Transformée de Fourier discrète : transformée de Fourier

1. Transformée de Fourier

Ce document introduit la transformée de Fourier discrète (TFD) comme moyen d'obtenir une approximation numérique de la transformée de Fourier d'une fonction.

Soit un signal $u(t)$ (la variable t est réelle, les valeurs éventuellement complexes). Sa transformée de Fourier(TF) est :

$$S(f) = \int_{-\infty}^{\infty} u(t) \exp(-j2\pi ft)\, dt$$

Si $u(t)$ est réel, sa transformée de Fourier possède la parité suivante :

$$S(-f) = S(f)^*$$

Le signal s'exprime avec sa TF par la transformée de Fourier inverse :

$$u(t) = \int_{-\infty}^{\infty} S(f) \exp(j2\pi ft)\, df$$

Lors du traitement numérique d'un signal, on dispose de $u(t)$ sur une durée T, par exemple sur l'intervalle $[-T/2, T/2]$. D'une manière générale, un calcul numérique ne peut se faire que sur une durée T finie. Une approximation de la TF est calculée sous la forme :

$$S_a(f) = \int_{-T/2}^{T/2} u(t) \exp(-j2\pi ft)\, dt$$

Soit un échantillonnage de N points, obtenu pour $0 \leq k \leq N-1$:

$$\begin{aligned} t_k &= -\frac{T}{2} + k\frac{T}{N} \\ u_k &= u(t_k) \end{aligned}$$

Une approximation est obtenue par la méthode des rectangles :

$$S_a(f) \simeq \frac{T}{N} \exp(j\pi fT) \sum_{k=0}^{N-1} u_k \exp(-j2\pi kf\frac{T}{N})$$

On recherche la TF pour les fréquences suivantes, avec $0 \leq n \leq N-1$:

$$f_n = \frac{n}{T}$$

c'est-à-dire :

$$S_a(f_n) = T \exp(j\pi n) \frac{1}{N} \sum_{k=0}^{N-1} u_k \exp\left(-j2\pi k\frac{n}{N}\right)$$

En notant S_n la transformée de Fourier discrète (TFD) de u_k, on a donc :

$$S_a(f_n) \simeq T \exp(j\pi n) S_n$$

Dans une analyse spectrale, on s'intéresse généralement au module de $S(f)$, ce qui permet d'ignorer le terme $exp(j\pi n)$

Le spectre obtenu est par nature discret, avec des raies espacées de $1/T$. C'est donc le spectre d'un signal périodique de période T. Pour simuler un spectre continu, T devra être choisi très grand par rapport à la période d'échantillonnage.

Le spectre obtenu est périodique, de périodicité $fe = N/T$, la fréquence d'échantillonnage.

2. Signal à support borné

2.a. Exemple : gaussienne

On choisit T tel que $u(t) = 0$ pour $|t| > T/2$. Considérons par exemple une gaussienne centrée en $t = 0$:

$$u(t) = \exp\left(-\frac{t^2}{a^2}\right)$$

dont la transformée de Fourier est

$$S(f) = a\sqrt{\pi} \exp(-\pi^2 a^2 f^2)$$

En choisissant par exemple $T = 10a$, on a $|u(t)| < 10^{-10}$ pour $t > T/2$

Chargement des modules et définition du signal :

```
import math
import numpy as np
from matplotlib.pyplot import *
from numpy.fft import fft
a=1.0
def signal(t):
    return math.exp(-t**2/a**2)
```

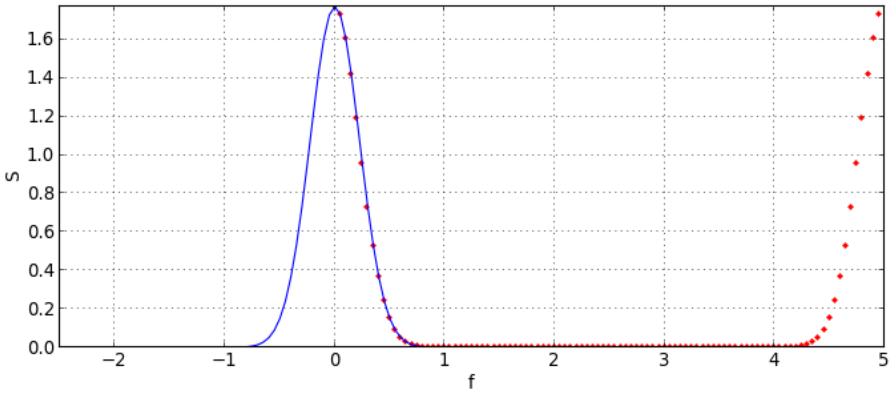
La fonction suivante trace le spectre (module de la TFD) pour une durée T et une fréquence d'échantillonnage fe :

```
def tracerSpectre(fonction,T,fe):
    t = np.arange(start=-0.5*T,stop=0.5*T,step=1.0/fe)
    echantillons = t.copy()
    for k in range(t.size):
        echantillons[k] = fonction(t[k])
    N = echantillons.size
    tfd = fft(echantillons)/N
    spectre = T*np.absolute(tfd)
    freq = np.zeros(N)
    for k in range(N):
```

```
        freq[k] = k*1.0/T
    plot(freq,spectre,'r.')
    xlabel('f')
    ylabel('S')
    axis([0,fe,0,spectre.max()])
    grid()
    return tfd
```

Voyons le spectre de la gaussienne obtenue avec la TFD superposée au spectre théorique :

```
T=20.0
fe=5.0
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
def fourierSignal(f):
    return np.sqrt(math.pi)*np.exp(-math.pi**2*f**2)
f = np.arange(start=-fe/2,stop=fe/2,step=fe/100)
spectre =np.absolute(fourierSignal(f))
plot(f,spectre,'b')
axis([-fe/2,fe,0,spectre.max()])
```



L'approximation de la TF pour une fréquence négative est donnée par :

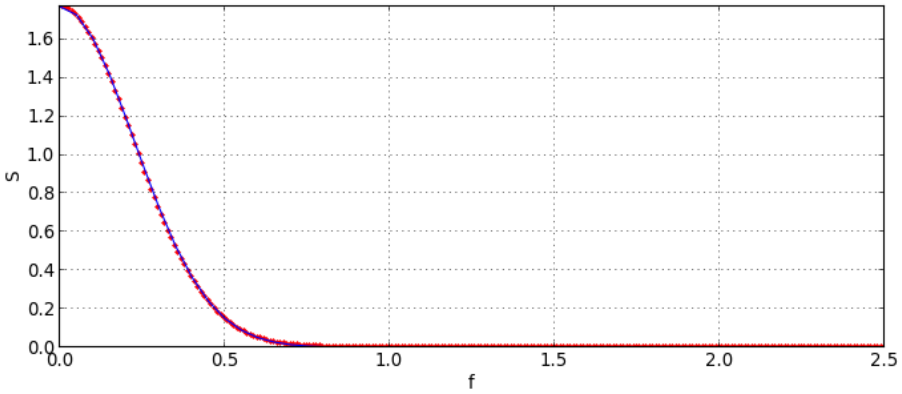
$$S_a(-f_n) \simeq T \exp(-j\pi n) S_{N-n}$$

La seconde moitié de la TFD ($f \in [f_e/2, fe]$) correspond donc aux fréquences négatives. Lorsque les valeurs du signal sont réelles, il s'agit de l'image de la première moitié (le spectre est une fonction paire). Dans ce cas, l'usage est de tracer seulement la première moitié $f \in [0, f_e/2]$.

Pour augmenter la résolution du spectre, il faut augmenter T. Il est intéressant de maintenir constante la fréquence d'échantillonnage :

```
T=100.0
fe=5.0
```

```
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
f = np.arange(start=-fe/2,stop=fe/2,step=fe/100)
spectre =np.absolute(fourierSignal(f))
plot(f,spectre,'b')
axis([0,fe/2,0,spectre.max()])
```



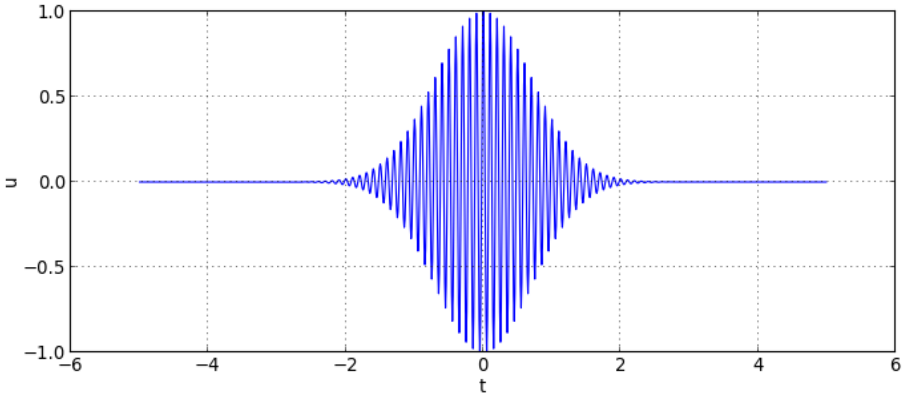
2.b. Exemple : sinusoïde modulée par une gaussienne

On considère le signal suivant (paquet d'onde gaussien) :

$$u(t) = \exp(-t^2/a^2)\cos(2\pi\frac{t}{b})$$

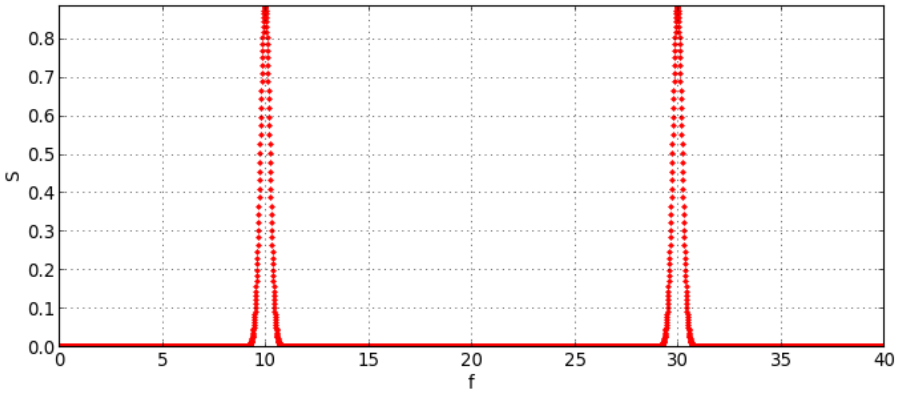
avec $b \ll a$.

```
a=1.0
b=0.1
def signal(t):
    return np.exp(-t**2/a**2)*np.cos(2.0*math.pi*t/b)
t = np.arange(start=-5,stop=5,step=0.01)
u = signal(t)
figure(figsize=(10,4))
plot(t,u)
xlabel('t')
ylabel('u')
grid()
```



Dans ce cas, il faut choisir une fréquence d'échantillonnage supérieure à 2 fois la fréquence de la sinusoïde, c.a.d. $fe > 2/b$.

```
T=100.0
fe=40
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
```



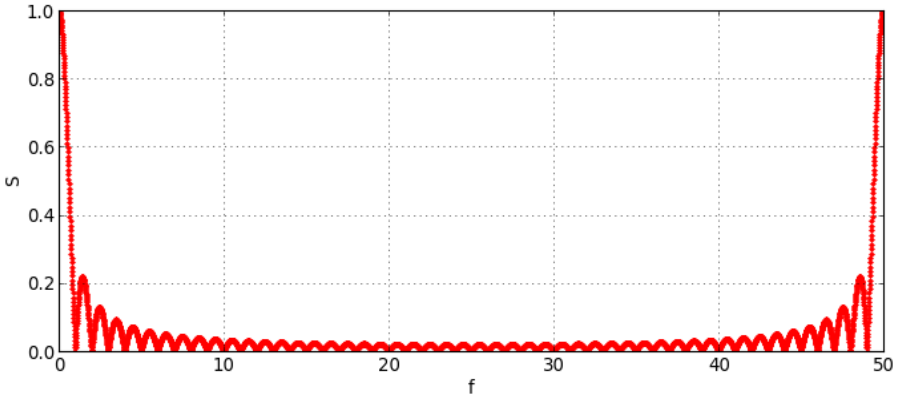
2.c. Fenêtre rectangulaire

Soit une fenêtre rectangulaire de largeur a :

```
a=1.0
def signal(t):
    if (abs(t) > a/2):
        return 0.0
    else:
        return 1.0
```

Son spectre :

```
T=100.0
fe=50
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
```



Une fonction présentant une discontinuité comme celle-ci possède des composantes spectrales à haute fréquence encore non négligeables au voisinage de $fe/2$. Le résultat du calcul est donc certainement affecté par le repliement de bande.

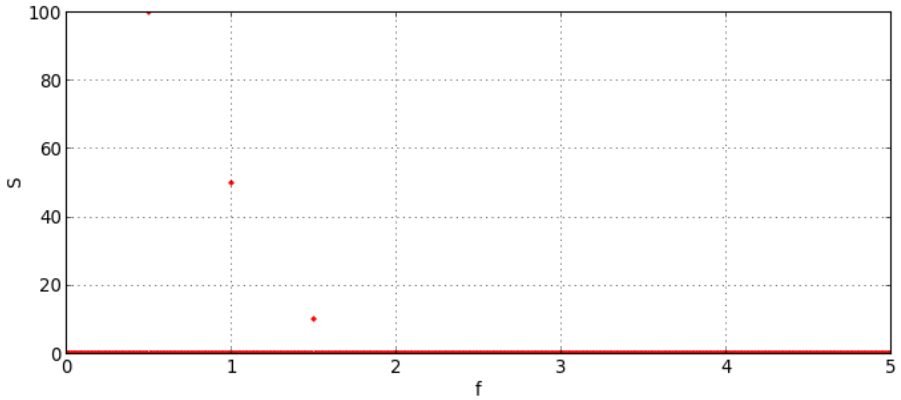
3. Signal à support non borné

Dans ce cas, la fenêtre $[-T/2, T/2]$ est arbitrairement imposée par le système de mesure. Par exemple sur un oscilloscope numérique, T peut être ajusté par le réglage de la base de temps. Considérons par exemple un signal périodique comportant 3 harmoniques :

```
b = 1.0 # periode
w0=1*math.pi/b
def signal(t):
    return math.cos(w0*t)+0.5*math.cos(2*w0*t)+0.1*math.cos(3*w0*t)
```

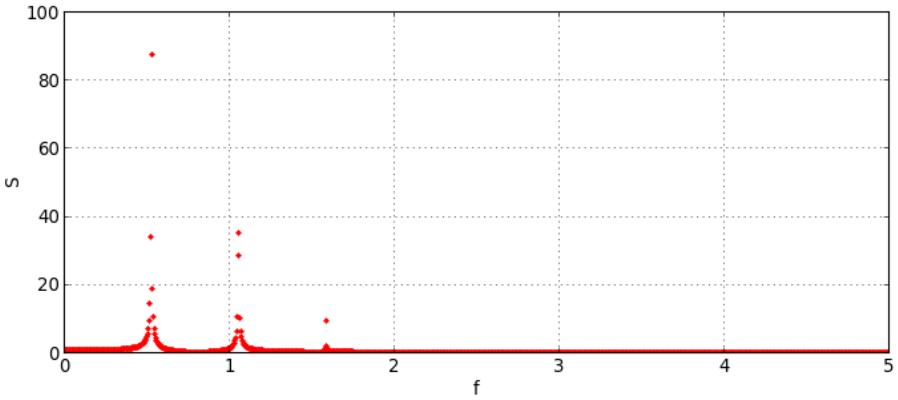
La fréquence d'échantillonnage doit être supérieure à $6/b$ pour éviter le repliement de bande. La durée d'analyse T doit être grande par rapport à b pour avoir une bonne résolution :

```
T=200.0
fe=8.0
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
axis([0,5,0,100])
```



On obtient une restitution parfaite des coefficients de Fourier (multipliés par T). En effet, lorsque T correspond à une période du signal, la TFD fournit les coefficients de Fourier, comme expliqué dans Transformée de Fourier discrète : série de Fourier. En pratique, cette condition n'est pas réalisée car la durée d'analyse est généralement indépendante de la période du signal. Voyons ce qui arrive pour une période quelconque :

```
b = 0.945875 # periode
w0=1*math.pi/b
T=200.0
fe=8.0
figure(figsize=(10,4))
tracerSpectre(signal,T,fe)
axis([0,5,0,100])
```



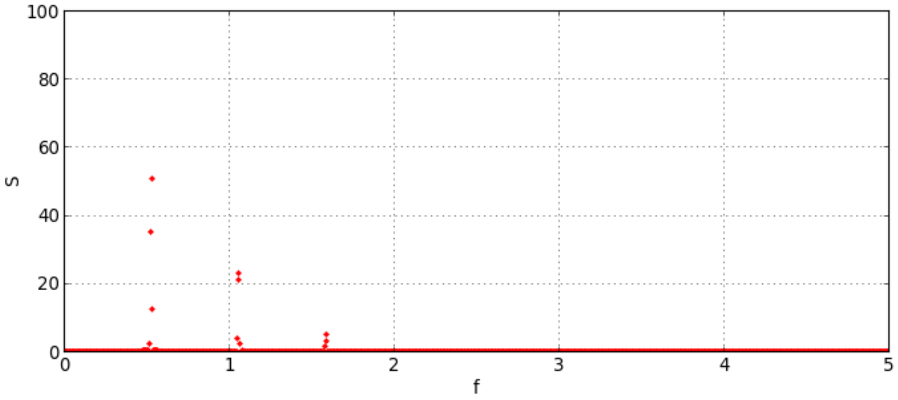
On constate un élargissement de la base des raies. Le signal échantillonné est en fait le produit du signal périodique défini ci-dessus par une fenêtre $h(t)$ rectangulaire de largeur T. La TF est donc le produit de convolution de S avec la TF de h :

$$H(f) = T\frac{\sin(\pi T f)}{\pi T f}$$

qui présente des oscillations lentement décroissantes dont la conséquence sur le spectre d'une fonction périodique est l'élargissement de la base des raies.

Pour remédier à ce problème, on remplace la fenêtre rectangulaire par une fenêtre dont le spectre présente des lobes secondaires plus faibles, par exemple la fenêtre de Hamming :

```
def hamming(t):
    return 0.54+0.46*math.cos(2*math.pi*t/T)
def signalHamming(t):
    return signal(t)*hamming(t)
T=200.0
fe=8.0
figure(figsize=(10,4))
tracerSpectre(signalHamming,T,fe)
axis([0,5,0,100])
```



On obtient ainsi une réduction de la largeur des raies, qui nous rapproche du spectre discret d'un signal périodique.