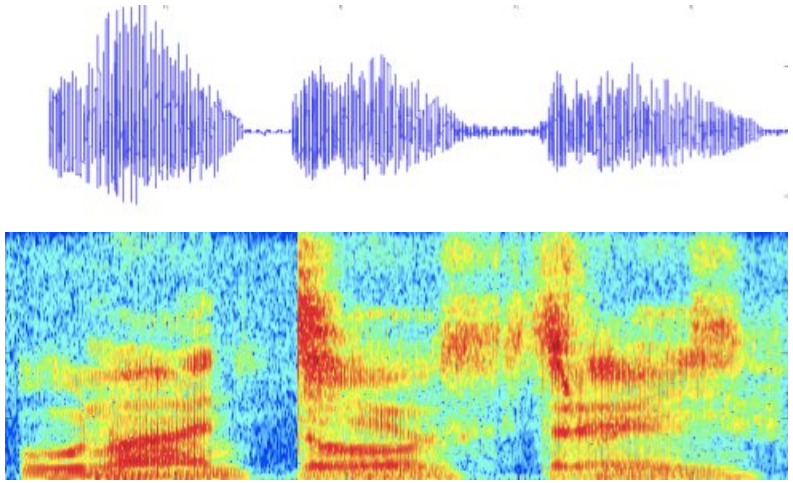
Estimation theory & parametric identification



Digital signals manipulation and filters synthesis

N. Mechbal, **M. Rébillat**, M. Guskov *[PIMM, ENSAM]*
marc.rebillat@ensam.eu

Robotic context

Standard digital signals synthesis

Resampling

Time alignement

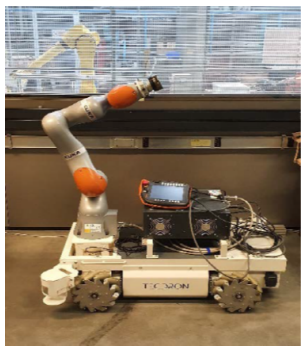
Continuous-time filtering of signals

Discrete-time filtering of signals

Digital filters synthesis

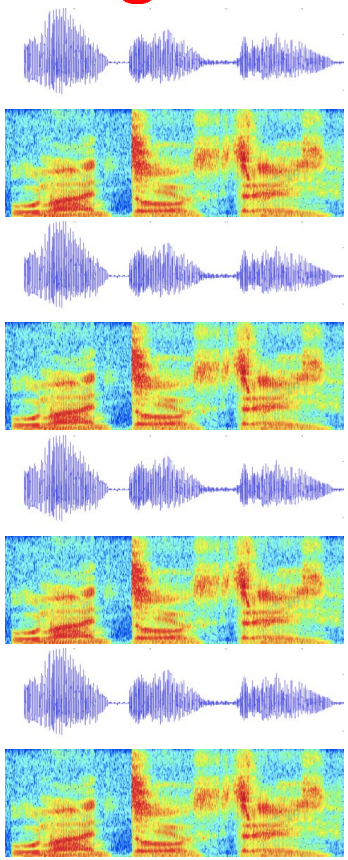
Model estimation of robotic systems [1, 2, 3]

Robots



$\Rightarrow$

Digital signals



$\Rightarrow$

Numerical models

Transfer function

$$H(p) = \frac{Y(p)}{U(p)}$$

State-space model

$$\dot{X} = f(X, U)$$

$$Y = g(X, U)$$

Objective: Estimate a reliable model and its parameters for a robotic system from measurements.

Course objectives

- **With M. Guskov**: Building models of flexible manipulator.
- **With N. Mechbal**: Model parameters estimation.

Within this part of the course

Understanding the signal processing steps of measurements coming from robotic sensors that are necessary to perform model parameters estimation:

1. How do I represent usefully a digital signal?
2. How am I sure to control the data acquisition chain?
3. How do I process reliably acquired signals?
4. Illustration in robotics applications.

Useful references *are made available on the last slide for more details [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].*

Part #2: Digital signals and filters synthesis

How do I generate excitation signals and filter and manipulate raw digital signals?

Three example of signals:

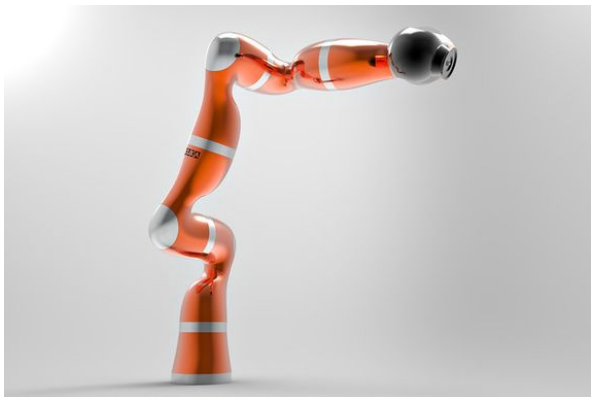
Sine sweep

Pink noise

Voice in noise

- How to **generate excitation signals** for experiments?
- How to **design digital filters** corresponding to requirements?
- How to **apply these filters** and to **manipulate digital signals** in practice?

Why focusing on signal synthesis?



For dynamic models identification, the design of adequate input signals is required. Some examples are:

- Excitation that **covers the frequency range** that corresponds to the use of the dynamical system.
- Excitation that **corresponds to realistic solicitations** encountered by the dynamical system.
- Excitation for to **mimic perturbations** seen by the dynamical system.

Note: What is presented here is not exhaustive and optimal excitation signals are highly dependent on the targeted system and application.

Standard deterministic test signals: sine sweeps

Sine sweep definition: A sine sweep is a sinusoidal signal with a frequency varying over time:

$$\boxed{x(t) = A \sin\left[\phi(t)\right]}$$

Several categories of sine sweep exists depending on the chosen variation of the instantaneous frequency $f(t) = \frac{1}{2\pi}\frac{d\phi}{dt}$ with time:

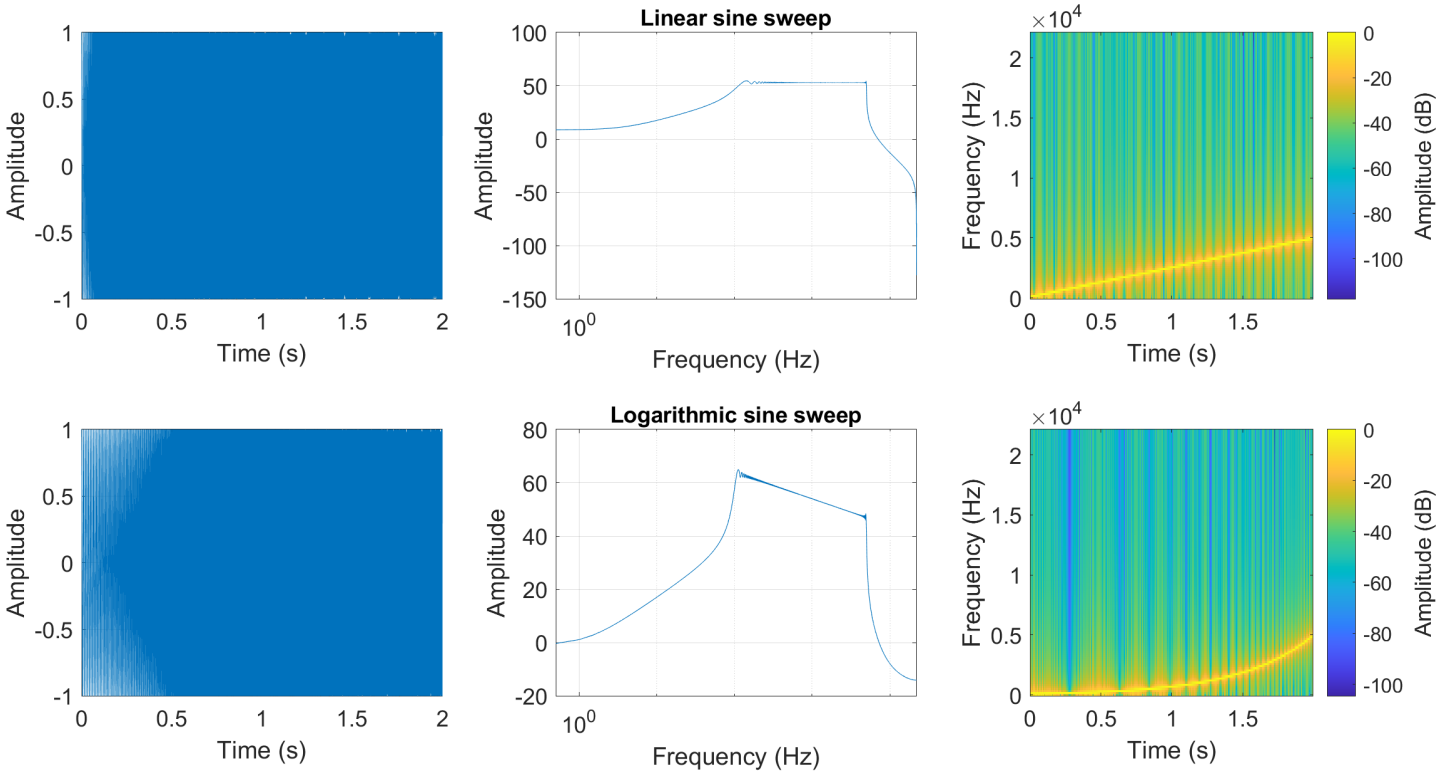
- **Linear** sweep:
$$\phi(t) = 2\pi\left(f_0 + (f_1 - f_0)\frac{t}{T}\right)t + \phi_0$$

- **Logarithmic** (or *exponential*) sweep:
$$\phi(t) = 2\pi\frac{f_1 T}{ln(f_2/f_1)}\left(e^{(t/T)ln(f_2/f_1)} - 1\right) + \phi_0$$

with parameters:

- A: the sine sweep amplitude.
- T: the sine sweep duration.
- f_1 and f_2: the start and stop frequencies of the sine sweep.

Comparison of linear and logarithmic sine sweeps



- No visible difference in the temporal domain.
- Flat spectrum for linear seep and spectrum in -6dB/decade for the logarithmic one.
- Linear evolution of frequency versus exponential evolution of frequency.

Sine sweep in practice

- **MATLAB/OCTAVE implementation:**

$$\boxed{\mathbf{Y = chirp(T,F0,T1,F1,method)}}$$

  - Y is samples of a linear swept-frequency signal at the time instances defined in array T.
  - The instantaneous frequency at time 0 is F0 Hertz.
  - The instantaneous frequency F1 is achieved at time T1.
  - Available methods are *'linear'*, *'quadratic'*, and *'logarithmic'*.
- **Sweep selection**: Choosing between a linear or a logarithmic sweep is not straightforward. It depends on:
  - **The background noise spectrum:** if it is flat prefer the linear sweep and if its spectrum is in $1/f$, prefer the logarithmic ones. Both cases are common in practice.
  - **The upcoming signal processing steps:** depending of the following signal processing algorithms, a linear or a logarithmic sweep maybe more adequate.

Standard random test signals: noises

Noise definition: a set of of random samples following a given power spectral density and chosen among a given probability density function.

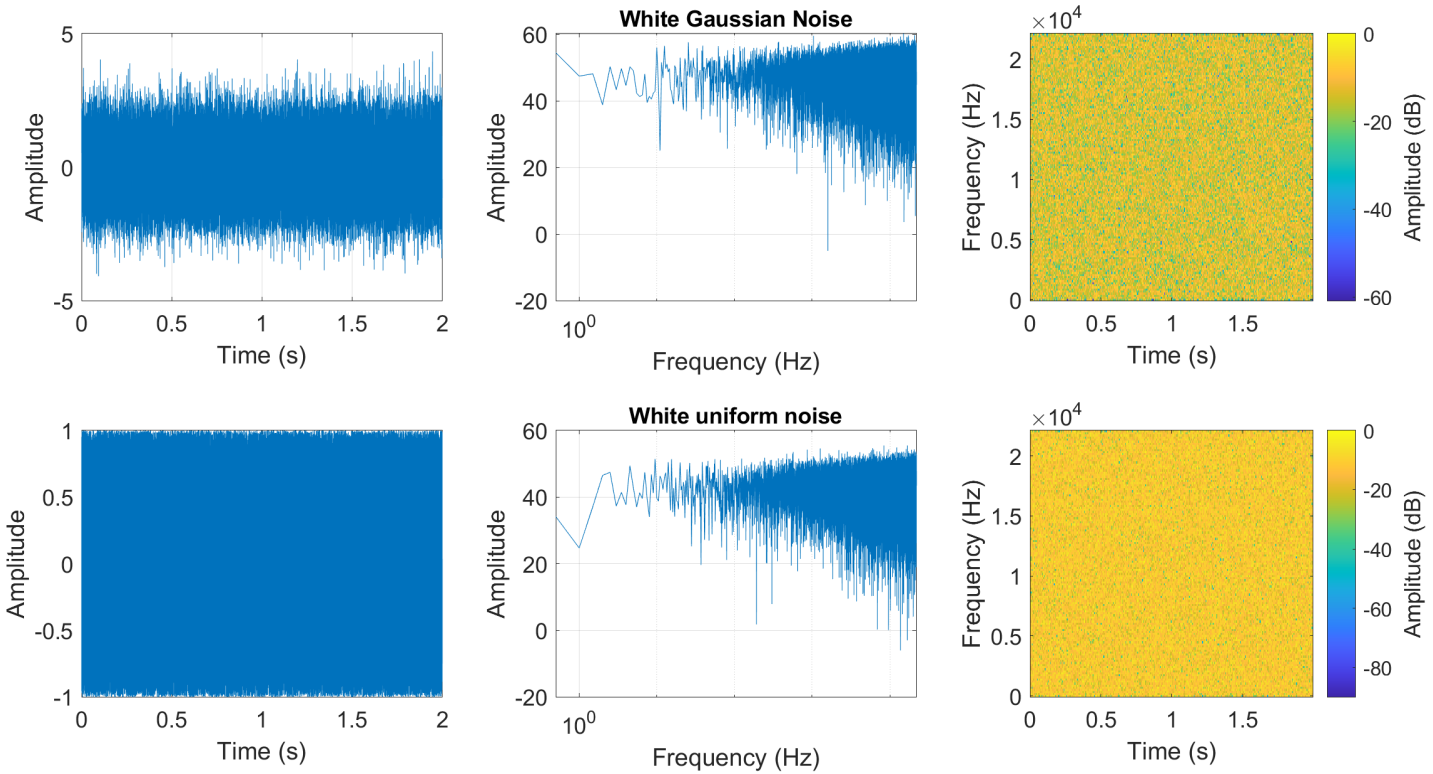
Spectral density function common choices:

- **White noise**: power spectral density equally spread among frequencies which is equivalent to no correlation between samples.
- **Pink noise**: power spectral density decreasing by -6 dB/decade which is equivalent to the introduction of some correlation between samples.

Probability density function common choices:

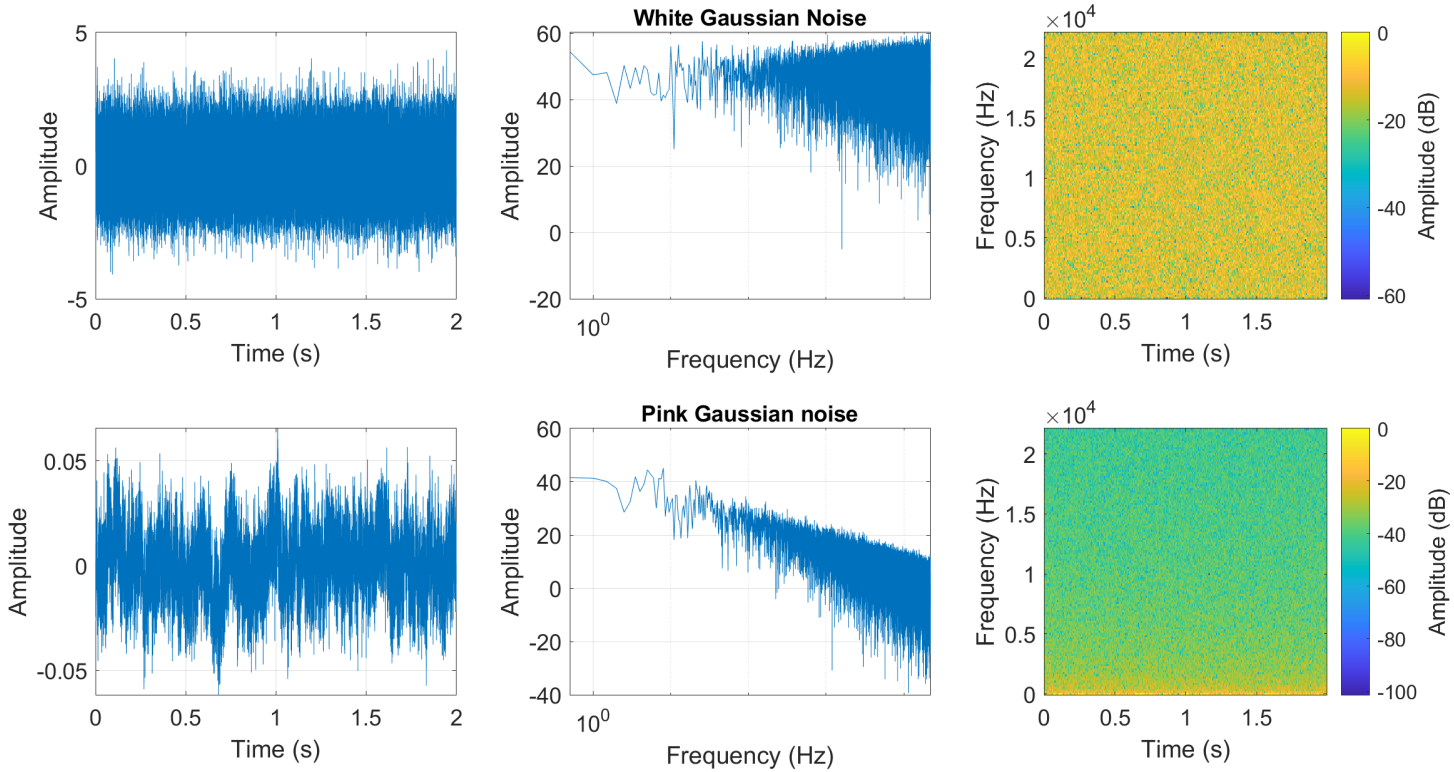
- **Gaussian**: samples taken from a Gaussian distribution.
- **Uniform**: samples taken from a Uniform distribution.

Comparison of Gaussian and Uniform White noises



- Difference in the temporal domain: larger amplitude for Gaussian noise than for Uniform noise.
- Flat spectrum for both noises.
- Uniform spread of energy in the time-frequency plane for both noises.

Comparison of White and Pink Gaussian noises



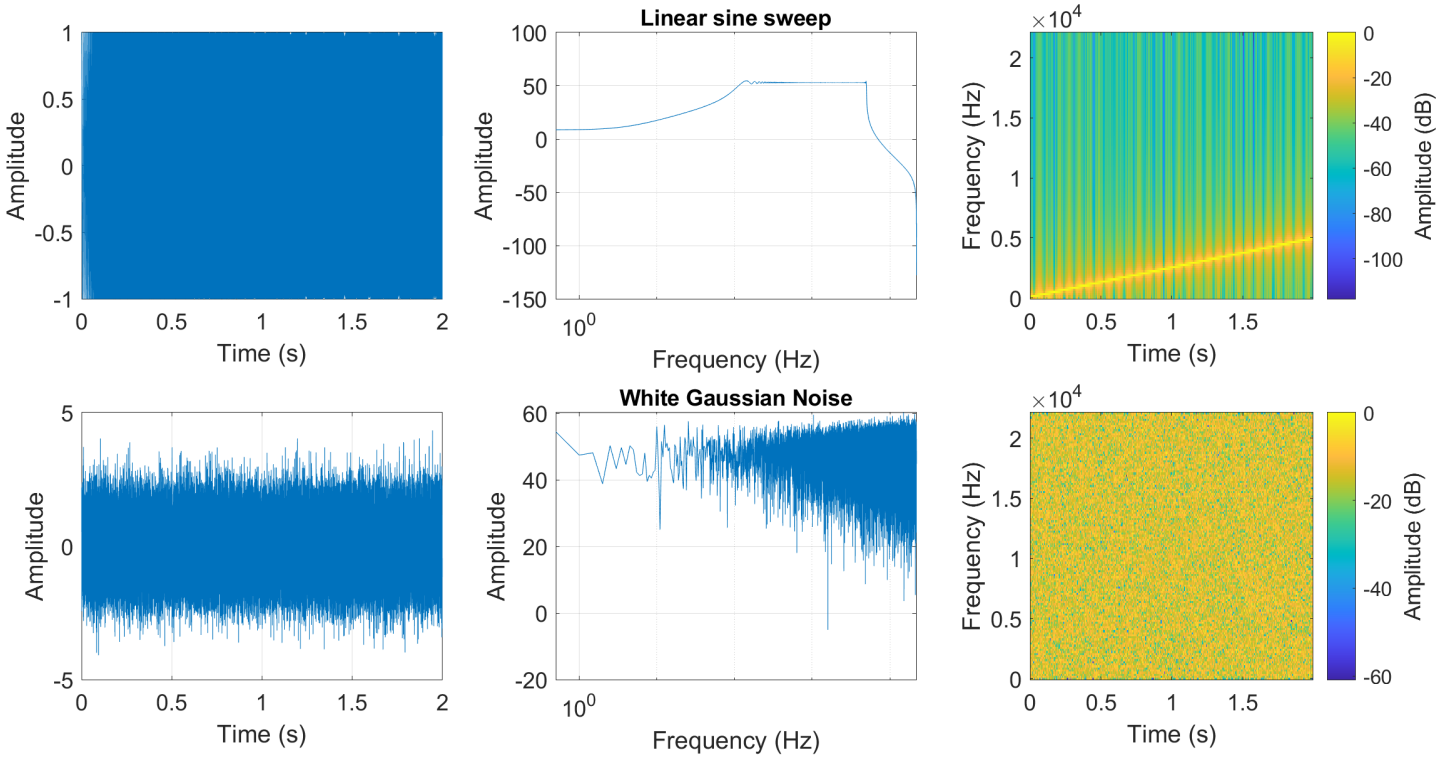
- Difference in the temporal domain: larger amplitude for white noise than for pink noise.
- Flat spectrum for white noise and decreasing sprectrum for the pink one.
- Uniform spread of energy in the time-frequency plane for white noise and not for the pink one.

Noises in practice

- MATLAB/OCTAVE white noises:
  - **Y = rand(N,M)**: returns an N-by-N matrix containing random values drawn from the standard uniform distribution on the open interval(0,1).
  - **Y = randn(N,M)** returns an N-by-N matrix containing random values drawn from the standard normal distribution with zero mean and unitary standard deviation.
- Noise selection: Choosing between a white/pink and uniform/gaussian is no is not straightforward. It depends on:
  - **The background noise spectrum:** if it is flat prefer the white noise and if its spectrum is in $1/f$, prefer the pink one. As stated previously both cases are common in practice.
  - **The tested system dynamics:** if the signal needs to be normalized prefer the uniform distribution and if not, both choices are possible.

Note: See later for pink noise generation.

Conclusions regarding standard signals synthesis



Several standard signals are available for experimentation:

- **Background noise** knowledge is required to select the optimal signal.
- **Upcoming signal processing** can greatly influence signal choice, and specifically help in choosing between deterministic of random signals.

[Homeworks #1] Practicing the standard signals

The power of a discrete time signal $x[n]$ is defined as:

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} x[n]^2$$

Given a useful signal $x[n]$ and a noise signal $s[n]$, the signal to noise ratio (SNR) is defined in dB as: $SNR = 20 \log_{10} (P_x/Ps)$.

1. Check in discrete time the continuous time Parseval equality $P_x = \int_{-\infty}^{+\infty} x(t)^2 dt = \int_{-\infty}^{+\infty} X(f)^2 df$ which allows to compute the signal power also in the frequency domain.
2. Generate an exponential sine sweep having a maximum amplitude of 5 V. Compute its power and plot its spectrogram to illustrate the exponential evolution of frequency with time.
3. Generate a Gaussian white noise of power equal to half the power of the previous exponential sine sweep. Plot its histogram to check the fact that samples are drawn from a Gaussian distribution.
4. Generate a signal that is the sum of the previous exponential sine sweep and Gaussian white noise. Assuming the exponential sine sweep is the useful signal, what is the SNR of this composite signal? Plot its spectrogram and comment it.
5. Program a function that generate a composite signal like the previous one and having a given SNR in dB. Plot the spectrograms of signals with various SNR and comment.

Robotic context

Standard digital signals synthesis

Resampling

Time alignement

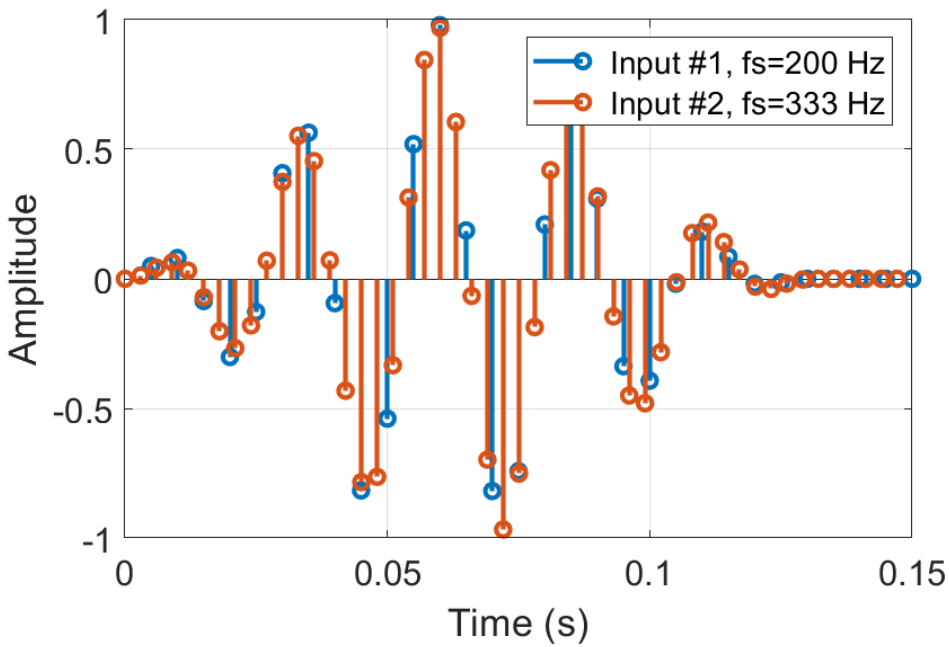
Continuous-time filtering of signals

Discrete-time filtering of signals

Digital filters synthesis

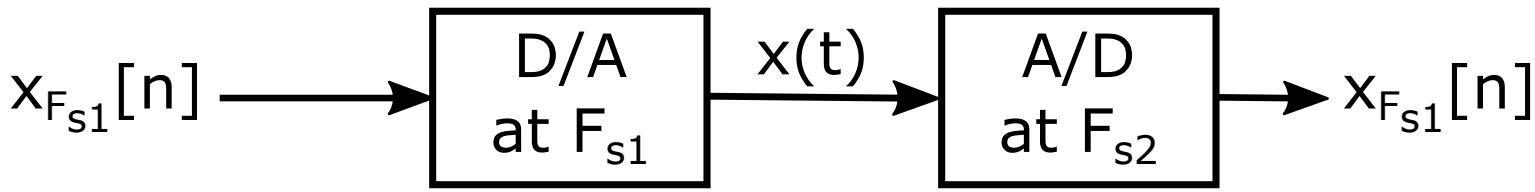
Discrete time domain manipulation

Problem #1: Experimental signals are not acquired with the same acquisition frequency and thus sampling frequencies need to be uniformed to perform further processing steps.



How is it possible to synchronize them, *i.e.* to resample them at the same sampling frequency?

Naive approach to the resampling issue

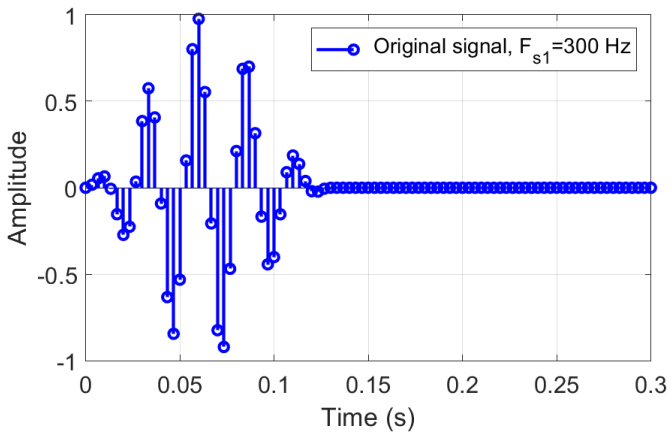


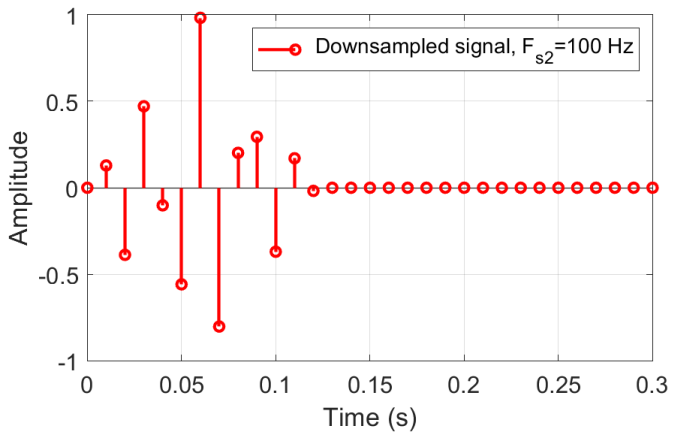
The easiest way would be to convert back signals to the analogic domain and then to resample them, but:

- Analogic signal are prone to noise.
- Analogic filters are not perfect.
- An additional quantification step will introduce additional noise.

Can we perform this operation by staying in the digital domain?

Downsampling by an integer factor





Objective:

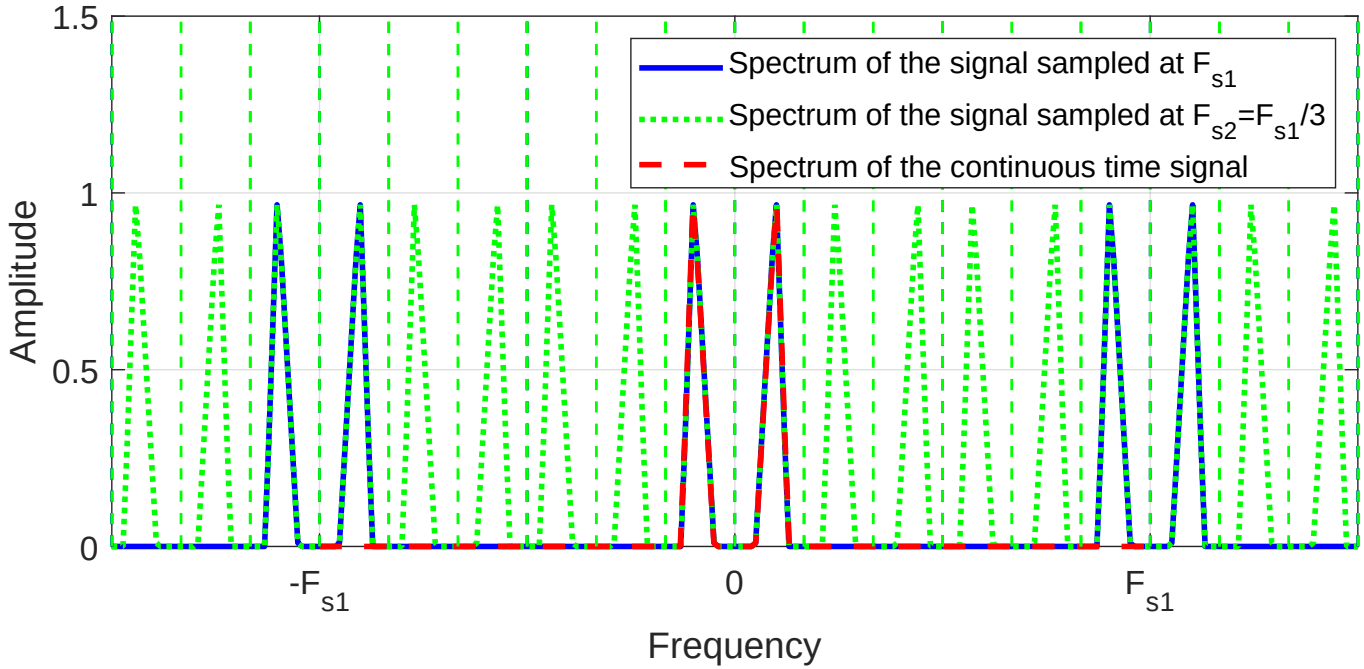
- Starting from a discrete-time signal $x_{F_{s_1}}[n] = x(nT_{s_1})$
- Build a discrete-time signal $x_{F_{s_2}}[n] = x(nT_{s_2})$
- With $F_{s_1}/F_{s_2} = T_{s_2}/T_{s_1} = L \in \mathbb{N}$.

Solution:

- Take one sample out of L in $x_{F_{s_1}}$ to build $x_{F_{s_2}}$.
- Then $x_{F_{s_2}}[n] = x_{F_{s_1}}[Ln]$

Note: Consequences of subsampling are not seen in discrete-time.

Downsampling effect on spectrum



Differences between the original and the downsampled signals are:

- The downsampled signal has a spectrum with a periodicity $F_{S_2} = F_{S_1}/L$.
- The maximum frequency in the downsampled signal shall not exceed $F_{S_2}/2 = F_{S_1}/(2L)$.
- The downsampled signal contains L times less samples than the original one.

Downsampling in practice: Decimation

In order to avoid any unintentional aliasing effect due to downsampling, a **numerical anti-aliasing filter at $F_{s_2}/2$ is systematically introduced**.

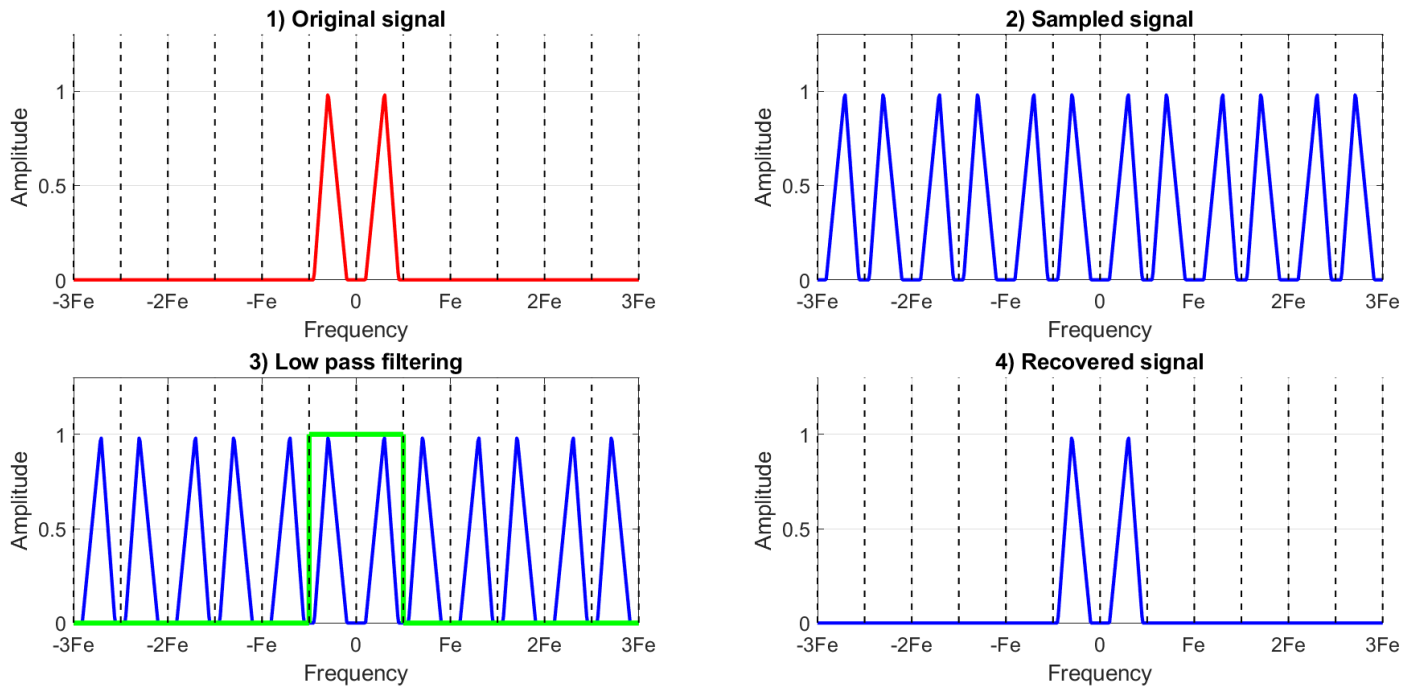
Downsampling at F_{s_2} + Anti-aliasing filter at $\frac{F_{s_2}}{2}$ = Decimation

In MATLAB/Octave, the following functions are available:

- $x_{us} = x[1 : L : end]$**:** this command takes one sample every L.
- **downsample(x,L):** this function downsamples input signal X by keeping every L-th sample starting with the first.
- **decimate(x,L)**: thus function resamples the sequence in vector x at $1/L$ times the original sample rate. By default, decimate filters the data with an 8th order Chebyshev Type I lowpass filter withcutoff frequency $0.8 \times (Fs/2)/L$, before resampling.

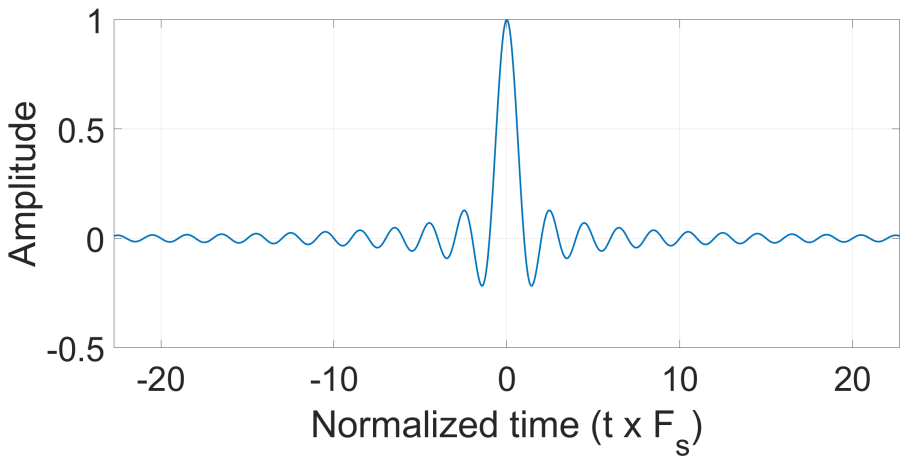
Note: Keep in mind that a new time vector needs to be defined!

Upsampling: reconstruction of a band-limited continuous time signal from the sampled one



- If $f_{max} < F_s/2$ samples are enough to recover the whole continuous time signal.
- **Upsampling can be achieved through band pass filtering.**

Upsampling: the ideal reconstruction filter impulse response



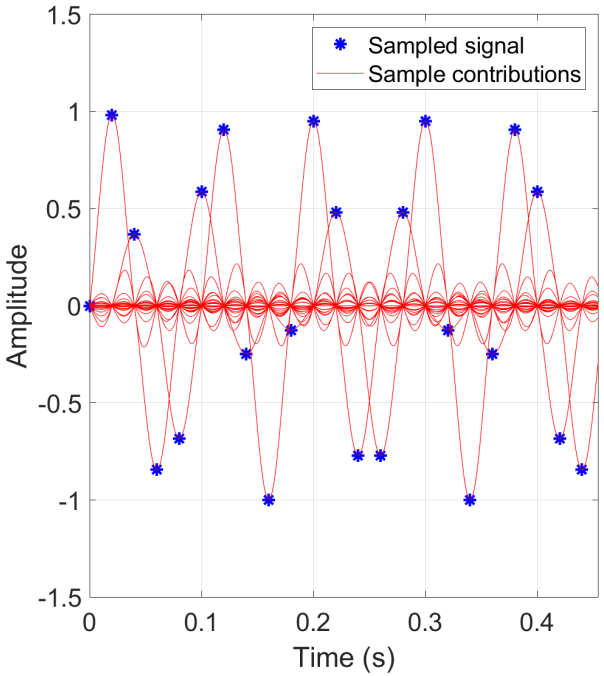
The ideal reconstruction filter is given by:

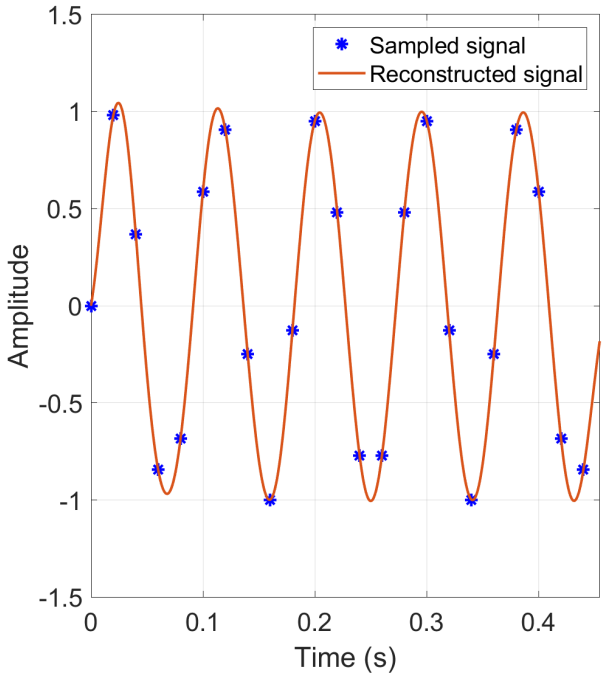
$$h(t) = \frac{sin(\pi t F_s)}{\pi t}$$

Usefull properties:

- $h(0) = 1$: It takes the **sample value at the corresponding sample time**.
- $\forall n \in \mathbb{N}^* \quad h(n) = 0$: **It does not disturb at other sample times**.
- It **exactly interpolates** between samples.

Upsampling: Reconstruction of a band-limited continuous time signal from the sampled one





- **Sampled signal:**

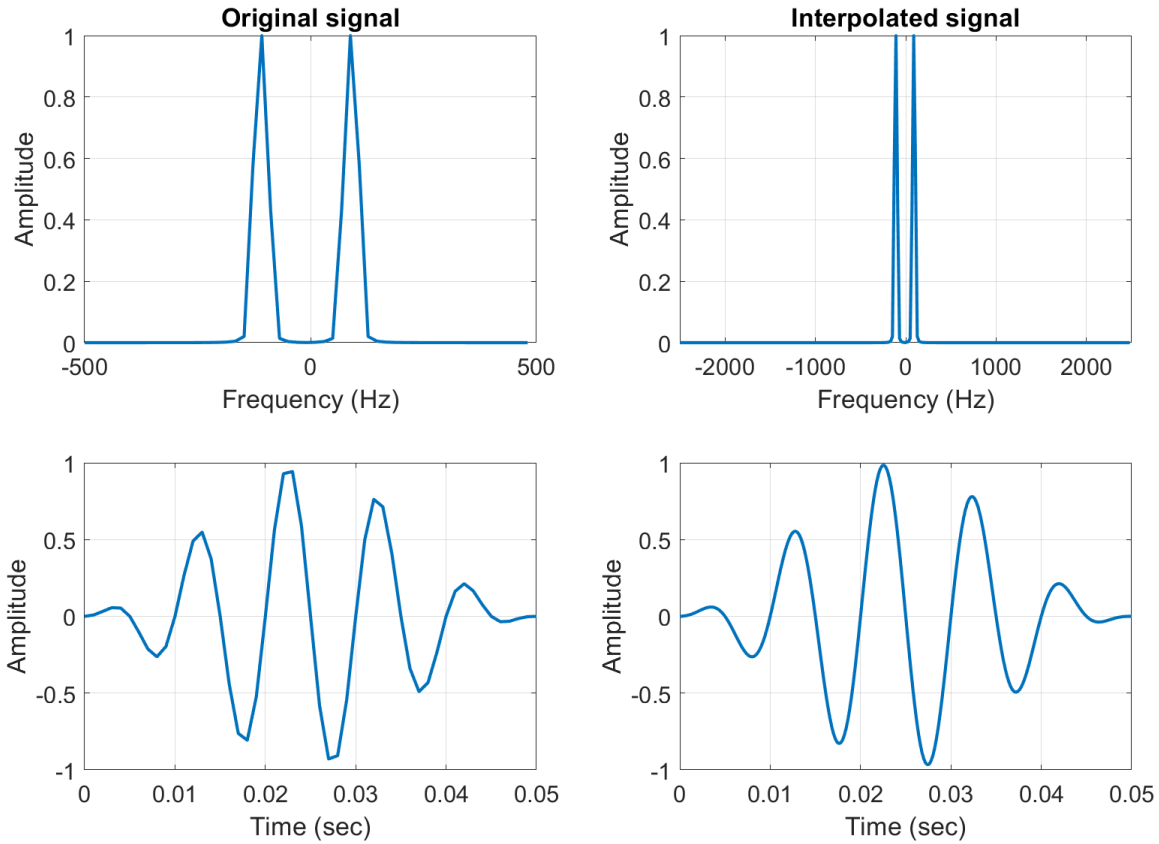
$$e^*(t) = e(t) \sum_{n=-\infty}^{+\infty} \delta(t - nT_e) = \sum_{n=-\infty}^{+\infty} e[n]\delta(t - nT_e)$$

- **Reconstructed signal:**

$$e^r(t) = (h * e^*)(t) = \sum_{n=-\infty}^{+\infty} e[n]h(t - nT_e)$$

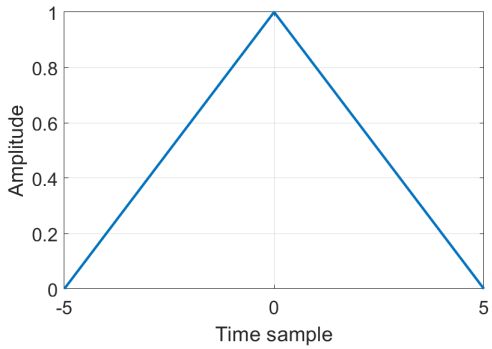
Upsampling: frequency domain interpolation

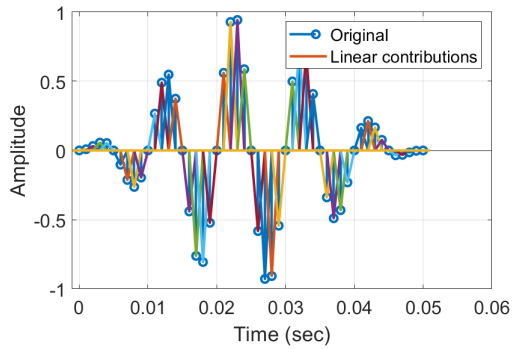
Upsampling is nothing else that extending the spectrum with zeros in the discrete frequency domain.

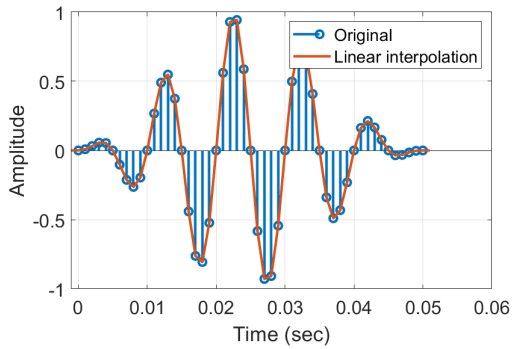


Note: Oversampling can be achieved also in the frequency domain.

Upsampling: Is linear interpolation a good idea?



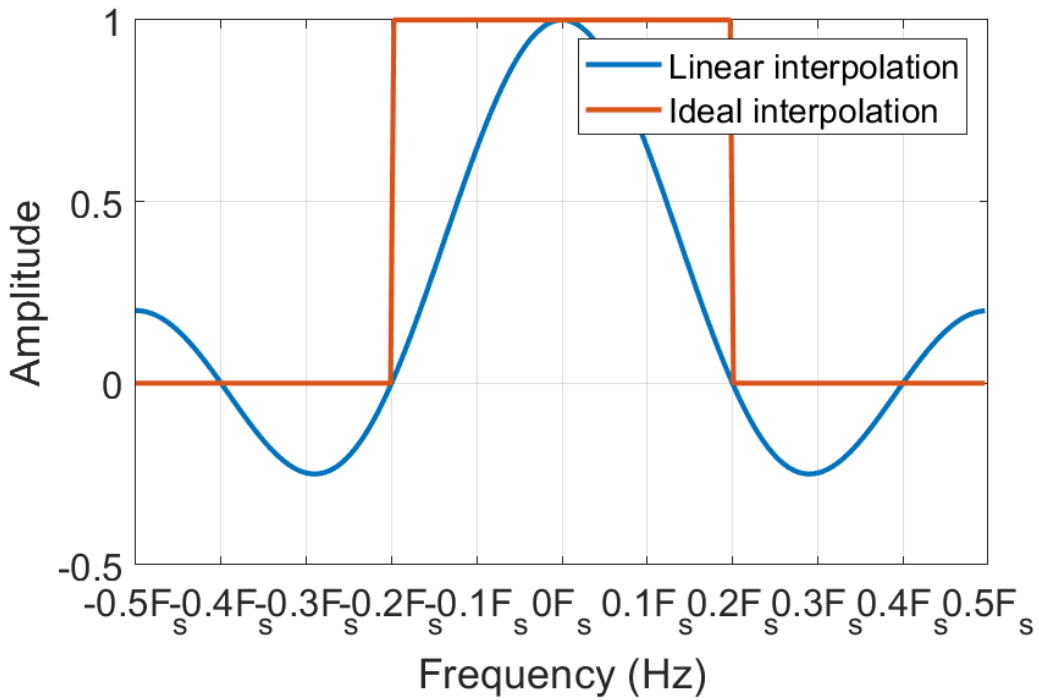




Linear interpolation (oversampling of $L = 5$):

- The linear interpolator impulse response is defined as $h[n] = 1 - |n|/L$ (different from the ideal one).
- The linearly interpolated signal globally resembles the original one in the time domain.
- **But is it the same in the frequency domain?**

Upsampling: linear interpolation spectrum



- Linear interpolation is not a band pass filter.
- Aliasing can occur with linear interpolation.
- Linear interpolation differs from ideal interpolation.
- This can be generalized to any interpolation!

Avoid interpolation!

But what about resampling for non-integers factors?

- We know how to oversample and undersample for integer factors.
- Any non integer number K can be written in practice as the ratio of two integers $K = P/L$.
- **Resampling by a factor K just consists in oversampling by P and then undersampling by L.**

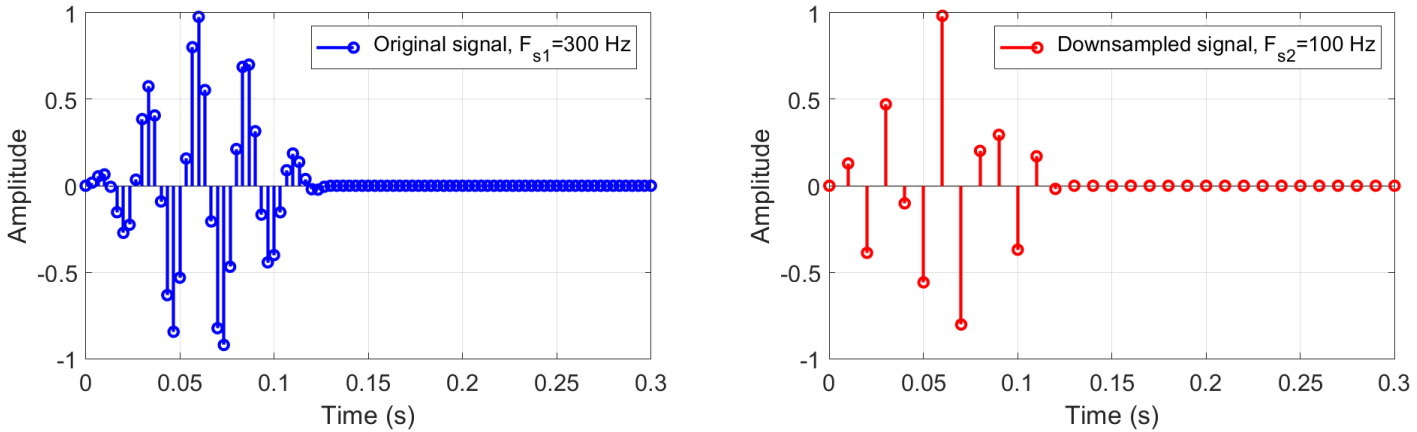
Example: Resampling by a factor 1.2 can be achieved by oversampling by a factor 6 and then undersampling by a factor 5.

Oversampling in practice

Several functions are available in MATLAB/Octave for oversampling:

- **upsample**(X,L): upsamples input signal X by inserting $L-1$ zeros between input samples. **[to avoid!]**
- **interp**: resamples data at a higher rate using lowpass interpolation (ideal interpolation).
- **interp1**: interpolation using linear or other approaches. **[to avoid!]**
- **interft**: interpolation in the Fourier transform domain (frequency interpolation).
- **resample**: resamples the values, X, of a uniformly sampled signal at P/L times the original sample rate using anti-aliasing filter (non-integer factor resampling).

Conclusions regarding resampling



Resampling dicrete-time experimental signals can be achieved if necessary.

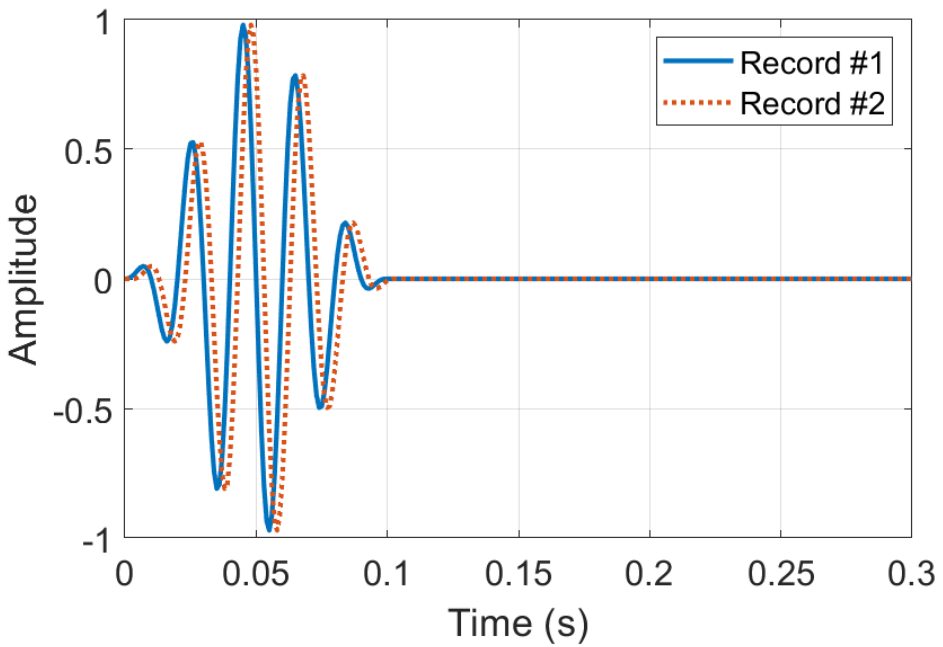
- Downsampling and upsampling can be performed in the **digital domain**.
- Resampling by **integer or non-integer** delays can be achieved.
- Pay attention to **spectral aliasing before downsampling**.
- Avoid interpolation and **prefer low pass or Fourier based interpolation**.

[Homeworks #2] Practicing resampling

1. Generate a signal made of 5 sinusoidal periods at the frequency $f_0 = 100$ Hz, sampled at 1 kHz and of duration 0.3 seconds.
2. Plot the discrete spectrum of this signal. What is the maximum frequency contained in this signal? Deduce the maximal downsampling factor that can be applied to this signal.
3. Apply various downsampling factors to this signal, below and above the maximum one. Illustrate aliasing effects that can occur due to downsampling.
4. Starting from the downsampled signal, try to recover the original one using low-pass interpolation, frequency domain interpolation and linear interpolation. Compare the signals in the time and frequency domains and conclude.
5. Starting from the downsampled signal, generate a new version of this signal that is sampled at 1.2 times the original one. Plot both signals in the time and frequency domains and conclude.

Time alignment

Problem #2 : Experimental signals are not acquired simultaneously (delay in some acquisition setup, different experiments, ...) and thus need to be synchronized to perform further processing steps.



How can we synchronise experimental signals?

How to quantify a delay between signal?

In order to quantify a delay between signals $x_1(t) \in \mathbb{R}$ and $x_2(t) \in \mathbb{R}$, they need to be *"compared in the time domain"* using **continuous correlation**.

Continuous correlation between $x_1(t)$ and $x_2(t)$

$$\boxed{\gamma_{x_1,x_2}(t) = \int_{-\infty}^{+\infty} x_1(\tau)x_2(\tau - t)d\tau}$$

Useful properties:

- $\Gamma_{x_1,x_2}(f) = \int_{-\infty}^{+\infty} \gamma_{x_1,x_2}(t)e^{-j2\pi ft}dt = X_1(f)\overline{X_2(f)}$
- $\forall t \in \mathbb{R} \quad |\gamma_{x,x}(t)| \leq |\gamma_{x,x}(0)|$

Why is it useful?

Training: Demonstrating $\Gamma_{x_1,x_2}(f) = X_1(f)\overline{X_2(f)}$

Training: Showing that $\forall t \in \mathbb{R} \quad |\gamma_{x,x}(t)| \leq |\gamma_{x,x}(0)|$

Note: Cauchy-Schwartz inequality

$$\left(\int_{-\infty}^{+\infty} x_1(t)x_2(t)dt\right)^2 \leq \left(\int_{-\infty}^{+\infty} x_1^2(t)dt\right)\left(\int_{-\infty}^{+\infty} x_2^2(t)dt\right)$$

Application to delay quantification

Let's consider that:

- $x_1(t) = x(t)$
- $x_2(t) = x(t - T)$ (a T-delayed version of $x_1(t)$)

What is the maximum of $\gamma_{x_1,x_2}(t)$?

$$\begin{aligned}\gamma_{x_1,x_2}(t) &= \int_{-\infty}^{+\infty} x_1(\tau)x_2(\tau - t)d\tau \\ &= \int_{-\infty}^{+\infty} x(\tau)x(\tau - t + T)d\tau \\ &= \gamma_{x,x}(t - T)\end{aligned}$$

$\gamma_{x_1,x_2}(t)$ is maximum at $t = T$!

Note: This will be extremely useful for delay quantification.

Discrete time correlation

Continuous time correlation can not be used in practice as:

- It is based on the knowledge of $x_1(t)$ and $x_2(t)$ over $\mathbb{R}$.
- $x_1(t)$ and $x_2(t)$ are assumed to be continuous time functions.
- $\gamma_{x_1,x_2}(t)$ is itself a continuous time function.

Discrete-time correlation $\gamma_{x_1,x_2}[n]$ between the discrete-time signals $\{x_1[n]\}_{n\in[0,N-1]}$ and $\{x_2[n]\}_{n\in[0,M-1]}$ is defined as:

$$\forall n \in [-M+1, N-1] \quad \gamma_{x_1,x_2}[n] = \sum_{m=0}^{N-1} x_1[m]x_2[m-n]$$

- The correlation sequence contains $M+N-1$ samples when correlating signals of lengths N and M.
- The same properties than for the continuous-time case hold for the discrete-time case.

Understanding discrete time correlation indexes

$$\forall n \in [-M+1, N-1] \quad \gamma_{x_1,x_2}[n] = \sum_{m=0}^{N-1} x_1[m]x_2[m-n]$$

Keep in mind that the discrete time sequences have **a finite number of sample**: $\{x_1[n]\}_{n\in[0,N-1]}$ and $\{x_2[n]\}_{n\in[0,M-1]}$.
Then:

- **Case #1:** $n < -M+1$:
  - $\forall m \in [0, N-1] \quad m-n > M+1$
  - Thus $\forall m \in [0, N-1] \quad x_2[m-n]$ does not exists.
- **Case #2:** $n > N-1$:
  - $\forall m \in [0, N-1] \quad m-n < 0$
  - Thus $\forall m \in [0, N-1] \quad x_2[m-n]$ does not exists.

Note: A **special time vector** will need to be built to visualize correlation functions.

Computing correlation in the t or f domains?

- **In continuous-time:**
  - The second property sates that:

$$\Gamma_{x_1,x_2}(f) = X_1(f)\overline{X_2(f)}$$

  - Thus is $\mathcal{F}$ denotes the continuous Fourier transform:

$$\gamma_{x_1,x_2}(t) = \mathcal{F}^{-1}\left[\mathcal{F}[x_1]\overline{\mathcal{F}[x_2]}\right](t)$$

- **In discrete-time:**
  - The second property states that:

$$\Gamma_{x_1,x_2}[k] = X_1[k]\overline{X_2[k]}$$

  - Thus is $\mathcal{F}_d$ denotes the discrete Fourier transform:

$$\gamma_{x_1,x_2}[n] = \mathcal{F}_d^{-1}\left[\mathcal{F}_d[x_1]\overline{\mathcal{F}_d[x_2]}\right][n]$$

Taking advantages of the computational efficiency of FFT, it can be advantageous to compute discrete-time correlation in the frequency domain!

Correlation in practice in MATLAB/OCTAVE

$$[C,\textbf{LAGS}] = \textbf{XCORR}(A,B)$$

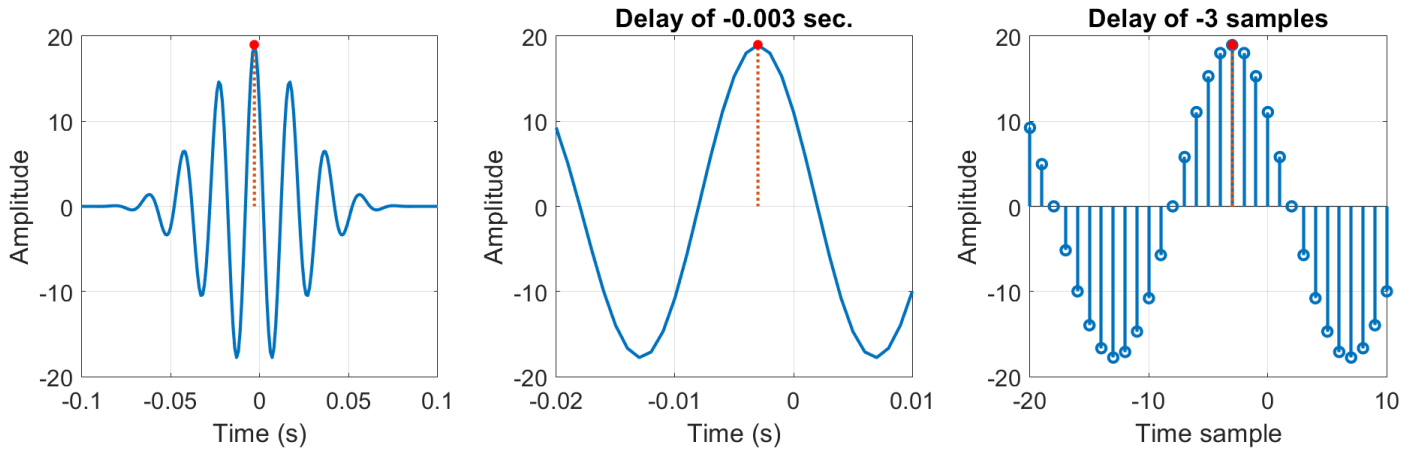
- A and B are length M vectors ($M > 1$)
- It returns the length $2M-1$ cross-correlation sequence C.
- If A and B are of different length, the shortest one is zero-padded.
- It also returns a vector of lag indexes (LAGS) that can be used to build the time vector associated with the sequence C by dividing it by F_s.

Note: Beware, statistical correlation function also exists in MATLAB/OCTAVE and shall not be confused with *XCORR*:

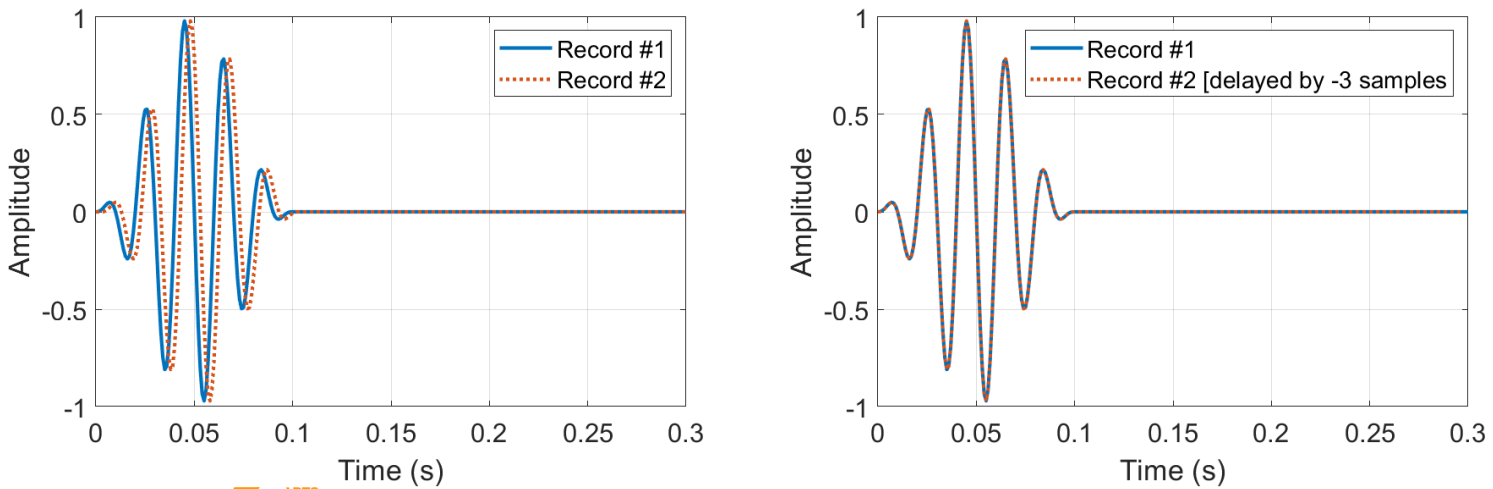
- **CORR(X)** returns a P-by-P matrix containing the pairwise linear correlation coefficient between each pair of columns in the N-by-P matrix X.
- **CORRCOEFF(X,Y)** calculates a matrix R of correlation coefficients between vectors X and Y.

Compensating integer number of samples delays

- Correlation analysis: identifying the number of samples corresponding with the delay



- Delay compensation: re-indexing samples



Quantifying non-integer delay

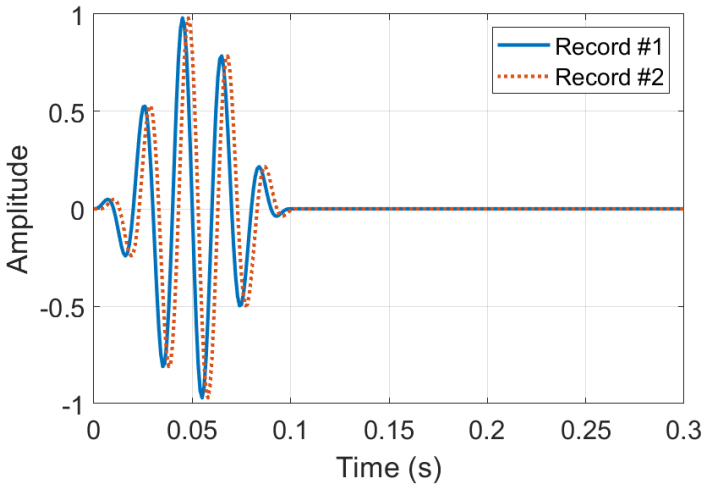
But what if the delay corresponds to a non-integer number of samples?

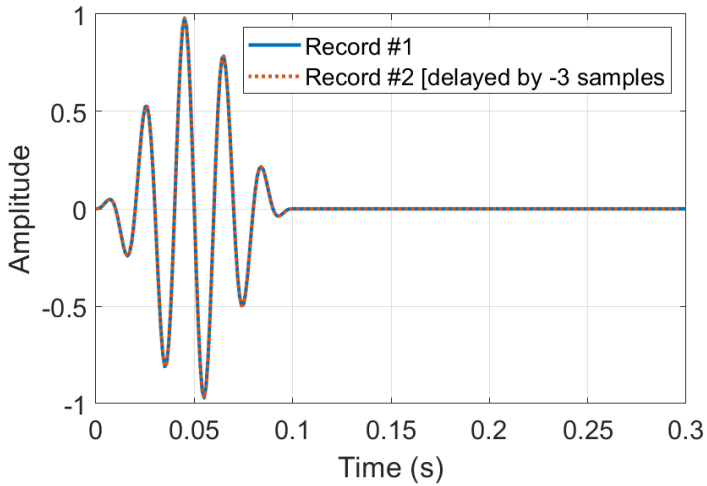
Note: This can happen when repeating several time the same experiments using a trigger signal. The uncertainty with which the experiment is recorded can vary within 0 and 1 sample.

The procedure to follow is:

1. **Upsample** signals $x_1[n]$ and $x_2[n]$ by a factor K.
2. Perform **correlation analysis** and estimate the number of samples corresponding to the delay at the sampling frequency KF_s.
3. **Compensate** the delayed signal at KF_s.
4. **Downsample** the corrected signal to go back to the original sampling frequency F_s.

Conclusions regarding time alignement





Time misalignement between experimental signals can be corrected if necessary.

- **Correlation** analysis between experimental signals allows for delay estimation.
- Delays corresponding to **integer and non-integer** number of samples can be compensated, eventually using resampling stategies.

[Homeworks #3] Practicing discrete-time correlation and time alignement.

1. Generate a White Gaussian Noise sequence as well as a delayed version of this sequence using a delay that corresponds to an integer number of samples.
2. Compute the discrete-time correlation of both sequences and estimate the delay in samples between both signals.
3. Compute the time vector associated with the discrete-time correlation of both sequences and estimate the delay in seconds between both signals.
4. Compensate the delay between the signals and plot them.
5. Repeat the same steps with a sinusoidal signals and comment eventual difficulties encountered.
6. Upsample the initial White Gaussian Noise sequence in order to generate a delayed version corresponding to a non-integer number of samples having the initial sampling frequency.
7. Estimate the delay between both signals and compensate it.
8. Add a independent Uniform Pink Noise to the delayed signal and repeat the procedure.
9. Illustrate the effect of the SNR on the delay estimation results.

Robotic context

Standard digital signals synthesis

Resampling

Time alignement

Continuous-time filtering of signals

Discrete-time filtering of signals

Digital filters synthesis

Why filtering?

Another common signal processing operation when dealing with experimental data is filtering. This can be necessary to:

- Remove undesirable noise from a recorded signals.
- Focus on a specific frequency part of the experimental signals.

The filtering problem can be cast as three sub-problems:

1. How do I **filter** my experimental data?
2. What are desirable **filters properties**?
3. How do I **design** a filter that suits my expectations?

Linear and time invariant (LTI) filters

The focus is put here on linear and time-invariant filters.

- **Linearity**: Given an input $x_1(t)$ producing and output $y_1(t)$, an input $x_2(t)$ producing and output $y_2(t)$, we have

$$x_1(t) + x_2(t) \rightarrow y_1(t) + y_2(t)$$

- **Time-invariance**: Given an input $x(t)$, the filter will always produce the same output $y(t)$, *i.e.* the filter output output will always be the same, whatever the instant at which the input signal is presented.
- LTI filters can be mathematically represented mathematically using **convolution**.

What is filtering in the continuous-time domain?

Filtering in the continuous-time domain consists in convolving the signal to be processed $x(t)$ with the filter impulse response $h(t)$. The filtered signal is denoted as $y(t)$.

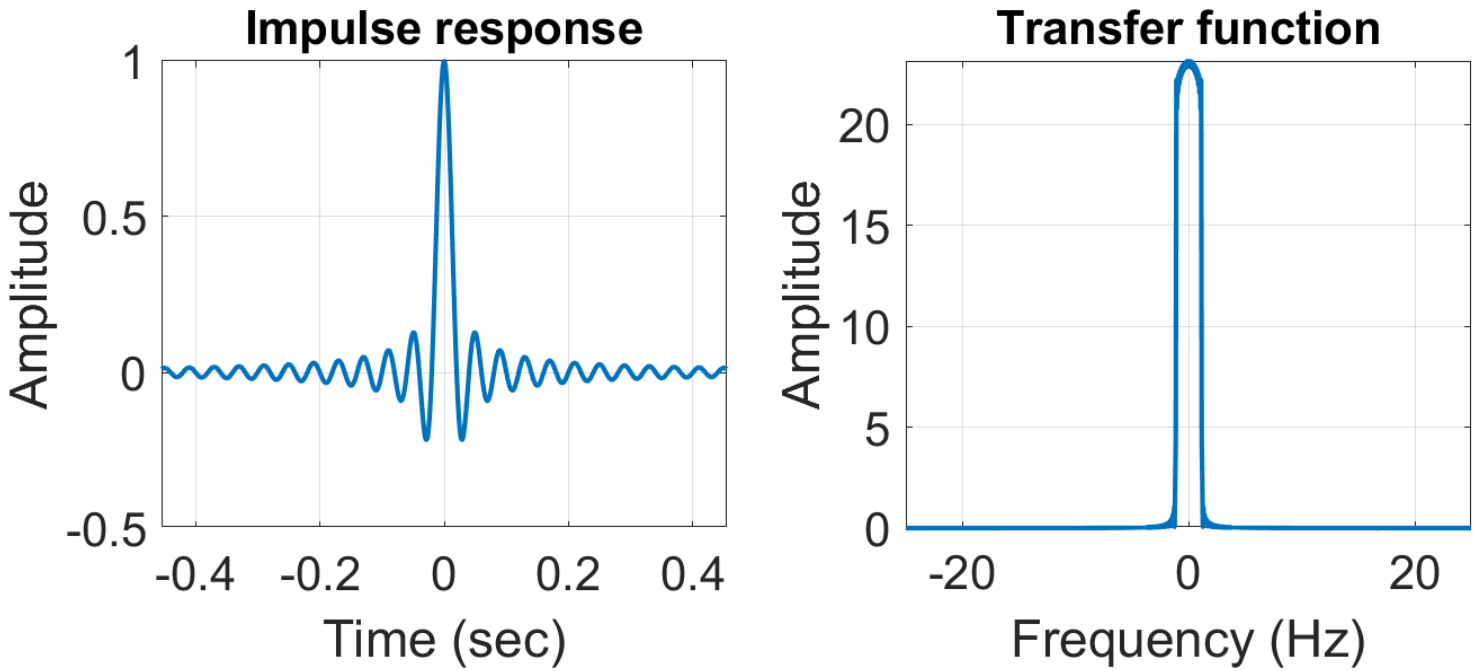
$$y(t) = (x*h)(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau = \int_{-\infty}^{+\infty} h(\tau)x(t-\tau)d\tau$$

- $*$ denotes the convolution product
- The Dirac impulse $\delta(t)$ is the neutral element of the convolution
- What is the difference with the correlation?

Defining a continuous time LTI filter

A continuous time filter is defined equivalently:

- By its **impulse response** $h(t)$.
- By its **transfer function** $H(f) = \mathcal{F}[h](f)$.



Take the example of ideal band pass filter:

$$h(t) = \frac{sin(\pi t F_s)}{\pi t} \quad \text{or} \quad H(f) = \begin{cases} 1, & \text{if } |f| < F_s/2 \\ 0, & \text{otherwise} \end{cases}$$

Filtering in the frequency domain

Demonstrate that $Y(f) = H(f)X(f)$

Filters forms in the frequency domain

Filters can be put in the frequency domain into various equivalent forms:

- As rational fractions developed or factorized using poles and zeros
- As a sum of elementary rational fractions
- As product of elementary rational fractions

$$H(p = j2\pi f) = \frac{\sum_{n=0}^{n_n} a_n p^n}{\sum_{m=0}^{n_d} b_m p^m} = \frac{\prod_{n=1}^{n_z} (p - p_n)}{\prod_{m=1}^{n_p} (p - p_m)} \quad (1)$$

$$= \sum_{n=1}^{n_f} \frac{\alpha_n}{p - p_n} = \prod_{n=1}^{n_p} \frac{\beta_n}{p - p_n} \quad (2)$$

Causality

A filter is said to be causal if its impulse response is null for *"negative"* times:

$$\forall t < 0 \quad h(t) = 0$$

- For a **causal** filter of impulse response $h(t)$, it is possible to write for an input signal $x(t)$:

$$y(t) = (h * x)(t) = \int_0^{+\infty} h(\tau)x(t-\tau)d\tau$$

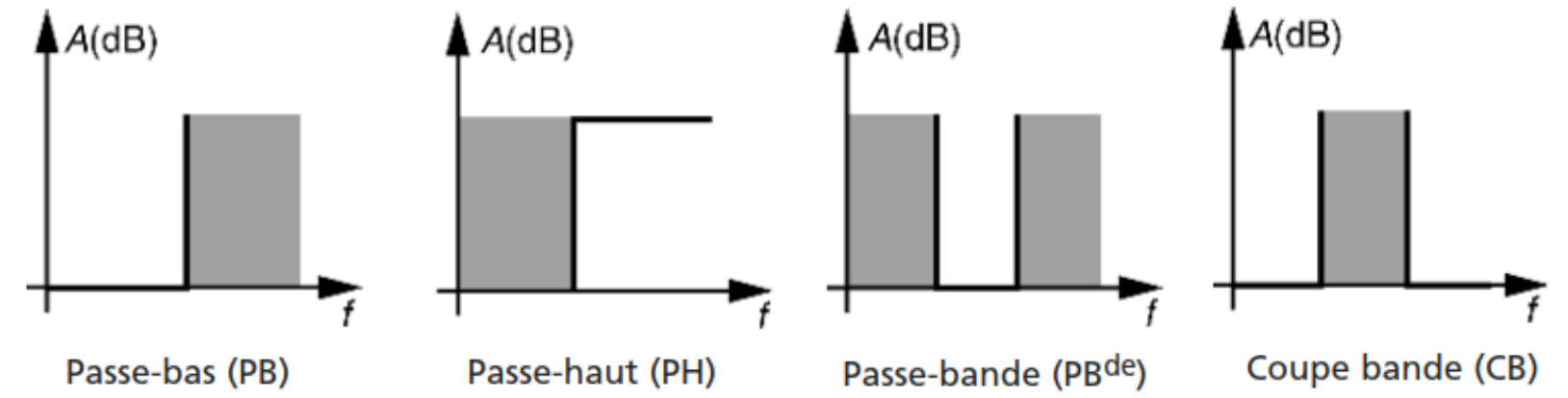
- The output signal $y(t)$ at time t thus depends only on $x(t)$ for $t \in [-\infty, t]$, *i.e.* on the past.
- For real-time, **online**, filtering, causal filters are mandatory. For **offline** processing, it is not the case.

Srict stability (BIBO)

A filter is said to be strictly stable if a bounded input $x(t)$ produces a bounded output $y(t)$.

- A necessary and sufficient conditions for strict stability is that $\int_{-\infty}^{+\infty} |h(t)|dt$ is finite.
- Strictly stable filters are mainly used in signal processing.
- Several criterion exists to assess the stability of a given filter [12, 13, 14, 15, 16]:
  - Sign of the real part of the poles
  - Routh criterion
  - ...

Ideal filters



- Filters can be **categorized** depending of the shape of their frequency response, and thus on their filtering effects.
- Usually, filters that are necessary in signal processing falls into one of these categories.
- Those ideal filters cannot be reached in practice.

Discrete time filtering

Continuous time filtering can not be used in practice as:

- It is based on the knowledge of $x(t)$ and $h(t)$ over $\mathbb{R}$.
- $x(t)$ and $h(t)$ are assumed to be continuous time functions.
- $y(t)$ is itself a continuous time function.

Discrete-time filtering $y[n]$ between the discrete-time signal $\{x[n]\}_{n\in[0,N-1]}$ and the impulse response $\{h[n]\}_{n\in[0,M-1]}$ is defined as:

$$\forall n \in [0, M+N-2] \quad y[n] = \sum_{m=0}^{N-1} x[m]h[n-m]$$

- The filtered sequence contains $M+N-1$ samples when filtering a signals of length N with an impulse response of length M.

Understanding discrete time filtered signal indexes

$$\forall n \in [0, M+N-2] \quad y[n] = \sum_{m=0}^{N-1} x[m]h[n-m]$$

Keep in mind that the discrete time sequences have **a finite number of sample**: $\{x[n]\}_{n\in[0,N-1]}$ and $\{h[n]\}_{n\in[0,M-1]}$.
Then:

- **Case #1:** $n < 0$:
  - $\forall m \in [0, N-1] \quad n - m < 0$
  - Thus $\forall m \in [0, N-1] \quad h[n-m]$ does not exists.
- **Case #2:** $n > M + N - 2$:
  - $\forall m \in [0, N-1] \quad n - m > M - 1$
  - Thus $\forall m \in [0, N-1] \quad h[n-m]$ does not exists.

Note: A **special time vector** will need again to be built to visualize filtered signals.

Filtering in the discrete frequency domain

Demonstrate that for $N = M$, we have $Y[k] = H[k]X[k]$

Towards the z-transform...

For a discrete-time filter $h[n]$ with $n \in [0, N-1]$, its DFT is given as:

$$\forall k \in [0, N-1] \quad H[k] = \sum_{n=0}^{N-1} h[n] e^{j2\pi nk/N} \tag{3}$$

$$= \sum_{n=0}^{N-1} h[n] \left(e^{j2\pi k/N}\right)^n \tag{4}$$

$$= \sum_{n=0}^{N-1} h[n] z_k^{-n} \tag{5}$$

with $z_k = e^{-j2\pi k/N}$.

z-transform and its link with the DFT

The z-transform is defined as:

$$\tilde{H}[z] = \sum_{n=0}^{N-1} h[n] z^{-n}$$

- It is a tool that is very classically used to deal with discrete-time filter.
- $\tilde{H}[z_k] = H[k]$ for $z_k = e^{-j2\pi k/N} = e^{-j2\pi f_k T_s}$ with $f_k = \frac{k}{NT_s}$.
- A multiplication by $z^{-1} = e^{-j2\pi f T_s}$ in the z-domain is equivalent to a delay of one sample in the temporal domain.

Relevant properties of discrete time filters

Discrete-time filters have properties similar to continuous-time filters:

- **Causality**: $\forall n < 0, h[n] = 0$ for a causal filter.
- **Stability**: $\sum_{n\in\mathbb{Z}} |h[n]|$ is finite for a stable filter.
- **Equivalent forms**: Filters can be put in the frequency domain into various equivalent forms:
  - As rational fractions developed or factorized using poles and zeros
  - As a sum of elementary rational fractions
  - As product of elementary rational fractions

Parametric representation of digital filters

Digital filtering operations can be represented:

- **In the discrete time domain as**

$$y[n] = \sum_{k=0}^{N_b-1} b[k]x[n-k] + \sum_{k=1}^{N_a-1} a[k]y[n-k]$$

- **Or equivalently using the z-transform as**

$$Y(z) = H(z)X(z) \quad \text{with} \quad H(z) = \frac{\sum_{k=0}^{N_b-1} b[k]z^{-k}}{\sum_{k=1}^{N_a-1} a[k]z^{-k}}$$

It follows that:

- The coefficients $\{a[k]\}_{k\in[0,N_a-1]}$ and $\{b[k]\}_{k\in[0,N_b-1]}$ entirely describes the digital filter.
- The vector $A = [a[0], a[1], \ldots, a[N_a-1]]$ and $B = [b[0], b[1], \ldots, b[N_b-1]]$ can be defined.
- With $B = [h[0], h[1], \ldots, h[M+N-2]]$ and $A = a[1] = 1$, it corresponds to the convolution formula .

Digital filters families

Digital filters can be separated in two main families:

- **FIR filters:** they correspond to Finite Impulse Response filters. For those filters $A = a[1] = 1$.
  - **Advantages**: Possible linear phase in real time.
  - **Drawbacks**: Longer filters.
- **IIR filters:** they correspond to Infinite Impulse Response filters. For those filters, at least one coefficient $a[k > 1] \neq 0$.
  - **Advantages**: Lower order (N_a and N_b) needed to meet specifications.
  - **Drawbacks**: Potential nonlinear phase (can be corrected for offline applications using *filtfilt*).

Note: Why do we call them IIR or FIR?

Discrete-imte filtering in practice using MATLAB/OCTAVE

Again, various functions are available for digital signals filtering in MATLAB/Octave:

- **conv**: this function convolves dicrete signals $x[n]$ and $h[n]$ to produce $y[n]$.
- $y =$ **filter(**B,A,x**)** filters the data in vector x with the filter described by vectors A and B (see later for A and B design) to create the filtered data y.
- $y =$ **filtfilt(**B, A, x**)** performs zero-phase forward and reverse digital filtering. It filters the data in vector x with the filter described by vectors A and B (see later for A and B design) to create the filtered data y.

Special focus on the *"filtfilt"* function

"It performs zero-phase forward and reverse digital filtering."

Definition of forward and reverse filters:

- **Forward direction:** $\forall n \in [0, M-1], h_f[n] = h[n]$ and thus $H_f[k] = H[k]$.
- **Backward direction:** $\forall n \in [0, M-1], h_b[n] = h[M-1-n]$ and thus $H_f[k] = \overline{H[k]}$.

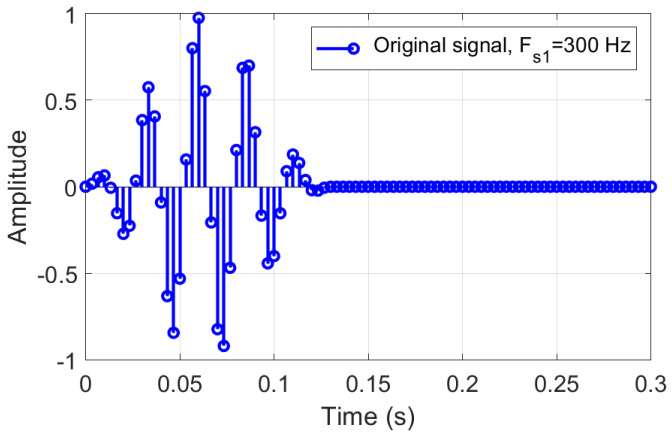
As a consequence:

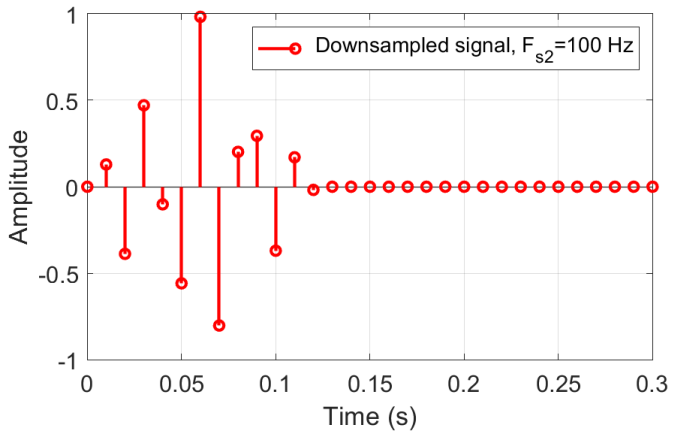
- In the dicrete-time domain: $y[n] = (h_f * h_b * x)[n]$
- In the discrete-frequency domain:

$$Y[k] = H_f[k]H_b[k]X[k] = H[k]\overline{H[k]}X[k] = |H[k]|^2 X[k]$$

There is no influence of the filtering operation on input signal phase as $|H[k]|^2 \in \mathbb{R}$.

Preliminary conclusions regarding filtering





Filtering dicrete-time signals if globally a convolution.

- Filters can be described either in the temporal domain using the **impulse response** or in the frequency domains using the **transfer function**.
- **Filtering can be efficiently achieved in the frequency domain** thanks to FFT.
- Filtering can be performed twice (in the forward and backward way) to **cancel phase effect**.

Robotic context

Standard digital signals synthesis

Resampling

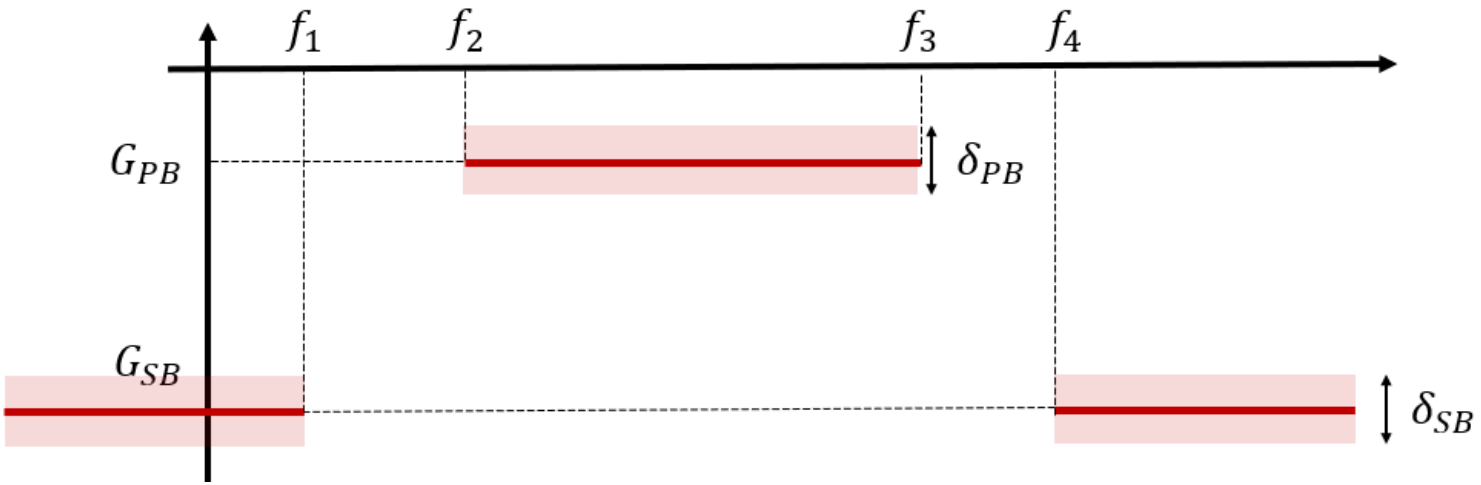
Time alignement

Continuous-time filtering of signals

Discrete-time filtering of signals

Digital filters synthesis

Design specifications



The filters to be designed may be adequately specified by the end user. Filters are usually specified in the frequency domain using:

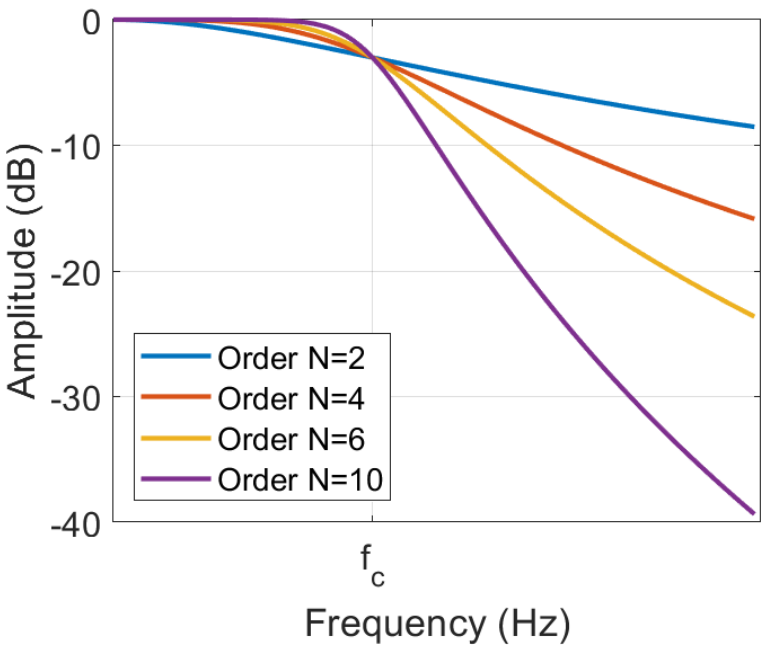
- Ideal passband gain G_{PB} and the ideal stopband gain G_{SB}.
- Transition frequencies $\{f_n\}_{n\in[1,N]}$ at which the passband or stopband gains must be achieved.
- Admissible ripples amplitudes δ_{BP} and δ_{SB} in the pass and stop bands.

Note: There exists many solutions to fulfill these specifications.

Design of continuous time filters #1: Butterworth filters

The magnitude of a N^{th} order Butterworth filter $H_B(f)$ is defined as:

$$|H_B(f)|^2 = \frac{1}{1 + (f/f_c)^{2N}}$$



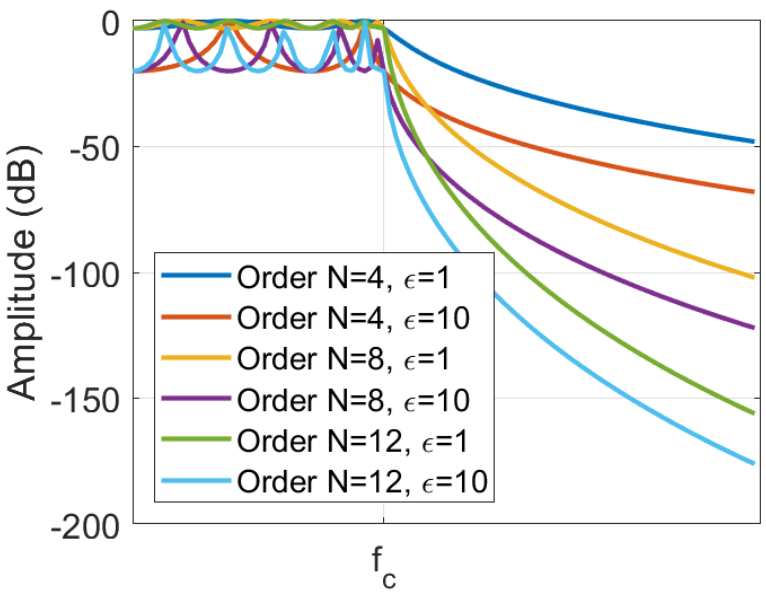
- Butterworth filters have a magnitude maximally flat in the passband.
- The bandpass gain can be adjusted as a global gain.
- The stopband gain can be adjusted using N.
- The transition frequency can be adjusted using f_c.
- Ripples cannot be adjusted.

Design of continuous time filters #2: Chebyshev filters

The magnitude of a N^{th} order Chebyshev filter $H_C(f)$ is defined as

$$|H_C(f)|^2 = \frac{1}{1+\epsilon^2 V_N^2(f/f_c)}$$

where $V_N(f/f_c)$ is a Chebyshev polynomial function.



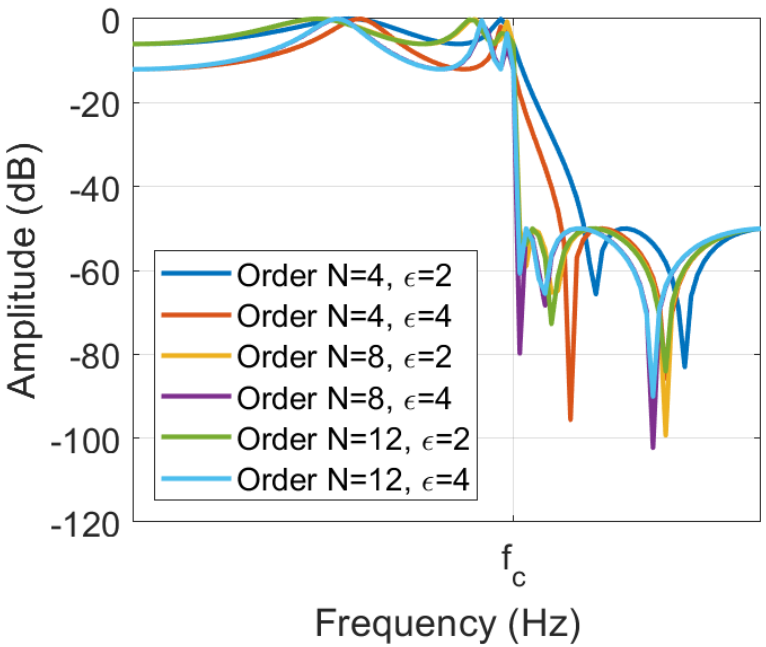
- Chebyshev filters are equiripple in the passband and monotonic in the stopband (type I) or the inverse (type II).
- The bandpass gain can be adjusted as a global gain.
- The stopband gain can be adjusted using N.
- The transition frequency can be adjusted using f_c.
- Ripples can be adjusted using ϵ.

Design of continuous time filters: Elliptic filters

The magnitude of a N^{th} order Elliptic filter $H_E(f)$ is defined as

$$|H_E(f)|^2 = \frac{1}{1+\epsilon^2 U_N^2(f/f_c)}$$

where $U_N(f/f_c)$ is a Jacobian elliptic function.



- Elliptic filters are equiripple in the passband stopband.
- The bandpass gain can be adjusted as a global gain.
- The stopband gain can be adjusted using N.
- The transition frequency can be adjusted using f_c.
- Ripples can be adjusted using ϵ.

Design of IIR filters #1: impulse invariance

Design a satisfying continuous time filter and sample its impulse response.

1. **Design a continuous time filter satisfying the specifications**
   Its expression in the Laplace domain and with a decomposition over its poles is given by p_k is $H_c(p) = \sum_{k=1}^{N} \frac{A_k}{p-p_k}$.
2. **Determine its continuous time impulse response**
   Remembering that $\mathcal{TL}[e^{p_k t}] = \frac{1}{p-p_k}$, then $h_c(t) = \sum_{k=1}^{N} A_k e^{p_k t}$.
3. **Sample its continuous time impulse response**
   Sampling at T_s: $h_d[n] = \sum_{k=1}^{N} A_k e^{p_k n T_s} = \sum_{k=1}^{N} A_k \left(e^{p_k T_s}\right)^n$.
4. **Compute the z-transform of this discrete time impulse response**
   Remembering that $\mathcal{TZ}[a^n] = \frac{1}{1-az^{-1}}$, $H_d(z) = \sum_{k=1}^{N} \frac{A_k}{1-e^{p_k T_s} z^{-1}}$
5. **Reorganize to obtain the vectors A and B**
   $H_d(z) = \sum_{k=1}^{N} \frac{A_k}{1-e^{p_k T_s} z^{-1}} = \frac{\sum_{k=0}^{N_b-1} b[k] z^{-k}}{\sum_{k=1}^{N_a-1} a[k] z^{-k}}$.

Design of IIR filters #2: bilinear transformation

Design a satisfying continuous time filter and apply the bilinear transformation.

1. **Design a continuous time filter satisfying the specifications**
   Its expression in the Laplace domain and with a decomposition over its poles is given by p_k is

$$H_c(p) = \sum_{k=1}^{N} \frac{A_k}{p - p_k}$$

.

2. **Apply the bilinear transformation** $p = \frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$

$$H_d(z) = H_c\left(\frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)\right)$$

3. **Reorganize to obtain the vectors** A **and** B

$$H_d(z) = \frac{\sum_{k=0}^{N_b-1} b[k]z^{-k}}{\sum_{k=1}^{N_a-1} a[k]z^{-k}}$$

.

FIR filter design by windowing

Main idea: Directly window the adequate impulse response.

1. Design the impulse response of a continuous time filter satisfying the specifications: $h_c(t)$
2. Given a FIR filter of length N_{FIR} truncate (or window) N_{FIR} samples from $h_c(t)$:

$$h_d[n] = \begin{cases} h_c(nT_s), & \text{if } n < N_{\text{FIR}} \\ 0, & \text{otherwise} \end{cases}$$

3. Assign the coefficients $B = [h_d[0], h_d[1], \ldots, h_d[N_{\text{FIR}}]]$ and $A = a[1] = 1$.

IIR filters design in practice in MATLAB/Octave

There are many methods for IIR filters design:

- **[B,A] = butter(**N,W_n**)**: designs an N^{th} order lowpass digital Butterworth filter and returns the filter coefficients B and A.
- **[B,A] = cheby1(**N,R,W_p**)** designs an N^{th} order lowpass digital Chebyshev filter (type I) with R decibels of peak-to-peak ripple in the passband.
- **[B,A] = cheby2(**N,R,W_p**)** designs an N^{th} order lowpass digital Chebyshev filter (type II) with R decibels of peak-to-peak ripple in the passband.
- **[B,A] = ellip(**N,R_p,R_s,W_p**)** designs an N^{th} order lowpass digital elliptic filter with R_p decibels of peak-to-peak ripple and a minimum stopband attenuation of R_s decibels.

FIR filters design in practice in MATLAB/Octave

There are many methods for FIR filters design:

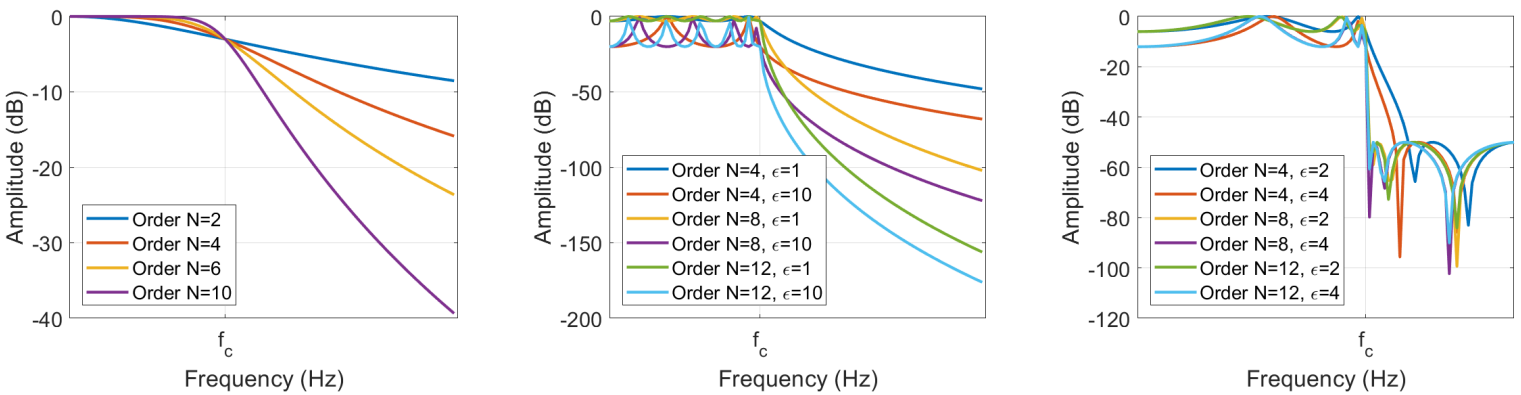
- **B = fir1(N,Wn)** designs an N'th order lowpass FIR digital filter and returns the filter coefficients in vector B.
- **B = fir2(N,F,A)** designs an Nth order linear phase FIR digital filter with the frequency response specified by vectors F and A and returns the filter coefficients in vector B.
- **B=firls(N,F,A)** returns a linear phase (real, symmetric coefficients) FIR filter which has the best approximation to the desired frequency response described by F and A in the least squares sense.
- **B=firpm(N,F,A)** returns a length N+1 linear phase (real, symmetric coefficients) FIR filter which has the best approximation to the desired frequency response described by F and A in the minimax sense.

For filters visualization:

- **[H,W] = freqz(B,A,N)** returns the N-point complex frequency response vector H and the N-point normalized frequency vector W.
- **[H,T] = impz(B,A)** computes the impulse response of a filter given numerator and denominator coefficients in vectors B and A.

Conclusion regarding digital filters design

How to design the optimal filter?



- There are many ways to design the most adequate digital filter.
- The best method is application dependent.
- It is important to understand the choices, advantages, and drawbacks associated with the various methods.

[Homeworks #4] Practicing filtering

1. Identify the type of filter which corresponds to the following difference equations. The sampling frequency is 20 kHz in all cases.
   $y[n] = x[n] + 0.5x[n-1] + 1.8cos[\pi/16]y[n-1] - 0.81y[n-2]$
   $y[n] = 0.634x[n] + 0.634x[n-1] - 0.268y[n-1]$
   $y[n] = 0.634x[n] - 0.634x[n-1] + 0.268y[n-1]$
   $y[n] = 0.1x[n] - 0.5x[n-1] + x[n-2] + 0.5y[n-1] - 0.1y[n-2]$
2. Design a digital filter which meets the following specifications using the bilinear method: sampling frequency: 1 Hz, 0 dB attenuation of the DC component, maximum attenuation of 1 dB at 0.1 Hz, minimum attenuation of 15 dB at 0.15 Hz. Consider a Butterworth and then a Chebychev model for the filter transfer function.
3. Implementing a digital filter on a digital signal processor requires the quantification of its coefficients. Thus, the actual filter characteristics are slightly modified with respect to the designed filter. Consider a 6 th order elliptic filter having a cutoff normalized frequency of $0.2F_s$, a ripple of 0.05 dB in the passband and a minimum attenuation of 60 dB in the stopband. Write a code to illustrate the effect of the quantification on 5 bits on the filter characteristics (transfer function and impulse response).
4. Consider a sum of two sinusoids, whose frequencies are 1 kHz and 1.56 kHz, sampled at 10 kHz. Extract the first sinusoid of this mixture using a lowpass FIR filter.
5. Design a 120 th order FIR highpass filter using the frequency sampling method. Consider a cutoff frequency $f_c = 6$ kHz, and a sampling frequency $F_s = 20$ kHz. Plot the filter transfer function. Compare its order to that of an IIR filter having a similar transfer function.

Bibliography

[1] W. Khalil and E. Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[2] J. Wu, J.and Wang and Z. You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5):414–419, 2010.

[3] K. R. Kozlowski. *Modelling and identification in robotics*. Springer Science & Business Media, 2012.

[4] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[5] Y. Meyer and P. Flandrin. Time-frequency/time-scale analysis. *Wavelet Analysis and its applications*, 1999.

[6] S. Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[7] W. M. Hartmann. *Signals, sound, and sensation*. Springer Science & Business Media, 2004.

[8] S. K. Mitra and Y. Kuo. *Digital signal processing: a computer-based approach*, volume 2. McGraw-Hill New York, 2006.

[9] K. Shin and J. Hammond. *Fundamentals of signal processing for sound and vibration engineers*. John Wiley & Sons, 2008.

[10] D. Havelock, S. Kuwano, and M. Vorländer. *Handbook of signal processing in acoustics*. Springer Science & Business Media, 2008.

[11] M. Weeks. *Digital signal processing using MATLAB & wavelets*. Jones & Bartlett Publishers, 2010.

[12] Étienne Tisserand, Jean-François Pautex, and Patrick Schweitzer. *Analyse et traitement des signaux-2e éd.: Méthodes et applications au son et à l'image*. Dunod, 2009.

[13] Francis Cottet. *Traitement des signaux et acquisition de données: cours et exercices corrigés*. Dunod, 2020.

[14] Frédéric De Coulon. *Théorie et traitement des signaux*, volume 6. PPUR Presses polytechniques, 1998.

[15] Maurice Bellanger. *Traitement numérique du signal*. Ed. Techniques Ingénieur, 1981.

[16] Messaoud Benidir. *Théorie et traitement du signal: Cours et exercices corrigés. Méthodes de base pour l'analyse et le traitement du signal*. Dunod, 2004.