Simulation numérique de circuits électriques

Votre script final sera envoyé par mail avant le 02 janvier
Attention : ce travail est noté !

1 Réponse d'un circuit du premier ordre à un signal d'entrée quelconque

1.1 Méthode d'Euler

On cherche à résoudre numériquement l'équation $\dfrac{\mathrm{d}y}{\mathrm{d}t} = F(y,t)$ avec la condition initiale $y(0) = y_0$ sur l'intervalle temporel $[0, t_{\max}]$
Pour cela, nous allons *discrétiser* l'équation en découpant l'intervalle en N petits intervalles de même largeur h : $[t_k; t_{k+1}]$ avec $t_0 = 0$ et $t_N = t_{\max}$.
La valeur de $y(t)$ à l'instant t_k est notée $y_k = y(t_k)$. On a donc bien y_0 la condition initiale.

1. Calculer h en fonction de $t_{\max}$.
2. Justifier la relation de récurrence suivante

$$y_{k+1} = y_k + h \times F(y_k, t_k)$$

3. Ecrire, en langage Python, une fonction `euler(F,y0,tmax,N)` qui prend en paramètres une fonction `F`, la valeur `y0` de la condition initiale $y(0) = y_0$, la valeur maximale du temps et la valeur de `N` (le nombre de points) et qui retourne 2 listes (la liste des t_k et celle des y_k)

1.2 Réponse d'un circuit RC

On réalise un circuit RC alimenté par un générateur de f.e.m. $e(t)$ avec $e(t) = 0$ pour $t < 0$.

1. Montrer que la tension $u(t)$ aux bornes du condensateur vérifie l'équation différentielle

$$\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{1}{\tau}u(t) = \frac{1}{\tau}e(t)$$

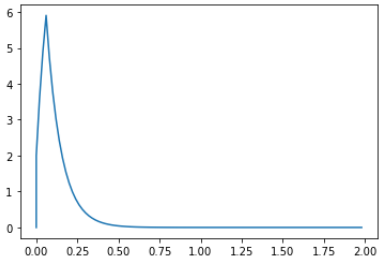
2. Donner alors l'expression de la fonction $F(y,t)$ définie plus haut et écrire la fonction Python `F(y,t)` associée.
3. Ecrire un script qui calcule numériquement la solution $u(t)$ pour $u(0) = 0$ et $e(t) = 10$ V pour $t \geq 0$
On prendra $\tau = 0,1$ s et $t_{\max} = 1$ s
4. Pour tracer une courbe avec Python, on rappelle la syntaxe :

```
import matplotlib.pyplot as plt
x=[ ... ] # x est une liste de N éléments
y=[ ... ] # y est une liste de N éléments.
#Il faut qu'il y ait autant d'éléments dans x que dans y
plt.clf()  # éventuellement pour effacer des figures précédentes
plt.plot(x,y)  #fabrique le graphique
plt.show() # affiche la figure
```

Tracer le graphe de la solution numérique (pour 100 points puis 1000 points)

5. La solution de l'équation différentielle est calculable à la main et son expression est $u(t) = 10\left(1 - \mathrm{e}^{-t/0.1}\right)$. On dit que c'est la solution numérique. Créer une liste `y2` à partir de la liste `x` obtenue avec la fonction `euler` :
`y2=[10*(1-exp(-10*t) for t in x]` après avoir importé le module `math`.

6. Quelle est la réponse $u(t)$ du circuit lorsque la tension d'entrée $e(t)$ est : $e(t) = 10$ pour $t \in [0; 0.05]$ et $e(t) = 0$ sinon ? Vous devriez obtenir ceci :



2 Décomposition spectrale d'un signal

2.1 Echantillonnage et F.F.T.

La transformée de Fourier permet de représenter le spectre de fréquence d'un signal non périodique. Un signal unidimensionnel est par exemple un signal électrique. Il peut être vu comme une fonction définie dans le domaine temporel $x : t \to x(t)$
Dans le cas du traitement numérique du signal, ce dernier n'est pas continu dans le temps, mais échantillonné.
Le signal échantillonné est obtenu en effectuant un prélèvement de certaines valeurs de $x(t)$ à des instants $t_k = kT_e$ où k est un entier et T_e est la période d'échantillonnage.
L'ensemble des valeurs échantillonnées est $\{x_k = x(t_k)\}$

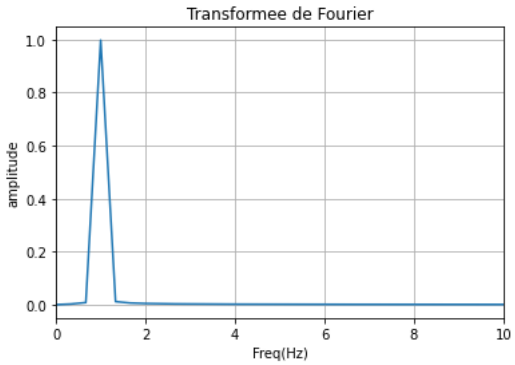
La transformée de Fourier rapide (notée F.F.T.) est un algorithme qui permet de calculer les transformées de Fourier discrètes d'un signal échantillonné. On note pour la suite $X(f)$ la F.F.T. du signal échantillonné à partir de $x(t)$.
Il existe plusieurs implantations dans Python de la F.F.T.. Nous allons utiliser celle du module `numpy`.

2.1.1 FFT d'un sinus

1. Créer une liste `te` contenant des instants allant de 0 à une durée de 3 s espacés de $T_e = 0,01$ s.
2. Créer ensuite une liste `xe` contenant les valeurs de $\sin(2\pi ft)$ aux différents instants de la liste `te`. On prendra $f = 1$ Hz.
3. Afficher la courbe qui représente `xe` en fonction de `te`
4. Recopier le script suivant et observer son rôle :

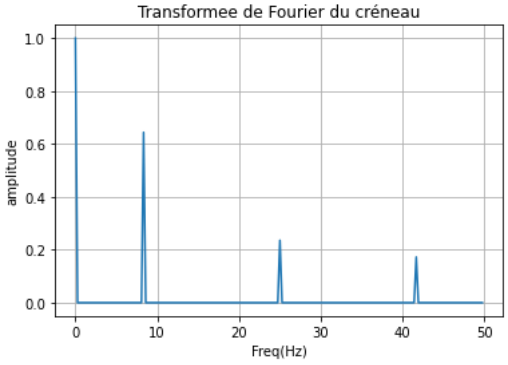
```
from numpy.fft import fft, fftfreq
import numpy as np
X=fft(xe)
N=len(xe)
Te=0.01
freq=fftfreq(N,d=Te)

X_abs=np.abs(X[:N//2])
X_norm=X_abs*2.0/N
freq_pos=freq[:N//2] #pour ne garder que les fréquences positives

plt.plot(freq_pos,X_norm, label="amplitude absolue")
plt.xlim(0,10)
plt.grid()
plt.xlabel(r"Freq(Hz)")
plt.ylabel(r"amplitude")
plt.title("Transformee de Fourier")
plt.show()
```

Vous devriez obtenir la courbe ci-dessous :



2.2 FFT d'un créneau

1. Télécharger le fichier `cren.csv` présent à cette adresse `http://physiquempsi4.free.fr/fichiers_index/cren.csv)`. La première colonne est le temps t et la deuxième la tension u.
2. En se servant du script « Lecture d'un fichier de données » présent à cette adresse `https://pastebin.com/tZvGE8mX`, créer deux listes `t` et `u` qui contiennent les valeurs correspondantes de t et de u respectivement.
3. Ecrire une fonction qui détermine la fréquence du signal associé aux 2 listes précédentes.
4. Adapter le script de calcul de FFT du sinus au signal présent. Vous devriez obtenir la courbe ci-dessous :



2.3 Recomposition d'un signal

On montre qu'un signal créneau $c(t)$ de fréquence f_0 se décompose en séries de Fourier :

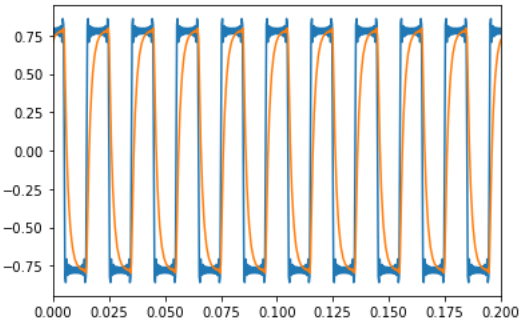
$$c(t) = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \cos\left(2\pi(2k+1)f_0 t\right)$$

1. A l'aide de Python, écrire une fonction `trace(n,tmax,f0,N)` qui affiche l'allure sur N points de la fonction
$$c(t) = \sum_{k=0}^{n} \frac{(-1)^k}{2k+1} \cos\left(2\pi(2k+1)f_0 t\right)$$
sur l'intervalle $[0, t_{\texttt{max}}]$
2. Afficher le spectre du créneau calculé pour $n = 20$, $f_0 = 100$ Hz, $t_{\texttt{max}} = 1$ s. Vérifier la cohérence du résultat.

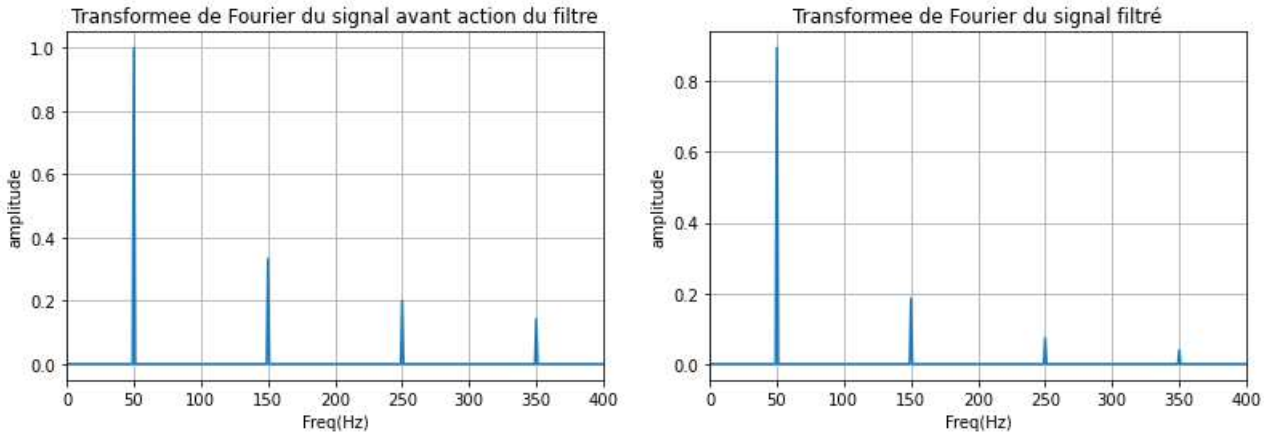
3 Action d'un filtre sur un signal créneau

Un filtre a pour fonction de transfert $\underline{H}(j\omega) = \dfrac{H_0}{1 + j\dfrac{\omega}{\omega_c}}$

1. Exprimer le gain en dB et le déphasage sortie/entrée
2. Tracer le diagramme de Bode pour $H_0 = 1$ et $\omega_c = 2\pi \times 100$. Pour obtenir une échelle logarithmique, il faut ajouter `plt.xscale("log")` avant `plt.show()`.
3. Tracer sur le même graphique le créneau avant filtrage et le signal filtré. Vous devriez obtenir :



4. Tracer ensuite sur le même graphique le spectre d'un créneau de fréquence $f_0 = 100$ Hz avant et après filtrage. Que remarque-t-on ?



5. (Facultatif) Reprendre l'étude précédente avec le filtre passe-bande $\underline{H} = \dfrac{1}{1 + jQ\left(\dfrac{f}{f_p} - \dfrac{f_p}{f}\right)}$ avec $f_p = f_0 = 100$ et pour différentes valeurs de Q.

Remarque : vous verrez l'an prochain comment appliquer un filtre (numérique) directement sur un signal échantillonné...

4 Effets non linéaires

1. Créer une liste `te` contenant des instants allant de 0 à une durée de 3 s espacés de $T_e = 0,01$ s.
2. Créer ensuite une liste `xe` contenant les valeurs de $\sin(2\pi f t)$ si ces valeurs sont inférieures à 0,5 (en valeur absolue)[sinon, écrêter à la valeur $\pm 0,5$] aux différents instants de la liste `te`. On prendra $f = 1$ Hz. L'écrêtage simulera l'action d'un filtre en saturation.
3. Afficher la courbe qui représente `xe` en fonction de `te`
4. Afficher le spectre du signal et vérifier que de nouvelles fréquences apparaissent.