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

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.

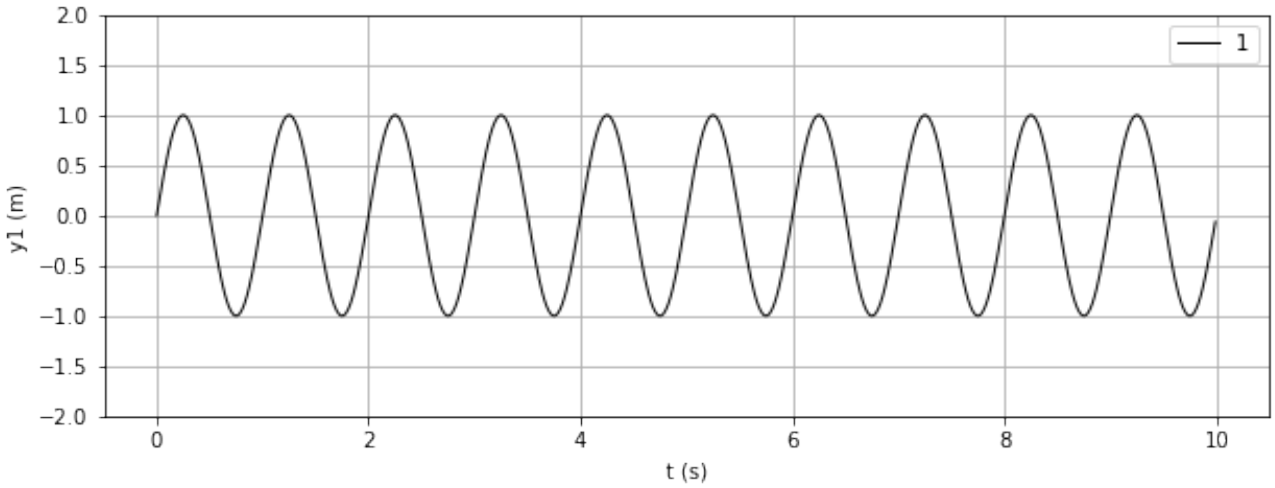
Ce qui donne l'expression mathématique suivante : A.sin((2π/T).i.Te)

Avec i prenant des valeurs entières.

In [1]:

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

# cette fonction permet d'afficher un graphique
# à un emplacement précis de la fenêtre graphique.
# Ainsi, on peut afficher plusieurs sous-graphiques
# sur la même fenêtre.

def affichage_graphique(n,y,l,yl) :
# Déclaration du nombre d'emplacements dans la fenêtre
    plt.subplot(3,1,n)
# Affichage de la courbe
    plt.plot(t,y,'k-',lw=1,label=n)
 # Impose les bornes min et max sur l'axe des ordonnées
    plt.ylim(-2,2)
    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
# Création des listes (vides) qui contiendront les valeurs
# du temps et des amplitudes
t=[]
y1=[]

# Boucle permettant de parcourir toutes les valeurs du temps
# discrétisé.
for i in range (0,1000) :
# La méthode append permet de rajouter une valeur en fin
```

```
# de la liste t
    t.append(i*Te)
# la fonction sinus est contenue dans la bibliothèque numpy
# la constante pi est contenue dans la bibliothèque numpy
    y1.append(Ymax*np.sin((2*np.pi/T)*i*Te))


# création du graphique

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

plt.show()
```



Etude préalable :

En tenant compte des renseignements donnés lignes 24, 25 et 33, 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 25 et 37 du programme ci-dessous, 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 26 et 27.

Sur le modèle de la ligne 37, compléter la ligne 38 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 39 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 paramètres (fournis entre parenthèses) pour fonctionner correctement.

Sur le modèle de la ligne 43, écrire la ligne de code nécessaire à l'affichage de y2.

Puis, toujours sur le modèle de la ligne 43, écrire la ligne de code nécessaire à l'affichage de y3.

In [2]:

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

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

# Création des listes (vides) qui contiendront les valeurs
# du temps et des amplitudes
t=[]
y1=[]
y2=[]
y3=[]

# Boucle permettant de parcourir toutes les valeurs du temps discrétisé.
for i in range (0,1000) :
# La méthode append permet de rajouter une valeur en fin de la liste t
    t.append(i*Te)
# la fonction sinus est contenue dans la bibliothèque numpy
# la constante pi est contenue dans la bibliothèque numpy
# On aurait pu aussi utiliser la bibliothèque math pour y avoir accés
# à l'aide des fonctions math.sin() et math.pi
    y1.append(Ymax*np.sin((2*np.pi/T)*i*Te))



    # création de la fenêtre graphique
```

```
plt.figure(2,figsize=(10,12))
affichage_graphique(1,y1,"courbe de référence","y1 (m)")



plt.show()
```