



Traitement 1D

Modulation AM

Outils Numériques / Semestre 5
Institut d'Optique / B3_0

Déroulement du bloc AM

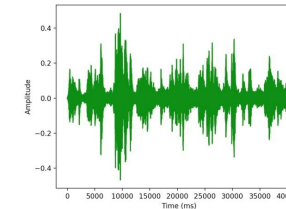
3 séances introductives (2h/séance)

2 blocs de 4 et 5 séances (2h/séance)

- Sur machine
- En binôme
- 2 encadrant.es par séance

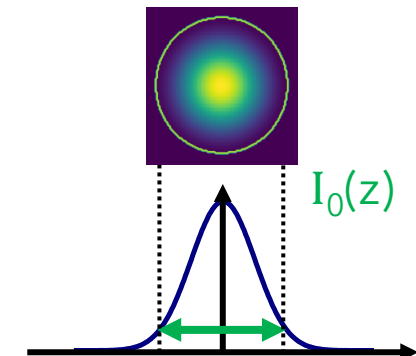
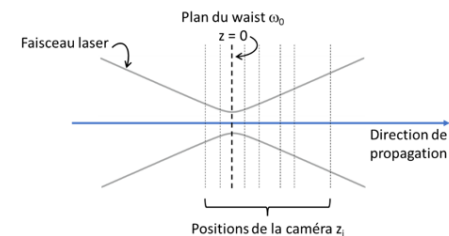
Bloc AM : Traitement de données 1D

Problème 1 : signal modulé en amplitude / acquisition numérique



Bloc Laser : Traitement de données 2D

Problème 2 : images d'un faisceau LASER en différents points d'un chemin optique



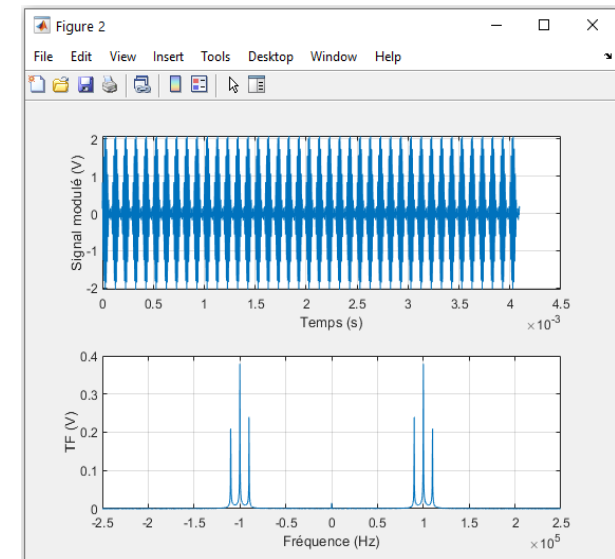
Contexte

- **Instrumentation numérique**

- Acquisition de données
- Sauvegarde de données
- Analyse des données
- Traitement des données

Signaux modulés en amplitude

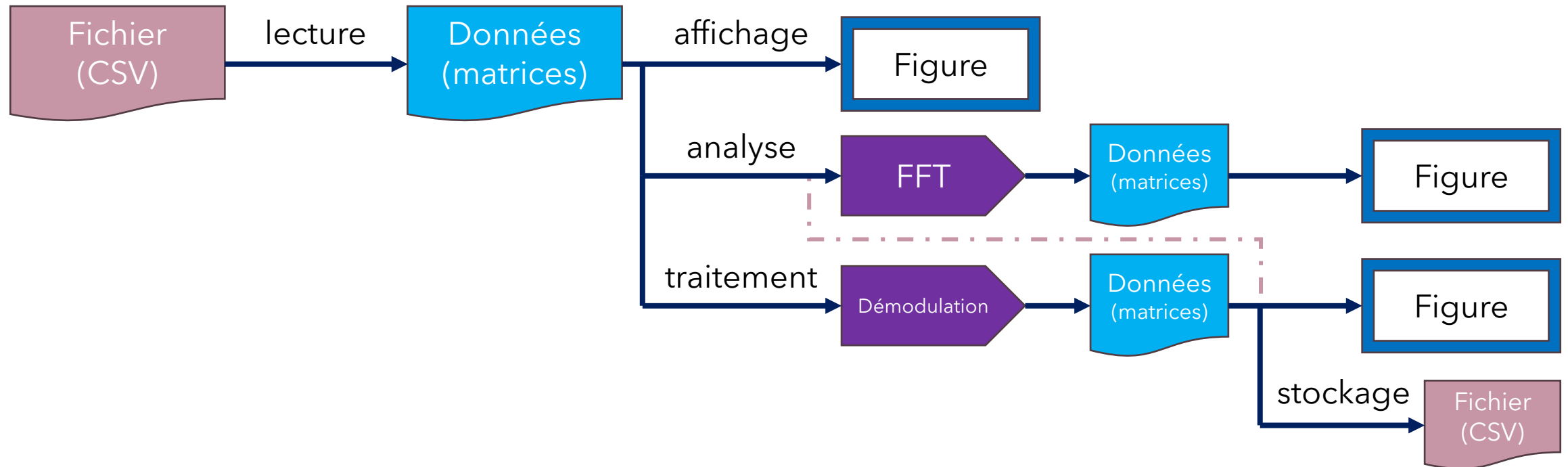
Transformée de Fourier



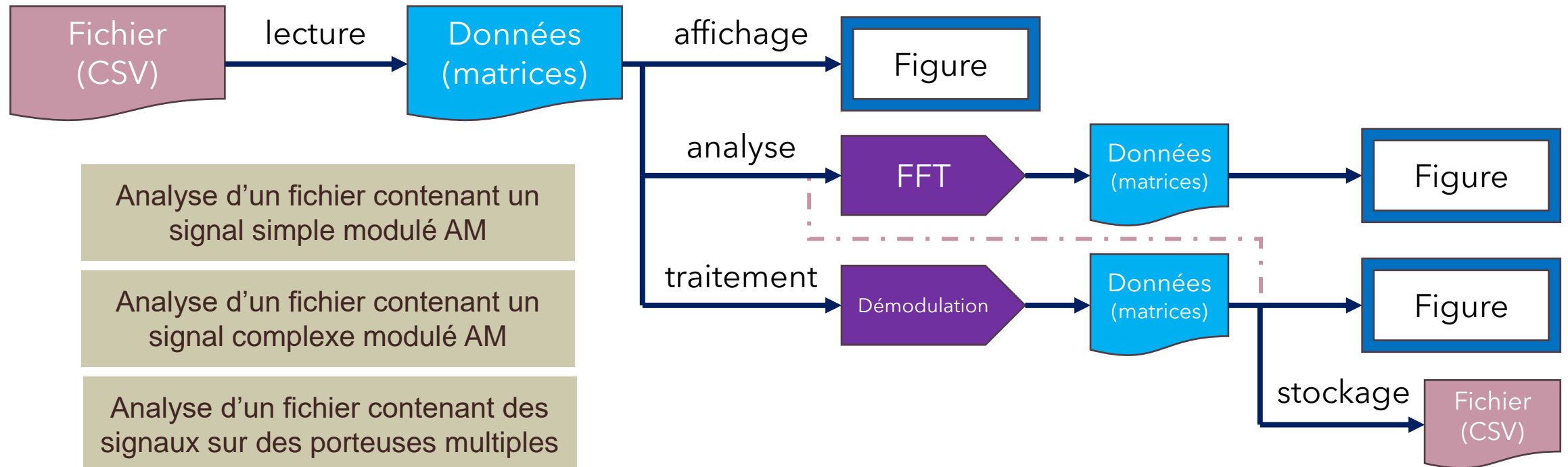
Données initiales / Démarche

Fichier
(CSV)

Etapes pour l'analyse



Etapes pour l'analyse



Travail à réaliser

- **Etape 1 : Afficher des données provenant d'un fichier**
 - Lire un fichier texte / tableur
 - Afficher les signaux contenus dans le fichier
- **Etape 2 : Calculer, afficher et analyser le spectre du signal**
 - Comprendre les données obtenues par le calcul
 - Afficher le spectre en recréant les axes fréquentiels
- **Etape 3 : Simuler le phénomène de modulation d'amplitude et sa démodulation**
 - Générer des signaux de tests et valider les étapes de démodulation
- **Etape 4 : Démoduler un signal quelconque**

Quelques fonctions intéressantes

- lire des fichiers CSV
 - **numpy** `.genfromtxt`
 - **pandas** `.read_csv`
- créer de vecteurs / matrices
 - **numpy** `.linspace` `.logspace`
 - **numpy** `.ones` `.zeros`
- afficher des figures
 - **pyplot** `.figure` `.plot` `.title` `.xlabel` `.ylabel` `.legend`
- calculer la FFT
 - **numpy** `.fft.fft` `.fft.fftshift`
- transcodage / Numpy types
 - **numpy** `.frombuffer` `.astype`
- encodage B64
 - **base64** `.b64encode` `.b64decode`
- encodage WAV
 - **scipy.io** `.wavfile.read` `.write`

Fichiers à analyser

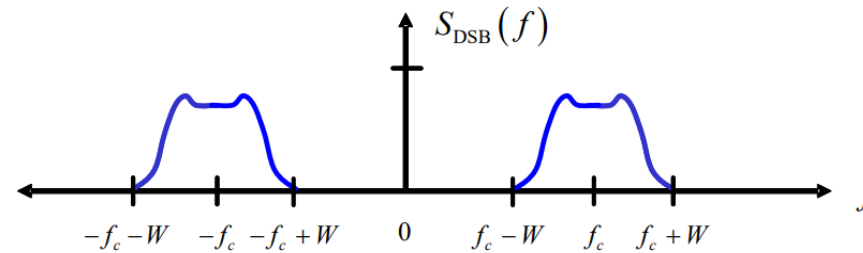
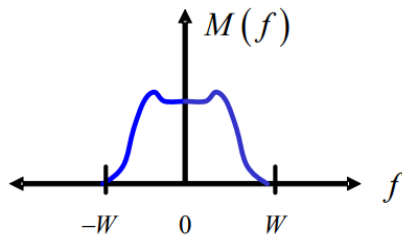
- B3_data_01.csv
 - Issu d'un oscilloscope VoltCraft
 - Modulante sinusoïdale
- B3_data_02.txt
 - Format de données binaire 64
 - Modulante sinusoïdale
 - Fichier sonore / 24 kHz / 16 bits
- B3_data_03.txt
 - Format de données binaire 64
 - Multi-porteuses sinusoïdales
 - Fichier sonore / 160 kHz / 16 bits

<http://lense.institutoptique.fr/ONIP/>

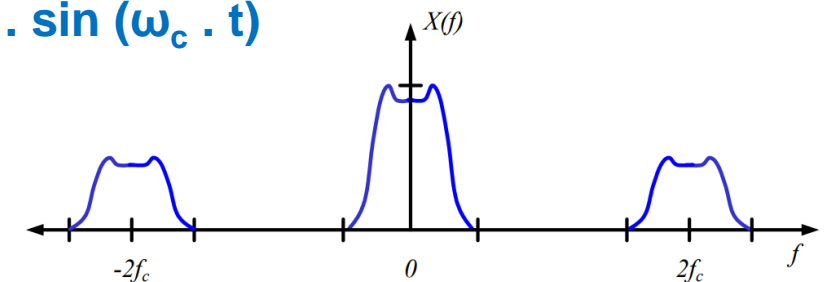
Rappels sur la modulation d'amplitude

<http://wcours.gel.ulaval.ca/2017/a/GEL3006/default/5notes/index.shtml>

$$m(t) \times p(t) = A_c \cdot \sin(\omega_c \cdot t)$$



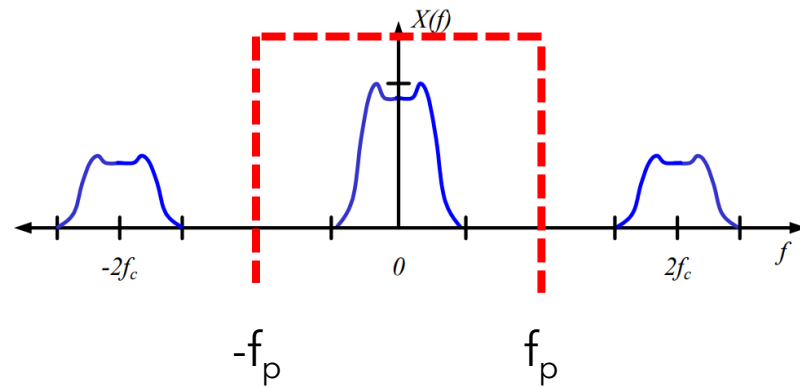
$$\times p(t) = A_c \cdot \sin(\omega_c \cdot t)$$



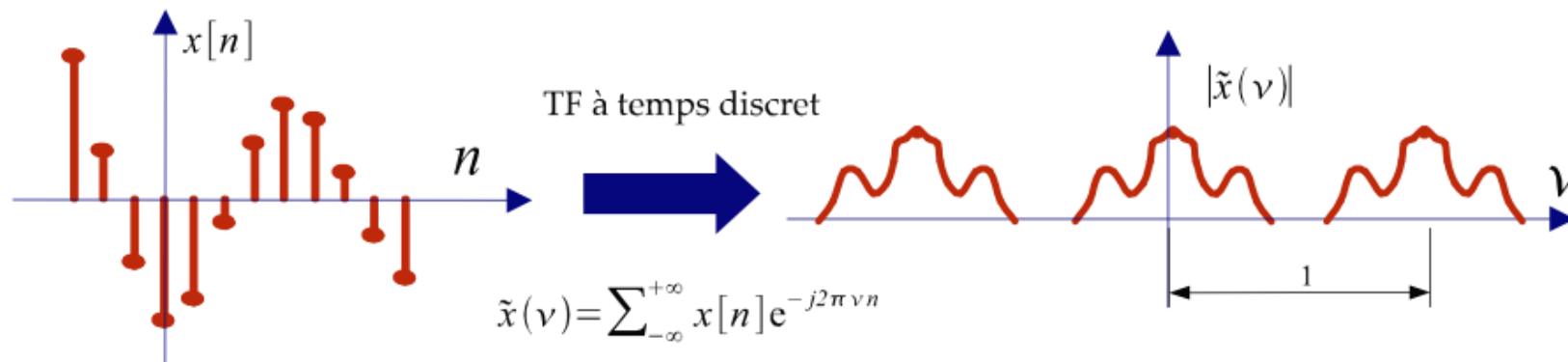
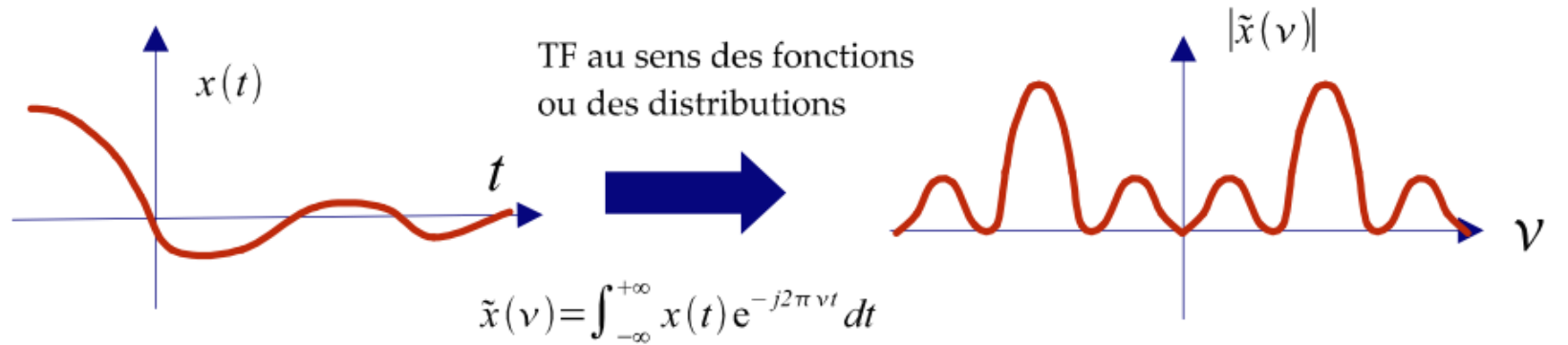
$$s(t) = (K \cdot m(t) + 1) \cdot p(t)$$

Filtrage

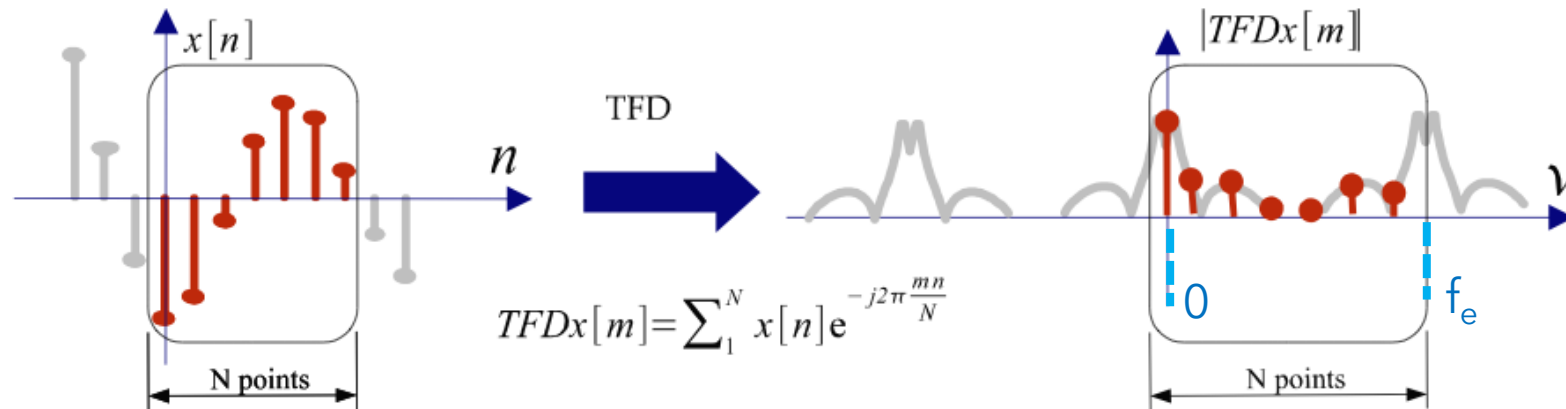
<http://wcours.gel.ulaval.ca/2017/a/GEL3006/default/5notes/index.html>



Rappel sur la Transformée de Fourier



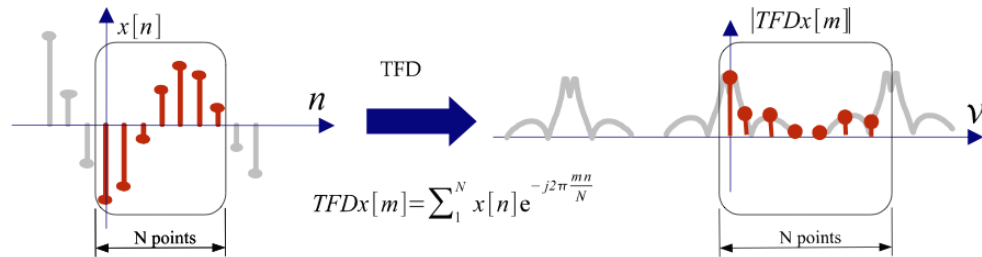
Rappel sur la FFT



`numpy .fft.fft .fft.fftshift`

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\} \quad k = 0, \dots, n-1.$$

Rappel sur la FFT



numpy .fft.fft .fft.fftshift

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\} \quad k = 0, \dots, n-1.$$

- If **A = fft(a, n)**, then **A[0]** contains the zero-frequency term
- Then **A[1:n/2]** contains the positive-frequency terms,
- and **A[n/2+1:]** contains the negative-frequency terms
- For an **even number** of input points, **A[n/2]** represents both positive and negative Nyquist frequency,
- For an **odd number** of input points, **A[(n-1)/2]** contains the largest positive frequency, while **A[(n+1)/2]** contains the largest negative frequency.

Format Binaire 64

- Codage ASCII
 - 1 caractère codé sur 8 bits / 1 octet
- Codage entier
 - 1 entier sur 4 octets
- Codage Base 64
 - 1 donnée sur 6 bits : 4 données sur 3 octets

base64 .b64encode .b64decode

```
import base64
```

```
encoded = base64.b64encode(b'data to be encoded')  
b'ZGF0YSB0byBiZSBpbmNvZGVk'
```

```
data = base64.b64decode(encoded)  
b'data to be encoded'
```

Conversion en signaux sonores

The Canonical WAVE file format

endian	File offset (bytes)	field name	Field Size (bytes)	
big	0	ChunkID	4	The "RIFF" chunk descriptor
little	4	ChunkSize	4	
big	8	Format	4	
big	12	Subchunk1 ID	4	The Format of concern here is "WAVE", which requires two sub-chunks: "fmt" and "data"
little	16	Subchunk1 Size	4	
little	20	AudioFormat	2	
little	22	NumChannels	2	The "fmt" sub-chunk
little	24	SampleRate	4	
little	28	ByteRate	4	
little	32	BlockAlign	2	
little	34	BitsPerSample	2	
big	36	Subchunk2 ID	4	The "data" sub-chunk
little	40	Subchunk2 Size	4	
little	44	data	Subchunk2Size	

The "RIFF" chunk descriptor

The Format of concern here is "WAVE", which requires two sub-chunks: "fmt" and "data"

The "fmt" sub-chunk

describes the format of the sound information in the data sub-chunk

The "data" sub-chunk

Indicates the size of the sound information and contains the raw sound data

scipy.io `.wavfile.read` `.write`

