Compétences : Représenter à l'aide d'un langage de programmation, la somme de deux signaux sinusoïdaux périodiques synchrone en faisant varier la phase à l'origine de de l'un des deux.

TP4 : Somme de deux sinusoïdes

Un casque anti-bruit permet de réduire l'intensité sonore perçue en utilisant une mousse absorbante. Un casque anti-bruit actif est plus performant. Comment fonctionne un casque anti-bruit actif ?

Document 1 Python

Se reporter aux rabats II et III au début de votre livre Hatier qui est une bonne introduction au langage de programmation de Python pour l'usage qu'on en a en physique et chimie :

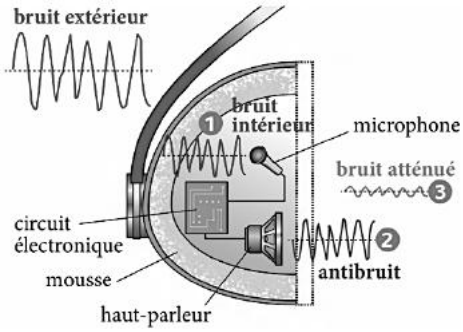
- Traitement de données (fonction tableur),
- Présentation des données sous forme de graphes (fonction grapheur),
- Modélisation à travers des fonctions (fonction simulateur)

Ou voir fiche méthode sur le Python. FM8.Python.pdf

Document 2 Principe de casque anti-bruit actif

L'air oscille sous l'effet d'ondes sonores, c'est-à-dire que sa pression augmente puis diminue successivement.

Dans le casque antibruit actif, on ajoute au bruit ① un second signal ② en opposition de phase de telle sorte que la variation de la pression globale soit quasiment nulle. Le bruit ③ qui parvient à l'oreille est alors atténué.



Document 3 Simulation somme de deux sinusoïdes.

https://www.sciences.univ-nantes.fr/sites/genevieve_tulloue/Ondes/general/somme.php

Travail à faire

Animation

Utiliser la simulation du document 3 pour voir la somme de deux sinusoïdes et comprendre la notion de deux signaux en phase et en opposition de phase

Python

On pourra un compilateur Python (Edupython, Thommy etc…) pour exécuter un programme Python.

Les fichiers « Sinusoïde1.py » , « Sinusoïde2.py », « Sinusoïde3.py » et « Sinusoïde4.py » sont sur Google Classroom.

- Ouvrir les fichiers « Sinusoïde1.py » , « Sinusoïde2.py ». Lire les commentaires et les codes Python pour comprendre le fonctionnement général de ces programmes.
- Exécuter ces programmes dans Edupython en appuyant sur le bouton « ▷ »
- Exécuter le programme « Sinusoïde3.py » avec pour la phase p les valeurs suivantes : 0, $\frac{\pi}{2}$, π et $2\,\pi$. (on pourra utiliser la valeur π de la bibliothèque Numpy : np.pi). Observer l'allure des courbes obtenues.
- Ouvrir le fichier « Sinusoïde4.py ». Lire les commentaires et le code Python pour comprendre le fonctionnement général du programme. Notamment expliquer à quoi sert l'instruction « for ».
- Exécuter le programme « Sinusoïde4.py » et observer l'influence de la valeur de phase sur la somme des deux sinusoïdes.

ANNEXE : PROGRAMMES PYTHON

```
"""
Programme sinusoïdes1.py
"""

#  Afficher une sinusoïde
# importation de la bibliothèque numpy pour des calculs numériques
import numpy as np
# importation de la bibliothèque pyplot pour tracer des graphes
import matplotlib.pyplot as plt
w=10

# liste de temps lt = liste de 1000 réels entre les valeurs 0 et 2
lt = np.linspace(0,2,1000)

# définition de la fonction s1 qui calcule y = sin(10*x)
def s1(t):
  return np.sin(w*t)

# Représentation d'un graphe sin avec couleur bleue "b" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2
plt.plot(lt,s1(lt),"b-",label="s1")

plt.xlabel("temps")
plt.ylabel("signal")
plt.title("sinusoïde seule")
plt.legend()
# montre le graphe courant.
plt.show()
```

```
"""
Programme sinusoïdes2.py
"""

#  Affichage de deux sinusoïdes s1 et s2 déphasées :
# importation de la bibliothèque numpy pour des calculs numériques
import numpy as np
# importation de la bibliothèque pyplot pour tracer des graphes
import matplotlib.pyplot as plt
w=10

# liste de temps lt = liste de 1000 réels entre les valeurs 0 et 2
lt = np.linspace(0,2,1000)

# définition de la fonction s1 qui calcule y = sin(10*x)
def s1(t):
  return np.sin(w*t)

# Dephase de p
p=2

# définition de la deuxième fonction s2 déphasée
def s2(t):
  return np.sin(w*t+p)

# Représentation d'un graphe sin(x) avec couleur bleue "b" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2
plt.plot(lt,s1(lt),"b-",label="s1")

# Représentation d'un graphe sin(x+phi) avec couleur verte "g" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2
plt.plot(lt,s2(lt),"g-",label="s2")

plt.xlabel("temps")
plt.ylabel("signal")
plt.title("sinusoïdes déphasées")
plt.legend()
# montre le graphe courant.
plt.show()
```

```
"""
Programme sinusoïdes3.py
"""

#  Sommation de deux sinusoïdes déphasées :
#  on trace la somme des deux sinusoïdes s1 et s2 pour une certaine valeur du déphasage
# importation de la bibliothèque numpy pour des calculs numériques
```

```
import numpy as np
# importation de la bibliothèque pyplot pour tracer des graphes
import matplotlib.pyplot as plt
w=10
# liste de temps lt = liste de 1000 réels entre les valeurs 0 et 2
lt = np.linspace(0,2,1000)

# définition de la fonction s1 qui calcule y = sin(10*x)
def s1(t):
  return np.sin(w*t)

# Dephase de p
p=2

# définition de la deuxième fonction s2 déphasée
def s2(t):
  return np.sin(w*t+p)

# définition de la fonction somme
def somme(t):
  return s1(t)+s2(t)

# Représentation d'un graphe sin(x) avec couleur bleue "b" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2 sur plt
plt.plot(lt,s1(lt),"b-",label="s1")

# Représentation d'un graphe sin(x+phi) avec couleur verte "g" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2 sur plt
plt.plot(lt,s2(lt),"g-",label="s2")

# Représentation de la somme avec couleur rouge "r" et des traits "-"
# pour les 1000 points de la liste lt entre 0 et 2 sur plt
plt.plot(lt,somme(lt),"r-",label="s1+s2")

plt.xlabel("temps")
plt.ylabel("signal")
plt.title("sinusoïdes déphasées")
plt.legend()
# montre le graphe courant.
plt.show()
"""
Programme sinusoïdes4.py
"""

# Utilisation d'une boucle **for** pour changer phi et visualiser l'effet du déphasage sur la somme des deux
sinusoïdes déphasées
# importation de la bibliothèque numpy pour des calculs numériques
import numpy as np
# importation de la bibliothèque pyplot pour tracer des graphes
import matplotlib.pyplot as plt
w=10
# liste de temps lt = liste de 1000 réels entre les valeurs 0 et 2
lt = np.linspace(0,2,1000)
# définition de la fonction s1 qui calcule y = sin(10*x)
def s1(t):
  return np.sin(w*t)
# définition de la deuxième fonction s2 déphasée
def s2(t):
  return np.sin(w*t+p)
# définition de la fonction somme
def somme(t):
  return s1(t)+s2(t)

# variation de p et représentation de la somme de s1 + s2
# pour les 7 points de la liste lt entre 0 et 3,14 sur plt
listep=np.linspace(0,np.pi,7)
for p in listep:
  plt.plot(lt,somme(lt),label="p="+str(round(p,1)))

plt.xlabel("temps")
plt.ylabel("signal")
plt.title("sinusoïdes déphasées")
plt.legend()
plt.show()
```