Transformée de fourier

La technique usuelle pour déterminer les fréquences caractéristiques ainsi que l'amplitude des harmoniques qui composent un signal, consiste à calculer sa transformée de fourier. Dans une approche numérique, cette opération est effectuée sur un ensemble de valeurs échantillonnées.

Le but de l'exercice consiste à programmer deux méthodes de calcul de la transformée de fourier à partir des algorithmes fournis.

1. Principe de l'algorithme :

* On entre d'abord le nombre de points d'échantillonnage *N*.

* On procède ensuite à l'acquisition des *N* points d'échantillonnage de la fonction dont on veut calculer la transformée. Dans le cas présent, les valeurs échantillonnées seront calculées par l'intermédiaire de la procédure *init* et stockées dans un tableau complexe double précision s de taille N.

* Après initialisation du tableau complexe double précision *ft_s* de taille N dans lequel on va stocker la transformée de fourier. Cette dernière est estimée en calculant la somme discrète suivante :

$$ft\_s(k) = \sum_{n=0}^{N-1} s(n) \exp\left(-i\frac{2\pi k}{N}n\right) = \sum_{n=0}^{N-1} s(n) \exp(-i\omega_k n), \qquad (1)$$

où $i^2 = -1$ avec $\omega_k = \dfrac{2\pi k}{N}$, l'indice k variant de 0 à *N-1*.

La transformée inverse s'écrit :

$$s(n) = \sum_{k=0}^{N-1} ft\_s(k) \exp\left(+i\frac{2\pi k}{N}n\right) = \frac{1}{N}\sum_{k=0}^{N-1} ft\_s(k) \exp(+i\omega_k n) \qquad (2)$$

* On sauvegarde finalement dans un fichier, pour toutes les valeurs de k, la partie réelle et la partie complexe de *ft_s(k)* grâce à la procédure *sauve*.

Cas test :

$$s(n) = \sin\left(\frac{2\pi}{N}n\right) = \frac{1}{2i}\left(\exp\left(+i\frac{2\pi}{N}n\right) - \exp\left(-i\frac{2\pi}{N}n\right)\right) \text{ par définition.} \qquad (3)$$

Dans la limite où N est suffisamment grand (> 2^{10} = 1024), les effets de taille finie deviennent négligeables. La transformée de fourier de s(n) met en évidence les deux modes principaux suivants :

pulsation	amplitude
$\omega_1 = \dfrac{2\pi}{N}$	$ft\_s(1) = -\dfrac{N}{2}i$
$\omega_{N-1} \equiv \omega_{-1} = -\dfrac{2\pi}{N}$	$ft\_s(N-1) = +\dfrac{N}{2}i$

En appliquant la relation (2), on vérifie que le résultat correspond bien à (3).

2. phase de programmation

Votre travail consiste à développer deux variantes de programme en traduisant les deux algorithmes de transformée de fourier fournis ci-après.

INDICATIONS :

1. Pour vous aider, les indices des tableaux commencent à 0 comme en python et non à 1 comme le voudrait la norme algorithmique.

2. Le nombre complexe i est prédéfini en python et se note 1j

3. Les opérateurs arithmétiques (+-*/) sont applicables sur les nombres complexes ainsi que les fonctions mathématiques courantes. Il faut importer les modules math et cmath pour calculer des expressions de la forme :
omega=cmath.exp(-math.pi*1j)

4. Les fonctions z.real et z.imag calculent respectivement la partie réelle et la partie imaginaire de z. La norme est donnée par abs(z).

Algorithme de la transformée de fourier :

```
/*******************************************************************************/
            Calcul de la transformée de fourier ft
Entrée
entier  32 bits       N           :     taille du tableau ci-après
tableau complexe 2x64 bits  s     :     tableau des valeurs échantillonnée de la
fonction

Sortie
tableau complexe 2x64 bits  ft_s  :     tableau des valeurs de la tranformée de
fourier
/*******************************************************************************/

procédure fourier(E/ entier N, tableau complexe s;
                  S/ tableau complexe ft_s)

   complexe omega, alpha

   pour k=0 à N-1
       ft_s(k) ← 0.
       omega ← exp(-2*pi*I/N*k) /* voir remarque ci-dessous */
       alpha ← 1.

       pour n=0 à N-1
             ft_s(k) ← ft_s(k) + s(n)*alpha
             alpha ← alpha*omega
       fin pour n
   fin pour k

fin procédure
```

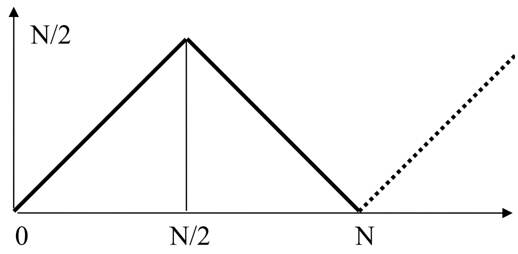
3. Algorithme récursif de la transformée de fourier rapide :

```
/*********************************************************************************/
            Calcul de la transformée de fourier rapide fft

Entrée
entier  32 bits       n            :     taille du tableau ci-après
ATTENTION n doit s'exprimer sous la forme d'une puissance de 2 !
tableau complexe 2x64 bits  a      :     tableau des valeurs échantillonnée de la
fonction

Sortie
tableau complexe 2x64 bits  fft_a  :     tableau des valeurs de la tranformée de
fourier
/*********************************************************************************/

procédure fft_rec(E/ entier n, tableau complexe a;
                  S/ tableau complexe fft_a)

   entier i
   complexe omega ← exp(-2.*pi*I/n)
   si n = 1 alors
       fft_a(0) ← a(0)
   sinon
       entier m ← n/2
       tableau complexe b[0..m-1] /* indice tableaux commence à 0 */
       tableau complexe c[0..m-1]
       tableau complexe fft_b[0..m-1]
       tableau complexe fft_c[0..m-1]
      complexe alpha ← 1.

       pour i=0 à m-1
             b(i) = a(2*i)
             c(i) = a(2*i+1)
       fin pour i

   /* appel récursif de fft_rec */
       fft_rec(m, b, fft_b)
       fft_rec(m, c, fft_c)

       pour i=0 à m-1
             fft_a(i)   ← fft_b(i) + alpha*fft_c(i)
             fft_a(m+i) ← fft_b(i) - alpha*fft_c(i)
             alpha ← alpha*omega;
       fin pour i
   fin si
fin procédure
```

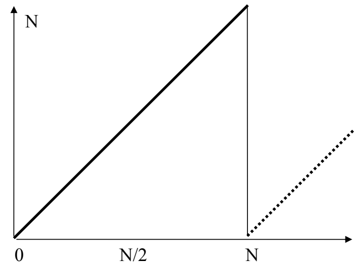
3. phase de tests A FAIRE

Proposer 2 cas tests simples supplémentaires pour vérifier le bon fonctionnement des deux programmes. Quel est l'intérêt principal de la méthode fft ?

Vérifier par programmation que les composantes de la fonction chapeau de période N ci-contre sont $C_0 = \frac{N^2}{4}$, pour n≠0 pair $C_n = 0$ et pour n impair $C_n = -\frac{N^2}{\pi^2 n^2}$.



Dans le cas de la fonction créneau de période N ci-contre, le calcul des composantes de Fourier donne pour $C_0 = \frac{N^2}{2}$, pour n≠0 $C_n = i\frac{N^2}{2\pi n}$.



Fournir pour chaque cas la routine *init* correspondante, ainsi que les valeurs des parties réelle et imaginaire calculées par votre programmation pour n∈ [0, 5]. Remarques.

On notera que

$$\sum_{n=0}^{N-1} \exp\left(-i\frac{2\pi k}{N}n\right) = \frac{1-\exp(-i2\pi k)}{1-\exp\left(-i\frac{2\pi k}{N}\right)} = 0 \text{ , autrement écrit } \sum_{\mathbf{n=1}}^{N-1} \exp\left(-i\frac{2\pi k}{N}n\right) = -1$$