Outils Numériques pour l'Ingénieur.e en Physique

Bloc AM / Fichiers de données

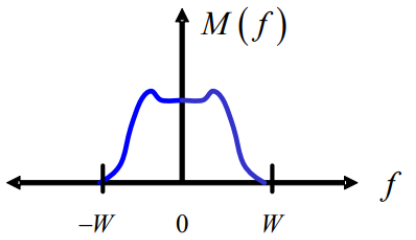
Rappel sur la modulation d'amplitude

Afin de faciliter le transport de signaux électriques (i.e. permettre le transport spécifique de plusieurs informations sur un canal de transmission), on utilise de la **modulation**. La plus facile à mettre en œuvre est la **modulation d'amplitude** (AM).

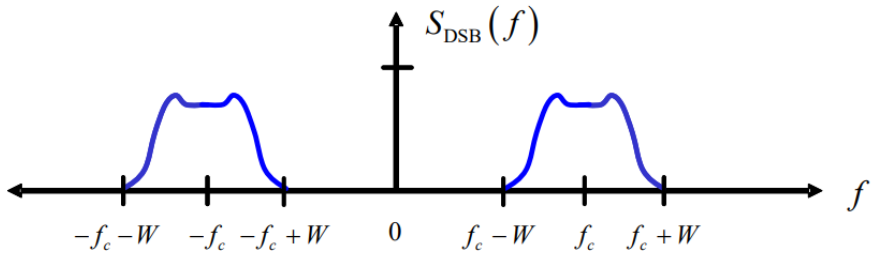
Elle consiste à moduler l'amplitude d'un signal porteur *p(t)* par un signal modulant *m(t)*.
Dans le cas de signaux sinusoïdaux, on a : **m(t)** un signal quelconque de pulsation maximale ω_m et $\mathbf{p(t) = A_p \, . \, sin \, (\omega_p \, . \, t)}$ avec $\omega_p >> \omega_m$

On obtient alors le signal modulé ***s(t) = m(t) . p(t)***.
Dans le cas des GBF Agilent, le signal modulé en sortie est du type : ***s(t) = (K . m(t) + 1) . p(t)*** où K est le taux de modulation.

Dans le cas de signaux périodiques quelconques, dont on connaît le spectre, on obtient alors le spectre suivant après modulation (tiré de http://wcours.gel.ulaval.ca/2017/a/GEL3006/default/5notes/index.chtml) :

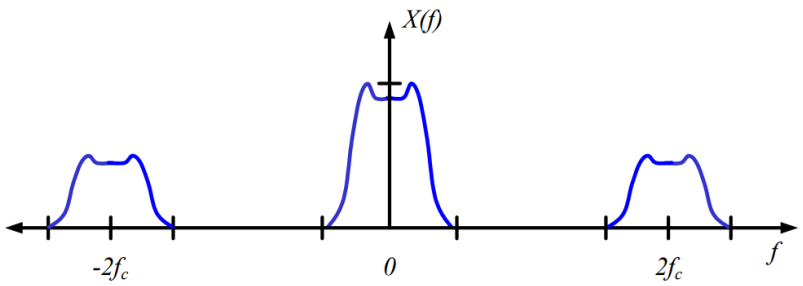


spectre initial

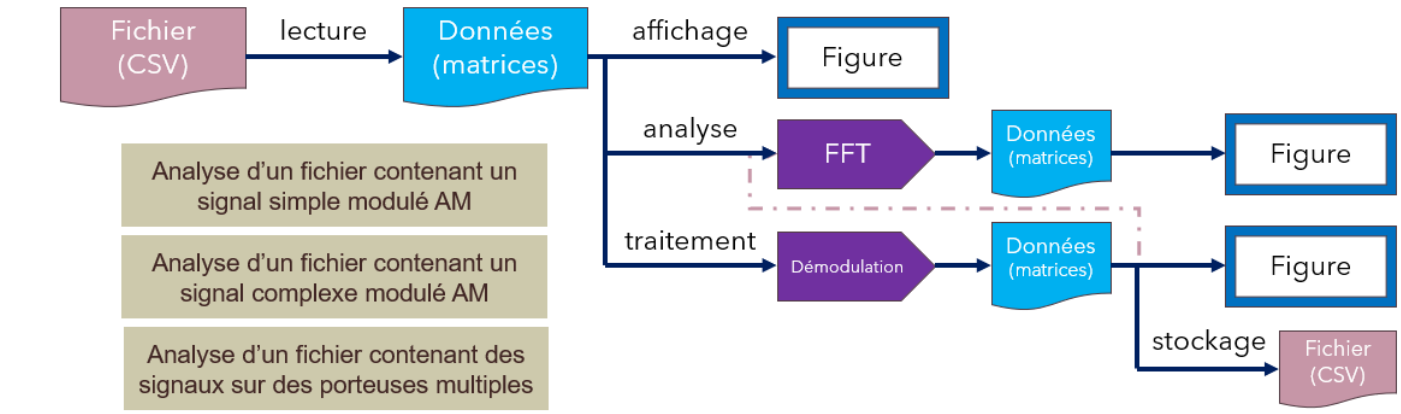


spectre après une modulation d'amplitude avec une porteuse de fréquence f_c

La **démodulation** d'un tel signal se fait en multipliant le signal modulé par la porteuse.
Ainsi : ***d(t) = s(t) . p(t)*** et on obtient le spectre résultant suivant (avec fc la fréquence de la porteuse). Il suffit alors de filtrer la partie centrale du spectre pour retrouver le signal modulé *m(t)*.



Travail demandé



INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Fonctions à maîtriser

- **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 .fftshift .ifft .fftfreq**
- **autres** — **size, numpy.abs, .shape …**
- **transcodage / Numpy types** — **numpy .frombuffer .astype**
- **encodage B64** — **base64 .b64encode .b64decode**
- **encodage WAV** — **scipy.io .wavfile.read .wavfile.write**

Exemple d'encodage et décodage en base 64

```
import base64
encoded = base64.b64encode(b'data to be encoded')
        >> b'ZGF0YSB0byBiZSBlbmNvZGVk'
data = base64.b64decode(encoded)
        >> b'data to be encoded'
```

Exemple de lecture d'un fichier binaire encodé en base 64

```
with open('B3_data_03.txt', 'rb') as file_to_decode :
    binary_file_data = file_to_decode.read()
    data = np.frombuffer(base64.b64decode(binary_file_data), np.int16)
```