TD IMSP1 : TRANSFORMEE DE FOURIER DISCRETE

PARTIE A :
TRANSFORMÉE DE FOURIER DISCRÈTE … PAS RAPIDE DU TOUT !
(qu'on pourrait appeler VSFT : Very Slow Fourier Transform !)

Les bibliothèques à importer seront numpy (renommée np), matplot.pyplot (renommée plt)

I. La Transformée de Fourier Discrète d'un signal numérique

I.1. Définition

Soit un signal échantillonné et « fenêtré » correspondant à un tableau U de N valeurs (a priori complexes) numérotées de n=0 à N-1, la transformée de Fourier discrète de ce signal correspond également à un tableau TFU de N valeurs (a priori complexes) numérotées de k=0 à N-1 calculées de la manière suivante :

$$\mathrm{TFU}[k] = \sum_{n=0}^{N-1} \mathrm{U[n]}.e^{-j2k\pi.\frac{n}{N}}$$

qui, dans nos cas de signaux réels, peut se décomposer en partie réelle et partie imaginaire :

$$\mathrm{TFU}[k] = \sum_{n=0}^{N-1} \mathrm{U[n]}.\cos(-2k.\pi.\frac{n}{N}) + j.\sum_{n=0}^{N-1} \mathrm{U[n]}.\sin(-2k.\pi.\frac{n}{N})$$

Mais, à l'instar des calculs de FFT, nous ne souhaitons retourner que les amplitudes des harmoniques (perte de l'information sur la phase) soit les modules de ces valeurs complexes.

I.2. Fonctions décomposant le calcul de la TFD

I.2.1. Module d'un complexe

Histoire de commencer piano piano, écrire une fonction **module(a,b)** retournant le module du complexe a+jb. (a et b seront nécessairement réels à l'appel de cette fonction)

I.2.2. Parties réelle et imaginaire du terme k de la TFD

Ecrire une fonction **trans_fourier_freq(U,k)** retournant la partie réelle et la partie imaginaire de la Transformée de Fourier pour une valeur de k.

Combien d'opérations sont effectuées pour la détermination d'un terme d'ordre k ?

En supposant que les fréquences soient monotones croissantes de k, quelle fréquence correspond à k=N-1 ? A quelle durée est associé l'incrément fréquentiel (et donc l'incrément unité de k) ?

I.2.3. Transformée de Fourier discrète

Ecrire une fonction **trans_fourier(U)** retournant le module de la Transformée de Fourier discrète sous la forme d'un tableau de taille N contenant les modules des N valeurs complexes TFU[k] (on appellera naturellement les fonctions précédentes)

Déterminer la complexité du calcul de la transformée de Fourier par cette fonction **trans_fourier(U)** en fonction de N et conclure sur l'optimisation de l'algorithme basique utilisé.

II. Test de l'algorithme sur un signal en « dents de scie »

Nous utiliserons la bibliothèque graphique déjà importée (pyplot) et nous ajoutons le module time pour tester la lenteur de l'algorithme en fonction de la taille N des tableaux de valeurs.

II.1. Le pseudo-signal numérique « Dents de scie » et sa TF

Ecrire une fonction **dents_de_scie_basique(E,p,T,Te)** dont les arguments sont respectivement E la valeur max de la tension, p le nombre (flottant) de périodes du signal dans la fenêtre temporelle, T la période du signal dents de scie et Te la période d'échantillonnage. Cette fonction doit retourner le temps de calcul, les représentations graphiques de DSc (tableau contenant les N valeurs instantanées de tension) fonction du temps et TFDSc (tableau des N valeurs d'amplitude de la TF discrète appliquée au signal dent de scie) fonction de la fréquence.
On utilisera numpy (np.arange) pour créer la listes des N instants t de mesure et DSc les valeurs de tension associées : rampe croissante du temps modulo E. On procédera de la même façon pour le tableau des fréquences et on utilisera **trans_fourier(U)**

Les tableaux de variables DSc et TFDSc seront des variables globales pour pouvoir les scruter.

On testera :

a- dents_de_scie_basique(5,10,1/1000,1/20000)
b- dents_de_scie_basique(5,20.48,1/1000,1/100000)

Commentaire sur le temps d'exécution
Commentaire sur les amplitudes de la TF
Commentaire sur l'interpolation entre les points des graphes

II.2. Réorganisation de la liste des fréquences

Le théorème de Nyquist-Shannon stipule que la fréquence d'échantillonnage doit être supérieure au double la fréquence maximale du signal. (à condition qu'il y en ait une)
On souhaite donc présenter la liste des fréquences sur l'intervalle [-fe/2,fe/2]
(La périodicité de la TF discrète le permet !)

(on appelle fréquence de Nyquist cette demi-fréquence d'échantillonnage)

Nous devons donc réorganiser la liste des fréquences depuis la liste originale :

[0,1/(N.Te),2/(N.Te)…..(N-2)/(N.Te),(N-1)/(N.Te)]

vers une présentation symétrique :

- si N est pair : [0,1,2,…(N/2)-1,-(N/2),-(N/2)+1,…,-1].(1/N.Te)
- si N est impair : [0,1,…,(N-1)/2,-(N-1)/2,-(N-1)/2 +1,…,-1].(1/N.Te)

Ecrire une fonction **freq_fourier(U,Te)** qui renvoie les listes de fréquences réorganisées.
Recopier **dents_de_scie_basique(E,p,T,Te)** dans l'éditeur et le renommer **dents_de_scie_reorga(E,p,T,Te)** en substituant le tableau des fréquences précédentes par le résultat de la fonction **freq_fourier(U,Te)** appliquée à DSc.

Commentaire sur le résultat graphique avec **plt.plot**
Comment avoir des amplitudes relatives de la TF ? Quel pic de référence proposez-vous ?

II.3. Améliorations de la lisibilité graphique

II.3.1. Amplitudes relatives au fondamental

Nous allons donc exprimer les amplitudes relatives (au fondamental) de la transformée de Fourier discrète. [*ce qui implique qu'une amplitude (peut-être supérieure) du continu sera tronquée sur la TFD*]

La fonction sur laquelle nous travaillons depuis le début nécessite l'argument T alors qu'en réalité un appareil numérique ne « connaît » évidemment pas la période du signal étudié dans la fenêtre temporelle.
Décidons d'extraire la valeur max **hors du continu** pour lui donner la valeur 100% d'amplitude. (en espérant qu'il s'agisse bien du fondamental !)
Nous utiliserons la fonction **del** (qui s'applique à une liste) pour éliminer la première valeur de la liste puis nous utiliserons **max** qui renvoie le max d'un tableau de valeurs.
Nous en profiterons donc pour rappeler comment passer d'une liste à un tableau et inversement.

Ecrire la fonction **valeur_du_max_hors_continu(U)** qui retourne la valeur maximale d'un tableau U après en avoir éliminé la première valeur.

II.3.2. Représentation « discrète » et fenêtre multi-graphes

Pour éviter une représentation continue du signal DSc échantillonné, on ne lie plus les points que l'on représente par des croix « + » rouges.
Pour éviter une représentation continue et une interpolation transversale indésirable de la TFDSc, on optera pour une représentation du spectre des amplitudes par des segments verticaux partant de l'axe des fréquences et montant jusqu'à la valeur de l'amplitude relative à cette fréquence. (*plt.vlines*)
On représentera également les deux représentations du signal (temporelle DSc et fréquentielle TFDSc) sur une seule et même figure séparant verticalement les deux graphes sur une même ligne (*subplot*)

Renommer finalement **dents_de_scie(E,p,T,Te)** une version remédiant au souci précédent en utilisant la **valeur_du_max_hors_continu(TFDSc)**

Appliquer au signal b comportant un calcul sur 2048 points :
dents_de_scie(5,20.48,1/1000,1/100000)

Utilisez les boutons de la fenêtre graphique (déplacement, zoom…) pour observer les premiers pics (de 0 à 6 kHz par exemple) dans chacun des deux cas. Commenter.

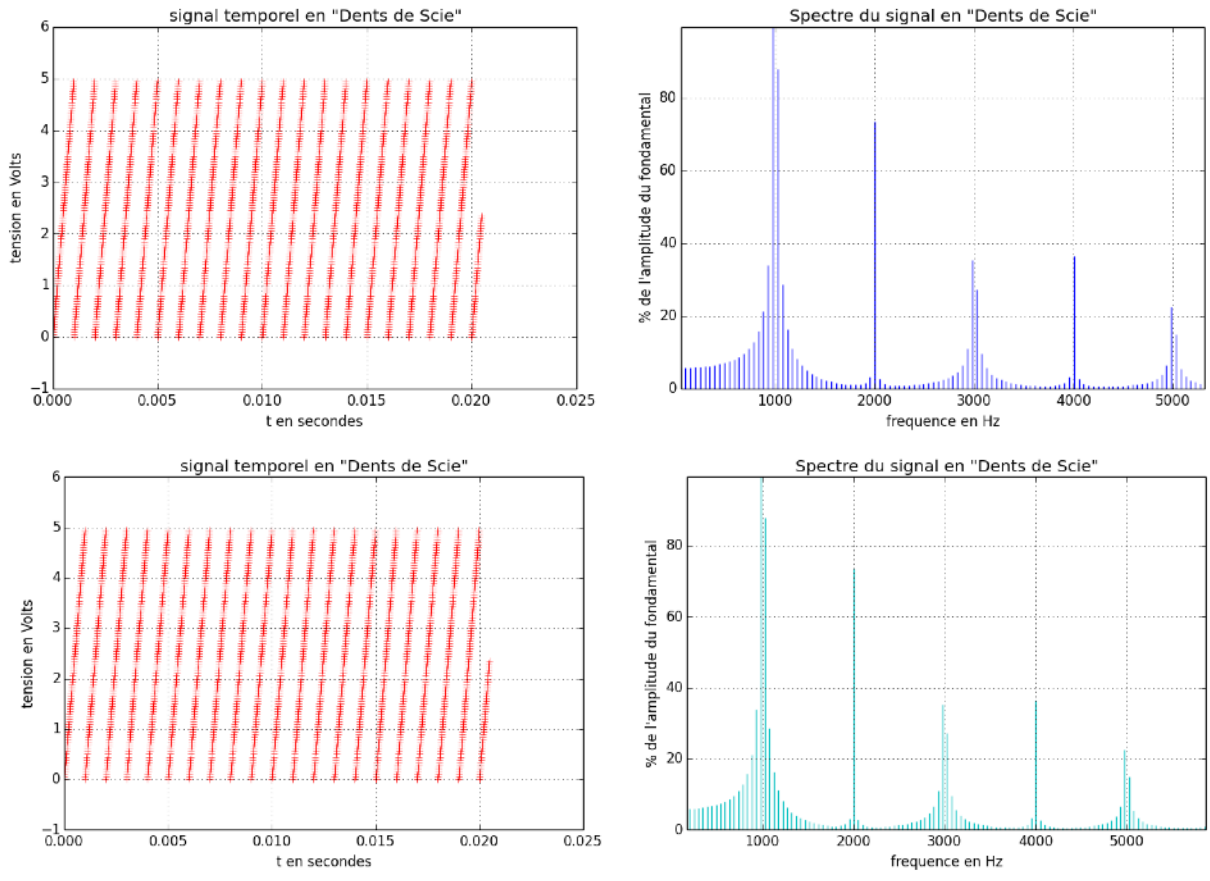
Que penser d'un calcul « en temps réel » avec cet algorithme ?

PARTIE B :
TRANSFORMÉE DE FOURIER DISCRÈTE … RAPIDE !
(FFT : Fast Fourier Transform)

Il est entendu que l'algorithme précédent n'est pas satisfaisant dès que l'on travaille sur des tableaux contenant des milliers de points et pourtant vos oscillos numériques de salle de TP de physique sont bien capables de restituer des FFT en quasiment un dixième de seconde alors qu'ils travaillent sur 1024 ou 2048 points. Quelle particularité remarquez vous sur ces nombre de points ?
Quelles subdivisions peut-on alors envisager ?

I. La FFT par Numpy

Nous allons utiliser **np.fft.fft(DSc)** pour le calcul de la transformée de Fourier discrète du signal en « dents de scie » précédent (valeurs numériques de l'exemple b).
Modifiez donc légèrement **dents_de_scie** et exécutez **dents_de_scie_par_numpy_fft**
Comparez la nouvelle durée d'exécution à la précédente.
Vérifiez (visuellement cela suffira) que **dents_de_scie_par_numpy_fft** donne exactement les mêmes valeurs que **dents_de_scie** (la procédure fft de numpy n'est pas une approximation !)
(Ci-dessous les deux TFD comparées dans la zone de 0 à 5 kHz : en haut la VSFT et en bas la FFT)



Puisque nous avons désormais une FFT, profitez-en pour voir la différence entre le résultat de cette FFT et la DSF attendue pour ce signal s'il était continu et de durée infinie.
Comment évolue l'amplitude des harmoniques dans la DSF d'un signal dents de scie ?
Qu'obtient-on lorsque la fenêtre temporelle contient un nombre entier de périodes ? Et sinon ?
Quel intérêt de calculer la FFT sur une fenêtre contenant de nombreuses périodes ?

II.L'algorithme de Cooley-Tukey

On dit parfois que l'ère du traitement de l'information a commencé le 17 aout 1964 lors de la publication de l'article de Cooley et Tukey :

"An Algorithm for the Machine Calculation of Complex Fourier Series."

II.1.Un calcul récursif basé sur un tableau de longueur N=2^n

Rappelons l'expression de la transformée de Fourier discrète :

$$\mathrm{TFU}[k]=\sum_{n=0}^{N-1}\mathrm{U[n]}.e^{-j2k\pi.\frac{n}{N}}$$

Remarquons qu'elle peut se décomposer en séparant les indices pairs et impairs :

$$\mathrm{TFU}[k]=\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m]}.e^{-j2k\pi.\frac{2m}{N}}+\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m+1]}.e^{-j2k\pi.\frac{(2m+1)}{N}}$$

$$\mathrm{TFU}[k]=\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m]}.e^{-j2k\pi.\frac{2m}{N}}+e^{-\frac{j2k\pi}{N}}.\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m+1]}.e^{-j2k\pi.\frac{2m}{N}}$$

On transforme ainsi un calcul sur un tableau de N valeurs en deux calculs sur des tableaux de N/2 valeurs assortis d'un calcul de l'exponentielle de (-2jkπ/N) et d'une somme. Or chacune de ces deux TF de taille N/2 peut elle même être décomposée en deux TF de taille N/4… et ainsi de suite jusqu'au bout c'est-à-dire jusqu'au calcul d'une TF de taille $N/2^n=1$! ne contenant donc qu'un terme : TF[a]=[a].

On peut également remarquer que le calcul est seulement nécessaire jusqu'à k=N/2. La symétrie observée à partir de la fréquence de Nyquist pouvant être exprimée sous la forme :

$$\mathrm{TFU}[k+\frac{N}{2}]=\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m]}.e^{-j2k\pi.\frac{2m}{N}}-e^{-\frac{j2k\pi}{N}}.\sum_{m=0}^{\frac{N}{2}-1}\mathrm{U[2m+1]}.e^{-j2k\pi.\frac{2m}{N}}$$

Proposer une fonction **exposantFFT(p, q)** qui renvoie la valeur de l'exponentielle complexe $e^{\frac{j2q\pi}{p}}$. (à laquelle la FFT suivante fera bien sûr appel)

Vérifier que la fonction **fftCT(U)** suivante code bien la fonction **récursive** précédente par la décomposition de Cooley-Tukey et retourne le tableau des N termes de la FFT.

```
def fftCT(U):
   N = len(U)
   if N == 1:
      return U
   else:
      FFTpaire = fftCT([U[i] for i in range(0, N, 2)])
      FFTimpaire= fftCT([U[i] for i in range(1, N, 2)])

      TFU = [0]*N
      for k in range(int(N/2)):
         TFU[k] = FFTpaire[k] + exposantFFT(N, -k) * FFTimpaire[k]
         TFU[k + int(N/2)] = FFTpaire[k] - exposantFFT(N, -k) *FFTimpaire[k]
       return np.array(TFU)
```

Modifiez votre **dents_de_scie_par_numpy_fft** en **dents_de_scie_par_fftCT** en substituant **abs(np.fft.fft(DSc))** par **abs(fftCT(DSc))**.

Comparez les temps d'exécution pour :

dents_de_scie_par_numpy_fft(5,20.48,1/1000,1/100000) et

dents_de_scie_par_fftCT(5,20.48,1/1000,1/100000)

Comparez les TF lorsque N n'est plus une puissance de 2. Commentez

II.2.Adaptation possible lorsque N≠2ⁿ

On se propose de modifier les lignes décrivant le signal échantillonné sur sa fenêtre réelle de durée p*T :

listetemps=np.arange(0,p*T,Te)
DSc=(E/T)*listetemps%E

pour effectuer une recopie partielle de la fenêtre temporelle pour atteindre un nombre de points **nouvN** puissance de 2 immédiatement supérieure à N=p*T/Te avant d'exécuter l'algorithme de Cooley-Tukey sur ce pseudo-signal.

On pourra avantageusement tirer parti des méthodes **np.tile** et **np.split** pour répéter et découper les tableaux.

Ce nouveau (et dernier !) programme sera nommé :

dents_de_scie_par_fftCTnonpuissancede2(E,p,T,Te)

Appliquez-le par exemple pour :

dents_de_scie_par_fftCTnonpuissancede2(5,20.5,1/1000,1/100000)

Zoomez entre 0 et 6 kHz. Commentez