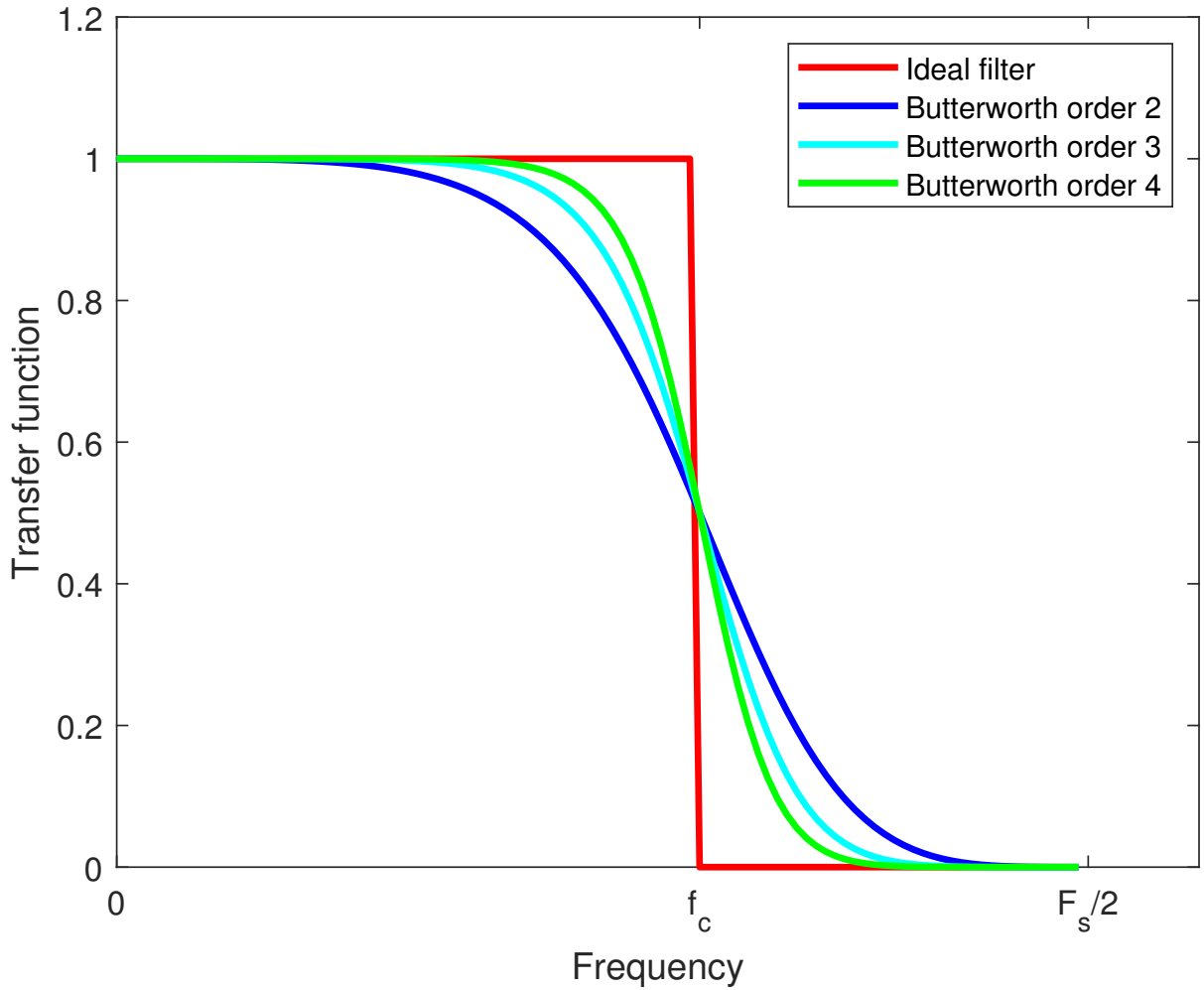
- Low-pass filter with cut-off frequency

$$f_c \approx \frac{0.442947 \times F_s}{\sqrt{L^2 - 1}}$$

Butterworth filters

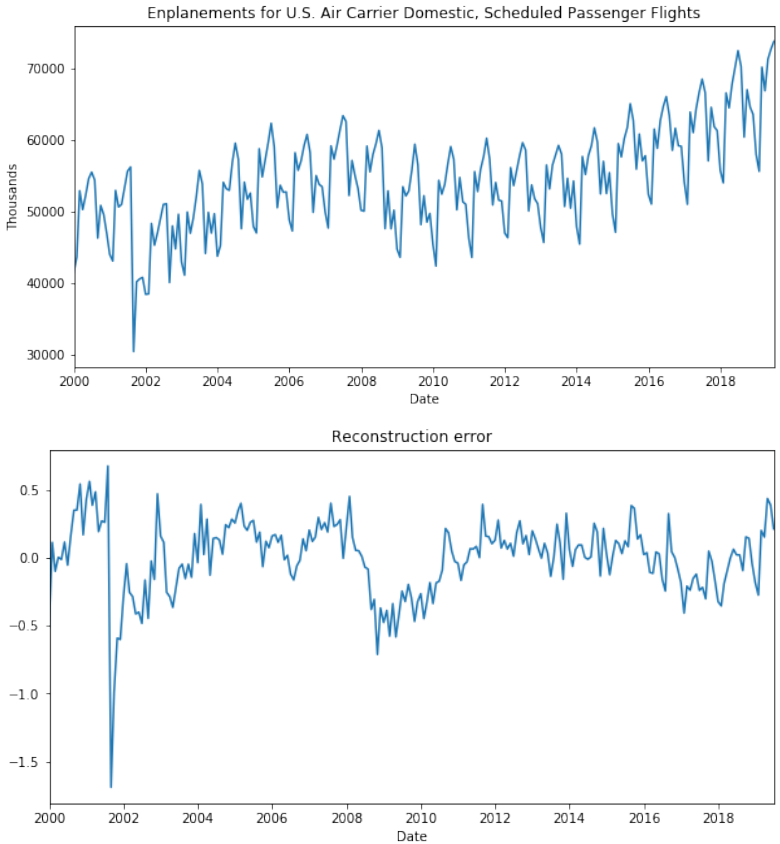
- Two parameters: the order $p = q$ and the cut-off frequency
- Can also be used for high-pass, band-pass, etc.
- The larger the order, the closer to the ideal filter: most of the time the order is kept quite small (except when e.g. the bandwidth is very tight)

Butterworth filters



Contents

Statistical vision



Would it make sense to compute the empirical mean of these time series ?

- Some properties of the signals are also linked to the distributions of values (and their evolution with time)
- In order to take that into account, we will now focus on the statistical properties of the signal

Statistical vision

- In order to better understand the properties of a signal, deterministic analysis such as Fourier has been extended to probabilistic and statistical analysis
- In this context, we assume that $x[1:N]$ corresponds to a realization of a **stochastic process** $X[1:N]$
- Each $X[n]$ can be seen as a random variable, with a (possibly unknown) probability distribution
- Statistical properties of $X[1:n]$ can be retrieved from estimates based on $x[1:n]$

Two fundamental properties: stationarity

- **Stationarity :** The statistical properties of the time series do not change over time
  - Order 1
$$\forall n, \quad \mathbb{E}\left[X[n]\right] = \mu$$
  - Order 2
$$\forall n_1, n_2, \quad \mathbb{E}\left[X[n_1]X[n_2]\right] = \gamma_X[|n_2 - n_1|]$$
  - Order 1 + Order 2 $\rightarrow$ wide-sense stationarity (most common assumption)

Two fundamental properties: ergodicity

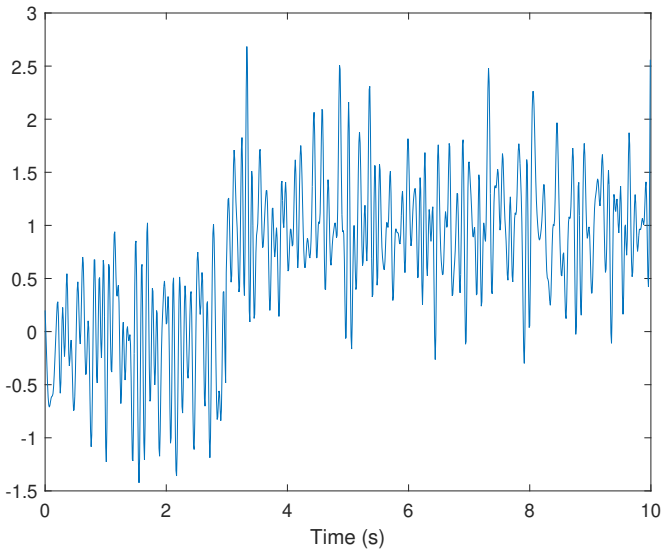
- **Ergodicity :** Statistical properties can be retrieved from temporal properties. Assuming that the time series is wide-sense stationary:
  - Order 1

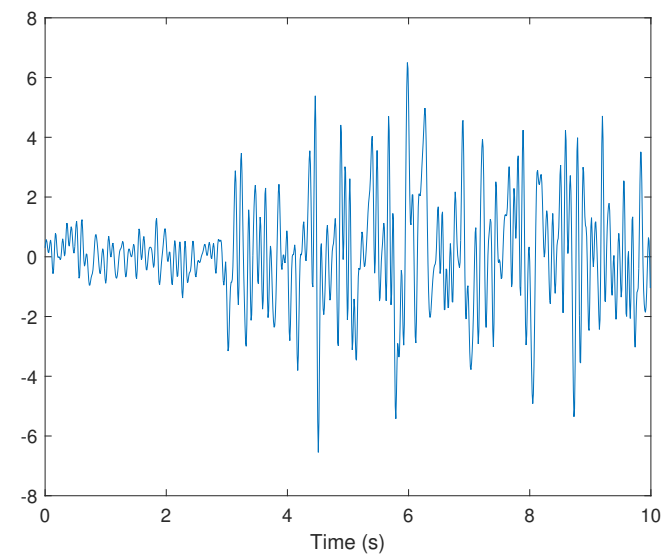
$$\forall n, \quad \mathbb{E}\left[X[n]\right] = \mu = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} x[n]$$

  - Order 2

$$\forall n_1, n_2, \quad \mathbb{E}\left[X[n]X[n+k]\right] = \gamma_X[k] = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} x[n]x[n+k]$$

Stationarity vs. non-stationarity





- None of these signals are stationary: any statistical features computed on the whole time series will be wrong
- In order to prevent this to happen, two solutions exist
  - Divide the signals into small frames where the signal is assumed to be stationary and ergodic
  - Use a change-point detection algorithm to detect these changes and work separately on each segment (see later)

Autocorrelation

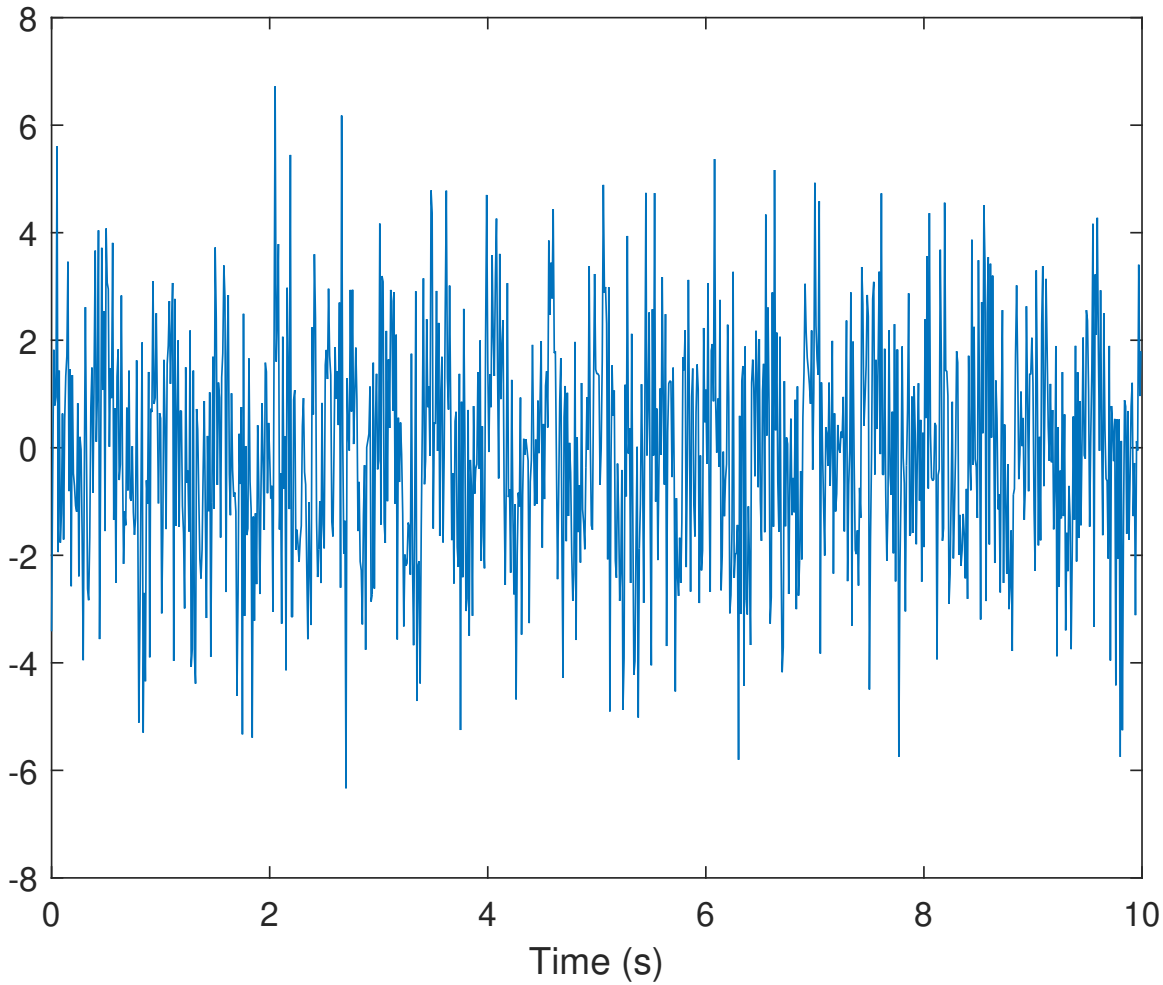
Assuming that $X[1:n]$ is ergodic and wide-sense stationary, we can estimate from $x[1:n]$ the autocorrelation function

- Autocorrelation function

$$\hat{\gamma}_x^{biased}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n+m] \quad \text{where } x[n] = 0 \text{ for } n \neq 0 \dots N-1$$

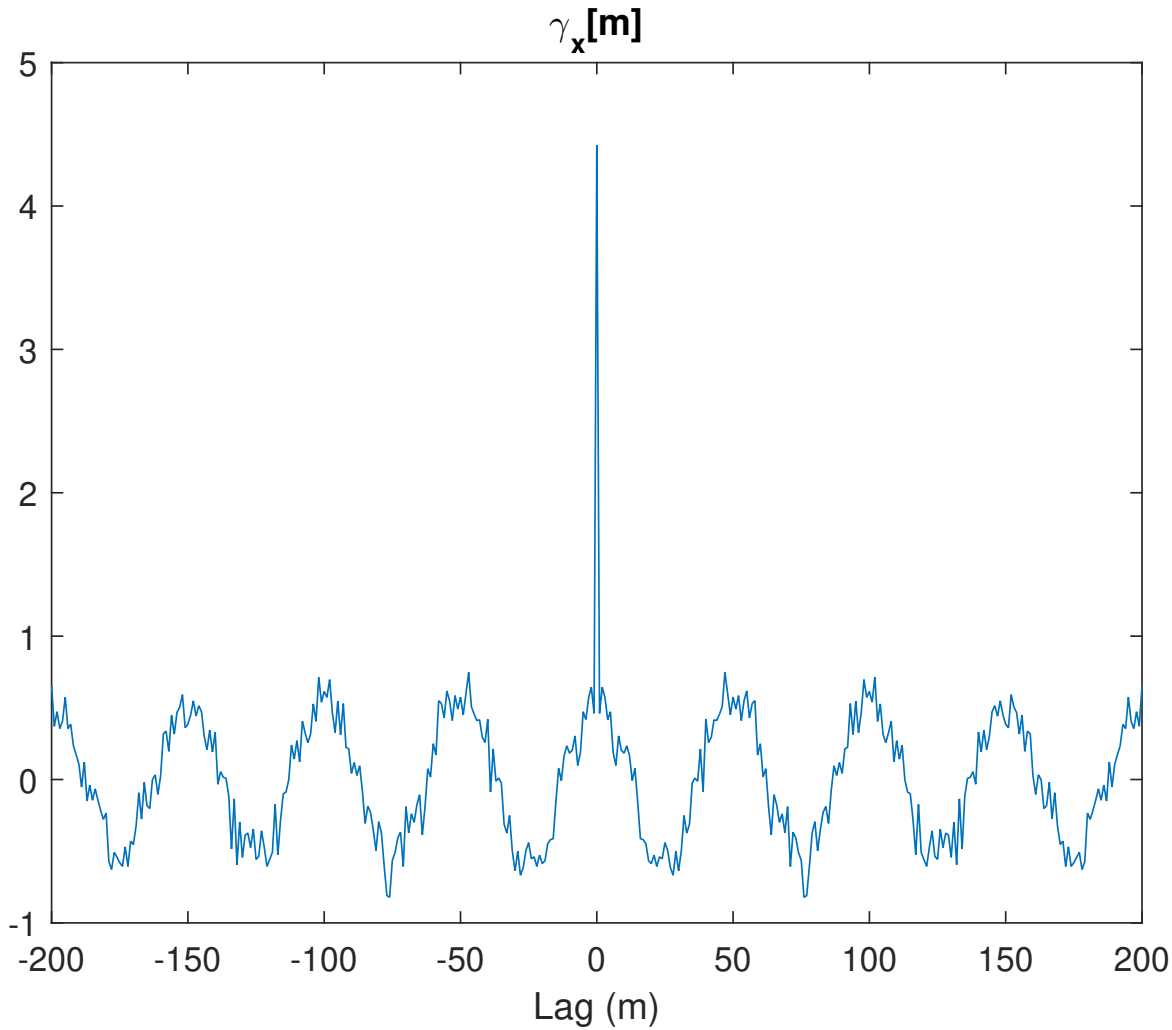
- This function helps (among other things) to discover the presence of periodic components within a signal

How to use the autocorrelation function



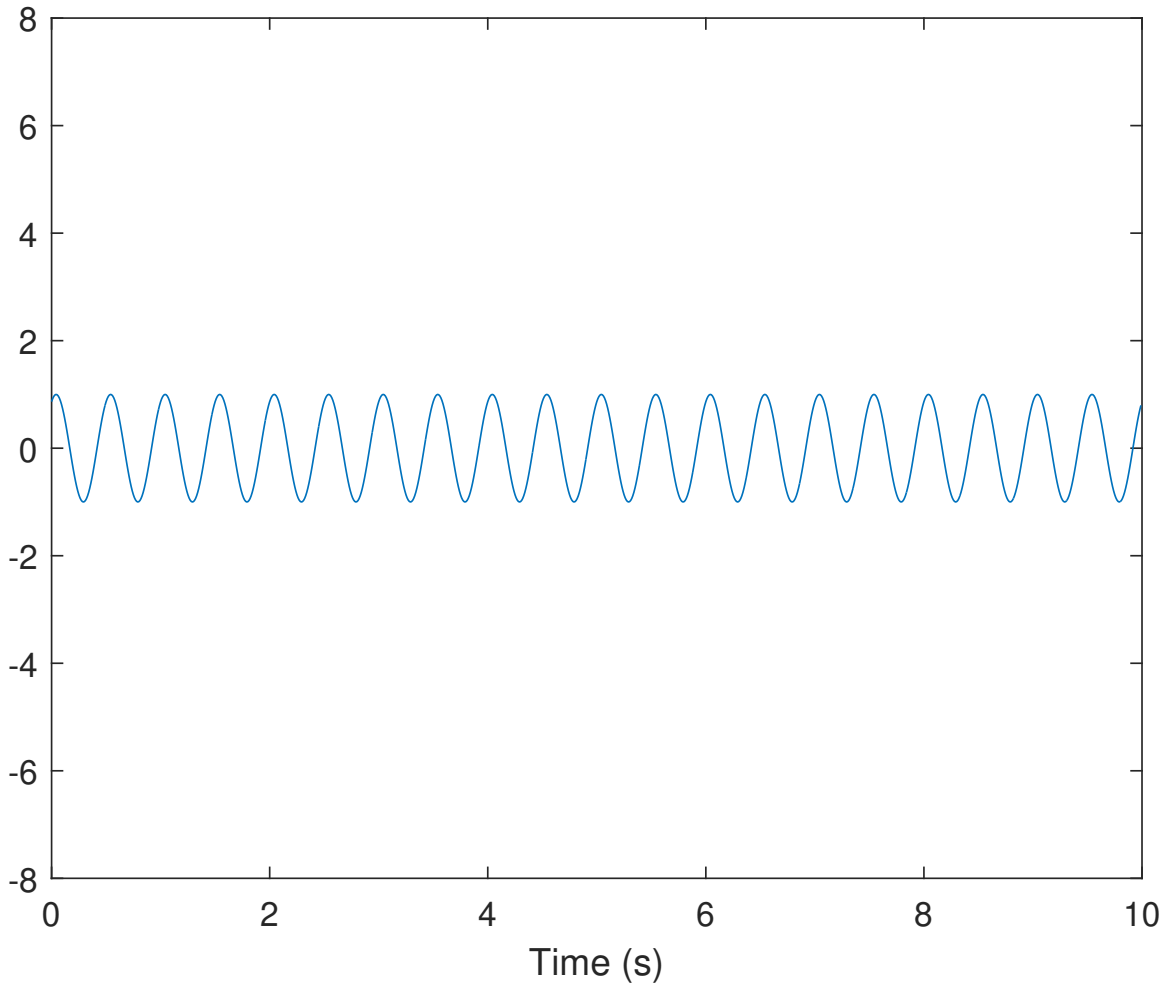
Original signal, sampling frequency 100 Hz

How to use the autocorrelation function



Autocorrelation function, peaks are visible for lags multiple of 50

How to use the autocorrelation function



A periodic signal with period $50 \times \frac{1}{100} = 0.5$ sec was hiding!

Statistical features

Assuming that $X[1:n]$ is ergodic and wide-sense stationary, we can estimate several statistical properties from $x[1:n]$

- Mean

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x[n]$$

- Autocorrelation function

$$\hat{\gamma}_x^{biased}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n+m] \quad \text{where } x[n] = 0 \text{ for } n \neq 0 \dots N-1$$

$$\hat{\gamma}_x^{unbiased}[m] = \frac{1}{N-|m|}\sum_{n=0}^{N-1} x[n]x[n+m] \quad \text{where } x[n] = 0 \text{ for } n \neq 0 \dots N-1$$

Features from autocorrelation function

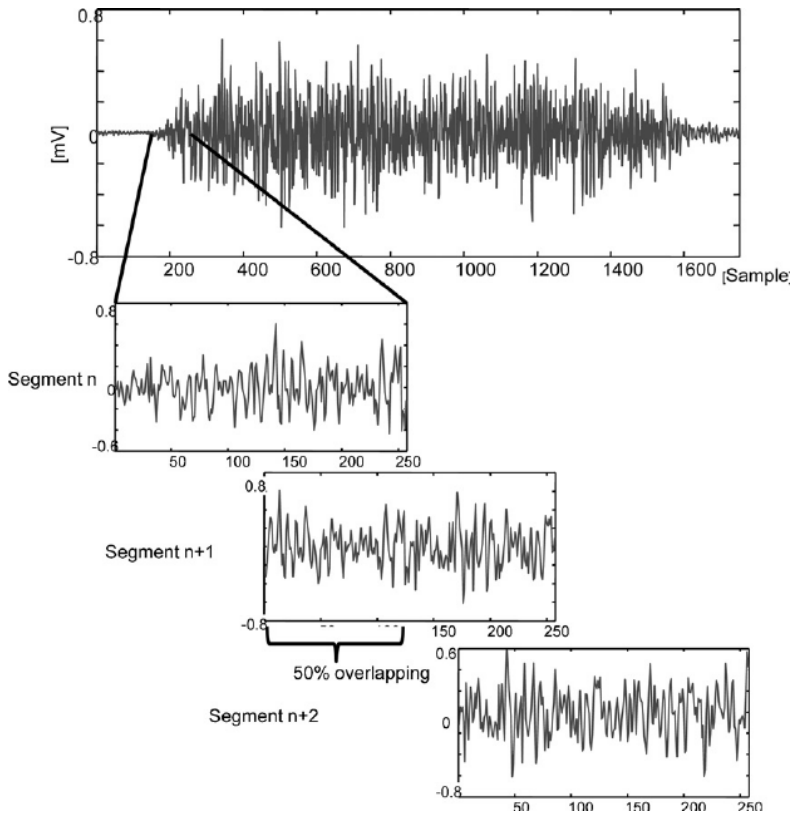
- Given a lag m, one useful feature is the renormalized autocorrelation coefficient

$$\gamma'_X[m] = \frac{\gamma_X[m]}{\gamma_X[0]}$$

  that quantifies the correlation between two samples spaced by m in the time series
- These features are very useful when the signal has a seasonality or periodic component: in this case, we store $\gamma'_X[m_0]$ where m_0 corresponds to the index of the first peak of the autocorrelation function ($m_0 > 0$)

Other statistics

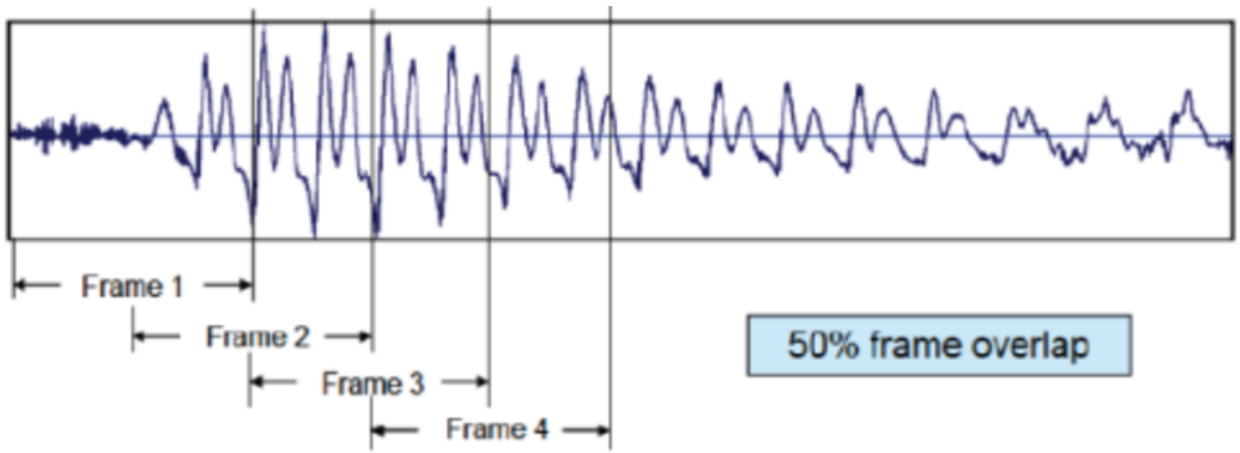
- Other quantities can be estimated such as minimum and maximum values, or robust statistics such as median or percentiles (5%, 25%, etc.)
- Most of the time, these purely statistical features remove the time information...
- If necessary, those quantities can be extracted on sliding windows, increasing the number of features while preserving the time information



Contents

Spectrogram

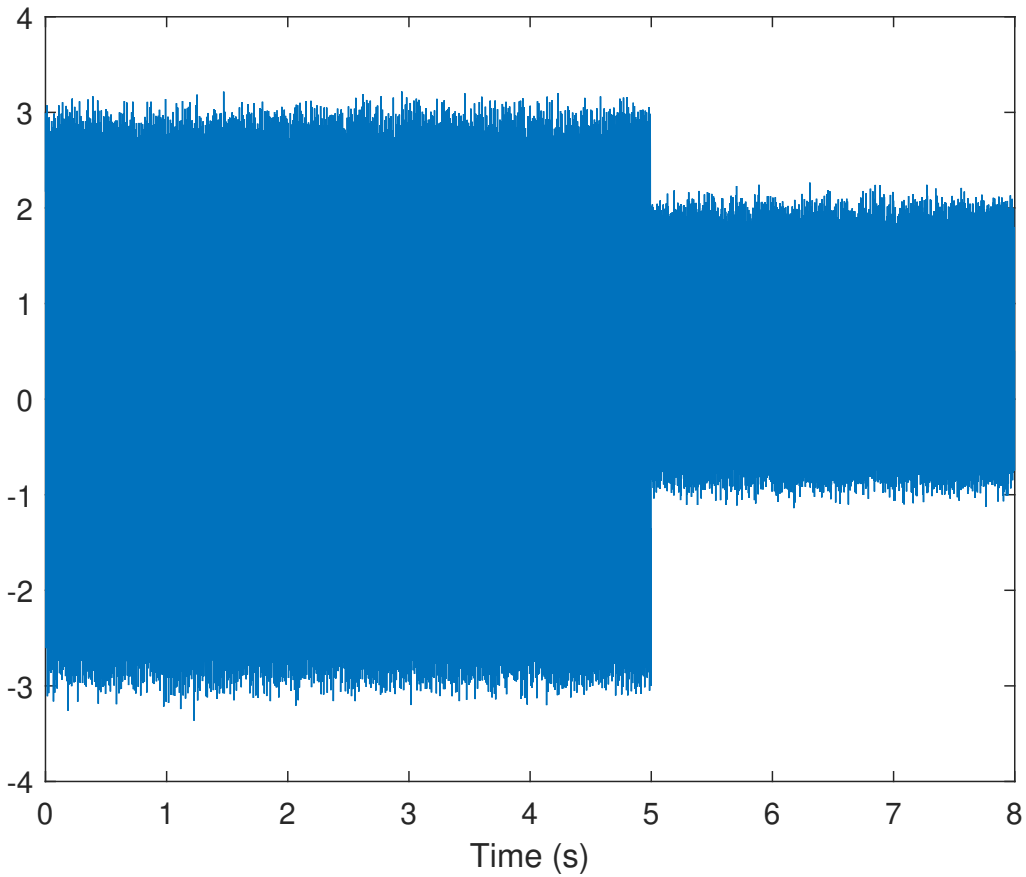
- When the properties of the time series tend to change with time (non-stationary signals), it is more careful to compute the DFT on sliding windows
- By sliding the window along the signal, we recover a time-frequency representation called **spectrogram**



- Matrix representation: each column corresponds to the DFT on the window of interest.

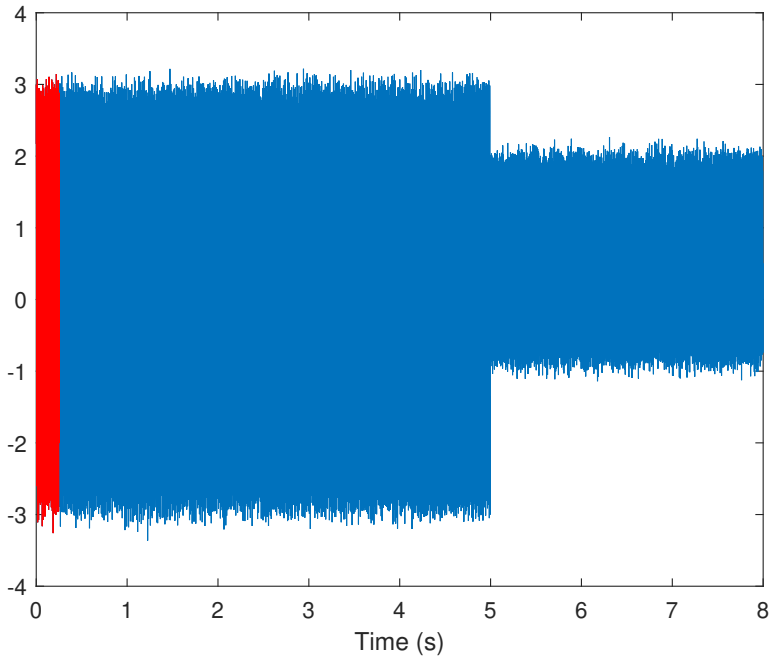
X-axis: frame number, Y-axis: frequency bin

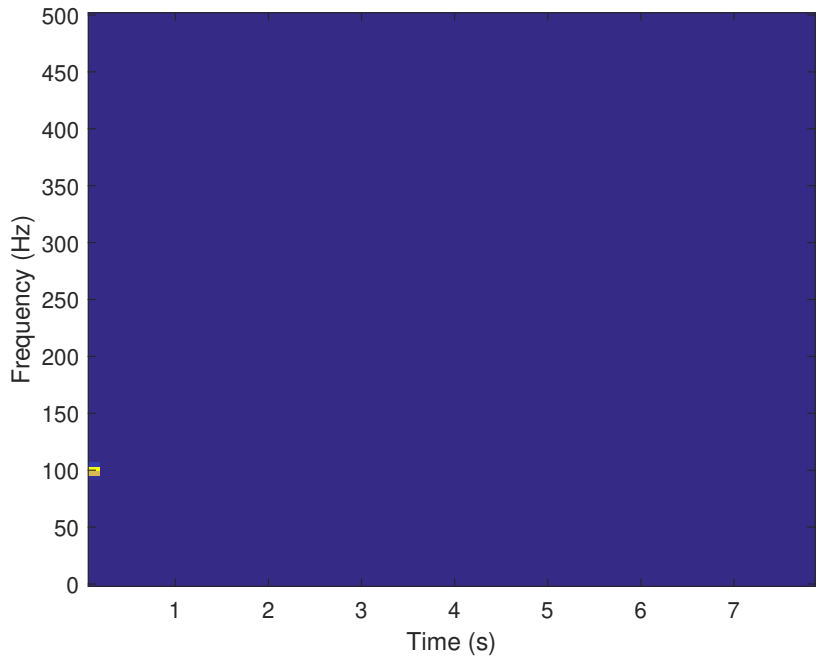
Example



Original signal, F_s = 1000 Hz

Example

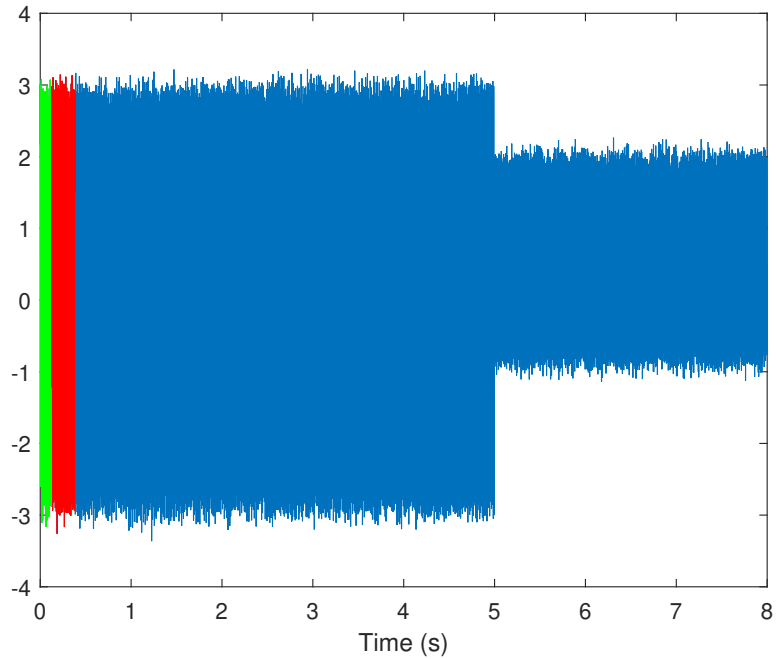


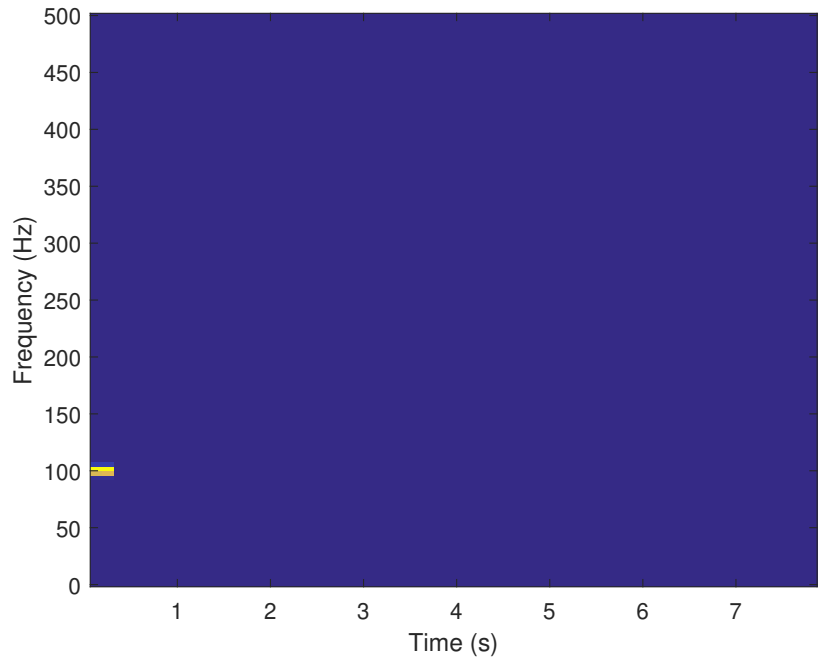


Window length $N_w = 256$

Computation of the DFT on the first frame and storage in the spectrogram matrix...

Example

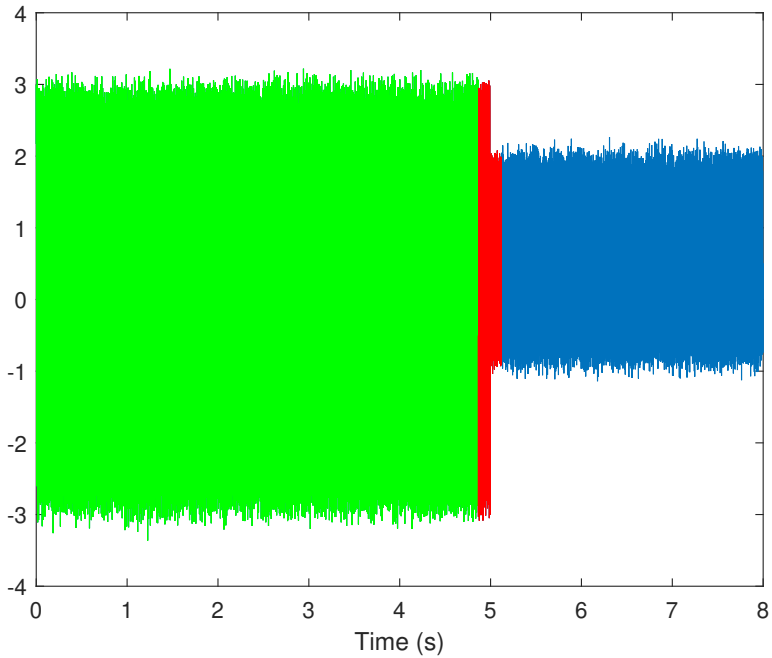


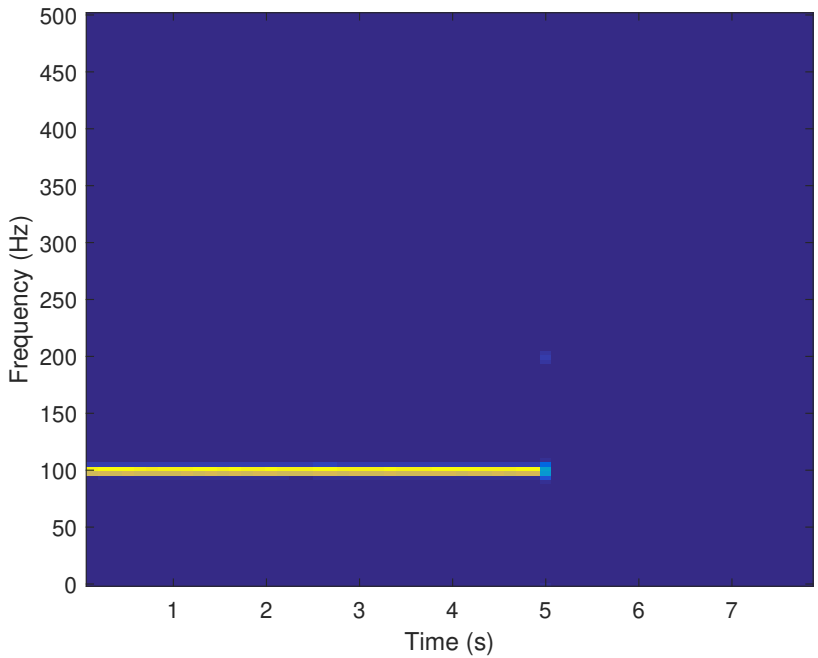


Window length $N_w = 256$

Computation of the DFT on the second frame and storage in the spectrogram matrix...

Example

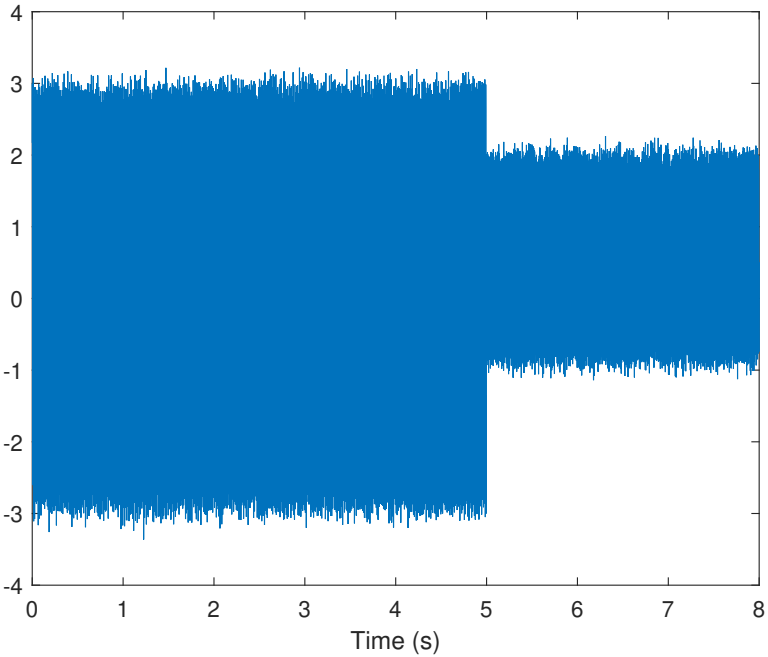


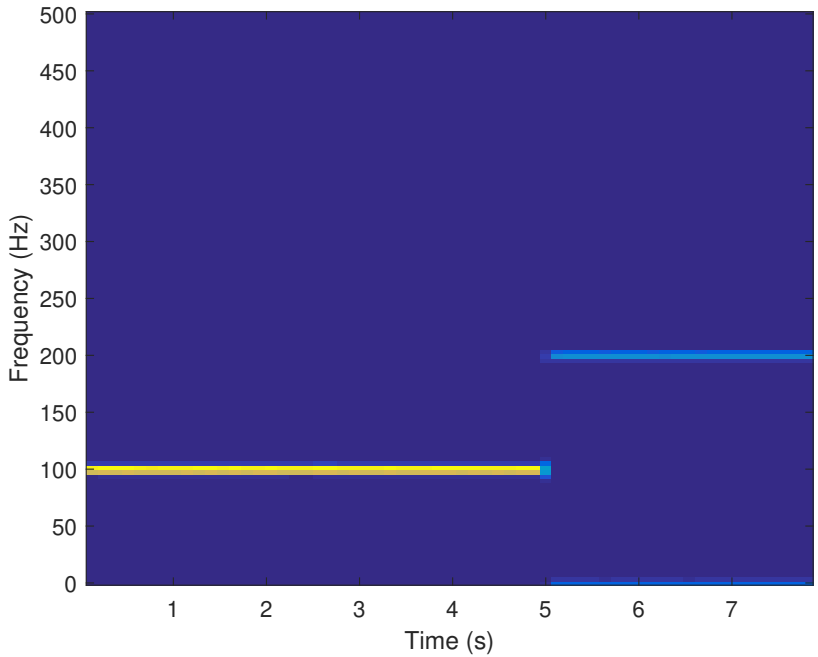


Window length $N_w = 256$

Same process...

Example





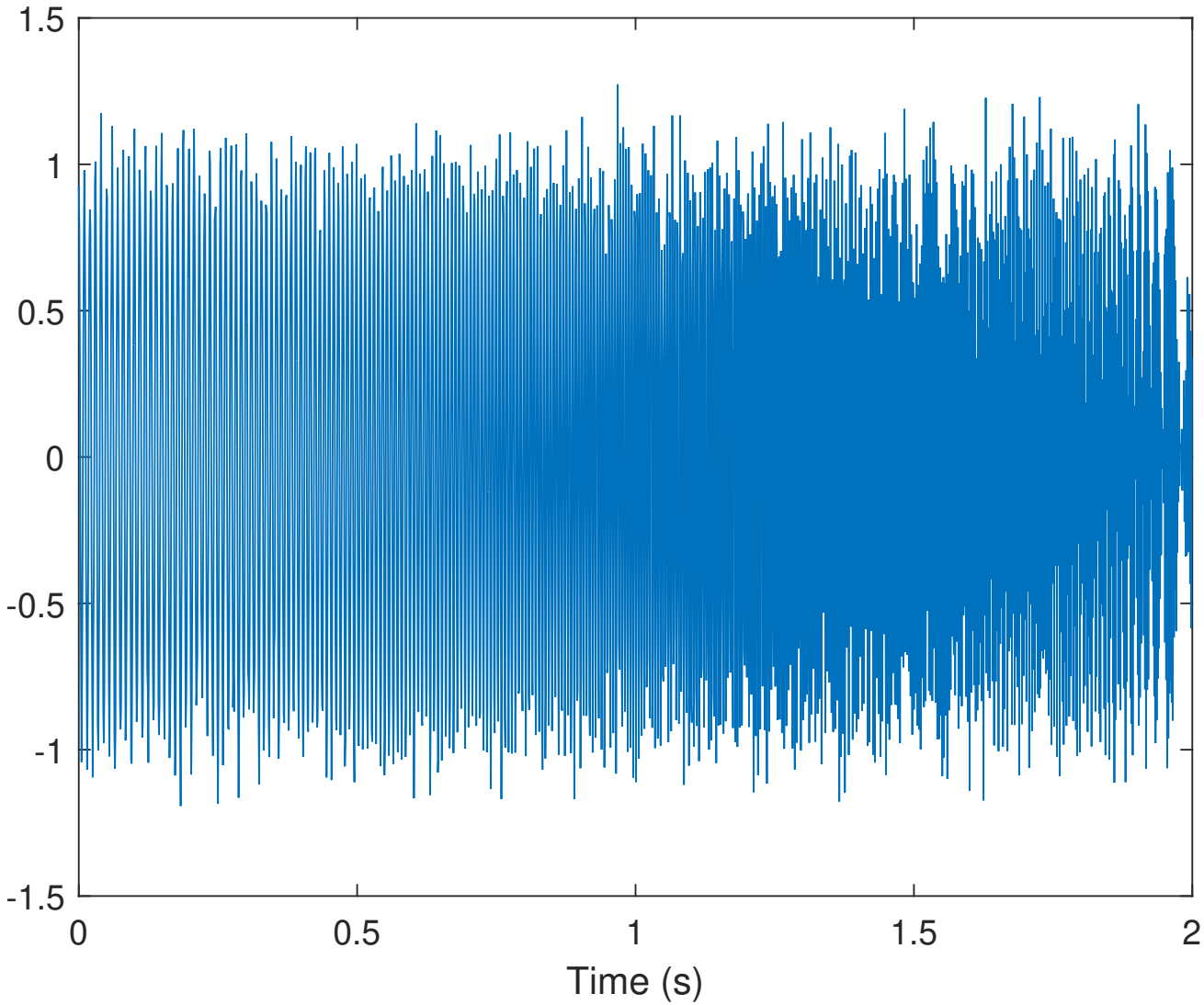
Window length $N_w = 256$

Final result

DFT vs. Spectrogram

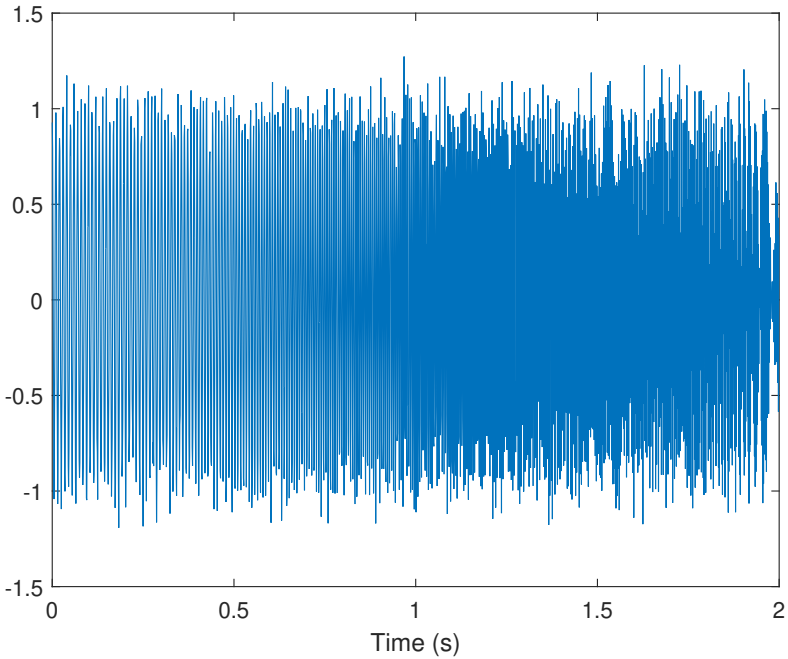
- Only use DFT when you are sure that there is no abrupt changes in the time series
- Note that using DFT will tend to average the frequency content on the whole time series, which can be tricky in some application contexts
- For safety, always first visualize the spectrogram to make sure that no significant changes occur

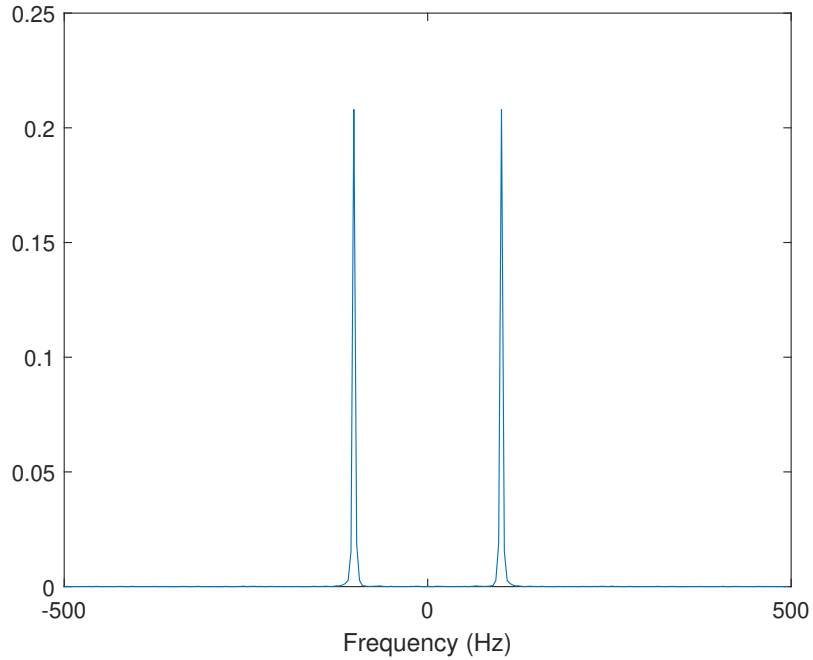
DFT vs. Spectrogram



Periodic phenomenon ? (sinusoid ?)

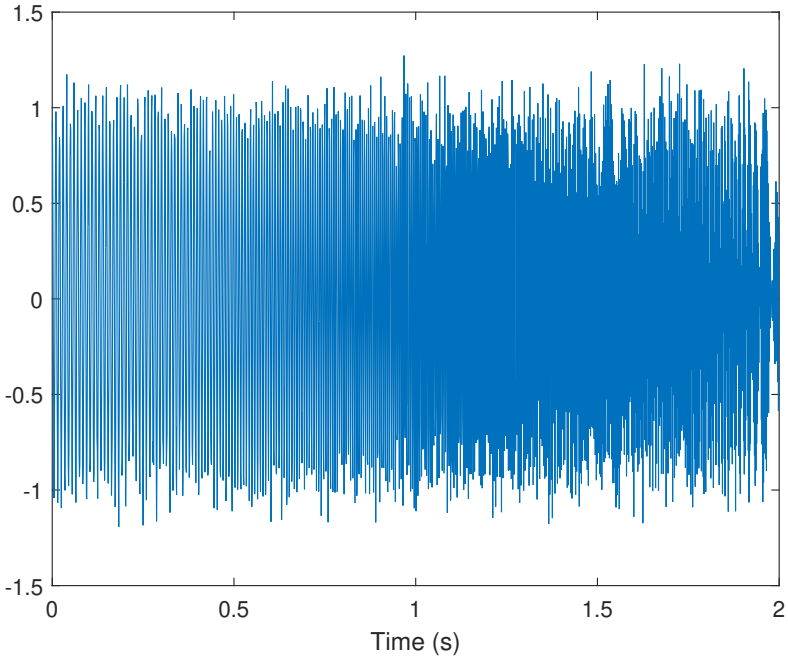
DFT vs. Spectrogram

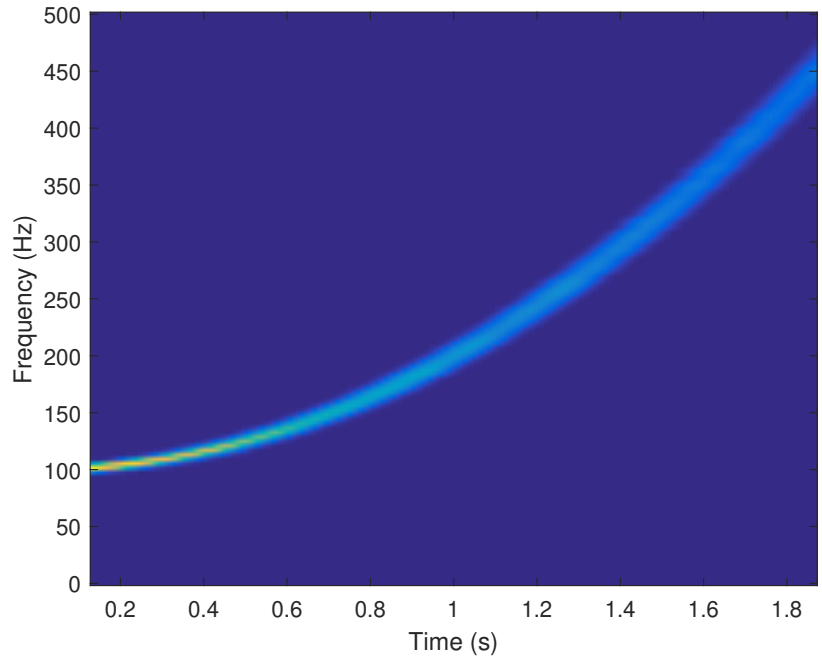




DFT suggests a sinusoidal phenomenon around frequency 100 Hz

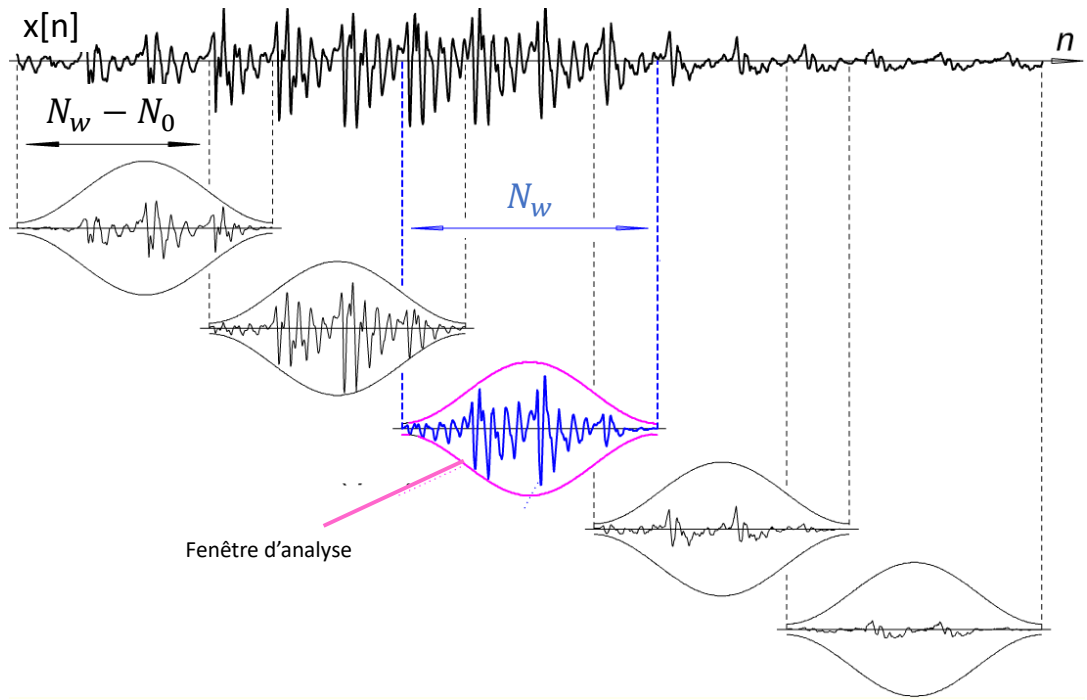
DFT vs. Spectrogram





In fact, chirp signal between 100 and 500 Hz !!

Hyperparameters for spectrogram



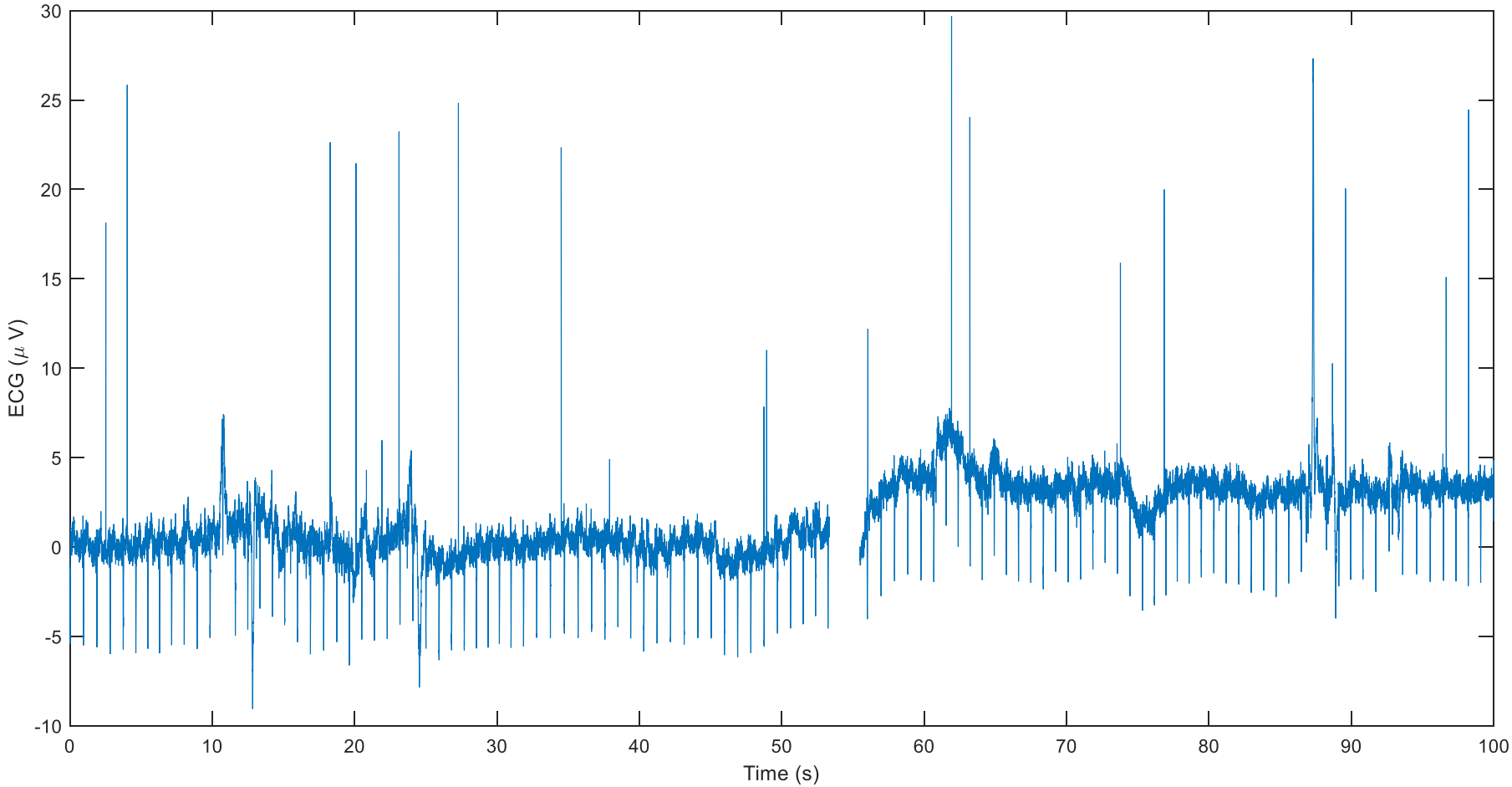
- N_w : window length (in samples)
  Often taken as a power of 2 (for FFT) and linked to the desired frequency resolution.
- N_o : overlap between two successive frames (in samples)
  Often taken as 50% or 75% of the window length and characterizes the time resolution (optimal when $N_o = N_w - 1$)
- w : analysis window (Hann, Hamming, Blackman...)
  Traditionally, in order to limit side effects, the signal frame is multiplied by an analysis window

Contents

The need for preprocessing

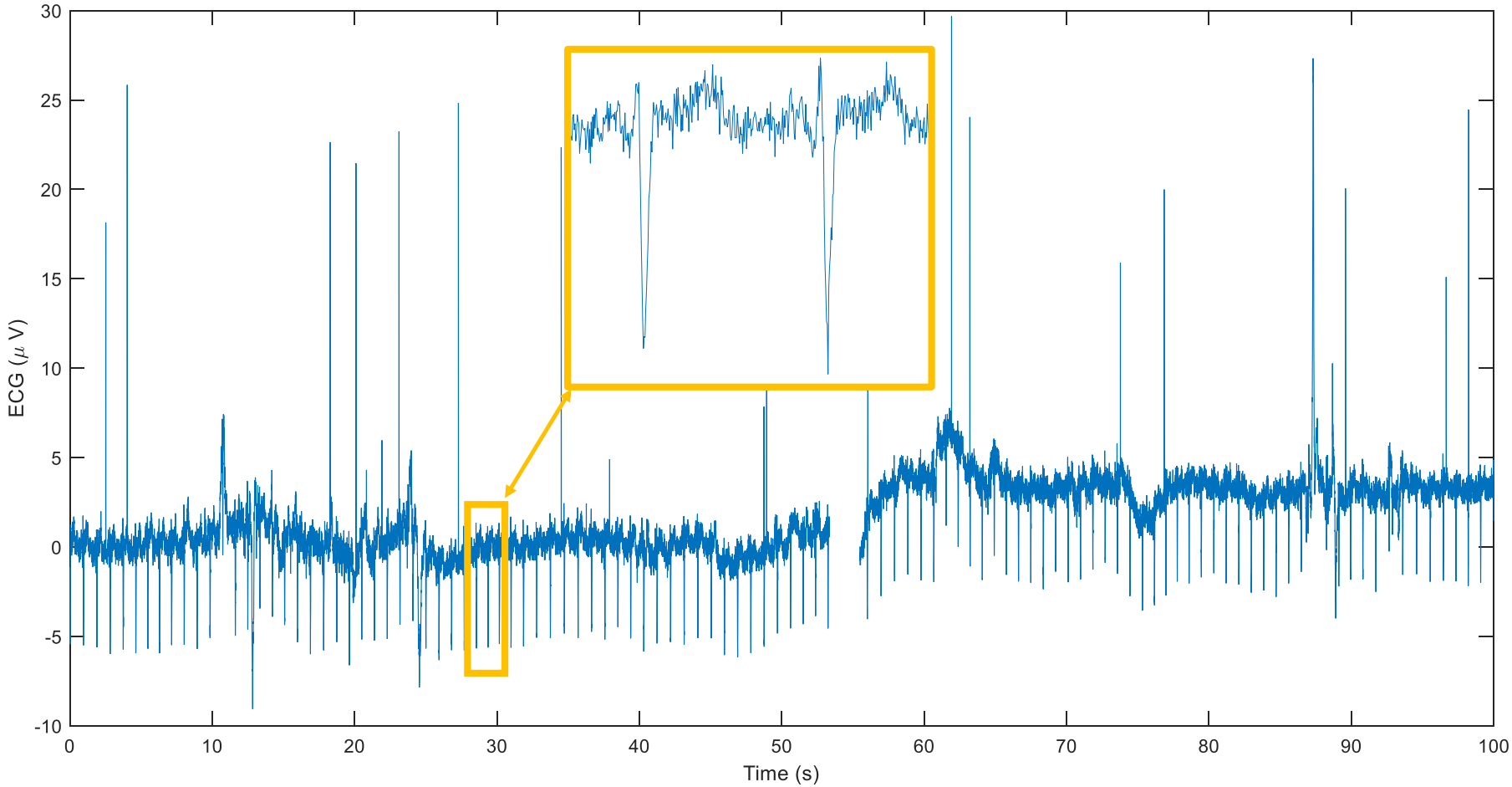
- Typical usecase: noisy time series with outliers and missing values
- In order to apply algorithms and to extract meaningful events, the data scientist needs to *clean* and *consolidate* the data
- Time-consuming and tedious task: fortunately, there are many tools that can be used
- Careful! All these preprocessing have a strong **impact** on the expected results and on the future learned rules!

Introductory example



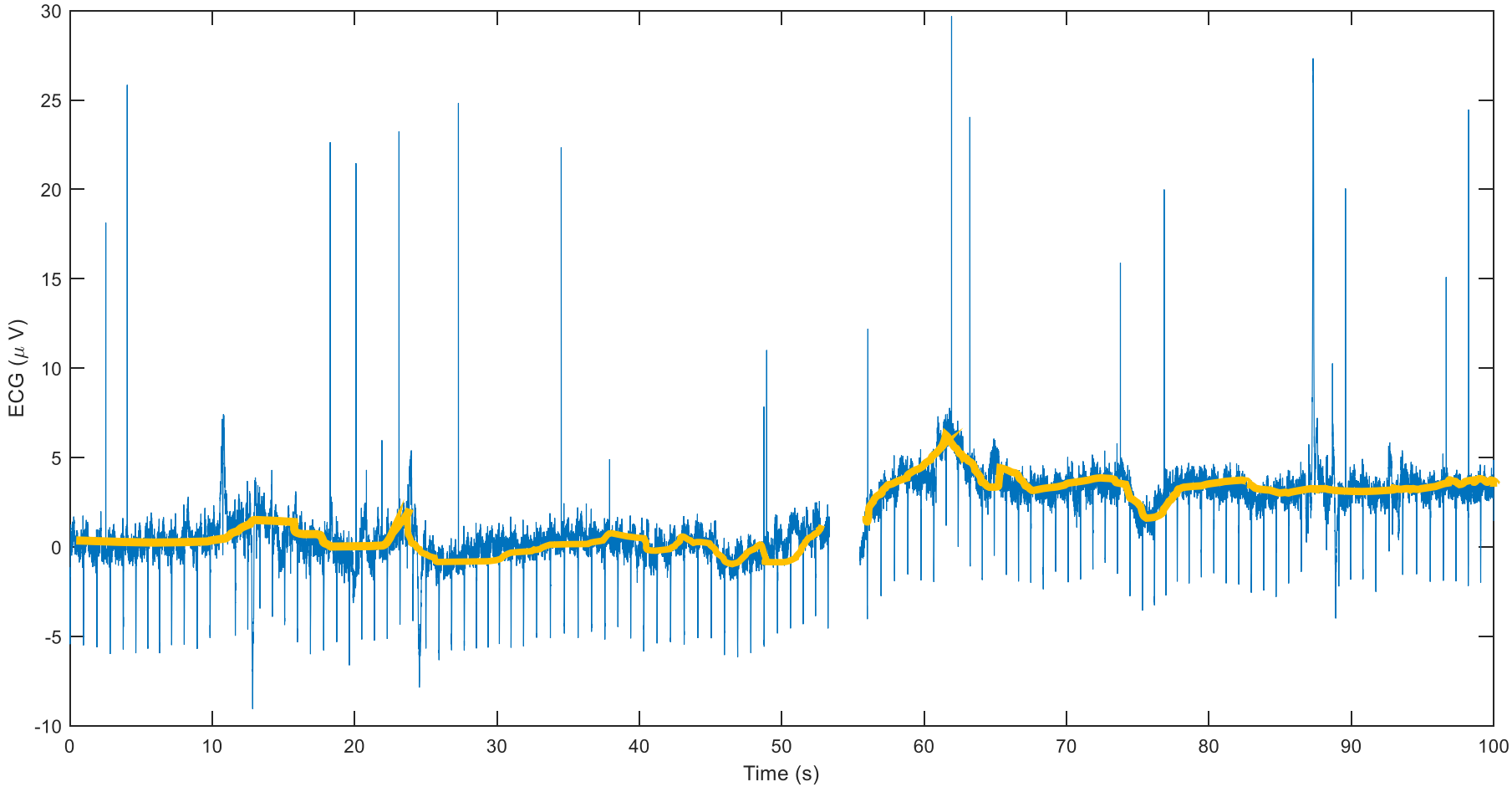
ECG signal during general anesthesia

Introductory example



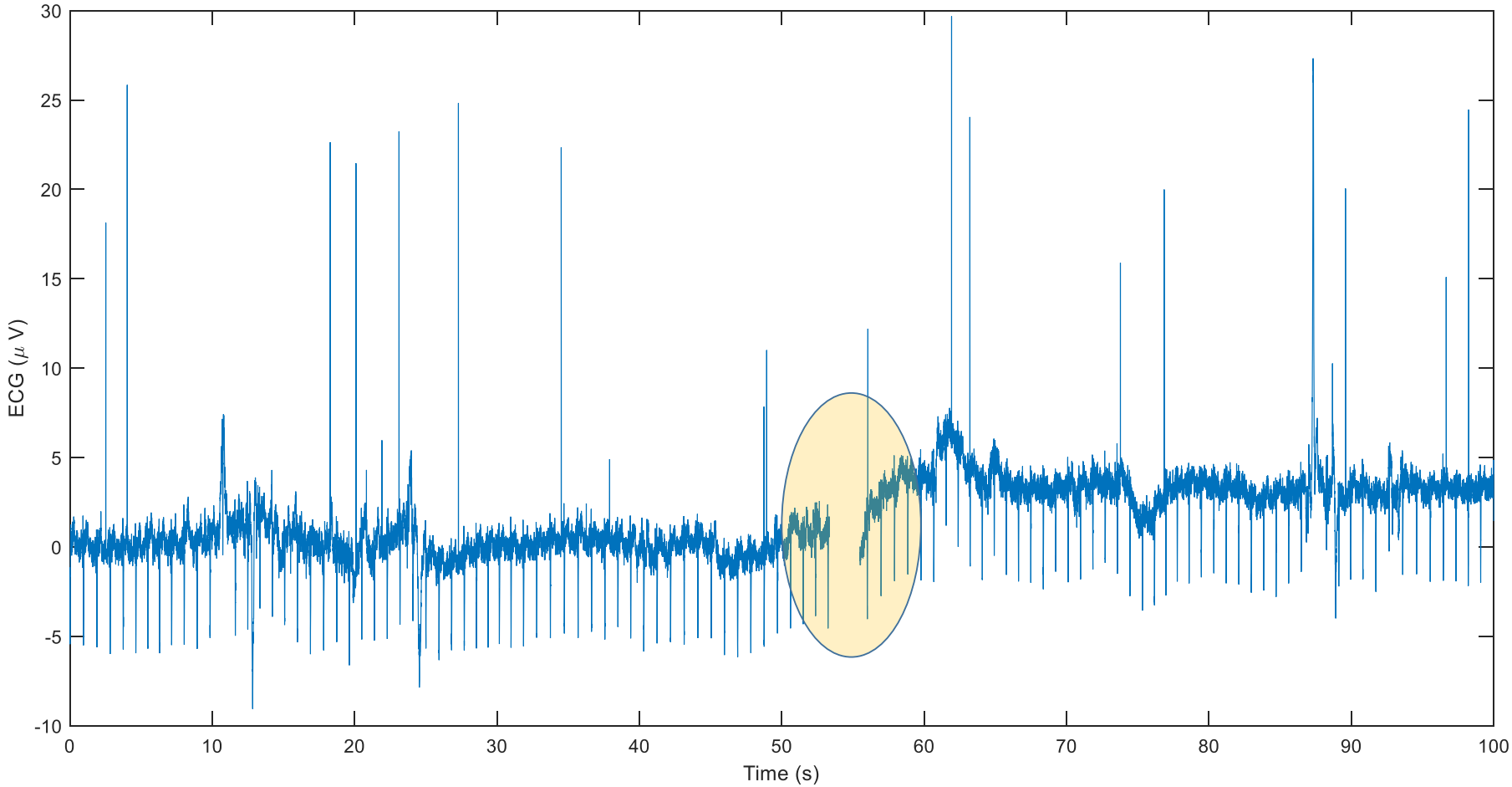
Presence of measurement noise $\rightarrow$ **Denoising**

Introductory example



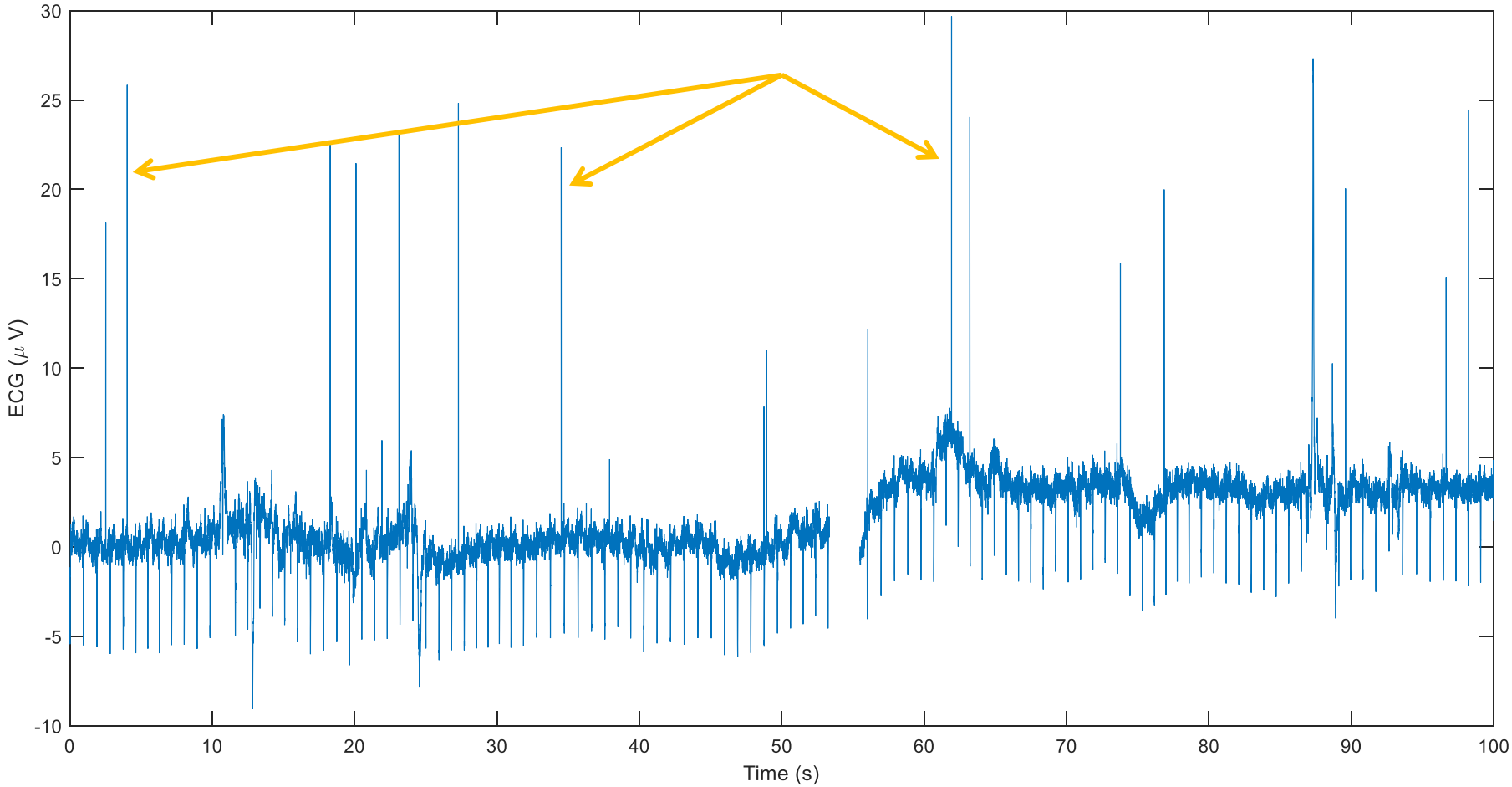
Presence of a trend $\rightarrow$ **Detrending**

Introductory example



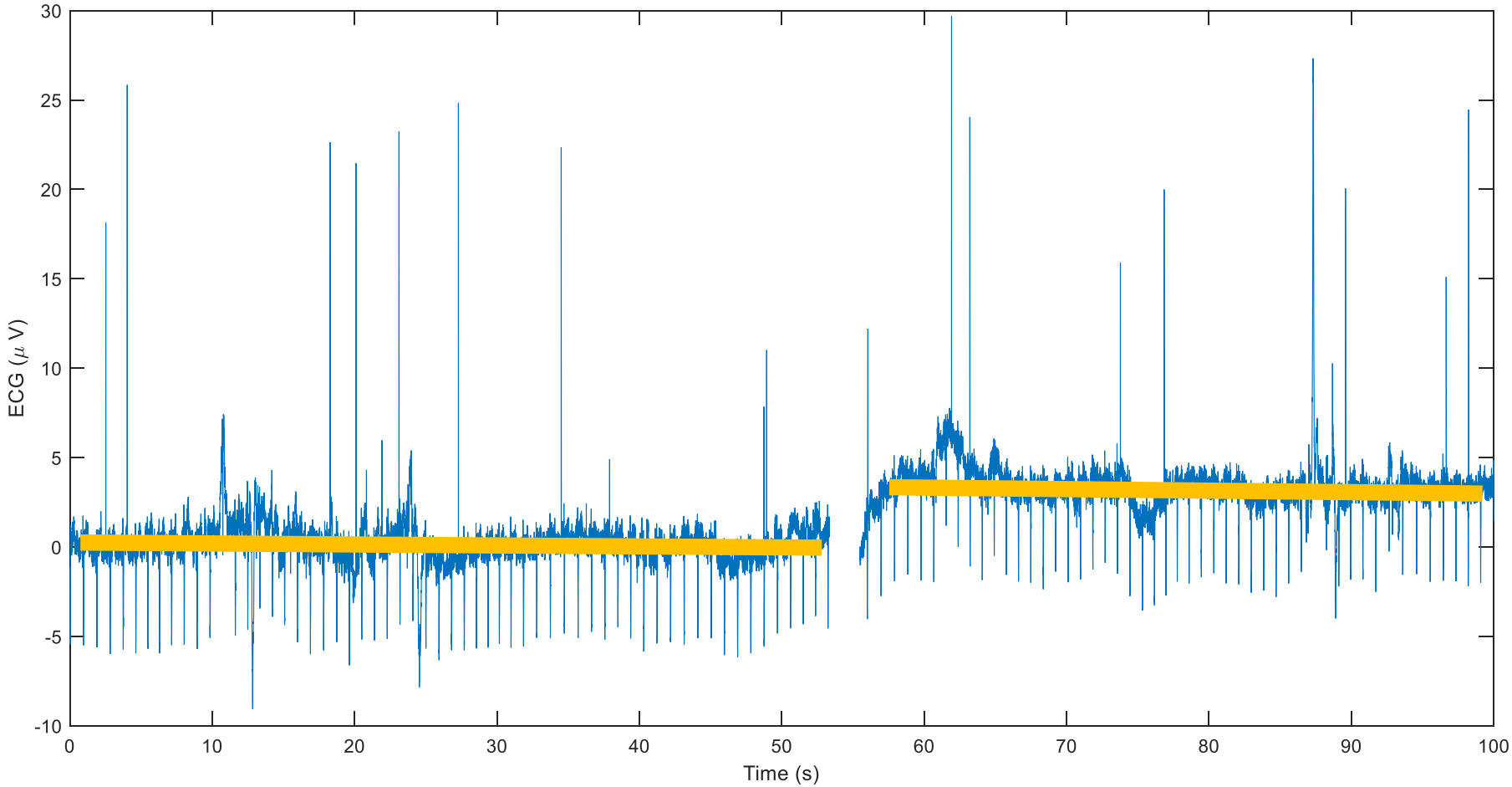
Data loss causing missing samples $\rightarrow$ **Interpolation**

Introductory example



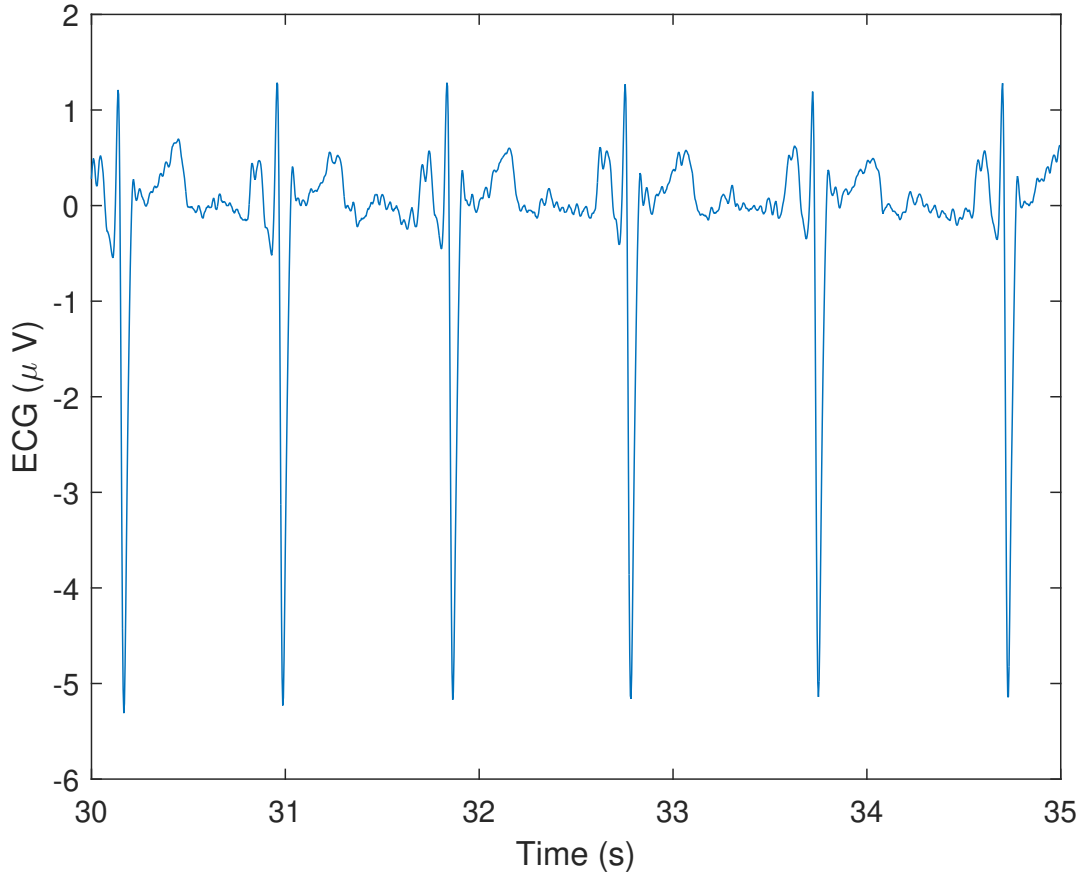
Presence of outliers $\rightarrow$ **Outlier removal and suppression of impulsive noise**

Introductory example



Break in stationarity $\rightarrow$ **Change-point detection**

Introductory example



When all preprocessings have been performed, it becomes possible to retrieve the heartbeats and thus to perform ML

Contents

Additive white Gaussian noise (AWGN) model

The most common model for noisy signals is

$$y[n] = x[n] + b[n]$$

- $x[n]$ is the clean (unknown) signal
- $b[n]$ is the measurement noise, assumed to be additive, white and Gaussian (AWGN)
- $y[n]$ is the measured signal
- $x[n]$ and $b[n]$ are uncorrelated

Denoising

Given a noisy signal $y[n]$ corrupted by AWGN, retrieve the clean signal $x[n]$

Notion of AWGN

An AWGN $b[n]$ is:

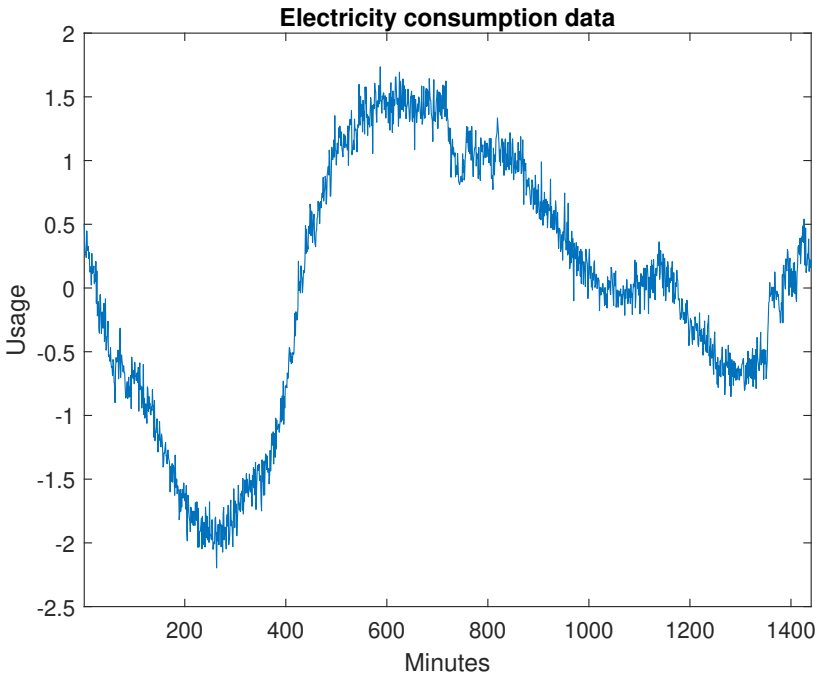
- **Additive**: the noise therefore corrupts all the samples
- **White**: stationary process with zero-mean and all samples are pairwise uncorrelated

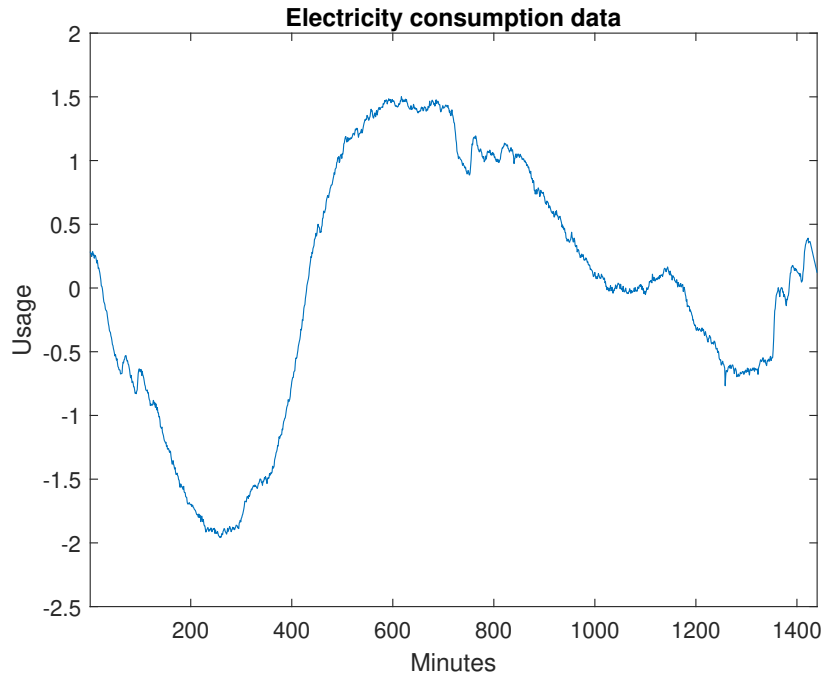
$$\gamma_b[m] = \begin{cases} \sigma^2 & m = 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Gaussian**: all samples are i.i.d. according to

$$b[n] \sim \mathcal{N}(0, \sigma^2)$$

Example





How can we remove the noise component?

Filtering

- The main solution consists in using results from signal processing and statistics
- Knowing that $\gamma_x[m] = \mathbb{E}[x[n]x[n+m]]$ and using the fact that $x[n]$ and $b[n]$ are uncorrelated, we get that

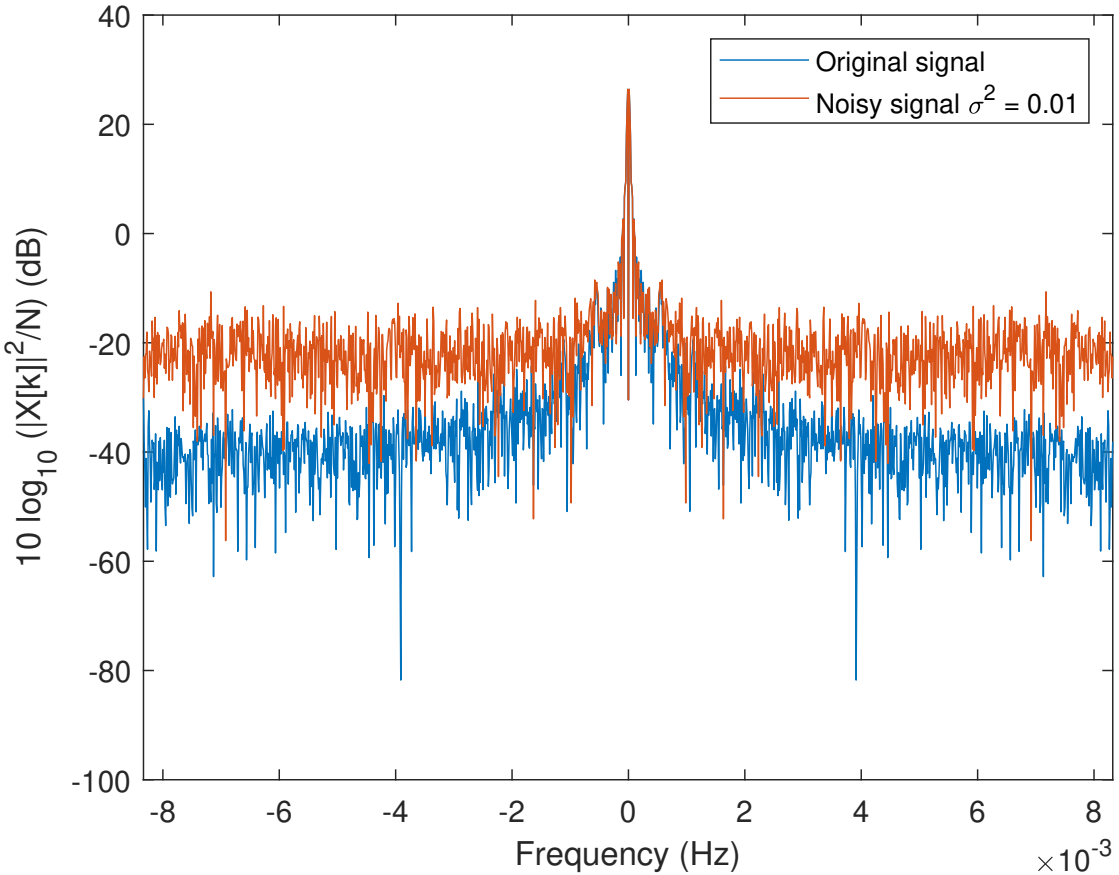
$$\gamma_y[m] = \gamma_x[m] + \gamma_b[m]$$

- By computing the DFT of this equation, we have

$$|Y[k]|^2 = |X[k]|^2 + N\sigma^2$$

- Adding AGWN is equivalent to adding a constant on the DFT of the signal (in linear scale)

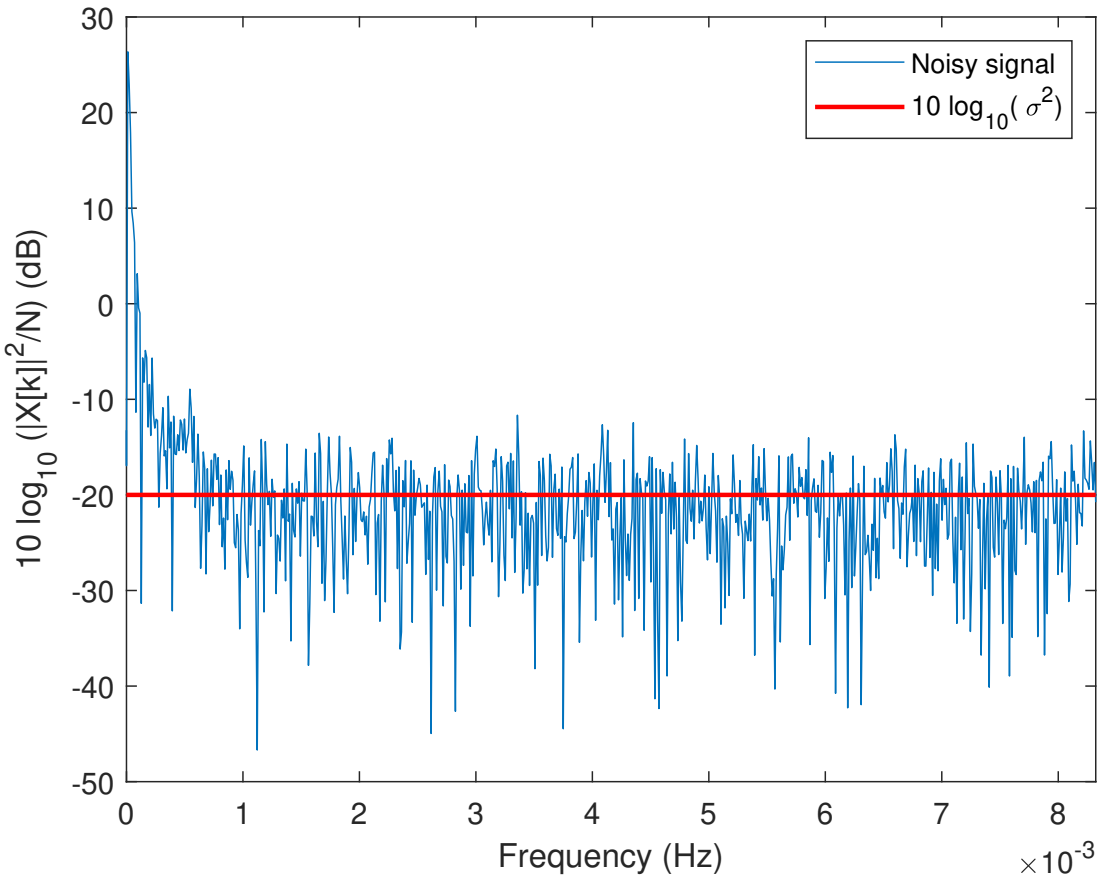
Example



In the frequency band where only AGWN is present (here with $\sigma^2 = 0.01$), the log-spectrum is equal to

$$10\log_{10}\left(\frac{|Y[k]|^2}{N}\right) = 10\log_{10}\left(\frac{|X[k]|^2}{N} + \sigma^2\right) = 10\log_{10}(0.01) = -20dB$$

Example

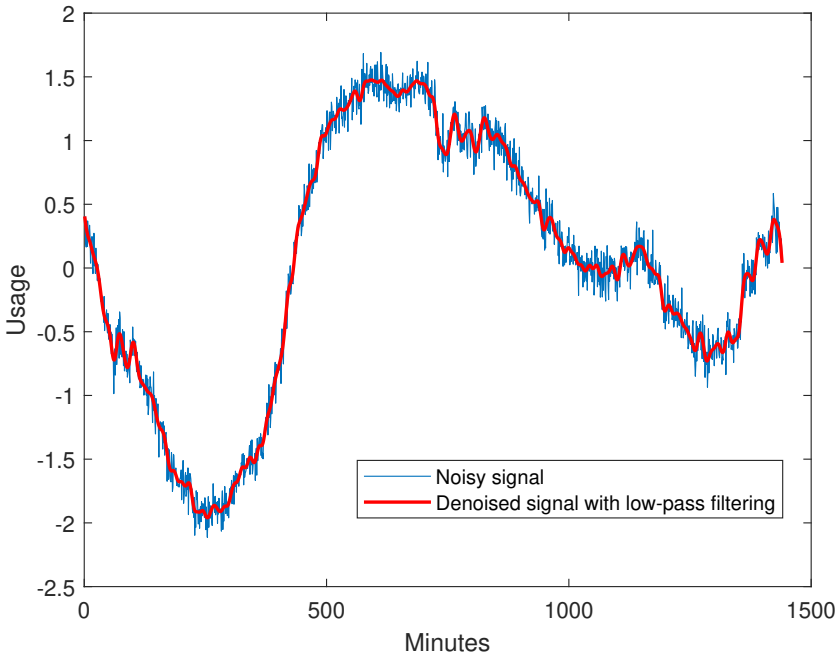


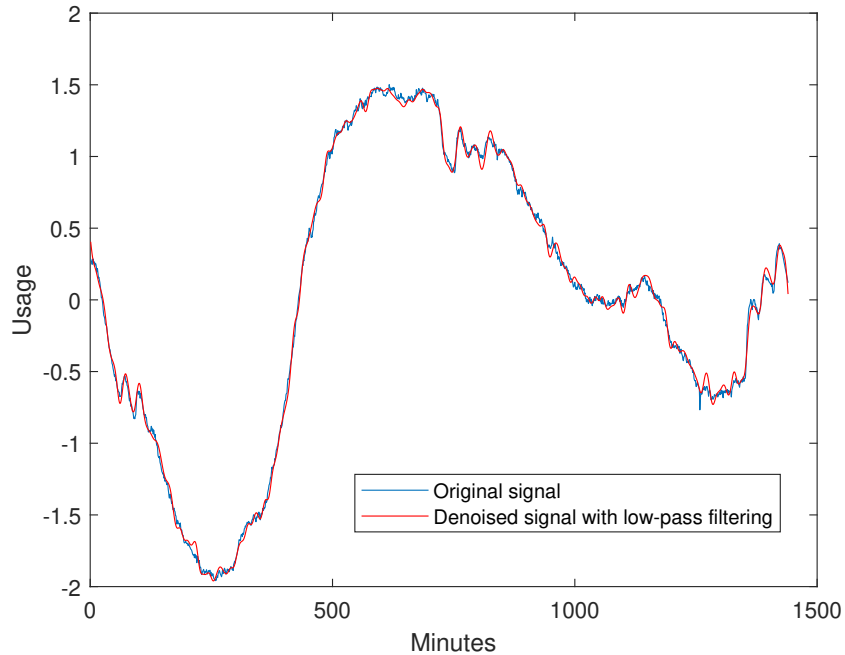
By plotting the log-spectrum of the noisy signal and knowing the noise variance σ^2, one can guess that all frequencies greater that e.g. 0.001 Hz are likely to only contain noise.

Filter design

- By observing the log-spectrum of the noisy signal and using either prior knowledge on the original signal bandwidth or on the noise level, we can determine the type of filter and associated cut-off frequencies that can be used for denoising
- From that, it is only digital filter design (see previous slides on moving average and Butterworth filters).

Example





Low-pass filtering (Butterworth filter of order 4) with $f_c = 0.001$ Hz

Other approaches

Other approaches for denoising include

- **Sparse dictionary techniques**, where the signal is approximated with parametric or learned function, thus removing the noise component
- **Decomposition techniques**, as noise may be considered independent of the signal component (EMD, SSA, ICA...)

Contents

Trend+Seasonality model

The trend+seasonality model writes as

$$x[n] = \underbrace{\alpha_1\beta_1(nT_s) + \ldots + \alpha_j\beta_j(nT_s)}_{x^{trend}[n]} + \underbrace{\alpha_{j+1}\beta_{j+1}(nT_s) + \ldots + \alpha_d\beta_d(nT_s)}_{x^{seasonality}[n]} + b[n]$$

- Seasonality: pseudo-periodic component
- Trend: smooth variations, systematic increase or decrease in the data

Detrending

Given a signal $x[n]$, estimate and remove the trend component $x^{trend}[n]$

Standard models

The most common trend models are:

- Constant trend

$$x^{trend}[n] = \alpha_0$$

- Linear trend

$$x^{trend}[n] = \alpha_1 (nT_s) + \alpha_0$$

- Polynomial trend

$$x^{trend}[n] = \sum_{k=0}^{K} \alpha_k (nT_s)^k$$

Least-square regression

- Least-square estimator: minimization of

$$\|\mathbf{x} - \boldsymbol{\beta}\boldsymbol{\alpha}\|_2$$

where

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0(0) & \cdots & \beta_K(0) \\ \beta_0(T_s) & \cdots & \beta_K(T_s) \\ \vdots & \ddots & \vdots \\ \beta_0((N-1)T_s) & \cdots & \beta_K((N-1)T_s) \end{pmatrix}$$

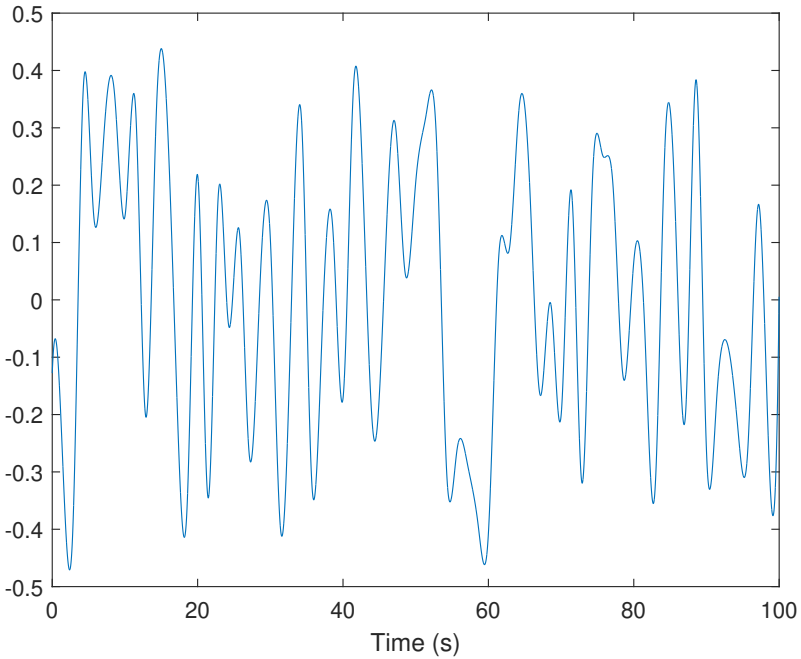
- Closed form solution

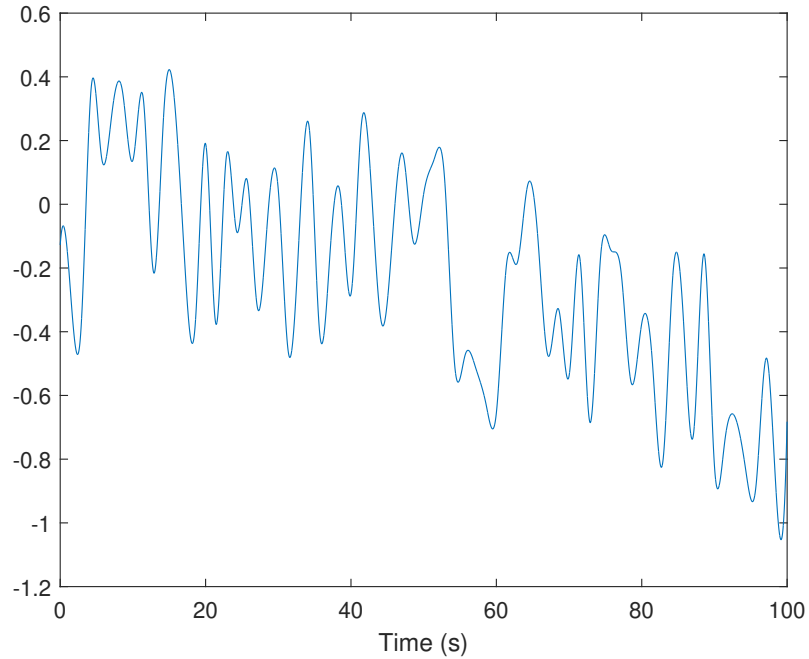
$$\hat{\boldsymbol{\alpha}} = \left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right)^{-1}\boldsymbol{\beta}^T\mathbf{x}$$

- Estimation of the trend

$$\mathbf{x}^{\mathbf{trend}} = \boldsymbol{\beta}\hat{\boldsymbol{\alpha}}$$

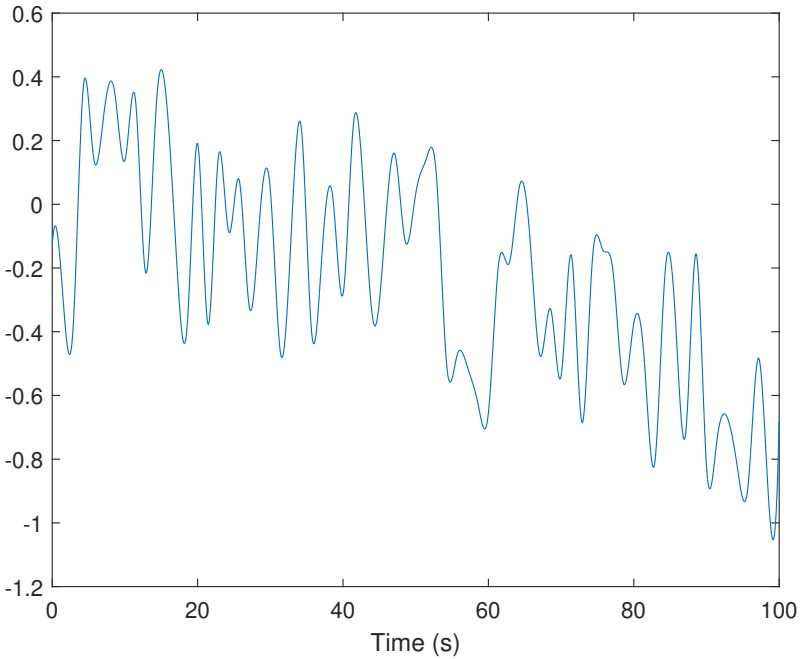
Example

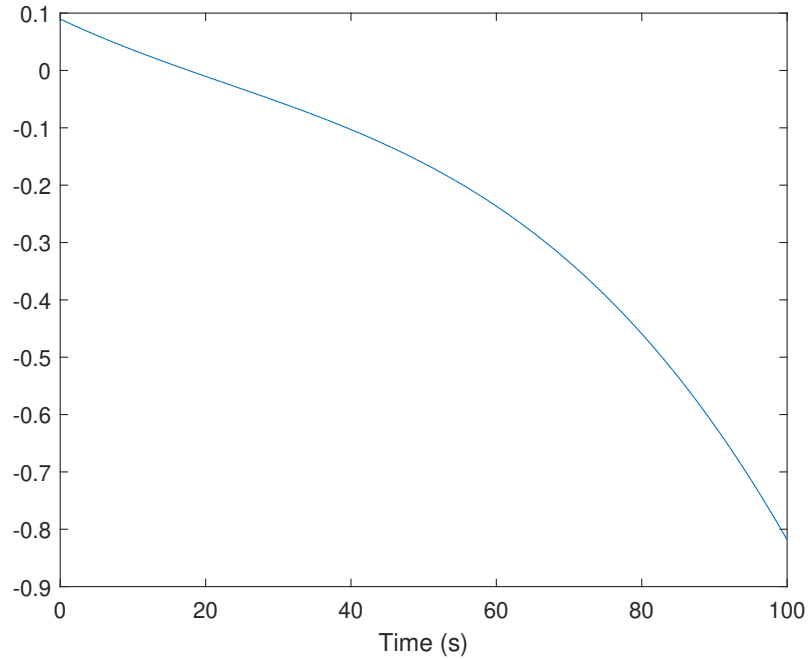




Signal with/without trend

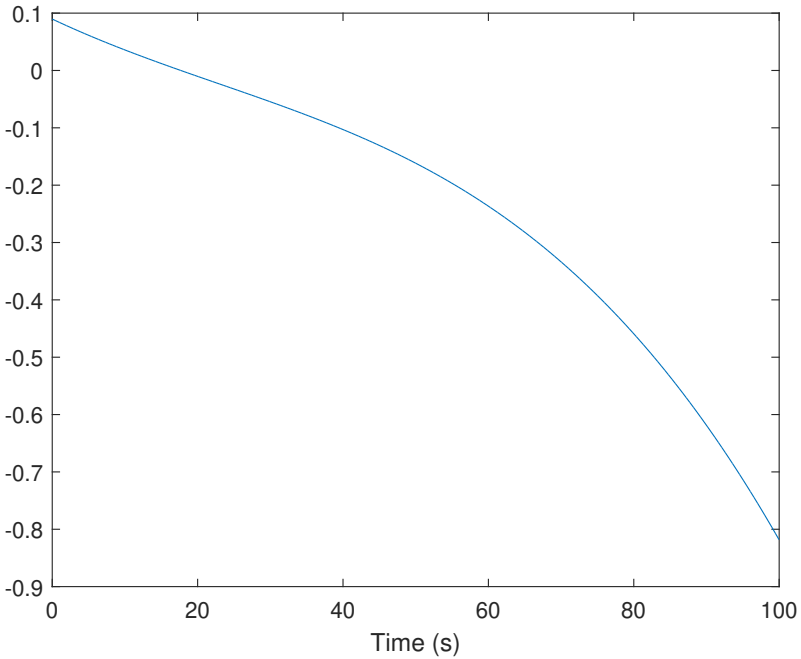
Example

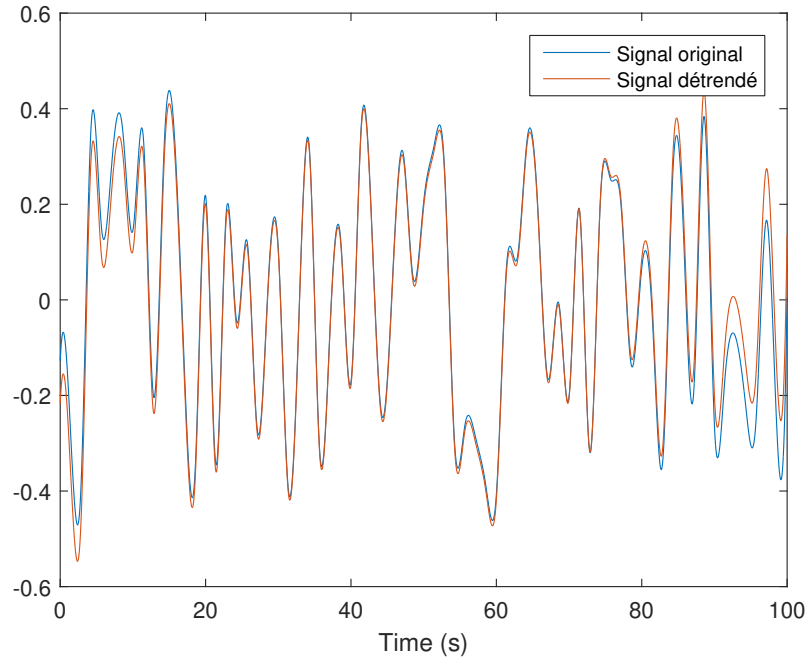




Regression on polynomials of order 3

Example





Regression on polynomials of order 3

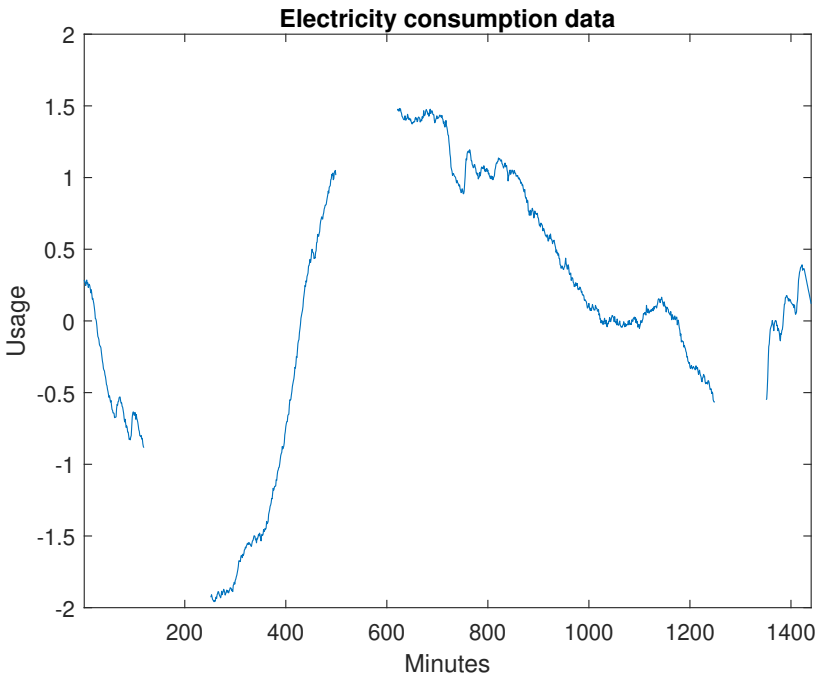
Other approaches

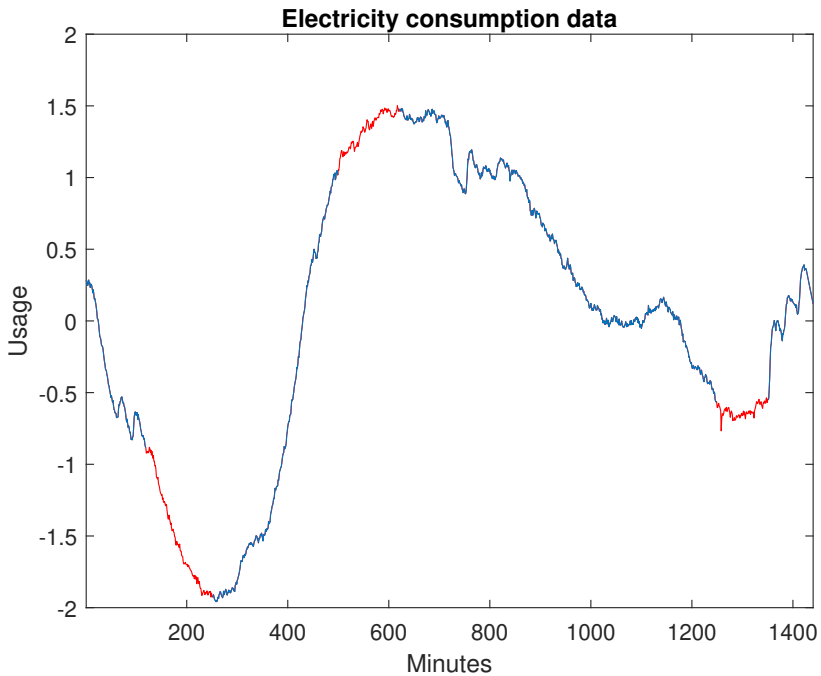
Other approaches for detrending include

- **Filtering techniques**, as trends often correspond to low frequencies or smooth components (low-pass/bandpass filters, Fourier or wavelets thresholding...)
- **Decomposition techniques**, as trends may be considered independent of the seasonality and/or the noise component (EMD, SSA, ICA...)

Contents

Interpolation of missing samples





Interpolation of missing samples

Given a signal x and a set of missing samples $\mathcal{T}$, estimate the missing samples $\hat{\mathbf{x}}_{\mathcal{T}}$

Interpolation of missing samples

- Missing data are very frequent :
  - Sensor malfunctions
  - Clipping effect
  - Corrupted samples
- Missing data can take several forms
  - Isolated samples: easy to handle
  - Contiguous samples (up to 100): necessitates a full reconstruction
- Interpolation includes prediction and inpainting [Lepot et al., 2017]

Polynomial interpolation

Given a time series $\mathbf{x}$ that we want to interpolate on the integer set $\mathcal{T} = [\![n_{start}, n_{end}]\!]$, the easiest interpolation strategy consists in using polynomial models for the reconstruction

- **Constant value**

$$\forall n \in \mathcal{T}, \quad \hat{x}[n] = \frac{x[n_{start} - 1] + x[n_{end} + 1]}{2}$$

- **Linear interpolation**

$$\forall n \in \mathcal{T}, \quad \hat{x}[n] = \beta_1 n + \beta_0$$

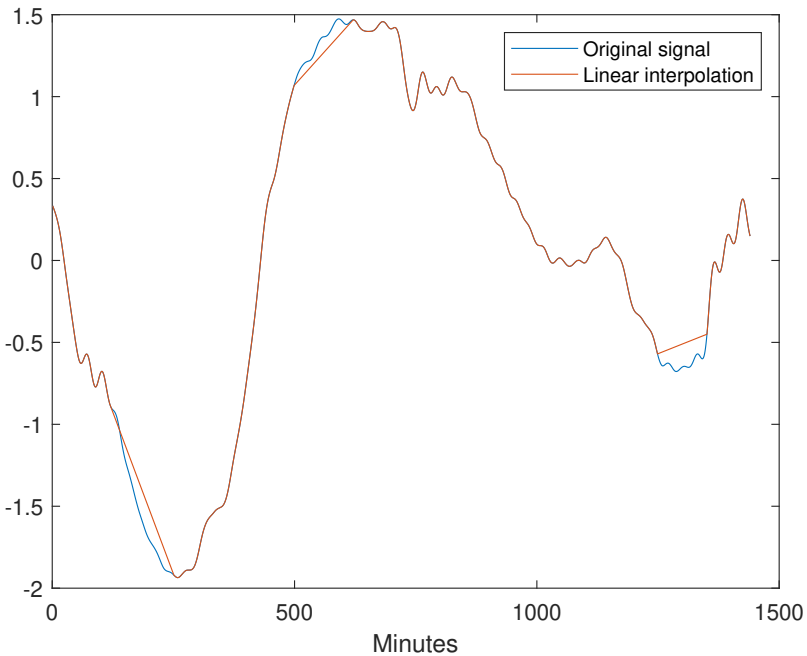
where β_0, β_1 are determined with the values $x[n_{start} - 1]$ and $x[n_{end} + 1]$

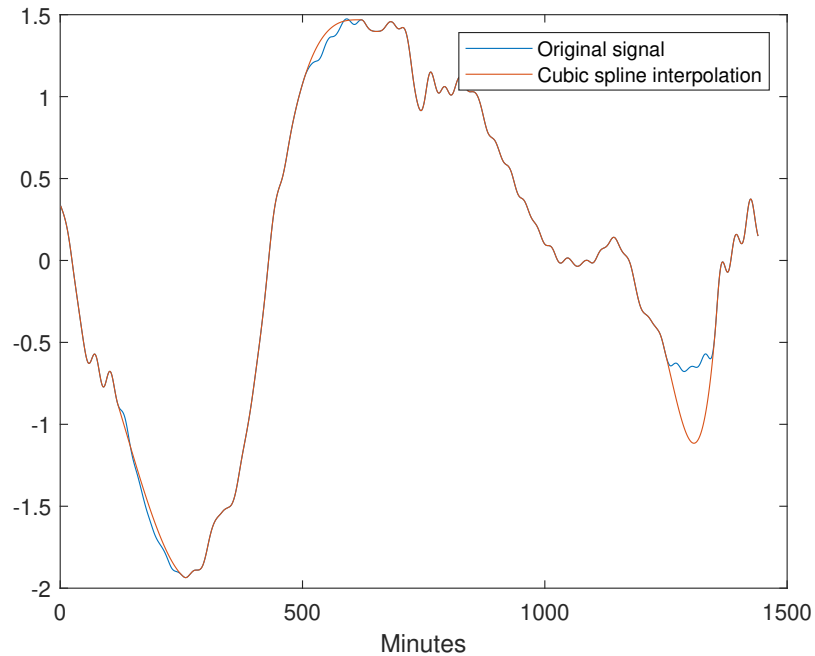
- **Cubic spline interpolation** [McKinley et al., 1998]

$$\forall n \in \mathcal{T}, \quad \hat{x}[n] = \beta_3 n^3 + \beta_2 n^2 + \beta_1 n + \beta_0$$

where β_k are determined by solving a system of equations based on $x[n_{start} - 2]$, $x[n_{start} - 1]$, $x[n_{end} + 1]$ and $x[n_{end} + 2]$

Example


Original signal
Linear interpolation
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
Minutes


Original signal
Cubic spline interpolation
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
Minutes

Pros and cons

- Easy to implement and good results for small segments
- In particular, when only a few missing samples: constant values is often the best
- When the degree of the polynomial increases, instabilities may occur (strong dependency with the neighborhood samples)
- When used extensively, may lead to a smoothing of the signal hence a change in the spectrum (boosting of the low frequencies)

Model-based interpolation

- For long segments of missing samples, interpolation becomes a full reconstruction task
- In this case a model is necessary to obtain a satisfactory interpolation
  1. Choice of an adequate model
  2. Parameter inference from the known samples
  3. Replacement of the missing samples by values in adequacy with the learned model

Model-based interpolation

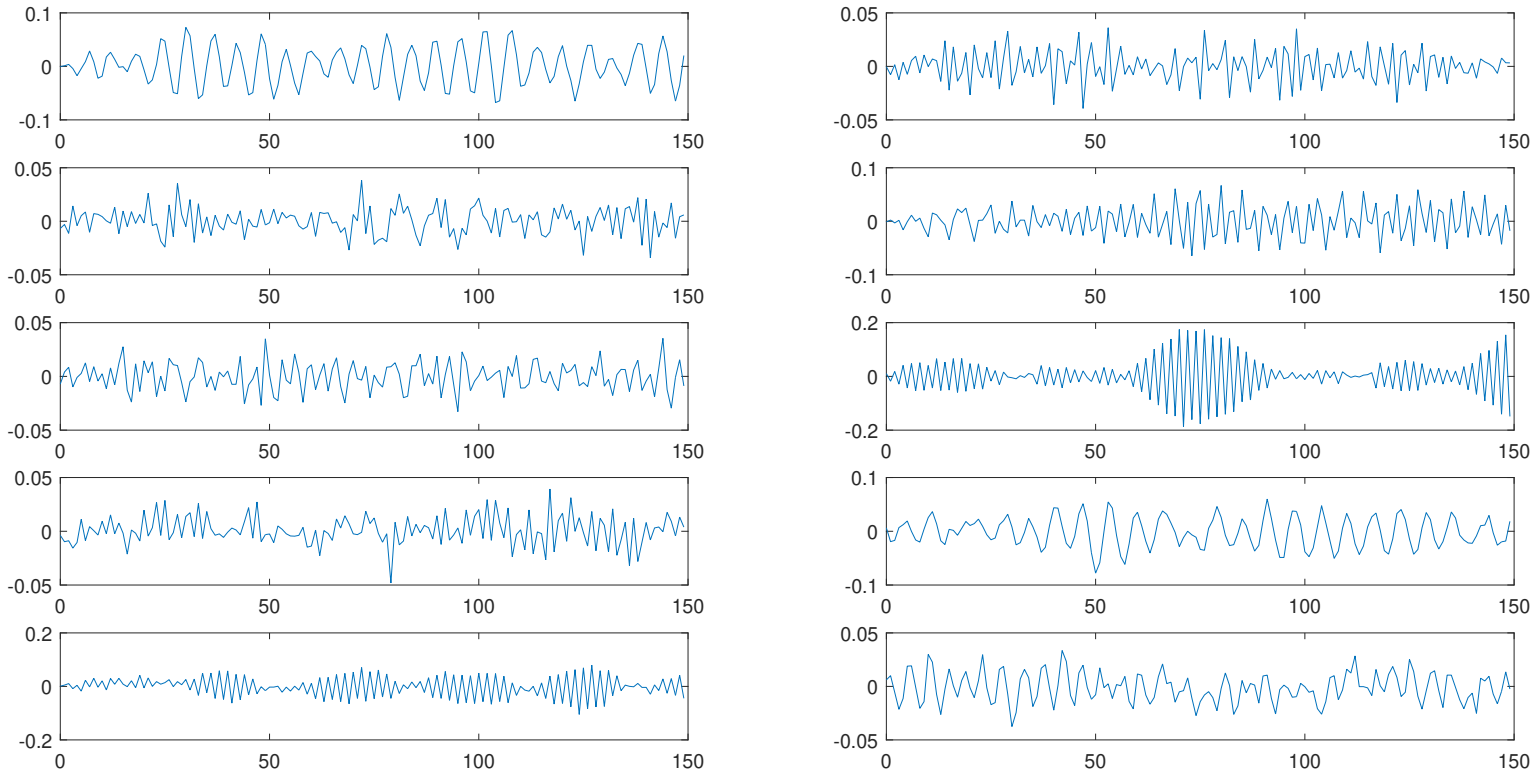
- Problem: how do we estimate the parameters from a time series with missing data?
- Iterative solution
  1. Initialization of the missing samples with simple rough estimates (set to zero, constant or linear interpolation...)
  2. Parameter inference from all samples
  3. Reconstruction of the missing samples from the learned model
  4. Repeat steps 2 and 3 until convergence

Autoregressive model $AR(p)$

$$x[n] = -\sum_{i=1}^{p} a_i x[n-i] + b[n]$$

- p : order of the model
- $a_1, \ldots, a_p$: AR coefficients
- $b[n]$: white noise (often called innovation)

Examples



Several $AR(5)$ processes

Parameter estimation

- In order to estimate the parameters, we can go back to the main equation

$$x[n] + a_1 x[n-1] + \ldots + a_p x[n-p] = b[n]$$

- By multiplying by $x[n-1]$ we obtain

$$x[n]x[n-1] + a_1 x[n-1]^2 + \ldots + a_p x[n-p]x[n-1] = b[n]x[n-1]$$

- By taking the expected value

$$\mathbb{E}\left[x[n]x[n-1]\right] + a_1 \mathbb{E}\left[x[n-1]^2\right] + \ldots + a_p \mathbb{E}\left[x[n-p]x[n-1]\right] = \mathbb{E}\left[b[n]x[n-1]\right]$$

- Since $b[n]$ and $x[n-1]$ are uncorrelated and $b[n]$ is a white noise

$$\mathbb{E}\left[b[n]x[n-1]\right] = \mathbb{E}\left[b[n]\right]\mathbb{E}\left[x[n-1]\right] = 0$$

- If $x[n]$ is wide-sense stationary, we have

$$\mathbb{E}\left[x[n_1]x[n_2]\right] = \gamma_x[|n_1 - n_2|]$$

Parameter estimation

- The equation becomes

$$\gamma_x[1] + a_1\gamma_x[0] + \ldots + a_p\gamma_x[p-1] = 0$$

- The same principle can be applied by multiplying the main equation by $x[n-2], x[n-3], \ldots$, leading to the following system of equations called **Yule-Walker** equations:

$$\begin{bmatrix} \gamma_x[0] & \gamma_x[1] & \cdots & \gamma_x[p-1] \\ \gamma_x[1] & \gamma_x[0] & \cdots & \gamma_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_x[p-1] & \gamma_x[p-2] & \cdots & \gamma_x[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} \gamma_x[1] \\ \gamma_x[2] \\ \vdots \\ \gamma_x[p] \end{bmatrix}$$

- Knowing the autocorrelation function $\gamma_x[m]$ for $m = 0 \ldots p$ is sufficient to estimate the parameters
- Most of the times, we use an empirical estimator of the autocorrelation function instead

AR-based interpolation

- For an $AR(p)$ model, given estimates of parameters $\hat{\mathbf{a}}$, the signal can be reconstructed by assuming that

$$x[n] \approx -\sum_{i=1}^{p} \hat{a}_i x[n-i]$$

- The prediction error on the whole time series writes

$$E(\mathbf{x}) = \sum_{n=p}^{N-1} \left| x[n] + \sum_{i=1}^{p} \hat{a}_i x[n-i] \right|^2$$

- The main idea is to minimize this quantity in order to retrieve appropriate values for the missing samples [Janssen et al., 1986]

AR-based interpolation

$$\mathbf{x}^* = \underset{\forall n \notin \mathcal{T}, \tilde{x}[n]=x[n]}{\operatorname{argmin}} E(\tilde{\mathbf{x}})$$

- This optimization problem has a closed form solution (least-square estimates) that is obtained by rewritting $E(\mathbf{x})$ as the sum of terms depending on the missing samples $n \in \mathcal{T}$ and other depending only on the known samples.
- By denoting $\mathbf{x}_{\mathcal{T}}$ the set of missing samples, the equation rewrites

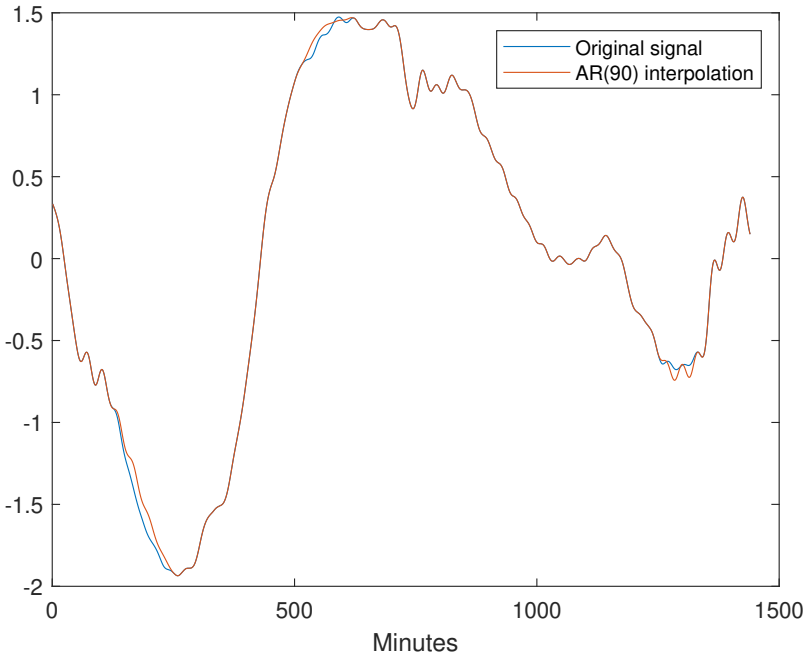
$$E(\mathbf{x}) = {\mathbf{x}_{\mathcal{T}}}^T \mathbf{B} \mathbf{x}_{\mathcal{T}} + 2\mathbf{x}_{\mathcal{T}}\mathbf{d} + C$$

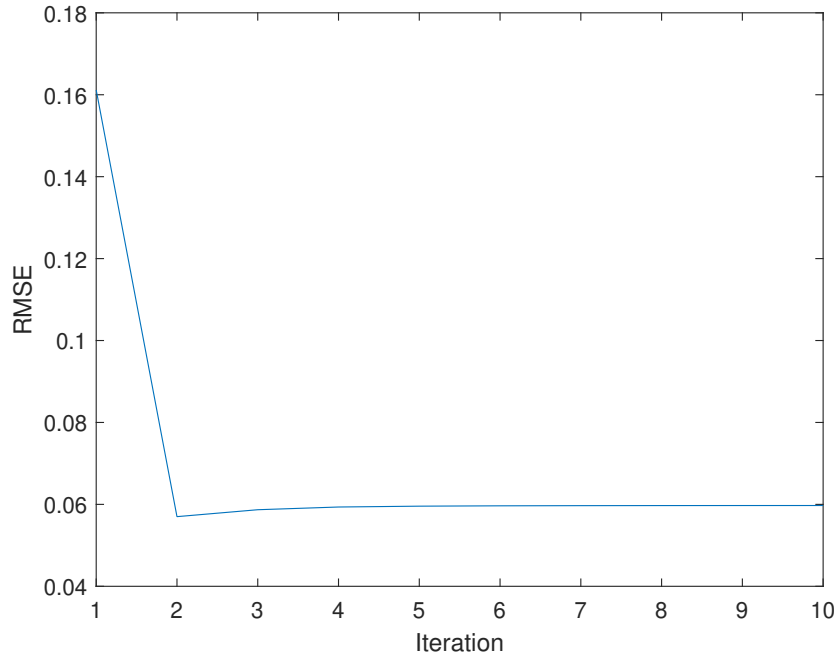
where

  - $\forall (t,t') \in \mathcal{T}, \quad b_{t,t'} = \begin{cases} \sum_{l=0}^{p-|t-t'|} \hat{a}_l \hat{a}_{l+|t-t'|} & \text{if } 0 \leq |t-t'| \leq p \\ 0 & \text{else} \end{cases}$
  - $\forall (t,t') \in \mathcal{T}, \quad d_t = \sum_{\substack{-p \leq k \leq p \\ t-k \notin \mathcal{T}}} b_{|k|} x[t-k]$
  - C is a constant only depending on the known samples
- The final problem is simply a linear system and thus easy to solve

$$\mathbf{B}\mathbf{x}_{\mathcal{T}} = -\mathbf{d}$$

Example

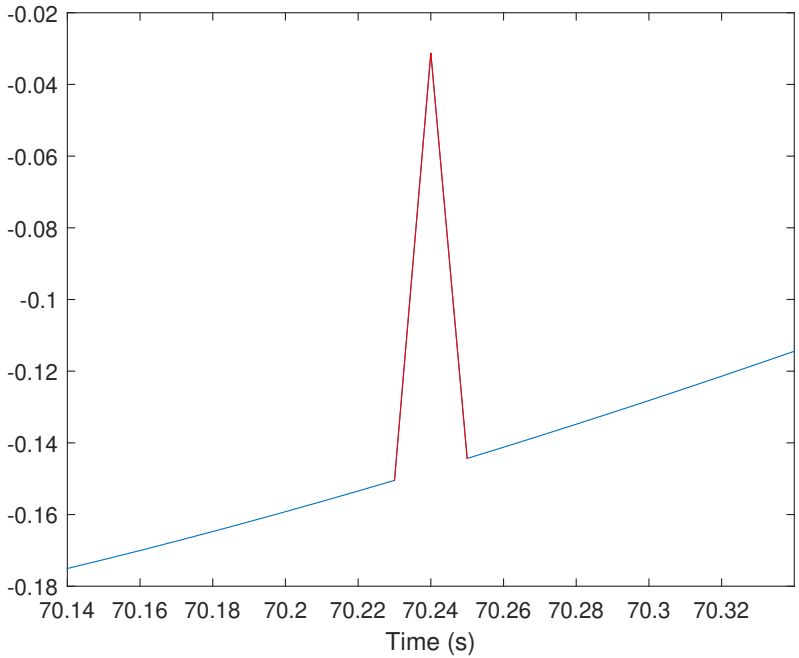


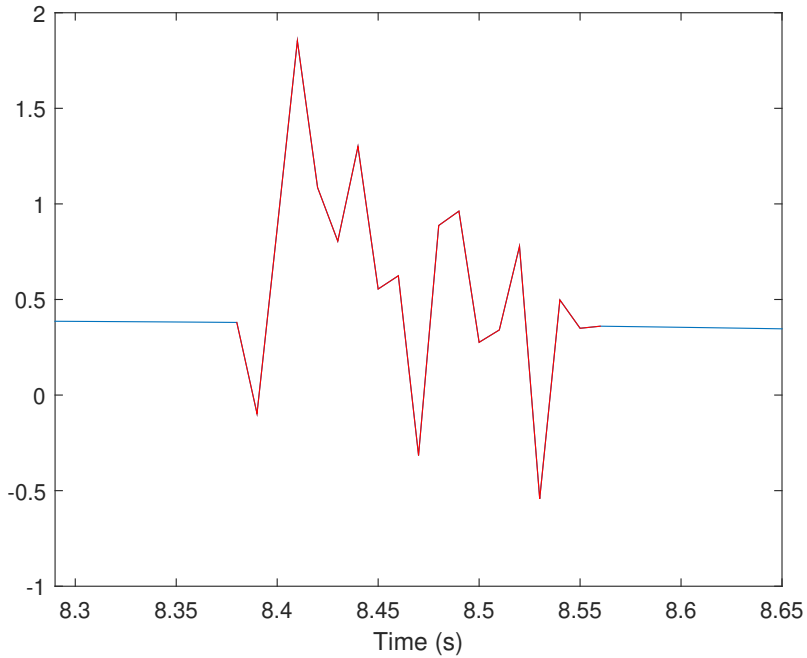


Interpolation with $AR(90)$ model

Contents

Outlier removal





Outliers, also called impulsive noise (as opposed to AWGN) correspond to spurious samples (isolated or continuous) that take unlikely values

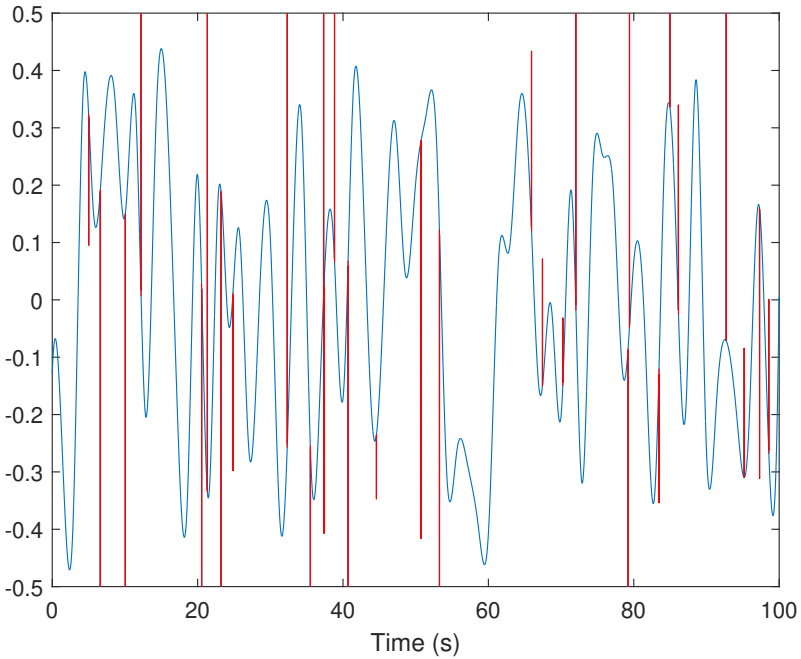
Outlier removal

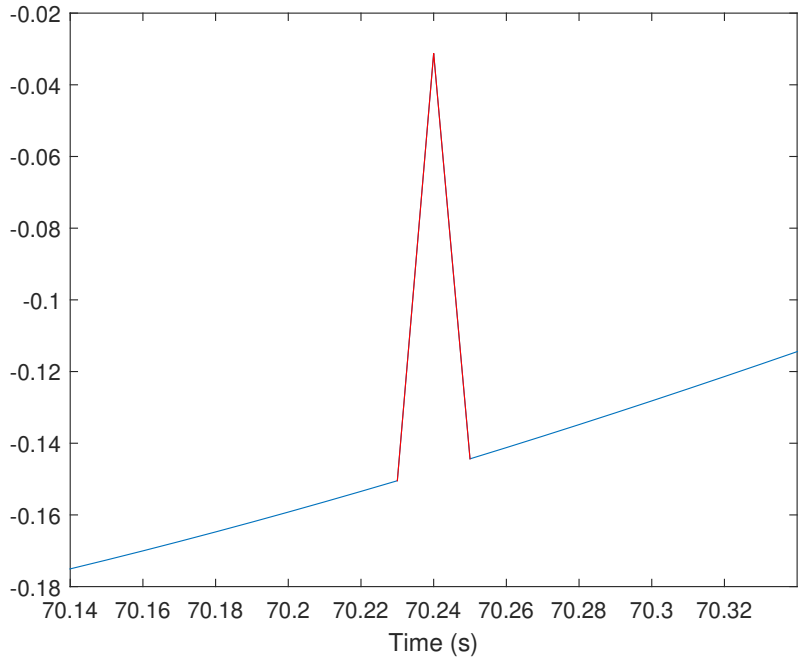
Outlier removal

Given a signal $x[n]$, outlier removal consists in detecting the locations $\mathcal{T}$ of the outliers (detection phase) and to replace these values with more adequate values (interpolation phase)

- Interpolation phase can be done by using the previously described algorithms. We will therefore focus on the detection phase.
- Two settings: isolated samples or contiguous group of samples
- Outliers are not only characterized by their values but also on their positions in the time series: context is fundamental

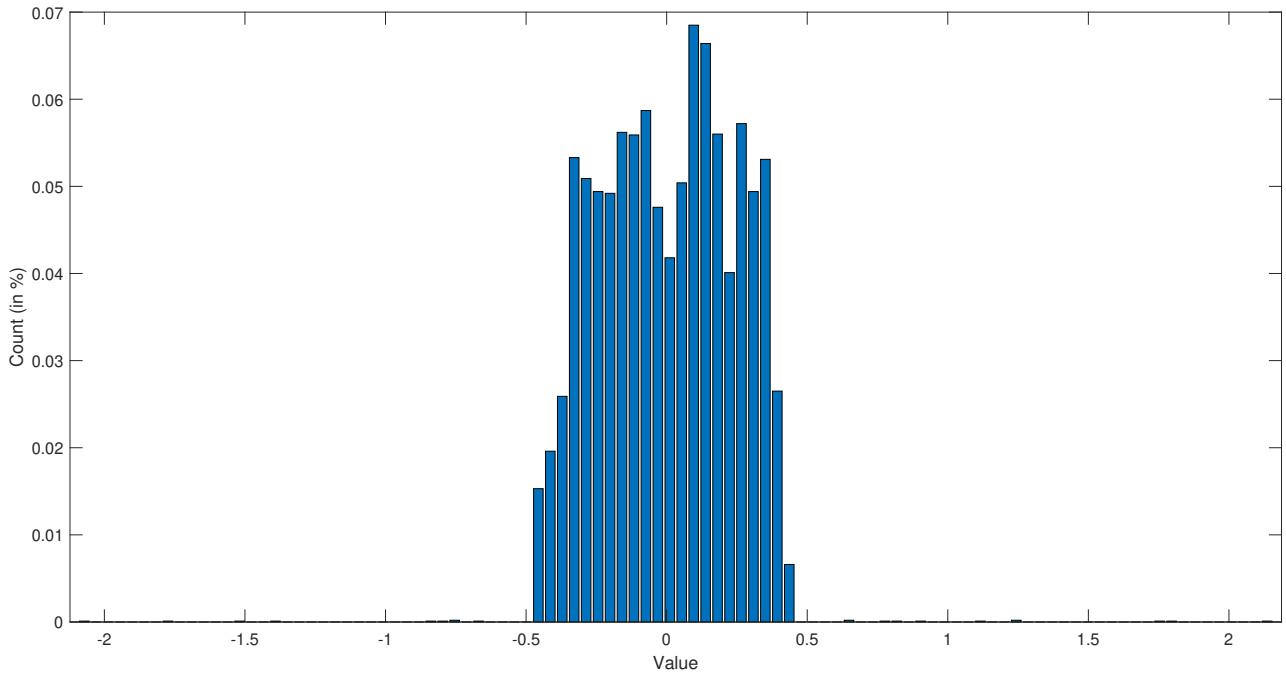
Isolated samples





Impulsive noise that only corrupts isolated samples

Histogram



If the values taken by the impulsive noise are particularly large with respect to the signal, they can be detected by looking at the histogram of the values taken by the samples: similar to outlier detection in statistical data

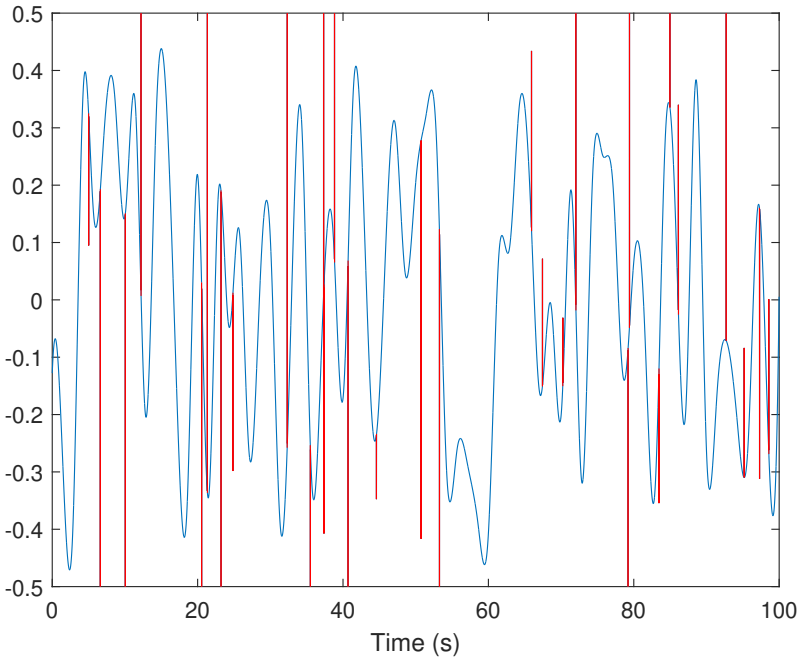
Median filtering

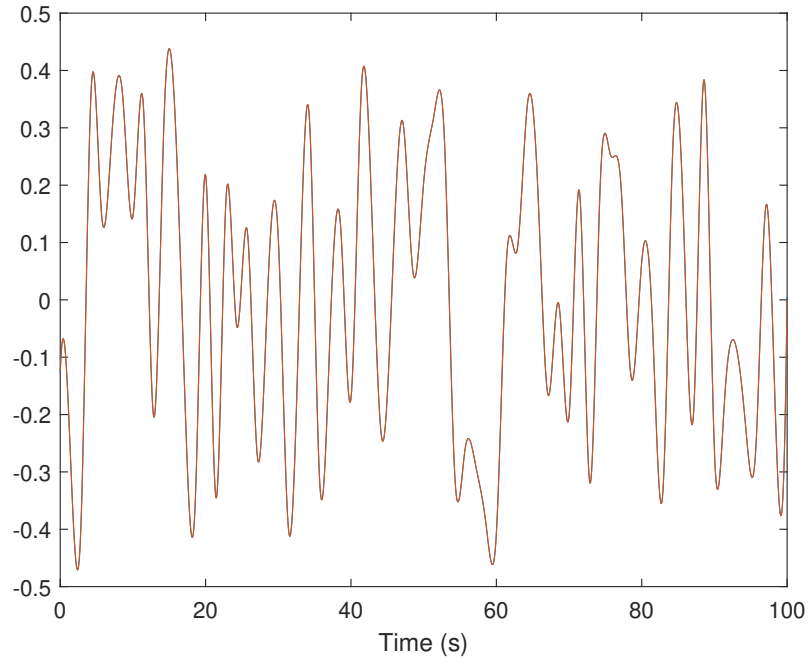
- Outliers can be detected AND removed by using a sliding median filtering that replaces each value by the median of the samples in a window of length $2w+1$:

$$\hat{x}[n] = \text{median}_{-w \leq i \leq +w} \{x[n-i]\}$$

- Median filtering allows to *smooth* the time series while preserving the discontinuities
- Example : original signal $[0.3 \;\; 0.4 \;\; 0.45]$ and noisy signal $[0.3 \;\; 0.9 \;\; 0.45]$
  - Moving average filter: $0.9 \rightarrow 0.55$
  - Median filter: $0.9 \rightarrow 0.375$

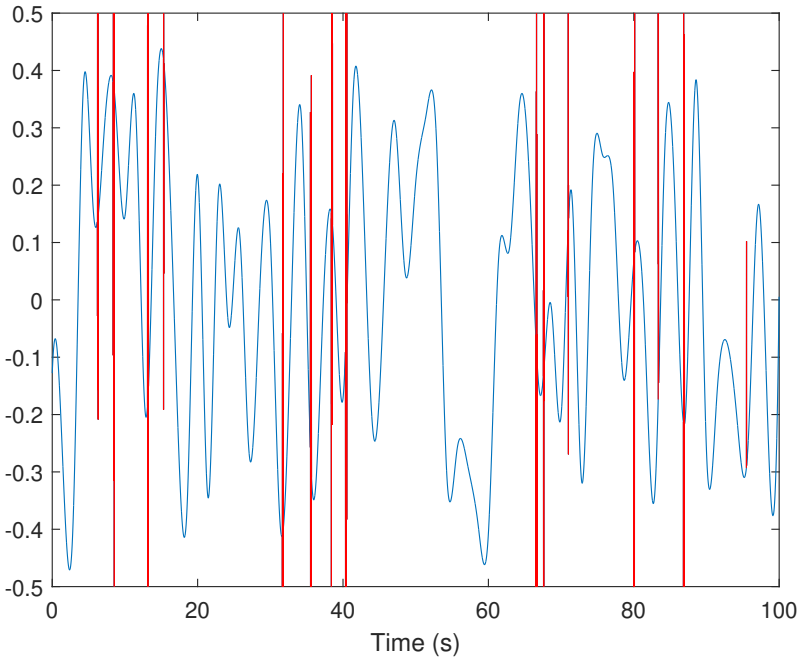
Median filtering

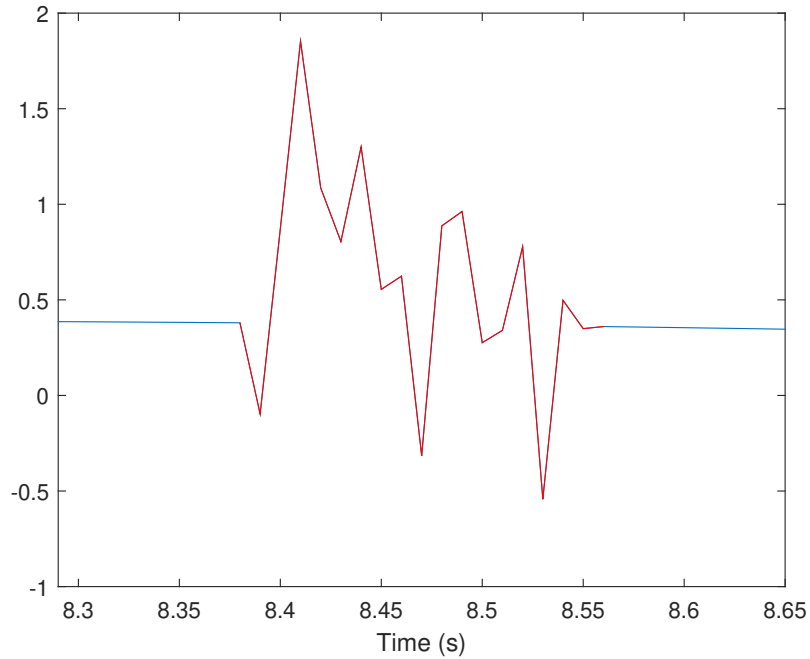




Perfect reconstruction with median filtering ($2w + 1 = 3$ samples)

Contiguous samples





Impulsive noise that corrupts groups of contiguous samples

Contiguous samples

- When the impulsive noise corrupts groups of contiguous samples, studying the values is not sufficient
- In order to retrieve the set of outliers $\mathcal{T}$, using a model may be necessary
- Outliers: samples that are *far* from their predicted values according to a model
- Same principle that model-based interpolation: parameter estimation, detection, interpolation and reiterate
- Note: this task is close to the Anomaly Detection task (see later)

AR-based outlier detection

$$x[n] = -\sum_{i=1}^{p} a_i x[n-i] + b[n]$$

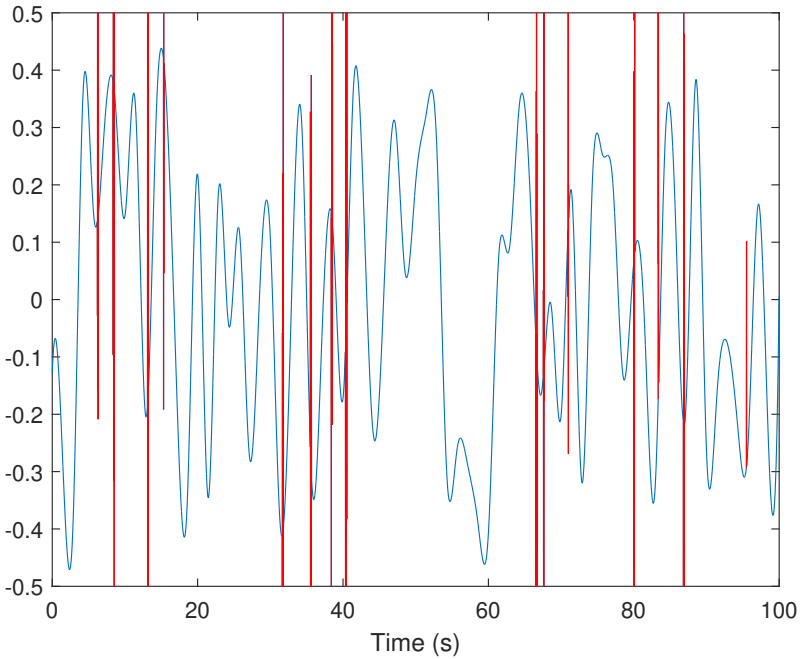
- Given estimates of the AR parameters $\hat{\mathbf{a}}$, the prediction error writes:

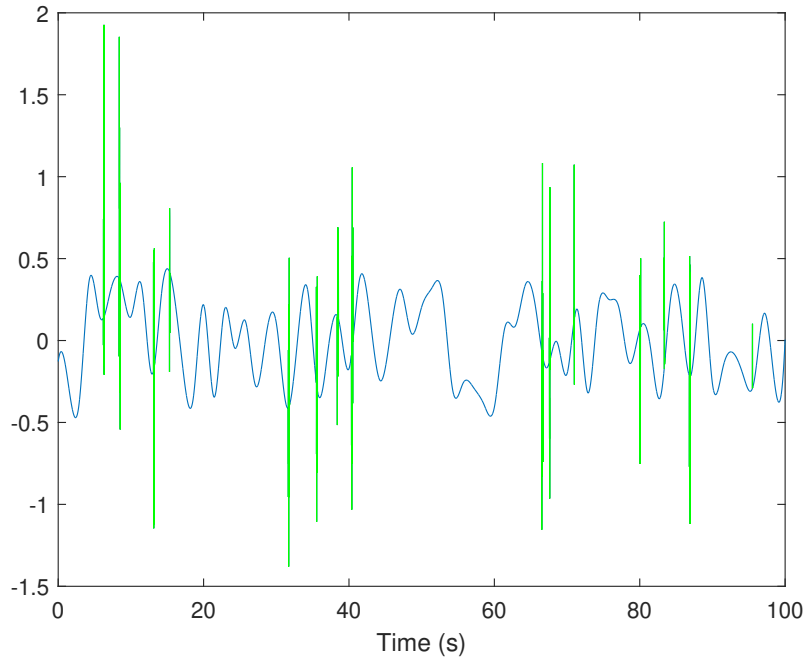
$$e[n] = x[n] + \sum_{i=1}^{p} \hat{a}_i x[n-i]$$

- If adapted model, good parameter estimation and low noise variance, this quantity must be rather small for samples that are not outliers [Oudre, 2015]
- Detection method with threshold λ :

$$\mathcal{T} = \{n \text{ s.t. } |e[n]| > \lambda\}$$

AR-based outlier detection





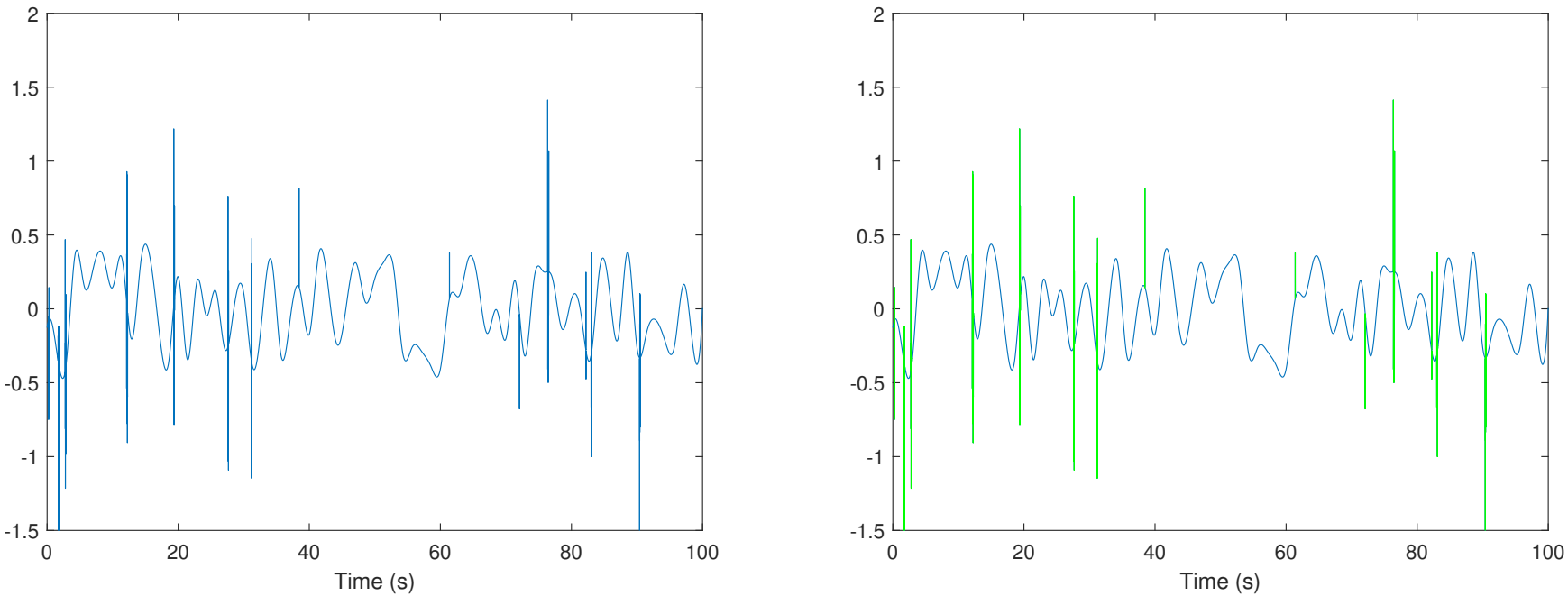
Detection with $AR(10)$ model

AR-based outlier detection and removal

In order to perform both detection and removal of impulsive noise, alternance between

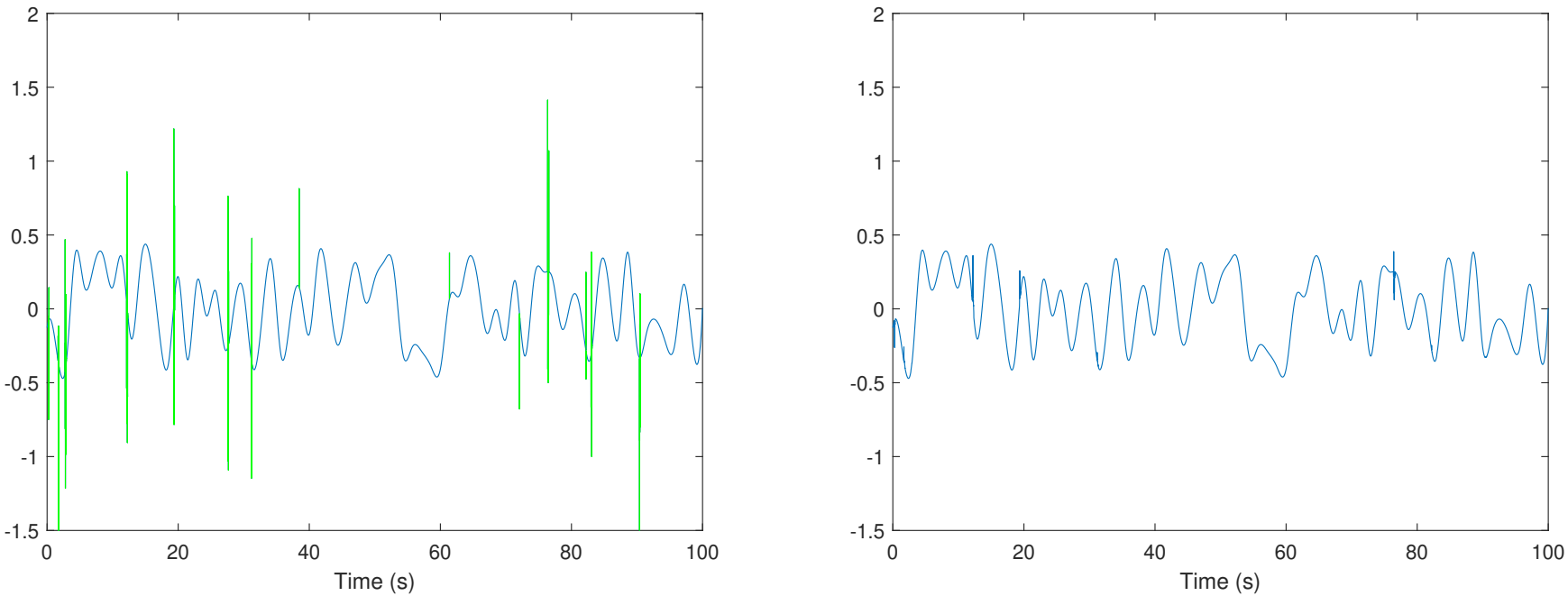
1. Estimation step: learn the AR parameters from the current time series
2. Detection step: detect the set of outliers
3. Interpolation step: replace these outliers by appropriate values
4. Reiterate steps 1, 2, 3

Example



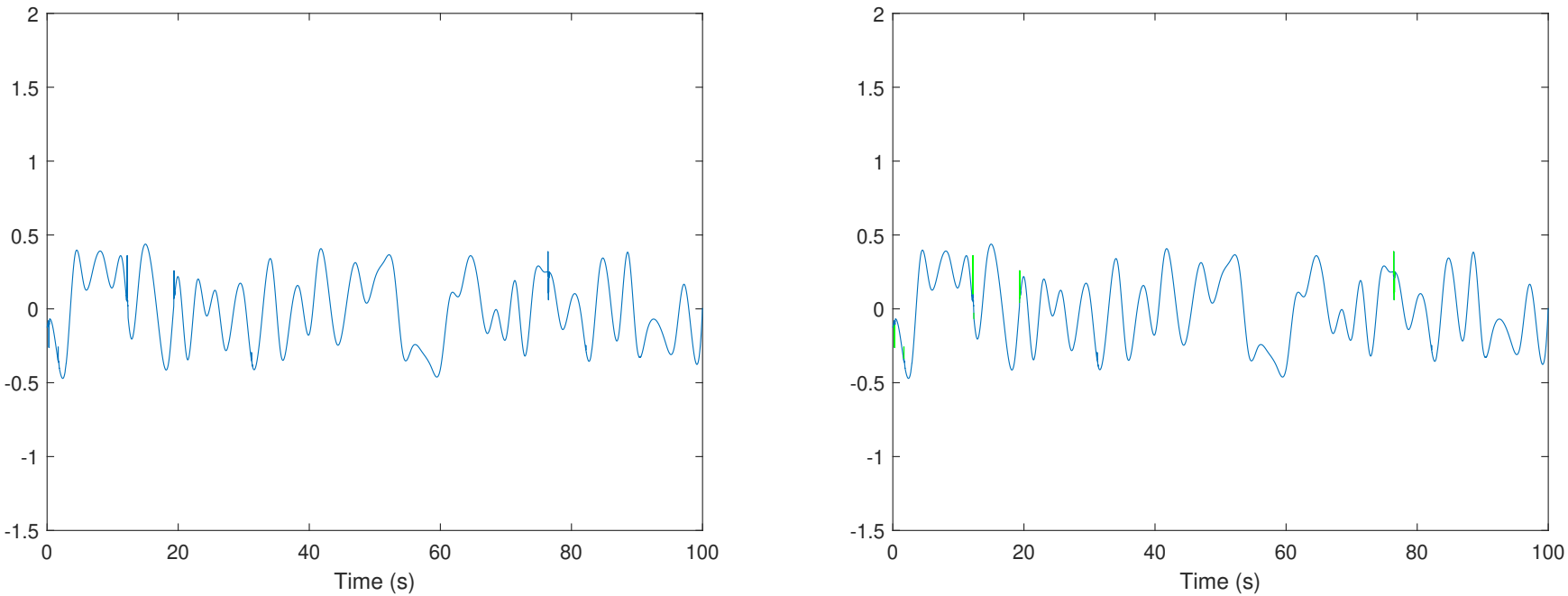
Iteration 1: Detection with $AR(10)$ model

Example



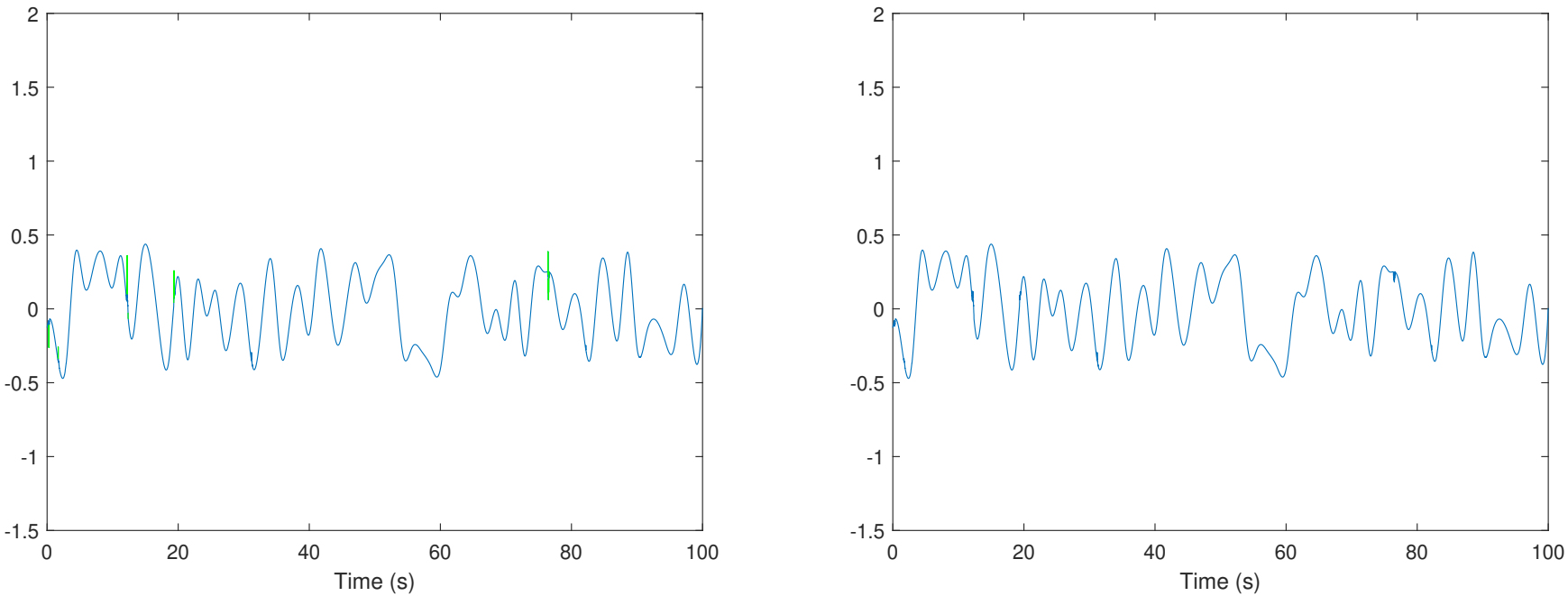
Iteration 1: Interpolation with $AR(10)$ model

Example



Iteration 2: Detection with $AR(10)$ model

Example



Iteration 2: Interpolation with $AR(10)$ model

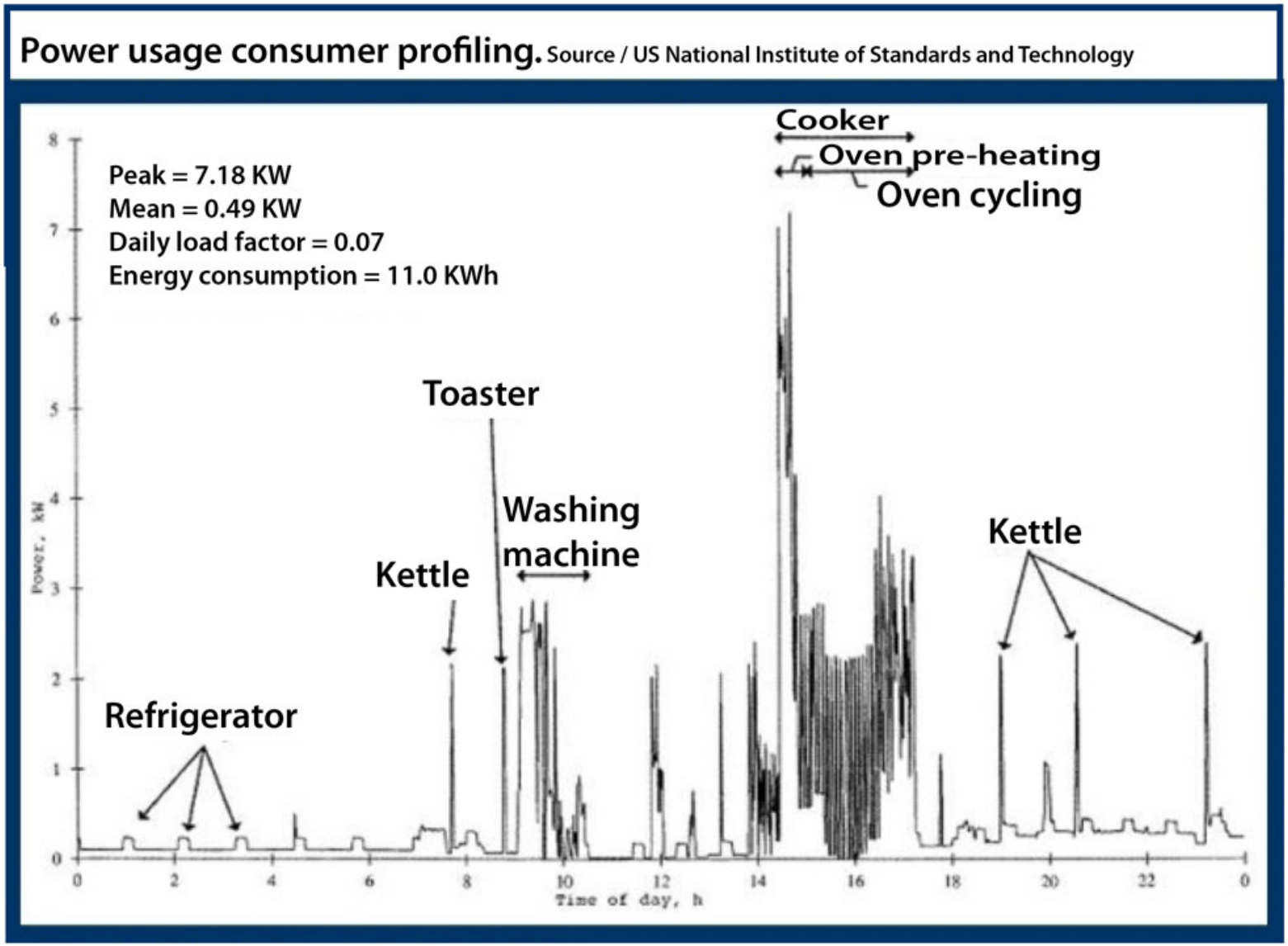
Contents

Discovering events in time series

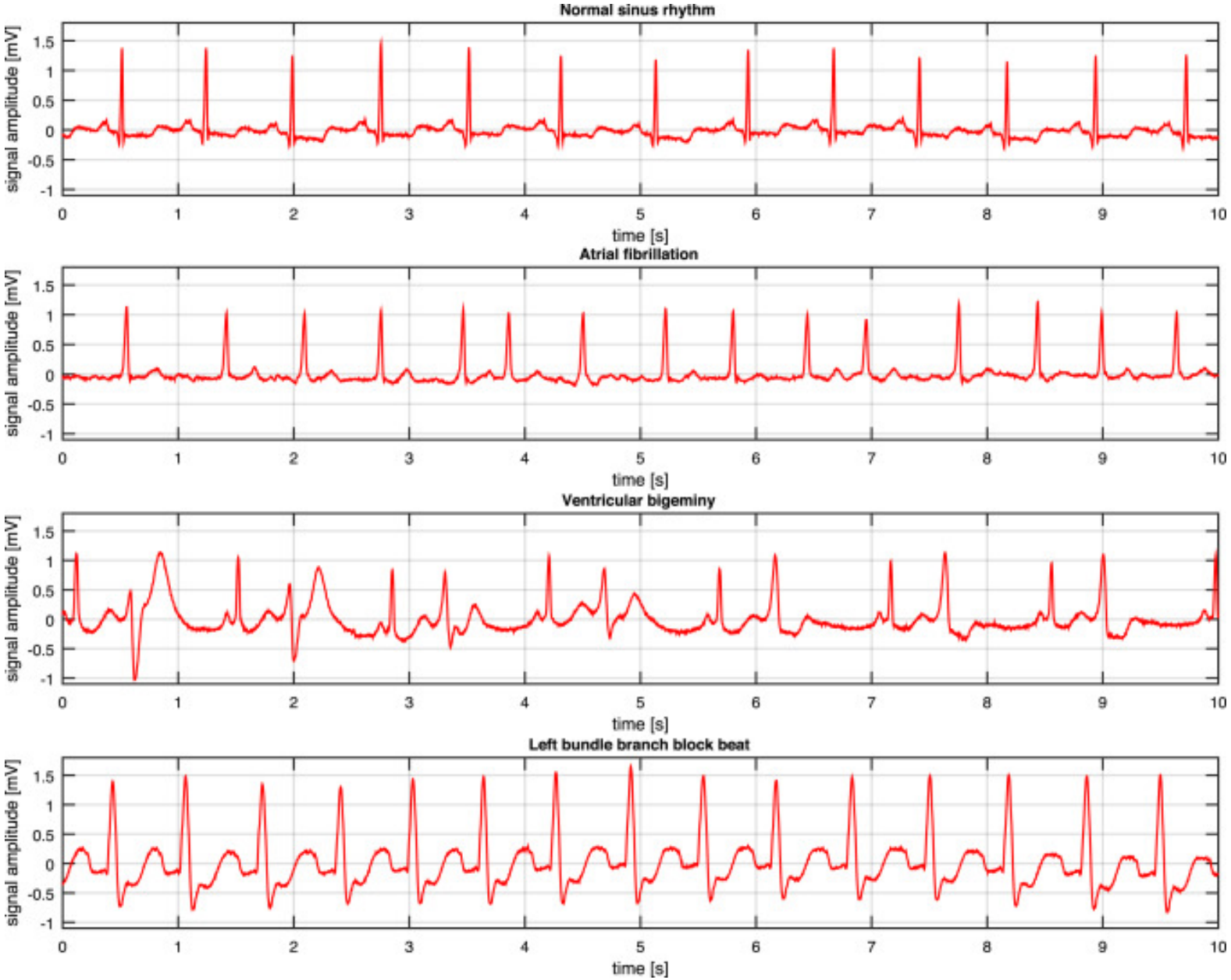
- Time series are widely used for monitoring
- When recorded for hours, days, weeks... data is likely to be redundant
- Three fundamental questions:
  - Are there some recurrent behaviors in my data ?
  - Were there significant changes in my data across time?
  - Was there something new or unusual in my data?
- Ill-posed problems: what is a *recurrent behavior?* what is a *significant change?* What is *new* or *unusual*?

Contents

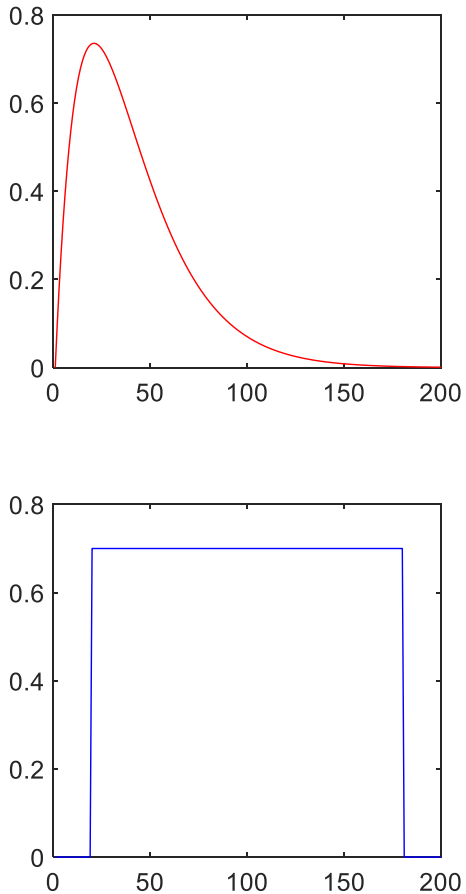
Motivation


Power usage consumer profiling. Source / US National Institute of Standards and Technology
Peak = 7.18 KW
Mean = 0.49 KW
Daily load factor = 0.07
Energy consumption = 11.0 KWh
Cooker
Oven pre-heating
Oven cycling
Toaster
Washing machine
Kettle
Kettle
Refrigerator
Power, kW
Time of day, h

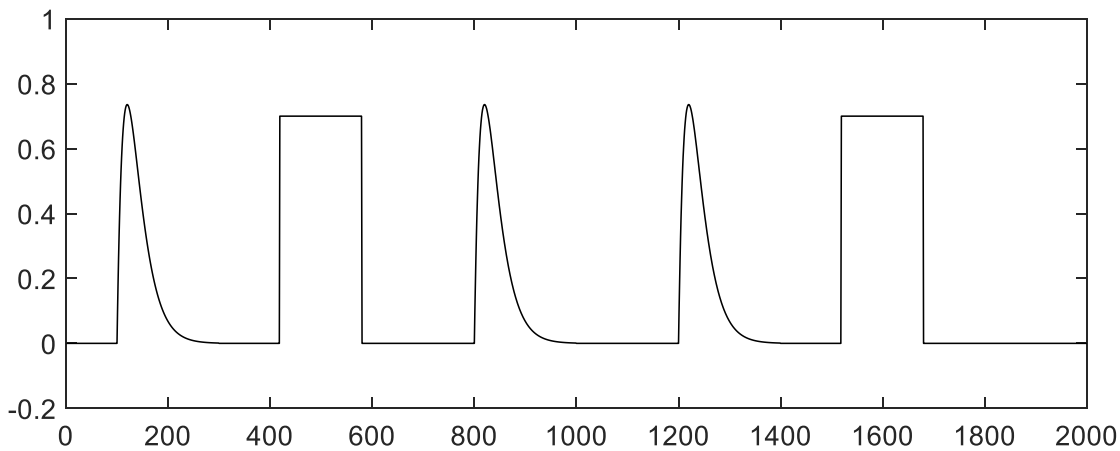
Motivation


Normal sinus rhythm
signal amplitude [mV]
time [s]
Atrial fibrillation
signal amplitude [mV]
time [s]
Ventricular bigeminy
signal amplitude [mV]
time [s]
Left bundle branch block beat
signal amplitude [mV]
time [s]

Problem 1: Pattern Detection

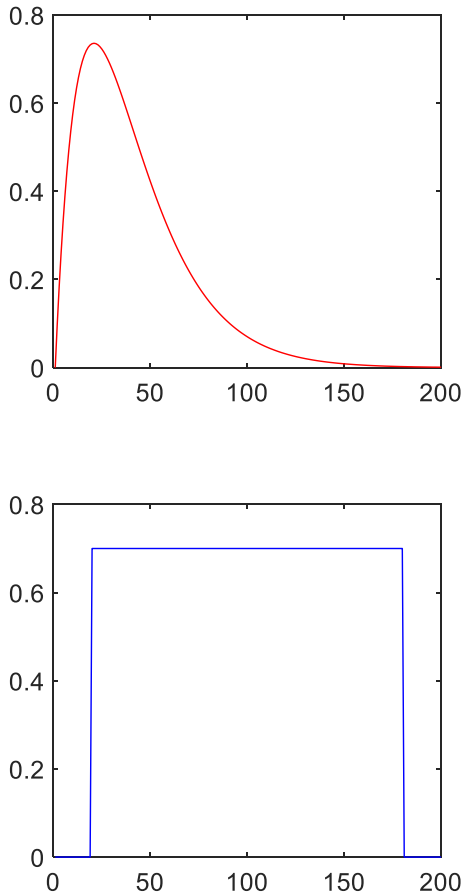


Dictionary of patterns

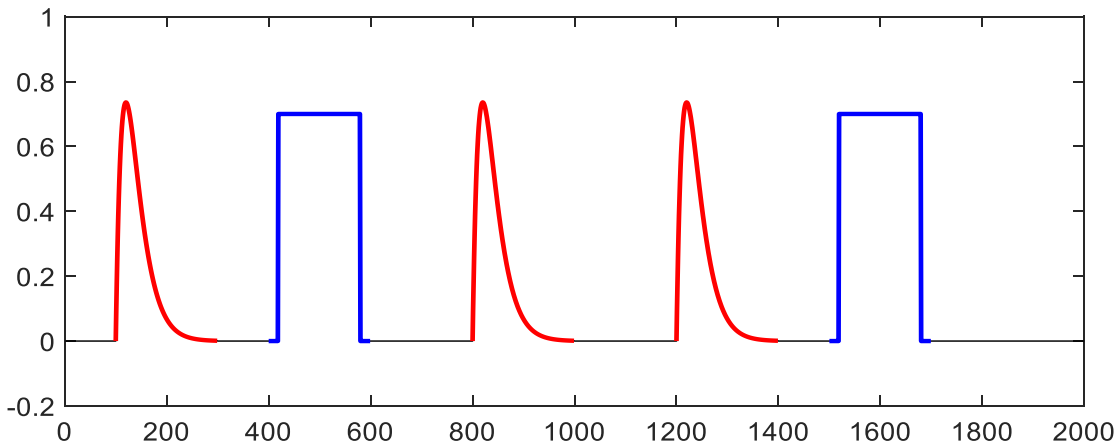


Input time series

Problem 1: Pattern Detection



Dictionary of patterns



Annotated time series

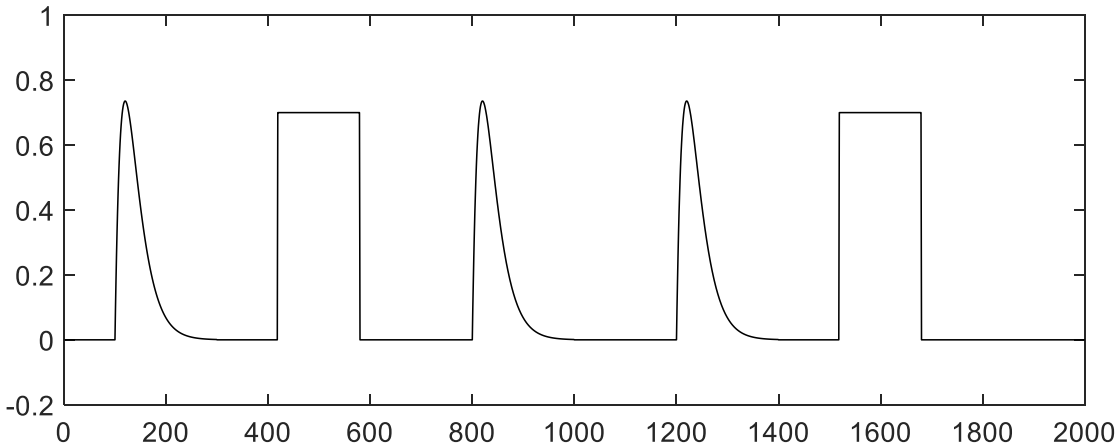
Problem 1: Pattern Detection

Pattern Detection

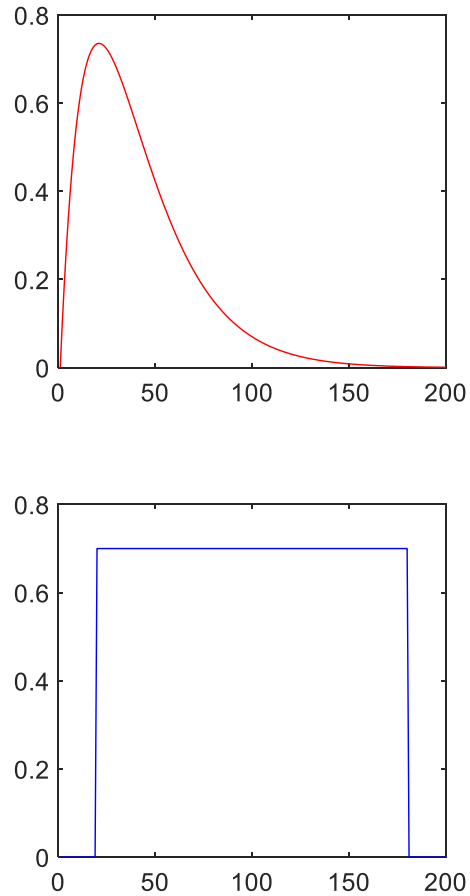
Given a dictionary of patterns, retrieve these patterns in an input time series

- The patterns/the time series can be multivariate
- The patterns may have different lengths
- The patterns can be annotated, i.e. be linked to a specific phenomenon of interest: in this context, pattern recognition will provide an automated annotation of the input time series

Problem 2: Pattern Extraction



Input time series



Extracted patterns

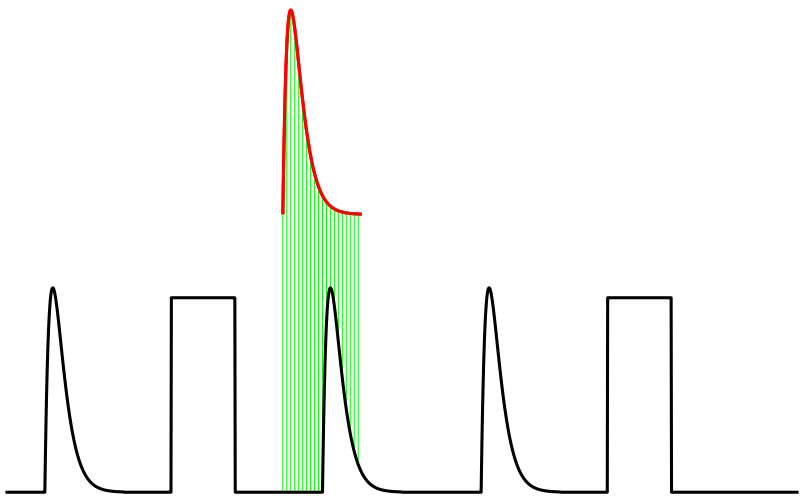
Problem 2: Pattern Extraction

Pattern Extraction

Given an input time series (or a set of time series), learn a dictionary of patterns

- A pattern is a *shape* that appears repetitively in the time series (but kinda blurry notion)
- All patterns are supposed to have the same length (for sake of simplicity)
- The extracted patterns can be used to characterize the time series, or studied individually

Pattern detection



- In order to find a known pattern $\mathbf{p}$ of length N_p in a time series $\mathbf{x}$ of length N, we need to compute all distances

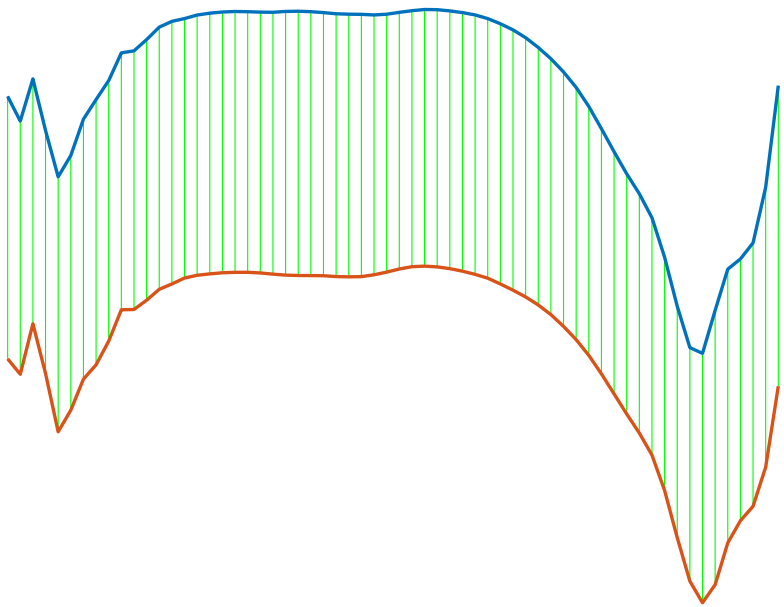
$$d[n] = d(\mathbf{p}, x[n : n + N_p - 1])$$

for $1 \leq n \leq N - N_d + 1$ and where $x[n : n + N_d - 1]$ is the sequence of $\mathbf{x}$ starting at n and of length N_p

- Intuitively, this is a costly operation since the distance needs to be computed approximately N times

Can we compute this quantity with good complexity?

Euclidean distance



$$d_{EUC}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{n=1}^{N} (x[n] - y[n])^2}$$

- Sensitive to time shifts, amplitude changes, offsets and dilatation/contraction
- Necessity to have a perfect match between the timelines
- Sensitive to outliers but approximately OK with low AWGN

Normalized Euclidean distance

In order to improve the robustness to changes in amplitude/offset, several authors recommand to use a normalized Euclidean distance :

$$d_{nEUC}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{n=1}^{N} (\tilde{x}[n] - \tilde{y}[n])^2}$$

where

$$\tilde{x}[n] = \frac{x[n] - \mu_x}{\sigma_x} \quad \text{z-score normalization}$$

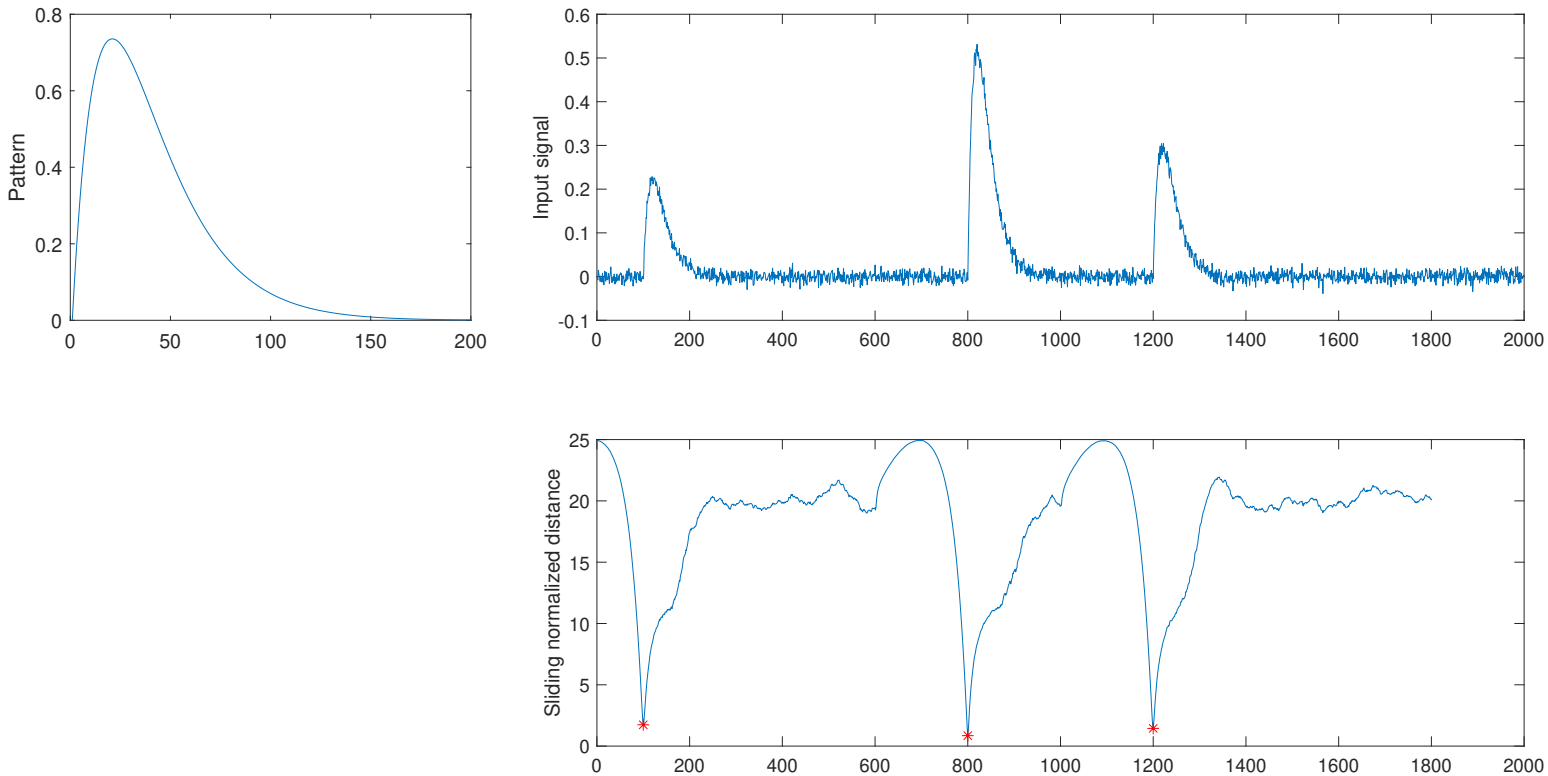
- Still sensitive to time shifts, dilatation/contraction, outliers and timelines
- Normalization may increase the sensitivity with respect to additive noise

Fast computation for Euclidean distance

$$d[n] = d(\mathbf{p}, x[n : n + N_p - 1])$$

- The sequence $d[n]$ is referred to in the literature as the **distance profile**
- A solution has been found for standard and normalized Euclidean distance. The whole computation is in $\mathcal{O}(N \log N)$ and does not depend on the pattern length N_p [Mueen et al., 2015]
- This solution is based on the properties of FFT (Fast Fourier Transform) that is a fundamental algorithm for computing the Discrete Fourier Transform of a time series in $\mathcal{O}(N \log N)$

Results with normalized Euclidean distance



Occurrences of the pattern can easily be retrieved in the signal by searching for peaks with small distance values

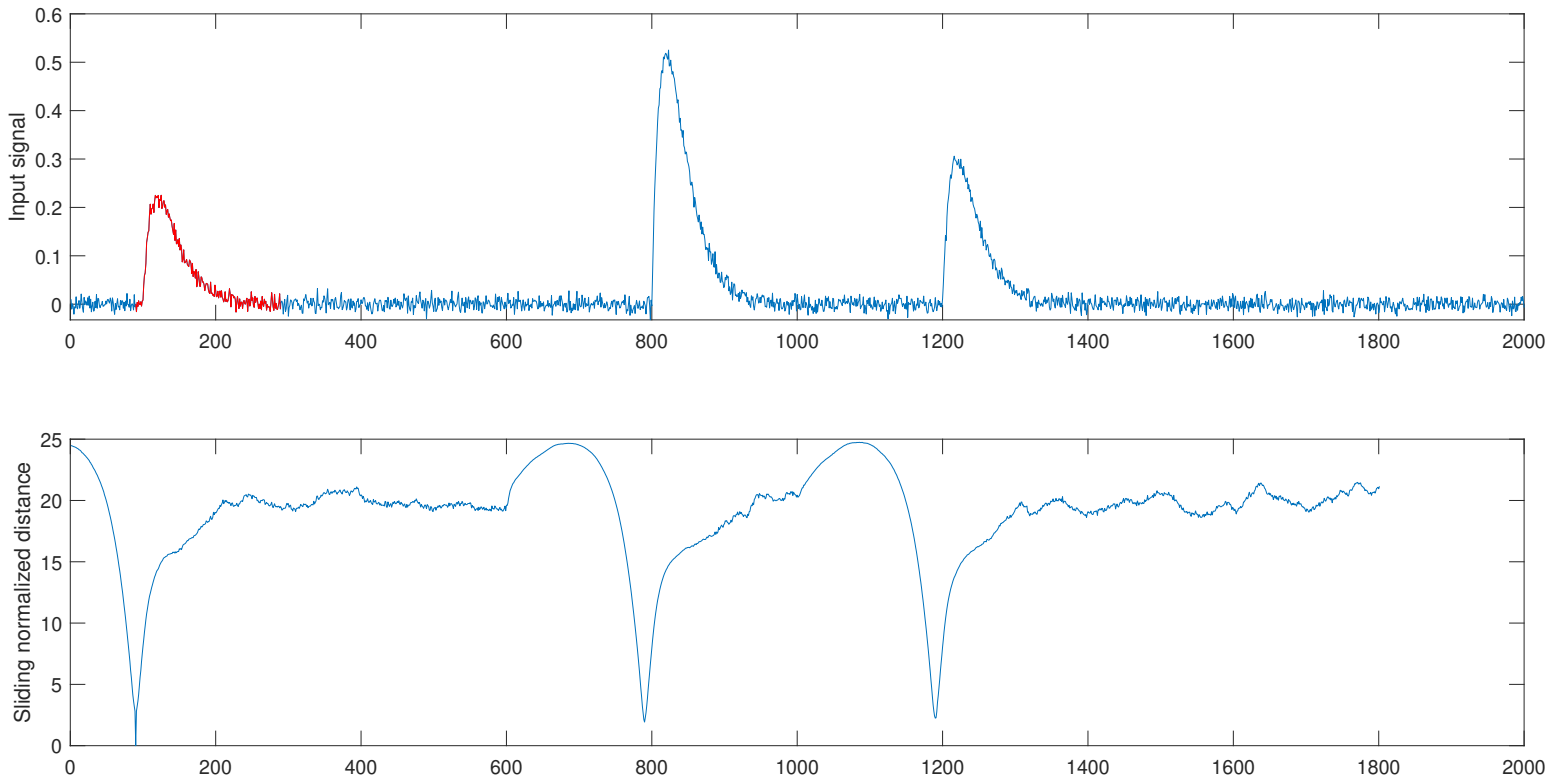
Pattern extraction

- We have seen methods to search for patterns in a time series: these methods need a predefined dictionary of templates
- In practice, one can be interested in the reverse question: how can I detect and extract patterns from a time series ?
- Unsupervised task: no prior knowledge except for the average duration of the researched patterns L
- **What is a pattern ?** Difficult question : repetitive shapes, notion of periodicity, etc.

Distance-based pattern extraction

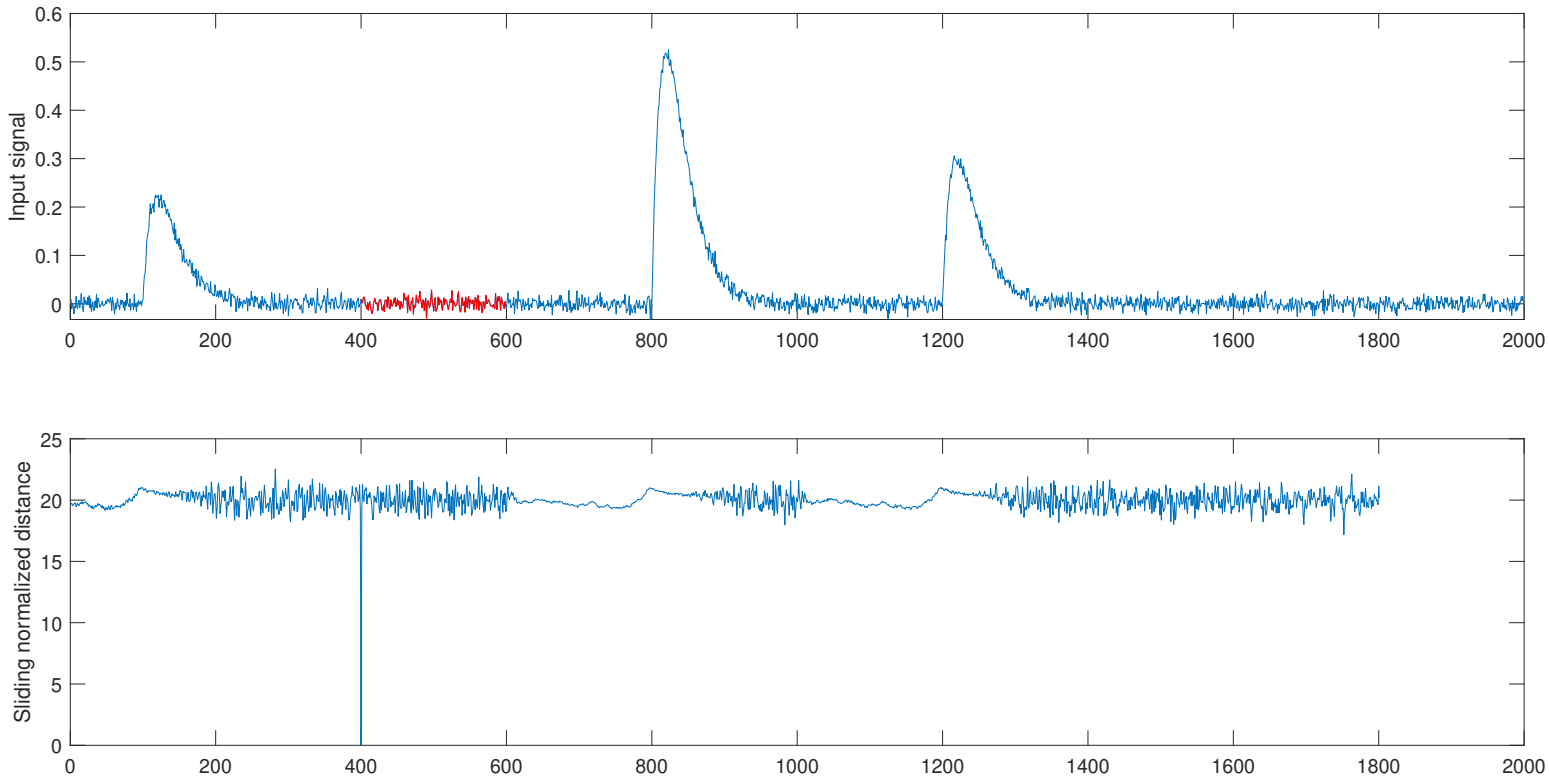
- Intuitively, we can re-use results from the previous part to automatically detect patterns
- A pattern is a subsequence that is likely to be found several times in the whole time series
- By computing a sliding distance between this subsequence and all subsequences in the time series, it should appear clearly that one (or several) others subsequences are very *close*
- **Solution:** use a brute-force algorithm to efficiently compute all distances?

Example



Here, we computed the distance profile of the sequence displayed in red: it is clear that this sequence appears 3 times in the time series. It might be a pattern!

Example



Here, we computed the distance profile of the sequence displayed in red: no obvious structure in the distance profile (except for the exact correspondence): unlikely to be a pattern!

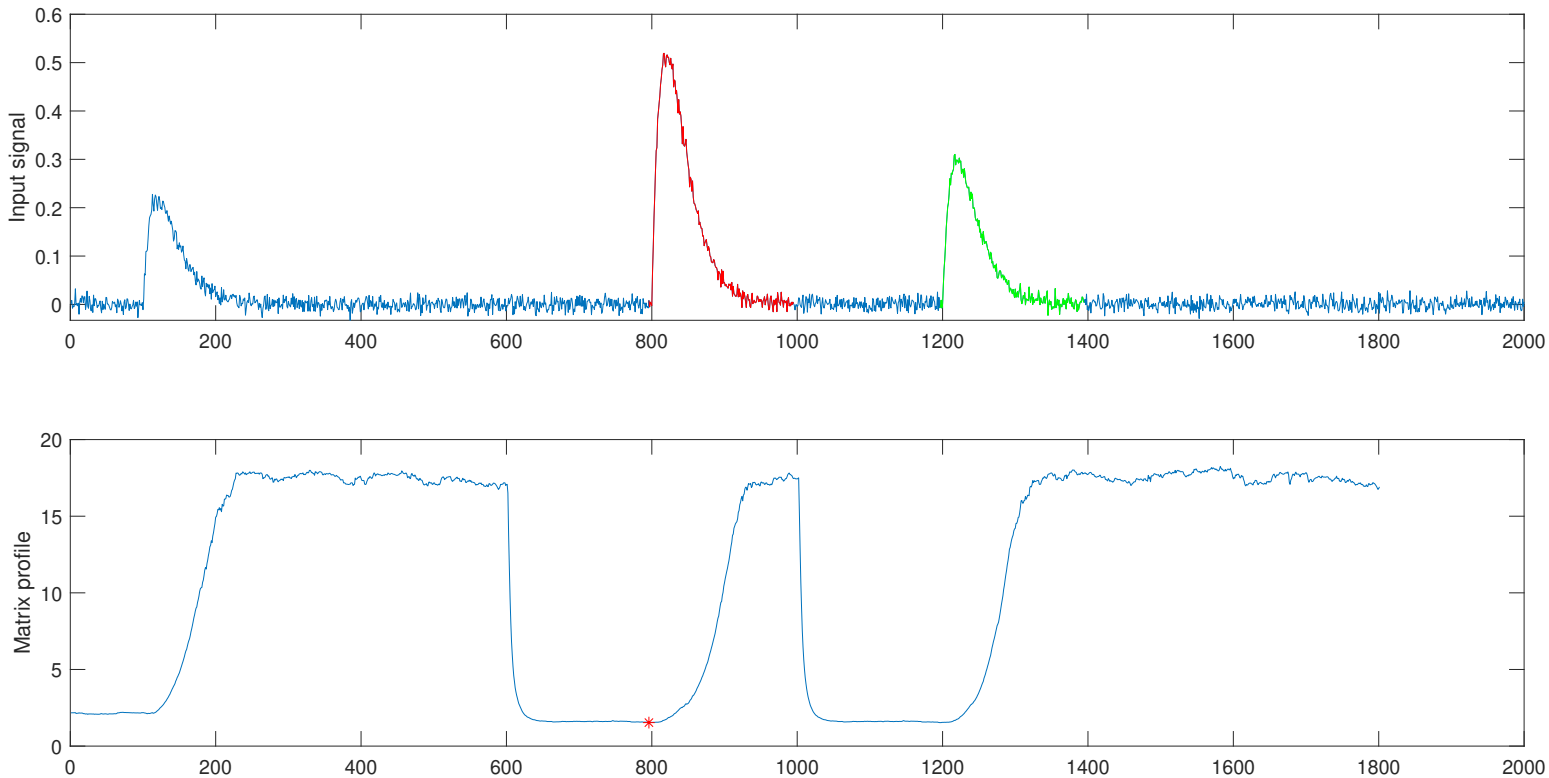
Matrix profile

- In fact, patterns are detectable by only looking at the minimal distance with all subsequences
- Beware of trivial matches ! Only compare with subsequences that do not overlap with the subsequence of interest
- **Matrix profile** [Yeh et al., 2016] : given a pattern length L, compute

$$m[n] = \min_{i>n+L \text{ or } i<n-L} d(x[n : n+L-1], x[i : i+L-1])$$

- Small matrix profiles values indicate that the subsequence has been found elsewhere in the time series, suggesting that it could be a pattern
- Efficient computation with techniques already mentioned (fast sliding distance computation)

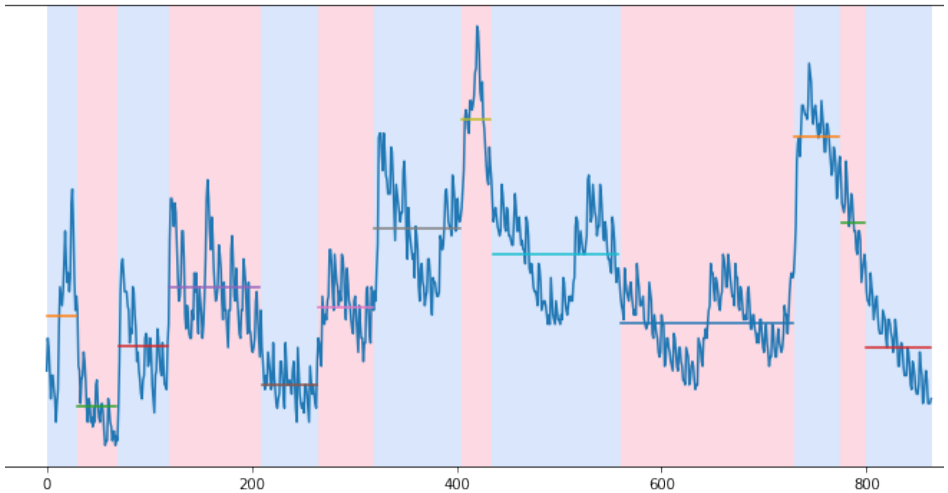
Example



In red : subsequence with minimal matrix profile value. In green : closest subsequence (according to the normalized Euclidean distance)

Contents

Problem 3: Change-Point Detection

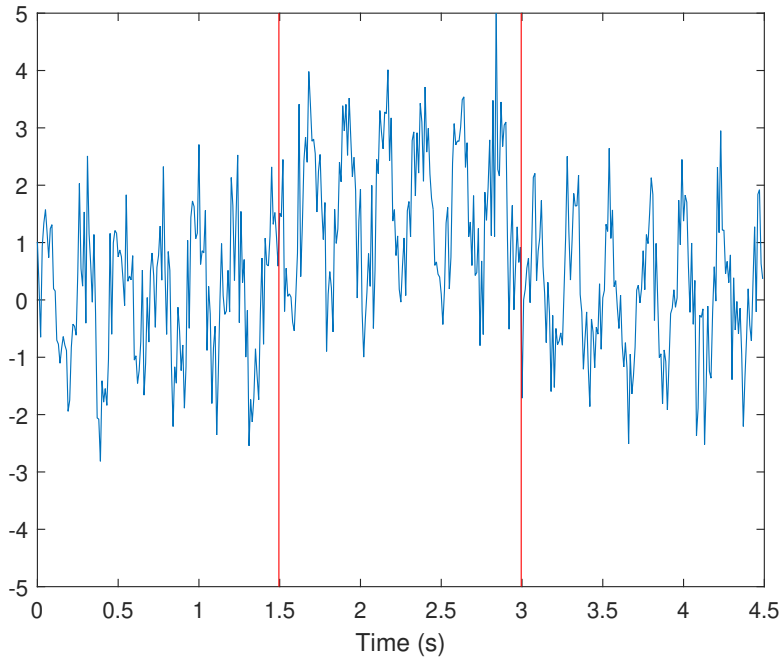


Change-Point Detection

Given a time series x, retrieve the times $(t_1, \ldots, t_K)$ where a significant change occurs

- Necessitates to estimate both the change-points but also the number of changes K
- Highly depends on the meaning given to change

Problem statement



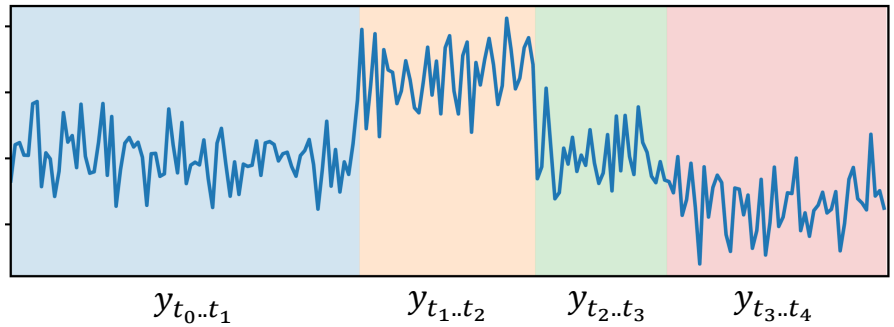
- When the changes are abrupt or when the estimation of the change-points is relevant in the context, we can use **change-point detection** methods
- Let assume that signal $x[n]$ undergoes abrupt changes at times

$$\mathcal{T}^* = (t_1^*, \ldots, t_{K^*}^*)$$

- Goal: retrieve the number of change-points K^* and their times $\mathcal{T}^*$
- One assumption: offline segmentation (but can easily be adapted to online setting) [Truong et al., 2020]

Problem statement

$$(\hat{t}_1, \dots, \hat{t}_K) = \underset{(t_1,\dots,t_K)}{\operatorname{argmin}} \sum_{k=0}^{K} c(x[t_k : t_{k+1}])$$

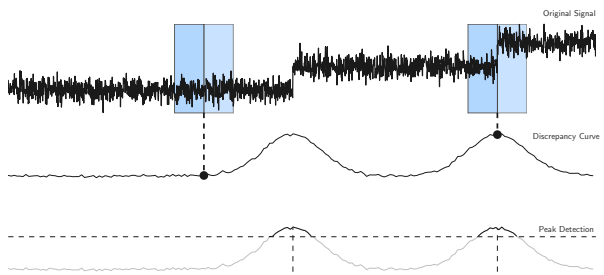


Cost function $c(.)$

- Measures the homogeneity of the segments
- Choosing $c(.)$ conditions the type of change-points that we want to detect
- Often based on a probabilistic model for the data

Problem solving

- Optimal resolution with dynamic programming
- Approximate resolution (sliding windows...)



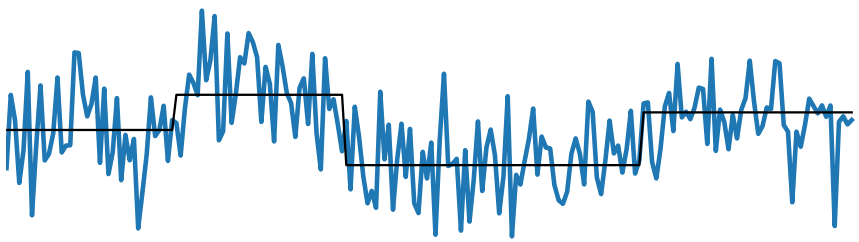
Change in mean

The most popular is indubitably the L2 norm [Page, 1955]

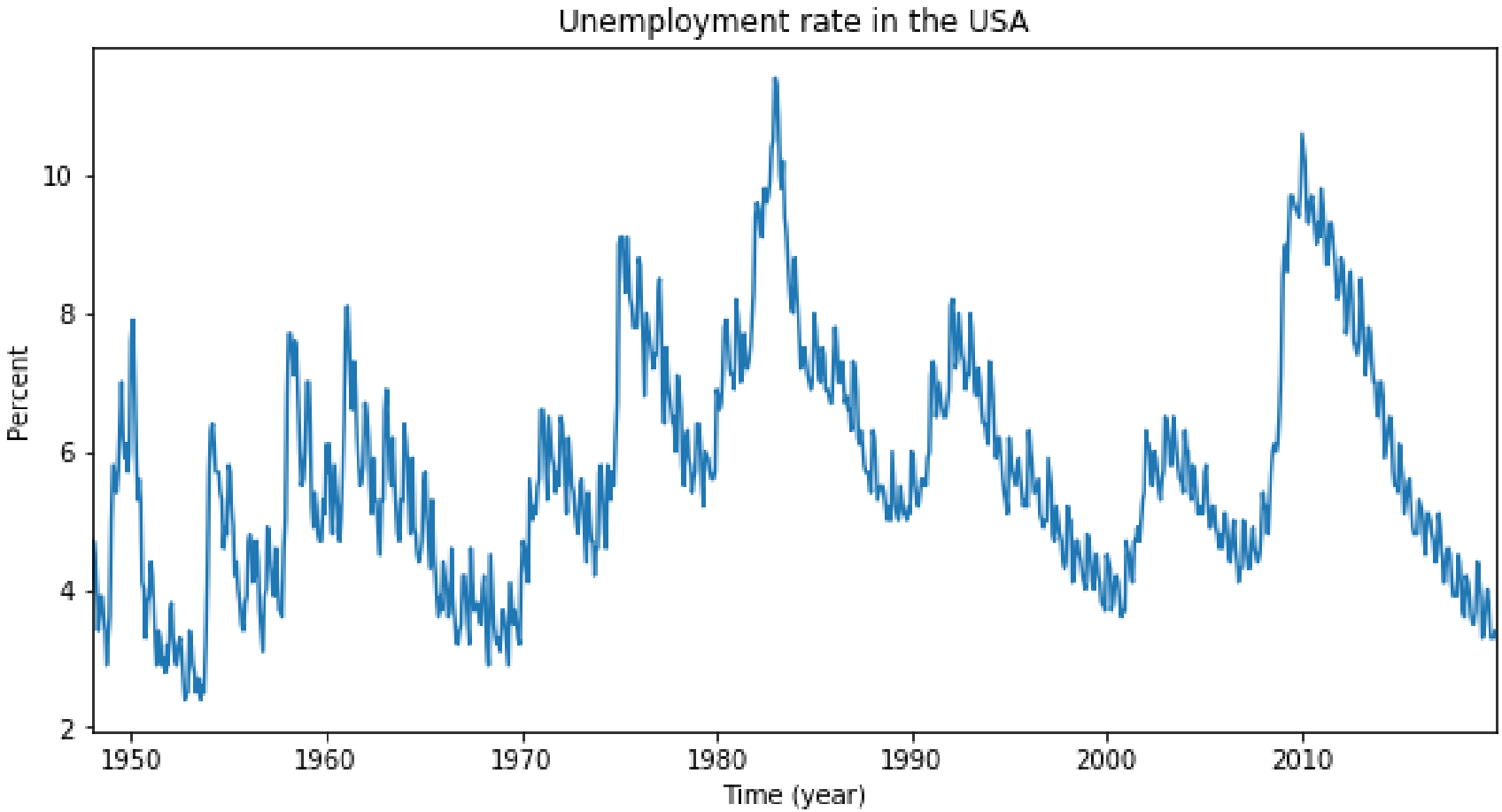
$$c_{L_2}(x[a:b]) = \sum_{n=a+1}^{b} \|x[n] - \mu_{a:b}\|_2^2$$

where $\mu_{a:b}$ is the empirical mean of the segment $x[a:b]$.

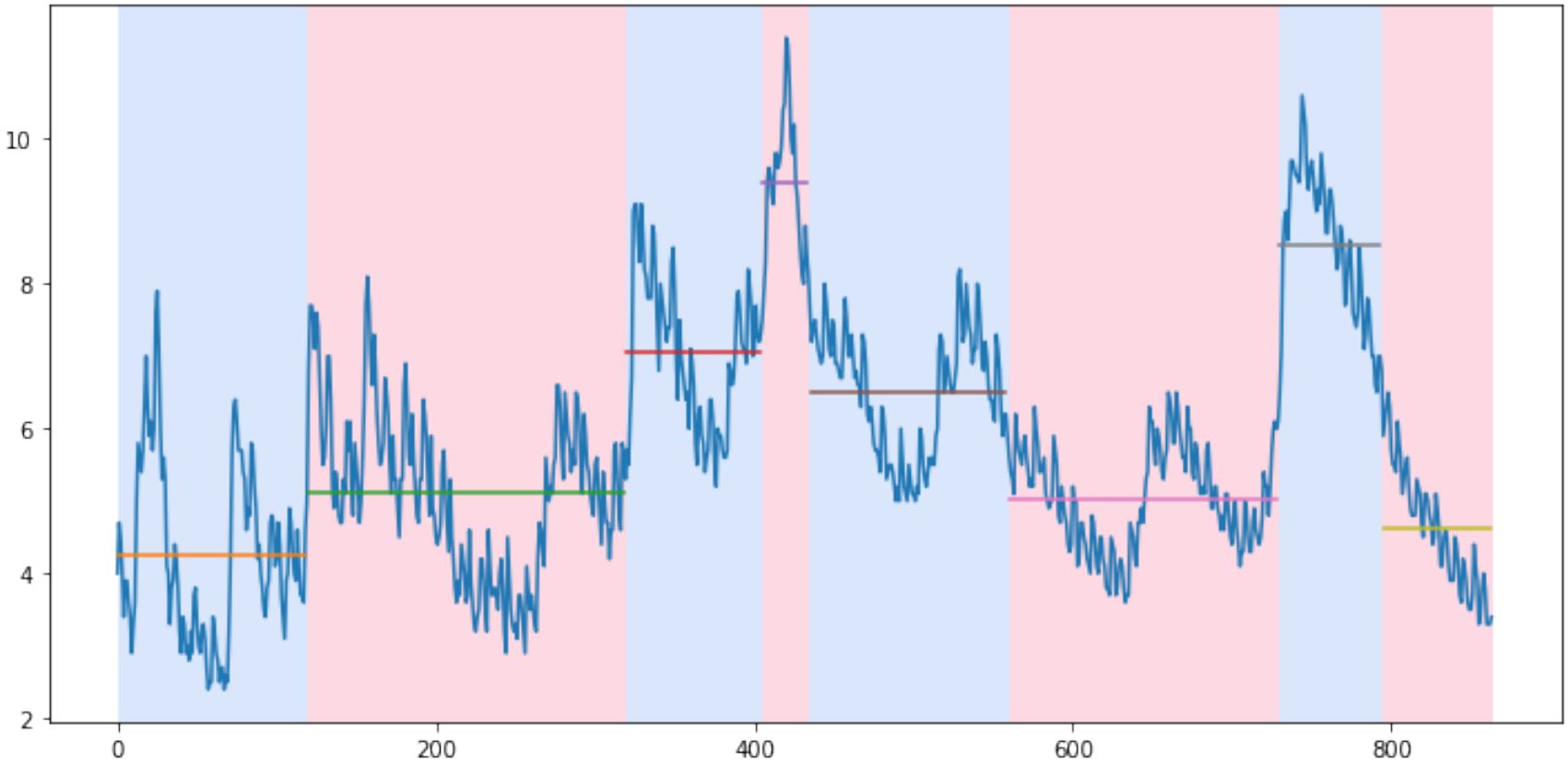
- Particular case of c_{ML} with Gaussian model with fixed variance
- Allows to detect changes in mean



Example

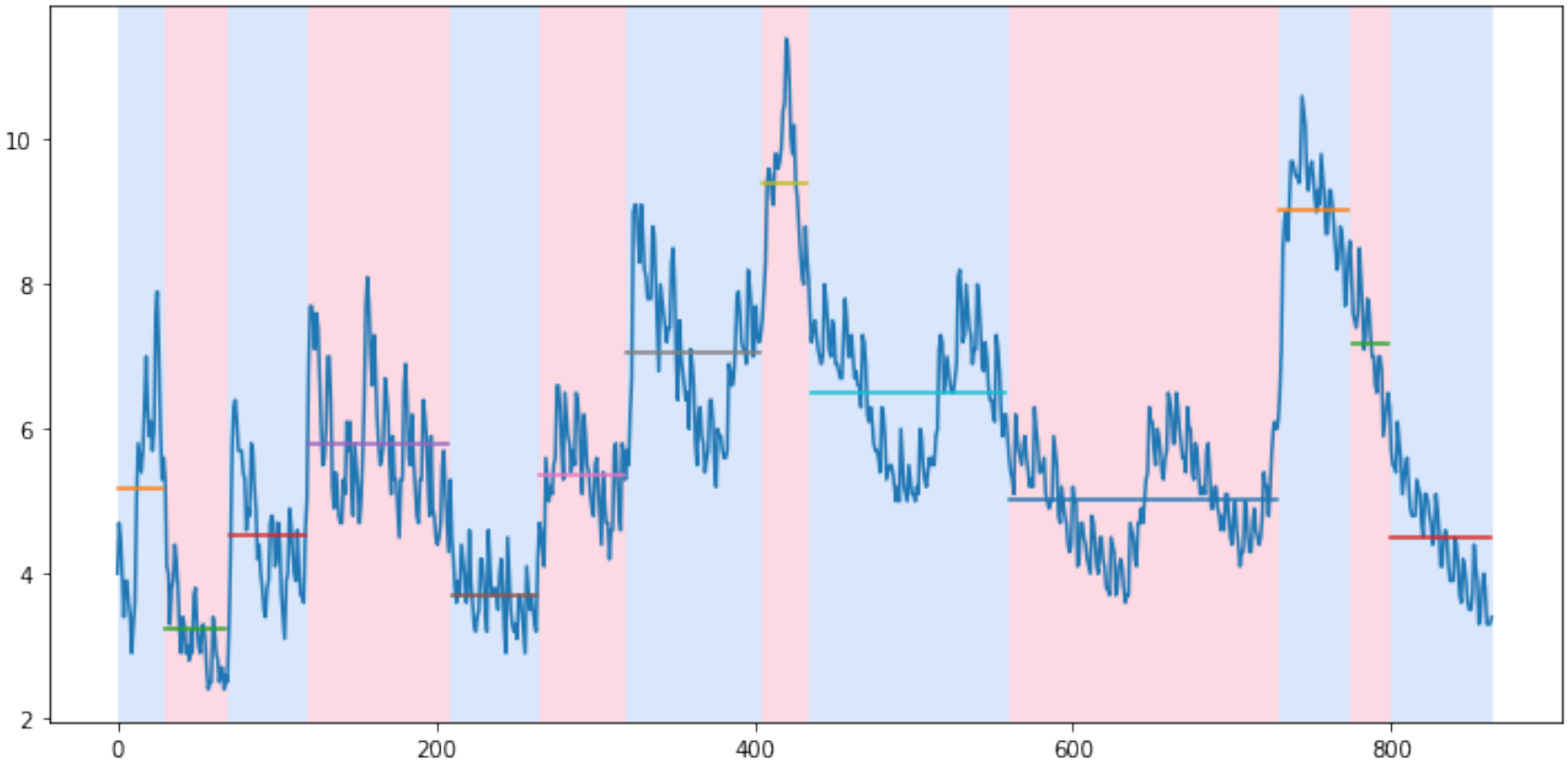

Unemployment rate in the USA
Percent
10
8
6
4
2
1950
1960
1970
1980
1990
2000
2010
Time (year)

Example: Change-Point Detection with c_{L_2}



$K = 7$

Example: Change-Point Detection with c_{L_2}



$K = 12$

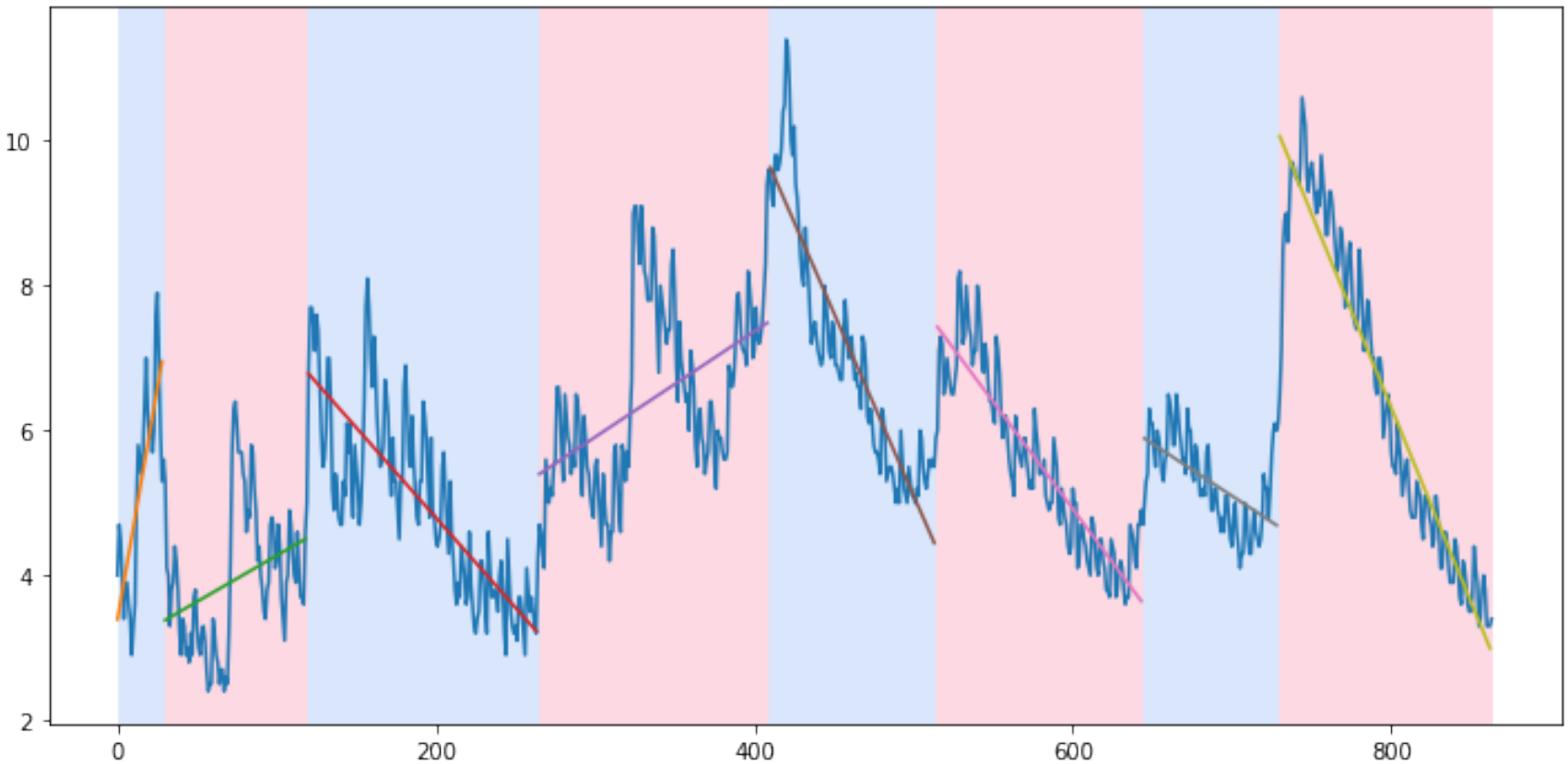
Change in slope and intercept

Change in slope and intercept can be handled in the general context of piecewise linear regression

$$c_{linear}(x[a:b]) = \min_{\boldsymbol{\alpha}} \sum_{n=a+1}^{b} \left\| x[n] - \sum_{i=1}^{M} \alpha_i \beta_i[n] \right\|_2^2$$

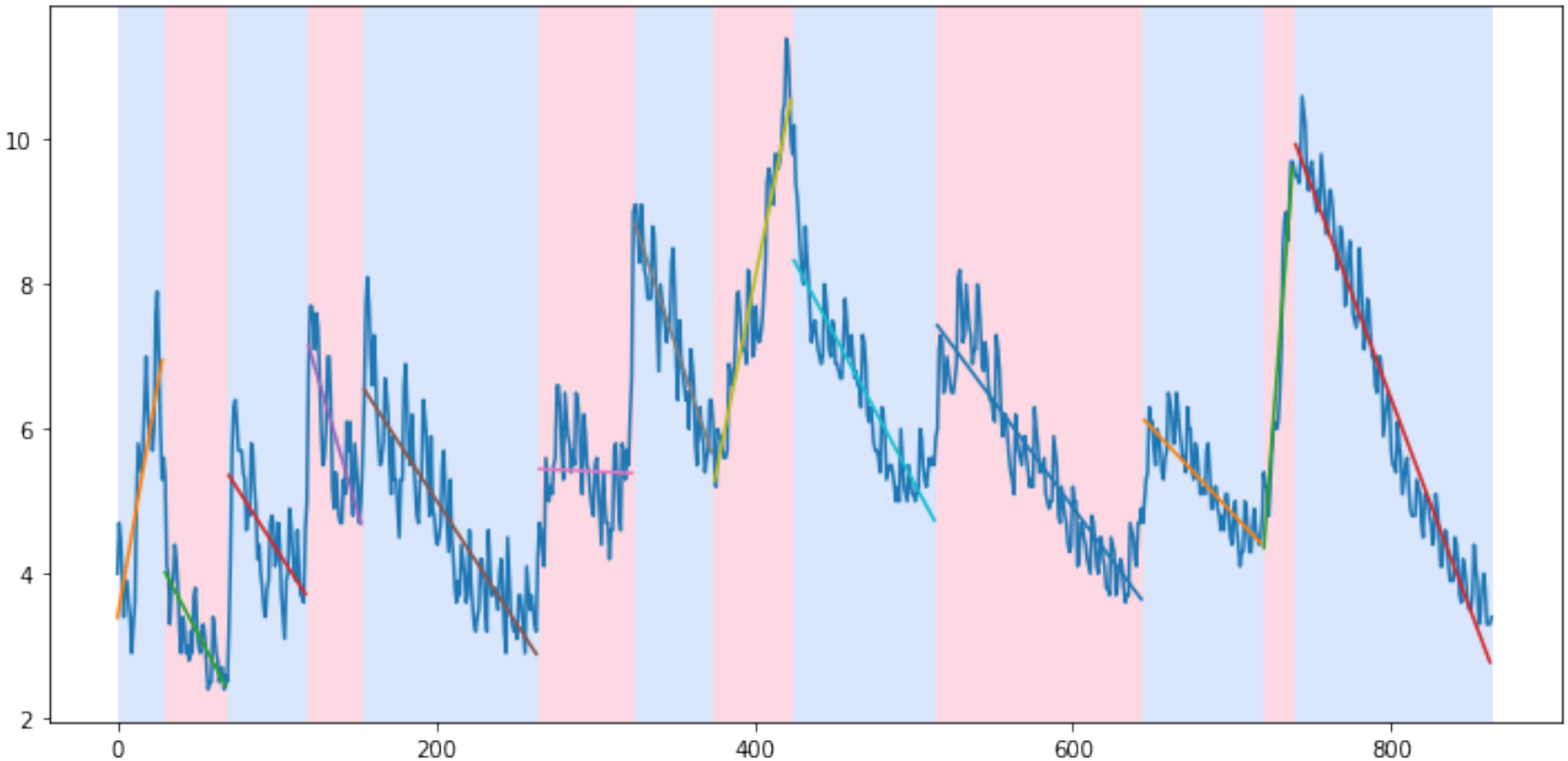
- Functions $\beta_1[n], \ldots, \beta_M[n]$ are covariate functions and we seek for changes in the regression parameters
- Allows to detect changes in trend, seasonality, etc... [Bai et al., 1998]
- For slope and intercept, we choose $\beta_1[n] = 1$ and $\beta_2[n] = n$

Example: Change-Point Detection with c_{linear}



$K = 7$

Example: Change-Point Detection with c_{linear}



$K = 12$

Search method

$$(\hat{t}_1, \dots, \hat{t}_K) = \underset{(t_1,\dots,t_K)}{\operatorname{argmin}} \sum_{k=0}^{K} c(x[t_k : t_{k+1}])$$

Convention : $t_0 = 0$, $t_{K+1} = N$

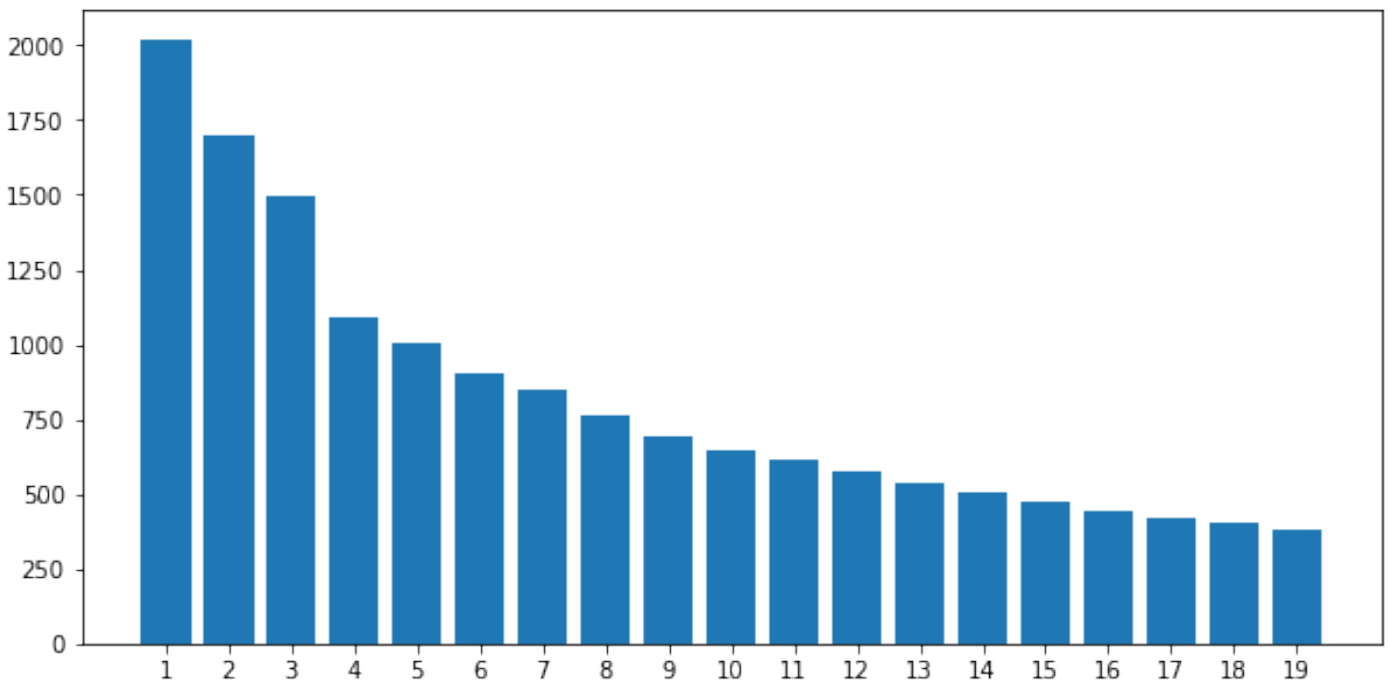
- Several methods can be used to solve this problem with a fixed K
- Optimal resolution with dynamic programming: find the true solution of the problem (but costly)
- Approximated resolution with windows: test for one unique change-point on a window

Finding the number of change points

- In all previously described algorithms, the number of change-point K was supposed to be known
- In practice, this parameter is difficult to set: as such, the total cost $\mathcal{V}(\mathcal{T}, \mathbf{x})$ will always decrease when K increases...
- Three solutions
  - Use heuristics by testing several values of K
  - Use a penalized formulation of the CPD problem to seek for a compromise between reconstruction error and complexity
  - Use supervised approaches from annotated signals

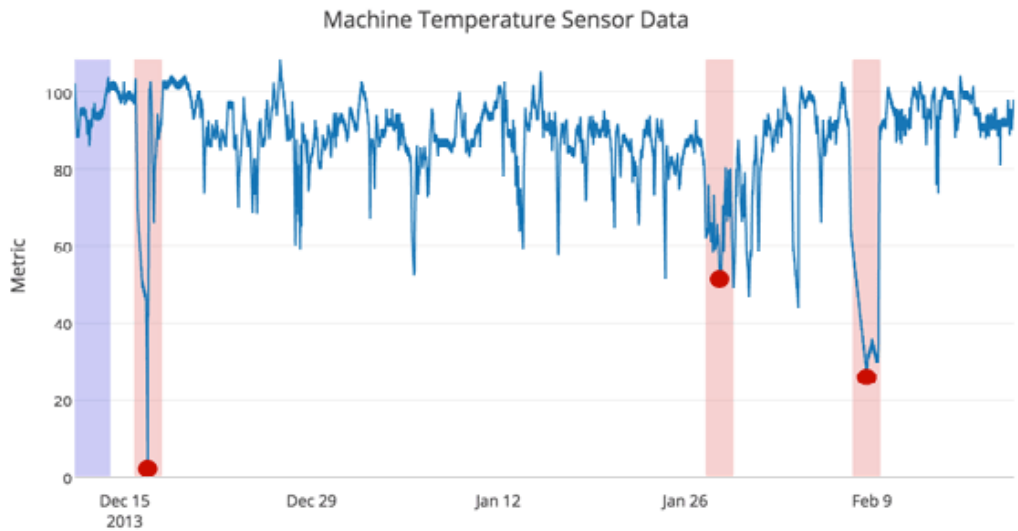
Heuristics for finding the number of change-points

- One easy solution is to test a set of change-points number K from 1 to K_{max} and to compute the sum of costs $\mathcal{V}(\mathcal{T}, \mathbf{x})$
- The *optimal* number of change-points can be estimated by searching for an elbow on the curve of $\mathcal{V}(\mathcal{T}, \mathbf{x})$ as a function of K



Contents

Problem 4: Anomaly Detection

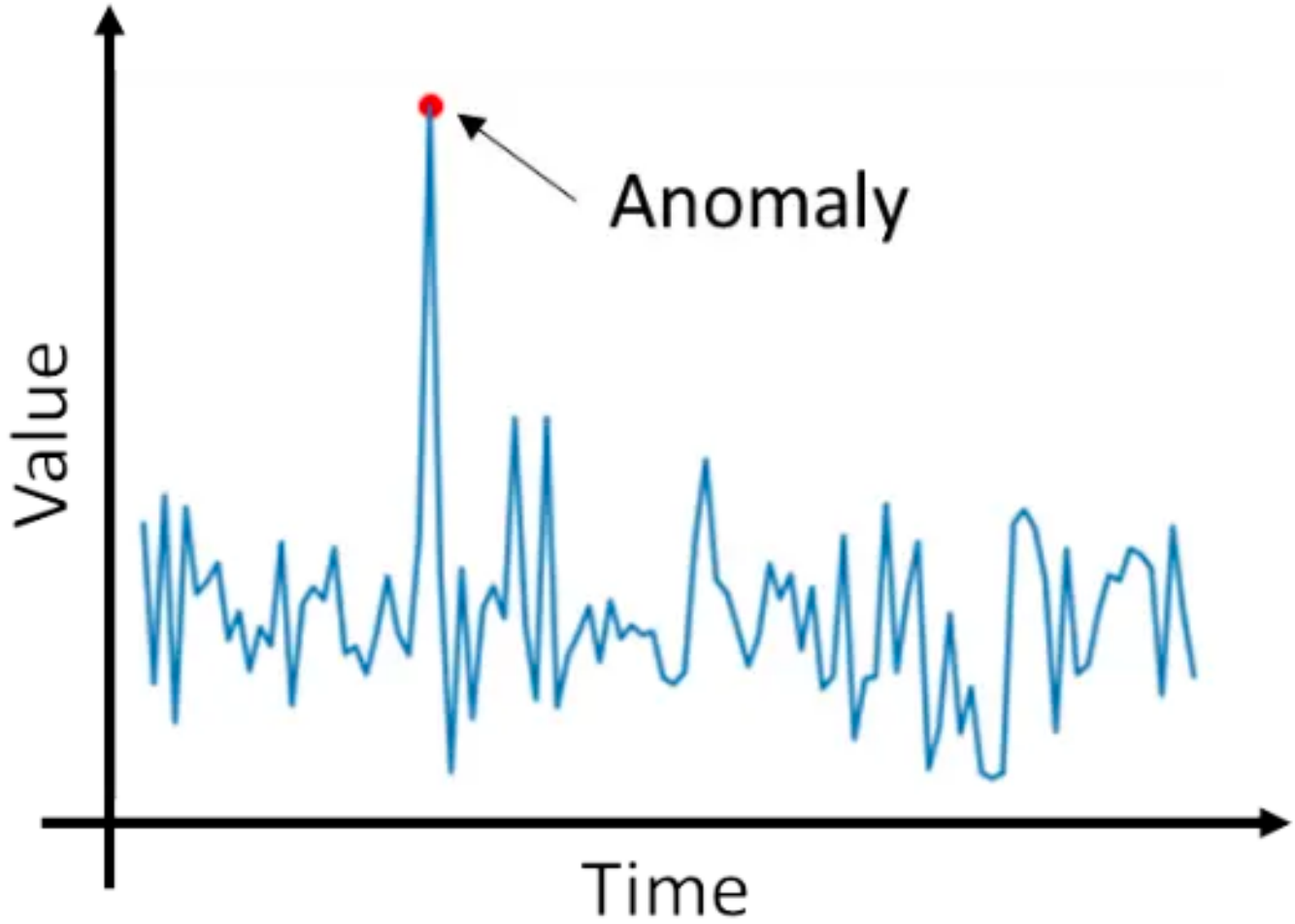


Anomaly Detection

Given a time series x, retrieve the set of samples $\mathcal{T}$ that corresponds to unusual phenomenon

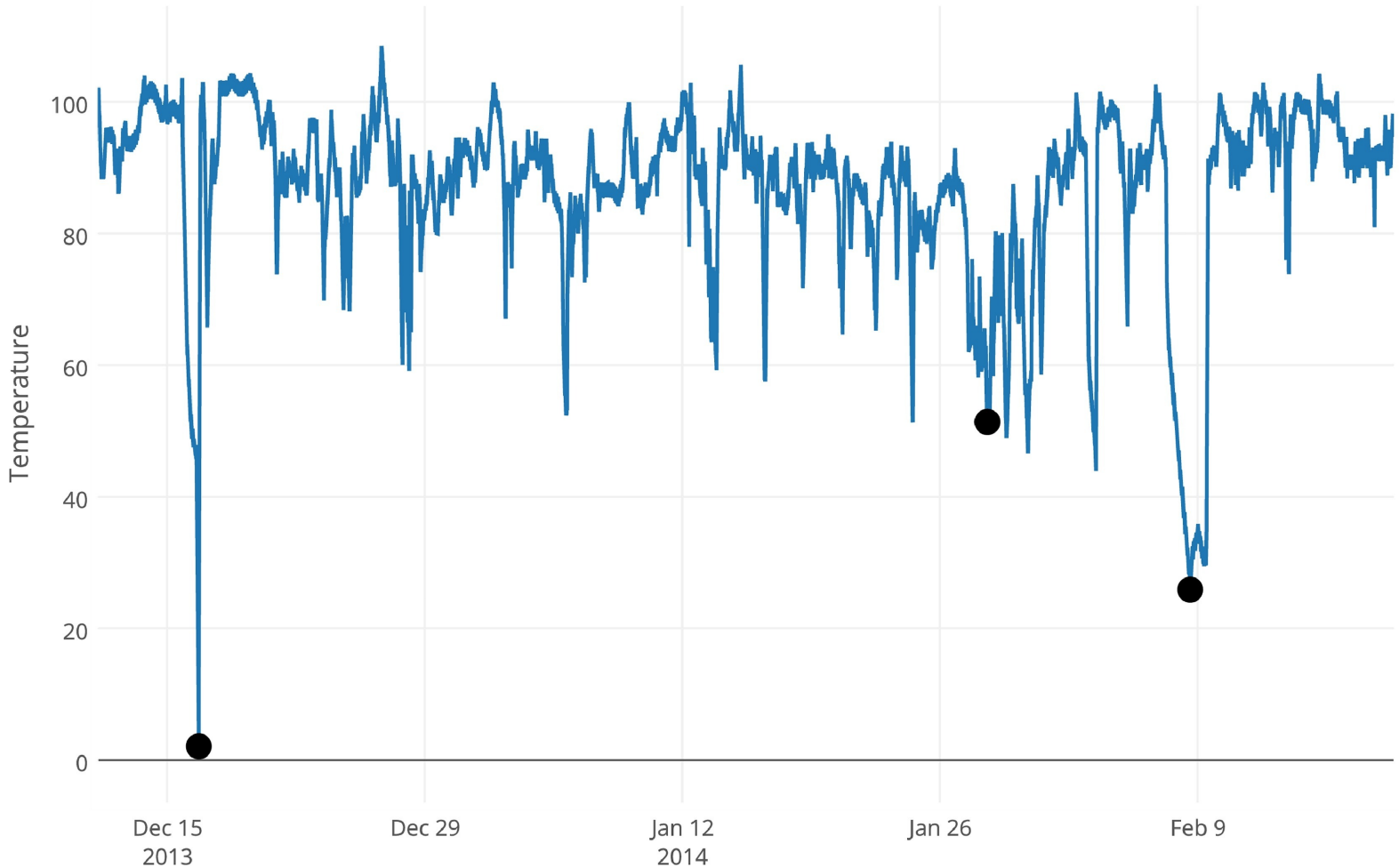
- May include isolated or contiguous samples (see previous slides on outlier detection/removal)
- Highly depends on the meaning given to usual/unusual

Introductory example



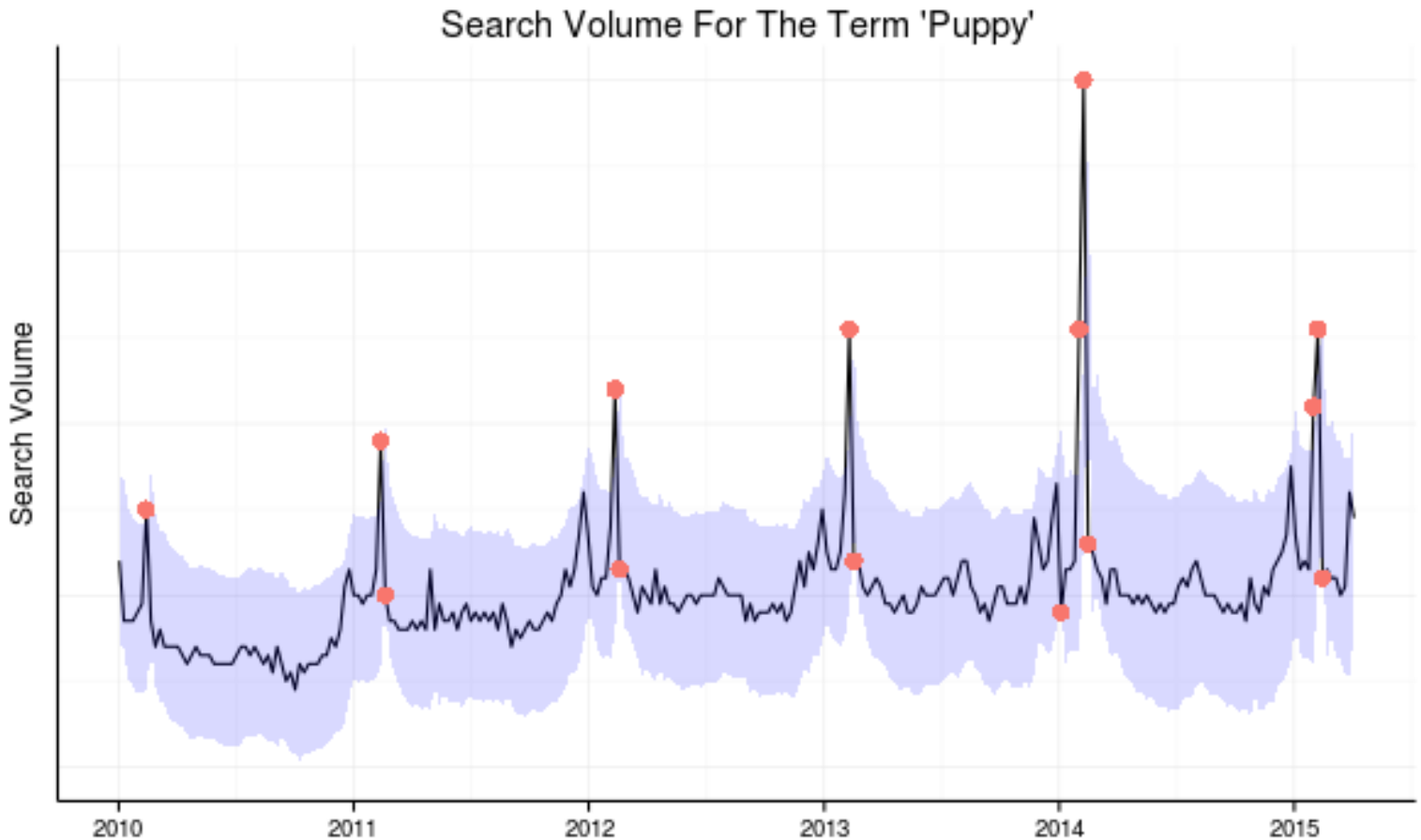
Easy: an anomaly is a too small or too large value (outlier)

Introductory example



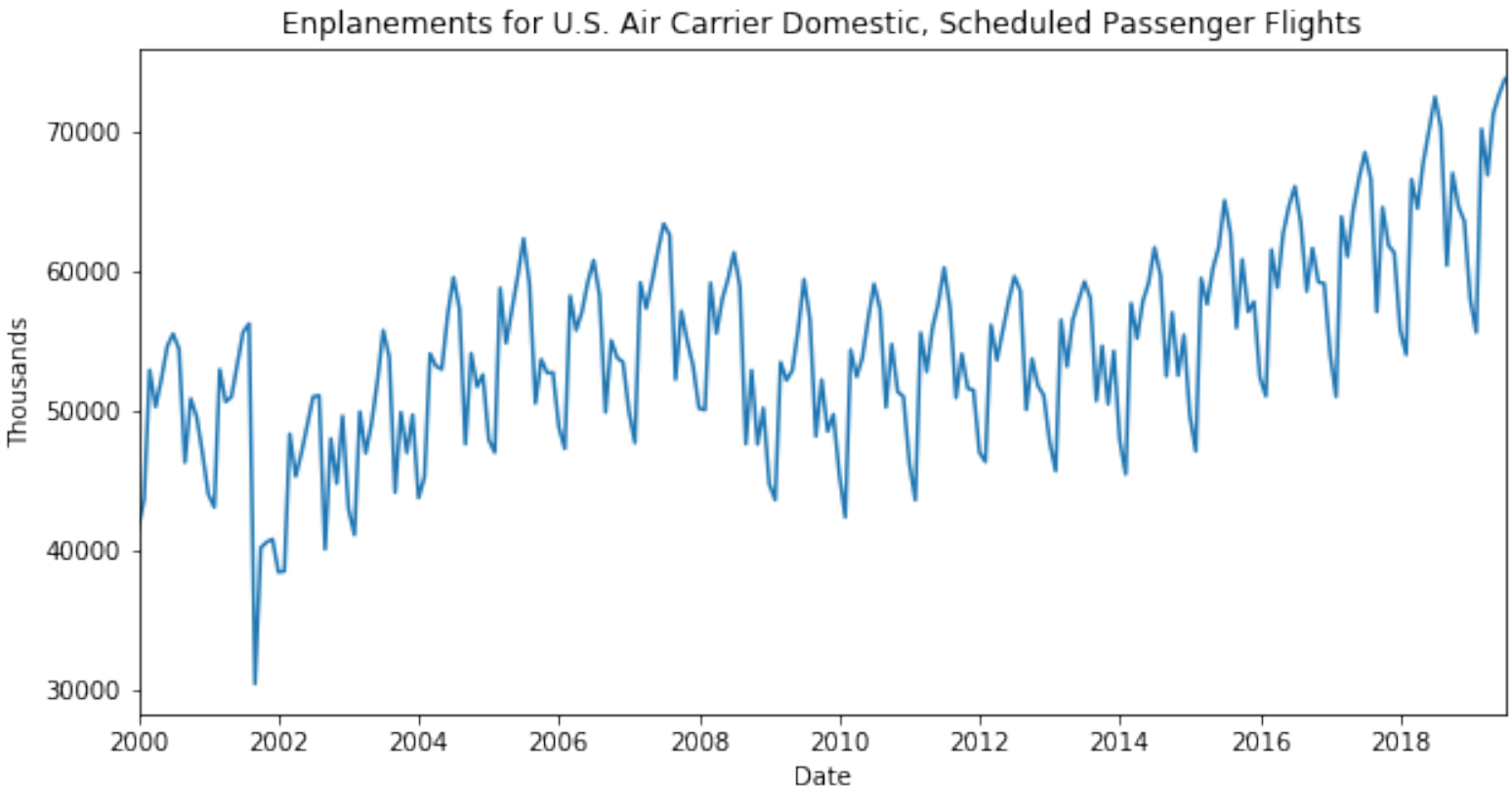
More complex: some small/large values are anomalies

Introductory example



Anomalies depend in the previous values

Introductory example



Anomalies correspond to unusual events

Anomaly Detection

Anomalies can take various forms and have different meanings [Chandola et al., 2009]:

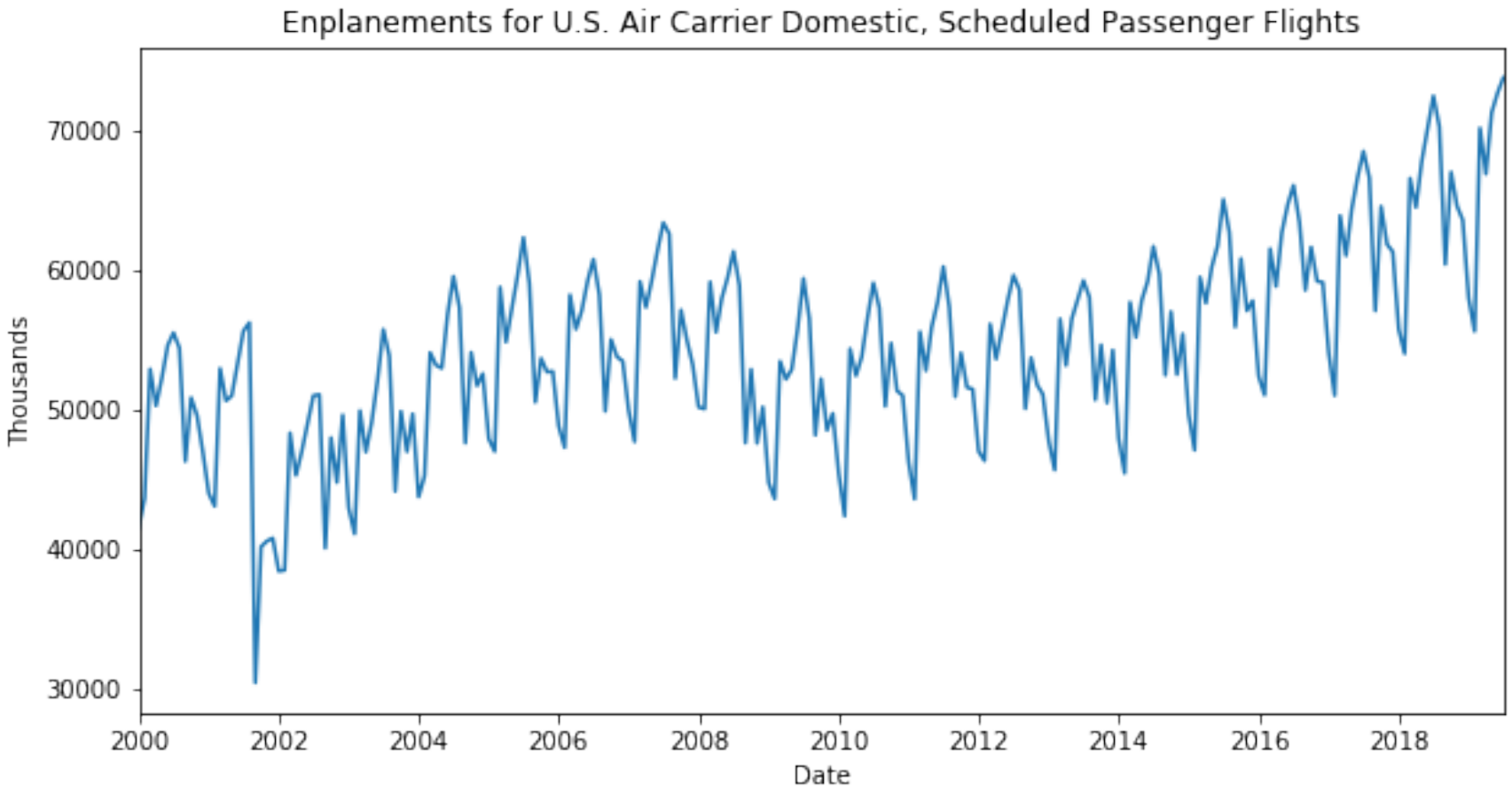
- Outliers, i.e. isolated samples with exceptionally large/low values
- Bursts of outliers, i.e. segments that do not coincide with what is observed usually in the time series (in terms of values)
- Unusual events that breaks the regularity within the time series

Outlier detection

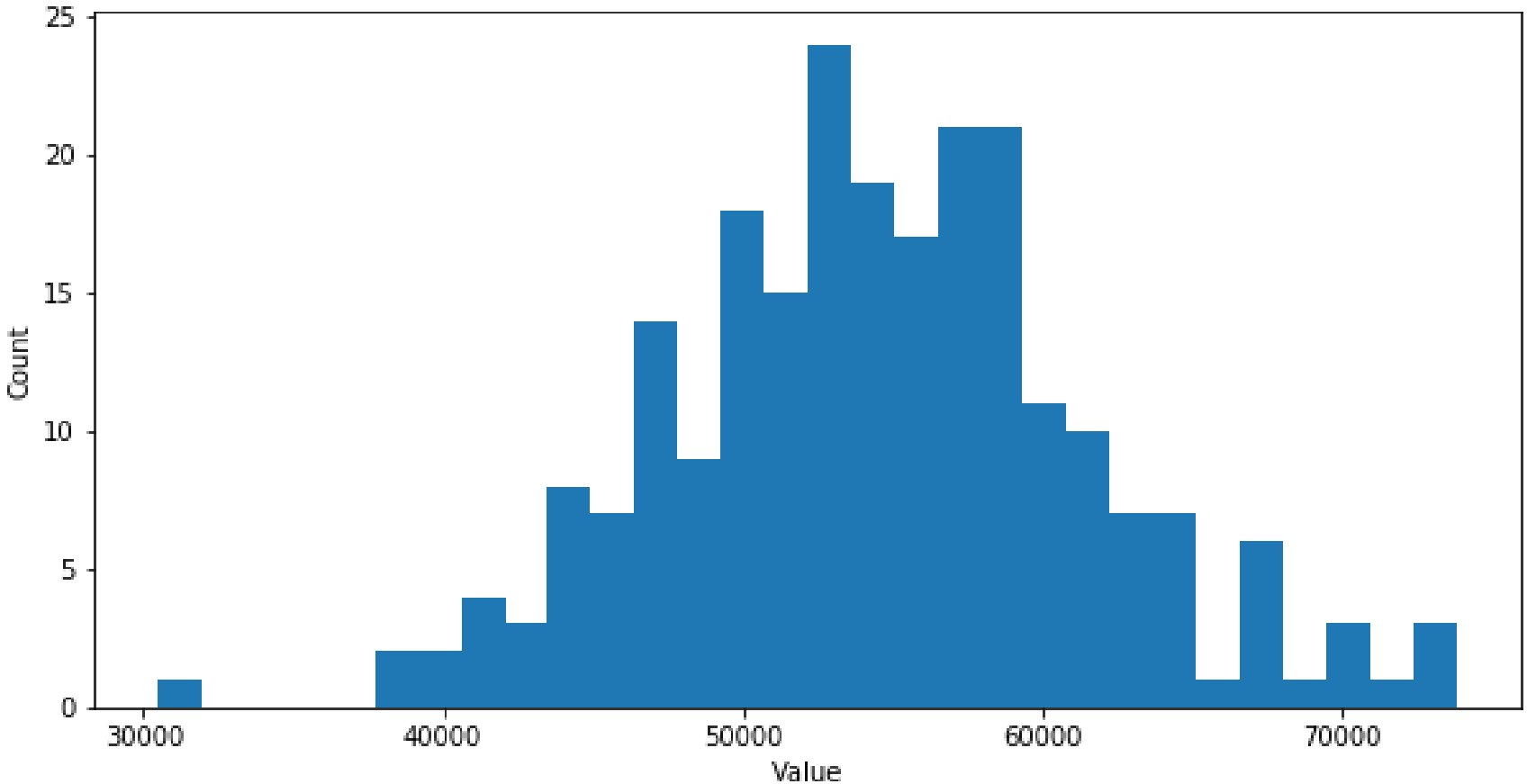
Simple anomalies (isolated samples or contiguous samples) can be detected with techniques already described

- Statistical methods:
  - Global: Histogram visualization to detect aberrant values
  - Adaptive: Threshold-based methods on sliding windows (mean/standard deviation or median)
- Model-based methods:
  - Residual and prediction error (trend+seasonality, sinusoidal or AR model)

Example

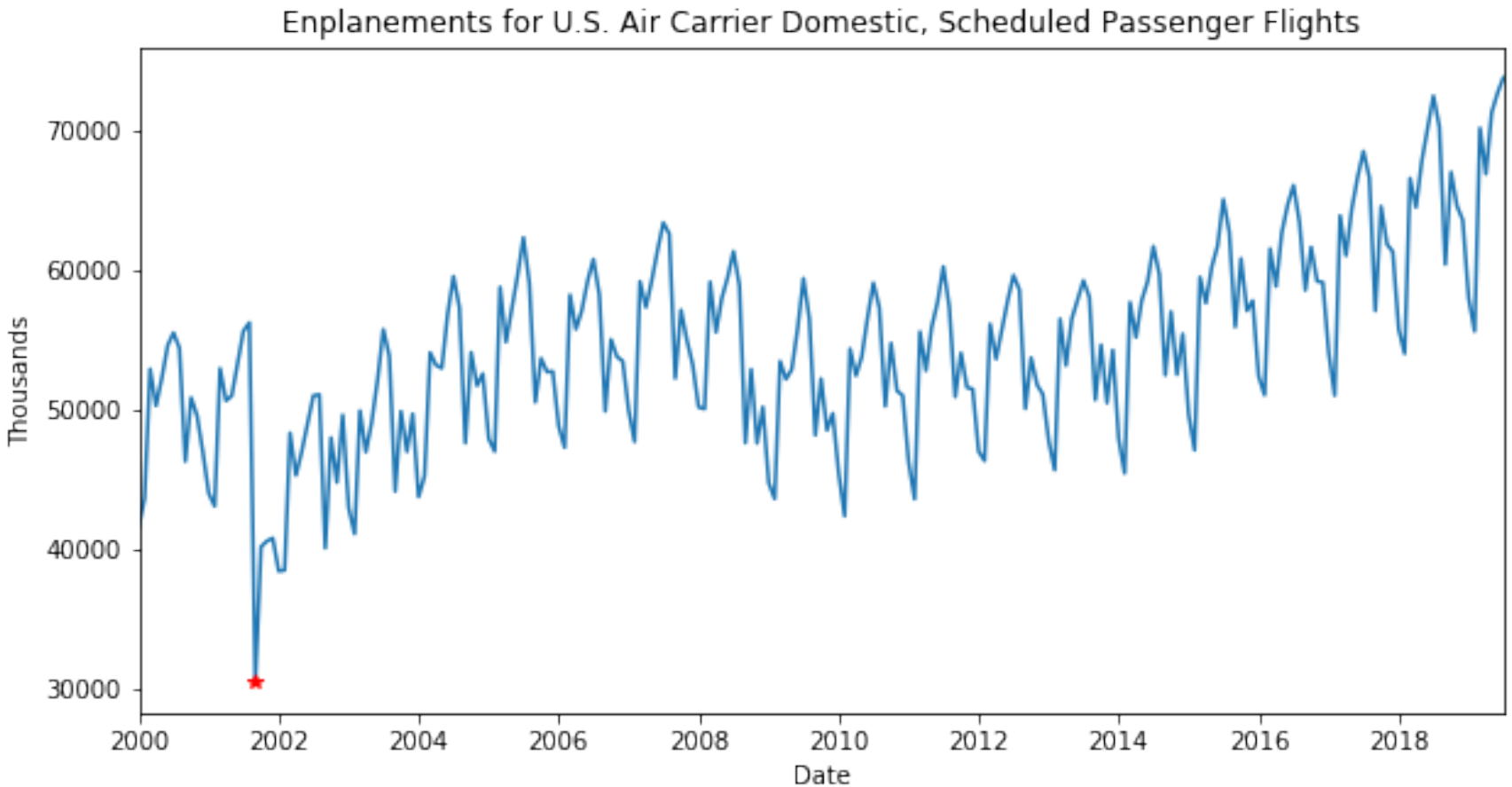

Enplanements for U.S. Air Carrier Domestic, Scheduled Passenger Flights
Thousands
70000
60000
50000
40000
30000
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
Date

Example: Histogram



One outlier can be considered as an anomaly

Example: Histogram



Only one detected anomaly

Adaptive statistical methods

- The main idea is to use sliding window and to perform a statistical test for outlier detection
- Contrary to histogram, these methods allow to take into account the local context but careful, time information is lost! Only the distribution of values is used for detection.
- Multitude tests can be used but the most common are
  - **Mu/sigma** [Roberts, 2000]:

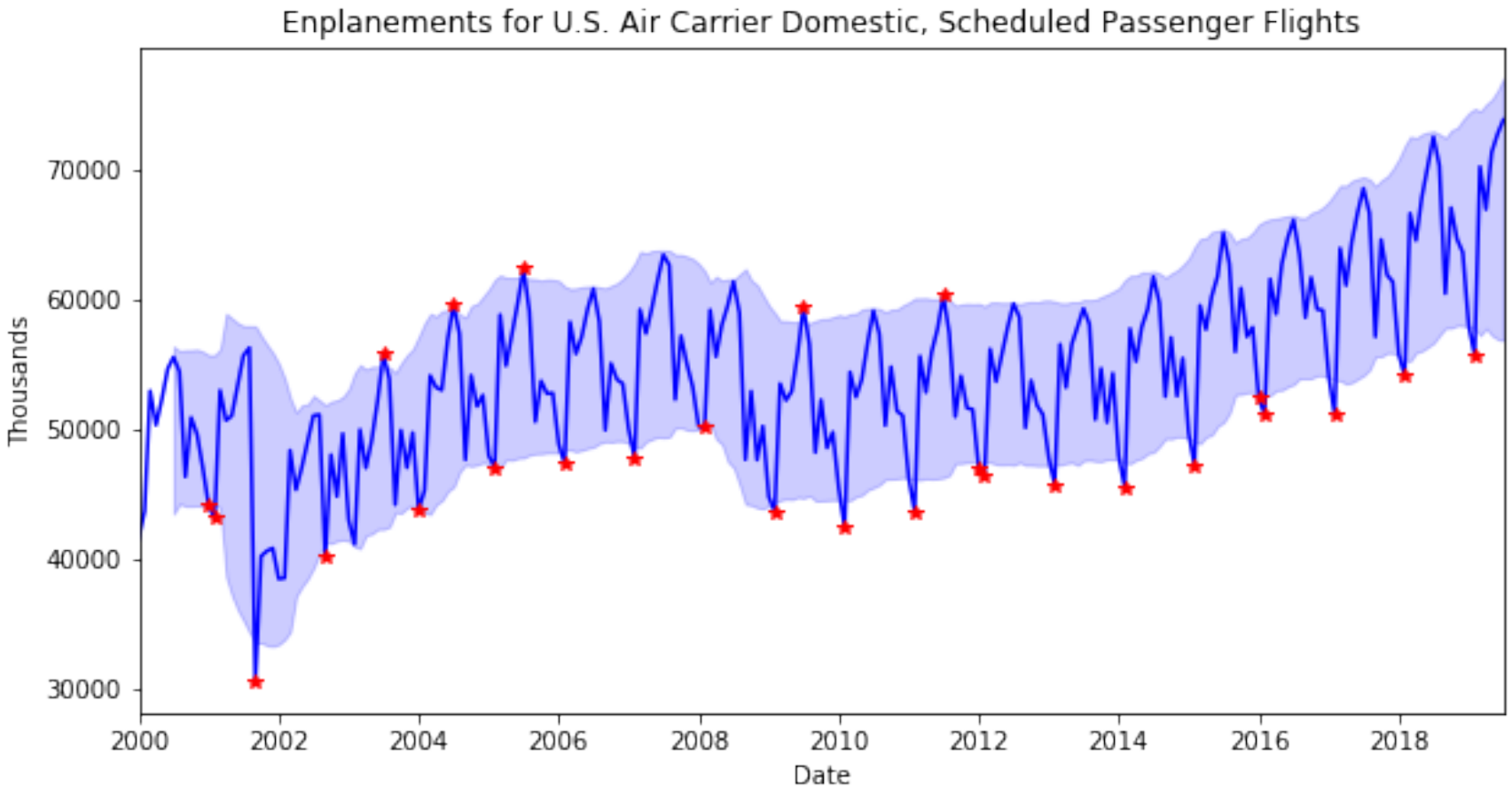
$$|x[n] - \mu_n| > \lambda\sigma_n$$

    where μ_n and σ_n are respectively the local mean/standard deviation around sample n and λ a threshold.
    Under i.i.d. Gaussian assumption, $\lambda = 1 \rightarrow 68\%$, $\lambda = 2 \rightarrow 95\%$, $\lambda = 3 \rightarrow 99.7\%$
  - **Median/median absolute deviation** [Leys et al., 2013]:

$$|x[n] - \text{med}_n| > \lambda\, \text{mad}_n$$

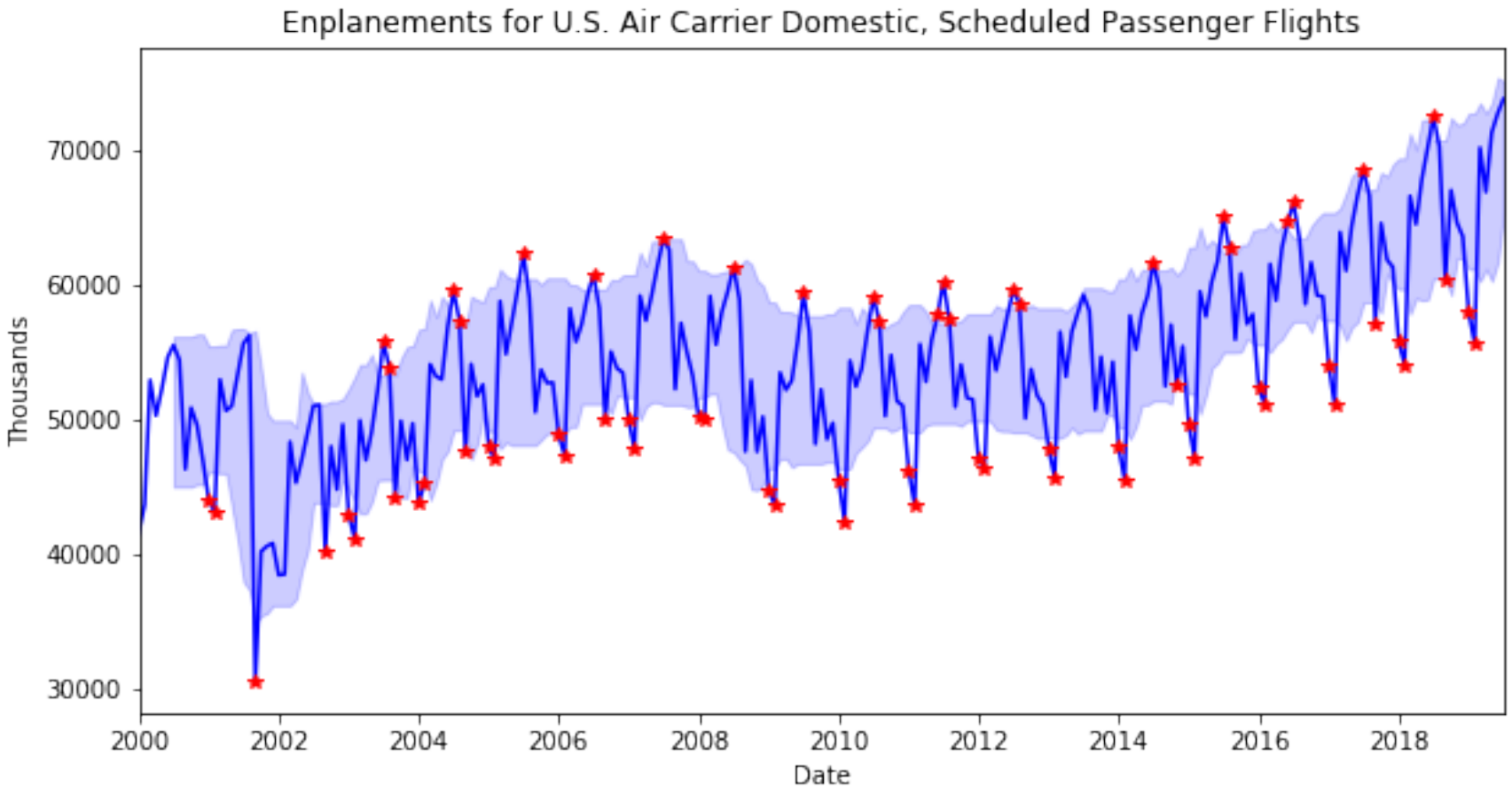
    where med_n and mad_n are respectively the local median/median absolute deviation around sample n and λ a threshold.

Example: Mu-Sigma



Mu-Sigma, $\lambda = 1.5$, window length of 12 samples

Example: Med-Mad

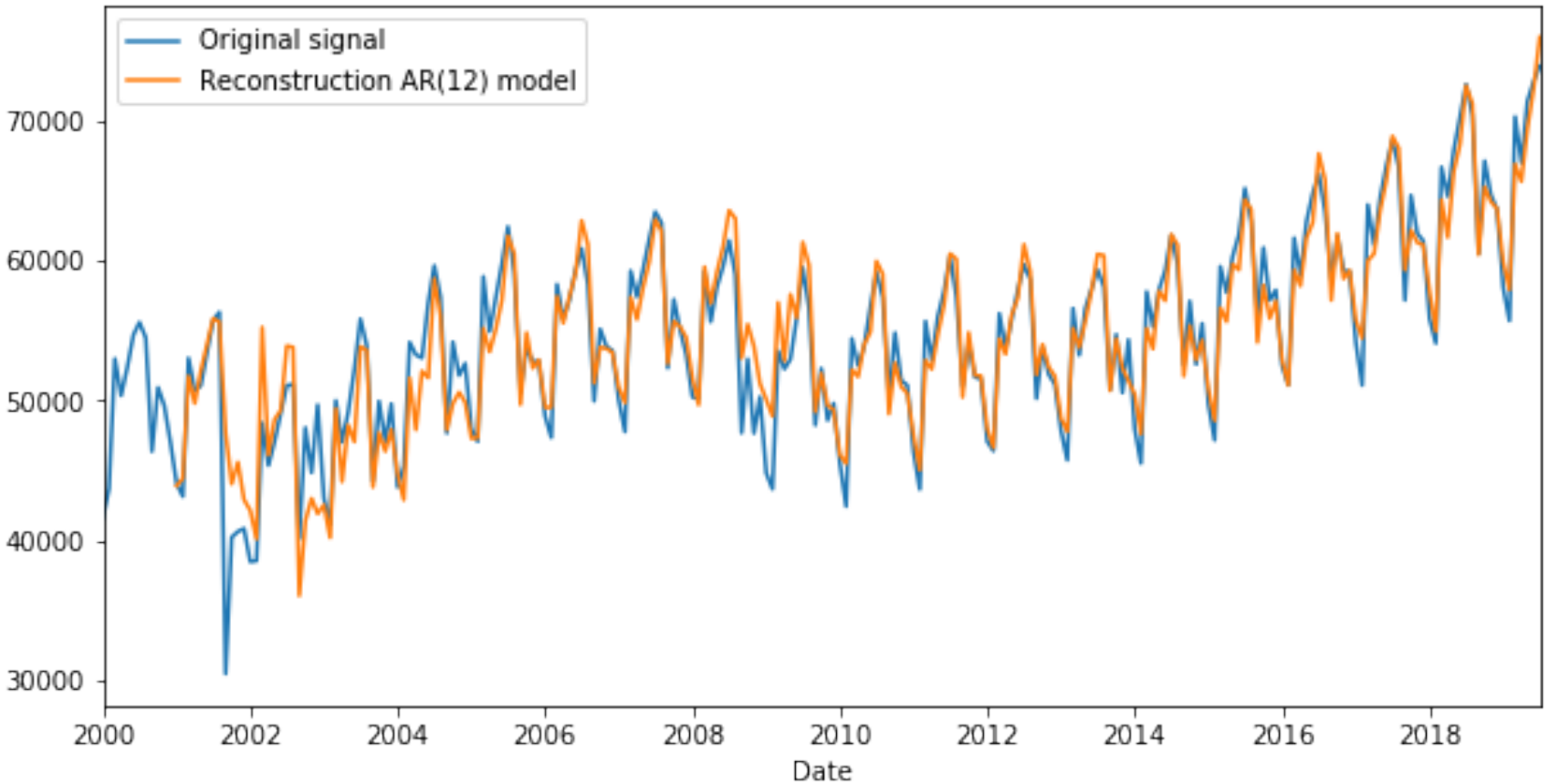


Med-Mad, $\lambda = 1.5$, window length of 12 samples

Model-based anomaly detection

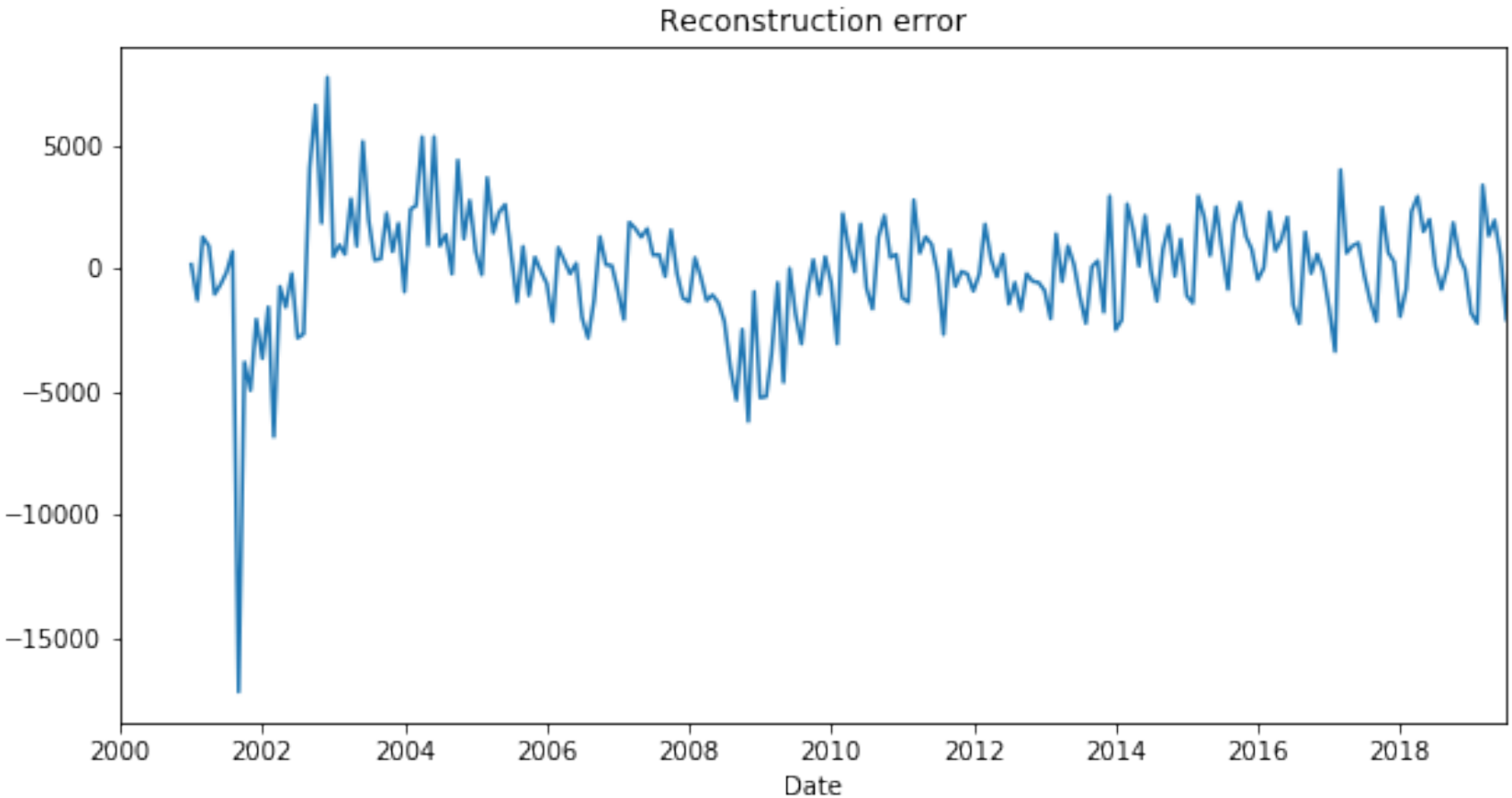
- Idea: use a time series model to detect anomaly [Yamanishi et al., 2002; Hill et al., 2010]
- Advantage: truly takes into account the temporal aspects
- Three steps:
  1. Choose an adequate model and learn the parameters
  2. Compute the prediction/signal reconstruction
  3. Anomalies are samples that diverge from the model

Example: AR model



AR model with $p = 12$

Example: AR model



Anomaly also changes the prediction of the next p samples

Distance-based methods

- Some anomalies may be more complex to detect as they are not characterized by aberrant values but by a new behavior that was not previously seen in the time series
- In this case, anomalies can only be defined as a divergence from a normal behavior
- This task is the dual of the task already seen for Pattern Detection/Extraction, and the same techniques can therefore be used
- Instead of searching for repetitive patterns, we are searching for non-repetitive events!

Unsupervised anomaly detection

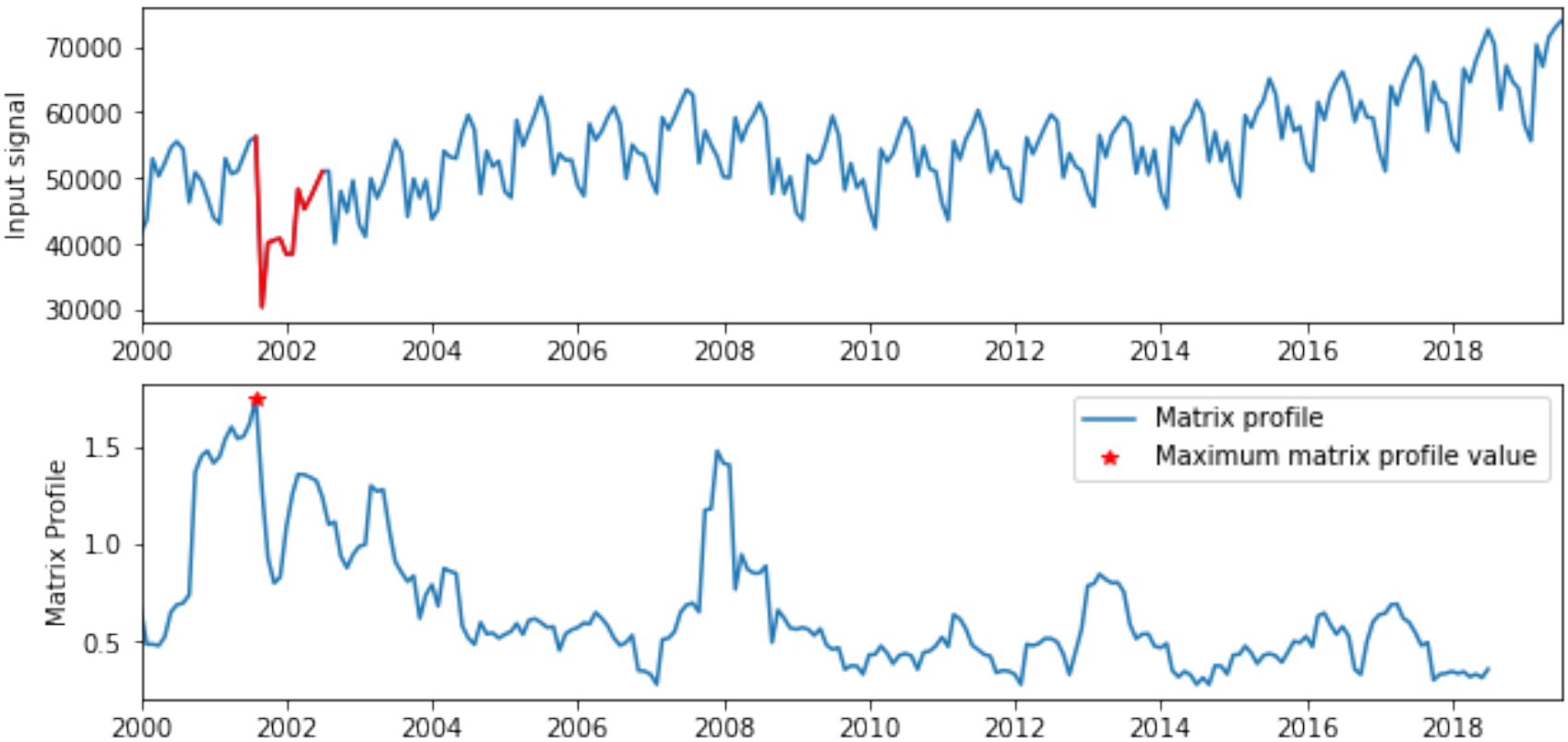
- Reminder : **Matrix profile** [Yeh et al., 2016] : given a pattern length L, compute

$$m[n] = \min_{i>n+L \text{ or } i<n-L} d(x[n : n+L-1], x[i : i+L-1])$$

- Small matrix profiles values indicate that the subsequence has been found elsewhere in the time series, suggesting that it could be a pattern
- Efficient computation with normalized Euclidean distance

What about large values in the matrix profile?

Example: matrix profile



Matrix profile with window of length $L = 12$ months

Matrix profile

- By examining large values on the matrix profile, anomalies can be detected
- Subsequences that are *far* from all subsequences in the signal: likely to correspond to new behaviors
- Advantages: no need for a parametric model
- Necessitates to have a rough idea of the scale of the anomaly (parameter L)