iNFO — IUT GRAND OUEST NORMANDIE

R 5.A.06

2025 – 2026

Sensibilisation à la programmation multimédia

Corrigé TD n° 1
Exercices Audios

ANNE Jean-François

Sensibilisation à la programmation multimédia

Programmation du Son

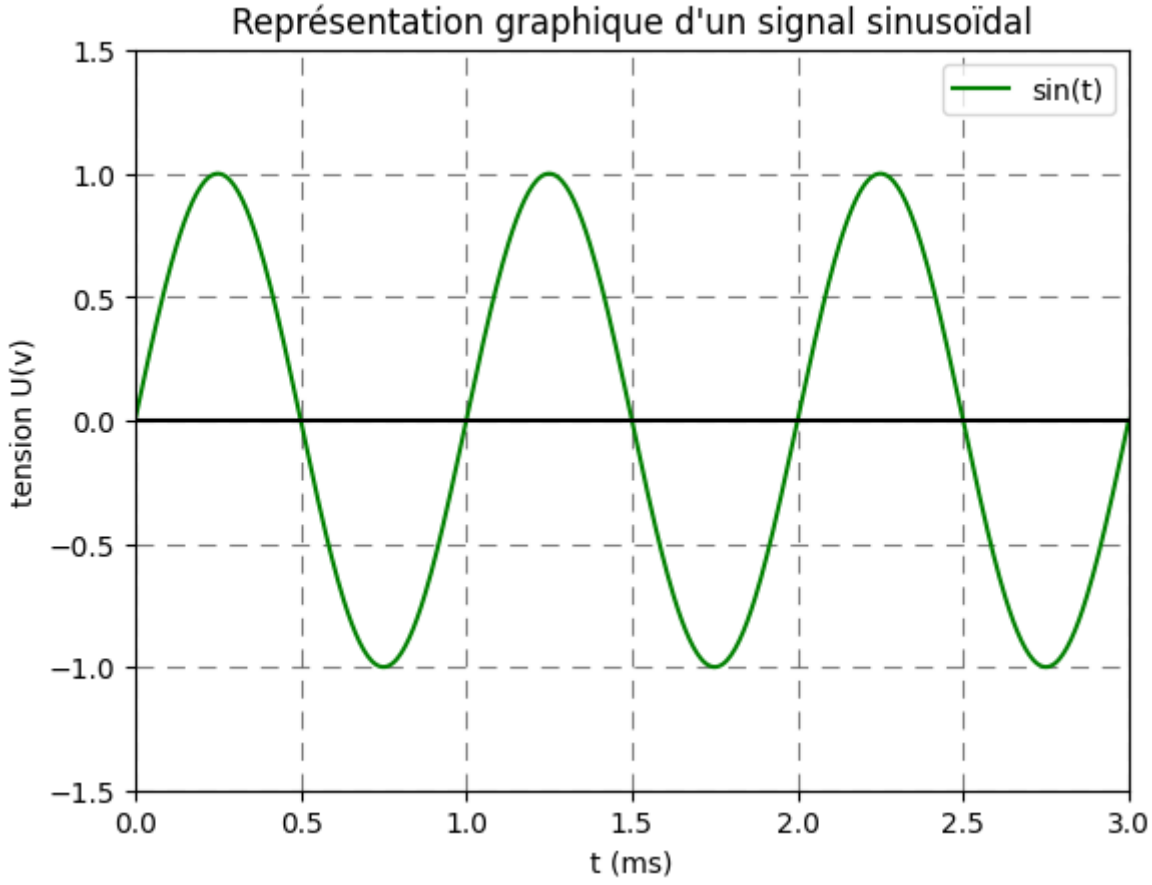
Le but de ce TD est de se familiariser avec le Sensibilisation à la programmation multimédia.

A. Les bases du Signal

1°) Exercice 1

Ecrire un programme python qui affiche un signal sinusoïdal de fréquence 1 kHz, d'amplitude 1 V et de phase 0°.

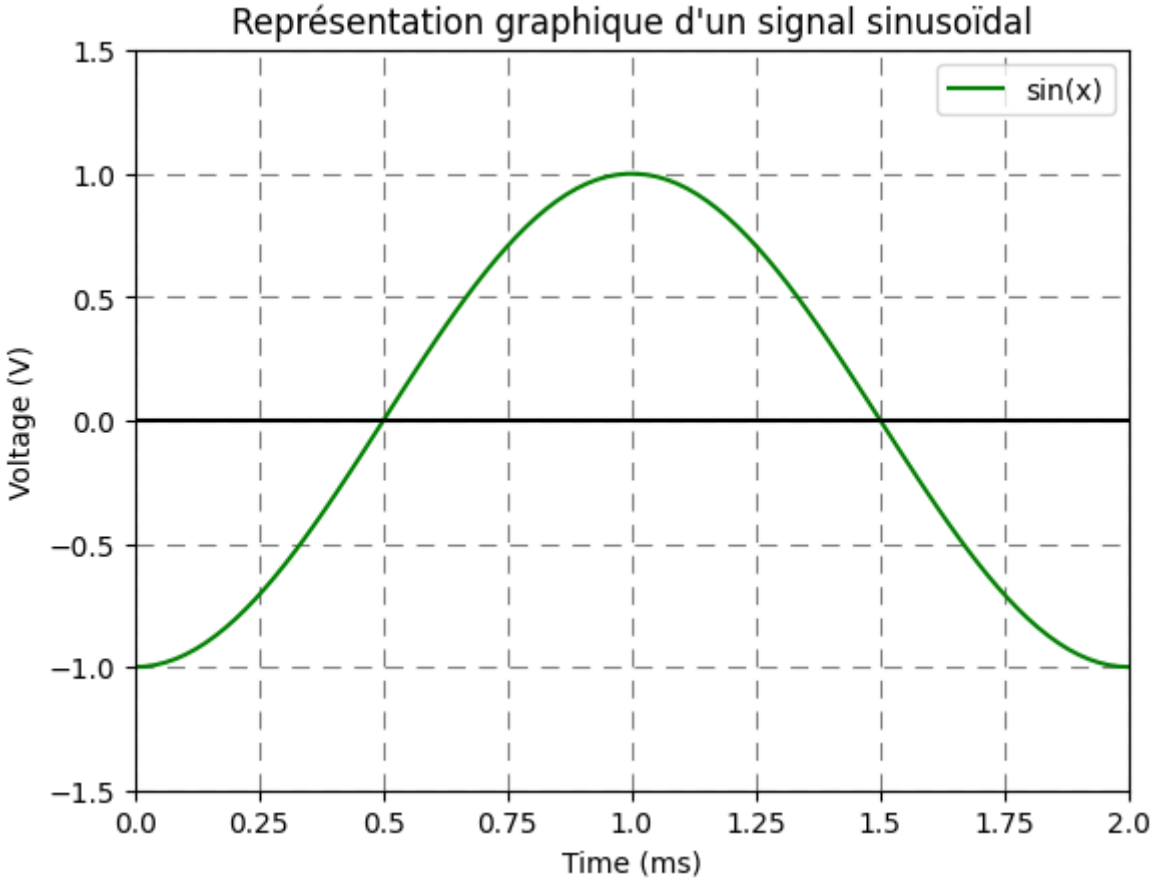
```
from matplotlib import pyplot as plt
from math import *
import numpy as np

# CARACTÉRISTIQUES :
tmax = 0.003  # durée du signal (sur l'axe des abscisses). Modifiable
T = 0.001  # période (s). Modifiable
# pulsation. ω = rapport du "tour complet" (2*pi) par la durée nécessaire pour le
parcourir = "vitesse angulaire" (rad/s)
ω = 2*pi/T
φ = 0  # phase à l'origine (rad). Modifiable
Umax = 1  # amplitude. Modifiable

repere = plt.axes(xlim=(0, tmax), ylim=(-1.5, 1.5))
repere.set_xlabel('t (ms)')
repere.set_ylabel('tension U(v)')
repere.set_title("Représentation graphique d'un signal sinusoïdal")

# nombre d'intervalles (n+1 points). Modifiable (augmenter si la courbe n'est pas
assez 'lisse')
n = 500

# liste des n+1 valeurs de t [0,n] (axe des abscisses)
t = [k*tmax/n for k in range(n+1)]
# calcul des n+1 valeurs de U (ordonnées). Une tension, mais cela pourrait être
une intensité, une élongation...
U = [Umax*sin(ω*t+φ) for t in t]

repere.plot(t, U, label='sin(t)', color='green')
plt.xticks(np.linspace(0, 0.003, 7), np.linspace(0, 3, 7))
repere.legend()
repere.grid(color='black', linestyle='--', dashes=[8, 6], alpha=0.5)
repere.axhline(0, color='black')
plt.grid(True)
plt.show()
```



2°) ***Exercice 2***

Ecrire un programme python qui affiche un signal sinusoïdal de phase 270° de fréquence 10 kHz, d'amplitude 2 V.

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

F = 5.e2            # No. of cycles per second, F = 500 Hz
T = 2.e-3           # Time period, T = 2 ms
phi = 3*pi/2        # déphasage
Fs = 50.e3          # No. of samples per second, Fs = 50 kHz
Ts = 1./Fs          # Sampling interval, Ts = 20 us
N = int(T/Ts)       # No. of samples for 2 ms, N = 100
repere = plt.axes(xlim=(0, 0.002), ylim=(-1.5, 1.5))
repere.set_title("Représentation graphique d'un signal sinusoïdal")
t = np.linspace(0, T, N)
signal = np.sin(2*np.pi*F*t+phi)

repere.plot(t, signal, label='sin(x)', color='green')
repere.legend()
repere.grid(color='black', linestyle='--', dashes=[8, 6], alpha=0.5)
repere.axhline(0, color='black')
plt.xticks(np.linspace(0, 0.002, 9), np.linspace(0, 2, 9))
plt.xlabel('Time (ms)')
plt.ylabel('Voltage (V)')
plt.show()
```

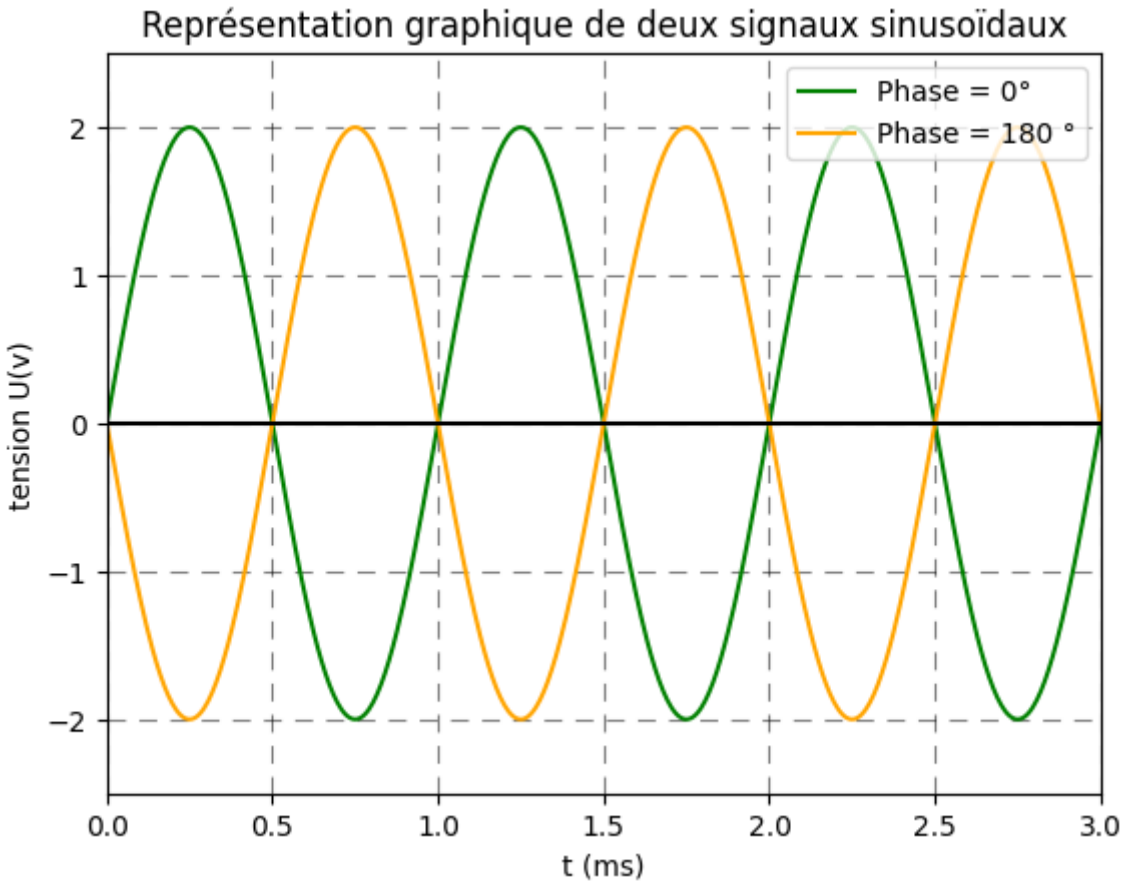


3°) ***Exercice 3***

Ecrire un programme python qui affiche un signal sinusoïdal de phase 0° et un signal sinusoïdal de phase 180°, de fréquence 1 kHz, d'amplitude 5 V, sur le même graphique.

```
from matplotlib import pyplot as plt
from math import *

# CARACTÉRISTIQUES :
tmax = 0.003  # durée du signal (sur l'axe des abscisses). Modifiable
T = 0.001  # période (s). Modifiable
# pulsation. ω = rapport du "tour complet" (2*pi) par la durée nécessaire pour le
parcourir = "vitesse angulaire" (rad/s)
ω = 2*pi/T
φ1 = 0  # phase à l'origine (rad). Modifiable
φ2 = pi  # phase à l'origine (rad). Modifiable
Umax = 2  # amplitude. Modifiable

repere = plt.axes(xlim=(0, tmax), ylim=(-2.5, 2.5))
repere.set_xlabel('t (ms)')
repere.set_ylabel('tension U(v)')
repere.set_title("Représentation graphique de deux signaux sinusoïdaux")

# nombre d'intervalles (n+1 points). Modifiable (augmenter si la courbe n'est pas
assez 'lisse')
n = 500

# liste des n+1 valeurs de t [0,n] (axe des abscisses)
t = [k*tmax/n for k in range(n+1)]
# calcul des n+1 valeurs de U (ordonnées). Une tension, mais cela pourrait être
une intensité, une élongation...
U1 = [Umax*sin(ω*t+φ1) for t in t]
U2 = [Umax*sin(ω*t+φ2) for t in t]

repere.plot(t, U1, label='Phase = 0°', color='green')
repere.plot(t, U2, label='Phase = 180 °', color='orange')
```

```
repere.legend(loc="upper right")
repere.grid(color='black', linestyle='--', dashes=[8, 6], alpha=0.5)
repere.axhline(0, color='black')
plt.grid(True)
plt.xticks(np.linspace(0, 0.003, 7), np.linspace(0, 3, 7))
plt.show()
```

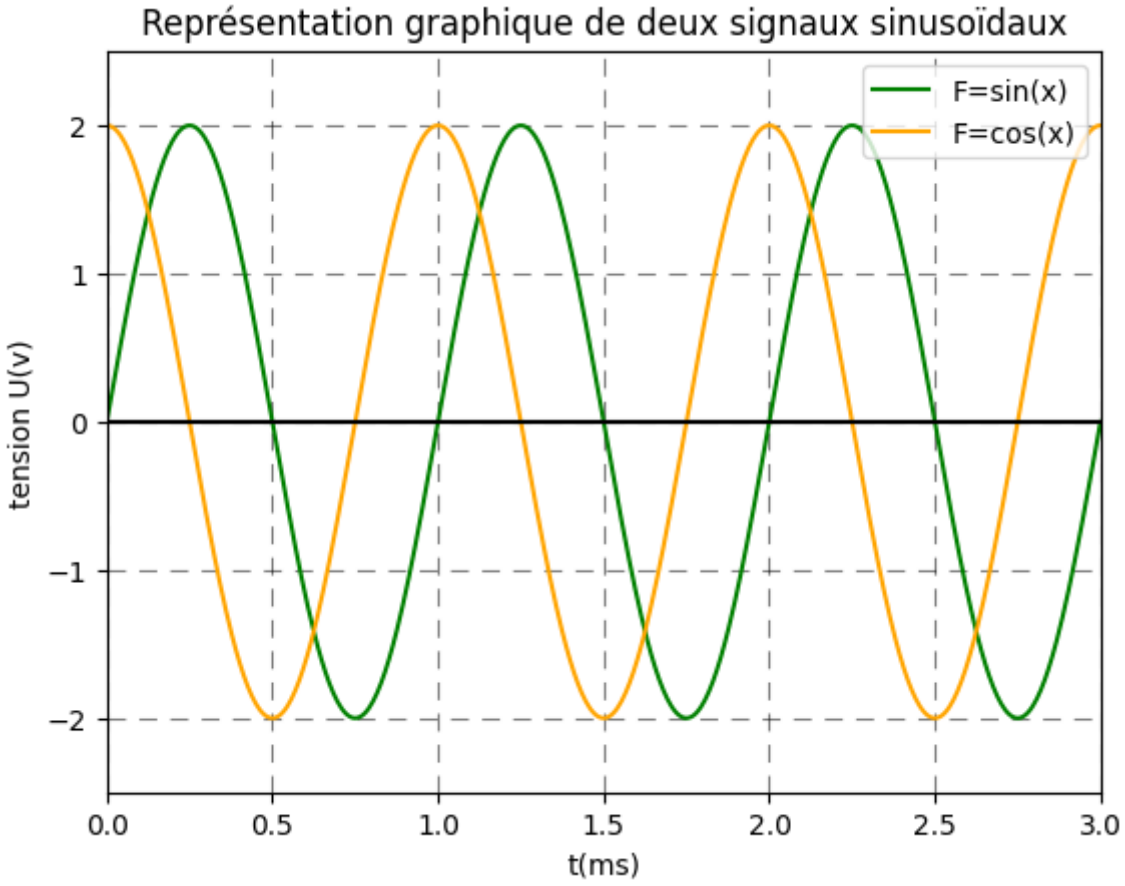


4°) ***Exercice 4***

Ecrire un programme python qui affiche un signal Cosinusoïdal et un signal sinusoïdal de fréquence 1 kHz, d'amplitude 2 V et de phase 0°, sur le même graphique.

```
from matplotlib import pyplot as plt
from math import *

# CARACTÉRISTIQUES :
tmax = 0.003  # durée du signal (sur l'axe des abscisses). Modifiable
T = 0.001  # période (s). Modifiable
# pulsation. ω = rapport du "tour complet" (2*pi) par la durée nécessaire pour le
parcourir = "vitesse angulaire" (rad/s)
ω = 2*pi/T
φ = 0  # phase à l'origine (rad). Modifiable
Umax = 2  # amplitude. Modifiable

repere = plt.axes(xlim=(0, tmax), ylim=(-2.5, 2.5))
repere.set_xlabel('t(ms)')
repere.set_ylabel('tension U(v)')
repere.set_title("Représentation graphique de deux signaux sinusoïdaux")

# nombre d'intervalles (n+1 points). Modifiable (augmenter si la courbe n'est pas
assez 'lisse')
n = 500

# liste des n+1 valeurs de t [0,n] (axe des abscisses)
t = [k*tmax/n for k in range(n+1)]
```

```
# calcul des n+1 valeurs de U (ordonnées). Une tension, mais cela pourrait être
une intensité, une élongation...
U1 = [Umax*sin(ω*t+φ) for t in t]
U2 = [Umax*cos(ω*t+φ) for t in t]

repere.plot(t, U1, label='F=sin(x)', color='green')
repere.plot(t, U2, label='F=cos(x)', color='orange')
repere.legend()
repere.grid(color='black', linestyle='--', dashes=[8, 6], alpha=0.5)
repere.axhline(0, color='black')
plt.xticks(np.linspace(0, 0.003, 7), np.linspace(0, 3, 7))
plt.grid(True)
plt.show()
```

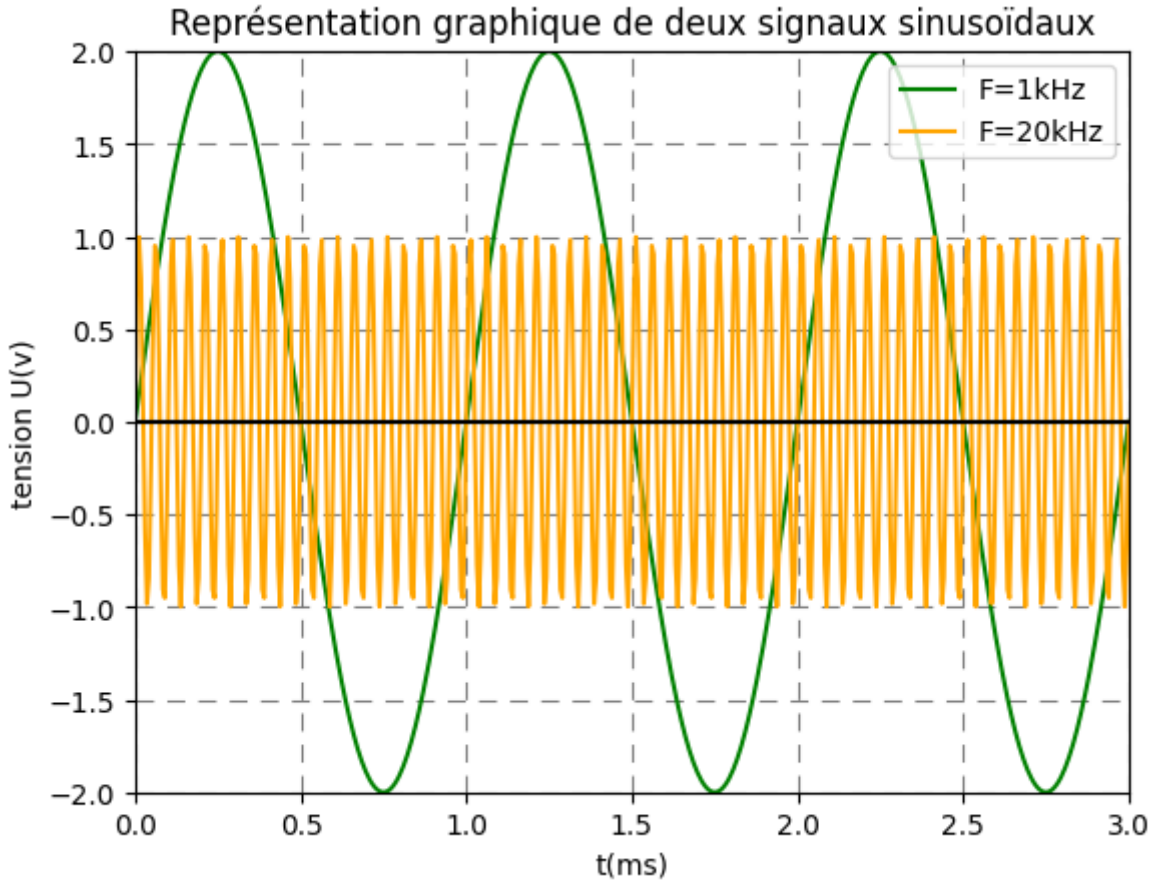


5°) ***Exercice 5***

Ecrire un programme python qui affiche un signal sinusoïdal de fréquence 20 kHz, d'amplitude 2 V et de phase 0°, et un signal de 20 Hz, d'amplitude 1 V et de phase 0°, sur le même graphique.

```
from matplotlib import pyplot as plt
from math import *

# CARACTÉRISTIQUES :
tmax = 0.003  # durée du signal (sur l'axe des abscisses). Modifiable
T1 = 0.001  # période (s). Modifiable
T2 = 0.00005  # période (s). Modifiable
# pulsation. ω = rapport du "tour complet" (2*pi) par la durée nécessaire pour le
parcourir = "vitesse angulaire" (rad/s)
ω1 = 2*pi/T1
ω2= 2*pi/T2
φ = 0  # phase à l'origine (rad). Modifiable
U1max = 2  # amplitude. Modifiable
U2max = 1  # amplitude. Modifiable

repere = plt.axes(xlim=(0, tmax), ylim=(-2, 2))
repere.set_xlabel('t(ms)')
```

```
repere.set_ylabel('tension U(v)')
repere.set_title("Représentation graphique de deux signaux sinusoïdaux")

# nombre d'intervalles (n+1 points). Modifiable (augmenter si la courbe n'est pas
assez 'lisse')
n = 500

# liste des n+1 valeurs de t [0,n] (axe des abscisses)
t = [k*tmax/n for k in range(n+1)]
# calcul des n+1 valeurs de U (ordonnées). Une tension, mais cela pourrait être
une intensité, une élongation...
U1 = [U1max*sin(ω1*t+φ) for t in t]
U2 = [U2max*sin(ω2*t+φ) for t in t]

repere.plot(t, U1, label='F=1kHz', color='green')
repere.plot(t, U2, label='F=20kHz', color='orange')
repere.legend()
repere.grid(color='black', linestyle='--', dashes=[8, 6], alpha=0.5)
repere.axhline(0, color='black')
plt.xticks(np.linspace(0, 0.003, 7), np.linspace(0, 3, 7))
plt.grid(True)
plt.show()
```



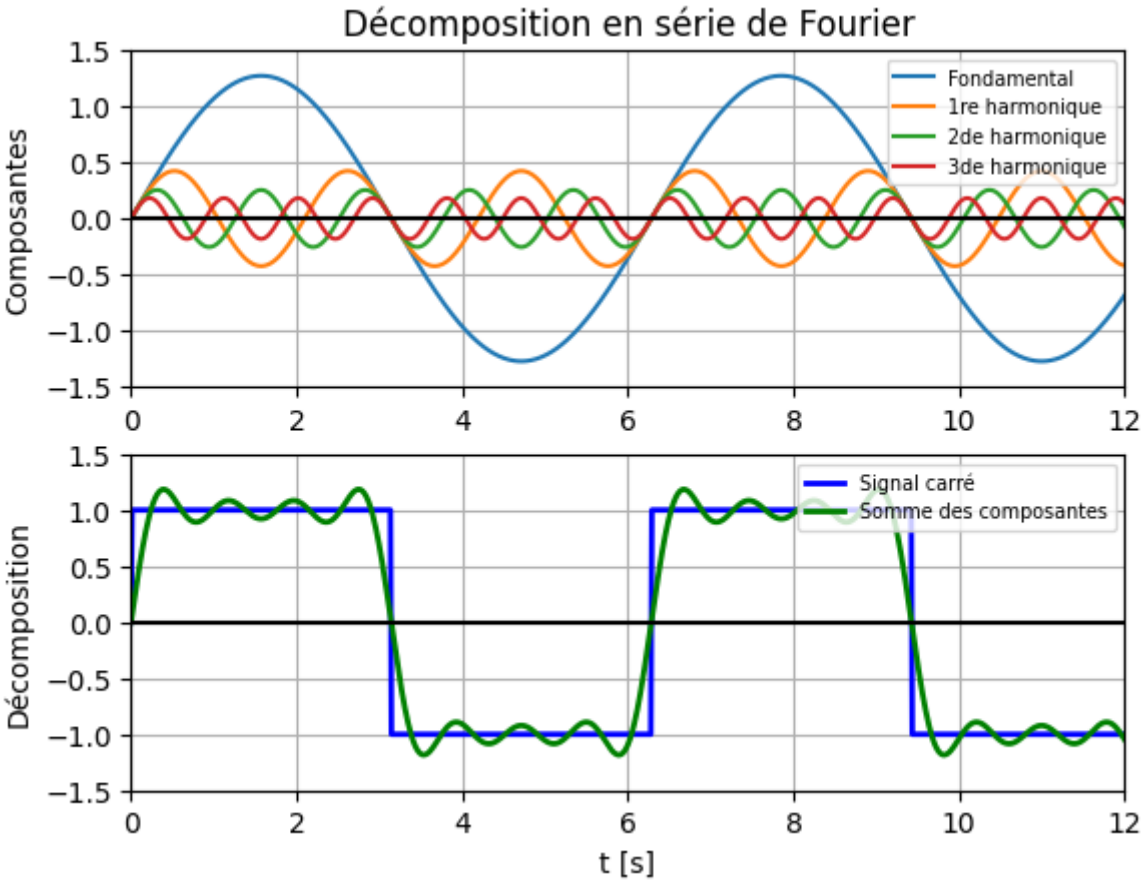
B. Les sommes de Fourier

1°) Exercice 1

On a vu qu'avec une somme de sinusoïdes, on peut reconstruire n'importe quel signal. Représenter un signal carré de 1 kHz d'amplitude 0 - 10V, puis essayer de reconstruire ce signal à partir des premières harmoniques.

```
"""
Exemple un peu plus complexe de figure, incluant 2 axes, légendes, axes, etc.
"""
```

```
import numpy as np
import matplotlib.pyplot as P


# Définition du signal carré
f = 1000  # fréquence en Hz
T = 1/f  # période en s
t = np.linspace(0, 2*T, 1000)  # échantillonnage de 2 périodes
x = 2*np.pi*f*t

# Signal carré
y = np.sign(np.sin(x))            # = ± 1

# 3 premiers termes de la décomposition en série de Fourier
y0 = 4/np.pi * np.sin(x)          # Fondamentale
y1 = 4/np.pi * np.sin(3*x) / 3    # 1re harmonique
y3 = 4/np.pi * np.sin(5*x) / 5    # 2de harmonique
y5 = 4/np.pi * np.sin(7*x) / 7    # 2de harmonique
# Somme des 3 premières composantes
ytot = y0 + y1 + y3 + y5

# Figure
fig = P.figure()                # Création de la Figure

# 1er axe: composantes
ax1 = fig.add_subplot(2, 1, 1,  # 1er axe d'une série de 2 × 1
                      ylabel="Composantes",
                      title="Décomposition en série de Fourier")
ax1.plot(x, y0, label="Fondamental")
ax1.plot(x, y1, label=u"1re harmonique")
ax1.plot(x, y3, label=u"2de harmonique")
ax1.plot(x, y5, label=u"3de harmonique")
ax1.set_xlim([0, 12])
ax1.set_ylim([-1.5, 1.5])
ax1.axhline(0, color='black')
ax1.axvline(0, color='black')
ax1.grid()
ax1.legend(loc="upper right", fontsize="x-small")

# 2nd axe: décomposition
ax2 = fig.add_subplot(2, 1, 2,  # 2d axe d'une série de 2 × 1
                      ylabel=u"Décomposition",
                      xlabel="t [s]")
ax2.plot(x, y, lw=2, color='b', label="Signal carré")
ax2.plot(x, ytot, lw=2, color='g', label="Somme des composantes")
ax2.legend(loc="upper right", fontsize="x-small")
ax2.axhline(0, color='black')
ax2.axvline(0, color='black')
ax2.set_xlim([0, 12])
ax2.set_ylim([-1.5, 1.5])
ax2.grid()
# Sauvegarde de la figure (pas d'affichage intéractif)
fig.savefig("figure.png")
```



2°) ***Exercice 2***

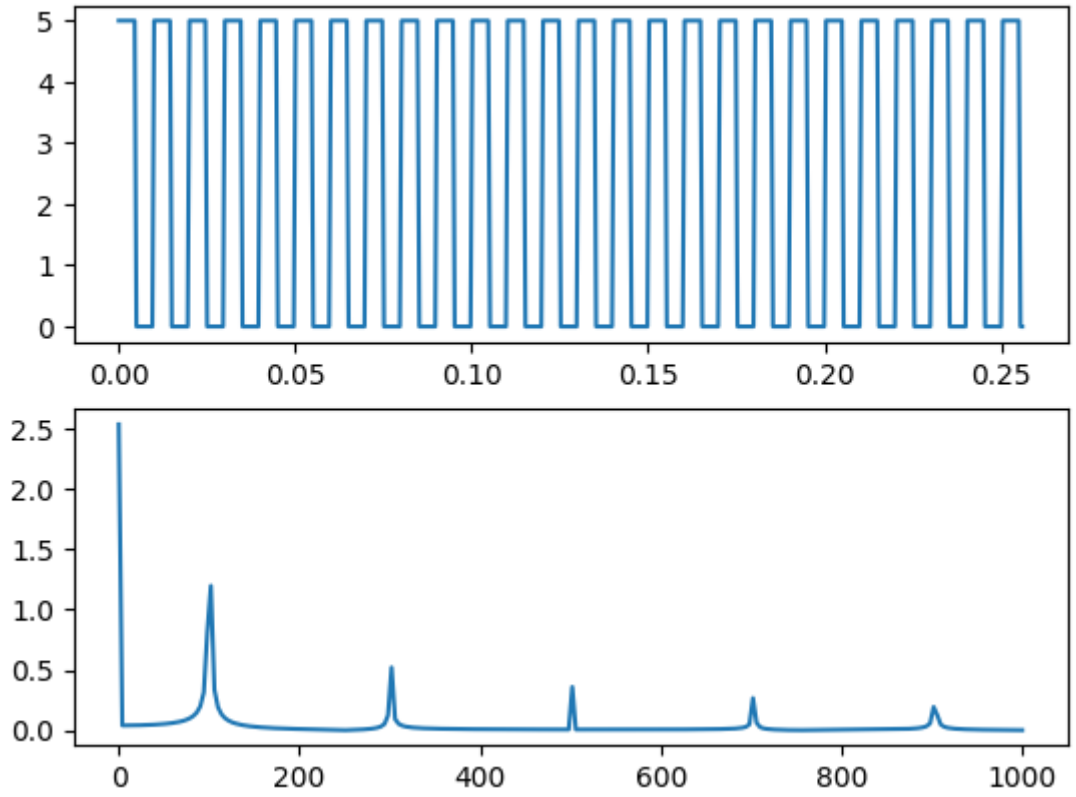
Faire la FFT d'un signal carré de fréquence 100 Hz, amplitude 0-5V, rapport cyclique 50%.

```
from scipy.fftpack import fft
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np

#
# configuration
# time analyse = L * (1/Fsample)
#
L = 512  # lenght buffer
Fsample = 2000  # frequency sample
Fsignal = 100  # frequency

# ******************************

# perdiode sample
Tsample = 1.0/Fsample

# time vector, start = 0s, stop = 0.1024s, step = (0.1024s / (1/Fe))
t = np.linspace(0.0, L*Tsample, L)

# signal definition, DC offet = 2.5V, Amplitude = 2.5V
signal = 2.5 + 2.5*signal.square(2 * np.pi * Fsignal * t, 0.5)

# plot time signal
plt.subplot(2, 1, 1)
plt.plot(t, signal)

# fft of time signal
yf = fft(signal)

# time vector of fft
```

```
xf = np.linspace(0.0, 1.0/(2.0*Tsample), L//2)

# plot spectral
plt.subplot(2, 1, 2)
plt.plot(xf, 1.0/L * np.abs(yf[0:L//2]))
```

ou

```
from scipy.fftpack import fft
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np

# To reset margins to 0 for all plots in a script
# plt.rcParams['axes.autolimit_mode'] = 'round_numbers'
# plt.rcParams['axes.xmargin'] = 0.
# plt.rcParams['axes.ymargin'] = 0.

#
# configuration
# time analyse = L * (1/Fsample)
#
L = 512  # lenght buffer
Fsample = 2000  # frequency sample
Fsignal = 100  # frequency

# ********************************

# perdiode sample
Tsample = 1.0/Fsample

# time vector, start = 0s, stop = 0.1024s, step = (0.1024s / (1/Fe))
t = np.linspace(0.0, L*Tsample, L)

# signal definition, DC offet = 2.5V, Amplitude = 2.5V
signal = 2.5 + 2.5*signal.square(2 * np.pi * Fsignal * t, 0.5)
plt.axes(xlim=(0, Fsample), ylim=(0, 6))


# plot time signal
plt.subplot(2, 1, 1)
plt.plot(t, signal)

# fft of time signal
yf = fft(signal)

# time vector of fft
xf = np.linspace(0.0, 1.0/(2.0*Tsample), L//2)
# To reset margins to 0 for a single Axes instance

# plot spectral
plt.subplot(2, 1, 2)
plt.plot(xf, 1.0/L * np.abs(yf[0:L//2]))
```



C. Les bases de lecture écriture de fichier

1°) Exercice 1

Ecrire un programme python qui lit un fichier wav et qui le joue sur le PC.

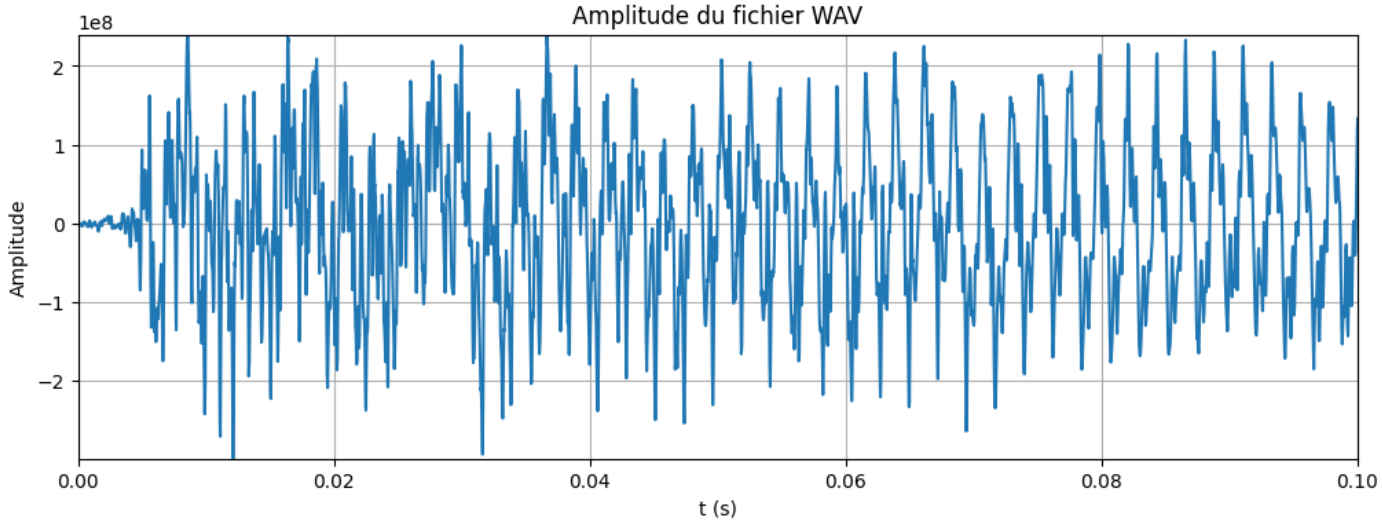
- http://prof-tc.fr/Lycee/file/Terminale%20S/Specifique/Fichiers/P1C2/Piano%20La3.wav

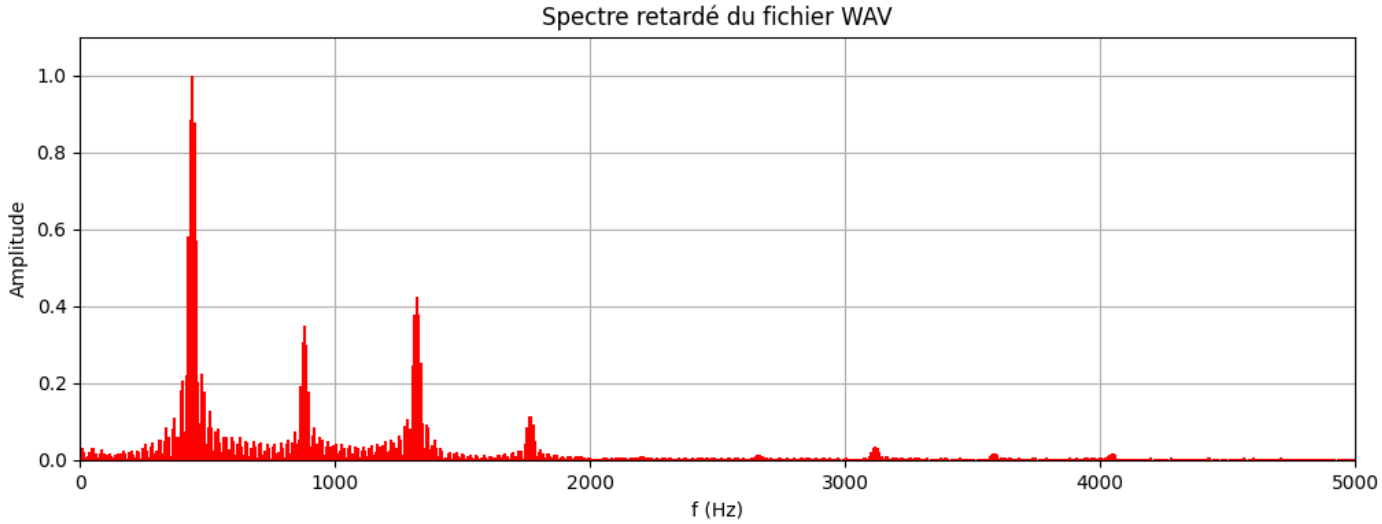
```
# importation des modules:
import scipy.io.wavfile as wave
from playsound import playsound

# Lecture du fichier WAV:
rate, data = wave.read('../Audios/La3-piano.wav')
n = data.size
duree = 1.0*n/rate

# Affichage de la Fréquence d'échantillonnage (en Hz) et durée de l'enregistrement:
print("La fréquence d'échantillonnage est de ", rate)
print("La durée de l'enregistrement est de ", duree)
playsound('../Audios/La3-piano.wav')
```

2°) Exercice 2

Ecrire un programme python qui lit un fichier wav et qui affiche son tracé en fonction du temps et qui affiche son spectre.

- http://prof-tc.fr/Lycee/file/Terminale%20S/Specifique/Fichiers/P1C2/Piano%20La3.wav






```
# importation des modules:
from numpy.fft import fft
import scipy.io.wavfile as wave
from matplotlib.pyplot import *
import numpy as np
import math

# Lecture du fichier WAV:
rate, data = wave.read('audio/piano_la3.wav')
n = data.size
duree = 1.0*n/rate

# Affichage de la Fréquence d'échantillonnage (en Hz) et durée de l'enregistrement:
print("La fréquence d'échantillonnage est de ",rate)
print("La durée de l'enregistrement est de ", duree)

# Tracé du signal:
te = 1.0/rate
t = np.arange(n)*te
figure(figsize=(12, 4))
plot(t, data)
xlabel("t (s)")
ylabel("amplitude")
```

```
axis([0, 0.1, data.min(), data.max()])
grid()

# Calcul du spectre avec la transformée de Fourier discrète, pour une fenêtre dont
on précise le début et la durée(en secondes):
def tracerSpectre(data, rate, debut, duree):
    start = int(debut*rate)
    stop = int((debut+duree)*rate)
    spectre = np.absolute(fft(data[start:stop]))
    spectre = spectre/spectre.max()
    n = spectre.size
    freq = np.arange(n)*1.0/n*rate
    vlines(freq, [0], spectre, 'r')
    xlabel('f (Hz)')
    ylabel('A')
    axis([0, 0.5*rate, 0, 1])
    grid()


figure(figsize=(12, 4))
tracerSpectre(data, rate, 0.0, 0.5)
axis([0, 5000, 0, 1.1])

#Pour une fenêtre de même durée débutant plus tard:

figure(figsize=(12, 4))
tracerSpectre(data, rate, 1.0, 0.5)
axis([0, 5000, 0, 1.1])
```

3°) Aller plus loin

1. https://fr.wikipedia.org/wiki/RDS_(radio)
2. https://www.sigidwiki.com/wiki/Radio_Data_System_(RDS) (anglais)
3. https://github.com/bastibl/gr-rds
4. https://www.gnuradio.org/

← 14. Synchronisation 16. A propos de l'auteur →

D. Webographie :

https://cpge.frama.io/fiches-cpge/Python/Analyse%20fr%C3%A9quentielle%20%28FFT%29/01%20-%20Transform%C3%A9e%20de%20Fourier%201D/

https://cpge.frama.io/fiches-cpge/Python/Filtres/80%20-%20Filtrage/

https://cpge.frama.io/fiches-cpge/Python/Graphiques/5-3D/

https://www.geeksforgeeks.org/displaying-3d-images-in-python/

https://terbium.io/2017/12/matplotlib-3d/

https://pysdr.org/fr/content-fr/rds.html

https://python.developpez.com/tutoriels/graphique-2d/matplotlib/#LII-D

https://www.codingame.com/playgrounds/53303/apprendre-python-dans-le-secondaire/complements-sur-le-module-matplotlib

http://www.python-simple.com/python-matplotlib/configuration-axes.php

http://numerique.ostralo.net/python_matplotlib/2_ElementsGeneraux.htm

http://culturesciencesphysique.ens-lyon.fr/ressource/numerisation-acoustique-Chareyron2.xml