Chapitre 2

Échantillonnage et Analyse spectrale des signaux

Échantillonnage

- **Observation du signal**

Pour des raisons pratiques, il est impossible d'observer un signal pour toutes les valeur de *t*. On observe donc uniquement le signal entre ***t0*** et ***t1***. On définit alors la durée d'observation : **D = *t1* – *t0*.**

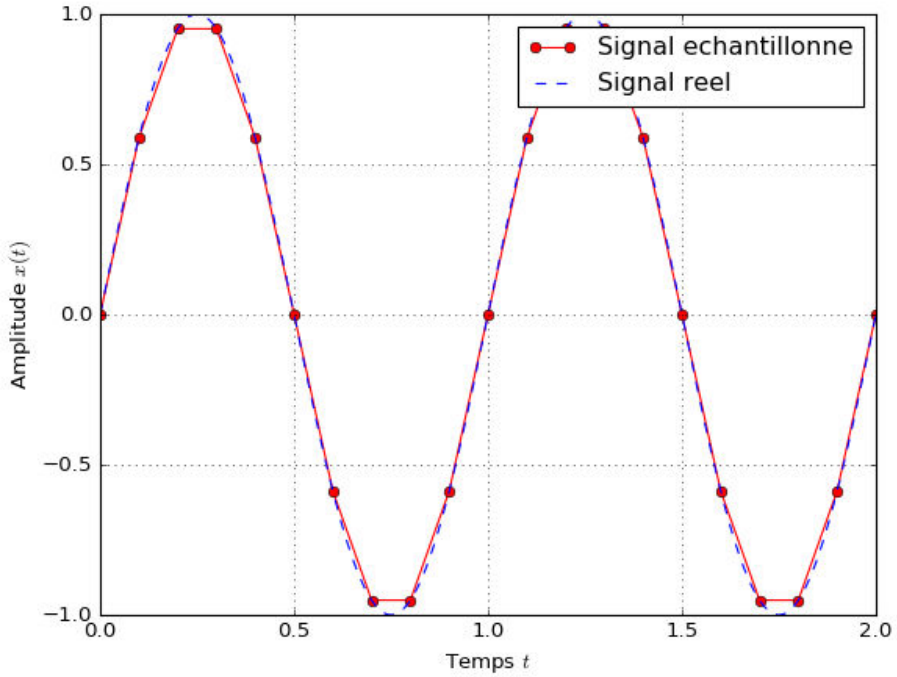
Principe d'échantillonnage:

Pour enregistre un signal dans un format numérique, on mesure ses valeurs sur une grille de points [***ti***] avec i∈[0,N−1], c'est **l'échantillonnage**. On enregistre donc ***N* valeurs**. Une grande quantité d'information est donc perdue par ce procédé. On définit aussi la fréquence d'échantillonnage :

$$f_e = \frac{N-1}{D}$$

Échantillonnage

La figure ci-dessous représente un signal enregistré toutes les 2s. On remarque les points rouges décrivent bien la forme du signal réel, l'échantillonnage est donc réussi.



Échantillonnage

Code Python

```
import numpy as np
import matplotlib.pyplot as plt
# Signal
T = 1.
def  signal(t):  return np.sin(2. * np.pi * t / T)
# Echantillonnage
D = 2. # Duree d'observation
fe = 10. # Frequence d'echantillonnage
N = int(D * fe) + 1 # Nombre de points enregistres
te = np.linspace(0., (N-1)/fe, N) # Grille d'echantillonnage
tp = np.linspace(0., D, 1000) # Grille plus fine pour tracer l'allure du signal parfait
# Trace du signal
plt.plot(te, signal(te), 'or-', label = u"Signal echantillonne")
plt.plot(tp, signal(tp), 'b--', label = u"Signal reel")
plt.grid()
plt.xlabel("Temps $t$")
plt.ylabel("Amplitude $x(t)$")
plt.legend()
plt.show()
```

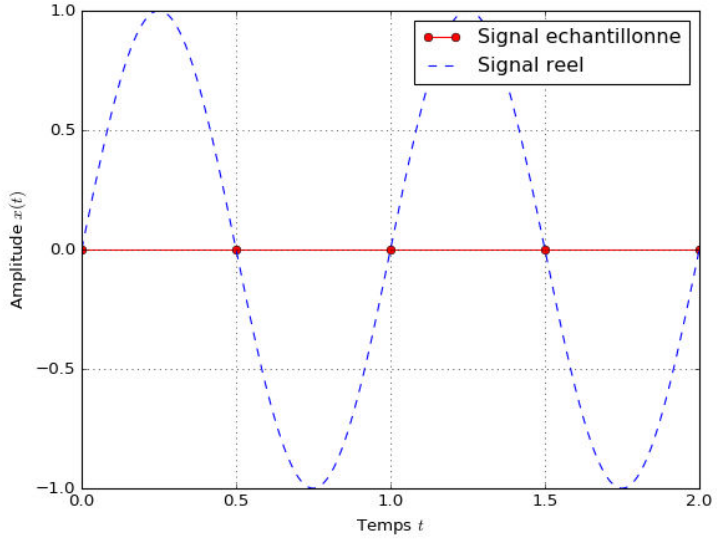
Échantillonnage

- **Théorème de Shannon-Nyquist**

Le **théorème de Shannon Nyquist** indique que la fréquence *f* du signal (ou celles de ses composantes) doit vérifier:

$$f \prec \frac{f_e}{2}$$

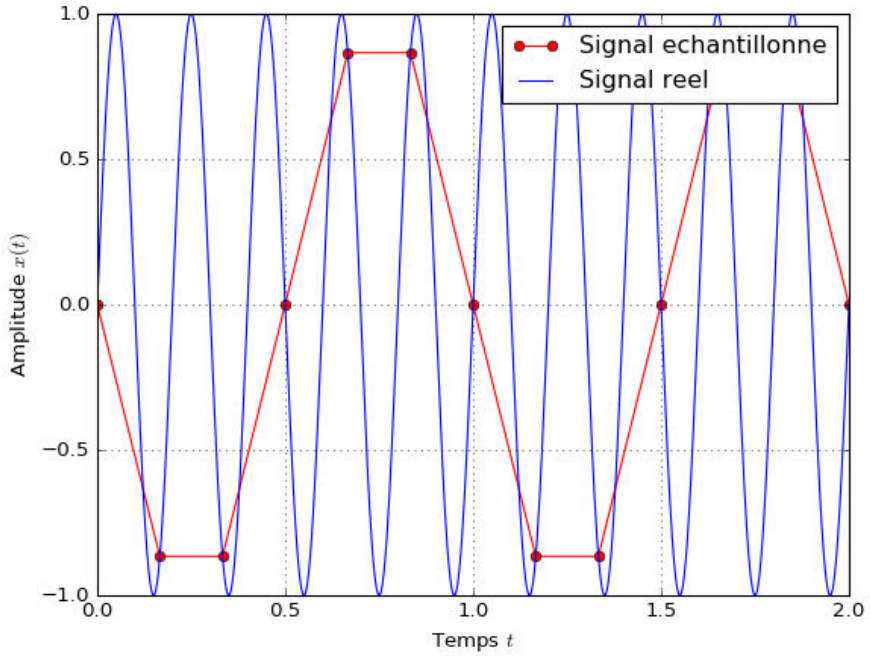
Le cas extrême où *f = fe/2* est représenté ci-dessous.



Échantillonnage

- **Repliement de spectre**

La figure ci-dessus laisse penser que si la fréquence du signal ne respecte pas **le théorème de Shannon-Nyquist**, alors le signal est perdu lors de l'échantillonnage. En réalité, le signal apparait comme ayant une fréquence différente comme le montre la figure ci-dessous.



Échantillonnage

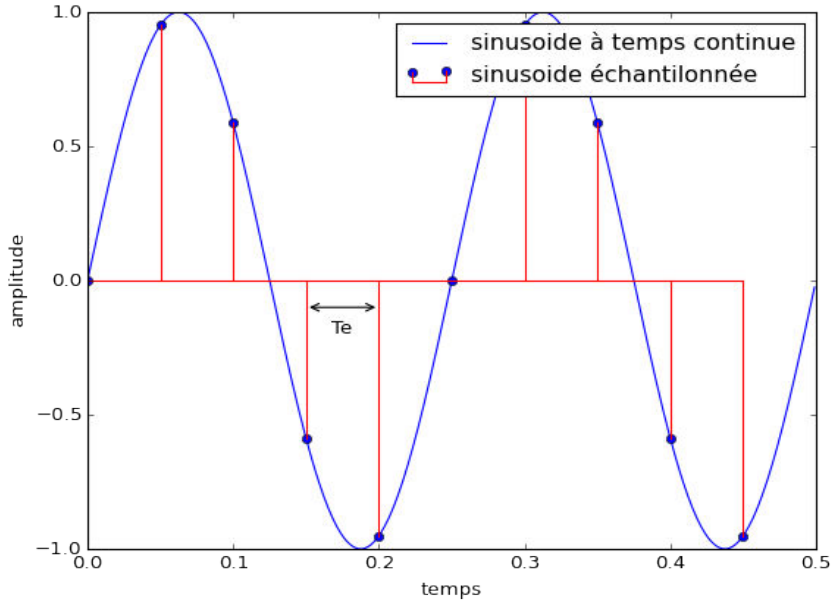
Pour échantillonner un signal, il est donc essentiel de retirer préalablement les composantes de fréquence à l'aide d'un filtre anti repliement.

$$f_e \succ 2 f_{\max}$$

La fréquence *fmax* étant la plus grande valeur des fréquences du signal. Dans le cas d'un signal périodique, elle est égale à l'inverse de sa période (fmax = 1/T).

Échantillonnage

Lorsque nous travaillons avec des outils numériques, la base temps t n'est pas continue mais discrète. La façon la plus simple de discrétiser la base temps consiste à prélever un échantillon du signal continu tous les Te secondes. Le nombre ***Te*** est appelé période d'échantillonnage. La fréquence d'échantillonnage ***Fe*** est définie par: $F_e = \frac{1}{T_e}$



Échantillonnage

Sous Numpy, il est possible de générer facilement une version discretisée de la base temps en utilisant les instructions:

```
from numpy import *

Fe=100  #Fréquence d'échantillonnage.
t=arange(0,1,1/Fe)  #vecteur allant de 0 à 1 par pas de Te=1/Fe.
```

Question

Générez un signal sinusoidal échantillonné ayant comme paramètres a=1, phi=0 et f0=10. La fréquence d'échantillonnage sera fixée à Fe=1000 Hz. Affichez ensuite le signal avec Matplotlib.

Avec :

$$x(t) = a\sin(2\pi f_0 t + \phi)$$

Échantillonnage

En modifiant votre programme précédent, complétez le tableau suivant.

Fe	Fréquence f0	Période T0 réelle	Période mesurée
1000Hz	10 Hz		
1000Hz	500 Hz		
1000Hz	990 Hz		

1. La deuxième ligne du tableau illustre la valeur limite du ratio Fe/f0. Lorsque Fe=2f0, l'échantillonnage n'est pas capable de "capturer" le mouvement de la sinusoïde.
2. La dernière ligne du tableau montre que la fréquence de la sinusoïde est repliée lorsque Fe<2f0. Nous parlons alors de **repliement du spectre** (aliasing).

Ces résultats sont formalisés par le **théorème de Shannon**. $f_e \succ 2 f_{\max}$

A titre d'exemple, comme la fréquence maximale perceptible par l'oreille est d'environ fmax=20kHz, la fréquence d'échantillonnage utilisée dans les supports numériques audios est généralement égale à 44.1KHz.

Décomposition en série de Fourier

Fourier a démontré qu'il était possible de décomposer un signal périodique de fréquence **f0** en une somme de plusieurs sinusoïdes de fréquence **kf0**. Mathématiquement, n'importe quel signal périodique ***s(t)*** peut donc s'exprimer sous la forme :

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(2\pi\, kf_0 t) + \sum_{k=1}^{\infty} b_k \sin(2\pi\, kf_0 t)$$

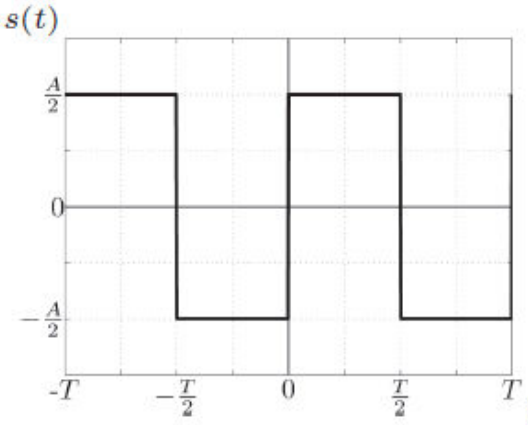
$$a_0 = \frac{1}{T}\int_T s(t)dt$$

$$a_k = \frac{2}{T}\int_T s(t)\cos(2\pi\, kf_0 t)dt$$

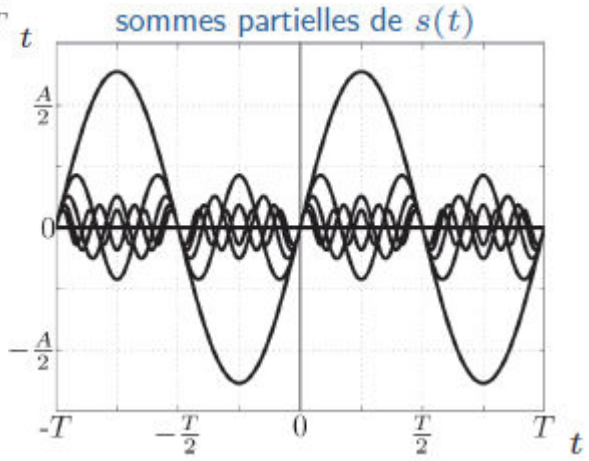
$$b_k = \frac{2}{T}\int_T s(t)\sin(2\pi\, kf_0 t)dt$$

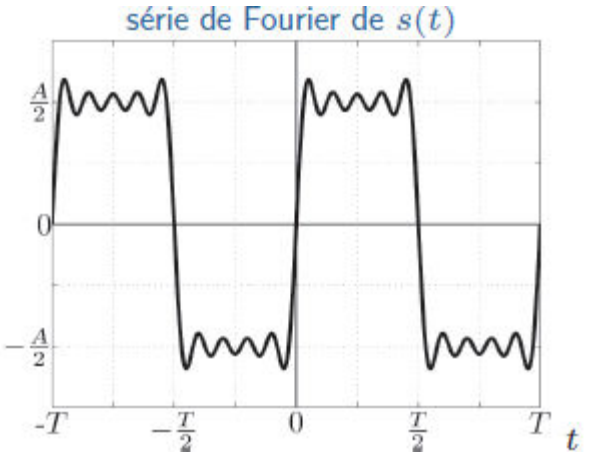
Décomposition en série de Fourier

Signal carrée:



S(t)	ak	b1	b2	b3	b4	b5	b6	b7	b8	b9
Carré	0	1	0	1/3	0	1/5	0	1/7	0	1/9





Décomposition en série de Fourier

Code Python

```
from pylab import *
numpy import *
f0=10
Fe=1000;
coef_b=[1,0,1/3,0,1/5,0,1/7,0,1/9]
Nombre_harmoniques = len(coef_b)
t=arange(0,0.3,1/Fe)
N=len(t)
x = zeros((N,Nombre_harmoniques))
s=zeros(N)
for indice in range(Nombre_harmoniques):
  # L'ensemble des harmoniques
  x[:,indice] =
coef_b[indice]*sin(2*pi*f0*(indice+1)*t)
  #Signal Reconstruit par la Somme Partielle de la
série de Fourier
s=s+coef_b[indice]*sin(2*pi*f0*(indice+1)*t)
```

```
fig=figure(1)
fig.patch.set_alpha(0.0)
subplot(121)
plot(t,x,label= "Ensemble des
harmoniques")
xlabel('temps (s)')
ylabel('amplitude')
show()
subplot(122)
plot(t,s,label="Signal reconstruit")
xlabel('temps (s)‘)
ylabel('amplitude')
show()
```

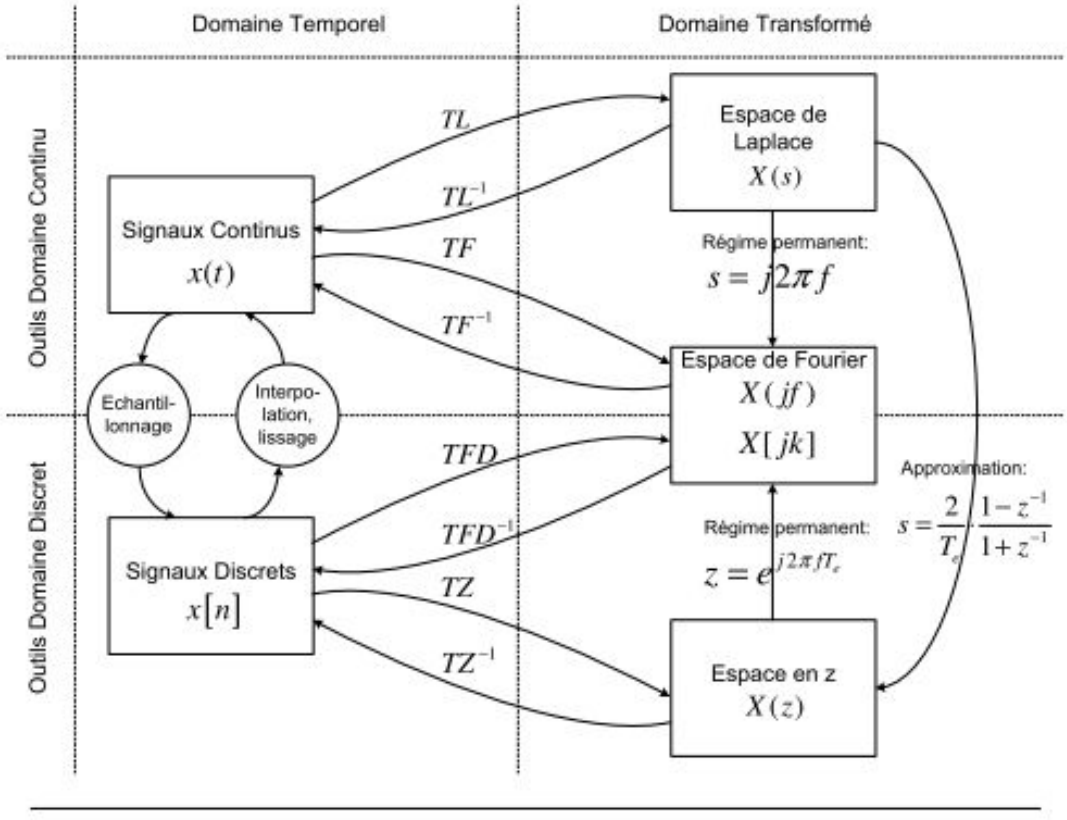
Analyse Spectrale

- **Principe**

L' analyse spectrale d'un signal consiste à construire son spectre, c'est-à-dire sa décomposition sous forme d'une somme de fonctions périodiques. Plusieurs outils existent selon le type de signal étudié. Dans la pratique, nous travaillons toujours avec des signaux apériodiques échantillonnés, l'outil de base pour construire le spectre est la Transformé de Fourier Discrète (DFT) ou son implémentation rapide, la FFT (Fast Fourier Transform). En python, le moyen le plus simple pour accéder aux algorithmes de FFT est la librairie **scipy**. L'algorithem FFT impose que le nombre d'échantillon N soit une puissance de 2.

Analyse Spectrale

- Espace des transformées



Analyse Spectrale

- **La transformée de Fourier (Signaux continus)**

La transformée de Fourier (TF) est très utilisée pour l'analyse des signaux. Cette transformée permet de mettre en valeur des éléments de notre signal difficilement visualisables dans le domaine temporel.

La TF des signaux continus se calcule par à la variable temps (t). C'est-à-dire l'information est disponible à chaque instant du temps. Mathématiquement parlant on dit que la fonction (signal) est continue sur tout l'intervalle d'intégration.

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-j\omega t} dt, \qquad avec \quad \omega = 2\pi f$$

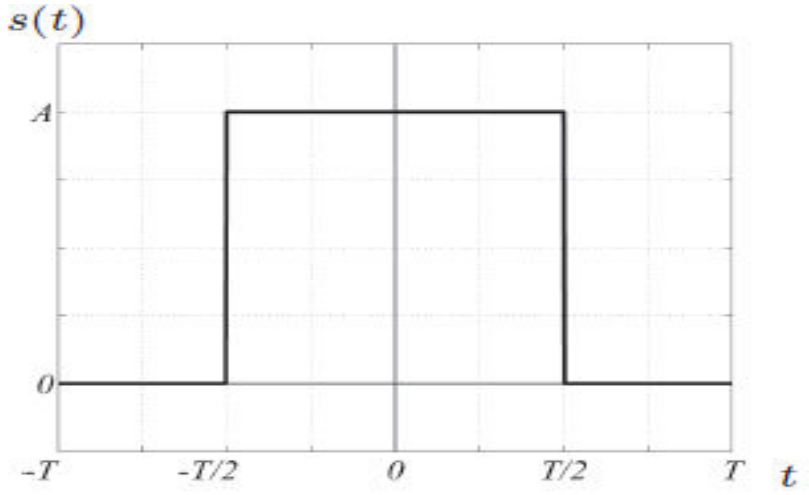
$S(f)$ est la TF$\{s(t)\}$ appelée le spectre du signal temporel $s(t)$.

Analyse Spectrale

Exemple

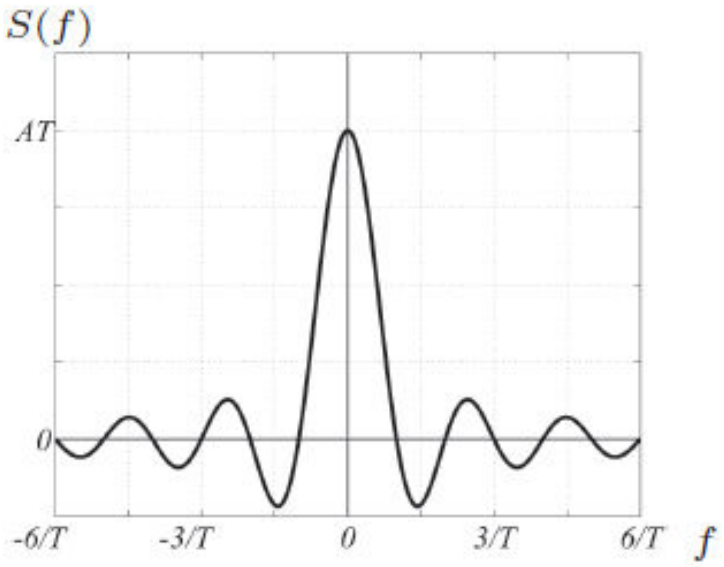
- Signal porte s(t) = A.rect (t/T) (impulsion rectangulaire de largeur T et d'amplitude A) défini sur $\mathbb{R}$ par :

$$s(t) = A\,rect\left(\frac{t}{T}\right) = \begin{cases} A & si \quad \frac{-T}{2} \le t \le \frac{T}{2} \\ 0 & ailleurs \end{cases}$$



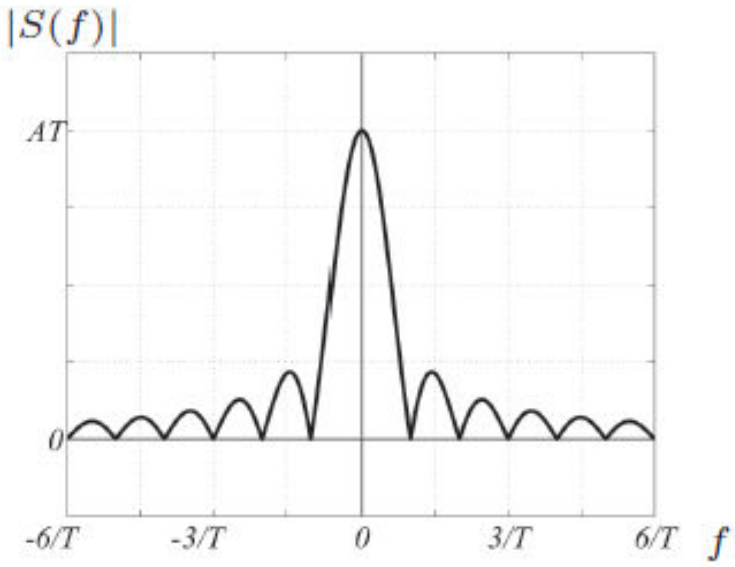
Analyse Spectrale

Spectre en fréquence : le spectre complexe du signal s(t) est une fonction réelle. Sa partie imaginaire est donc nulle et son module est égale à la valeur absolue de sa partie réelle.



spectre réel

$$S(f) = AT\,\mathrm{sinc}\,(Tf).$$



spectre d'amplitude

$$|S(f)| = |AT\,\mathrm{sinc}\,(Tf)|.$$