Transformée de Fourier discrète et filtrage d'un signal périodique non sinusoïdal

# 1 DSF et synthèse de Fourier

## 1.1 Décomposition en Série de Fourier : DSF

Pour toute fonction T-périodique on a

$$f(t) = a_0 + \sum_1^\infty a_k(f)\cos(k\omega t) + b_k(f)\sin(k\omega t)$$

avec

$$a_0(f) = \frac{1}{T}\int_0^T f(t)\mathrm{dt}$$

$a_0$ représente la valeur moyenne de f sur sa période.

$$a_k(f) = \frac{2}{T}\int_0^T f(t)\cos(k\omega t)\mathrm{dt}$$
$$b_k(f) = \frac{2}{T}\int_0^T f(t)\sin(k\omega t)\mathrm{dt}$$

## 1.2 Recomposition d'un signal

on note $S_n$ la série de Fourier de rang n définie par

$$S_{n,f}(t) = a_0 + \sum_1^n a_k(f)\cos(k\omega t) + b_k(f)\sin(k\omega t)$$

Cette série converge vers f lorsque n tend vers l'infini $\lim_{n\to+\infty} S_n(t) = f(t)$

## 1.3 Un premier exemple

Le signal à étudier est $e(t) = 2 + \sin(t) + 0.5\sin(3 * t)$.
Compléter le programme "transformée de Fourier discrete incomplet" dans le but de déterminer le spectre de e(t), tracer la fonction et son spectre.
Le calcul de l'intégrale pourra se faire numériquement en utilisant la méthode des trapèzes ou bien en utilisant la fonction quad de python (cf 1.4.2 pour l'utilisation)

## 1.4 synthèse d'un signal carré

l'idée est de montrer comment reconstituer le signal à partir de sa DSF
Nous allons étudier une fonction "carré", qui est impaire et pour laquelle les coefficients $a_n$ sont nuls. Nous calculerons donc uniquement les coefficients $b_n$

1. Écrire en python une fonction carré $f(t)$ qui renvoie 1 pour t positf et -1 sinon. L'étude se fera $-0.5 \leq t \leq 0.5$ et la fonction carrée est alors considérée comme périodique de période 1 et d'amplitude 1.

2. Calculer les 50 premiers coefficients $b_n$ que vous mettez dans un vecteur bnCarre et que vous aurez au préalable remplis de zéros (méthode np.zeros). On intègre entre -0.5 et +0.5 (c'est équivalent à l'intervalle [0,1]).
   Vous pouvez utiliser la fonction **quad** du module **scipy.integrate** qui renvoie ici un nombre complexe, de partie imaginaire négligeable. On récupère la partie réelle en prenant la première composante, sans oublier que Python commence l'indexation à 0.
   On utilise également une propriété intéressante de la fonction quad : la fonction donnée en premier argument est d'habitude une simple fonction $t \to f(t)$, et l'intervalle de variation de la variable t. Mais si elle admet un paramètre, $(t,k) \to f(t,k)$, la variable d'intégration est toujours la première, t, mais on peut spécifier la valeur du paramètre, k, à l'aide de l'option args = (k).

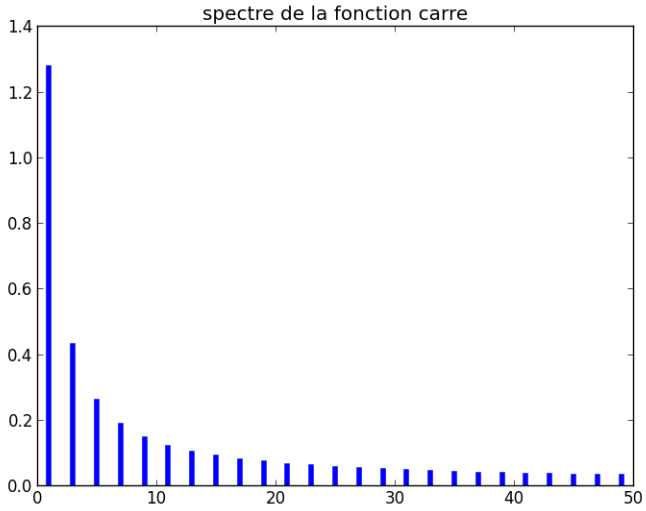
   ```
   exemple de syntaxe d'utilisation de quad
   quad(lambda t,k: carre(t)*sin(2*pi*k*t),
    -0.5,0.5,args=(k))[0]
   ```

   rem : l'utilisation du mot clé **lambda** n'est pas obligatoire, on peut aussi définir séparément la fonction à intégrer. C'est vous qui voyez....

3. Tracer le spectre de f(t) (cette partie est codée). On prendra comme amplitude la valeur absolue de $b_n$.
   Quelques options de tracé sur l'exemple suivant :

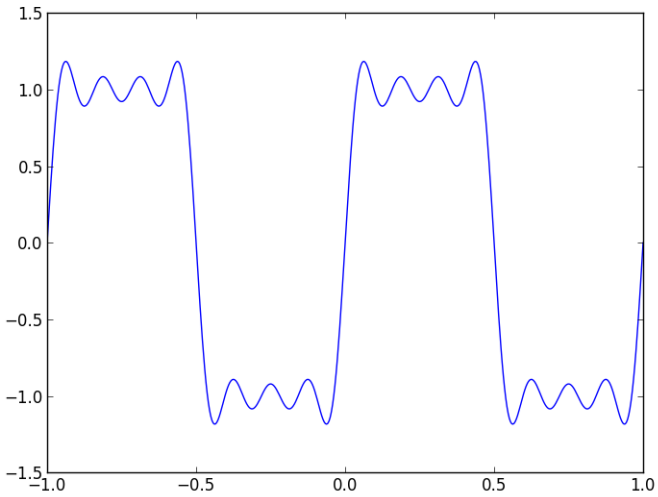
   ```
   plt.plot([1,2],[3,4],'b',linewidth=4)
   ```

   On obtient ceci :

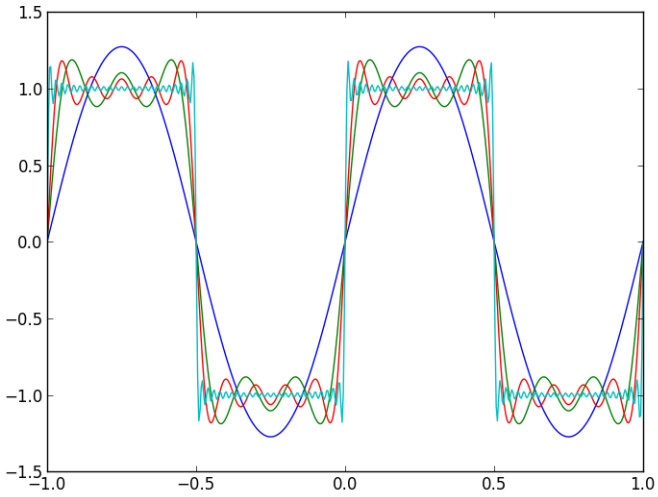



   **rem** on montre mathématiquement que l'amplitude du terme correspondant au fondamental est $\frac{4A}{\pi} \simeq 1.27$ si A=1, on retrouve bien cette valeur.

4. On trace ensuite les premières sommes de Fourier.
   Fixer votre valeur de n, 7 par exemple puis calculer et tracer $S_7(t)$.On obtient ceci :




On peut aussi superposer les courbes pour diverses valeurs de n



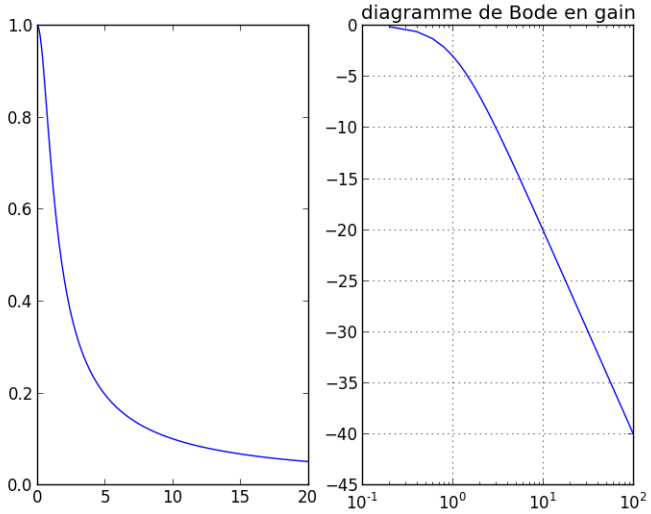

## 1.5 utilisation de l'existant

Pour terminer cette première partie, nous allons utiliser la fonction fft de python (module numpy, bibliothèque fft) pour déterminer le spectre du signal étudié.
Cette partie est entièrement codée, vous avez simplement à comprendre et à changer la valeur de $f_e$ pour voir les éventuels problèmes qui peuvent apparaitre.
On commence par un signal possédant une fréquence, puis un plus complexe avec 4 composantes. A chaque fois, dans la fenêtre de tracé, se trouvent la partie temporelle à gauche et fréquentielle à droite.

# 2 filtrage d'un signal carré, approche harmonique

## 2.1 filtre RC d'ordre un

1. rappeler la fonction de transfert d'un filtre passe bas (RC) d'ordre 1, à l'aide de la variable réduite $x = \omega/\omega_0$.
   En python, a fonction semilogx permet de tracer une courbe dans un repère semi-logarithmique. Vérifier que la courbe de réponse en gain du diagramme de Bode a les propriétés attendues.




2. On applique sur ce filtre un signal sinusoïdal $u(t) = U_m \cos(\omega t)$. Exprimer la sortie en fonction de H, le module de la fonction de transfert, $\varphi$, son argument, Um et $\omega$
3. En généralisant ce qui précède et ce que l'on a vu en cours, on peut donc écrire, pour un signal d'entrée égal à Sn(t), la somme de Fourier de rang n d'un signal périodique, que le signal à la sortie du filtre, noté $S'n(t)$ est donné par :

$$S'_n(t) = \sum_{k=1}^{n} H(k\omega) b_k(f) \sin(k\omega t + \varphi(k\omega))$$

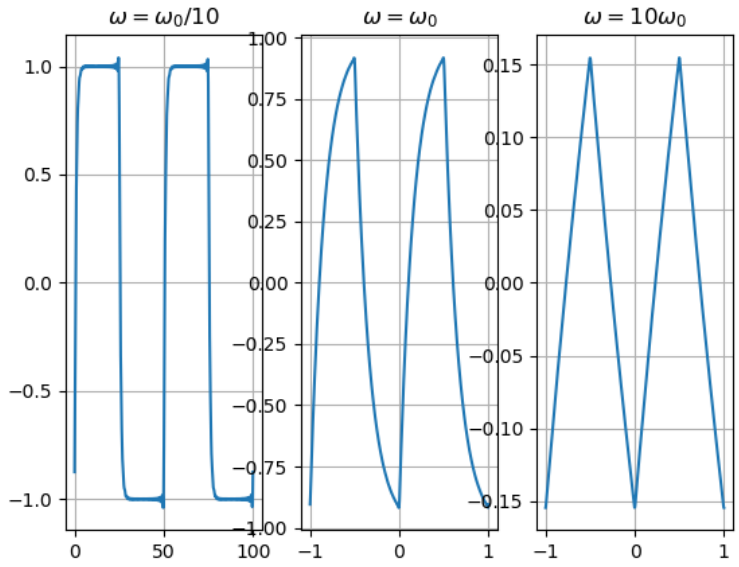
(valeur moyenne nulle pour le signal d'entrée et les $a_n$ valent 0)
Tracer $S'n(t)$ dans les 2 cas suivant :

(a) $\omega_=0.1\omega_0$ ou $x = 0.1$. interpréter.

(b) $\omega_=\omega_0$ ou $x = 1$. interpréter

(c) $\omega_=10\omega_0$ ou $x = 10$. interpréter

On obtient comme graphe du signal filtré les courbes ci-dessous
Noter la forte atténuation et la déformation lorsque le signal est en dehors de la bande passante




4. Observer et comparer les spectres des signaux filtrés par rapport zu spectre initial du signal carré.

## 2.2 extraction d'un harmonique

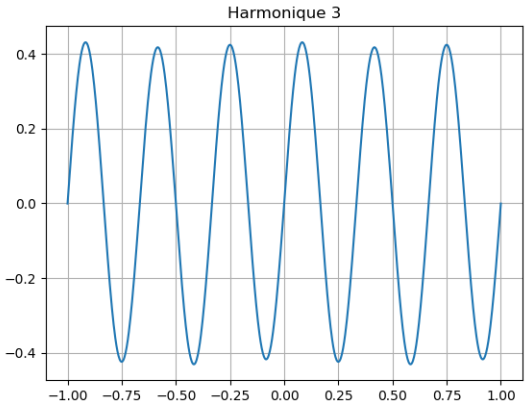
On rappelle que la fonction de transfert d'un filtre passe bande du second d'ordre et de gain unitaire ($H_0 = 1$) peut s'écrire sous la forme

$$H(j\omega) = \frac{1}{1 + jQ(x - \frac{1}{x})}$$

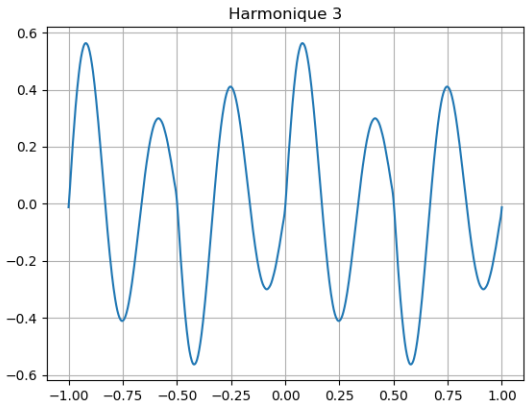
avec $x = \frac{\omega}{\omega_0}$.
La fonction **polar** du module **cmath** permet d'extraire le module et l'argument d'un nombre complexe. On met le résultat dans les tableaux h(k) et phi(k) (on travaille toujours sur les 50 premiers harmoniques). Polar vous évite donc le calcul explicite du module et argument.
On souhaite extraire uniquement le troisième harmonique du signal, on règle donc la pulsation centrale du filtre $\omega_0$ à $3\omega$. On a donc $x = 1/3$

$\Rightarrow$ Dans ce qui suit vous n'avez plus de code à écrire, juste à faire varier la valeur du facteur de qualité à la ligne 315 et comprendre.

Pour un facteur de qualité élevé (1000 par exemple) la bande passante est suffisamment étroite (elle vaut $\Delta f = \frac{f_0}{Q}$),le filtre suffisamment sélectif pour ne laisser passer (ou extraire) que le troisième harmonique par exemple sans le déformer.



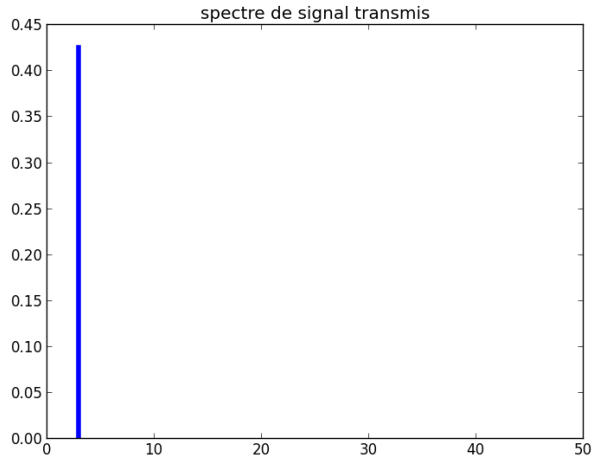

Avec un facteur de qualité plus faible (5 par exemple) ce n'est plus le cas, le fondamental et l'harmonique 5 (les plus proches en fréquence) ne sont pas suffisamment atténués, le signal de sortie est complexe.




l'harmonique 3 en particulier n'est plus isolé.

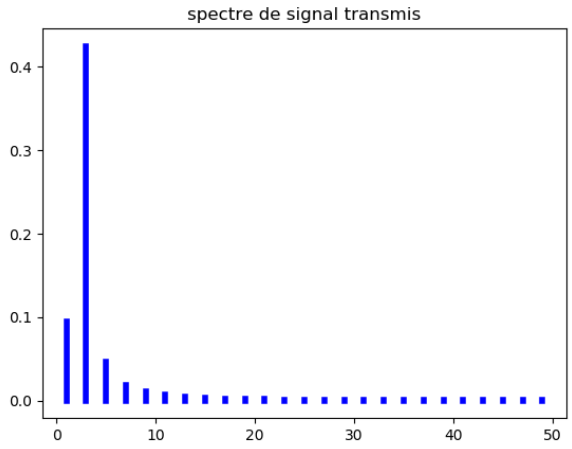
Tout ceci se voit très bien sur le spectre du signal transmis (en conservant $\omega_0 = 3\omega$ donc la bande passante du filtre est centrée sur le 3eme harmonique)

Si Q=1000 :




le filtre est très sélectif, il isole parfaitement le 3eme harmonique

Si Q=5 :



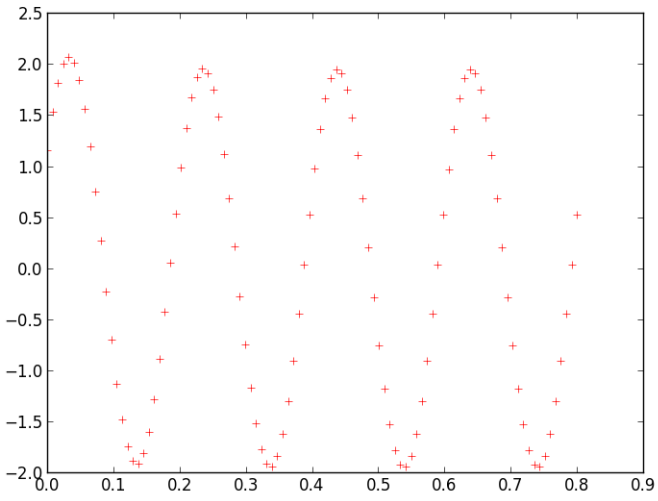

ici le filtre est peu sélectif

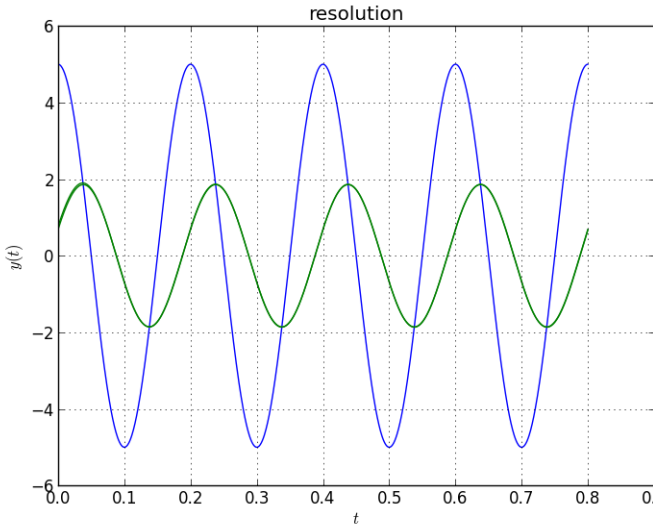
# 3 filtrage passe bas d'ordre un, approche temporelle

## 3.1 approche théorique

Dans ce qui suit nous allons procéder d'une façon différente, basée non plus sur l'analyse harmonique mais sur l'analyse temporelle. Nous revenons au cas du 2.1, un filtre Rc passe bas d'ordre un.

1. Retrouver à partir de l'expression de $\underline{H}(j\omega)$ l'équation différentielle reliant e(t) et s(t), sous la forme classique
$$\dot{s} + \frac{s(t)}{\tau} = \frac{e(t)}{\tau} \quad (1)$$
2. Nous allons *discrétiser* l'équation différentielle, c'est à dire calculer non plus une fonction continue du temps s(t) mais *une suite* de valeurs, approchant la solution à différents instants. Cette technique est particulièrement bien adaptée aux signaux échantillonnés. En prenant un "petit" nombre de points, cela donne ceci par exemple :






A gauche, le signal de sortie du filtre avec une entrée sinusoïdale, à droite la même chose mais avec plus de points de discrétisation. Le résultat semble continu. On note évidemment l'atténuation et le déphasage de la sortie par rapport à l'entrée.

$\Rightarrow$ alors comment établir cette suite de points ?

supposons que le signal d'entrée soit échantillonné à une fréquence $f_e$.
$e(t)$ est alors une suite de points $e_n$ telle que $e_n = e(t_n) = e(nT_e)$ où $T_e$ est la période d'échantillonnage.

Et bien nous allons créer une suite $s_n$ telle que $s_n \simeq s(nTe)$ et calculer cette suite par récurrence, la relation de récurrence étant déduite de l'équation différentielle.

a) Comment créer un échantillonnage de N points sur 4 périodes en python ?

b) Montrer que $s(nT_e + Te) \simeq s(nT_e) + T_e \frac{ds}{dt}(nT_e)$

c) En déduire qu'à l'instant $t_n = nT_e$ l'équation (1) s'écrit :

$$\frac{s_{n+1} - s_n}{Te} + \frac{s_n}{\tau} = \frac{e_n}{\tau}$$

d) Exprimer la relation de récurrence.

e)Compléter la partie correspondante du programme.

3. Écrire l'équation différentielle d'un filtre passe-haut d'ordre 1.
4. Optionnel : programmer un filtre d'ordre deux, un passe bas par exemple :

$$\underline{H} = \frac{H_0}{1 - x^2 + j\frac{x}{Q}}$$

Retrouver l'équation différentielle et discrétiser une dérivée seconde :

$$\frac{d^2f}{dt^2} = \frac{f(t+\tau) + f(t-\tau) - 2f(t)}{\tau^2}$$

## 3.2 exemple

La dernière partie du programme contient le filtrage d'un signal bruité par un filtre passe bas d'ordre un. On trace d'abord le spectre du signal bruité puis celui du signal bruité filtré par une technique similaire à celle étudiée au 3.1. Tout est codé, vous pouvez changer les paramètres :Il y a 3 fréquences en jeu : f, $f_0$ et $f_e$.

1. Sur le spectre du signal bruité, identifier les différentes fréquences qui apparaissent.
2. idem sur le signal filtré.