TP 14 : Filtrage
Simulation numérique

Les points du programme :
- Étudier le filtrage linéaire d'un signal non sinusoïdal à partir d'une analyse spectrale.
- Capacité numérique : simuler, à l'aide d'un langage de programmation, l'action d'un filtre sur un signal périodique dont le spectre est fourni. Mettre en évidence l'influence des caractéristiques du filtre sur l'opération de filtrage.

Objectifs
- Savoir construire un signal à partir de son spectre.
- Simuler l'action d'un filtre sur ce signal (on se donnera la fonction de transfert du filtre).

Matériel : *Votre ordinateur avec Anaconda (Spyder).*

1. Filtre passe-bas

Récupérer dans le dossier SL3/TP14 du cahier de prépa le fichier *TP14_passebas.py*. L'ouvrir avec Spyder. Ce script est séparé en plusieurs cellules. Vous avez la possibilité d'exécuter chaque cellule séparément grâce à l'icone .

☐ Exécuter la première cellule afin d'importer les bibliothèques `numpy` et `matplotlib.pyplot`.

1.1. Construction du signal d'entrée

On considère un signal d'entrée $e(t)$, périodique, de fréquence f_e (par exemple f_e = 1 kHz). Sa pulsation est $\omega_e = 2\pi.f_e$.
La décomposition de Fourier de ce signal est :

$$e(t) = c_0 + \sum_{n=1}^{+\infty} c_n.\cos(n.\omega_e.t + \varphi_n)$$

Q1. Que représente c_0 ? Comment s'appelle le premier terme de la somme (pour $n = 1$) ? Quelle est sa pulsation et sa fréquence ?
Comment s'appellent les termes pour $n > 1$? Quelle est la pulsation et la fréquence du terme numéro n ?

Q2. On considère le signal harmonique suivant : $e(t) = 0{,}5 + \cos(\omega_e.t)$.
Donner la valeur des coefficients c_n et φ_n pour ce signal.

En Python, on se donne la liste des c_n et celle des φ_n, pour le signal d'entrée $e(t)$:

```
c_e = [0.5,1]
phi_e = [0,0]
```

On souhaite ensuite tracer le signal $e(t)$. On dispose pour cela, dans le code, d'une fonction `signal(f_1,c,phi,t)`, déjà écrite, qui permet d'obtenir le signal $e(t)$. Lire pour cela sa description dans le code.

Pour l'utiliser :
- On crée une liste d'instants `t[i]` avec `t = np.linspace(0,5*Te,1000)` *(cf. code)*.
- On fait l'affectation : `entree = signal(f_e,c_e,phi_e,t)`.
  `entree` est alors une liste qui contient les valeurs de $e(t)$.

☐ Compléter le code pour pouvoir créer la liste `entree`.

☐ Compléter le code pour pouvoir tracer la liste `entree` en fonction de la liste `t`. On légendera l'axe des abscisses ("t (s)") et des ordonnées ("signal"), ainsi que la courbe.

Q3. Le signal tracé est-il cohérent avec un cosinus d'amplitude 1 et de valeur moyenne 0,5 ?

☐ On souhaite tracer un autre signal. Modifier les listes `c_e` et `phi_e` pour tracer le signal :

$$e(t) = 0{,}5 + 2{,}5.\cos(\omega_e.t) + 1{,}5.\cos(2\omega_e.t) + 1{,}5.\cos\left(3\omega_e.t + \frac{\pi}{2}\right)$$

☐ Exécuter la cellule pour afficher le signal correspondant.

1.2. Diagramme de Bode du filtre

On commence par l'exemple d'un filtre passe-bas. Sa fonction de transfert, son gain et son argument sont :

$$\underline{H}(j\omega) = \frac{1}{1+j\frac{\omega}{\omega_c}}, \quad G(\omega) = \left|\underline{H}(\omega)\right| = \frac{1}{\sqrt{1+\omega^2/\omega_c^2}}, \quad \Delta\varphi(\omega) = \arg\left(\underline{H}(\omega)\right) = -\arctan(\omega/\omega_c)$$

☐ Compléter les fonctions `gain(omega_c,omega)` et `dephasage(omega_c,omega)` pour le filtre passe-bas.

☐ Exécuter la cellule pour afficher le diagramme de Bode et vérifier qu'il a bien l'allure attendue.

1.3. Construction du signal de sortie

On rappelle l'action d'un filtre sur le signal d'entrée :

$$e(t) = c_0 + \sum_{n=1}^{+\infty} c_n.\cos(n.\omega_e.t + \varphi_n) \xrightarrow{filtre} s(t) = c_0' + \sum_{n=1}^{+\infty} c_n'.\cos(n.\omega_e.t + \varphi_n')$$

avec $c_n' = c_n \times G(n.\omega_e)$ et $\varphi_n' = \varphi_n + \Delta\varphi(\omega)$

Notons c_n' et φ_n' les coefficients du spectre du signal de sortie, et c_n et φ_n ceux du signal d'entrée.

Q4. Donner l'expression de c_n' en fonction de c_n et du gain G. A quelle pulsation ce gain doit-il être pris ?

Q5. Donner l'expression de φ_n' en fonction de φ_n et de l'argument $\arg(H)$. A quelle pulsation cet argument doit-il être pris ?

☐ Compléter alors les deux lignes `c_s.append` et `phi_s.append` en indiquant ce qu'il faut ajouter aux listes `c_s` et `phi_s` pour les remplir (d'après la question précédente).

☐ En utilisant la fonction `signal` définie précédemment, compléter la ligne sortie = …. qui calcule le signal de sortie à partir de ses coefficients de Fourier `c_s` et `phi_s`.

La suite du code contient déjà les lignes qui permettent de tracer le signal de sortie.

☐ L'exécuter : ceci doit afficher les signaux e(t) et s(t).

Q6. Actuellement dans le code, que vaut la fréquence du signal d'entrée ? Et la fréquence $f_c = \omega_c/(2\pi)$ du filtre passe-bas ?

1.4. Interprétation

Le code correspondant à la partie 4 est déjà écrit. Il s'agit de lignes, déjà complètes, qui permettent de tracer le diagramme de Bode du filtre sur la figure du haut, et les coefficients c_n et c_n' sur la figure du bas.

☐ Conservez f_e = 1 kHz, faire un essai avec f_c = 1 kHz, puis f_c = 10 kHz et f_c = 100 Hz. Interpréter à chaque fois.

2. Filtre passe-bande

On s'intéresse maintenant à l'action d'un filtre passe bande. Sa fonction de transfert, son gain et son argument sont :

$$\underline{H}(j\omega) = \frac{1}{1 + jQ\left(\frac{\omega}{\omega_0} - \frac{\omega_0}{\omega}\right)}$$

$$G(\omega) = \left|\underline{H}(\omega)\right| = \frac{1}{\sqrt{1+Q^2\left(\frac{\omega}{\omega_0}-\frac{\omega_0}{\omega}\right)^2}}, \qquad \Delta\varphi(\omega) = \arg\left(\underline{H}(\omega)\right) = -\arctan\left(Q\left(\frac{\omega}{\omega_0} - \frac{\omega_0}{\omega}\right)\right)$$

2.1. Action sur le signal précédemment utilisé

Le script correspondant est déjà écrit :

- ☐ Télécharger et ouvrir le script *TP14_Passebande.py*.
- ☐ Exécuter ce script dans son ensemble. Si tout fonctionne, faire plusieurs essais (en variant Q et f_0 du filtre) et interpréter à chaque fois.
- ☐ Essayer de choisir Q et f_0 pour ne sélectionner que l'harmonique $n = 2$ du signal $e(t)$.

2.2. Action sur un signal rectangle

Pour étudier l'action du filtre sur un signal rectangle, il faut d'abord en construire un. Ses coefficients de Fourier sont les suivants :

$$\begin{cases} \text{si } n \text{ pair} : c_n = 0 \quad et\ \varphi_n = -\pi/2 \\ \text{si } n \text{ impair} : c_n = \dfrac{2}{\pi.n} \quad et\ \varphi_n = -\pi/2 \end{cases}$$

En Python, le symbole % permet d'obtenir le reste dans la division euclidienne. On peut donc tester si n est pair avec `n%2 == 0` : ceci est vrai si et seulement si n est pair.
Il faut définir les listes `c_e` et `phi_e` à l'aide de la définition ci-dessus. On utilisera une boucle du type suivant :

```
c_e = []
phi_e = []
nb_harm = 20
for n in range (nb_harm):
    phi_e.append(       )      # à compléter
    if n%2==0:
        c_e.append(     )           # à compléter
    else :
        c_e.append(     )      # à compléter
```

- ☐ Tester pour `nb_harm` = 3 ; 5 ; 20... et voir que votre signal d'entrée est de plus en plus rectangulaire. Pour la suite on prendra par exemple 20 termes.

On s'intéresse ensuite à l'action du filtre passe-bande sur le signal rectangle. On souhaite se placer dans trois configurations différentes :

- ☐ Choisir les caractéristiques du filtre pour ne sélectionner que le fondamental du signal rectangle.

Q7. Dans quel domaine du diagramme de Bode un filtre a-t-il une action de dérivateur ?

- ☐ Choisir les caractéristiques du filtre pour réaliser la dérivée du signal rectangle (on prendra Q assez faible pour éviter une résonance).

Q8. Dans quel domaine du diagramme de Bode un filtre a-t-il une action d'intégrateur ?

- ☐ Choisir les caractéristiques du filtre pour réaliser une primitive du signal rectangle.

Ceux qui ont terminé peuvent câbler un circuit RLC pour réaliser un passe-bande et reproduire en vrai, avec un GBF et un oscilloscope les observations du TP.

Mémo succinct pour Python

Graphiques

```
import matplotlib.pyplot as plt
```

`plt.figure(mon_titre, figsize=(W,H))` *crée ou sélectionne une figure dont la barre de titre contient mon_titre et dont la taille est W×H (en inches, uniquement lors de la création de la figure)*

`plt.plot(X,Y,dir_abrg)` *trace le nuage de points d'abscisses dans X et d'ordonnées dans Y ; dir_abrg est une chaîne de caractères qui contient une couleur ("r"-ed, "g"-reen, "b"-lue, "c"-yan, "y"-ellow, "m"-agenta, "k" black), une marque ("o" rond, "s" carré, "*" étoile,...) et un type de ligne ("" pas de ligne, "-" plain, "--" dashed, ":" dotted, ...) ; options courantes : label=..., linewidth=..., markersize=...*

`plt.axis("equal"), plt.grid()` *repère orthonormé, quadrillage*

`plt.xlim(a,b), plt.ylim(a,b)` *plages d'affichage ; si a > b, inversion de l'axe*

`plt.xlabel(axe_x,size=s,color=(r,g,b)), plt.ylabel(axe_y,...)` *étiquettes sur les axes, en réglant la taille s et la couleur de la police de caractères (r, g et b dans [0,1])*

`plt.legend(loc="best",fontsize=s)` *affichage des labels des "plot" en légende*

`plt.show()` *affichage des différentes figures et remise à zéro*

Fonctions mathématiques

```
import numpy as np
```

Les fonctions de numpy sont vectorisées

`np.pi, np.e` → *Constantes π et e*

`np.abs, np.sqrt, np.exp, np.log, np.log10, np.log2` → `abs`, *racine carrée, exponentielle, logarithmes népérien, décimal, en base 2*

`np.cos, np.sin, np.tan` → *Fonctions trigonométriques (angles en radians)*

`np.arccos, np.arcsin, np.arctan` → *Fonctions trigonométriques réciproques*

`np.arctan2(y,x)` → *Angle dans]–π,π]*

`np.cosh, np.sinh` *(trigonométrie hyperbolique)*

Ressource complète : https://www.banquept.fr/documents/2018/MementoPython_BanquePT.pdf