Etude de l'influence de l'amplitude et de la période pour un signal périodique (version élèves)

Nous allons étudier un signal sinusoïdal. Un tel signal se répète identique à lui-même tous les 2 π, au bout d'une durée T (période en s).

Son évolution au cours du temps t se traduit par la fonction mathématique : A.sin((2π/T).t)

Où A est l'amplitude

Comme le temps t ne peut pas être continu, il faut le discrétiser, c'est à dire calculer t pour des valeurs entières, multiples d'une petite durée appelée *période d'échantillonnage* et notée Te.

Te doit être suffisamment petit par rapport à T.

Ce qui donne l'expression mathématique suivante : A.sin((2π/T).i.Te)

Avec i prenant des valeurs entières.

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# cette fonction permet d'afficher un graphique à un emplacemnt précis de la fenêtre␣
 ↪graphique.
# Ainsi, on peut afficher plusieurs sous-graphiques sur la même fenêtre.
def affichage_graphique(n,y,l,yl) :
    plt.subplot(3,1,n)                          # Déclaration du nombre d'emplacements dans la␣
 ↪fenêtre
    plt.plot(t,y,'k-',lw=1,label=n)             # Affichage de la courbe
    plt.ylim(-2,2)                              # Impose les bornes min et max sur l'axe des␣
 ↪ordonnées
    plt.grid()
    plt.xlabel('t (s)')
    plt.ylabel(yl)
    plt.legend()

# Déclaration des variables
Ymax=1    # amplitude en m
T=1       # période en s
Te=0.01    # période d'échantillonnage en s
t=[]                                            # Création des listes (vides) qui␣
 ↪contiendront les valeurs
y1=[]                                           # du temps et des amplitudes

for i in range (0,1000) :                       # Boucle permettant de parcourir toutes␣
 ↪les valeurs du temps discrétisé.
    t.append(i*Te)                              # La méthode append permet de rajouter une␣
 ↪valeur en fin de la liste t
    y1.append(Ymax*np.sin((2*np.pi/T)*i*Te))    # la fonction sinus est contenue dans la␣
 ↪bibliothèque numpy
```

```
                                                    # la constante pi est contenue dans la␣
 ↪bibliothèque numpy

# création du graphique

plt.figure(2,figsize=(10,12))          # création de la fenêtre graphique
affichage_graphique(1,y1,"courbe de référence","y1 (m)")    #appel de la fonction gérant␣
 ↪l'affichage du sous-graphique

plt.show()
```

Etude préalable :

En tenant compte des renseignements donnés lignes 6, 7 et 12, répondez aux questions suivantes : 1. Combien d'échantillons temporels aura-t-on? 2. Par quel calcul simple aurait-on pu prévoir la durée du signal créé? 3. Par quel calcul simple aurait-on pu prévoir le nombre de périodes affichées? 4. Combien d'échantillons temporels a-t-on par période?

Nous souhaitons étudier l'influence des paramètres A et T sur l'évolution temporelle du signal sinusoïdal.

Pour cela, nous avons déjà écrit en lignes 24 et 25, la création d'un signal sinusoïdal de référence noté y1.

Nous avons également déjà déclaré les listes y2 et y3 sur les lignes de code 20 et 21.

Sur le modèle de la ligne 25, compléter la ligne 26 de manière à créer un signal sinusoïdal nommé y2 d'amplitude deux fois plus grande que y1.

Puis, toujours sur le même modèle, compléter la ligne 27 de manière à créer un signal sinusoïdal nommé y3 de période deux fois plus grande que y1.

Nous souhaitons de plus, afficher ces trois signaux en trois graphiques situés l'un au-dessous de l'autre. Nous allons pour cela utiliser la méthode plt.subplot(nombre de lignes, nombre de colonnes, index) de la bibliothèque matplotlib.pyplot as plt.

L'affichage est géré par une fonction nommée affichage_graphique qui a besoin d'un certain nombre de renseignements (fournis entre parenthèses) pour fonctionner correctement.

Sur le modèle de la ligne 31, écrire la ligne de code nécessaire à l'affichage de y2.

Puis, toujours sur le modèle de la ligne 31, écrire la ligne de code nécessaire à l'affichage de y3.

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

def affichage_graphique(n,y,l,yl) :
    plt.subplot(3,1,n)                    # Déclaration du nombre d'emplacements dans la␣
 ↪fenêtre
    plt.plot(t,y,'k-',lw=1,label=n)       # Affichage de la courbe
    plt.ylim(-2,2)                        # Impose les bornes min et max sur l'axe des␣
 ↪ordonnées
    plt.grid()
    plt.xlabel('t (s)')
    plt.ylabel(yl)
    plt.legend()
```

```
# Déclaration des variables
Ymax=1    # amplitude en m
T=1       # période en s
Te=0.01    # période d'échantillonnage en s
t=[]                                                # Création des listes (vides) qui␣
 ↪contiendront les valeurs
y1=[]                                               # du temps et des amplitudes
y2=[]
y3=[]

for i in range (0,1000) :                            # Boucle permettant de parcourir toutes␣
 ↪les valeurs du temps discrétisé.
    t.append(i*Te)                                   # La méthode append permet de rajouter une␣
 ↪valeur en fin de la liste t
    y1.append(Ymax*np.sin((2*np.pi/T)*i*Te))         # la fonction sinus est contenue dans la␣
 ↪bibliothèque numpy
                                                     # la constante pi est contenue dans la␣
 ↪bibliothèque numpy



plt.figure(2,figsize=(10,12))          # création de la fenêtre graphique
affichage_graphique(1,y1,"courbe de référence","y1 (m)")



plt.show()
```