FILTRAGE NUMÉRIQUE

Importation des bibliothèques utiles :

```
import numpy as np    # importation de la bibliothèque numpy
import matplotlib.pyplot as plt    # pour tracer des courbes
```

La bibliothèque numpy contient toutes les fonctions utiles `np.cos`, `np.sin`, `np.exp` et la constante π : `np.pi`

I. Représentation temporelle d'un signal

1) Signaux sinusoïdal et créneau

Rédiger deux fonctions python `sinusoide(E,T,t)` et `creneau(E,T,t)`.

- La première renvoie la valeur à l'instant t d'un signal sinusoïdal $u(t) = E\cos(2\pi t/T)$
- La seconde renvoie la valeur à l'instant t d'un signal créneau symétrique définie par :

$$u(t) = \begin{cases} E & \text{si} \quad t \in [\, nT, nT + T/\,2\,[\\ -E & \text{si} \quad t \in [\, nT + T/2, (n+1)T\,[\end{cases}$$

Rédiger un code python permettant de tracer les courbes des deux signaux précédents entre les dates $t_{\min} = 0$ et $t_{\max} = 5$ ms, avec $N = 1000$ points. On prendra $E = 5$V et une fréquence $f = 1000$ Hz.

2) Synthèse de Fourier

La somme de Fourier de rang n d'un créneau symétrique d'amplitude E et de période T est donnée ci-dessous :

$$F_n(t) = \sum_{k=0}^{n} \frac{4E}{(2k+1)\pi} \sin\left(2\pi(2k+1)\frac{t}{T}\right)$$

Rédiger une fonction python `fourier(n,E,T,t)` qui renvoie la somme de Fourier de rang n à l'instant t.

Écrire ensuite un code en langage python permettant de tracer cette somme de Fourier si $E = 5$V, $f = 1000$ Hz, entre les instants $t_{\min} = 0$ et $t_{\max} = 5$ ms, toujours avec $N = 1000$ points.

Vous ferez varier n pour observer l'évolution de la forme du signal.

3) Simulation d'un échantillonnage

Nous allons simuler de façon informatique l'échantillonnage d'un signal analogique $s(t)$, c'est à dire le prélèvement d'une valeur du signal toutes les T_e secondes, où T_e est la **période d'échantillonnage**. Dans le programme rédigé, les valeurs ci-dessous seront fixées et déclarées comme variables globales :

```
f = 1000        # fréquence du signal s(t)
T = 1/f         # période de s(t)
tmin = 0        # début de l'échantillonnage
tmax = 5*T      # fin de l'échantillonnage
```

```
N = 10000       # nombre de points d'échantillonnage
Te = (tmax-tmin)/N
```

Écrire une fonction `liste_ech_signal(s)` prenant en paramètre un signal analogique `s`, c'est à dire une fonction python de la forme `s(E,T,t)` comme par exemple les fonctions `sinusoide` ou `creneau` rédigées au paragraphe 1) et qui renvoie la liste des N échantillons de `s` prélevés entre `tmin` et `tmax`

Nous allons maintenant utiliser cet échantillonnage pour réaliser un filtrage numérique. La liste des N échantillons de $s(t)$ sera notée $(s_n)_{0\leqslant n\leqslant N-1}$

II. Filtrage numérique

1) Action d'un filtre passe-bas d'ordre 1

La fonction de transfert d'un filtre passe-bas d'ordre 1 est :

$$\underline{H}(j\omega) = \frac{1}{1+j\dfrac{\omega}{\omega_c}} = \frac{\underline{s}(t)}{\underline{e}(t)}$$

où ω_c est la pulation de coupure. Lorsqu'on repasse dans le domaine temporel, cela correspond à l'équation différentielle :

$$\underline{s}(t) + \frac{1}{\omega_c}\, j\omega\, \underline{s}(t) = \underline{e}(t) \implies \frac{1}{\omega_c}\,\frac{\mathrm{d}s}{\mathrm{d}t} + s = e$$

Pour des données numériques échantillonnées avec une période T_e, l'idée est de remplacer la dérivée temporelle de la sortie s par le taux d'accroissement :

$$\frac{\mathrm{d}s}{\mathrm{d}t} \longleftrightarrow \frac{s_{n+1}-s_n}{T_e}$$

ce qui permet de transformer l'équation différentielle en :

$$\boxed{\frac{1}{\omega_c}\frac{s_{n+1}-s_n}{T_e} + s_n = e_n \iff s_{n+1} = s_n\,(1-\omega_c T_e) + \omega_c T_e\, e_n}$$

La liste des échantillons de l'entrée $(e_n)_{0\leqslant n\leqslant N-1}$ (telle que renvoyée par la fonction `liste_ech_signal` et T_e étant connues, la liste des échantillons $(s_n)_{0\leqslant n\leqslant N-1}$ de la sortie du filtre se calcule à l'aide de la récurrence ci-dessus, avec une condition initiale pour s_0. On a donc remplacé une équation différentielle par une récurrence.

- Écrire une fonction python `filtre_PB_num(Eech, fc)` qui prend en paramètre une liste `Eech` de N échantillons de l'entrée du filtre ainsi que la fréquence de coupure du filtre et qui renvoie une liste formée des N échantillons de la sortie.
- Tracer la courbe donnant les échantillons s_n en ordonnée en portant en abscisse les instants nT_e correspondants pour les signaux `sinusoide` et `creneau` rédigés au paragraphe **I.1)** Faire varier la fréquence de coupure pour observer son influence sur (s_n).

2) Action d'un filtre passe-haut d'ordre 1

La fonction de transfert d'un filtre passe-haut du premier ordre est :

$$\underline{H}(j\omega) = \frac{j\dfrac{\omega}{\omega_c}}{1 + j\dfrac{\omega}{\omega_c}} = \frac{\underline{s}(t)}{\underline{e}(t)}$$

où ω_c est la pulsation de coupure. Lorsqu'on repasse dans le domaine temporel, cela correspond à l'équation différentielle :

$$\underline{s}(t) + \frac{1}{\omega_c}\, j\omega\, \underline{s}(t) = \frac{1}{\omega_c}\, j\omega\, \underline{e}(t) \quad \Longrightarrow \quad \frac{1}{\omega_c}\,\frac{\mathrm{d}s}{\mathrm{d}t} + s = \frac{1}{\omega_c}\,\frac{\mathrm{d}e}{\mathrm{d}t}$$

qu'on peut ransformer en :

$$\boxed{\frac{1}{\omega_c}\frac{s_{n+1} - s_n}{T_e} + s_n = \frac{1}{\omega_c}\frac{e_{n+1} - e_n}{T_e} \quad \Longleftrightarrow \quad s_{n+1} = s_n\,(1 - \omega_c T_e) + e_{n+1} - e_n}$$

Écrire une fonction python `filtre_PH_num(Eech, fc)` qui prend en paramètre une liste `Eech` de N échantillons de l'entrée du filtre ainsi que la fréquence de coupure du filtre et qui renvoie une liste formée des N échantillons de la sortie.

Tracer à nouveau les courbes représentant la sortie s_n en fonction de nT_e et observer l'influence de la fréquence de coupure `fc`

III. Analyse spectrale numérique

Dans toute cette partie on fera en sorte que l'échantillonnage des signaux se fasse exactement sur une période T. On écrira donc dans le programme :

```
f = 1000        # fréquence du signal
T = 1/f
tmin = 0        # début de l'échantillonnage
tmax = T        # fin de l'échantillonnage
N = 10000       # nombre de points d'échantillonnage
Te = (tmax-tmin)/N      # Te = T/N
```

1) Principe du calcul

Un signal analogique $s(t)$, T-périodique, peut être décomposé en série de Fourier. Aux instants t où il est continu on peut écrire, avec $\omega = 2\pi/T$:

$$s(t) = \frac{A_0}{2} + \sum_{k=1}^{+\infty} A_k \cos(k\omega t) + B_k \sin(k\omega t)$$

avec :

$$A_k = \frac{2}{T}\int_0^T s(t)\cos(k\omega t)\,\mathrm{d}t \quad \text{et} \quad B_k = \frac{2}{T}\int_0^T s(t)\sin(k\omega t)\,\mathrm{d}t$$

la relation donnant A_k étant aussi valable pour $k = 0$. On peut aussi supposer que $B_0 = 0$.

Il est utile d'introduire les coefficients de Fourier complexes $\underline{C}_k = A_k - jB_k$ qui peuvent donc se calculer par :

$$\forall\, k \in \mathbb{N},\ \underline{C}_k = \frac{2}{T}\int_0^T s(t)\, e^{-jk\,\omega t}\,\mathrm{d}t$$

Lorsqu'on ne dispose que d'une liste d'échantillons $(s_n)_{0\leqslant n\leqslant N-1}$ le calcul de cette intégrale se fait grâce à la méthode d'Euler explicite :

$$\underline{C}_k = \frac{2}{T}\sum_{n=0}^{N-1}\int_{nT_e}^{(n+1)T_e} s(t)\,e^{-jk\,\omega t}\,\mathrm{d}t \approx \frac{2}{T}\sum_{n=0}^{N-1} s(nT_e)\,e^{-jk\,\omega\,nT_e}\,T_e$$

Comme $T_e = T/N$, cette expression se transforme en :

$$\boxed{\forall\, k\in\mathbb{N},\ \underline{C}_k = \frac{2}{N}\sum_{n=0}^{N-1} s_n\,e^{-2\pi\,jnk/N}}$$

où s_n est la valeur échantillonnée à l'instant nT_e. Cette somme s'appelle **transformée de Fourier discrète** (c'est celle qui est utilisée aussi dans la transformée de Fourier rapide FFT, grâce à un algorithme qui permet d'augmenter la rapidité du calcul de la somme).

Les amplitudes C_k réelles des pics (amplitude de l'harmonique de rang k) dans le spectre en amplitude de $s(t)$ sont alors données par :

$$\forall\, k\in\mathbb{N}^*,\ C_k = \sqrt{A_k^2+B_k^2} = |\underline{C}_k| \quad \text{et} \quad \frac{A_0}{2} = \frac{\underline{C}_0}{2} = \frac{1}{N}\sum_{n=0}^{N-1} s_n$$

Remarquons que la composante continue du signal $A_0/2$ est alors la **moyenne arithmétique** des N échantillons.

2) Calcul d'un spectre en amplitude

- Écrire une fonction python `coefficient_C(k,Sech)` qui prend en paramètre le rang `k` du coefficient à calculer et une liste `Sech` de N échantillons d'un signal analogique $s(t)$, l'échantillonnage étant supposé être réalisé sur une durée valant exactement une période T de $s(t)$.

  Cette fonction renvoie le coefficient complexe $\underline{C}_k$. La bibliothèque `numpy` dispose de la fonction `np.exp` qui gère aussi les complexes. On rappelle qu'en python, le nombre complexe `j` s'écrit `1j`

- Comme $T = NT_e$, la fréquence f du signal $s(t)$ vaut :

$$f = \frac{1}{T} = \frac{F_e}{N} \quad \text{avec} \quad F_e = \frac{1}{T_e} \ \text{ fréquence d'échantillonnage}$$

  Ainsi, l'amplitude $|\underline{C}_k|$ de l'harmonique de rang k correspond à la fréquence $kf = k\frac{F_e}{N}$.

  Écrire un code python qui permet de tracer le spectre en amplitude de la `sinusoide` et du `creneau` décrits au paragraphe I.1)