# IFT 4031/7031,
# Machine Learning for Signal Processing
# **Week3: Signal Processing Primer**

Cem Subakan

# Admin

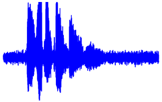
- How were the first and second labs?
  - Comment était le labs 1, 2?
- The next lab is on friday!
  - On aura le troisième lab en vendredi.
- How are the project proposals coming along? The proposals are due mid-october.
  - Comment va la réflection sur les projets? Les propositions de projets sont dûs mi-octobre!

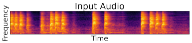
# Today

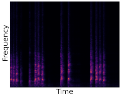
- Signals/Time series, what are they?
  - Qu'est-ce qu'on comprend quand on dit 'signaux/séries temporelles' ?

# Today

- Signals/Time series, what are they?
  - Qu'est-ce qu'on comprend quand on dit 'signaux/séries temporelles' ?
- How do we represent signals?
  - Comment est-ce qu'on répresente les signaux?
    - Time Domain / Frequency Domain / Time+Frequency Domain

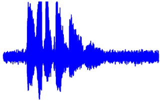


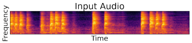


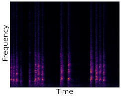



# Today

- Signals/Time series, what are they?
  - Qu'est-ce qu'on comprend quand on dit 'signaux/séries temporelles' ?
- How do we represent signals?
  - Comment est-ce qu'on répresente les signaux?
    - Time Domain / Frequency Domain / Time+Frequency Domain



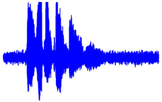


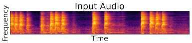



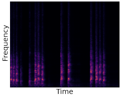

- The Fourier Transform
  - The Short-Time Fourier Transform

# Today

- Signals/Time series, what are they?
  - Qu'est-ce qu'on comprend quand on dit 'signaux/séries temporelles' ?
- How do we represent signals?
  - Comment est-ce qu'on répresente les signaux?
    - Time Domain / Frequency Domain / Time+Frequency Domain



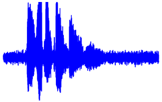


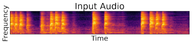



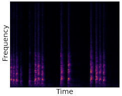

- The Fourier Transform
  - The Short-Time Fourier Transform
- Filtering / Convolution
  - Convolution

# Today

- Signals/Time series, what are they?
  - Qu'est-ce qu'on comprend quand on dit 'signaux/séries temporelles' ?
- How do we represent signals?
  - Comment est-ce qu'on répresente les signaux?
    - Time Domain / Frequency Domain / Time+Frequency Domain



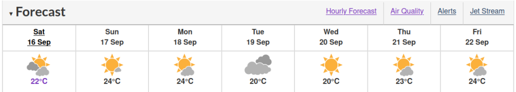


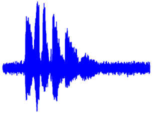




- The Fourier Transform
  - The Short-Time Fourier Transform
- Filtering / Convolution
  - Convolution
- Sampling
  - Échantillonnage

# Signals

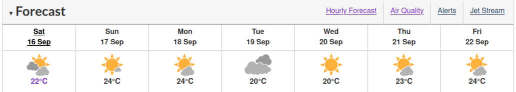
- A dry definition: A signal is an ordered collection of numbers
  - Une définition sec: Un signal est une collection des chiffres en ordre.

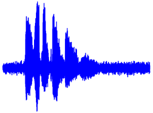



# Signals

- A dry definition: A signal is an ordered collection of numbers
  - Une définition sec: Un signal est une collection des chiffres en ordre.







# Table of Contents

# Representing Signals/Time Series

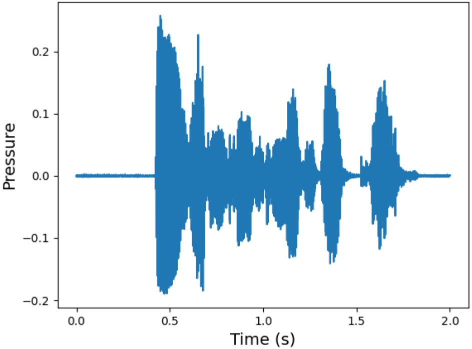
- There are different ways of representing time series/signals.
  - Il existe plusieurs façons de répresenter les signaux.
- Each representation type has its pros / cons.
  - Chaque type de représentation a leurs avantages / desavantages.

# Representing Signals/Time Series

- There are different ways of representing time series/signals.
  - Il existe plusieurs façons de répresenter les signaux.
- Each representation type has its pros / cons.
  - Chaque type de représentation a leurs avantages / desavantages.
- Some typical options: Time Domain, Frequency Domain, Time+Frequency Domain

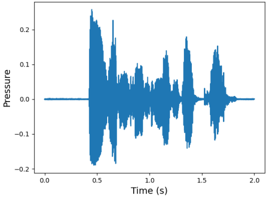
## Sound as Signal

- We will start with sounds as an example, but any time series would do.
  - On commencera avec les sons comme un exemple, mais n'importe quel time series serait ok.
- Let's `listen`.



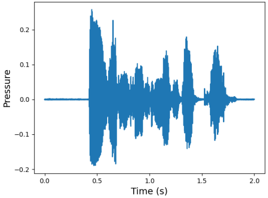
# The time domain

- It's kinda difficult what to take out of this by looking at it.
  - Je sais pas quoi faire en regardant ça.



# The time domain

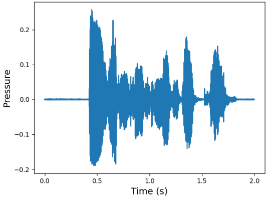
- It's kinda difficult what to take out of this by looking at it.
  - Je sais pas quoi faire en regardant ça.



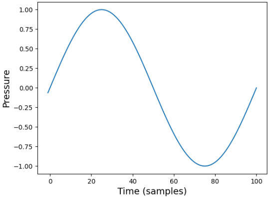

- It's possible however to express this in terms of simpler waveforms.
  - C'est possible par contre de l'exprimer ce waveform en terme des waveform plus simples.

# The time domain

- It's kinda difficult what to take out of this by looking at it.
  - Je sais pas quoi faire en regardant ça.

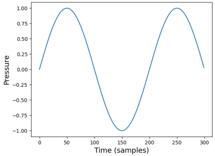



- It's possible however to express this in terms of simpler waveforms.
  - C'est possible par contre de l'exprimer ce waveform en terme des waveform plus simples.
- Sinusoids!! (They are special)

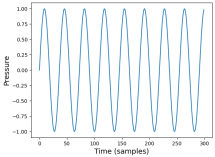


# Sinusoids

- 80 Hz, `listen`.




- 440 Hz, `listen`. This is how we tune our instruments.
  - On tune nos instruments à La 440Hz.

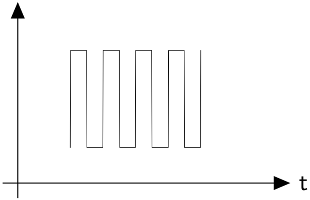



- 880 Hz, `listen`
- 1760 Hz, `listen`
- 15000 Hz, `are you able to hear this at all?`
  - Pouvez-vous entendre ceci?

# Table of Contents

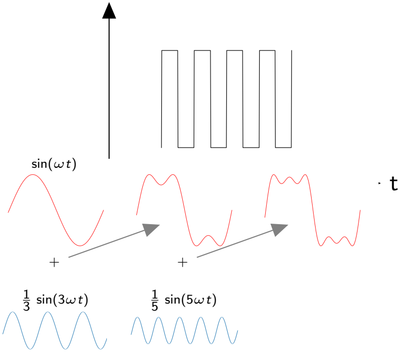
# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.



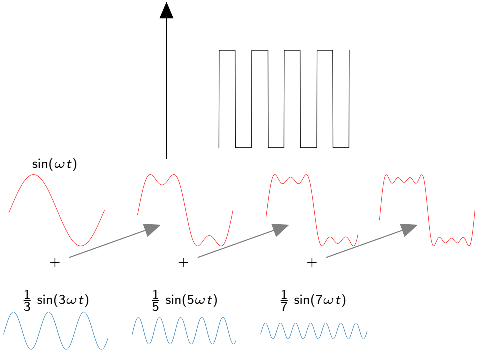
# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.



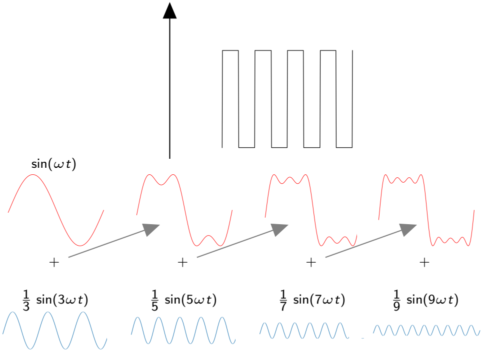
# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.



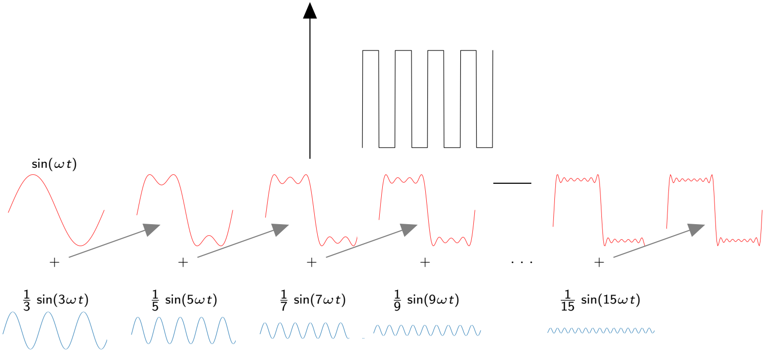
# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.



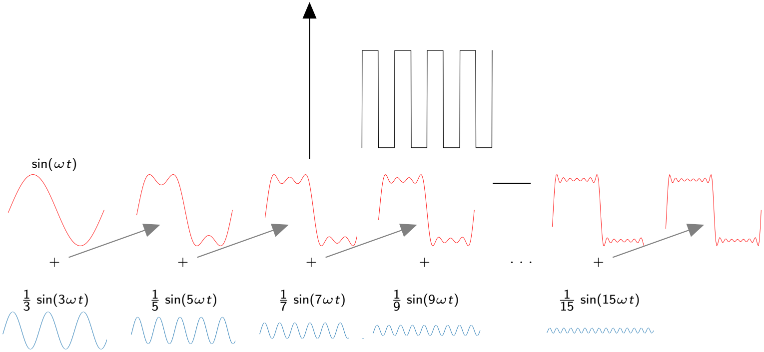
# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.



# Decomposing a signal

- Let's decompose a square wave with sines.
  - Decomposons un square wave.




- So we can approximate a square wave with
  - Alors on peut approximer un onde carré avec

$$SW(\omega t) \approx \sin(\omega t) + \frac{1}{3}\sin(3\omega t) + \frac{1}{5}\sin(5\omega t) + \frac{1}{7}\sin(7\omega t) + \ldots$$

# Frequency Representation

- The goal is to find the contribution of each sinusoid (or 'frequency').
  - Le but est d'obtenir la contribution de chaque sinusoid (ou 'fréquence).

# Frequency Representation

- The goal is to find the contribution of each sinusoid (or ‘frequency’).
  - Le but est d’obtenir la contribution de chaque sinusoid (ou ‘fréquence).
- We can calculate similarity between different sinusoids and the input waveform.
  - On peut calculer des similarités entre differents sinusoids et l’entrée.

# Frequency Representation

- The goal is to find the contribution of each sinusoid (or 'frequency').
  - Le but est d'obtenir la contribution de chaque sinusoid (ou 'fréquence).
- We can calculate similarity between different sinusoids and the input waveform.
  - On peut calculer des similarités entre differents sinusoids et l'entrée.
- Do you remember how we calculate the similarity between two vectors?
  - Est-ce que vous vous en souvenez comment-t-on calcule similarité entre deux vecteurs?

# Frequency Representation

- The goal is to find the contribution of each sinusoid (or 'frequency').
  - Le but est d'obtenir la contribution de chaque sinusoid (ou 'fréquence).
- We can calculate similarity between different sinusoids and the input waveform.
  - On peut calculer des similarités entre differents sinusoids et l'entrée.
- Do you remember how we calculate the similarity between two vectors?
  - Est-ce que vous vous en souvenez comment-t-on calcule similarité entre deux vecteurs?

- **Inner Product!**

# Table of Contents

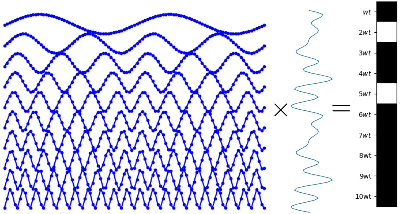
# Fourier Transform

- Fourier Transform is basically a projection over sinusoids.
  - Dans le fond, FT est une projection sur des sinusoids.
- Let's do that for this signal.
  - Faisons ça pour ce signal.

# Fourier Transform

- Fourier Transform is basically a projection over sinusoids.
  - Dans le fond, FT est une projection sur des sinusoids.
- Let's do that for this signal.
  - Faisons ça pour ce signal.

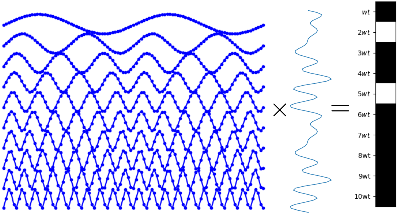
- FT in principle:



# Fourier Transform

- Fourier Transform is basically a projection over sinusoids.
  - Dans le fond, FT est une projection sur des sinusoids.
- Let's do that for this signal.
  - Faisons ça pour ce signal.

- FT in principle:




- This is nice, but is this general enough?
  - Bon, mais est-ce assez générale?

# No! You need cosines to span the space

- The signal we saw before was,
  - Le signal qu'on avait vu était:

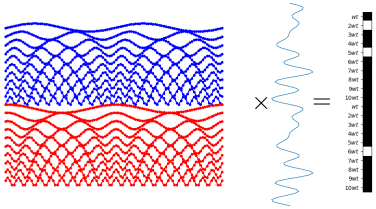
$$y(t) = \sin(2\omega t) + \sin(5\omega t) + \underbrace{\cos(6\omega t)}_{\text{We missed this guy!}}$$

## No! You need cosines to span the space

- The signal we saw before was,
  - Le signal qu'on avait vu était:

$$y(t) = \sin(2\omega t) + \sin(5\omega t) + \underbrace{\cos(6\omega t)}_{\text{We missed this guy!}}$$

- Let's extend our bases,
  - Élargons notre ensemble de bases:

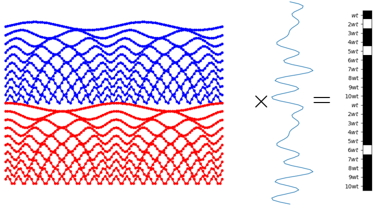


# No! You need cosines to span the space

- The signal we saw before was,
  - Le signal qu'on avait vu était:

$$y(t) = \sin(2\omega t) + \sin(5\omega t) + \underbrace{\cos(6\omega t)}_{\text{We missed this guy!}}$$

- Let's extend our bases,
  - Élargons notre ensemble de bases:




- But wait, I remember that Fourier Transform gave us complex numbers. What's that about?
  - Mais chuis confus-là, je me souviens que FT nous donnait des chiffres complexes?

# Fourier Transform Formal Definition

- Remember Euler's formula:

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$

- So it seems like, we can use complex exponentials to project on to cosines and sines at the same time.
  - On alors peut utiliser les exponentiels complexes pour projeter sur les cosines et sines en meme temps.

# Fourier Transform Formal Definition

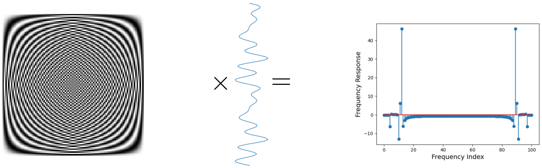
- Remember Euler's formula:
$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$
- So it seems like, we can use complex exponentials to project on to cosines and sines at the same time.
  - On alors peut utiliser les exponentiels complexes pour projeter sur les cosines et sines en meme temps.
- Discrete Fourier Transform (DFT):
$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-j2\pi\frac{k}{N}n\right), \text{ where}$$
$$\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}],\ k \in \{0, \ldots, N-1\}.$$
- Note that $k$ corresponds to frequencies, and we have the same number of frequencies as the signal length $N$.
  - Notez que $k$ indice est la fréquence, et on a meme nombres de fréquences que le longeur $N$.

# Fourier Transform Formal Definition

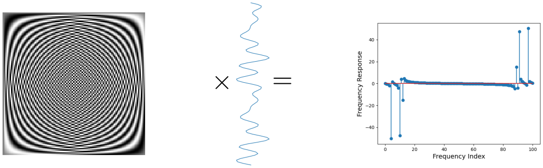
- Remember Euler's formula:
$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$$
- So it seems like, we can use complex exponentials to project on to cosines and sines at the same time.
  - On alors peut utiliser les exponentiels complexes pour projeter sur les cosines et sines en meme temps.
- Discrete Fourier Transform (DFT):
$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-j2\pi \frac{k}{N} n\right), \text{ where}$$
$$\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}],\ k \in \{0, \ldots, N-1\}.$$
- Note that $k$ corresponds to frequencies, and we have the same number of frequencies as the signal length $N$.
  - Notez que $k$ indice est la fréquence, et on a meme nombres de fréquences que le longeur $N$.
- By the way, do you see that this is a matrix? $\exp\left(-j2\pi \frac{k}{N} n\right)$
  - Vous voyez que c'est une matrice?

# DFT in action

- Real Part / La partie réel




- Imaginary Part / La partie imaginaire



# Remarks

- Notice that les results are symmetric. This is because of the construction of the DFT matrix.
  - Notez que les résultats sont symmetriques. C'est à cause de la construction de la matrice DFT.

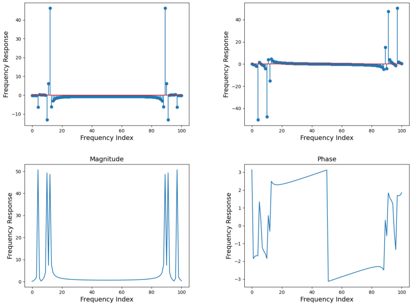
# Remarks

- Notice that les results are symmetric. This is because of the construction of the DFT matrix.
  - Notez que les résultats sont symmetriques. C'est à cause de la construction de la matrice DFT.
- Note that the DFT matrix repeats after the half. We will get back to this later.
  - Notez que les matrices DFT répetent après la moitié. On va parler de ça après.

# Another way of interpreting DFT

- Note that we get complex numbers. We can calculate magnitude and phase.
  - Notez qu'on obtiens des nombres complex. Donc on peut calculer la magnitude et la phase.

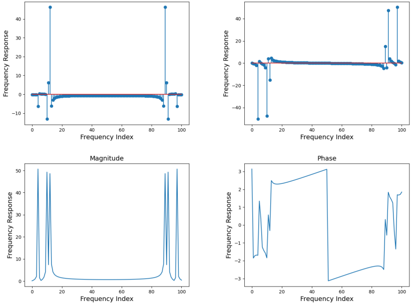
$$
\begin{aligned}
|X_k| &= \sqrt{\mathrm{Re}(X_k)^2 + \mathrm{Im}(X_k)^2} \\
\angle X_k &= \mathrm{atan2}(\mathrm{Im}(X_k), \mathrm{Re}(X_k))
\end{aligned}
$$



# Another way of interpreting DFT

- Note that we get complex numbers. We can calculate magnitude and phase.
  - Notez qu'on obtiens des nombres complex. Donc on peut calculer la magnitude et la phase.

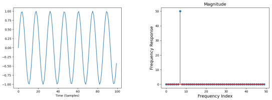
$$
\begin{aligned}
|X_k| &= \sqrt{\mathrm{Re}(X_k)^2 + \mathrm{Im}(X_k)^2} \\
\angle X_k &= \mathrm{atan2}(\mathrm{Im}(X_k), \mathrm{Re}(X_k))
\end{aligned}
$$



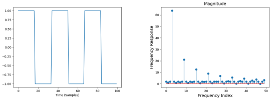

- Magnitude is easy to interpret. Phase is not as easy always.
  - Magnitude est facile est interpreter. La phase n'est pas toujours si facile à interpreter.

# DFTs of different signals

- A single sinusoid

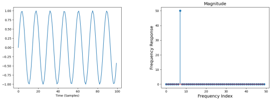



- Square wave

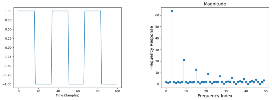


# DFTs of different signals

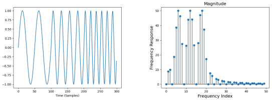
- A single sinusoid




- Square wave




- Sinusoids with increasing frequencies

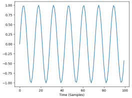


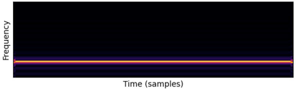

- So, for time varying signals, DFT is not that great.
  - Pour les signaux qui varie avec le temps DFT n'est pas idéal.

# Table of Contents

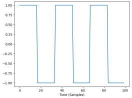
# Time-frequency representation

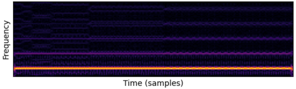
- A single sinusoid



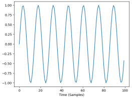

- Square wave

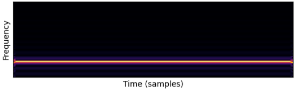


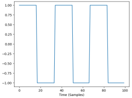
# Time-frequency representation

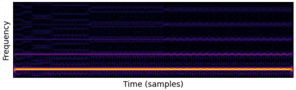
- A single sinusoid



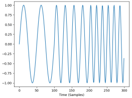


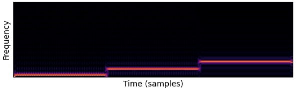

- Square wave





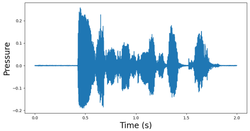

- Sinusoids with increasing frequencies





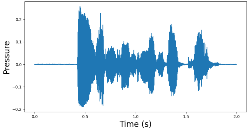
# A real example for speech

**Time Domain**

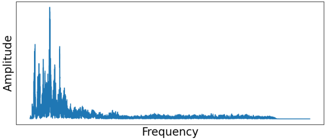


# A real example for speech

**Time Domain**




**Frequency Domain –** but what is this really? I see low freqs, but where is time information? – On voit freq basses mais pas d'info sur temps.

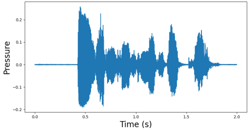


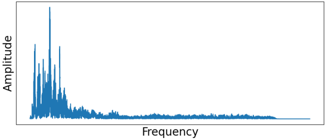
# A real example for speech

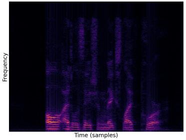
**Time Domain**

**Frequency Domain –** but what is this really? I see low freqs, but where is time information? – On voit freq basses mais pas d'info sur temps.

**Time-Frequency Domain –** We 'see' the signal. – on voit le signal.

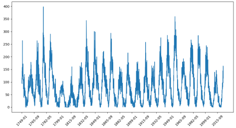


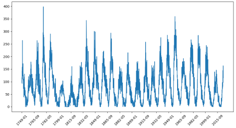
# An example from another domain

**Time Domain** Counts of sunspots wrt. time

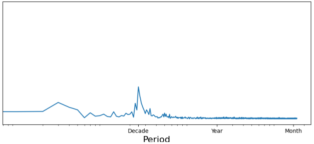


# An example from another domain

**Time Domain** Counts of sunspots wrt. time

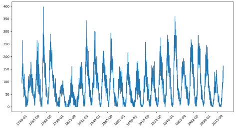


**Frequency Domain**

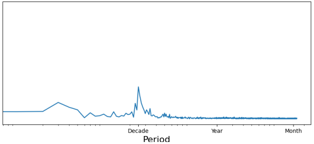


# An example from another domain

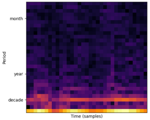
**Time Domain** Counts of sunspots wrt. time



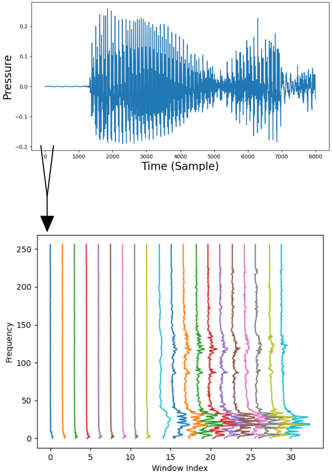
**Frequency Domain**



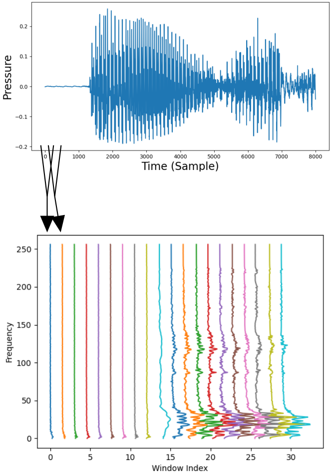

**Time-Frequency Domain**



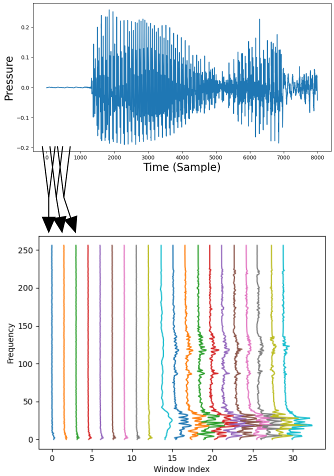
# How to Calculate the Spectrogram (STFT)



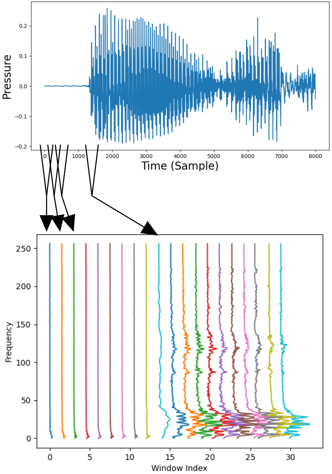
# How to Calculate the Spectrogram (STFT)



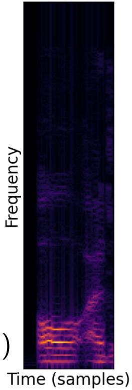
# How to Calculate the Spectrogram (STFT)



# How to Calculate the Spectrogram (STFT)

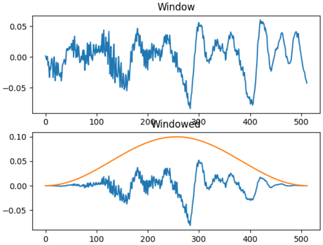


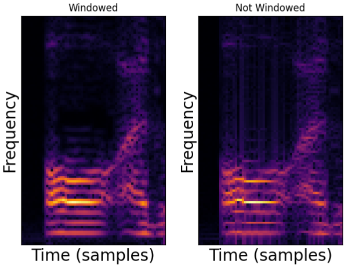

It's nice to show it as an image! (imshow())



# STFT considerations

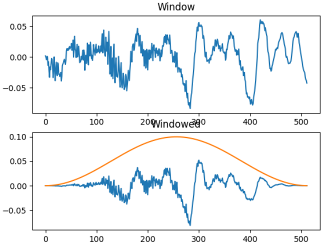
- Windowing (Utilisation de fenetres)
  - We apply a window, before calculating the DFT.
    - On applique une fenetre avant de calculer le DFT.

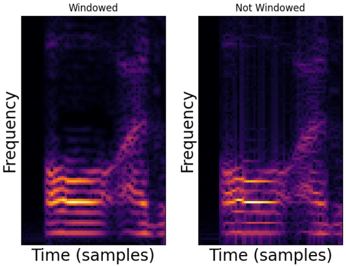


# STFT considerations

- Windowing (Utilisation de fenetres)
  - We apply a window, before calculating the DFT.
    - On applique une fenetre avant de calculer le DFT.

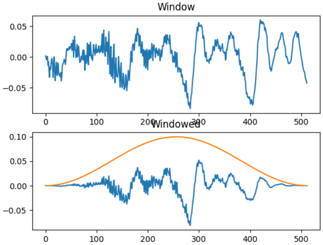


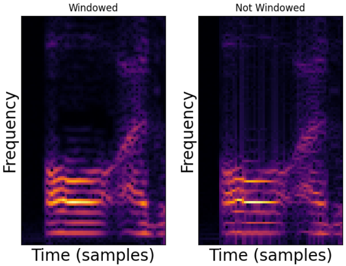




- Notice that not windowed STFT contains more high frequency artifacts
  - Notez que STFT qui n'est pas windowé contiens plus des artifacts de haute fréquence.

# STFT considerations

- Windowing (Utilisation de fenetres)
  - We apply a window, before calculating the DFT.
    - On applique une fenetre avant de calculer le DFT.

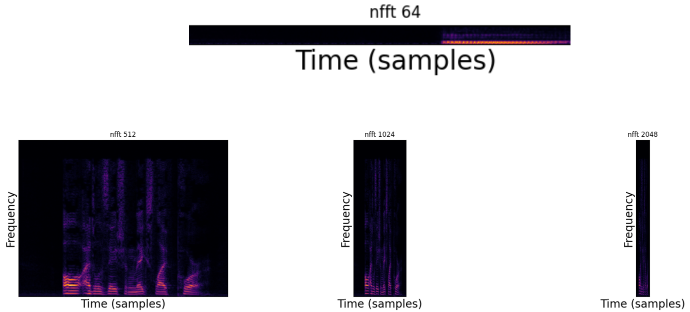



- Notice that not windowed STFT contains more high frequency artifacts
  - Notez que STFT qui n'est pas windowé contiens plus des artifacts de haute fréquence.
- We add an overlap to make up for using tapering windows. / On ajoute un overlap aussi pour compenser pour les fenetres qui réduit en amplitude.

# Time/Frequency Tradeoff

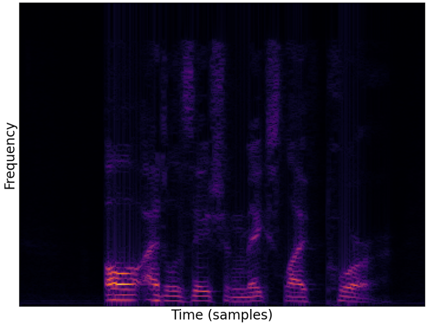
- Heisenberg's uncertainty principle - We can not know the frequency and time location of a wave.
  - On ne peut pas déterminer la fréquence et et la localisation temporelle d'une onde.
- In the context of spectrograms: Big DFT sacrifice temporal resolution, Small DFTs have bad frequency resolution
  - Grand DFT a une mauvaise resolution temporelle, Petit DFTs ont une mauvaise resolution fréquentielle



# Table of Contents

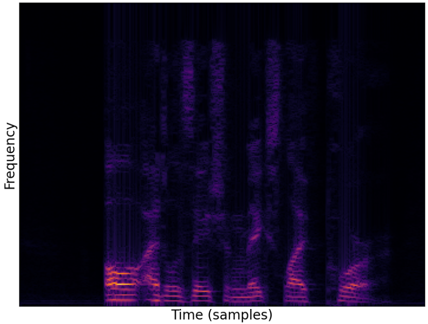
# Mel-Frequency Spectrograms

- You notice that most of the energy is concentrated on the lower frequencies.
  - Vous vous rendez compte que l'energie est plutot concentrée dans les frequences bas.



# Mel-Frequency Spectrograms

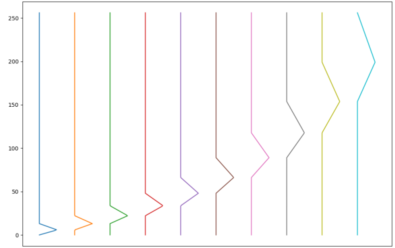
- You notice that most of the energy is concentrated on the lower frequencies.
  - Vous vous rendez compte que l'energie est plutot concentrée dans les frequences bas.




- This is a bit wasteful.

# Mel-Frequency Spectrograms

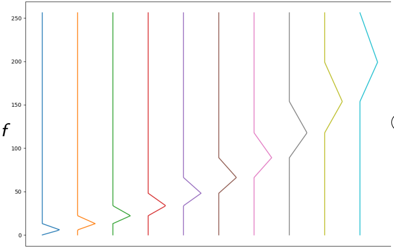
- We can ‘warp’ the frequency axis.



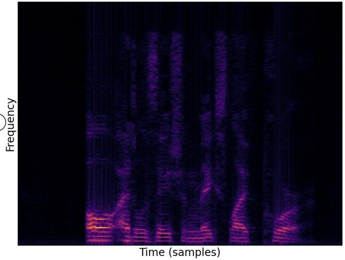
# Mel-Frequency Spectrograms

- We can ‘warp’ the frequency axis.

$\sum_f$



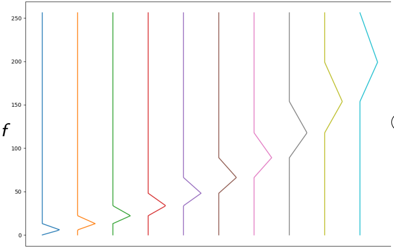

$\odot$



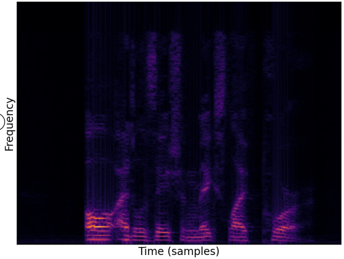
# Mel-Frequency Spectrograms

- We can 'warp' the frequency axis.

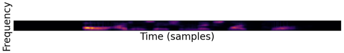
$\sum_f$




$\odot$

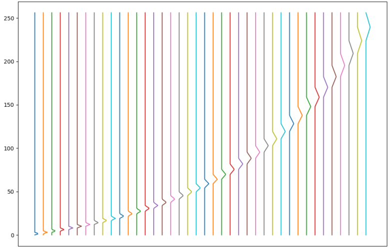



$=$



# Mel-Frequency Spectrograms

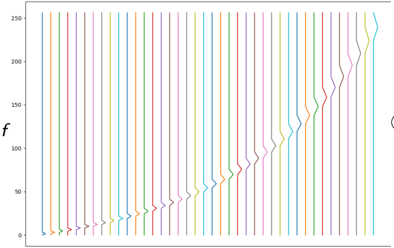
- We can 'warp' the frequency axis with more filters also (e.g. 40).



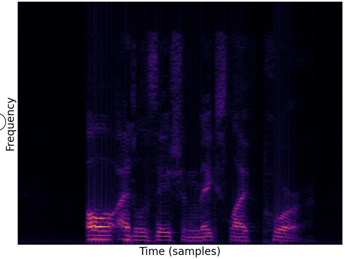
# Mel-Frequency Spectrograms

- We can 'warp' the frequency axis with more filters also (e.g. 40).

$\sum_f$



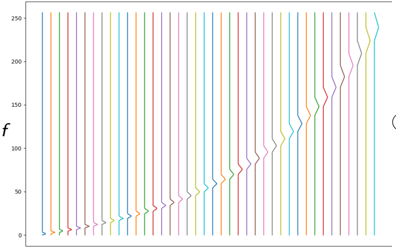

$\odot$



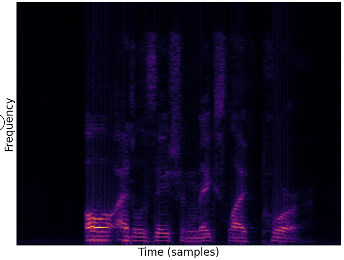
# Mel-Frequency Spectrograms

- We can ‘warp’ the frequency axis with more filters also (e.g. 40).

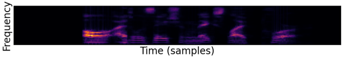
$\sum_f$




$\odot$

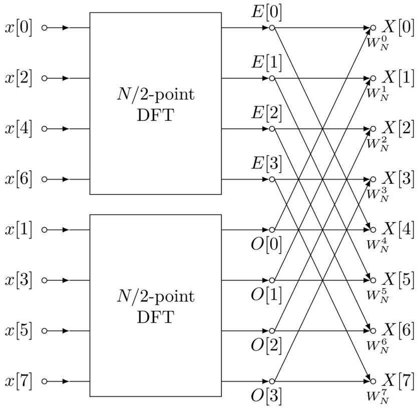



$=$



# Fast Fourier Transform (FFT)

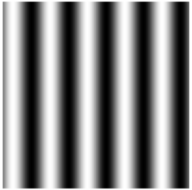
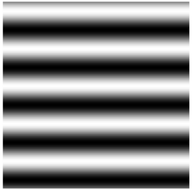
- DFT matrix has symmetries.
- DFT can be decomposed into DFTs of half the size.
  - La DFT peut etre decomposé aux 2 DFTs avec la moitié de la taille originale.
- We reduce the complexity from $\mathcal{O}(N^2)$(*why*?) to $\mathcal{O}(N \log N)$.



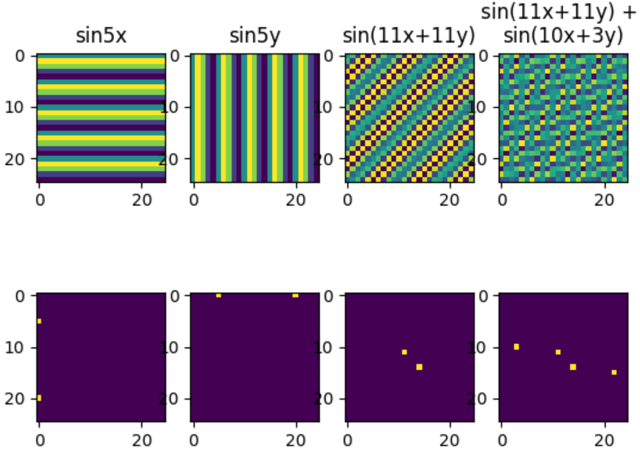

- Whenever you can use FFTs, use em!

# Image DFTs

- We can généralize to images as well
  - On peut généraliser aux images aussi.
- The DFT bases in 2d
  - Les bases DFT en 2d

# Some example DFTs



# How to calculate 2d DFTs?

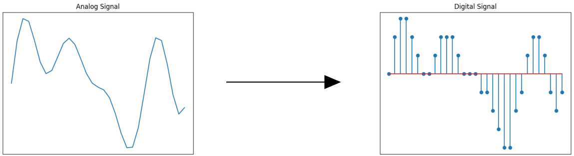
- For images $Y = FXF$
- For Tensors $F_{il_1} X_{ijk} F_{jl_2} F_{kl_3} \rightarrow Y_{l_1, l_2, l_3}$

# Table of Contents

# Table of Contents

# Analog to Digital

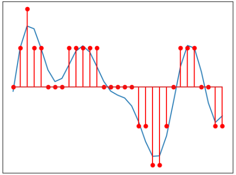
- How do we store / acquire signals?
  - Comment est-ce qu'on mets les signaux dans les ordinateurs?
- We convert the analog signals to digital.
  - On convertis signaux analogues au digital.




- It's not straightforward how to do this conversion.
  - C'est pas trivial comment faire cette conversion.

# Signal Representation – Quantization

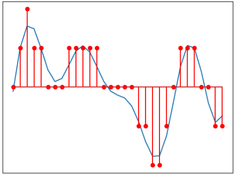
- We have to quantize a signal in order to digitize. (Quantize the y-axis)
  - On est obligé de quantiser pour digitiser. (l'axe y)



- We measure the precision in terms of bits. More bits we use, more signal-to-noise ratio we have.
  - On mésure la precision de la quantization en terme de bits. Plus de bits qu'on utilise, ça donne plus de SNR.

# Signal Representation – Quantization

- We have to quantize a signal in order to digitize. (Quantize the y-axis)
  - On est obligé de quantiser pour digitiser. (l'axe y)



- We measure the precision in terms of bits. More bits we use, more signal-to-noise ratio we have.
  - On mésure la precision de la quantization en terme de bits. Plus de bits qu'on utilise, ça donne plus de SNR.
- Example average numbers:
  - 8bit - 48dB poor
  - 12 bits - 72dB okayish
  - 16 bits - 96 dB good
  - 24 bits - 144 dB overkill

# Quantization in practice

8, 7, 5 bits

4, 3, 2 bits, 1 bit

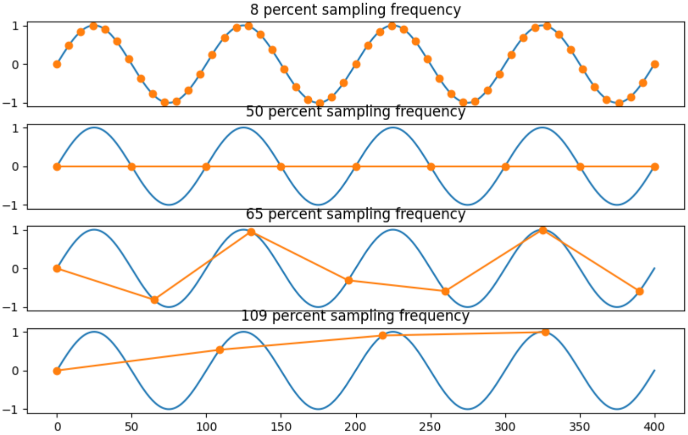
# Table of Contents

# Sampling / Échantillonnage

- How often we sample is also important!
  - C'est aussi important la fréquence avec la quelle on échantillon. (l'axe de temps)
    - We use Hz to measure sampling freq. / On utilize Hz pour mésurer la fréquence d'echantillonnage.
- Important: Nyquist Rate: We must sample x2 above the highest frequency we want to represent.
  - Important: Le rate de Nyquist: On doit échantillonner 2 fois au délà de la fréquence qu'on veut répresenter.
- Perceptual Limits:
  - Our ears: We hear up to 20kHz (declines with age) So above 40kHz sampling is required.
  - Seeing: We perceive only up to 60Hz, so 120 Hz or up is required.
- Limites perceptuels:
  - Nos oreilles: La limite est 20kHz. Donc on a besoin d'une fréquence d'échantillonnage de 40Hz.
  - Nos yeux: La limite est 60Hz. On a besoin donc 120Hz.
- Common sampling rates:
  - Speech: 8kHz, 16kHz, Music: 32kHz, 44.1kHz, Pro-Audio 96kHz
  - Movies: 24fps (recently 48fps) HDTV: 60fps, ...

# Aliasing

- Frequencies above 50% of the sampling rate are mis-represented.
  - Les fréquences sur 50% du rate d'échantillonnage sont mal représentés.



# In real-life

- Electricity consumption sampled poorly.
  - Une mauvaise échantillonnage de la consomptions de l'éléctricité.

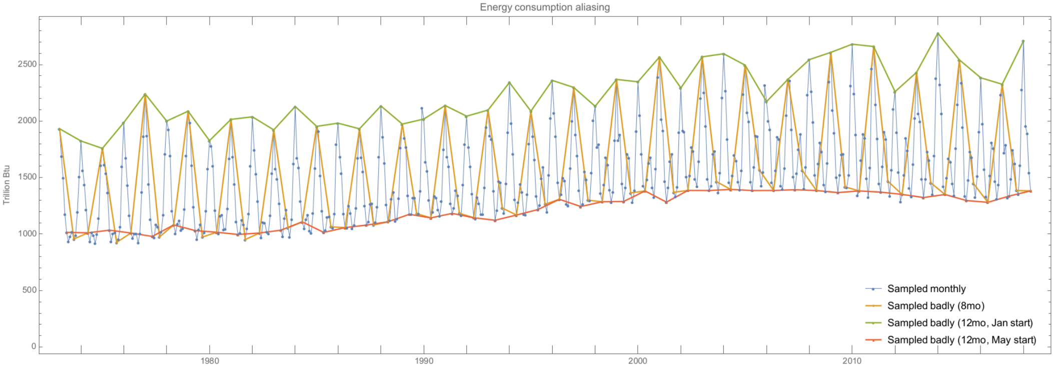



image taken from UIUC MLSP

# In real-life

- Electricity consumption sampled poorly.
  - Une mauvaise échantillonnage de la consomptions de l'éléctricité.
- Different sampling leads to different conclusions.
  - Différent l'échantillonnage mène à des conclusions différentes.

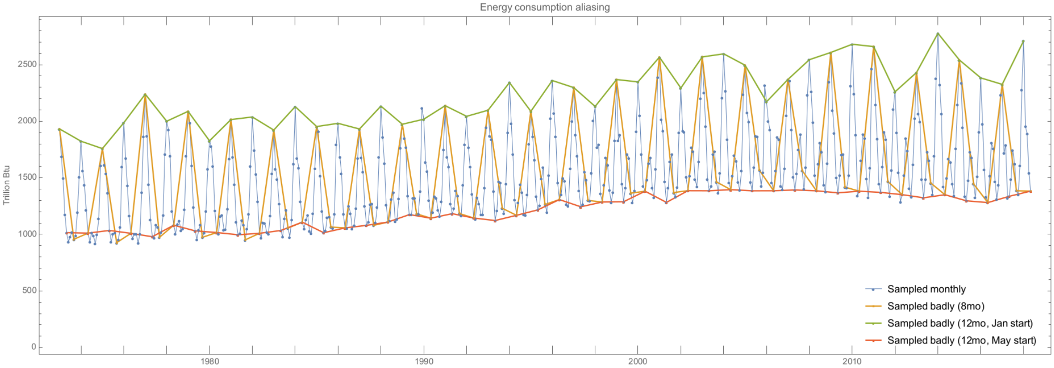
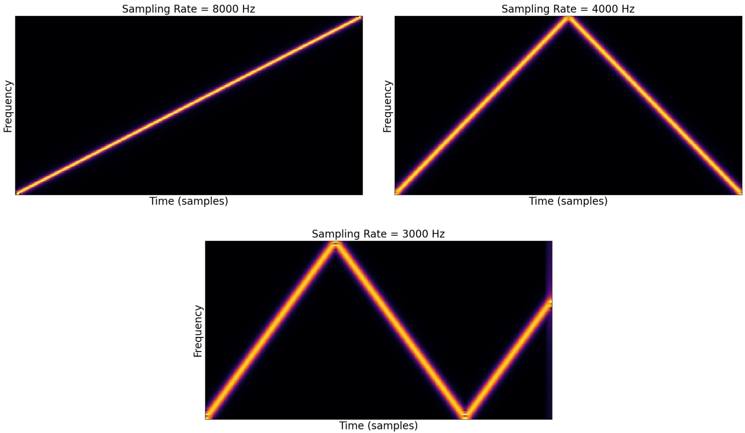



image taken from UIUC MLSP

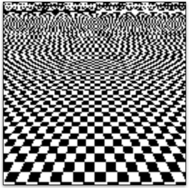
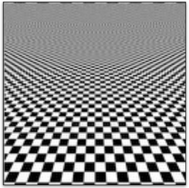
# Aliasing in Sinusoid Sweeping




We have frequency sweeps from 0-4kHz, for different sampling rates. Notice the aliasing when sampling rate is lower!

`Sweep 1` `Sweep 2` `Sweep 3`

# Aliasing in Images/Video

https://www.youtube.com/watch?v=R-IVw8OKjvQ

# Table of Contents

# Convolution

- This is an extremely important operation that comes up all the time. (e.g. filtering)
  - C'est une opération extremement importante qu'on va voir souvent.

$$
\begin{aligned}
x(t) * w(t) := & \sum_{i=0}^{M-1} x(i)w(t-i) \\
= & x(0)w(t) + x(1)w(t-1) + x(1)w(t-2) + \dots \\
& + x(M)w(t-M)
\end{aligned}
$$

# Convolution

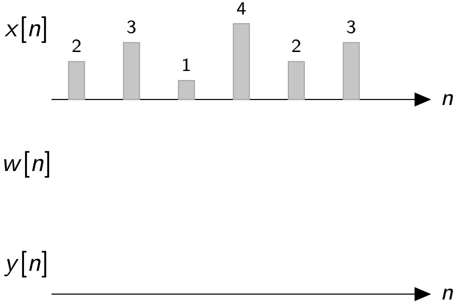
- This is an extremely important operation that comes up all the time. (e.g. filtering)
  - C'est une opération extremement importante qu'on va voir souvent.

$$
\begin{aligned}
x(t) * w(t) :=& \sum_{i=0}^{M-1} x(i)w(t-i) \\
=& x(0)w(t) + x(1)w(t-1) + x(1)w(t-2) + \dots \\
&+ x(M)w(t-M)
\end{aligned}
$$

- If $x$ is of length $M$ and $w$ is of length $N$, the result is of length $M+N-1$.
  - Si $x$ est de longeur $M$ et $w$ est du longeur $N$, le résultat est de longeur $M+N-1$.

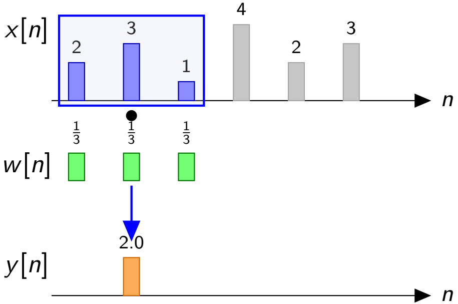
# Convolution: slide, multiply, sum

- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).



# Convolution: slide, multiply, sum

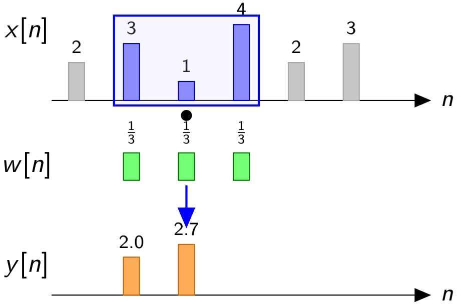
- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).




$$y[1] = \frac{2+3+1}{3} = 2.0$$

# Convolution: slide, multiply, sum

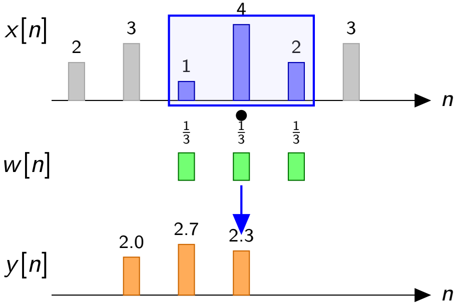
- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).




$y[2] = \frac{3+1+4}{3} \approx 2.7$

# Convolution: slide, multiply, sum

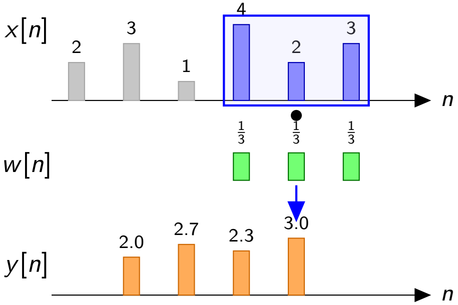
- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).




$$y[3] = \frac{1+4+2}{3} \approx 2.3$$

# Convolution: slide, multiply, sum

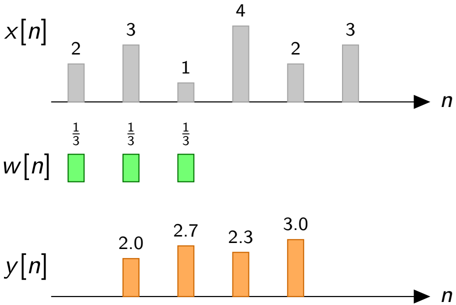
- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).




$$y[4] = \frac{4+2+3}{3} = 3.0$$

# Convolution: slide, multiply, sum

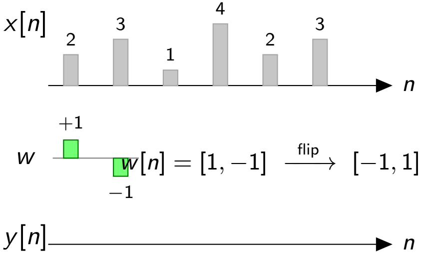
- Slide the kernel $w[n]$ across the signal $x[n]$; each output $y[n]$ is the weighted sum of the overlap. Here $w = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ (a local average).




The kernel *smooths* the signal: each output is a local average.

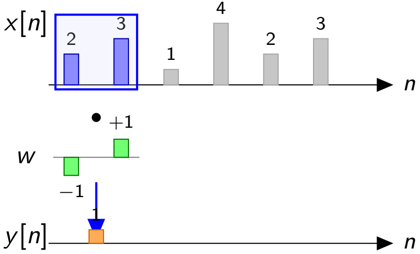
# Convolution: flip, then slide

- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \rightarrow [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).



# Convolution: flip, then slide

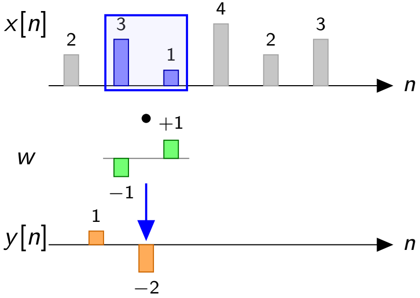
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




$$y[1] = (+1) \cdot 3 + (-1) \cdot 2 = 1$$

# Convolution: flip, then slide

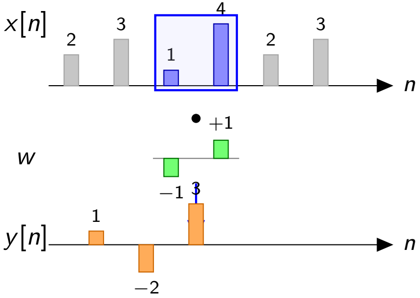
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




$$y[2] = (+1) \cdot 1 + (-1) \cdot 3 = -2$$

# Convolution: flip, then slide

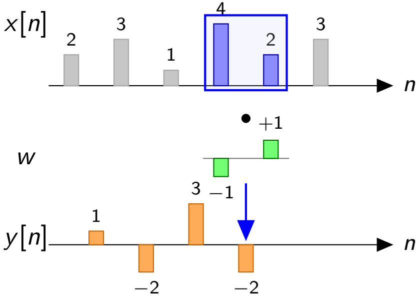
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




$$y[3] = (+1) \cdot 4 + (-1) \cdot 1 = 3$$

# Convolution: flip, then slide

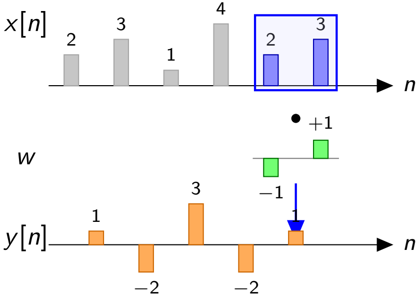
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




$$y[4] = (+1) \cdot 2 + (-1) \cdot 4 = -2$$

# Convolution: flip, then slide

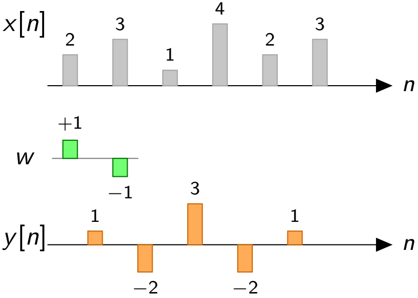
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




$$y[5] = (+1) \cdot 3 + (-1) \cdot 2 = 1$$

# Convolution: flip, then slide

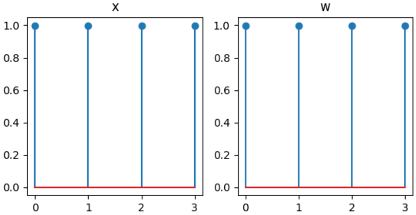
- Now $w = [1, -1]$. Convolution *flips* the kernel first: $[1, -1] \to [-1, 1]$, then slides. The result is the difference $y[n] = x[n] - x[n-1]$ (an edge detector).




The difference $y[n] = x[n] - x[n-1]$ responds at *edges* (changes) in the signal.

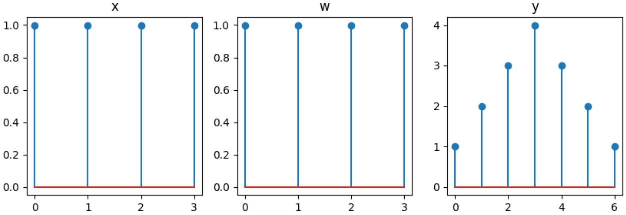
# Convolution

$$y = x * w$$



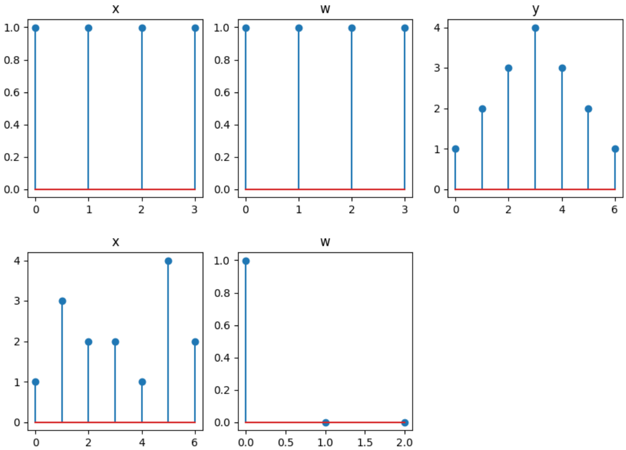
# Convolution

$$y = x * w$$



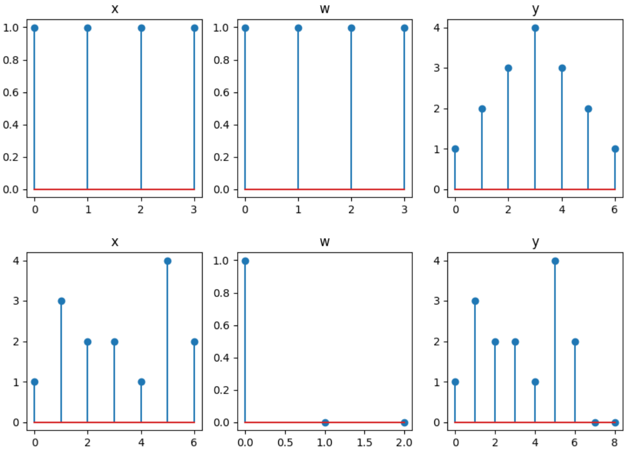
# Convolution

$$y = x * w$$



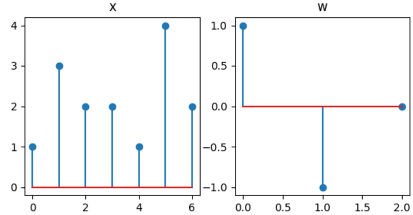
# Convolution

$$y = x * w$$



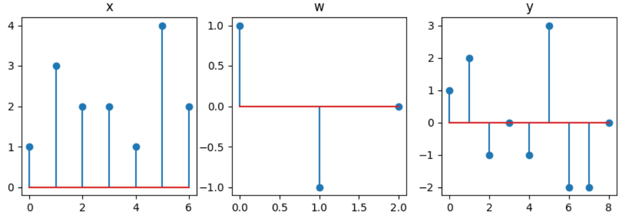
# Convolution

$$y = x * w$$



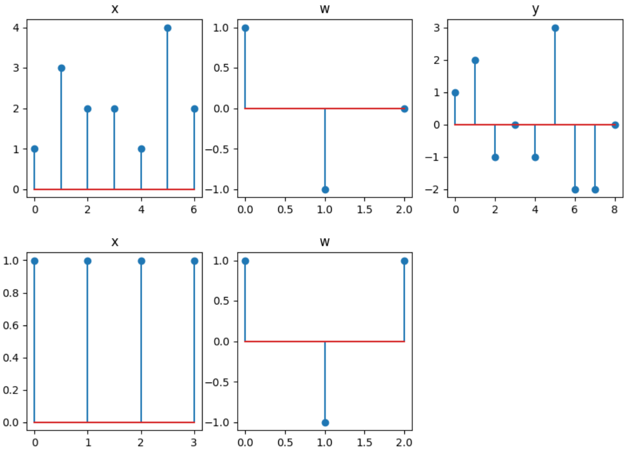
# Convolution

$$y = x * w$$



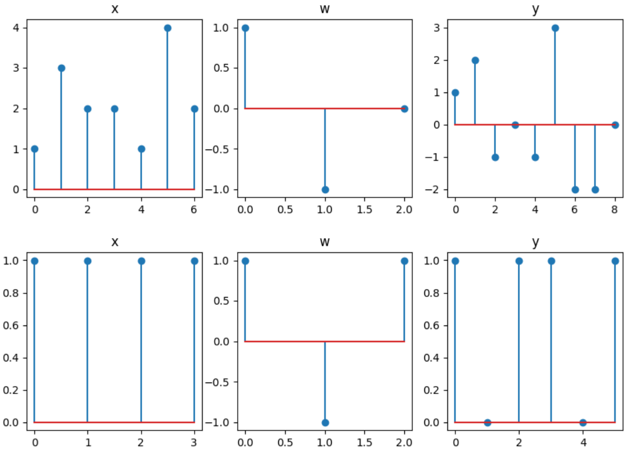
# Convolution

$$y = x * w$$



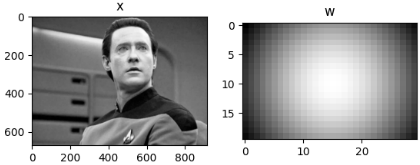
# Convolution

$$y = x * w$$



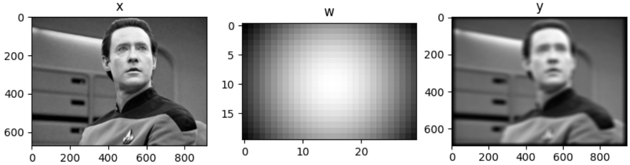
# Image Convolution

$$y = x * w$$



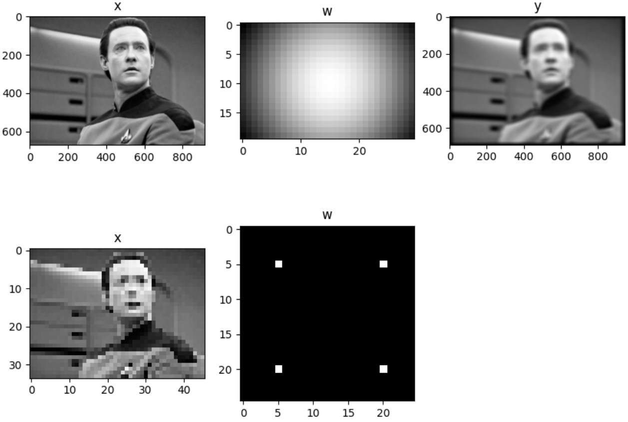
# Image Convolution

$$y = x * w$$



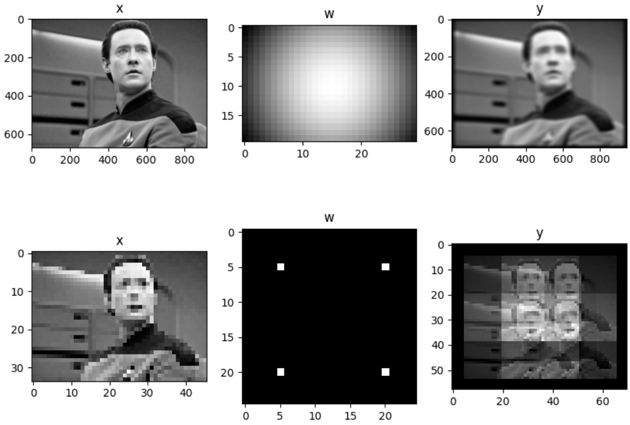
# Image Convolution

$$y = x * w$$



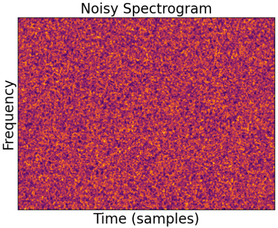
# Image Convolution

$$y = x * w$$



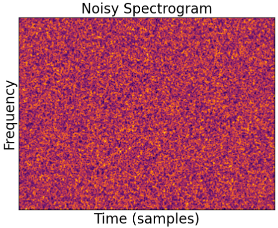
# Filtering Audio with Convolution

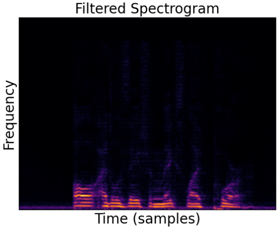
- Consider this noisy audio. `Listen`.
- The filtered version with an averaging kernel. `Listen`.
  - La version filté avec un noyau de moyenne.



# Filtering Audio with Convolution

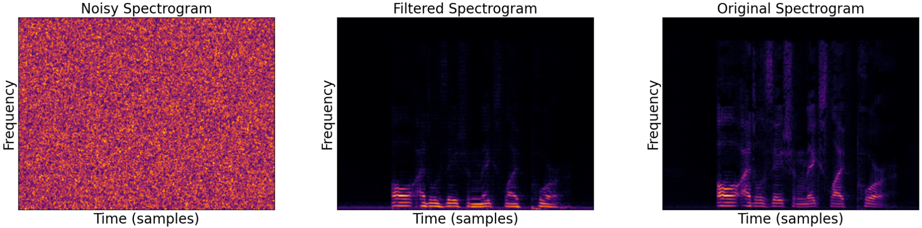
- Consider this noisy audio. `Listen`.
- The filtered version with an averaging kernel. `Listen`.
  - La version filté avec un noyau de moyenne.



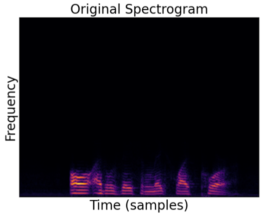
# Filtering Audio with Convolution

- Consider this noisy audio. `Listen`.
- The filtered version with an averaging kernel. `Listen`.
  - La version filté avec un noyau de moyenne.

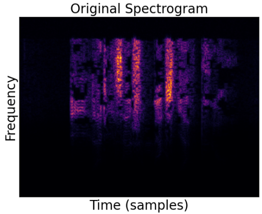


# More on filtering

- Low-pass filtering (cut-off at 1kHz) Listen.

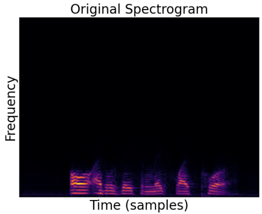



- High-pass filtering (cut-off at 4kHz) Listen.

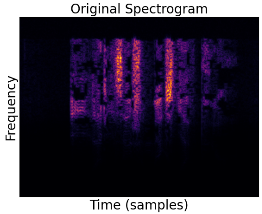


# More on filtering

- Low-pass filtering (cut-off at 1kHz) `Listen`.




- High-pass filtering (cut-off at 4kHz) `Listen`.




- There are also band-stop, band-pass filtering..
  - Il existe aussi band-stop, band-pass filtering.

# Fast Convolution

- Convolution is an $\mathcal{O}(N^2)$ operation.
- However we can use FFT to get it down to $\mathcal{O}(N \log N)$.
  - On peut prendre avantage de le FFT pour avoir une complexité de $\mathcal{O}(N \log N)$.
- Convolution in time domain, is multiplication in the Fourier Domain.
  - Convolution dans le domaine de temps est multiplication dans le domaine de Fourier.

$$
\begin{aligned}
F(x * w) =& Fx \odot Fw \\
\rightarrow x * w =& F^{-1}(Fx \odot Fw)
\end{aligned}
$$

- Use FFT whenever you can!

# Convolution as a Matrix Multiplication

- Convolution can be expressed as a matrix multiplication.
  - Convolution peut-etre exprimée comme une multiplication de matrices.

$$w * x = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \underbrace{\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ 0 & w_2 & w_1 & 0 \\ 0 & 0 & w_2 & w_1 \\ 0 & 0 & 0 & w_2 \end{bmatrix}}_{:=C} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

- $C$ is a Circulant / Toeplitz matrix.

# Convolution as a Matrix Multiplication

- Convolution can be expressed as a matrix multiplication.
  - Convolution peut-etre exprimée comme une multiplication de matrices.

$$w * x = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \underbrace{\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ 0 & w_2 & w_1 & 0 \\ 0 & 0 & w_2 & w_1 \\ 0 & 0 & 0 & w_2 \end{bmatrix}}_{:=C} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

- $C$ is a Circulant / Toeplitz matrix.
- And the eigenvectors are sinusoids.

# Convolution as a Matrix Multiplication

- Convolution can be expressed as a matrix multiplication.
  - Convolution peut-etre exprimée comme une multiplication de matrices.

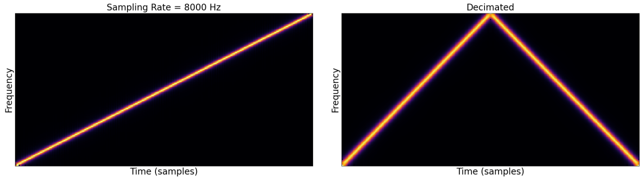
$$w * x = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \underbrace{\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ 0 & w_2 & w_1 & 0 \\ 0 & 0 & w_2 & w_1 \\ 0 & 0 & 0 & w_2 \end{bmatrix}}_{:=C} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

- $C$ is a Circulant / Toeplitz matrix.
- And the eigenvectors are sinusoids.
- Not only that, the eigenvectors form the DFT matrix!
  `https://web.mit.edu/18.06/www/Spring17/Circulant-Matrices.pdf`

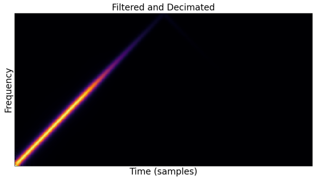
# Table of Contents

# Downsampling

- Let's say we want to downsample. Simple decimation introduces aliasing.
- Si on sous-echantillonne, simple decimation introduit de l'aliasing.



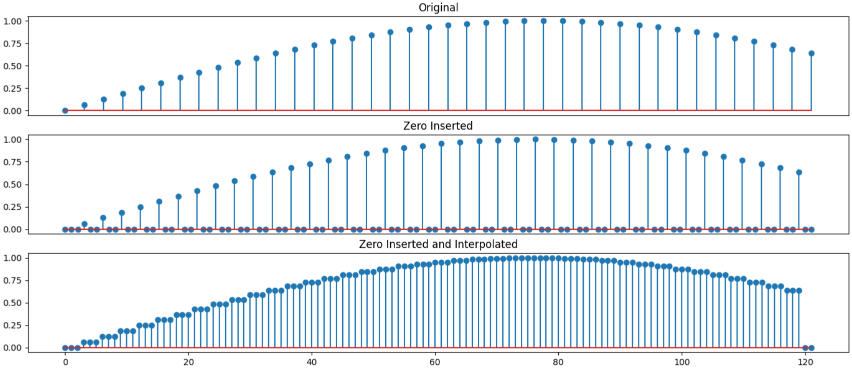

- The solution is to first filter and then downsample.
  - La solution est de filtrer et puis downsampler




- We lose the high-freqs, but we at least avoid aliasing.
  - On perd les hautes frequences mais on moins pas d'aliasing.

# Upsampling

- To upsample by a factor of $L$, the procedure is to first insert $L-1$ zeros, and then to interpolate.
  - Pour upsampler de la facteur $L$, le procedure est de d'abord inserer $L-1$ zéros, et puis faire de l'interpolation.



# Recap

- Signal Representations
  - Time, Frequency, Time-Frequency
- Discrete Fourier Transform
- Short-Time Fourier Transform
- Sampling, Resampling
- Convolution

# Furher Reading

- `http://www.dspguide.com/pdfbook.htm` – nice free book, check it out.
- `https://dspguru.com/dsp/howtos/`

# Next week

- We get started with machine learning (sometimes for signal processing)